Lenticular Objects: 3D Printed Objects with Lenticular Lens Surfaces That Can Change their Appearance Depending on the Viewpoint

Jiani Zeng[*]
MIT
Cambridge, USA
jnzeng@mit.edu

Honghao Deng[*]
MIT
Cambridge, USA
honghaod@mit.edu

Yunyi Zhu[*]
MIT CSAIL
Cambridge, USA
yunyizhu@mit.edu

Michael Wessely
MIT CSAIL
Cambridge, USA
wessely@mit.edu

Axel Kilian
MIT Architecture
Cambridge, USA
akilian@mit.edu

Stefanie Mueller
MIT CSAIL
Cambridge, USA
stefanie.mueller@mit.edu

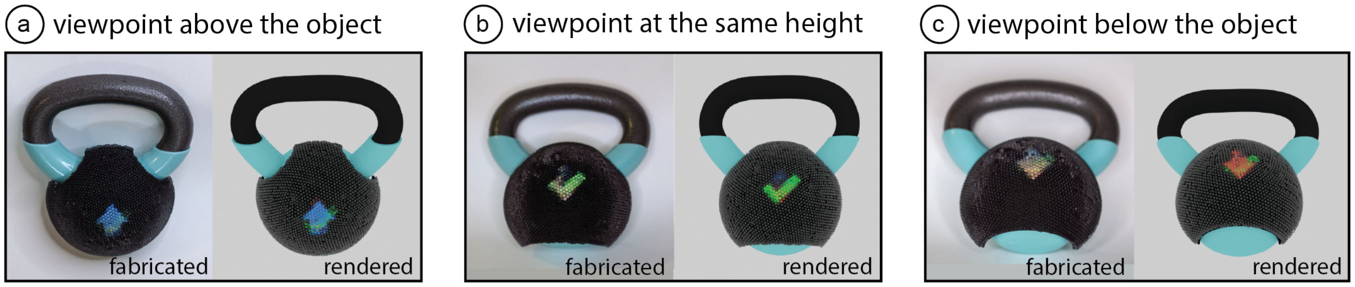
a viewpoint above the object
b viewpoint at the same height
c viewpoint below the object
fabricated
rendered
fabricated
rendered
fabricated
rendered

Figure 1: 3D printed objects with lenticular lens surfaces enable viewers to see different appearances from different viewpoints (a: looking down; b: looking horizontally; c: looking up). Our user interface supports designers in setting up different viewpoints and assigning the corresponding textures. On export, it automatically generates the files for fabrication. Designers can then 3D print the object geometry, lenses, and underlying color patterns in a single pass with a multi-material 3D printer.

ABSTRACT

In this paper, we present a method that makes 3D objects appear differently under different viewpoints. We accomplish this by 3D printing lenticular lenses across the curved surface of objects. By calculating the lens distribution and the corresponding surface color patterns, we can determine which appearance is shown to the user at each viewpoint.

We built a 3D editor that takes as input the 3D model, and the visual appearances, i.e. images, to show at different viewpoints. Our 3D editor then calculates the corresponding lens placements and underlying color pattern. On export, the user can use ray tracing to live preview the resulting appearance from each angle. The 3D model, color pattern, and lenses are then 3D printed in one pass on a multi-material 3D printer to create the final 3D object.

To determine the best fabrication parameters for 3D printing lenses, we printed lenses of different sizes and tested various post-processing techniques. To support a large number of different appearances, we compute the lens geometry that has the best trade-off between the number of viewpoints and the protrusion from the object geometry. Finally, we demonstrate our system in practice with a range of use cases for which we show the simulated and physical results side by side.

[*]authors contributed equally

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**.

KEYWORDS

multi-material 3D printing; optics; lenticular lenses; design tools.

ACM Reference Format:
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 (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3472749.3474815

1 INTRODUCTION

Lenticular printing refers to the method of using lenticular lenses to show different images from different viewpoints [30]. It is commonly applied in advertisement and arts to achieve special purpose effects, such as showing various images on one commercial motion card [9, 48] or creating artistic effects with the illusion of depth [8, 19].

Traditionally, lenticular prints are limited to 2D. They are fabricated by placing flat lenticular sheets onto 2D color patterns, which are composed of multiple images. Depending on the viewpoint, light rays enter the lens at different angles and thus reflect off different parts of the underlying color pattern. Therefore, under different viewpoints, a different part of the color pattern is visible to the viewer's eye and as a result the viewer sees only a particular image that is contained in the color pattern.

One reason why lenticular prints do not yet exist in the form of 3D objects is that up until recently, no fabrication process existed that was able to manufacture the lenticular lenses and high resolution color patterns on doubly curved surfaces. However, over the last years, multi-material 3D printers have been developed that can print with a multitude of materials and in high-resolution color [4]. For example, the 3D printer Stratasys J55 [40] can print with optical clear materials, which can be used for the lenses, as well as CMYK materials, which can be used to print the color pattern.

In this paper, we explore how to leverage the recent advances in multi-material 3D printing to create curved 3D objects with lenticular surfaces. To facilitate the creation of objects that look different from different viewpoints, we provide designers with a 3D editor that takes as input the 3D model, the desired viewpoints, and the corresponding images and then computes the lens placement and color pattern across the object's surface (Figure 1). Before fabrication, designers can preview the resulting object from each viewpoint via ray tracing, and then send it to the 3D printer.

To determine the best fabrication parameters for 3D printing lenses, we printed different lens sizes (2mm-5mm) and tested different post-processing techniques (different types of varnishes and oil). To achieve a large number of different appearances on one 3D object, we computed the lens geometry that has the best tradeoff between a large number of viewpoints and avoiding protrusion of the lenses from the object geometry. We show that although our system can support up to 19 different appearances per object as shown by our digital ray tracing simulation, inaccuracies in current fabrication techniques do not yet allow for these results in practice. We demonstrate the usefulness of lenticular objects with 4 application examples in product design and HCI showing the simulated and the physical results side by side.

In summary, we contribute:

- an end-to-end fabrication pipeline for printing lenticular objects in a single pass on a multi-material 3D printer by printing lenticular lenses and color patterns on doubly-curved object geometries;
- an interactive 3D editor plug-in that allows designers to define multiple viewpoints and assign the corresponding textures and that generates the resulting fabrication files;
- four application scenarios that demonstrate lenticular objects made with our fabrication pipeline.

2 RELATED WORK

Our work is related to HCI research that uses optics for novel use cases, especially for tangible applications; projects that fabricate lenticular lenses in custom shapes and sizes; and technologies that display appearances depending on the viewpoint.

2.1 Optics in HCI for Tangible Applications

In HCI and graphics, innovative use of optics has led to many novel technologies. While there exist a large body of work to improve screen-based display technology via optics, for instance to achieve hand-held stereoscopic interfaces (HoloFlex [13]), Matusik et al. [32]), to create improved VR headsets (ThinVR [37], Near-eye light field displays [26]), or to illuminate scenes (AnyLight [45]), we focus our review of the related work on optics embedded into tangible objects.

To create new types of tangibles, researcher made use of different types of optics, such as lenses, mirrors and light fibers. For instance, WonderLens [29] enhances interaction with paper by using either mirrors to allow users to make copies of printed patterns or deformable lenses to create paper animations. Lumino [3] are tangible cubes with embedded light fibers that allow interactive tabletops to sense interactions, such as stacking and rotating of cubes, through transmitted marker information. FlyEye [51] shows how light fibers can be used to integrate sensing into hand-held objects by measuring light reflected off the user's finger with an image sensor. Rock-Paper-Fibers [38] builds on this and presents a hand-held device in the form of a light fiber bundle that can sense when users rearrange it into different shapes. LightBundle [25], finally, provides a design space for different interactions with light fiber bundles.

Beyond sensing interaction, researchers also investigated how to use optics to transform tangibles into displays. LightCloth [15], for instance, is a woven piece of cloth that consists of light fibers with LEDs attached to each fiber to create color on the surface. Similarly, FuSA^2 [33] is a furry color screen made from bundled light fibers with one color per fiber. While these works show the potential of using optics to convert tangible objects into displays, they use external light sources, i.e. LEDs, to create different appearances on the objects surface. In contrast, our work is based on ambient light. Finally, while these works show the same appearance from each angle, our work can display viewpoint dependent content.

2.2 Fabricating Lenticular Lenses

Lenticular lenses exist both as cylindrical and spherical lenses ('microlenses'). Cylindrical lenses are traditionally fabricated in sheets by melting and extruding plastic pellets into thin cylinders, and are common commercially available products [7]. Spherical microlenses can be fabricated in labs using lithography techniques. For instance, Jonušauskas et al. [23] fabricate microlenses using femtosecond 3D laser lithography and Jonušauskas et al. [24] fabricated aspheric microlenses using direct laser writing. Researchers also investigated the use of inkjet printers in the fabrication of microlenses. For instance, Alamán et al. [2] investigated types of inks that can be used to inkjet print micro-optics and determined how inkjet printing can be used for the pre-patterning process.

To fabricate optics of custom shapes and sizes, researchers have also used digital fabrication tools, such as 3D printers and CNC milling machines. For instance, Weyrich et al. [49] mill micro-mirrors in a specific arrangement so that reflected directional light forms an image. Directional Screens [35] consist of milled mirrors from an aluminum projection screen, the fabricated mirrors show image content only for users that sit in a specific location to preserve energy.

More closely related to our work, researchers have also 3D printed optics, such as light fibers (Printed Optics [50], Papillon [6]) and lenses. Sugiyama et al. [44], for instance, explore how to best 3D print lenses by improving lens quality through multiple print passes. To improve image quality, Content-adaptive Lenticular Prints [46] fabricate lenses of different sizes and at different heights of the backplane of flat lenticular displays. Finally, the Magic Lens [34] prints customized lens arrays that when placed on a flat 2D printed color pattern show different images when users rotate the lens arrays. In contrast, in our system, we 3D print lens arrays on top of curved 3D objects. In addition, our method changes an object's appearance when the user looks at the object from different viewpoints rather than when rotating the lenses.

