

Generation of surface for producing image using caustics

L.Jiang, Z.Jiang

Abstract—A continuous surface which generates a given gray-scale image when illuminated by parallel light is computed via caustics and manufactured. The test image and the lens surface were first divided to $N \times N$ grid, where N is determined by the resolution of the test image in pixels ($N = 500$ in the first trial). The lens cells were morphed with gradient descend such that each cell (i,j) has area proportional to the brightness of image pixel (i,j) . The heightmap of the lens surface was then found by applying Snell's law and small angle approximation. Morphing the grid and finding heightmap both rely on solving Poisson's equation in Python. A simulation was built to test the output surface, and test images were mostly reproduced at $N = 1000$. The surface is then manufactured using acrylic by CNC mill, and tested under parallel light.

I. INTRODUCTION

A given gray-scale image can be produced by reflection or refraction of light rays through curved surfaces or objects. Some methods had been used to obtain such surfaces in computing graphics, for example, reflectance field[1], refractive steganography[2], subsurface scattering[3], and holography[4].

Caustics provides satisfactory illumination effect among these methods, with the setup shown in figure 1. The shape of the transparent material, or specifically the refractive surface, needs to calculated such that it produces the expected caustic image under parallel and normal incident light. Methods proposed by Finckh et al.[5], Papas et al.[6] and Yue et al[7]. are restricted by either complexity of patterns or continuity of surfaces. However, Yue et al.[8] suggests a different approach using Poisson's equation, which yields continuous surfaces and highly reproduced patterns. Matt Ferraro[9] simplified the above method using Julia without significant loss of outcome image quality. Our aim is to reproduce this result using Python.

II. METHOD

The method described by Yue et al.[8] consists of two steps. First, the corresponding relationship between incident light and the resultant image is determined. Then, the refractive surface is computed.

A. Morphing lens grid

As demonstrated in Figure 2, for a sample image with $N \times N$ pixels, each pixel is assigned an index (i, j) , which does not change over the course of calculations. Since pixels have the same size, the (u, v) coordinate system is essentially a

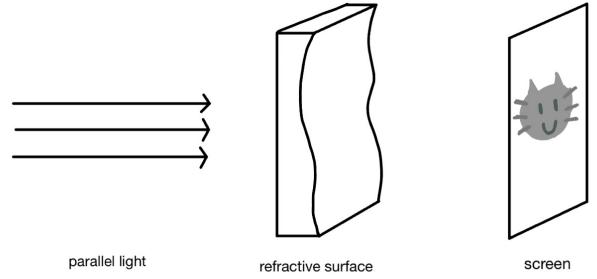


Fig. 1. Illustration of display system. Parallel light from either sun or projector, perpendicularly incident on a transparent refractive surface, results in caustic image on the projection screen.

uniform square grid.

Next, the refractive surface is also divided into $N \times N$ grids. The index of a particular cell is determined by the index of point on its top left corner, as shown in figure 2. As such, image pixel (i, j) corresponds to lens cell (i, j) . Initially, the grids on refractive plane with coordinates (x, y) will also forms a uniform mesh, but this will change as individual points moves through morphing. Consequentially, the area of lens cell (i, j) can be morphed to be proportional to the light intensity of pixel (i, j) on the image.

Mathematically, this process can be thought as a mapping of the (u, v) coordinate system on the image to the (x, y) coordinate system on the refractive surface.

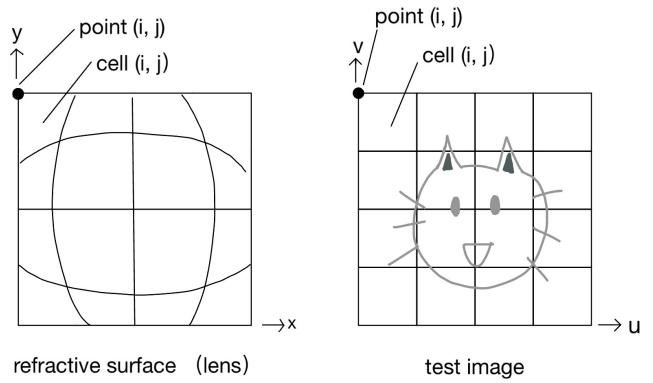


Fig. 2. Corresponding relationship of cells and positions. Each cell (i, j) on the (u, v) plane of the test image corresponds to a cell (i, j) on the (x, y) plane of the refractive surface.

