Computational Mirror Cup and Saucer Art

KANG WU, RENJIE CHEN, XIAO-MING FU, and LIGANG LIU, University of Science and Technology of China, China

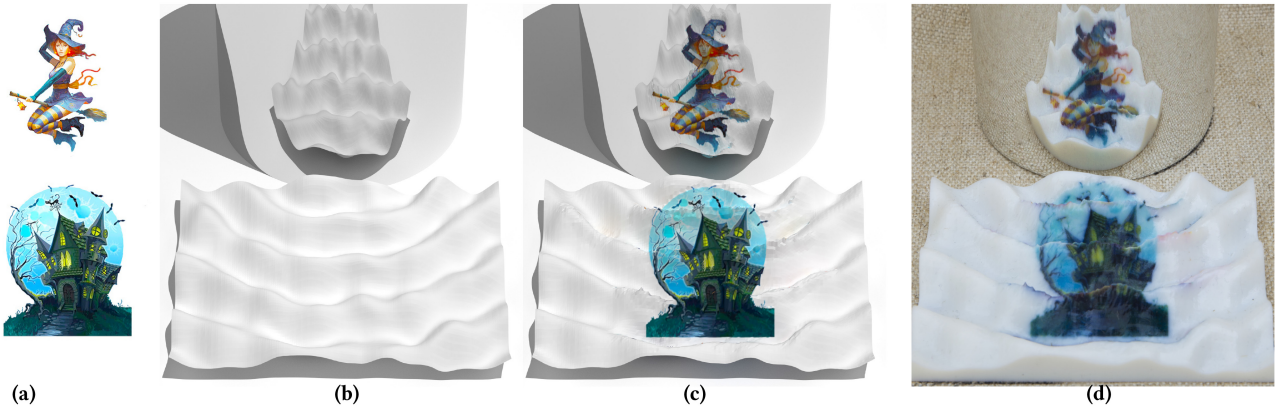


Fig. 1. Our system takes two completely different images as inputs: (a) a reference saucer shape (bottom image in (b)), and a mirror (upper image in (b)). A textured saucer is automatically generated by slightly deforming the input saucer so that the direct view of the saucer and the reflected view through the mirror match the input images (c). To validate our system, we use 3D printing technology to fabricate the resulting saucer (d).

In the mirror cup and saucer art created by artists Yul Cho and Sang-Ha Cho, part of the saucer is directly visible to the viewer, while the other part of the saucer is occluded and can only be seen as a reflection through a mirror cup. Thus, viewers see an image directly on the saucer and another image on the mirror cup; however, the existing art design is limited to wave-like saucers. In this work, we propose a general computational framework for mirror cup and saucer art design. As input, we take from the user one image for the direct view, one image for the reflected view, and the base shape of the saucer. Our algorithm then generates a suitable saucer shape by deforming the input shape. We formulate this problem as a constrained optimization for the saucer surface. Our framework solves for the fine geometry details on the base shape along with its texture, such that when a mirror cup is placed on the saucer, the user-specified images are observed as direct and reflected views. Through extensive experiments, we demonstrate the effectiveness of our framework and the great design flexibility that it offers to users. We further validate the produced art pieces by fabricating the colored saucers using three-dimensional printing.

CCS Concepts: • **Computing methodologies** → **Shape modeling;**

Additional Key Words and Phrases: Mirror cup and saucer art, differentiable rendering, black-white shape enhancement, sparse spike movement

ACM Reference format:
Kang Wu, Renjie Chen, Xiao-Ming Fu, and Ligang Liu. 2022. Computational Mirror Cup and Saucer Art. *ACM Trans. Graph.* 41, 5, Article 174 (July 2022), 15 pages.
https://doi.org/10.1145/3517120

This research is supported by the National Key R&D Program of China (2020YFC1523102), the National Natural Science Foundation of China (62072422, 62025207), and the Anhui Provincial Natural Science Foundation (2008085MF195).
Authors' addresses: K. Wu, R. Chen (corresponding author), X.-M. Fu, and L. Liu, School of Mathematics, University of Science and Technology of China, Hefei, Anhui, China; emails: kang910042009@gmail.com, {renjiec, fuxm, lgliu}@ustc.edu.cn.

0730-0301/2022/07-ART174 $15.00
https://doi.org/10.1145/3517120

1 INTRODUCTION

Anamorphic art [Baltrusaitis 1977] refers to an image or drawing that is distorted so that it can only be recognized when viewed in a particular way or using a special device. In mirror anamorphic art, the viewer typically sees an object both directly and indirectly through a mirror. Generally, the direct view, i.e., what the viewer sees directly, is different from the reflected view, i.e., what the viewer sees via a mirror. There are three factors in mirror anamorphosis, including the mirror, the viewer, and the locus where the distorted object lays, which is referred to as the *surface of distortion* [De Comite 2011].

Recently, artists Yul Cho and Sang-Ha Cho designed a variant of mirror cup and saucer art (Figure 2). In their designs, the surface of distortion is a wavy saucer, wherein some parts of the surface are hidden in the direct view due to occlusion by other parts. Hence, this art form is dubbed, "mirror cup and saucer." This art is appealing as the two views are in harmony, rendering it a more integrated art piece. Furthermore, the surface of distortion can be a more general shape instead of repeating patterns like waves, making this art more customizable.

However, it takes tremendous effort and is a tedious and error-prone process for the artists to manually design and carefully tune the saucer surface to distinguish the perceived images in both the

Fig. 2. Artists use wave-shaped saucers to generate mirrored cups where the pattern on the saucer is still meaningful by itself. More examples can be found at http://luycho.com/.

direct and reflected views. One general solution is to separate the reflected area from the directly viewed area by using a corrugated saucer; however, this implies that the saucer has a fixed *corrugated* shape. Suppose one would like to specify the two images and the shape of the saucer simultaneously. In this case, it is nearly impossible to find a solution, since not all saucer shapes can adequately resolve the conflict between the reflected and directly viewed areas. Thus, tailoring a new shape for each set of input images is not an easy task, making it highly challenging to produce mirror cup and saucer art with desired images and saucer surfaces.

Given two input images and a saucer shape, our goal is to generate a textured saucer mesh so that the reflected and direct views match the two input images and the mesh surface is close to the input saucer shape. To achieve our goal, we formulate a constrained optimization problem, where the variables represent the positions of mesh vertices and the texture colors are defined on the mesh. The objective is to minimize the visual differences between input images and images rendered with the specified camera position, while simultaneously reducing the deviation between the input surface and the deformed surface. Furthermore, we add the constraint that the shape deviation from the input surface to the deformed surface is bounded from above.

To solve our optimization problem, it is intuitively straightforward to resort to differentiable rendering technology, e.g., Liu et al. [2019], Zhang et al. [2020], Li et al. [2018], and Zhang et al. [2019], in place of the standard rendering pipeline for the direct and reflected images. Consequently, differentiable rendering technology allows us to obtain the gradients of the rendered image and enables the use of conventional numerical optimization algorithms. Nevertheless, there are still two challenges in achieving low visual differences as well as small deformation distortions. First, a standard ray tracer is needed to generate the reflected image, which is what the viewer sees through the mirror cup; however, applying differentiable ray tracing rendering technology to our problem is more computationally expensive compared to differentiable rasterization. Second, it is non-trivial to obtain a favorable tradeoff between low visual differences and small surface deformation distortions. Generally, optimizing a surface to be spiky or corrugated leads to low visual differences but large deformation distortions (Figure 3(b)).

In this article, we present a novel computational framework for mirror cup and saucer art designs. We solve the constrained optimization problem using gradient-based methods to achieve

(a) Inputs **(b)** W/o shape distortion **(c)** Ours

Fig. 3. (a) The input images (upper) and surface (bottom). (b) Optimized visual differences without considering shape distortion. (c) Our result. In (b) and (c), the upper image shows the saucer and cup rendered from the viewer's perspective, while the bottom shows a close-up bird's-eye view of the saucer.

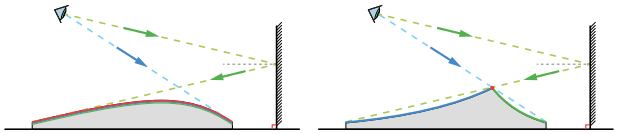


Fig. 4. Left: The conflict region (red) contributes to both $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$. Right: A spike (red dot) separates the conflict area into two parts, each of which contributes to one rendered image.

low visual difference while maintaining low shape distortion. Our success relies on two key techniques. First, we develop a new method to directly compute an image from a saucer surface in a mirrored cup, which enables us to render reflected images highly efficiently via rasterization. The second is a conflict-reduction strategy for achieving a desirable tradeoff between low visual differences and small shape deformations. We observe that maintaining low shape deformation usually leads to a large conflict area, which contributes to both the reflected and directly viewed images, thereby preventing visual differences from being optimized to a low level (Figures 4 and 7). To reduce this conflict while retaining low deformation distortion, we develop a black-white enhancement technique and a sparse spike strategy.

We apply the proposed computational tools for mirror cup and saucer art design and construction with various images and shapes as inputs. Extensive experiments and fabricated results by three-dimensional (3D) printing demonstrate the feasibility and practicability of our proposed technique.

2 RELATED WORK

2.1 Anamorphic Art

Anamorphosis is a type of visual art that utilizes the perspective technique to produce art pieces that appear normal from a particular angle but distorted from other angles. Artists typically use distorted lenses, mirrors, or other optical devices to devise complex anamorphoses. The earliest known example of this technique is *Leonardo's Eye* by Leonardo da Vinci, included in the *Codex*

Atlanticus. Then, it appeared mostly in 16th- and 17th-century drawing manuals, where it was utilized in various ways. In the late 16th century, a new kind of anamorphosis became popular in England, wherein two different images are divided into strips and painted onto different sides of a corrugated carrier. Both images can be seen properly when viewed from a certain angle or through a mirror. An illustration of this basic technique was first published in a 1583 book on perspective drawing by Giacomo Barozzi da Vignola and lgnazio Danti. In 1646, Athanasius Kircher invented the latin phrase "Tabula scalata" to describe this kind of artwork. Since then, these techniques have become increasingly popular for artists to produce.

In modern times, many scholars have studied this art form. Sánchez-Reyes and Chacón [2016, 2020] and Schüller et al. [2014] proposed their own methods for generating anamorphic art, which can only be viewed properly from a certain viewing position. For anamorphic images that need to be viewed through a parametric mirror surface (e.g., cylinder), Hunt et al. [2000] and Rausch et al. [2012] explored the connections between mathematical and artistic formulations of anamorphosis. The problem they study is simpler than ours, as the meaning of the directly viewed image is not taken into account. In this manner, they draw patterns directly based on the light path. Instead of creating anamorphic images, Čučaković and Paunović [2015] and De Comite [2011] produced anamorphic art by using three-dimensional objects; however, they also did not consider the integrity of the directly viewed object itself, and their problem can also be solved by following the light path.