2.3 Viewpoint Dependent Appearances

There are several different techniques for creating viewpoint dependent appearances. While there exist a large body of work on digital techniques (e.g., via polarization of LCD screens [14, 27], projection on refractive surfaces [16], spatially aware displays [28], and frame-interleaving displays [1]), we focus our review of the related work on physical display methods that do not require a digital screen or projection.

Several physical display techniques have created viewpoint dependent appearances either by designing the object geometry in a way that makes it appear different from different viewpoints, or by leveraging viewpoint dependent reflectance properties of colors. Hsiao et al. [18], for instance, create wire-objects that display different contours from different viewpoints, while Pjanic et al. [36] and Sakurai et al. [39] show how to fabricate color patterns that have different hues when viewed from different viewpoints.

Closest to our work are approaches that fabricate a color pattern that encodes multiple images but only shows one of them to the viewer. This can be done by, for instance, using parallax barriers stereograms [20], which consist of a color pattern with an opaque cover with vertical barriers that shield different parts of the color pattern depending on the viewpoint. Lumii [17] uses a similar approach but uses two print layers and a structurally different barrier layer through which the viewer sees the image, which allows the image to be of higher resolution. Finally, instead of using barriers, a different way to show only a part of the underlying color pattern is to use lenses and their magnifying effect [30]. Since each lens magnifies only a part of the color pattern, users only see one of the encoded images from a particular viewpoint. So far, this approach has only been realized on flat or single curved surfaces (Ji et al. [22], Vection Field [11]). In our work, we instead integrate the lenses onto doubly curved 3D object geometries to create 3D objects with varying appearances.

3 BACKGROUND: LENTICULAR DISPLAYS

Figure 2 shows a simplified illustration of a lenticular lens that displays a different appearance under each viewpoint. To create this effect, the lens has a color pattern consisting of multiple colored image spots underneath. Because of the magnifying effect of the lens, it displays the color from only one of the colored image spots, which is only a small portion of the entire area under the lens (Figure 2a). Because each lens only shows one image spot from each viewpoint, the colored image spot is the equivalent to one pixel in a display. Which image spot the viewer sees and thus which color the 'pixel' has, depends on the viewpoint of the viewer due to the different incident angles of the light hitting the lens (Figure 2b). Since each lens represents one pixel in the display, multiple lenses together form a lenticular display and collectively show an image that varies dependent on the viewpoint.

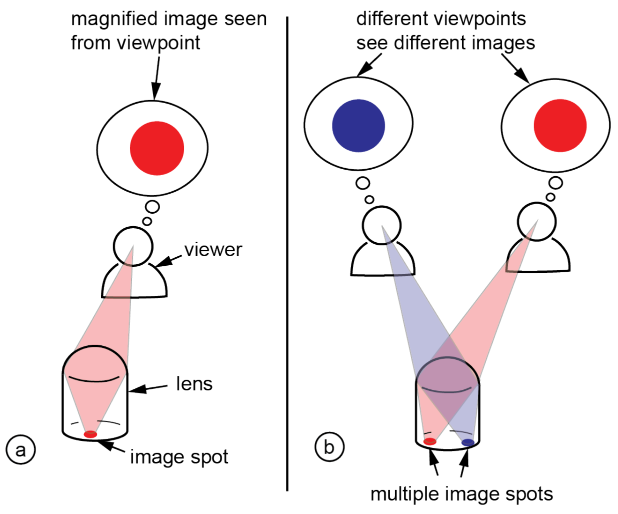
magnified image seen from viewpoint
viewer
lens
image spot
a
different viewpoints see different images
b
multiple image spots

Figure 2: (a) Magnifying effect of a lens: from each viewpoint, the user sees only a small fraction of the underlying color pattern, i.e. one image spot. (b) From different viewpoints, users see different image spots under the lens, thus from each viewpoint the lens shows a different color.

The quality of a lenticular display is determined by the size of the lenses and the number of image spots that fit under a lens. The smaller the lens, the smaller each pixel is, and thus the more pixels can fit in a given area, resulting in a higher resolution of the image. The number of image spots that fit under a lens determines how many different colors a pixel can take on, and thereby determines how many images the viewer can see from the overall display. We show later in our paper how we compute the lens geometry that has the best trade-off between the number of viewpoints (i.e. image spots underneath the lens) and the protrusion from the object geometry. In addition, we will report the results from experiments, in which we investigated the trade-off between the size and the resulting quality of a lens. Before reporting on these experiments, we will illustrate in the next section how to create 3D printed objects with lenticular surfaces with our custom design tool.

4 DESIGN TOOL FOR 3D LENTICULAR DISPLAYS

To support designers in creating 3D lenticular displays, we developed a design tool that is integrated into an existing 3D modeling environment (Rhino3D)[1]. Designers start by loading the 3D model of the object and then defining a set of viewpoints and corresponding visual appearances. Our editor then automatically places the lenses on the 3D geometry and assigns the corresponding color pattern to each lens. Before fabrication, designers can preview the resulting object via ray tracing. On export, our tool provides a set of fabrication files ready for 3D printing.

In the next section, we demonstrate the functionality of our editor at the example of a kettlebell that guides the user into the correct exercise pose, i.e. shows a checkmark when the user holds the kettlebell at the correct height, and an upward and downward arrow when the kettlebell is being held too low or too high (Figure 1).

4.1 Defining Viewpoints by Placing Virtual Cameras

After loading the 3D model of the object, designers define the viewpoints by placing virtual cameras in the viewport at the desired 3D positions (Figure 3a). To do this, designers first click the 'new viewpoint' button, which creates a virtual camera in the viewport. It also adds the viewpoint to the list of all viewpoints in the panel. Designers can then either move the camera in the viewport or alternatively enter a 3D coordinate to position it. While the designer is moving the camera, our editor automatically orients the camera to always face towards the object. Designers can verify that the viewpoint is correctly positioned by clicking 'go to viewpoint' in the panel, which shows the view from the selected virtual camera to the object.

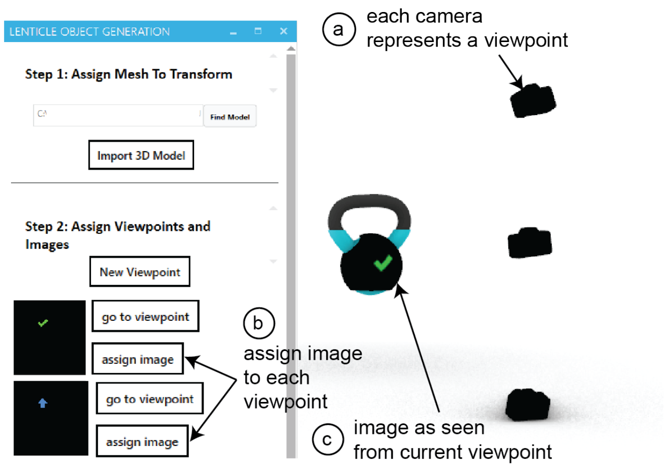
LENTICLE OBJECT GENERATION
Step 1: Assign Mesh To Transform
Find Model
Import 3D Model
Step 2: Assign Viewpoints and Images
New Viewpoint
go to viewpoint
assign image
go to viewpoint
assign image
a
each camera represents a viewpoint
b
assign image to each viewpoint
c
image as seen from current viewpoint

Figure 3: Defining Viewpoints: (a) add viewpoints by placing virtual cameras, (b) assigning images to each viewpoint, (c) verifying the texture appears from the assigned viewpoint.

[1]code available at `https://github.com/yunyi-zhu/lenticular-object`

4.2 Specifying Appearances for Each Viewpoint by Loading 3D Textures

After specifying a viewpoint, designers can define which appearance that the object should have from the viewpoint. In our system, since we work with 3D models, an appearance is represented by a 3D model texture. These textures need to be prepared upfront, i.e. either created by the designer or retrieved from a 3D model texture library. After preparing the texture, designers can assign a texture to a viewpoint by clicking the 'assign image' button (Figure 3b). Once a texture is added to a viewpoint in the panel, the corresponding camera is assigned to the texture (Figure 3c). Designers can verify the texture from a specific viewpoint by looking through it from the viewport of the camera.

4.3 Exporting the Fabrication Files: Generating Lenses and Color Patterns

Before fabricating the object, designers can preview the appearances from each viewpoint via ray tracing. To enable this, our system first creates the lenses on the object geometry and recomposes the imported 3D textures into the color pattern underneath each lens (Figure 4a). It then allows the designer to simulate the visual result from the different viewpoints using the raytracer (Figure 3b).

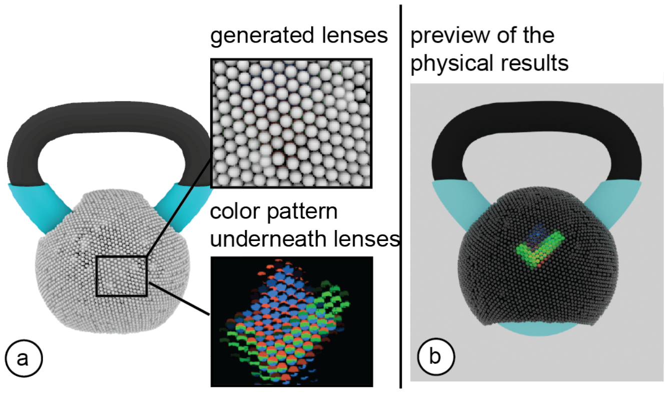
generated lenses
preview of the physical results
color pattern underneath lenses
a
b

Figure 4: Previewing the resulting object: (a) Generating the lens geometry and underlying color pattern. (b) Simulating the optical result using ray tracing to provide an accurate preview of the object before fabrication.