Here, an assumption is made such that all light incident on lens cell (i, j) is refracted to pixel (i, j) , so no light is lost. Thus, the following must be true,

$$\frac{A(i, j)}{\Sigma A} = \frac{b(i, j)}{\Sigma b} \quad (1)$$

where $A(i, j)$ is the area of cell (i, j) on lens, and $b(i, j)$ indicates the brightness or light intensity of pixel (i, j) on image. Since the refractive surface is divided equally at the beginning, so the cells need to expand or shrink. A scalar field L is used to determine how much the cells are different from the expected value.

$$L = \frac{b(i, j)}{\Sigma b} - \frac{A(i, j)}{\Sigma A} \quad (2)$$

If $L(i, j) > 0$ at a point $p(x, y)$, it means the cell is too small and needs to expand. The magnitude of scalar field L is expected to decrease through mapping, so the points can be updated as $\frac{d\vec{p}}{dt} = -\nabla L$, where ∇L is the gradient of L (a vector field points to increasing direction) and t is a virtual time defined to represent computational process. However, L is not a continuous scalar field, so ∇L is a discontinuous vector field, which performs poorly in computation. Therefore, this method is not used.

Inspired by computational fluid dynamics, instead of finding gradient of L , a concept 'pressure field' is applied. The analogy between cells with positive and negative L with source and sink with high and low pressure allows us to regard the mapping process as relaxation of source and sink, where flux flows from high pressure to low pressure. Therefore, with a pressure field ϕ , we have:

$$\frac{d\vec{p}}{dt} = \nabla\phi \quad (3)$$

The concept, divergence, is also used to connect the pressure field ϕ to the scalar field L . Divergence of a vector field is a scalar field, which represents the outward flux from an infinitesimal volume around a given point. For example, consider the case heating air. Velocity of air molecules generates a vector field, when air is heated in a region, air molecules expands in all directions, so the divergence of the velocity vector field is positive. Similarly, in our case, cells with positive L expands and with negative L shrinks, the greater L is, the more it needs to expand. Therefore, we obtain the following equation:

$$\nabla^2\phi = L \quad (4)$$

Solving this Poisson's equation gives a continuous and well behaved scalar field ϕ . Positions $p(x, y)$ of points (i, j) is updated repeatedly by computing the difference L , solving equation(4) and updating according to equation(3).

A square mesh is used to compute L . Gauss-Seidel method is applied to solve Poisson's equations, with Neumann boundary condition and over-relaxation parameter $\sigma = 1.94$. The vertices of the square mesh is updated using equation $\vec{p}(t_{n+1}) = \vec{p}(t_n) + \Delta t \nabla\phi$, where n indicates the n^{th} computation step. For the points on the edge, their movement is restricted within the side they lie on. A suitable Δt needs to be found for each iteration so that vertices do not surpass their neighbours. To

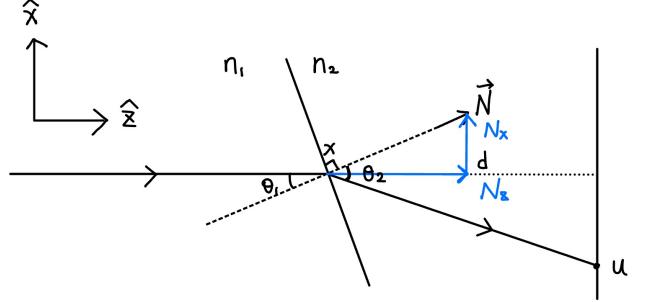


Fig. 3. Refractive in x direction. Light incident at coordinate x on the lens is refracted onto coordinate u on the screen. Using Snell's law, $\frac{\sin(\theta_1)}{\sin(\theta_2)} = \frac{n_2}{n_1}$, and small angle approximation, $\frac{\theta_1}{\theta_2} = \frac{n_2}{n_1}$, equation links θ_1 and θ_2 is obtained. By geometry, assuming d is much greater than x and u , $\theta_1 = \theta_2 - \tan^{-1}(\frac{x-u}{d})$. Normalizing N_z gives $N_x = \tan(\theta_1)$.

achieve this, the program iterates through all points, finding the limiting case of dt such that the closest two points would be coincidental, and returns half of dt to be Δt [8].