Recently, computer graphics techniques have been applied in this problem. Hansford and Collins [2007] created anamorphic 3D digital models, providing a tool for artists and architects. De Comite and Grisoni [2015] proposed a deformation technique for both surfaces and 3D shapes to generate catoptric anamorphosis using raycasting. They back-project a target image onto the surface of an object to generate anamorphic art. We solve a similar problem in Section 3.4.2. Given a finished design piece, Stojakovic and Tepavcevic [2016] identified and analyzed several problems that cause distortion in Plane Geometric Anamorphosis due to the dislocation of the viewpoint, and they provide suggested image designs that are less prone to viewpoint changes. Jovanovic et al. [2016] studied how to use an industrial robot to automatically generate anamorphic structures on a curved wall by utilizing modular bricklike elements. In this article, we study the computational model for designing a special type of anamorphic art, called mirror cup and saucer art, which to our knowledge, has never been investigated in the graphics community.

2.2 Computational Optics Art

For the past couple of decades, computer graphics technologies have been successfully applied to the automatic design of various types of optical art, including caustics [Papas et al. 2011; Schwartzburg et al. 2014], reflectance [Weyrich et al. 2009], shadow [Mitra and Pauly 2009; Zhao et al. 2016], and so on. What's more, graphics-based approaches can be used for designing a wide variety of optical art in which changes in viewpoint [Oliva et al. 2006], figure-ground organization [Kuo et al. 2016], illumination from a certain direction[Alexa and Matusik 2010; Bermano et al. 2012], viewing directions [Keiren et al. 2009; Pjanic and Hersch 2015; Sakurai et al. 2018; Sela and Elber 2007; Snelgrove et al. 2013], or casting shadows onto external planar surfaces [Hsiao et al. 2018; Min et al. 2017] can change the viewer's perception of an art piece. This work is close to Mitra and Pauly [2009] and Hsiao et al. [2018], where 3D volumes are computed such that their shadows best approximate multiple input binary images. This work also relates to Sakurai et al. [2018]. In both works, the surface of an object is divided into sets of areas, with each set showing a separate image that can be appreciated from a specific viewing setup. The first main difference in this work is that one of the images needs to be observed via a curved mirror, and the viewing position is the same for both images, while for Sakurai et al. the viewing position w.r.t. the object changes between the images. The second difference is that to generate the aforementioned area sets for multiple images, we deform the input surface, while Sakurai et al. use microstructural stripe patterns.

2.3 Differentiable Rendering

The rendering pipeline can be seen as a function that turns input 3D objects into rendered images; hence given the derivatives of the rendering pipeline, we relate the changes in the 3D objects with those in the rendered images. To obtain the derivatives, Loper and Black [2014] proposed an approximate differentiable renderer that provides derivatives of pixel colors w.r.t. the 3D geometry parameters. Kato et al. [2018] introduced a function to approximate the backward gradient of rasterization to achieve differentiable rendering. Liu et al. [2019] and Chen et al. [2019] replaced the non-differentiable part of the standard rasterizer with smooth functions. Li et al. [2018] and Zhang et al. [2019] proposed differentiable ray tracers to make the rendering process differentiable. For some 3D reconstruction tasks, such as 3D face reconstruction [Genova et al. 2018; Richardson et al. 2017; Tewari et al. 2018, 2017; Tran and Liu 2018], material inference [Deschaintre et al. 2018; Liu et al. 2017], and others [Henderson and Ferrari 2018; Jimenez Rezende et al. 2016; Kundu et al. 2018; Zienkiewicz et al. 2016], differentiable rendering layers were specially designed to obtain gradient flow from image to vertex positions, materials, or other parameters using application-tailored neural networks.

Usually, raytracing is needed to render scenes with complex lighting, such as reflections, and differentiable raytracing is used to obtain the derivatives of the rendering parameters. However, differentiable ray tracers generally require excessive computational resources, including CPU runtime and memory consumption. Nimier-David et al. [2019] proposed a versatile differentiable renderer, *Mitsuba 2*, which has significantly reduced the required computational resources. However, while conducting this research, we found that it was still challenging to apply it to our problem setup. To further lower the requirements and make our problem solvable, we chose to generate a reflected shape of the saucer and use differential rasterization, in particular, **Soft Rasterizer (SoftRas)** [Liu et al. 2019] to render the reflected image. SoftRas replaces the non-differentiable part of the standard rasterization pipeline with a differentiable one; thus, we directly differentiate the rendered images and obtain derivatives of the 3D geometric parameters.

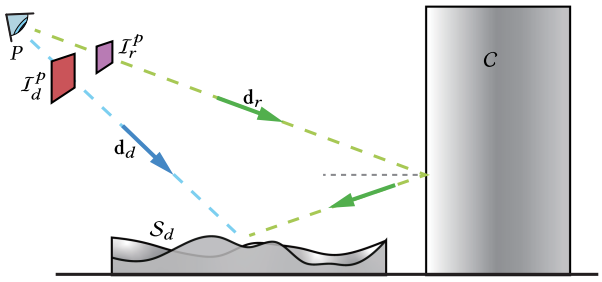


Fig. 5. Input illustrations. The viewer at P sees an image $\mathcal{I}_d^p$ directly on the saucer $\mathcal{S}_d$ along the direction $\mathbf{d}_d$ and another image $\mathcal{I}_r^p$ through mirror cup C from the direction $\mathbf{d}_r$.

3 METHOD

3.1 Problem and Formulation

Problem. We denote the two input images as $\mathcal{I}_d$ and $\mathcal{I}_r$, and the input saucer shape as $\mathcal{S}$, represented by a heightfield. The output is a textured surface $\mathcal{S}_d$. From the specified viewing position P and directions, the user sees an image $\mathcal{I}_d^p$ on the surface $\mathcal{S}_d$ in the direction $\mathbf{d}_d$ and another image $\mathcal{I}_r^p$ on the mirror cup C reflecting the textured surface $\mathcal{S}_d$ in the direction $\mathbf{d}_r$ (Figure 5). The target surface $\mathcal{S}_d$ should satisfy the following two requirements:

- *Visual fidelity*: the visual difference from $\mathcal{I}_d^p$ (or $\mathcal{I}_r^p$) to $\mathcal{I}_d$ (or $\mathcal{I}_r$) is small.
- *Shape preservation*: the distortion of the deformation from $\mathcal{S}$ to $\mathcal{S}_d$ is low.

Without loss of generality, we assume that the input mirror cup C is a cylindrical surface, and $\mathcal{S}$ and $\mathcal{S}_d$ are triangulated heightfields. For each vertex $\mathbf{v}_i$ in $\mathcal{S}$, we associate its 2D position $(x_i, y_i) \in \mathbb{R}^2$ with its height value $z_i \in \mathbb{R}$, and we denote the height of its corresponding vertex in $\mathcal{S}_d$ as $h_i \in \mathbb{R}$. We denote $\mathbf{h} = \{h_i\}$ as the unknown height vector.

Generating $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$. To build a computational tool that satisfies the aforementioned two requirements, we adopt a differentiable renderer to render the textured surface $\mathcal{S}_d$ to generate $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$.

To make the rendering processing more efficient, we represent $\mathcal{S}_d$ as a triangulated heightfield with colored triangle facets instead of using texture mapping. We denote the color vectors for all triangle facets as $\mathbf{c} = \{(r_i, g_i, b_i)\}$. $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$ are functions of $\mathbf{h}$ and $\mathbf{c}$, denoted as $\mathcal{I}_d^p(\mathbf{h}, \mathbf{c})$ and $\mathcal{I}_r^p(\mathbf{h}, \mathbf{c})$.

Naïve formulation. To find the $\mathcal{S}_d$ value that satisfies both of the visual fidelity and the shape preservation requirements, we formulate a constrained optimization problem as follows:

$$\begin{aligned} \min_{\mathbf{h},\mathbf{c}} \quad & E_{\text{visual}}(\mathbf{h}, \mathbf{c}) + wE_{\text{deform}}(\mathbf{h}) \\ \text{s.t.} \quad & |z_i - h_i| < \delta, \forall i, \\ & 0 \le r_j, g_j, b_j \le 1, \forall j, \end{aligned} \tag{1}$$

where $E_{\text{visual}}(\mathbf{h}, \mathbf{c})$ measures the color difference between the input and the rendered images. $E_{\text{deform}}(\mathbf{h})$ denotes the deformation distortion, and w is a positive weight. In the inequality constraints, the movement of each vertex is bounded to be less than a given threshold δ, while the color is constrained within the range $[0, 1]$.

Visual fidelity term. We define $E_{\text{visual}}(\mathbf{h}, \mathbf{c})$ as the sum of the squared Frobenius norm of the color difference between the input images and the rendered images:

$$E_{\text{visual}}(\mathbf{h}, \mathbf{c}) = \|\mathcal{I}_d - \mathcal{I}_d^p(\mathbf{h}, \mathbf{c})\|_F^2 + \|\mathcal{I}_r - \mathcal{I}_r^p(\mathbf{h}, \mathbf{c})\|_F^2. \tag{2}$$

Shape-preservation term. Sorkine et al. [2004] propose editing surfaces using the Laplacian coordinates, as this captures the geometric details. This technique inspired us to define $E_{\text{deform}}(\mathbf{h})$ as the squared L_2 distance between the Laplacians of the heightfields:

$$E_{\text{deform}}(\mathbf{h}) = \sum_{\mathbf{v}_i} (Lz_i - Lh_i)^2, \tag{3}$$

where L is the discrete Laplace operator.

Reformulation as an unconstrained optimization problem. It is easy to obtain a set of initial $\mathbf{h}$ and $\mathbf{c}$ values that satisfies the hard constraints, e.g., $h_i = z_i, \forall i$ and $r_j = g_j = b_j = 0.5, \forall j$. Hence, we propose using barrier functions to keep variables inside the feasible region. We apply popular log barriers to ensure that variables are positive:

$$\phi(x) = \begin{cases} -\log(x), & \text{if } x > 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{4}$$

Then, our hard constraints are converted into an energy term:

$$\begin{aligned} E_{\text{barrier}}(\mathbf{h}, \mathbf{c}) = & \sum_{\mathbf{v}_i} \left(\phi(\delta - (z_i - h_i)) + \phi(\delta + (z_i - h_i))\right) \\ & + \sum_{\mathbf{f}_j} \sum_{c_j \in \{r_j, g_j, b_j\}} \left(\phi(c_j) + \phi(1 - c_j)\right), \end{aligned} \tag{5}$$

where $\mathbf{f}_j$ indicates the jth triangle facet. Then, Problem (1) is reformulated as an unconstrained optimization problem:

$$\min_{\mathbf{h},\mathbf{c}} \quad E(\mathbf{h}, \mathbf{c}) = E_{\text{visual}}(\mathbf{h}, \mathbf{c}) + wE_{\text{deform}}(\mathbf{h}) + E_{\text{barrier}}(\mathbf{h}, \mathbf{c}). \tag{6}$$

Log barriers prevent color values from being exactly equal to 0 or 1. Fortunately, we resolve this issue using the texturing technique presented in Section 3.4.