Clicking the 'export fabrication files' button generates the fabrication files for 3D printing, i.e. exports the object geometry and the lenses (.vrml) and an image file for the color pattern (.png). Designer can then upload the fabrication files to the slicer and after slicing finished, send them to the 3D printer. In the next section, we describe the slicing and fabrication process as well as the post-processing techniques we used to fabricate the objects with lenticular surfaces shown in this paper.

5 FABRICATION AND POSTPROCESSING

We first explain which 3D printer and printing materials we use, and then detail our fabrication process, which includes slicing the 3D model, 3D printing the object, and then post-processing the surface of the object. We then report on experiments investigating the print quality of different lens sizes as well as the color pattern resolution of our 3D printer.

5.1 Fabrication Process

3D Printer and Printing Materials: We fabricate our objects using a multi-material 3D printer that can print the object geometry, color patterns, and lenses in one pass. We fabricated all the objects in this paper using the Stratasys J55 3D printer [40], which uses polyjet technology. To print the lenses, we use the Stratasys polyjet material VeroUltraClear, a type of clear 3D printable acrylic [42]. To print the color patterns, we use materials from the VeroVivid family (VeroCyan-V, VeroMagenta-V, and VeroYellow-V) in combination with VeroPureWhite as the base material [41], which can create different colors on a per-voxel basis.

Slicing: Before printing, we slice the fabrication file that contains the object geometry and lenses (.vrml), which was exported from the 3D editor. To do this, we load the fabrication file into the slicer GrabCAD that can be used with the Stratasys 3D printers. After loading the .vrml file, we associate each part with the corresponding print materials in the 'print settings'. To further improve the quality of the printed lenses, we use the 'glossy' finish instead of the 'matte' finish in the 'print settings'. Clicking the 'save' button generates a file in .print file format, which we subsequently load into the 3D printer for fabrication.

3D Printing and Removing Support Material: After 3D printing, lenses printed at an angle are either fully or partially covered in support material. To remove the support material that the Stratasys 3D printers use (SUP710 [43]), we used an Objet Powerblast WaterJet machine, which washes off the support in less than 15 minutes.

Polishing the Lenses: Although we chose the 'glossy' 3D printing setting for the lenses, the lenses that are in contact with support material will still be matte since the 3D printer cannot use the glossy finish in those areas. To address this, we post-process the lenses by spraying 3 layers of varnish on the surface of the lenses.

Since the 3D printing process leads to imperfections in the lenses as well as in the color patterns, we ran a set of experiments to determine the print quality of different lens sizes and color pattern resolutions, which we report on in the next section.

5.2 Fabrication Quality of Different Lens Sizes

Our preliminary experiments showed that the quality of the fabricated lenses depends on several different parameters, such as the lens size, the orientation in which the lens is printed, and the post-processing technique applied to the lens after fabrication. We therefore fabricated lenses using different parameters and compared the physical result with the rendered view of the lenses to evaluate their quality.

Procedure: We generated four different lens sizes, ranging from a diameter of 2mm - 5mm in 1mm increment, using the lens geometry as determined in Section 6. Since the print orientation also impacts the optical quality, we 3D printed each lens size in five orientations (1) facing upwards, (2) facing 45° upwards, (3) facing sideways, (4) facing 45° downwards, (5) facing downwards.

After printing the lenses, we aligned the backplane of each lens onto an ink-jet printed checkerboard with a checker size of 500 microns (printer: Xerox Workcentre 7970, paper: HP Premium Plus Photo Paper | Soft Gloss). We took pictures from the top of the lenses with a camera (model: Canon T3i, focal length: 18mm, distance from lens top: 40cm). We then compared the photos with the rendered image of the same lens with the same checkerboard pattern, which shows the resulting image without any print imperfections (rendering engine: LuxCoreRender [31]).

To evaluate different post-processing methods, we compared four techniques: (1) no post-processing, (2) lenses coated with three layers of painting varnish spray (Liquitex), (3) lenses coated with three layers of clay gloss varnish spray (Krylon) and (4) lenses rubbed with baby oil (Johnson). For each technique, we fabricated a new copy of the lenses. To make sure that the post-processing effect is stable, we put the post-processed lenses in a room for 14 days before taking photos.

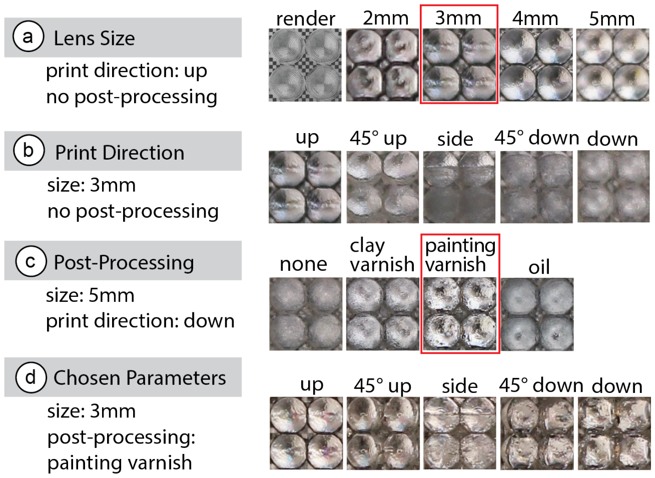

Figure 5: Fabricated lenses compared with rendered lenses. (a) Larger lenses have better fabrication accuracy, (b) lens quality varies among different printing directions, (c) clay varnish and painting varnish both improve the lens quality, (d) lens quality of our chosen parameters (3mm lenses with painting varnish).

Result: Figure 5 shows the photos of the lenses compared with the rendered image from the same viewpoint. For the lens sizes, larger lenses show an image that more closely resembles the rendered image than smaller lenses. Concerning the print orientation, lenses printed facing upwards result in the best lens surface quality because no support material is needed and thus all parts of the lens are printed with glossy finish. All other angles need support material in varying amounts and thus the lens quality decreases. From those lenses printed with support material the best print quality was 45° upwards, followed by lenses facing downwards and 45° downwards, and the worst print quality resulted from printing lenses sideways. For post-processing, both types of varnish improved the glossiness of the lens surface, with the painting varnish providing the best surface quality. In contrast, the baby oil made the lens glossy when rubbed, but the effect faded away after 14 days.

Which lens size to use is a trade-off between image quality and overall image resolution. While smaller lenses are of less print quality, they provide a higher resolution, which more closely represents the details of the input image. We found that for our application scenarios, 3mm lenses with post-processing preserved the important visual features of our input textures while providing sufficient visual quality. However, for one of our example applications (earpod case), we chose to use 2mm lenses to fit more lenses onto the otherwise small geometry.

5.3 Resolution of 3D Printable Color Pattern

We also investigated the 3D printing resolution of the color pattern, which forms the image spots underneath the lenses. If the printable color pattern resolution is too low, the image spots assigned to the different viewpoints merge together on the backplane of the lens causing the viewer to see the wrong image from the viewpoint. The color pattern resolution required depends on the size of the lens, i.e. larger lenses have larger image spots (see Section 6).

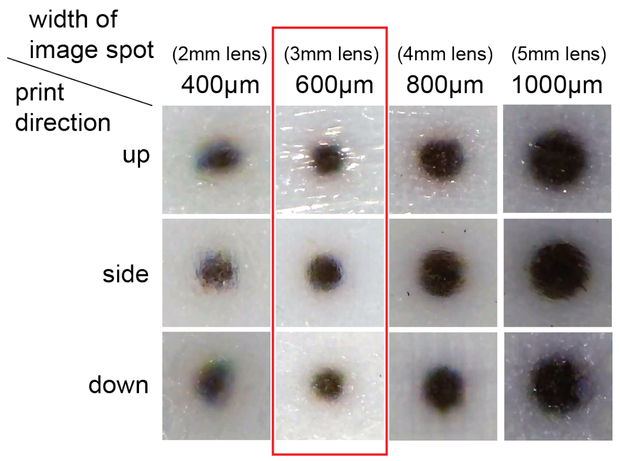

Figure 6: 3D printed dots of different sizes from different print directions. Dots that have a 600 micron diameter are the smallest that have good print quality.

Procedure: To determine whether the 3D printer's resolution can accurately fabricate the image spots for different lens sizes, we ran an experiment in which we 3D printed circle patterns representing the image spots of different lens sizes (2mm lens: 400 microns, 3mm lens: 600 microns, 4mm lens: 800 microns, 5mm lens: 1000 microns) and then analyzed their quality under a microscope. To account for different print orientations, we printed the circle patterns on the top, side and bottom surfaces of a cube with a side length equal to 10 times the circle's diameter. We then used a microscope (model: inskam-316) to take photos.

Result: As shown in Figure 6, the larger the printed circle pattern, i.e. image spot, the better the print quality. We found that the color pattern of 600 microns, which corresponds to the 3mm lenses, was the smallest that had a good print quality with sharp edges. Thus, with our color print resolution, lenses that are 3mm or larger can maintain the computed number of viewpoints with different images.

6 LENS GEOMETRY DESIGN

We next describe how we computed the lens geometry that represents the best trade-off between a large number of viewpoints and avoiding protrusion of the lenses from the object geometry). We then render an example lens with the resulting lens geometry to show that it supports the calculated number of viewpoints.

Optimal Lens Geometry: Figure 7a shows the three parameters that determine the geometry of a lens: (1) the diameter d_{base} of the lens, (2) the radius r of the top spherical surface of the lens that shapes its curvature, and (3) the height h_{base} of the substrate (the cylindrical base) of the lens. The overall lens geometry is determined by these three parameters, i.e. the ratio of $d_{base} : r : h_{base}$. We further define the width of the image spot underneath the lens as L as shown in Figure 7b.