B. Finding Heightmap

After morphing the grids, the z coordinates, or a scalar field $z = h(x, y)$ is computed. A normal vector \vec{N} to the refractive surface can be represented by taking gradient of the surface $z = h(x, y)$, which is:

$$\vec{N} = \left(\frac{dh}{dx}, \frac{dh}{dy}, -1 \right) \quad (5)$$

where the z component is normalized to be 1. Define a vector N_{xy} contains only x and y component, we have:

$$\vec{N}_{xy} = \nabla h \quad (6)$$

The two component of N can be calculate using Snell's law and small angle approximation3,

$$N_x = \tan \frac{\tan^{-1}(\frac{u-x}{d})}{(n_1 - n_2)} \quad (7)$$

$$N_y = \tan \frac{\tan^{-1}(\frac{v-y}{d})}{(n_1 - n_2)} \quad (8)$$

where $n_1 = 1.49$ and $n_2 = 1$ are the refractive index of acrylic and air. Therefore, a second Poisson's equation is obtained by taking divergence of both sides,

$$\nabla^2 h = \nabla \cdot \vec{N}_{xy} \quad (9)$$

Initially, the refractive surface is set as a plane, so the distance $d(x, y)$ between the refractive surface and the projection screen is uniform. Components of \vec{N} in x and y direction and gradient of vector field \vec{N}_{xy} are then computed using equation (7), (8). A heightmap is found by solving equation (9), so that an updated scalar field $d(x, y)$ can be calculated. These processes are repeated until convergence. The mesh is then organized into .stl format for manufacture.

C. Computational Simulation

The refractive surface obtained by combining results from the above two steps is tested using a simulation program.

Each cell (i, j) is split into two triangles. For each triangle, the normal vector is obtained using the plane defined by vertices. Incident light in the direction of $(0, 0, 1)$ is divided into parallel and orthogonal components with respect to the normal vector of the cell. According to Snell's law, the output parallel component is scaled by factor of $\frac{n_1}{n_2}$ and the output perpendicular component remains unchanged. Thus, the direction of light (output vector) after refractive is found. For simplicity, all light that lands on the triangle is assumed to be transmitted through the center. Under this assumption, the output vector is scaled to find landing point on screen. If the landing point is within a grid, the area of triangle is added to corresponding point (u, v) . If the landing point is outside the grid, the incident is recorded, and the light is considered lost. These process are shown in Figure 4.

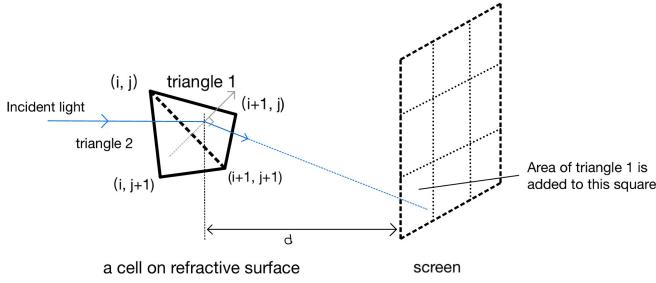


Fig. 4. Simulation of refraction for one piece of triangular surface.

After repeating the process for all triangles, the average of all points is scaled to 128, or half the maximum brightness in grey-scale. The result is saved as a .png file.

III. RESULTS

A. Simulation Result

Simulation results are shown in figure 5, where the test images are put on the left and the results on the right. Some accuracy is lost at small details, and the outline of objects tend to get distorted in the process. Nevertheless, the images are highly reproduced.

B. Manufactured Result

The acrylic lens are manufactured using a Roland MDX-40A Benchtop CNC Mill. Multiple setups were attempted with mixed results. The setup that yielded the smoothest surface was using R3 ball nose as both roughing and finishing tool. The resulting parts are then wet sanded with 400, 600, 800, 1000, 1200, and 1500 grit sandpapers. The parts are then washed and dried, before the fine and finish Tamiya polishing compounds are applied.



Fig. 5. Input images compared with simulated outputs. Resolutions are 500, 3000 and 900 separately.