Challenges. With differentiable rendering technology, we easily obtain gradients of $E(\mathbf{h}, \mathbf{c})$ w.r.t. $\mathbf{h}$ and $\mathbf{c}$. Hence, gradient-based numerical optimizers can be applied to $E(\mathbf{h}, \mathbf{c})$. However, two challenges remain. First, the reflected image $\mathcal{I}_r^p$ typically needs to be rendered with ray tracing; however, a differentiable ray tracer is computationally expensive and prohibitive. Second, the visual fidelity requirement and the shape-preserving requirement are generally mutually exclusive. In fact, we achieve high visual fidelity by simply ignoring deformation distortion (Figure 3). Thus, it is challenging to achieve a good tradeoff between these two requirements.

3.2 Reflected Shapes

Key idea. Instead of using ray tracing to obtain the reflected image $\mathcal{I}_r^p$, we first compute the reflected shape $\mathcal{S}_r$ of $\mathcal{S}_d$, i.e., the shape formed as a reflection of $\mathcal{S}_d$ in the mirror cup C. Then, $\mathcal{S}_r$ is rendered into $\mathcal{I}_r^p$ using the rasterization technique. Note that straight line segments become curves after reflection. However,

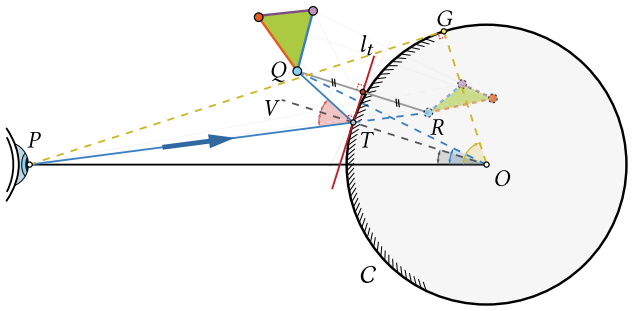


Fig. 6. Given a point Q outside the cylindrical reflector C, the viewer at P sees its reflection R at point $T \in C$. For each point Q, the position of point R can be found by solving a planar geometry problem.

with the high-resolution meshes that we use, we can approximate each relfected mesh edge as a straight line segment. Therefore, we only need to compute the reflected positions for vertices.

Problem conversion. Since the mirror cup C is assumed to be a cylindrical surface, each point Q in 3D must have the same height as its reflection R in C. Hence, we reformulate this problem as a 2D geometry problem: Given a camera position P, a point Q, and a circle C with radius r in the plane, we want to find the reflection of Q, denoted as R (Figure 6).

Eberly [2008] derived an analytic solution to this problem by equivalently solving for T, which is the point on C where the ray from P reflects and then hits Q. Assuming points P, Q, and O are not colinear, then T can be expressed as $T = xP + yQ$, with x and y being coefficients to be solved for. Let Q' be the point symmetric to Q w.r.t. the normal at T, which should fall on the line of OP; therefore,

$$(P - T) \times (Q' - T) = 0 = (2\langle Q, T\rangle - 1)y - 1 + x)P \times Q,$$

leads to the following quadratic equation in x and y,

$$2\langle Q, P\rangle xy + 2|Q|^2 y^2 - y - 1 + x = 0.$$

However, since T falls on the unit circle (W.L.O.G.),

$$|T|^2 = 1 = |P|^2 x^2 + 2\langle Q, P\rangle xy + |Q|^2 y^2.$$

Then, x can be found by solving a quartic equation, and y follows.

One problem with the above analytic solution is that it becomes numerically unstable when OP and OQ are close to being parallel. In the following, we propose a simple alternative method that trivially generalizes to other types of mirrored surfaces as long as they are convex.

If we denote the tangent line passing through T as l_t, then R and Q are symmetric w.r.t. l_t. Thus, if we know the angle value of $\angle POT$, then l_t can be obtained and R follows. Hence, we focus on the computation of $\angle POT$ in the following.

Computing $\angle POT$. Note that when $\angle PTV = \angle QTV$, $\angle POT$ is as desired. According to this condition, we seek $\angle POT$ to satisfy $\angle PTV = \angle QTV$. We limit our discussion to the top left quarter circle, as shown in Figure 6.

Proposition 3.1. *When $\angle POT$ increases from 0 to $\min\{\angle POQ, \angle POG\}$, where $G \in C$ is such that $PG \perp OG$, $\angle PTV$ strictly increases while $\angle QTV$ strictly decreases, and $\angle PTV - \angle QTV$ strictly increases from negative values to positive values.*

We provide a proof of this proposition in appendix A. According to Proposition 3.1, we use a binary search to find $\angle POT$ such that $\angle PTV = \angle QTV$.

ALGORITHM 1: Mirror Cup and Saucer Optimization

Input : A triangulated heightfield $\mathcal{S}$ and two images, $\mathcal{I}_d$ and $\mathcal{I}_r$
Output: A textured mesh $\mathcal{S}_d$
Initialize $\mathcal{S}_d \leftarrow \mathcal{S}$ and color each triangle in gray (0.5, 0.5, 0.5);
$\mathcal{S}_d \leftarrow$ `Black-WhiteOptimization`($\mathcal{S}_d$) via solving Equation (8);
$\mathcal{S}_d \leftarrow$ `SparseSpikeStragey`($\mathcal{S}_d$) via solving Equation (10);
$\mathcal{S}_d \leftarrow$ `Texturing`($\mathcal{S}_d$);

3.3 Optimization

With the reflected shape, we render $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$ using the SoftRas framework [Liu et al. 2019], which replaces the non-differentiable steps of the standard rasterizer with smooth functions. This allows us to easily obtain the derivatives of $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$ w.r.t. the variables $\mathbf{h}$ and $\mathbf{c}$. As a feasible initialization, we set $h_i = z_i, \forall i$ and $r_j = g_j = b_j = 0.5, \forall j$. We optimize the objective function of Equation (6) using a gradient descent method. Specifically, we first use the *Adamax* algorithm [Kingma and Ba 2014] to compute a descent direction. Then, a backtracking line search is performed to determine the step size to ensure that the hard constraints are satisfied. We terminate the optimization algorithm when the relative change of the objective energy is less than the threshold (10^{-4} in our experiments). We dub our optimization method *DR solver*.

3.4 Our Algorithm

Conflict areas. We observed that, after performing the aforementioned optimization via DR solver, the rendered images $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$ contain significant visual artifacts compared to the input images (Figure 7(b)). Furthermore, we found that these artifacts are caused by the conflict areas that contribute to both $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$ (Figure 4, left). We analyze this behavior and conjecture that there are two causes for these conflict areas. First, since the objective of Equation (6) is a weighted sum of the visual fidelity term and the shape-preservation term, a small visual difference somewhere is not likely to trigger sufficient deformation in the relevant area and resolve the conflict. Consequently, when pixels in $\mathcal{I}_d$ and $\mathcal{I}_r$ corresponding to the conflict area share similar colors, the visual difference is small and the conflict area remains as a result. Second, the shape-preservation term preserves the shape similarity between the input and output shapes; hence, smooth regions in the input surface are prone to induce conflict areas. These causes of conflict areas indicate that it is difficult to achieve a favorable tradeoff between visual fidelity and shape-preservation.

Two-stage algorithm. To better balance visual fidelity and shape-preservation, we propose a two-stage strategy. In the first stage, we reduce as many conflict areas as possible to mitigate the occurrence of visual artifacts while maintaining low geometric distortion (Section 3.4.1). In the second stage, we perform texturing to match the input images for better visual fidelity

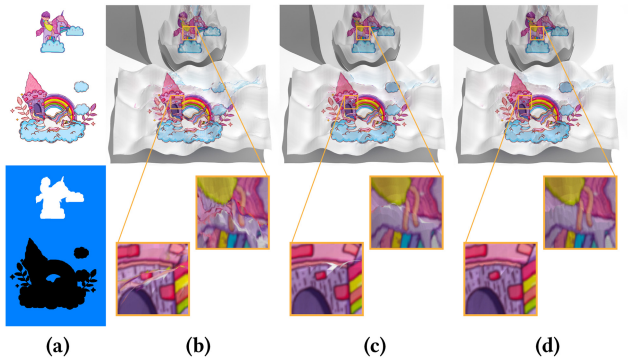


Fig. 7. (a) The input images (upper) and their black-white enhanced images (bottom). (b) The result generated by optimizing Problem (6). (c) The result obtained by using the black-white enhanced images as our target images, i.e., solving Equation (8). (d) After the black-white enhanced optimization, the sparse spike strategy is further applied to improve the result. We run the texturing step in all three methods.

(Section 3.4.2). Figure 8 shows an example, and the pseudocode is shown in Algorithm 1.

3.4.1 Reducing Conflict Areas. We propose two ways to reduce conflict areas based on their cause to eliminate as many visual artifacts as possible.

- *Black-white enhancement*: Enhance the visual fidelity energy to allow more necessary deformation so that conflicts are mitigated (Figure 7(c)). Intuitively, this step tries to separate the saucer into two sets of areas; one set is directly seen by the viewer, while the other is indirectly seen by the viewer via a reflection in the mirror.
- *Sparse spike strategy*: Limit spikes in the surface so that the conflict area is further reduced (Figure 7(d)).

Black-white enhancement. To avoid small visual differences that cause visual artifacts, we replace the input images $\mathcal{I}_d$ and $\mathcal{I}_r$ with images containing contrasting colors (denoted as $\mathcal{I}_d^e$ and $\mathcal{I}_r^e$) as follows:

(1) Detect foreground regions in both $\mathcal{I}_d$ and $\mathcal{I}_r$.
(2) $\mathcal{I}_d^e$ and $\mathcal{I}_r^e$ are obtained by filling the foreground regions of $\mathcal{I}_d$ and $\mathcal{I}_r$ with *black* and *white* accordingly, and the background regions are filled with light blue, i.e., RGB value (0, 0.5, 1), (Figure 8(a), upper right).

The reason for choosing these colors is that black and white have the largest difference in RGB values, which enhances visual differences, in particular magnifying where visual differences are small. For the same reason, we choose to recolor the background in light blue, as it has the largest difference in RGB value from both black and white.

We then define the new visual fidelity term as follows:

$$E_{\text{visual}}^{\text{new}}(\mathbf{h},\mathbf{c}) = |\mathcal{I}_d^e - \mathcal{I}_d^p(\mathbf{h},\mathbf{c})|_F^2 + |\mathcal{I}_r^e - \mathcal{I}_r^p(\mathbf{h},\mathbf{c})|_F^2. \tag{7}$$

Replacing $E_{\text{visual}}(\mathbf{h},\mathbf{c})$ in Equation (6) with $E_{\text{visual}}^{\text{new}}(\mathbf{h},\mathbf{c})$ in Equation (7) leads to a new optimization problem:

$$\min_{\mathbf{h},\mathbf{c}} \quad E_{\text{visual}}^{\text{new}}(\mathbf{h},\mathbf{c}) + wE_{\text{deform}}(\mathbf{h}) + E_{\text{barrier}}(\mathbf{h},\mathbf{c}), \tag{8}$$

which is optimized using DR solver (Figure 7(c)).