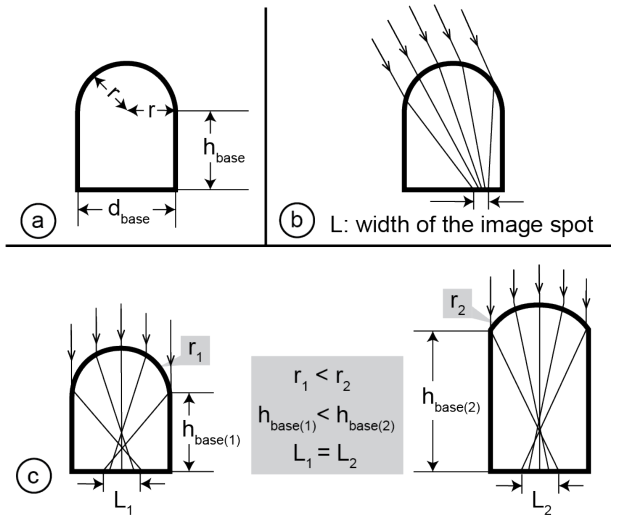

Figure 7: (a) Basic lens parameters: lens diameter d_{base}, radius r and substrate height h_{base} (b) image spot L: the area underneath the lens that can be seen from one specific viewing angle, (c) to achieve the same image spot size, a smaller radius (i.e., more curvy lens), requires a smaller height.

The lens geometry determines how many viewpoints each lens can support. The number of viewpoints is determined by how many image spots fit underneath each lens. There are several factors that determine the size of the image spots. First, for a given lens curvature, the lens substrate has a specific height at which the image spot is minimized. The more curvy the lens surface is, the smaller the substrate height when it achieves its local minimum (Figure 7c). Since the lenses are added to the object's surface, we want to minimize the height so that the lenses do not protrude too far out from the object geometry. We thus use the most curvy lens since it results in the smallest height and thus least protrusion from the surface.

The most curvy lens is created by using the smallest radius r, which is $\frac{d_{base}}{2}$.

With the radius r determined, we next find the lens substrate height h_{base} that minimizes the width of the image spot L so that the lens can hold the largest possible number of image spots. For this, we use different lens substrate heights h_{base} as input and plot the resulting image spot widths. For a given lens substrate height h_{base}, the image spot width L also depends on the viewing angle, i.e. if the viewer looks straight down onto the lens, the magnification is larger and the image spot smaller than when the viewer looks at it from the side. We therefore plot the upper bound of the image spot width for each given lens substrate height. The detailed computation is included in the Appendix and the result is shown in Figure 8a. We can see that the image spot width L is minimized when the substrate height $h_{base} = 0.5d_{base}$ with the image spot width being $L = 0.2d_{base}$.

In conclusion, the lens geometry that supports the highest number of viewpoints while protruding least from the object surface has a lens curvature of $r = 0.5d_{base}$ and a substrate height of $h_{base} = 0.5d_{base}$. Therefore, the lens geometry we use has a ratio of $d_{base} : r : h_{base} = 2 : 1 : 1$ (Figure 8b).

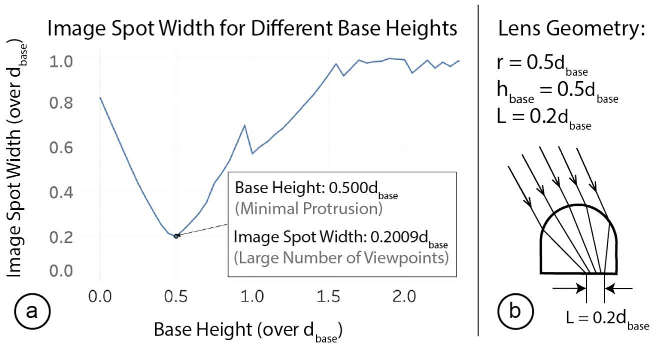

Figure 8: (a) Finding the base height that minimizes the image spot for $r = 0.5d_{base}$; (b) the resulting lens geometry is $d_{base} : r : h_{base} = 2 : 1 : 1$, which results in $L = 0.2d_{base}$.

Viewing Range: The lens geometry results in a viewing angle range of 83.6°, i.e. the viewer can see the correct image spot when they are no more than $46.8° = \frac{83.6°}{2}$ away from the upwards direction of the lens. We determined this value by tracing 20 parallel viewing rays from each viewing angle and determined the viewing angle valid when all viewing rays from a viewing angle were reflected onto the backplane where the image spot is supposed to be located (see Appendix).

Number of Viewpoints: As mentioned above, the width of an image spot from any viewing angle is at most $L = 0.2d_{base}$. To determine the resulting number of viewpoints, we calculated the maximum number of image spots that can fit into the lens backplane. Although the exact image spots can have non-circular shapes, L represents the upper bound of the image spot width across all viewing angles. We used the Wolfram Alpha calculator's circle packing function [52] and found that 19 image spots fit into the lens backplane. To verify that we can support 19 different viewpoints, we created a lens with 19 different image spots (Figure 9a) and simulated the appearances this lens creates from each of the 19 viewpoints using ray tracing. Figure 9b shows the result, i.e. that the lens is indeed capable of creating different appearances from 19 different viewpoints.

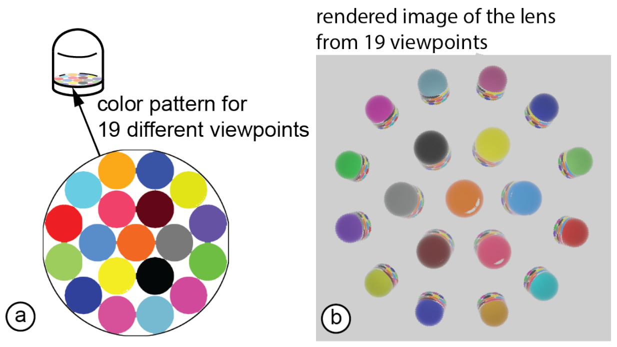
rendered image of the lens from 19 viewpoints
color pattern for 19 different viewpoints
a
b

Figure 9: (a) Our lens geometry allows packing 19 image spots onto the backplane of a lens and thus can support 19 viewpoints with different images; (b) ray traced results of the lens showing 19 different colors from different viewpoints.

To evaluate how many different viewpoints can be achieved considering the limitations of current fabrication technology, we fabricated lenses at different angles (facing up, 45° up, sideways, 45° down, down). We printed 19 lenses per angle with each lens containing a different number of viewpoints, ranging from 1 – 19 viewpoints. After fabrication, we took pictures from all visible viewpoints. Our results show that lenses printed facing upwards and downwards have the highest number of visible viewpoints, i.e. the upward printed lens can show up to 19 viewpoints and the downward printed lens can show up to 14 viewpoints. Other print orientations had smaller number of view points (i.e., 45° up: 12 viewpoints, 45° down: 9 viewpoints, sideways: 7 viewpoints).

7 IMPLEMENTATION

Our 3D editor is implemented in the 3D modeling software *Rhino3D* as a *Grasshopper* plugin. Our software pipeline first distributes the lenses across the surface of the 3D model. It then maps the pixels of the different input 3D color textures to the lens positions on the 3D model. Next, it computes for each lens how the color pixels of the different 3D textures should be arranged on the backplane of the lens to show the correct appearance for each viewpoint. Finally, it generates the fabrication files and exports them.

7.1 Distributing Lenses Across the 3D Model

Our goal when distributing the lenses across the 3D object surface is to pack them as closely together as possible while not colliding with each other. Two lenses are not colliding when the distance between the two lens centers is equal or greater than the diameter of the lenses. Since lenses are uniform circles, an efficient way to pack them is hexagonal packing [54]. Hexagonal packing can be achieved by first dividing the surface into equilateral triangles and then placing a lens at each corner of the triangle. Since lenses should not collide, we set the edge length of the equilateral triangle to the lens diameter.

To implement this, our system first converts the 3D model into a triangular mesh using the instant meshes open source library [21] (Figure 10a/b). Since instant meshes requires an .obj file as input, our system temporarily exports the 3D model from our editor and uses the exported file for the instant meshes conversion. In addition to the 3D model geometry in .obj format, instant meshes requires several parameters: first, instant meshes requires the target face count, i.e. the number of faces into which the mesh should be converted. Our system determines the number of faces by dividing the surface area of the 3D model by the area of the corresponding equilateral triangle. Next, instant meshes requires as input the rotational and position symmetry. Setting them to a value of '6' results in a hexagonal-directional field suitable for triangular packing. Once instant meshes finished the conversion, our system imports the converted triangular mesh back into our design tool. Our system then uses the corners of the triangles, which represent all the vertices of the mesh, as the centers for the lenses to place the lenses across the 3D surface (Figure 10c/d).

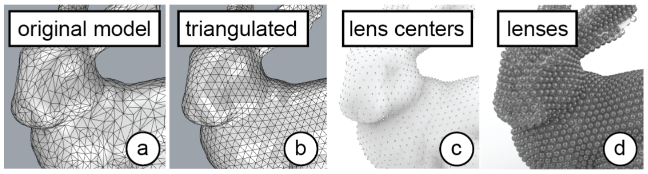

Figure 10: Placing the lenses: (a) original 3D model, (b) triangular mesh with equilateral triangles, (c) each triangle corner (vertex) is a location for a lens center, (d) lenses placed.