IV. DISCUSSION

A. Computational process

To morph the grids, a $(N + 1) \times (N + 1)$ vector field is required (as there is $(N + 1) \times (N + 1)$ points in total), however, the $N \times N$ scalar field Φ is not sufficient to generate such a vector field. For the purpose of obtaining sufficient information, the scalar field is mirrored with respect to all four of its edges, turning the scalar field to $3N \times 3N$. Only the $(N + 2) \times (N + 2)$ data points in the center is used. A demonstration of the mirroring step for a 2×2 field is shown in Figure 56. This method simplifies the program and also automatically applies Neumann boundary conditions.

The computational process is extremely time consuming to use on pictures with high resolution since the algorithm has a $O(n^2)$ complexity, it takes about 30 minutes for 500×500 and about 5 hours for 3000×3000 . Therefore, a suitable resolution need to be chosen for each test image, such that it takes the program reasonable amount of time to finish calculation and generates a relative high-quality output. Most images were initially tested with a version that has reduced

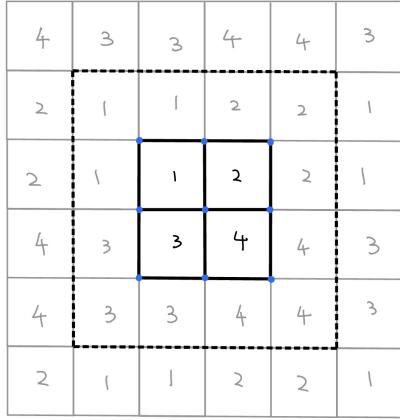


Fig. 6. Mirror for a 2×2 scalar field for demonstration. The 4×4 data in the dotted square is used to obtain 3×3 vectors (blue points).

resolution (300×300 or less). Numba was used to speed up calculations for high-resolution images.

B. Errors and limitations

The most efficient way to simulate a 3-D surface is triangle mesh, while square mesh is used in the program, which may leads to decrease of accuracy. In addition, when calculating height-map, interference effect is ignored. This effect is hard to avoid in reality, which limits the output image quality. Moreover, the fact that light passes through the entirety of the cells and may land on multiple pixels is not taken into account in the simulation program, resulting in great amount of error. However, these error can be diminished using high resolution.

A focus distance is assumed in advance when solving the height-map, but the whole refractive surface moves further from the screen during iteration process, resulting in difficulty to determine the exact output surface focus distance. Also, this method limit the distance d between refractive surface and screen to a range, output images quality drops significantly if d is out of range.

However, despite the resulting image being similar to the input, the simulation program suggests that less than 2% of the light actually land at where they are expected to land. Such result suggests that the accuracy of the heightmap is less than expected.

C. Manufacturing

A piece of acrylic (as proposed by Matt Ferraro) was shaped using CNC machine. Initially, a 1mm square nose is used for roughing, and a engraving tool with 0.02mm tip is used for finishing. This yielded a rough and nontransparent surface. Later approaches used R3 ball noses for both roughing and finishing with mixed results. By far, the smoothest surface is achieved using a R3 ball nose as roughing tool with 0.37mm

cut-in stepping and 0.05mm path interval, and a R2 ball nose as finishing tool with 0.37mm cut-in stepping and 0.20mm path interval with 0.37mm cut-in stepping and 0.05mm path interval.



Fig. 7. The surface of the manufactured acrylic lens

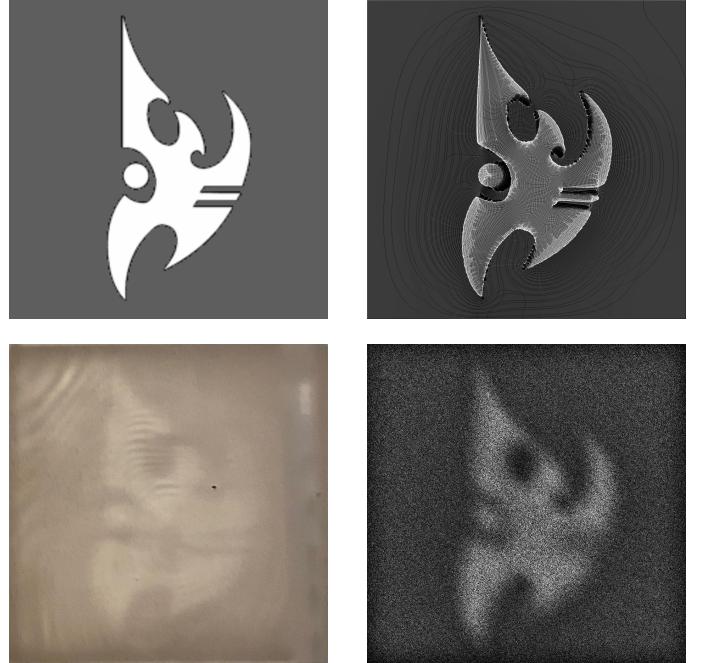


Fig. 8. In order: input image, simulated output with no error, manufactured object output, simulated output with max $\pm 0.01\text{mm}$ error