Sparse spike strategy. As the right side of Figure 4 shows, spikes on the surface help separate the conflict areas. However, these spikes imply high geometric distortion, so we allow only a few spikes, which are formed by moving as few vertices as possible. We propose the following sparse spike energy term:

$$E_{\text{sparse}}(\mathbf{h}) = \sum_{\mathbf{v}_i} |h_i - h_i^e|, \tag{9}$$

where h_i^e denotes the height of the resulting vertex $\mathbf{v}_i$ after the black-white enhanced optimization step (Equation (8)). This sparse spike energy term keeps the number of moving vertices as small as possible. As a result it tends to encourage spikes rather than general deformations. Then, our optimization objective becomes

$$\widehat{E}(\mathbf{h},\mathbf{c}) = E_{\text{visual}}^{\text{new}}(\mathbf{h},\mathbf{c}) + \lambda E_{\text{sparse}}(\mathbf{h}) + E_{\text{barrier}}(\mathbf{h},\mathbf{c}), \tag{10}$$

where λ is a positive weight.

We optimize $\widehat{E}(\mathbf{h},\mathbf{c})$ iteratively, by combining the Adamax algorithm with the proximal gradient method [Parikh and Boyd 2014]. $\widehat{E}(\mathbf{h},\mathbf{c})$ can be rewritten in the following separated form:

$$\widehat{E}(\mathbf{h},\mathbf{c}) = \varphi(\mathbf{h},\mathbf{c}) + \psi(\mathbf{h}), \tag{11}$$

where

$$\begin{aligned} \varphi(\mathbf{h},\mathbf{c}) &= E_{\text{visual}}^{\text{new}}(\mathbf{h},\mathbf{c}) + E_{\text{barrier}}(\mathbf{h},\mathbf{c}), \\ \psi(\mathbf{h}) &= \lambda E_{\text{sparse}}(\mathbf{h}). \end{aligned} \tag{12}$$

We denote the variables $\mathbf{h}$ and $\mathbf{c}$ at the kth iteration as $\mathbf{h}^k$ and $\mathbf{c}^k$, respectively. Given $\mathbf{h}^{k-1}$ and $\mathbf{c}^{k-1}$, we propose the following procedure for computing $\mathbf{h}^k$ and $\mathbf{c}^k$:

(1) Use Adamax to compute a descent direction for $\varphi(\mathbf{h}^{k-1},\mathbf{c}^{k-1})$, denoted as $(\mathbf{d}_\varphi^h, \mathbf{d}_\varphi^c)$.
(2) Apply the proximal gradient to compute $\mathbf{d}^h$, the descent direction for $\mathbf{h}$:

$$\mathbf{d}^h = \mathbf{h}^{k-1} - \text{prox}_{\alpha\psi}\left(\mathbf{h}^{k-1} - \alpha\mathbf{d}_\varphi^h\right), \tag{13}$$

where $\alpha > 0$ is a step size, and $\text{prox}_{\alpha\psi}$ denotes the proximal mapping of the function ψ:

$$\text{prox}_{\alpha\psi}(\mathbf{x}) = \arg\min_{\mathbf{h}}(\psi(\mathbf{h}) + \frac{1}{2\alpha}\|\mathbf{h}-\mathbf{x}\|_2^2). \tag{14}$$

With $\psi(\mathbf{h}) = \lambda\|\mathbf{h}-\mathbf{h}^e\|_1$, Equation (14) has a closed form solution:

$$(\text{prox}_{\alpha\psi}(\mathbf{x}))_i = \begin{cases} x_i - h_i^e - \lambda\alpha & \text{if } x_i - h_i^e > \lambda\alpha, \\ 0 & \text{if } |x_i - h_i^e| \le \lambda\alpha, \\ x_i - h_i^e + \lambda\alpha & \text{if } x_i - h_i^e < -\lambda\alpha. \end{cases} \tag{15}$$

Throughout our experiments, we set $\alpha = 0.2$.
(3) $\mathbf{d}^h$ and $\mathbf{d}_\varphi^c$ form the descent direction. A backtracking line search uses this direction to determine a suitable step size β to sufficiently decrease the objective $\widehat{E}$, while ensuring that the hard constraints are not violated. Finally, we update $\mathbf{h}^k = \mathbf{h}^{k-1} - \beta\mathbf{d}^h$ and $\mathbf{c}^k = \mathbf{c}^{k-1} - \beta\mathbf{d}_\varphi^c$.

As an initialization for Problem (10), we use $\mathbf{h}$ and $\mathbf{c}$ from the black-white enhanced optimization (Equation (8)). The optimization process terminates as soon as the relative change of $\widehat{E}(\mathbf{h},\mathbf{c})$ is less than 10^{-4}. As Figure 7(d) shows, the visual artifacts are significantly reduced after performing the black-white enhancement and applying the sparse spike strategy.

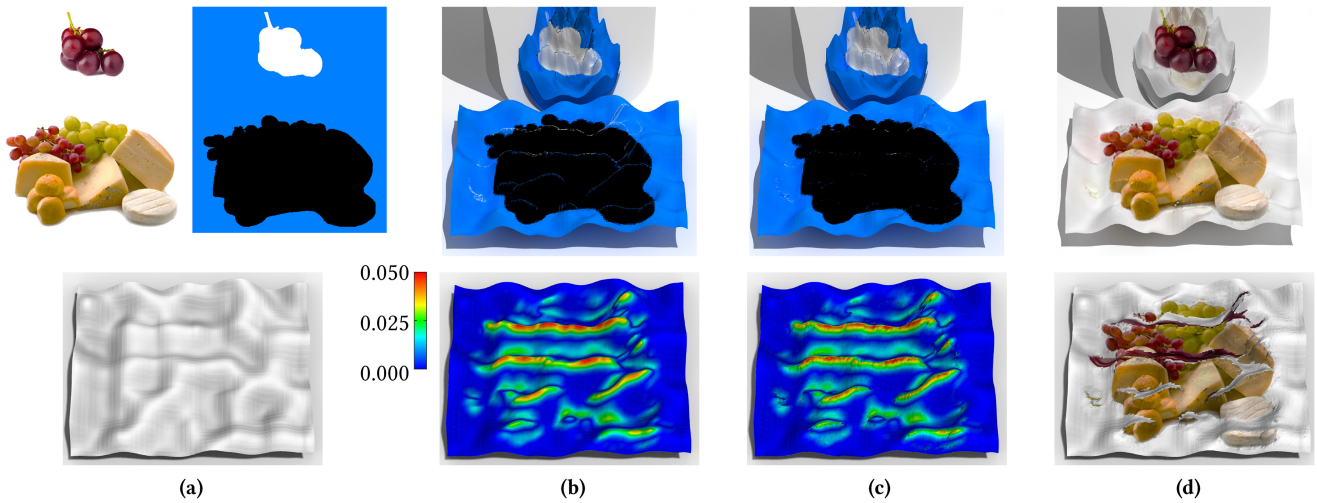


Fig. 8. (a) The input images (top left) and surface (bottom), and the black-white enhanced images (top right). (b) Top: the rendered images after the black-white enhancement. Bottom: color-coded absolute height change relative to the input. (c) The rendered images after applying the sparse spike strategy (top) and the height changes relative to the input (bottom). (d) Top: the final rendered images after texturing. Bottom: the resulting textured surface.

3.4.2 Texturing. We have replaced $\mathcal{I}_d$ and $\mathcal{I}_r$ with $\mathcal{I}_d^e$ and $\mathcal{I}_r^e$ to reduce conflict areas, and as a result, the renderings of the per-face colored heightfield $(\mathbf{h}, \mathbf{c})$ do not match the input images. To produce the target views, we need to apply coloring to the heightfield.

Problem definition. During this step, the surface $\mathcal{S}_d$ is fixed, since the conflict areas have been optimized. In general, we can easily work with texture images in a much higher resolution than that of the heightfield. Therefore, to obtain rich texture details, we represent $\mathcal{S}_d$ as a texture-mapped triangular heightfield, which means we need to specify both the texture coordinates of each vertex and the texture image.

We combine the input images $\mathcal{I}_d$ and $\mathcal{I}_r$ as the texture image. Then, what remains are the texture coordinates of $\mathcal{S}_d$, denoted as $\mathbf{t}$. Note that $\mathcal{I}_d^p$ and $\mathcal{I}_r^p$ are functions of $\mathbf{t}$, denoted as $\mathcal{I}_d^p(\mathbf{t})$ and $\mathcal{I}_r^p(\mathbf{t})$. We define an objective function of $\mathbf{t}$ as the visual fidelity $E_{\text{visual}}(\mathbf{t}) = |\mathcal{I}_d - \mathcal{I}_d^p(\mathbf{t})|_F^2 + |\mathcal{I}_r - \mathcal{I}_r^p(\mathbf{t})|_F^2$.

Computing texture coordinates $\mathbf{t}$. As the texture coordinates of each triangle in $\mathcal{S}_d$ are mutually independent, we solve them separately. For a triangle $\mathbf{f}_j$, we first determine the view in which it is visible and then compute its texture coordinates. There are four different cases for the perspective projection of $\mathbf{f}_j$:

(1) visible only in the direct view;
(2) visible only in the reflected view;
(3) visible in both the direct and reflected views;
(4) visible in neither the direct view nor the reflection view.

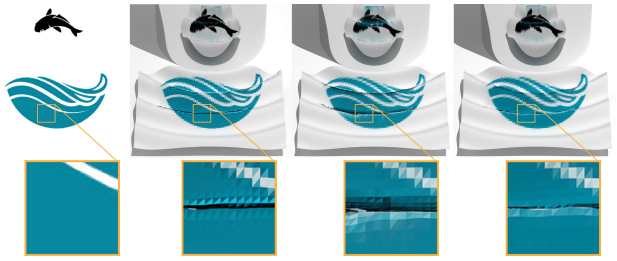
Triangle $\mathbf{f}_j$ is then projected into the 2D image space using rasterization. For the first and second cases, the transformed coordinates are the resulting texture coordinates. For the third case, we use the transformed coordinates in the reflected image $\mathcal{I}_r$ as the texture coordinates. Then, the pixel colors are modified at the transformed positions of $\mathcal{I}_r$ to the average of the colors at the transformed coordinates in $\mathcal{I}_d$ and $\mathcal{I}_r$. For the fourth case, triangle $\mathbf{f}_j$ is invisible; therefore, the texture coordinates are arbitrary.