7.2 Mapping Color Pixels to Lens Locations

Once our system positioned the lenses across the 3D model's surface, it next maps the color pixels of each texture image to the lens positions on the 3D model (Figure 11a). To know which texture pixels belongs to which lens, our system uses UV mapping, which maps a vertex on a mesh to a 2D coordinate on the texture image. When a 3D object with texture mapping is imported, a list of vertex-to-UV coordinate mappings is already included in the 3D model. For points on the surface of the 3D model that are not vertices, our system can further compute their UV-coordinates in the texture image using their barycentric coordinates, i.e. by finding which face they are on and then interpolating them as the weighted average of the vertices on the face.

After computing all UV coordinates, our system maps each color pixel in the 3D texture to a lens. To do this, our system first finds the lens center's pixel location in the color texture using the UV mapping. Assuming that the input geometry has an even UV-mapping, i.e. the distance between two points on the texture is proportional to the geodesic distance between the source points on the object, our system then finds the pixel-radius of each lens on the texture by sampling a point on the circumference of the lens and then finding the distance between its corresponding pixel and the lens center's corresponding pixel. Therefore, every pixel that is within

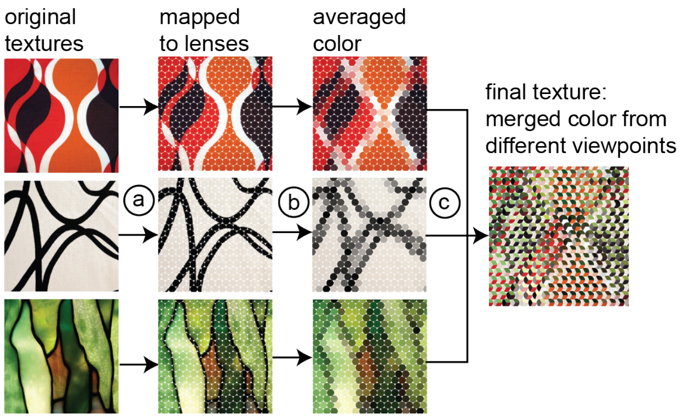

Figure 11: Mapping color pixels onto lenses for each 3D texture: (a) Find which pixels are underneath which lens for each texture. (b) For each texture, average the color underneath each lens. (c) Divide the space underneath each lens using ray-tracing and assign the averaged color of the corresponding texture for each lens.

the pixel-radius distance to the lens center's corresponding pixel are mapped to the lens.

Our system requires that all 3D textures have the same UV mapping. Thus, once the UV mapping is computed, we can reuse the assignment of color pixels on a 3D texture to a specific lens position on the object geometry for all viewpoints and their 3D color textures.

7.3 Averaging Color Pixels in each Image Spot

Next, our system averages the colors of all pixels in an image spot that belong to the same viewpoint (Figure 11b). We do not keep the individual color pixels in an image spot because when the viewer shifts the head slightly, they would see a different color pixel each time, resulting in unrelated patterns even with slight head movement. In addition, the image spot is very small due to the magnifying effect of the lens. Thus, printing all color pixels from the input texture into the small image spot would require a print resolution beyond what our 3D printer is capable of. The smallest pixel size that our 3D printer supports is 200 microns, which translates to only 7 pixels for the image area underneath a 3mm lens. Because of this, our system averages the color of all pixels in an image spot that belong to the same viewpoint.

7.4 Arranging Image Spots Underneath a Lens According to the Viewpoint

The previous processing step determined the average color assigned to each lens for each of the input textures. Next, our system arranges the average colors from the different input textures underneath each lens to show the correct color at each viewpoint (Figure 11c).

Before our system can distribute the image spots underneath the lens, it first has to determine which area underneath the lens is visible from which viewpoint. To accomplish this, our system casts rays from different viewpoints to the top surface of the lens. When the various rays hit different points on the top surface of the lens, they enter the lens at different angles and thus reflect off different positions at the bottom of the lens (our system approximates the back of the lens with a flat plane because of the small lens size). Since the color pattern is placed at the bottom of the lens, we know that the positions were the rays reflect off the backplane are visible from the specific viewpoint from which the ray was cast. We thus only keep the color of the texture that should be visible from this viewpoint and delete all other textures in this area.

7.5 Exporting the Fabrication Files

Finally, once the lenses are generated and the color pattern is correctly distributed across all lenses, our system exports the fabrication files, i.e. the geometry file (.vrml) with the lenses and the object geometry, and the color pattern image (.png) that is referenced in the .vrml file. To prepare for rendering, our system also generates a viewpoint file (.txt) that records the 3D coordinates of the viewpoints defined in the system, which the render environment uses as the locations for the render cameras. Our system renders the exported object using the raytracing plugin LuxCoreRenderer [31] in Blender [5].

8 APPLICATION SCENARIOS

Being able to design objects that look different from different viewpoints enables a wide range of novel applications in product design and HCI. In the following section, we highlight several application scenarios that focus on using 3D lenticular objects. In the scenarios, the objects demonstrate that our system can produce objects with: (1) different geometric complexities (e.g. single and doubly curved surfaces), (2) image complexities (e.g. patterns, symbols and text), and (3) number of distinct images (ranging from 2 to 5).

8.1 Guiding User's Body Poses

Our work enables the development of tangible objects that can guide users into placing or holding the object at a specific body pose. Figure 12 illustrates this at the example of a piece of exercise equipment, i.e. a kettlebell. This lenticular kettlebell guides the user into the correct exercise position, i.e. a front raise, by displaying if it is held at the correct height. The kettlebell displays a downwards arrow if it is held too high (Figure 12a), an upwards arrow if it is held too low (Figure 12c), and a check mark when the user holds it at the correct height (Figure 12b). This can prevent the user from exercising at the incorrect body pose, which has been shown to lead to injuries [12]. The kettlebell is an example of a lenticular object that displays symbols (up/down arrow, checkmark) printed with 3mm lenses on a doubly curved surface with three different viewing angles (low angle, eye level, high angle). In this example, rather than printing the entire kettlebell, we bought a 10lb kettlebell and augmented it with a 3D printed lenticular shell.

8.2 Dynamic Message According to User's Position

Our work also enables objects that display dynamic message or surface pattern according to the user's position. Figure 13 shows this at the example of a lampshade made for a user's bedroom that displays "Good Night" when the users lays in bed (i.e. looks at the lamp from the same height), and displays "Good Day" when the

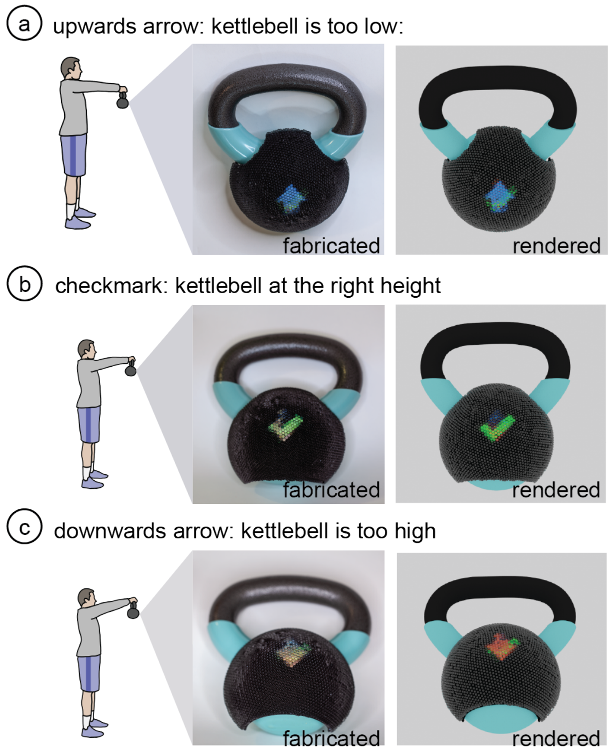
a
upwards arrow: kettlebell is too low:
fabricated
rendered
b
checkmark: kettlebell at the right height
fabricated
rendered
c
downwards arrow: kettlebell is too high
fabricated
rendered

Figure 12: Pose Guidance: this kettlebell guides the user to a correct front raise pose by displaying arrows when the kettlebell is too high or too low and a check mark when the kettlebell is at the right height.

user looks at it while sitting up in bed (i.e. looks at the lamp from above). This application is an example of how lenticular objects created with our system can display text messages ('Good Night', 'Good Day'). The lamp was printed with 3mm lenses on a doubly curved surface with two different viewing angles (same height as the lamp, above the lamp).

8.3 Dynamic Visual Effects for Product Design

When a user plays with, passes by or moves around an object, they change their viewpoint towards the object. We leverage this to create dynamic visual effects on products as part of the object's product design. Figure 14 shows this at the example of an earpod case that transitions between different colors when it changes its viewing angle with respect to the viewer. This application is an example that displays a pattern (colorful stripes) printed with 2mm lenses on a surface that is flat on the front and doubly curved in the corners with five different viewpoints (one for the yellow, green, cyan, purple, magenta stripe patterns). The discrepancy between the fabricated and rendered earpod case in the right-most viewpoint results from fabrication limitations, i.e. since the earpod case was printed with 2mm lenses the printing resolution for the color pattern is not high enough to support the magenta sections.