The manufactured objects are also severely affected by the contour lines created by the CNC mill, which are impossible to remove them with sandpaper without significantly changing the heightmap. The edges of the contour lines cast shadows on the projection plane and severely affects the quality of the output image. Unfortunately, the Roland MDX-40A CNC does not offer the option to remove the contour lines, only upcut or downcut. With alternate settings, this issue can be partially suppressed, at the cost of some accuracy.

The tolerance of CNC milling also limits the performance of the acrylic lens. The standard CNC tolerance is set around $\pm 0.127\text{ mm}$. If this is included in the simulation, with every

point's z coordinate off by a random number between 0 to 0.01 mm, the resulting output would be a lot closer to the output of the manufactured lens.

V. CONCLUSION

In conclusion, shape of a refractive surface, which is able to form a specific caustic image under parallel light, was computed using methods proposed by Matt Ferraro[9] and Yue et al[8]. Concepts in computational fluid dynamics are used to morph the image to the surface, then height-map of the surface is calculated. A computational simulation was built to test the code. Simulation results shows that the target image is highly reproduced with high resolution. The refractive surface was then manufactured by CNC machine and tested under lamp light. Even though only simple caustic patterns were able to be reproduced due to technical limitations, the experimental result proves effectiveness of the computational process. For future improvements, methods for solving poisson equation with greater accuracy and speed can be used. In addition, the computational process can be modified to improves edge representation.

REFERÊNCIAS

- [1] Martin Fuchs, Ramesh Raskar, Hans-Peter Seidel e Hendrik PA Lensch. “Towards passive 6D reflectance field displays”. Em: *ACM Transactions on Graphics (TOG)* 27.3 (2008), pp. 1–8.
- [2] Marios Papas, Thomas Houit, Derek Nowrouzezahrai, Markus H Gross e Wojciech Jarosz. “The magic lens: refractive steganography.” Em: *ACM Trans. Graph.* 31.6 (2012), pp. 186–1.
- [3] Miloš Hašan, Martin Fuchs, Wojciech Matusik, Hanspeter Pfister e Szymon Rusinkiewicz. “Physical reproduction of materials with specified subsurface scattering”. Em: *ACM SIGGRAPH 2010 papers*. 2010, pp. 1–10.
- [4] Christian Regg, Szymon Rusinkiewicz, Wojciech Matusik e Markus Gross. “Computational highlight holography”. Em: *ACM Transactions on Graphics (TOG)* 29.6 (2010), pp. 1–12.
- [5] Manuel Finckh, Holger Dammertz e Hendrik Lensch. “Geometry construction from caustic images”. Em: *European Conference on Computer Vision*. Springer. 2010, pp. 464–477.
- [6] Toshiya Hachisuka, Wojciech Jarosz, Guillaume Boucharde, Per Christensen, Jeppe Revall Frisvad, Wenzel Jakob, Henrik Wann Jensen, Michael Kaschalk, Claude Knaus, Andrew Selle et al. “State of the art in photon density estimation”. Em: *Acm Siggraph 2012 Courses* (2012), pp. 1–469.
- [7] Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi e Tomoyuki Nishita. “Pixel art with refracted light by rearrangeable sticks”. Em: *Computer Graphics Forum*. Vol. 31. 2pt3. Wiley Online Library. 2012, pp. 575–582.
- [8] Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi e Tomoyuki Nishita. “Poisson-based continuous surface generation for goal-based caustics”. Em: *ACM Transactions on Graphics (TOG)* 33.3 (2014), pp. 1–7.
- [9] Matt Ferraro. “Hiding Images in Plain Sight: The Physics Of Magic Windows”. Em: (2021). URL: <https://mattferraro.dev/posts/caustics-engineering>.