3.5 Implementation Details and Discussions

The parameter σ in SoftRas. SoftRas [Liu et al. 2019] uses the probability map and aggregate functions to make the standard rasterization pipeline differentiable. In the probability map computation, a parameter (denoted as σ) is used to control the influence range of each triangle in the image plane for evaluating derivatives. The rendered image gets more and more blurred as σ increases. A large σ significantly blurs the rendered images, leading to large visual differences (Figure 9(b)). In contrast, we observe that a small σ causes the optimization to be trapped early by a local minimum due to the small influence range (Figure 9(c)).

To effectively reduce visual distortion, we develop a two-step procedure by first using a large σ parameter for optimization until convergence and then switching to a small σ for refinement (Figure 9(d)). Since the visual fidelity term does not play a key role in the sparse spike strategy, this two-step strategy is only applied in the black-white enhanced optimization, and the sparse spike strategy uses the small σ. In practice, we set large and small values for σ as 10^{-5} and 10^{-7} accordingly.

Target images for optimization. Through construction, images rendered by SoftRas are blurrier compared to those from standard rasterizers. Therefore, if we directly use the input images and the black-white enhanced images as the target images in Problems (6), (8), and (10), respectively, then the differences in sharpness introduce additional errors. Thus, we generate a temporary textured shape by first choosing $\mathcal{I}_d$, the image viewed directly on the saucer by the observer, as the texture image of the saucer shape $\mathcal{S}$. Then, the texturing step is used to compute the texture coordinates. Next, we use SoftRas to render the temporary textured shape to produce an image $\widehat{\mathcal{I}}_d$. When the parameter σ of SoftRas changes, image $\widehat{\mathcal{I}}_d$ is re-rendered using the new σ. We use the same process



(a) Inputs **(b)** $\sigma = 10^{-5}$ **(c)** $\sigma = 10^{-7}$ **(d)** Two-step scheme

Fig. 9. Optimization results using different values for parameter σ.

to generate image $\widehat{\mathcal{I}}_r$ for $\mathcal{I}_r$, the image viewed on the mirror cup, but with a different shape, i.e., the reflected shape of $\mathcal{S}$. Finally, $\widehat{\mathcal{I}}_d$ and $\widehat{\mathcal{I}}_r$ are used as the target images in Problem (6), replacing $\mathcal{I}_d$ and $\mathcal{I}_r$. For black-white enhancement images $\mathcal{I}_d^e$ and $\mathcal{I}_r^e$, we perform the same procedure to generate $\widehat{\mathcal{I}}_d^e$ and $\widehat{\mathcal{I}}_r^e$ to replace $\mathcal{I}_d^e$ and $\mathcal{I}_r^e$ in Problems (8) and (10).

Balance weights. It is difficult to find universal balance weights w in Equation (8) and λ in Equation (10) for different inputs, as they have drastically different visual fidelity energies and shape-preserving energies. Therefore, we propose an input-relevant scheme to automatically set balance weights. First, we measure the compatibility between the input images and the input surface as follows:

(1) Texture the input mesh $\mathcal{S}$ temporarily with $\mathcal{I}_d^e$ by reversing the perspective projection process. Then, the reflection of the mesh in the mirror is rendered to produce a reflected image, denoted as $\mathcal{I}_r^c$.
(2) Generate another temporary textured surface by projecting $\mathcal{I}_r^e$ onto the reflected shape of $\mathcal{S}$ and render the temporary textured shape to obtain a directly viewed image, denoted as $\mathcal{I}_d^c$.
(3) The compatibility score is computed as $\rho = \|\widehat{\mathcal{I}}_d^e - \mathcal{I}_d^c\|_F^2 + \|\widehat{\mathcal{I}}_r^e - \mathcal{I}_r^c\|_F^2$.

After extensive testing, we empirically set $w = 0.08\rho$ when $\sigma = 10^{-5}$ and $w = 0.2\rho$. When $\sigma = 10^{-7}$ in the black-white enhanced optimization, $\lambda = 6 \times 10^{-5}\rho$ for the sparse spike strategy. These weights are fixed throughout our experiments.

Different solvers. The Adamax solver used in our method could be replaced by other gradient-based optimizers, such as LBFGS. We test four gradient-based optimization solvers for our problem (Figure 10), including LBFGS [Liu and Nocedal 1989], SGD [Qian 1999], RMSprop [Tieleman and Hinton 2012], and Adamax [Kingma and Ba 2014]. We use the implementations provided by Pytorch [Paszke et al. 2019] in the experiments. The experimental results show that LBFGS and SGD do not converge, while Adamax converges to a smaller value than RMSprop. Hence, we adopt the Adamax solver provided by Pytorch in all of our experiments.

Cylindrical mirror cups. Our algorithm can be extended to any other cylindrical mirror cups with convex directrix and vertical

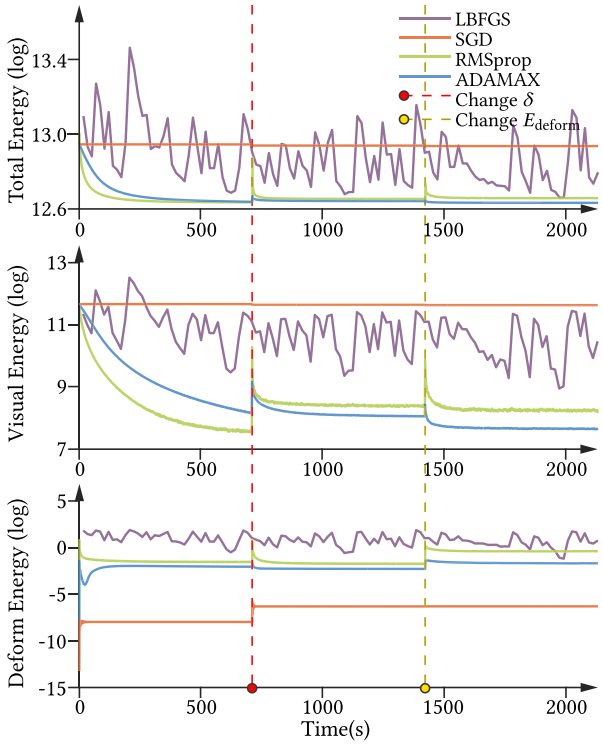


Fig. 10. We test four different solvers and compare how the three loss functions progress on the example in Figure 13(d). Our algorithm consists of three stages. At about 700 seconds, we change the δ value to enter the second stage. At about 1400 seconds, we replace the deformation energy E_{deform} with the sparse energy E_{sparse} to enter the last stage.

generatrix, e.g., elliptical cylinders and regular n-gonal prisms (Figure 19). For these cups, our method for computing reflected shapes is still applicable. If a mirror cup has a non-convex directrix, then a point on the saucer may yield multiple reflections on the cup. If a mirror cup does not have a vertical generatrix, then a point will not necessarily have the same height as its reflection. Since these two situations are not the focus of our work, we take them as future subjects of study.

4 EXPERIMENTAL RESULTS

We tested our method with various input images and saucer surfaces. We implemented our method in Python and performed all the experiments on a desktop PC with a 4.2-GHz Intel Core i7-7700K processor with 32 GB memory and a Nvidia GTX 1080Ti GPU.

Experimental setting. The radius of the cylindrical mirror cup C is 0.4. The origin is placed at the center of the bottom of the mirror cup, and we specify the z-axis direction as the vertical upward direction. The saucer is placed on the negative y-axis. The xy-size of our saucer shape ($\mathcal{S}_d$) is 1×1, and its resolution is 150×150 (Figure 14 shows the results of inputs with some other resolutions). The height values of the saucer vary from 0 to 1.5, and we set the threshold δ value of the height offset to be 0.05. The viewer/camera is placed at $(0, -5.5, 5)$. To ensure that the rendered images cover

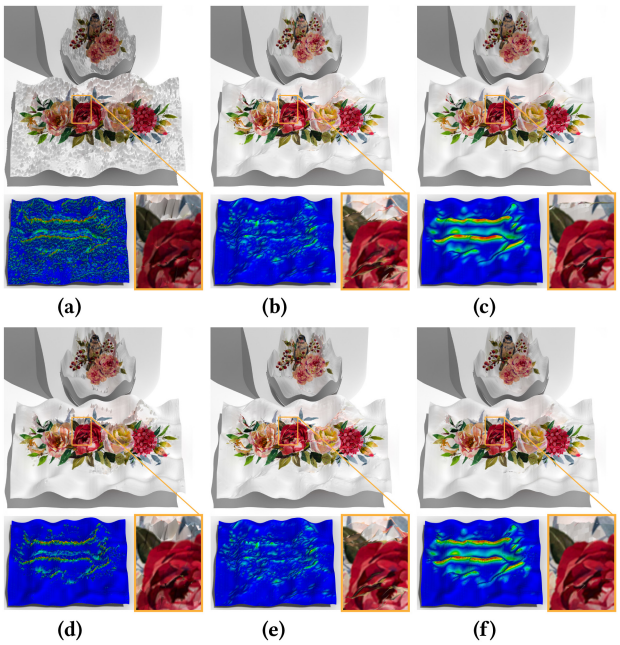


Fig. 11. An ablation study with six different settings. (a) Without any deformation energy. (b) Using Laplacian deformation energy without black-white enhanced images. (c) Using Laplacian deformation energy with black-white enhanced images. (d) Only using sparse deformation energy. (e) Using sparse deformation energy to refine the result in (b). (f) Using sparse deformation energy to refine the result in (c).

the heightfield, we set the target viewing positions for the direct view and the reflected view at $(0, 0, -0.8)$ and $(0, 0, 0.1)$, respectively. The viewing angles are set to 4.5° and 3° accordingly. The rendered images ($\mathcal{I}_d^p(\mathbf{h}, \mathbf{c})$ and $\mathcal{I}_r^p(\mathbf{h}, \mathbf{c})$) has resolution 512×512.

Quality metrics. We use two objective metrics to measure the quality of the results. E_{visual}, as defined in Equation (2), measures the visual fidelity, while $E_{\text{deform}}(\mathbf{h})$, defined in Equation (3), measures the geometric distortion. In addition, we also visualize the change in height using color coding.

Ablation tests. There are three main components of our method that may affect the resulting quality: (1) the black-white enhancement; (2) the deformation energy $E_{\text{deform}}(\mathbf{h})$ in Equation (3), and (3) the sparse spike energy $E_{\text{sparse}}(\mathbf{h})$ in Equation (9). We performed a thorough ablation study to show the necessity of each component. Ideally, these experiments should be performed on fabricated artifacts, as when viewed in real life, some of the color and height differences might appear more or less significant. However, the fabrication process is likely to introduce errors due to imperfections in the manufacturing process, and the viewing position may not perfectly match the specified position due to practical misalignments; therefore, we choose to use rendered results for these experiments. There are six different settings as follows.

In the first setting, we discard the deformation energy $E_{\text{deform}}(\mathbf{h})$ and the sparse spike energy $E_{\text{sparse}}(\mathbf{h})$. Only the black-white enhanced optimization is performed to deform $\mathcal{S}$. The resulting surface is extremely spiky, as shown in Figure 11(a).