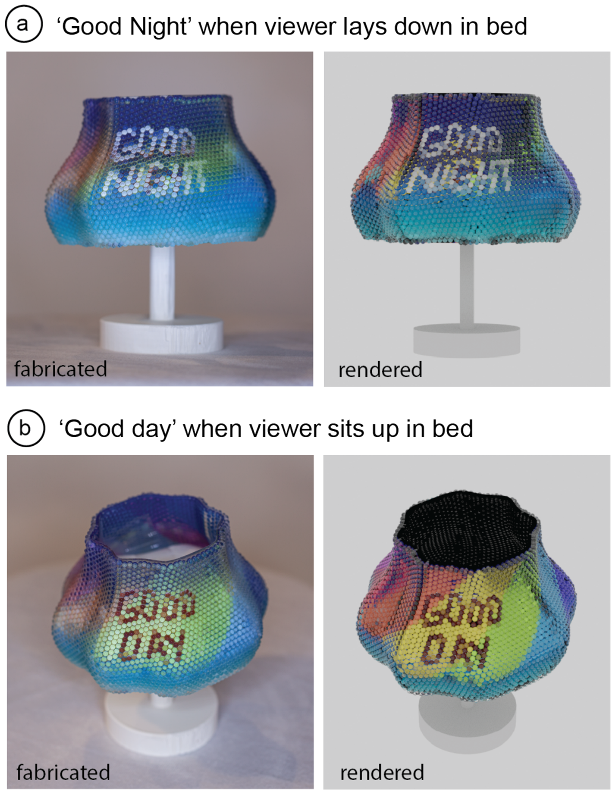
a 'Good Night' when viewer lays down in bed
fabricated
rendered
b 'Good day' when viewer sits up in bed
fabricated
rendered

Figure 13: This lampshade displays different greeting messages when the users looks at it from different heights (laying down in bed vs. sitting up in bed).

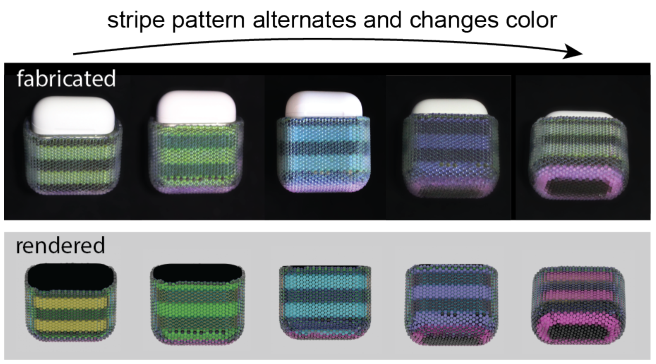

Figure 14: This earpod case flickers between five alternating bright-dark stripe patterns of different colors as the user handles it, creating unique aesthetics effects as part of the object's product design.

8.4 Images Only Seen by Specific Users

Different viewers tend to have different viewing angles on the same object. We can use this to create appearances that are only accessible by specific users. Figure 15 shows this at the example of a visual design that is only visible from the user's viewing angle. When the user looks down on their pair of shoes, the pair of shoes display a 'Cheer Up' message, however, other people cannot see the message and instead see a plain shoe design. This application is an example of how our system enables displaying a text message. The pair of shoes was printed with 3mm lenses on a doubly curved surface with two different viewpoints (regular shoe design [47] from the side and 'cheer up' message from the top). The entire shoe including the lenticular lenses was printed in one piece.

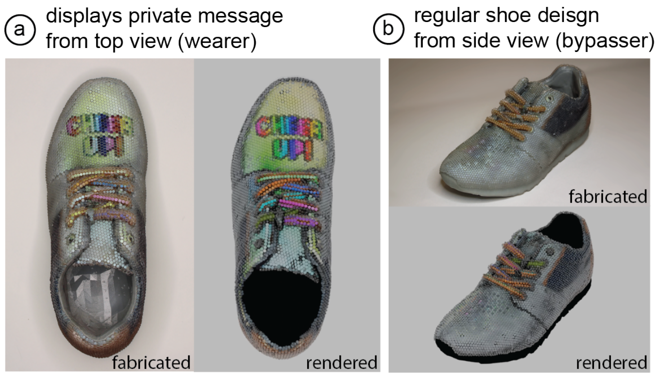
a displays private message from top view (wearer)
b regular shoe deisgn from side view (bypasser)
fabricated
rendered
fabricated
rendered

Figure 15: Visual designs only visible from the user's viewing angle: (a) A pair of shoes that shows a supportive message to the wearer looking down on the shoes, which (b) other people looking from the side at the shoes cannot see.

In summary, our four application scenarios demonstrate that our system can create lenticular objects with different geometric and image complexities as well as different number of viewpoints. In the next section, we will discuss limitations and future work of our system.

9 DISCUSSION AND LIMITATIONS

Our current implementation is subject to several limitations, which can be addressed in future work:

Impact of Lenses on Object Geometry and Surface Haptics: In our current work, we place the lenses on top of the surface of the 3D model, which extends the geometry by a lens layer of 3mm. While a layer of such lenses may not be an issue for larger objects, smaller objects with thin geometries will substantially increase in thickness. For future work, we plan to integrate the lens substrate into the object geometry to minimize protrusion from the surface. In addition, covering objects with lenses changes the tactile qualities of the object. We hope that with future developments in 3D printing, it will become possible to print lenses that are small enough that they no longer change the surface haptics. This may also address another challenge, which is that lenses that protrude from the surface can collide when geometries are highly curved, which can happen in our system but would no longer occur when lenses are smaller.

Impact of Non-Uniform UV and UV Seams on Color Pattern: As we explained in the implementation, we assume that the input geometry has a uniform UV texture mapping. If the UV texture mapping is uniform, the circular lens geometry refers to a circular area on the input texture, thus the mapping in our system creates the correct result. If the UV mapping is non-uniform, the circular backplanes of the lenses are mapped to non-circular areas on the input texture and thus assuming a circular-mapping may result in lenses being assigned to the wrong color pixels. This can be resolved by sampling a large number of points on the circumference of the lens and mapping them to the input texture to accurately represent the non-uniform mapping. Another problem that can occur when performing the UV mapping is that the lens center and the sampled point on the lens circumference may be on opposite sides of the UV texture when the lens is located on a seam. This causes the mapped radius to be much larger than the actual radius. We detect such cases by setting an upper bound for the mapped radius and subsequently do not assign a color. For future work, we plan to perform UV mapping for all points inside the lens's backplane and not just the circumference so that all of the points are mapped to the correct location on the input texture.

Showing Optical Limitations in User Interface: Our current user interface supports the designer in specifying viewing angles, assigning color textures, and generating fabrication files for the lenticular object. However, our current system relies on the designer to check in the rendering environment whether the generated result matches their specified design. In cases where the defined viewpoints exceed optical limitations, e.g. a viewpoint is outside the viewing range of a lens or two viewpoints are too close together, there can be cross talk between different images. In future iterations, we will extend our user interface to only support those viewing angles that are valid.

Fabricating Lenticular Objects on Consumer Level 3D Printers: While we fabricated our lenticular objects on a high-end Stratasys J55 3D printer, it may also be possible to fabricate lenticular objects on low-cost 3D printers, albeit at the expense of manual assembly. For instance, Formlabs 3D printers [10] have been shown to be capable of 3D printing lenses, while the Da Vinci 3D Printer [53]) can print the color patterns in full color. Thus, by printing the lenses as a separate shell, it may be possible to fabricate lenticular objects on consumer level 3D printers.

Dynamic Lens Sizes and Other Types of Lenses: In our current system, all lenses on the object's surface have the same size. While this is sufficient for scenarios that requires an even distribution of viewpoints, some scenarios might benefit from having lenses of different sizes located on different parts of the object's surface. For future work, we plan to improve our algorithm to support dynamic lens sizes based on the required image resolution and number of viewpoints. In addition, while in this paper, we have focused on spherical lenses, other lens types, such as cylindrical lenses, exist. A benefit of cylindrical lenses is that they have higher spatial resolution while having the drawback that different viewpoints can only change along one direction. Figure 16 shows an application of this that we created manually: a minimalist product that only shows text instructions when it is held at eyesight and is otherwise clear. Compared to the text in the shoe application, the text resolution in this minimalist design example is higher. Since the application only needs to support one direction of movement, cylindrical lenses are sufficient to support this transition.

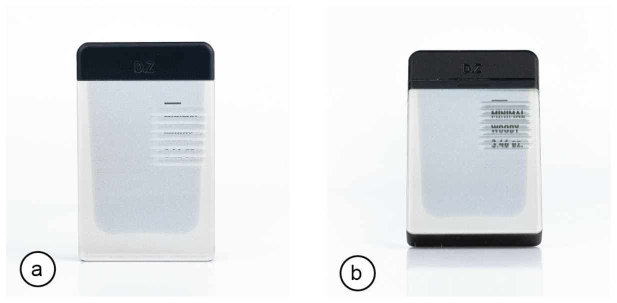

Figure 16: Cylindrical Lenses: A container that shows information only when needed: (a) the container does not show any text when looking from an upper angle, (b) the container shows informative texts when held at eye level.

10 CONCLUSION

In this paper, we presented a method to extend lenticular displays to 3D object geometries, enabling 3D objects to look different from different viewing angles. We showed how our design tool supports designers in creating 3D lenticular lens displays by enabling them to define viewpoints and assign corresponding color textures. We also provided information on the fabrication process and evaluated the 3D printing quality of different lens sizes, the print resolution of the color pattern, as well as the effects of different post-processing techniques. We discussed our implementation pipeline that automatically computes the lens distribution across the object geometry and assigns the correct color pixels to each lens to achieve the desired appearance from each viewing angle. We then demonstrated various example applications that highlight different geometric and image complexities as well as different numbers of viewpoints. For future work, we plan to further improve our implementation by integrating the lens substrate into the geometry of the object, improving the UV-mapping algorithm to cover non-uniform UV mappings and seams of the UV textures, and to extend our 3D editor to consider optimal limitations. In addition, we plan to explore the use of dynamic lens sizes and other types of lenses, such as cylindrical lenses.

ACKNOWLEDGMENTS

We thank Neil Gershenfeld for granting us access to the Stratasys J55 printer in his lab and Tom Lutz for 3D printing our models on the 3D printer. This paper is based upon work supported by the National Science Foundation under Grant No. 1716413.