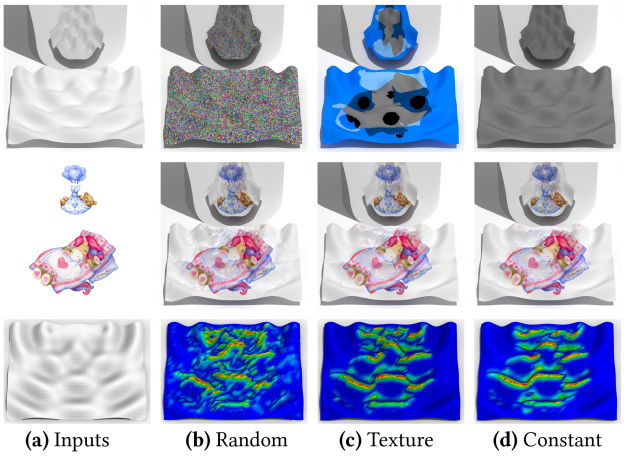


Fig. 12. Different initializations of $\mathbf{c}$. (c) We generate $\mathbf{c}$ using the texturing procedure. (d) We set $(r_i, g_i, b_i) = (0.5, 0.5, 0.5), \forall i$. From top to bottom, we show the initial $\mathbf{c}$, the resulting rendered images, and the absolute values of the resulting height changes.

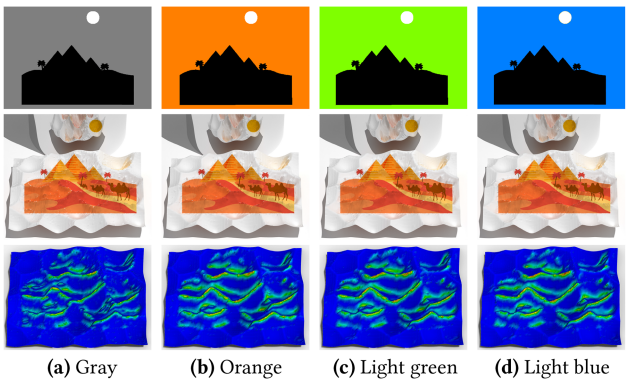


Fig. 13. Four different background colors in black-white enhancement.

The second setting does not use the deformation energy $E_{\text{deform}}(\mathbf{h})$, i.e., only Equation (10) is solved. As Figure 11(d) shows, even though the geometric distortion is concentrated in a small portion of the saucer, large distortion is still present.

In the third and fourth settings, we do not apply the black-white enhancement. The third setting only solves Equation (6) (Figure 11(b)). The fourth setting first solves Equation (6) and then applies the sparse spike strategy, where $E_{\text{visual}}^{\text{new}}(\mathbf{h}, \mathbf{c})$ is replaced by $E_{\text{visual}}(\mathbf{h}, \mathbf{c})$ (Figure 11(e)).

The fifth setting does not use $E_{\text{sparse}}(\mathbf{h})$, i.e., only Equation (8) is solved (Figure 11(c)). Our method is the sixth setting (Figure 11(f)).

The texturing step is performed for all six settings. In comparing Figure 11(b) (or Figure 11(e)) with Figure 11(c) (or Figure 11(f)), we see that the black-white enhancement significantly reduces visual distortion. From the comparison between Figures 11(b) (or Figure 11(c)) and Figure 11(e) (or Figure 11(f)), we see that the sparse spike energy also helps reduce visual artifacts.

Different initializations of $\mathbf{c}$. The initialization for $\mathbf{c}$ in $\mathcal{S}_d$ can affect the results of the optimization problem (Equation (8)). We

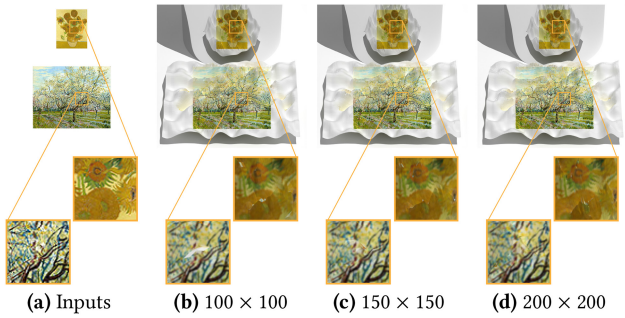


Fig. 14. Results with heightfields of different resolutions.

tested three different types of initializations: (1) random values (Figure 12(b)), (2) $\mathbf{c}$ generated by the texturing step (Figure 12(c)), and (3) uniform gray color initialization, $(r_i, g_i, b_i) = (0.5, 0.5, 0.5)$ (Figure 12(d)). For the second initialization, we use the black-white enhanced images $\mathcal{I}_d^e$ and $\mathcal{I}_r^e$ in the texturing step.

For the results in Figure 12, $(E_{\text{visual}}, E_{\text{deform}})$ in (b), (c), and (d), are (3330, 0.22), (2401, 0.18), and (2405, 0.16), respectively. As both the visual comparison and the quantitative quality comparison show, the first initialization leads to the lowest quality, and the other two initializations lead to comparable results. Therefore, we choose the third initialization throughout our experiments for its simplicity.

Background color for black-white enhancement. In Figure 13, we show an example of the black-white enhancement, where we fill the non-salient background regions with four different colors: (1) gray (0.5, 0.5, 0.5), (2) orange (1, 0.5, 0), (3) light green (0.5, 1, 0), and (4) light blue (0, 0.5, 1). It is apparent that the latter three colors lead to visual results that are almost identical. Furthermore, they produce lower visual distortion (2596, 2585, and 2540, respectively) and geometric distortion (0.2 for all three colors), compared to the first choice (gray), where the visual distortion and geometric distortion are 2798 and 0.22 accordingly.

The black-white enhancement mitigates the emergence of conflict areas by magnifying visual distortions. Therefore, in general, better results can be obtained if there is high contrast between the background color and the foreground color (black or white). In this experiment, the gray color has lower contrast with black (or white), compared to the other three colors. As a result, gray produces less favorable results, so we use light blue in all our experiments.

Resolution of the heightfield. We use heightfields with three different resolutions as input for optimization, as shown in Figure 14. Unsurprisingly, low-resolution heightfields lead to artifacts (see the zoomed-in views in Figure 14(b)). Figure 14(c) and (d) show that comparable results can be obtained for resolutions 150×150 and 200×200. However, compared to a resolution of 150×150, it takes twice the time to optimize with a resolution of 200×200. The reason is that the differential rendering has a time complexity that is nearly linear to the number of triangles in $\mathcal{S}$. To balance the running time and the visual quality, we set the resolution of our heightfield surface to 150×150 by default.

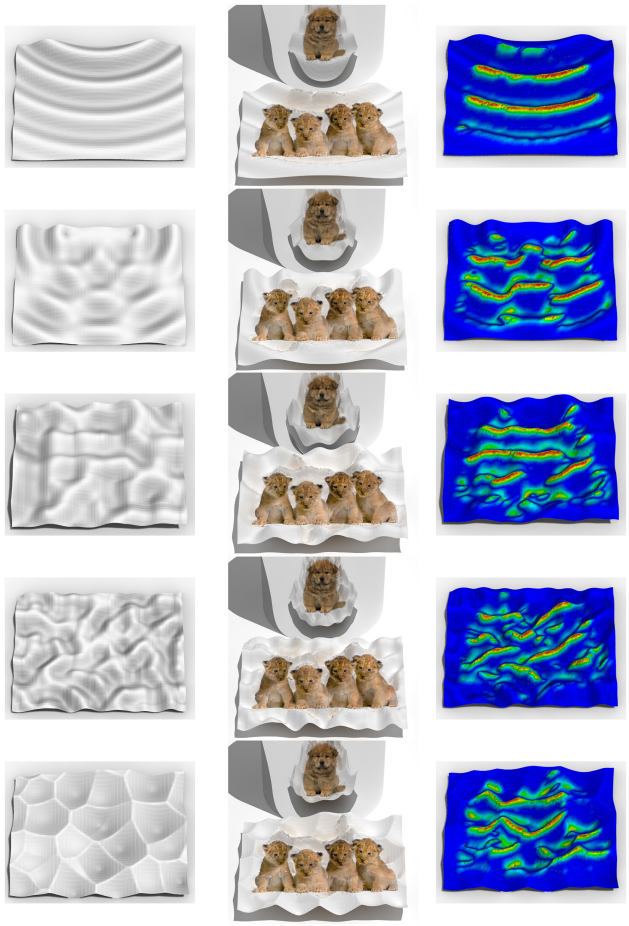


Fig. 15. Different shapes and one input image. From left to right, we show the input shapes, the resulting rendered images, and the absolute values of the resulting height changes. From top to bottom, the visual distortions are 2100, 2480, 2541, 2388, and 2563.

Varying input surfaces. In Figure 15, we test five different input surfaces $\mathcal{S}$ for one given pair of images. Our method generates visually comparable results, and the visual distortions are similar. This indicates that our algorithm produces suitable deformations that adapt to the input surface $\mathcal{S}$. Thus, our method is robust enough to handle various input surfaces in practice.

Varying input images. In Figure 16, we produce five results using different input images with the same surface. It can be seen that our method is capable of dealing with different input images, as it manages to reduce the visual error to a low level and generate high-quality output.

Stress test. We perform a stress test of our method with three typical input surfaces $\mathcal{S}$, along with three different input images. The results in Figure 18 demonstrate the robustness of our algorithm. The second column shows, given either a planar or a random heightfield as the input shape, our algorithm manages to produce a high quality result instead of getting stuck in a suboptimal local solution. The third row shows that when the input

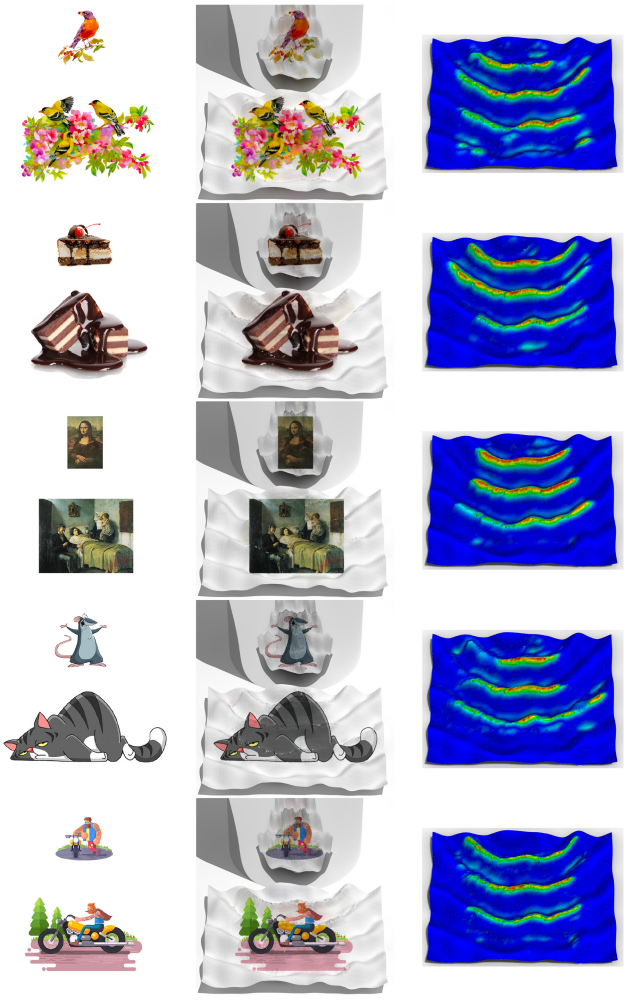


Fig. 16. Different input images and one saucer shape. From left to right, we show the input images, the resulting rendered images, and the absolute values of the resulting height changes. From top to bottom, the visual difference errors are 3218, 2824, 2387, 3157, and 2422.

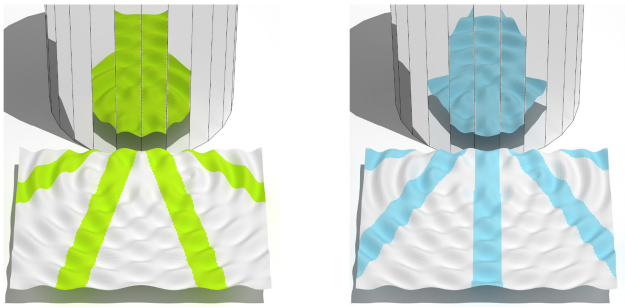


Fig. 17. When the mirror cup C is a regular n-gonal prism, only the colored part of the saucer surface $\mathcal{S}$ can be seen on the mirror. With these two placement options for C, the green areas (left) and the blue areas (right) hardly overlap.

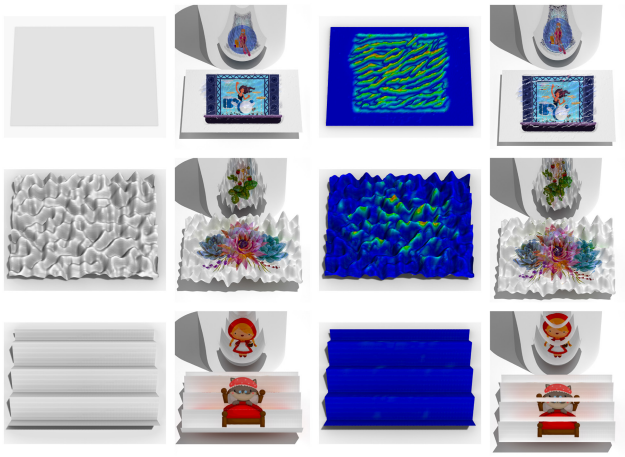


Fig. 18. Three examples of stress tests. From left to right, we show the input shapes, the resulting rendered images, the absolute values of the resulting height changes, and the resulting rendered images from an offset viewpoint. The camera is placed at $(0.0, -4.5, 6.0)$ and looks at $(0.0, 0.0, -0.5)$. From top to bottom, the input shapes are a plane, a random heightfield, and tabula scalata.

shape is sufficient, our algorithm can optimize only the texture of the saucer while the shape remains fixed. As seen in these examples, our framework does not enforce any constraints on the input shape, instead it takes any shape (in the form of a heightfield) as input. Although the resulting shape can be very different from the input in certain cases, such as the plane.

Multiple reflection images. When the mirror cup C is a regular n-gonal prism, we can obtain multiple reflected images. As Figure 17 shows, the area on the saucer $\mathcal{S}$ that is reflected onto the cup C does not cover the entire surface of the saucer $\mathcal{S}$. In general, there are two options for where to place C to produce two reflected areas that are nearly non-overlapping (Figure 17). Consequently, we are able to generate two reflected images and each of them corresponds to one reflection area.

When generating two reflected images, some algorithmic details are different from our default cylinder case. First, two reflected shapes must be computed. Second, the visual fidelity energy contains three terms, two for the reflections and one for the direct view. Third, the black-white enhanced images become solid-color enhanced images, i.e., we fill in the colored regions of the three input images with *red*, *green*, and *blue*, respectively, and the other regions are filled in with *white*. Figure 19 shows four examples.

More examples. We collect twenty three different images and five surface shapes as inputs for our method. Figure 20 shows the results with 18 combinations. For the five remaining examples, we use 3D printing technology to manufacture them, as shown in Figure 21. These results show that our method is indeed robust for different input images and saucer shapes in practice.

Runtime. Typically, each iteration of our algorithm takes about 0.72 seconds for a heightfield with a resolution of 150×150 and the rendered images with 512×512 resolution. Due to the slow convergence, it can take thousands of iterations for our optimizer

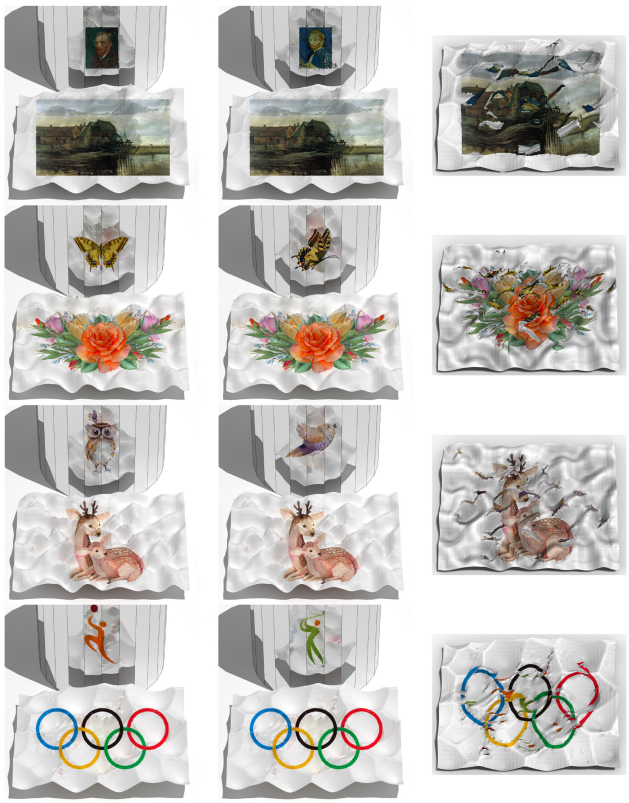


Fig. 19. Four examples with two reflected images. In the left and middle columns, the resulting rendered images with two different reflected images are shown. We show the textured saucers in the right column.

to converge. In all our tests, we observed that none of the examples took more than 30 minutes. In future work, we believe that substantial increases in speed are possible, as our code is not yet optimized.

5 CONCLUSION

We propose a computational tool for automatically designing mirror cup and saucer artwork. To reduce visual differences while limiting geometric distortion, we develop a method for conflict-area minimization followed by a texturing step. Two essential techniques are proposed to minimize conflict areas: (1) black-white enhancement and (2) sparse spike strategy. To effectively and efficiently perform these two techniques, we propose first computing the reflected surfaces for the reflected images and then using differentiable rendering technology to compute the objective function gradients for optimization. We have demonstrated the capability and practicality of our method on an extensive set with forty six examples. Our system allows users to adjust the weights in Problems (8) and (10), to balance the tradeoff between visual fidelity and shape preservation; for example, large weights w and λ, one can reduce surface deformation while allowing greater image difference.

Visual distortion. In practice, real users' perspectives may not exactly match a specified viewing position and direction, which causes apparent visual distortions. We believe that this can be alleviated by adding visual distortion for some offset viewing positions into the visual fidelity energy term; this allows actual perspectives to slightly deviate from the specified viewing positions.

3D printed examples. It can be seen from Figure 21 that the real examples fabricated by 3D printing are visually not as perfect as in the simulation. The quality of fabricated objects depends on the printer's precision and accuracy in both geometry and color. In our experiments, we used a SLA 3D printer (Stratasys J55) to fabricate all our examples. We observed that the printed color is blurred due to the penetration of differently colored printing material at the surface edges. Recently, Elek et al. [2017], Sumin et al. [2019], and Nindel et al. [2021] have developed solutions to this very problem. In the future, we would like to integrate these solutions into our system.

User interactions. Due to the long running time of our method, it is currently infeasible to provide an interactive user interface for design. Consequently, it is challenging to incorporate user's design intention into the resulting mirror cup art. In the future, we wish to speed up our system, using more efficient differentiable rendering technologies and numerical solvers, thereby enabling an interactive user interface.

Shape preservation term. We measure shape deviation using the difference in the Laplacian coordinates of the heightfields. However, other geometric features, such as edge absence, curvature, extrimities, and so on, can be equally important. In the future, we hope to incorporate these factors to allow better preservation of the input saucer shape.

Uncolored areas. In this work, we choose not to color the areas of the saucer surface that are invisible in the input viewing direction, and this can make the saucer less appealing as an art piece. In general, viewers tend to prefer a saucer surface with overall continuous and smooth texture, which makes it look more pleasant when viewed from other directions. In future work, it is worth considering the smoothness of color transitions over the saucer surface, specifically, between neighboring colored areas, given that the saucer's aesthetic quality is important.

APPENDIX

A PROOF FOR PROPOSITION 3.1

Proof. Suppose that $\angle TPO$ decreases as $\angle POT$ increases. Then the slope of the moving direction of T should be less than the slope of PT, which means the slope of l_t should be less than that of PT. Hence, $\angle POT > \angle POG$, which is impossible, since $\angle POT \leq \min\{\angle POQ, \angle POG\}$. Therefore $\angle PTV = \angle TOP + \angle TPO$ is strictly increasing. Similarly, $\angle QTV$ is strictly decreasing, and $\angle PTV - \angle QTV$ is strictly increasing in that range.