REFERENCES

[1] Maneesh Agrawala, Andrew C. Beers, Ian McDowall, Bernd Fröhlich, Mark Bolas, and Pat Hanrahan. 1997. The Two-User Responsive Workbench: Support for Collaboration through Individual Views of a Shared Space. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. 327–332. https://doi.org/10.1145/258734.258875
[2] Jorge Alamán, Raquel Alicante, Jose Ignacio Peña, and Carlos Sánchez-Somolinos. 2016. Inkjet Printing of Functional Materials for Optical and Photonic Applications. *Materials* 9, 11 (2016). https://doi.org/10.3390/ma9110910
[3] Patrick Baudisch, Torsten Becker, and Frederik Rudeck. 2010. Lumino: Tangible Blocks for Tabletop Computers Based on Glass Fiber Bundles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. 1165–1174. https://doi.org/10.1145/1753326.1753500

[4] Patrick Baudisch and Stefanie Mueller. 2017. Personal Fabrication. *Foundations and Trends® in Human–Computer Interaction* 10, 3–4 (2017), 165–293. https://doi.org/10.1561/1100000055

[5] Blender. 2021. *Blender Webpage.* https://www.blender.org/

[6] Eric Brockmeyer, Ivan Poupyrev, and Scott Hudson. 2013. PAPILLON: Designing Curved Display Surfaces with Printed Optics. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13).* 457–462. https://doi.org/10.1145/2501988.2502027

[7] Churchill Container. 2021. *Lenticular Cups.* https://www.churchillcontainer.com/lenticular-cups/

[8] Neil Dodgson. 2005. Autostereoscopic 3D Displays. *Computer* 38, 8 (Aug. 2005), 31–36.

[9] Gregory Ferenstein. 2013. *An Anti-Abuse Ad with a Secret Message Only Children Can See.* https://techcrunch.com/2013/05/06/an-anti-abuse-ad-with-a-secret-message-only-children-can-see/

[10] FormLabs. 2021. *FormLabs Webpage.* https://formlabs.com/

[11] Masahiro Furukawa, Hiromi Yoshikawa, Taku Hachisu, Shogo Fukushima, and Hiroyuki Kajimoto. 2011. "Vection Field" for Pedestrian Traffic Control. In *Proceedings of the 2nd Augmented Human International Conference* (Tokyo, Japan) *(AH '11).* Association for Computing Machinery, New York, NY, USA, Article 19. https://doi.org/10.1145/1959826.1959845

[12] M Gallagher. 1996. Ten most common causes of training injury. *Muscle & Fitness* (1996).

[13] Daniel Gotsch, Xujing Zhang, Juan Pablo Carrascal, and Roel Vertegaal. 2016. HoloFlex: A Flexible Light-Field Smartphone with a Microlens Array and a P-OLED Touchscreen. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16).* Association for Computing Machinery, New York, NY, USA, 11. https://doi.org/10.1145/2984511.2984524

[14] Chris Harrison and Scott E. Hudson. 2011. A New Angle on Cheap LCDs: Making Positive Use of Optical Distortion. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11).* 537–540. https://doi.org/10.1145/2047196.2047266

[15] Sunao Hashimoto, Ryohei Suzuki, Youichi Kamiyama, Masahiko Inami, and Takeo Igarashi. 2013. LightCloth: Senseable Illuminating Optical Fiber Cloth for Creating Interactive Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13).* 603–606. https://doi.org/10.1145/2470654.2470739

[16] Ryuji Hirayama, Hirotaka Nakayama, Atushi Shiraki, Takashi Kakue, Tomoyoshi Shimobaba, and Tomoyoshi Ito. 2019. Projection of multiple directional images on a volume structure with refractive surfaces. *Opt. Express* 27 (Sep 2019), 27637–27648. https://doi.org/10.1364/OE.27.027637

[17] Matthew Hirsch, Daniel Leithinger, and Thomas Baran. 2016. Lumii: DIY Light Field Prints. In *ACM SIGGRAPH 2016 Studio (SIGGRAPH '16).* Article 3. https://doi.org/10.1145/2929484.2929487

[18] Kai-Wen Hsiao, Jia-Bin Huang, and Hung-Kuo Chu. 2018. Multi-View Wire Art. *ACM Trans. Graph.* 37, 6, Article 242 (Dec. 2018). https://doi.org/10.1145/3272127.3275070

[19] Tunglu Hung. 2000. *Fan Lihua.* https://library-artstor-org/asset/ASIAARTIG_10313977571

[20] Frederic E Ives. 1903. Parallax stereogram and process of making same.

[21] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-Aligned Meshes. *ACM Transactions on Graphics* 34, 6, Article 189 (2015).

[22] Chengang Ji, Guodong Zhu, Cheng Zhang, Sunghyun Nam, Qiaochu Li, Liangping Xia, Weiguo Zhang, Debasish Banerjee, and Lingjie Jay Guo. 2017. Lenticular-Lens-Based Colored Antiglare Dashboard Surfaces. *Advanced Materials Technologies* 2, 1 (2017), 1600177. https://doi.org/10.1002/admt.201600177

[23] Linas Jonušauskas, Darius Gailevičius, Lina Mikoliūnaitė, Danas Sakalauskas, Simas Šakirzanovas, Saulius Juodkazis, and Mangirdas Malinauskas. 2017. Optically Clear and Resilient Free-Form µ-Optics 3D-Printed via Ultrafast Laser Lithography. *Materials* 10, 1 (2017), 12. https://doi.org/10.3390/ma10010012

[24] L. Jonusauskas, A. Žukauskas, P. Danilevicius, and M. Malinauskas. 2013. abrication, replication, and characterization of microlenses for optofluidic applications. In *Advanced Fabrication Technologies for Micro/Nano Optics and Photonics VI*, Vol. 8613. International Society for Optics and Photonics, SPIE, 210–217. https://doi.org/10.1117/12.2001061

[25] Hsin-Liu (Cindy) Kao, Meng Yee (Michael) Chuah, Michael Degen, Jason Tucker, and Hiroshi Ishii. 2014. LightBundle: Grasping Light through Plant-Inspired Interactions *(CHI EA '14).* 1849–1854. https://doi.org/10.1145/2559206.2581194

[26] Douglas Lanman and David Luebke. 2013. Near-Eye Light Field Displays. In *ACM SIGGRAPH 2013 Emerging Technologies (SIGGRAPH '13).* Article 11. https://doi.org/10.1145/2503368.2503379

[27] Douglas Lanman, Gordon Wetzstein, Matthew Hirsch, Wolfgang Heidrich, and Ramesh Raskar. 2011. Polarization Fields: Dynamic Light Field Display Using Multi-Layer LCDs. *ACM Transactions on Graphics* 30, 6 (2011), 1–10.

[28] Johnny C. Lee, Scott E. Hudson, and Edward Tse. 2008. Foldable Interactive Displays *(UIST '08).* 287–290. https://doi.org/10.1145/1449715.1449763

[29] Rong-Hao Liang, Chao Shen, Yu-Chien Chan, Guan-Ting Chou, Liwei Chan, De-Nian Yang, Mike Y. Chen, and Bing-Yu Chen. 2015. WonderLens: Optical Lenses and Mirrors for Tangible Interactions on Printed Paper. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15).* 1281–1284. https://doi.org/10.1145/2702123.2702434

[30] Gabriel Lippmann. 1908. La Photographie Integrale. *Comptes-Rendus Academie des Science* 146 (1908), 446–451.

[31] LuxCore. 2021. *LuxCore Renderer Webpage.* https://luxcorerender.org/

[32] Wojciech Matusik, Clifton Forlines, and Hanspeter Pfister. 2008. Multiview User Interfaces with an Automultiscopic Display. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Napoli, Italy) *(AVI '08).* Association for Computing Machinery, New York, NY, USA, 363–366. https://doi.org/10.1145/1385569.1385634

[33] Kosuke Nakajima, Yuichi Itoh, Takayuki Tsukitani, Kazuyuki Fujita, Kazuki Takashima, Yoshifumi Kitamura, and Fumio Kishino. 2011. FuSA Touch Display: A Furry and Scalable Multi-Touch Display *(ITS '11).* Association for Computing Machinery, New York, NY, USA, 35–44. https://doi.org/10.1145/2076354.2076361

[34] Marios Papas, Thomas Houit, Derek Nowrouzezahrai, Markus Gross, and Wojciech Jarosz. 2012. The Magic Lens: Refractive Steganography. *ACM Trans. Graph.* 31, 6, Article 186 (2012). https://doi.org/10.1145/2366145.2366205

[35] Michal Piovarči, Michael Wessely, Michał Jagielski, Marc Alexa, Wojciech Matusik, and Piotr Didyk. 2017. Directional Screens. In *Proceedings of the 1st Annual ACM Symposium on Computational Fabrication (SCF '17).* Article 1. https://doi.org/10.1145/3083157.3083162

[36] Petar Pjanic and Roger D. Hersch. 2015. Color Changing Effects with Anisotropic Halftone Prints on Metal. *ACM Trans. Graph.* 34, Article 167 (Oct. 2015). https://doi.org/10.1145/2816795.2818083

[37] Joshua Ratcliff, Alexey Supikov, Santiago Alfaro, and Ronald Azuma. 2020. ThinVR: VR Displays with Wide FOV in a Compact Form Factor. In *ACM SIGGRAPH 2020 Emerging Technologies (SIGGRAPH '20).* Article 5. https://doi.org/10.1145/3388534.3407302

[38] Frederik Rudeck and Patrick Baudisch. 2012. Rock-Paper-Fibers: Bringing Physical Affordance to Mobile Touch Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12).* 1929–1932. https://doi.org/10.1145/2207676.2208334

[39] Kaisei Sakurai, Yoshinori Dobashi, Kei Iwasaki, and Tomoyuki Nishita. 2018. Fabricating Reflectors for Displaying Multiple Images. *ACM Trans. Graph.* 37, 4, Article 158 (2018). https://doi.org/10.1145/3197517.3201400

[40] Stratasys. 2021. *Stratasys J55 Digital Anatomy 3D Printer.* https://www.stratasys.com/3d-printers/j55

[41] Stratasys. 2021. *Stratasys Vero.* https://www.stratasys.com/materials/search/vero

[42] Stratasys. 2021. *Stratasys VeroUltraClear.* https://www.stratasys.com/materials/search/veroultraclear

[43] Stratasys. 2021. *SUP710 Support Material.* https://www.stratasys.com/explore/blog/sup710-support-material-on-the-j55

[44] Kei Sugiyama and Koji Tsukada. 2019. Proposal of Lens Shaping Method Using UV Printer. In *The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19).* 125–127. https://doi.org/10.1145/3332167.3356894

[45] Yuichiro Takeuchi, Shunichi Suwa, and Kunihiko Nagamine. 2016. AnyLight: Programmable Ambient Illumination via Computational Light Fields. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces (ISS '16).* 39–48. https://doi.org/10.1145/2992154.2992188

[46] James Tompkin, Simon Heinzle, Jan Kautz, and Wojciech Matusik. 2013. Content-Adaptive Lenticular Prints. *ACM Trans. Graph.* 32, 4, Article 133 (2013). https://doi.org/10.1145/2461912.2462011

[47] CG Trader. 2021. *Sports Shoe 3D Model from CGTrader.* https://www.cgtrader.com/free-3d-models/sports/equipment/shoes-96a692c9-3e05-4d19-9162-6eab9495482b

[48] Walgreens. 2020. *Lenticular Prints.* https://photo.walgreens.com/store/lenticular-print-details

[49] Tim Weyrich, Pieter Peers, Wojciech Matusik, and Szymon Rusinkiewicz. 2009. Fabricating Microgeometry for Custom Surface Reflectance. *ACM Trans. Graph.* 28, 3, Article 32 (July 2009). https://doi.org/10.1145/1531326.1531338

[50] Karl Willis, Eric Brockmeyer, Scott Hudson, and Ivan Poupyrev. 2012. Printed Optics: 3D Printing of Embedded Optical Elements for Interactive Devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12).* 589–598. https://doi.org/10.1145/2380116.2380190

[51] Raphael Wimmer. 2010. FlyEye: Grasp-Sensitive Surfaces Using Optical Fiber. In *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '10).* 245–248. https://doi.org/10.1145/1709886.1709935

[52] WolframAlpha. 2021. *Wolfram Alpha Circle Packing.* https://www.wolframalpha.com/input/?i=circle+packing

[53] XYZPrinting. 2021. *Da Vinci 3D Printer.* https://www.xyzprinting.com/en-US/product/da-vinci-color

[54] Chuanming Zong. 2014. Packing, covering and tiling in two-dimensional spaces. *Expositiones Mathematicae* 32, 4 (2014), 297–364.

A FINDING LARGEST IMAGE SPOT FOR A LENS GEOMETRY

To achieve the maximum number of supported viewpoints, we need to determine the lens geometry that minimizes the width of the image spots on the backplane of the lens L. Minimizing L allows to pack more projected image spots from different viewpoints onto the lens backplane. To accomplish this, we need to optimize for the lens parameters d_{base} and h_{base} (while fixing $r = \frac{d_{base}}{2}$).

To accomplish this, we first compute the *largest image spot* $L_{max}(d_{base}, r = \frac{d_{base}}{2}, h_{base})$ for a given lens geometry $d, r = \frac{d_{base}}{2}$, h_{base} at an arbitrary incident angle $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$. By minimizing over all L_{max}, we can find the optimal lens geometry that supports the largest amount of viewpoints. We first show the steps of tracing one ray from its intersection with the lens to its projection on the lens backplane, then leverage to a set of parallel rays that simulate a viewpoint, and finally find the largest image spot for the lens parameters d_{base}, $r = \frac{d_{base}}{2}$ and h_{base} over all sets of parallel rays.

A.1 Trace One Ray

To find the image spot of a given viewpoint, we first need to determine how to trace a specific ray on a given lens geometry. When one input ray hits the top surface of the lens, it is first refracted and it then traverses the inside of the lens until it hits the lens backplane.

To represent one ray that intersects with a lens, we define two parameters: (1) the direction of the ray with angle θ (radians) from the vertical axis, and (2) the horizontal coordinate $x_0 \in (-\frac{d_{base}}{2}, \frac{d_{base}}{2})$ (millimeters) of its intersection point with the lens's top surface, with the center axis of the lens being the zero coordinate (Figure 17a). To represent where the refracted ray hits the backplane, we use the horizontal coordinate x_r of the intersection point between the refracted ray and the lens's backplane (Figure 17a). Our goal is to find x_r with given x_0, θ and lens geometry parameters.

To predict the behavior of the ray, we first find the incident angle θ_{in}: the angle between the incoming ray and the normal of the lens surface at the incident point. In Figure 17b, we show that $\theta_{in} = \theta + arcsin(\frac{x}{r})$. we then use the refraction equation $sin(\theta_{in}) \cdot n_{air} = sin(\theta_{out}) \cdot n_{lens}$ to compute the refracted angle θ_{out}. Since we approximate $n_{air} = 1$ and we know that $n_{lens} = 1.52$, we can compute that $\theta_{out} = arcsin(sin(\theta + arcsin(\frac{x_0}{r})) \cdot \frac{1}{1.52})$.

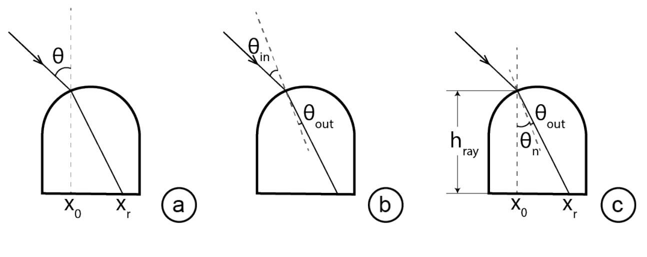

Figure 17: (a) A ray is defined with its direction θ and its intersection coordinate x_0 with the lens surface; the goal is to find the coordinate x_r where the refracted ray intersects with the lens's backplane. (b) Compute the incident θ_{in} and refracted angle θ_{out}. (c) Find x_r with θ_{out}.

Step 2: Find x_r using θ_{out}: As shown in Figure 17c, we can find x_r by first finding the distance between x_0 and x_r using trigonometry. In particular, $x_r - x_0 = h_{ray} \tan(\theta_{out} + \theta_n)$.

Including lens geometry parameters, we compute that $h_{ray} = h_{base} - \sqrt{r^2 - (\frac{d_{base}}{2})^2} + \sqrt{r^2 - x_0^2}$ and $\theta_n = \arcsin \frac{x_0}{r}$

Combining $h_{ray}, \theta_n, \theta_{out}$ together, we compute that

$$
\begin{aligned}
&x_r(x_0, \theta, d_{base}, r, h_{base}) \\
&\quad = x_0 + h_{ray} \tan(\theta_{out} + \theta_n) \\
&\quad = x_0 + \tan\Bigg[\arcsin\left(\sin\left(\theta + \arcsin\left(\frac{x_0}{r}\right)\right) \cdot \frac{1}{1.52}\right) \\
&\qquad + \arcsin\left(\frac{x_0}{r}\right)\Bigg) \cdot \left(h_{base} - \sqrt{r^2 - \left(\frac{d}{2}\right)^2} + \sqrt{r^2 - x_0^2}\right)\Bigg]
\end{aligned}
$$

The value of x_r is then clipped between $[-\frac{d_{base}}{2}, \frac{d_{base}}{2}]$ so that it only considers the rays that hit the backplane.

A.2 Width of the Image Spot from a Viewpoint

As explained in Figure 7b, the width of the image spot is the range of x_r resulting from all viewing rays generated from one viewpoint that intersect with the entire lens. In our model, we approximate input viewing rays from one viewpoint as a group of parallel rays with the direction from the viewpoint to the center of the lens. We can perform this approximation because the size of the lens is relatively small compared to the distance between the viewpoint and the lens (typically larger than 10 centimeters).

We calculate the width of the image spot for a single viewpoint whose viewing angle is θ. The viewing rays towards the lens are approximated as a set of rays that cover the entire surface of the lens with angle θ and horizontal coordinate $x_0 \in (-\frac{d_{base}}{2}, \frac{d_{base}}{2})$ from the surface center of the lens. The width of the image spot for a certain viewing angle θ is the range of x_r across all input x_0. In other words,

$$
\begin{aligned}
L(\theta, d_{base}, r, h_{base}) = &\max_{x_0 \in (-\frac{d_{base}}{2}, \frac{d_{base}}{2})} x_r(x_0, \theta, d_{base}, r, h_{base}) \\
&- \min_{x_0 \in (-\frac{d_{base}}{2}, \frac{d_{base}}{2})} x_r(x_0, \theta, d_{base}, r, h_{base})
\end{aligned}
$$

A.3 Largest Image Spot Size of a Lens Geometry

In the previous step, we determined the width of the image spot from one specific viewpoint to a given lens geometry. Next, we need to consider the widths of image spots generated by all possible viewpoints (i.e. for all $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$. To account for the worst case scenario, we find the maximum of all L computed and name it L_{max}. Note that for simplicity, L_{max} is also denoted as L in the main text as the width of the image spot for a given lens geometry.

In conclusion, the upper bound of widths of the image spots from all viewpoints is

$$
L_{max}(d_{base}, r, h_{base}) = \max_{\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})} (L(\theta, d_{base}, r, h_{base}))
$$