When $\angle POT = 0$, we have $\angle PTV = 0$ and $\angle QTV > 0$, hence $\angle PTV - \angle QTV < 0$. When $\angle POT = \angle POQ < \angle POG$, we have $\angle PTV > 0$ and $\angle QTV = 0$, hence $\angle PTV - \angle QTV > 0$. When $\angle POT = \angle POG < \angle POQ$, we have $\angle PTV = 90°$. If $\angle QTV > 90°$, then QT will intersect with C, and therefore $\angle QTV \leq 90°$, and $\angle PTV - \angle QTV \geq 0$. In summary, $\angle PTV - \angle QTV$ changes from negative to positive, when $\angle POT$ increases from 0 to $\angle POQ$. □

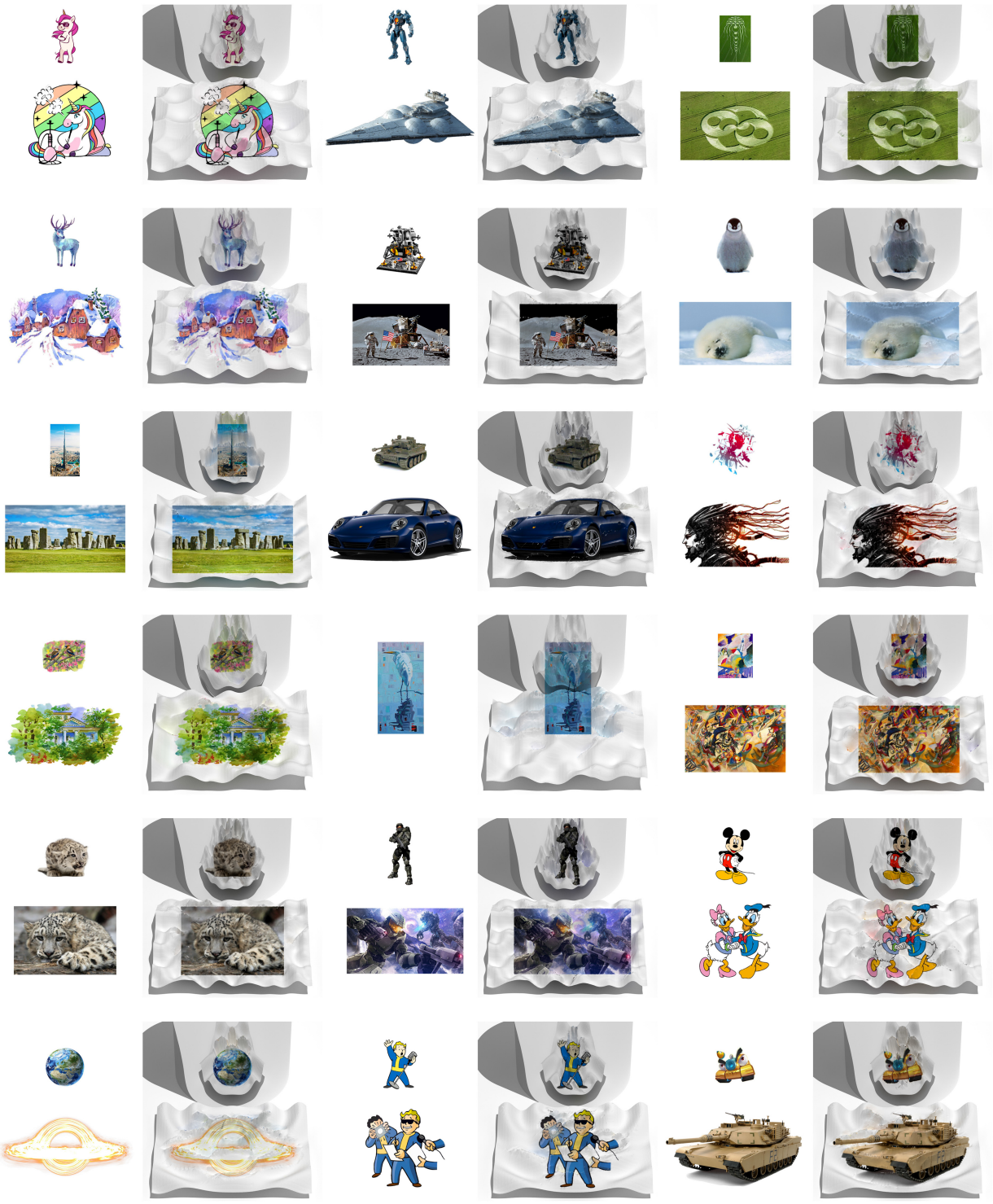


Fig. 20. Eighteen mirror cup art examples generated by our method.

Fig. 21. A gallery of five mirror cup art pieces physically fabricated with 3D printing.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive suggestions and valuable comments.

REFERENCES

Marc Alexa and Wojciech Matusik. 2010. Reliefs as images. *ACM Trans. Graph.* 29, 4 (2010), 60–1.

Jurgis Baltrusaitis. 1977. *Anamorphic Art.* Chadwyck-Healey Translated by WJ Strachan. Cambridge, England.

Amit Bermano, Ilya Baran, Marc Alexa, and Wojciech Matusk. 2012. Shadowpix: Multiple images from self shadowing. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 593–602.

Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. 2019. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems*. 9609–9619.

Aleksandar Čučaković and Marijana Paunović. 2015. Cylindrical mirror anamorphosis and urban-architectural ambience. *Nexus Netw. J.* 17, 2 (2015), 605–622.

Francesco De Comite. 2011. A new kind of three-dimensional anamorphosis. In *Bridges 2011: Mathematics, Music, Art, Architecture, Culture*. 33–38.

Francesco De Comite and Laurent Grisoni. 2015. Numerical anamorphosis: An artistic exploration. In *SIGGRAPH ASIA 2015 Art Papers*. 1–7.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-image svbrdf capture with a rendering-aware deep network. *ACM Trans. Graph.* 37, 4 (2018), 1–15.

D. Eberly. 2008. Computing a point of reflection on a sphere. Retrieved from http://www.geometrictools.com.

Oskar Elek, Denis Sumin, Ran Zhang, Tim Weyrich, Karol Myszkowski, Bernd Bickel, Alexander Wilkie, and Jaroslav Krivanek. 2017. Scattering-aware texture reproduction for 3D printing. *ACM Trans. Graph.* 36, 6 (2017).

Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. 2018. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8377–8386.

Dianne Hansford and Daniel Collins. 2007. Anamorphic 3D geometry. *Computing* 79, 2–4 (2007), 211–223.

Paul Henderson and Vittorio Ferrari. 2018. Learning to generate and reconstruct 3D meshes with only 2D supervision. In *Proceedings of the 29th British Machine Vision Conference (BMVC'18)*.

Kai-Wen Hsiao, Jia-Bin Huang, and Hung-Kuo Chu. 2018. Multi-view wire art. *ACM Trans. Graph.* 37, 6 (2018), 242–1.

James L. Hunt, B. G. Nickel, and Christian Gigault. 2000. Anamorphic images. *American J. Phys.* 68, 3 (2000), 232–237.

Danilo Jimenez Rezende, S. M. Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. 2016. Unsupervised learning of 3d structure from images. *Adv. Neural Inf. Process. Syst.* 29 (2016), 4996–5004.

Marko Jovanovic, Vesna Stojakovic, Bojan Tepavcevic, Dejan Mitov, and Ivana Bajsanski. 2016. Generating an anamorphic image on a curved surface utilizing robotic fabrication process. In *Complexity and Simplicity": Proceedings of the 34th eCAADe Conference*. 185–191.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3907–3916.

Jeroen Keiren, Freek van Walderveen, and Alexander Wolff. 2009. Constructability of trip-lets. In *Proceedings of the 25th European Workshop on Computational Geometry*. 251–254.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/abs/1412.6980.

Abhijit Kundu, Yin Li, and James M. Rehg. 2018. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3559–3568.

Ying-Miao Kuo, Hung-Kuo Chu, Ming-Te Chi, Ruen-Rone Lee, and Tong-Yee Lee. 2016. Generating ambiguous figure-ground images. *IEEE Trans. Vis. Comput. Graph.* 23, 5 (2016), 1534–1545.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph.* 37, 6 (2018), 1–11.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Program.* 45, 1 (1989), 503–528.

Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. 2017. Material editing using a physically based rendering network. In *Proceedings of the IEEE International Conference on Computer Vision*. 2261–2269.

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 International Conference on Computer Vision*. 7708–7717.

Matthew M. Loper and Michael J. Black. 2014. OpenDR: An approximate differentiable renderer. In *European Conference on Computer Vision*. Springer, 154–169.

Sehee Min, Jaedong Lee, Jungdam Won, and Jehee Lee. 2017. Soft shadow art. In *Proceedings of the Symposium on Computational Aesthetics*. 1–9.

Niloy J. Mitra and Mark Pauly. 2009. Shadow art. *ACM Trans. Graph.* 28, 5 (2009), 1–7.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.* 38, 6 (2019), 1–17.

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

Aude Oliva, Antonio Torralba, and Philippe G. Schyns. 2006. Hybrid images. *ACM Trans. Graph.* 25, 3 (2006), 527–532.

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.

Neal Parikh and Stephen Boyd. 2014. Proximal algorithms. *Found. Trends Optim.* 1, 3 (2014), 127–239.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* 32 (2019), 8026–8037.

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

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural Netw.* 12, 1 (1999), 145–151.

Kimberly Rausch et al. 2012. The mathematics behind anamorphic art. In *Proceedings of Bridges 2012: Mathematics, Music, Art, Architecture, Culture*. Tessellations Publishing, 513–514.

Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. 2017. Learning detailed face reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1259–1268.

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

Javier Sánchez-Reyes and Jesús M. Chacón. 2016. Anamorphic free-form deformation. *Comput. Aid. Geom. Des.* 46 (2016), 30–42.

Javier Sánchez-Reyes and Jesús M. Chacón. 2020. How to make impossible objects possible: Anamorphic deformation of textured NURBS. *Comput. Aid. Geom. Des.* 78 (2020), 101826.

Christian Schüller, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Appearance-mimicking surfaces. *ACM Trans. Graph.* 33, 6 (2014), 1–10.

Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. 2014. High-contrast computational caustic design. *ACM Trans. Graph.* 33, 4 (2014), 1–11.

Guy Sela and Gershon Elber. 2007. Generation of view dependent models using free form deformation. *Vis. Comput.* 23, 3 (2007), 219–229.

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

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP'04)*. 175–184.

Vesna Stojakovic and Bojan Tepavcevic. 2016. Distortion minimization: A framework for the design of plane geometric anamorphosis. *Nexus Netw. J.* 18, 3 (2016), 759–777.

Denis Sumin, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, J. KŢivánek, Karol Myszkowski, and Tim Weyrich. 2019. Geometry-aware scattering compensation for 3D printing. *ACM Trans. Graph.* 38, 4 (2019).

Ayush Tewari, Michael Zollhöfer, Pablo Garrido, Florian Bernard, Hyeongwoo Kim, Patrick Pérez, and Christian Theobalt. 2018. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2549–2559.

Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. 2017. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 1274–1283.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. Technical Report, University of Toronto.

Luan Tran and Xiaoming Liu. 2018. Nonlinear 3d face morphable model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7346–7355.

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

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-space differentiable rendering. *ACM Trans. Graph.* 39, 4 (2020), 143.

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A differential theory of radiative transfer. *ACM Trans. Graph.* 38, 6 (2019), 1–16.

Haisen Zhao, Lin Lu, Yuan Wei, Dani Lischinski, Andrei Sharf, Daniel Cohen-Or, and Baoquan Chen. 2016. Printed perforated lampshades for continuous projective images. *ACM Trans. Graph.* 35, 5 (2016), 1–11.

Jacek Zienkiewicz, Andrew Davison, and Stefan Leutenegger. 2016. Real-time height map fusion using differentiable rendering. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'16)*. IEEE, 4280–4287.

Received July 2021; revised January 2022; accepted February 2022