Image-space Caustics and Curvatures

Xuan Yu Feng Li Jingyi Yu
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA
{xuan,feng,yu}@cis.udel.edu

Abstract

Caustics are important visual phenomena, as well as challenging global illumination effects in computer graphics. Physically caustics can be interpreted from one of two perspectives: in terms of photons gathered on scene geometry, or in terms of a pair of caustic surfaces. These caustic surfaces are swept by the foci of light rays. In this paper, we develop a novel algorithm to approximate caustic surfaces of sampled rays. Our approach locally parameterizes rays by their intersections with a pair of parallel planes. We show neighboring ray triplets are constrained to pass simultaneously through two slits, which rule the caustic surfaces. We derive a ray characteristic equation to compute the two slits, and hence, the caustic surfaces. Using the characteristic equation, we develop a GPU-based algorithm to render the caustics. Our approach produces sharp and clear caustics using much fewer ray samples than the photon mapping method and it also maintains high spatial and temporal coherency. Finally, we present a normal-ray surface representation that locally parameterizes the normals about a surface point as rays. Computing the normal ray caustic surfaces leads to a novel real-time discrete shape operator.

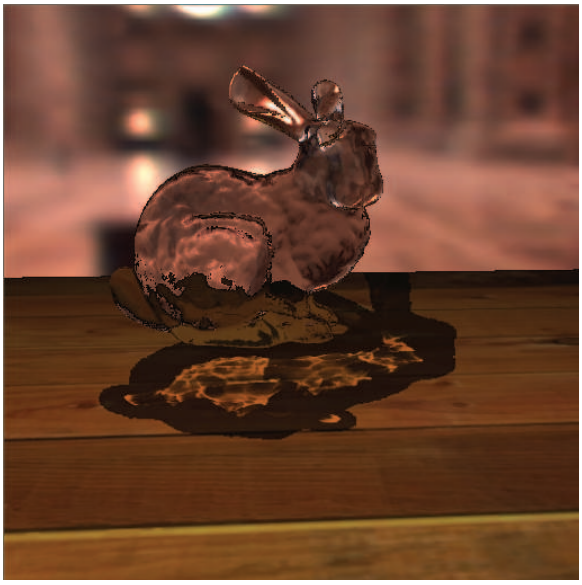

Figure 1. We use our caustic-surface-based algorithm to render the refraction caustics cast by a crystal bunny of 69473 triangles. On an NVidia GeForce7800, our method renders at 115 fps at an image resolution of 512x512.

1. Introduction

Caustics are important visual phenomena, as well as challenging global illumination effects in computer graphics. Bright caustic patterns are caused by the close bunching together of light rays. Traditionally, accurate caustics are rendered offline using backward ray-tracing [1] or photon mapping [9]. Both methods require tracing and gathering a large number of rays or photons to produce photorealistic results. Recently, GPU-based variants of path tracing algorithms have been developed to support near real-time caustics rendering [4, 20, 16]. There, the focus has been to determine the distribution, the size, and the density of photons.

Alternatively, caustics can be interpreted from the standpoint of caustic surfaces. These caustic surfaces are swept by the foci of light rays [6, 10]. In applied optics and computer vision, researchers have explored using the geometric attributes of the caustic surfaces to design catadioptric mirrors [17]. Although the basic theory of caustic surfaces is well understood, very little work has been done on estimating caustic surfaces of sampled rays. We [22] have recently proposed a local ray model [21] to approximate the reflection caustic surfaces of discrete mirror surfaces. They have also shown reflection distortions can be analyzed from the perspective of the reflection caustics. This indicates the im-

portance of correctly estimating the caustic surfaces when rendering.

In this paper, we develop a novel algorithm to approximate the caustic surfaces of sampled rays using graphics hardware. Our approach locally parameterizes rays by their intersections with a pair of parallel planes. We show neighboring ray triplets are constrained to pass simultaneously through two slits, which rule the caustic surfaces. We then derive a ray characteristic equation to compute the two slits, and hence, the caustic surfaces. Based on this characteristic equation, we develop a GPU-based algorithm to render the caustics. Our approach produces sharp and clear caustics using much fewer ray samples than the photon mapping method and it maintains high spatial and temporal coherency. Finally, we present a novel normal-ray surface representation that locally parameterizes the normals about a surface point as rays. Computing the normal ray caustic surfaces leads to a new real-time discrete shape operator.

Our key contributions include:

- A new framework that relates the caustic surfaces, the caustics, and the two-slit ray structure.
- A GPU-based algorithm that estimates the caustic surfaces by locally fitting the two-slit structure.
- A real-time caustics rendering algorithm using the ray characteristic equation.
- An image-space algorithm that estimates the curvatures from the normal caustic surfaces.

2. Previous Work

When light rays interact with a reflective or refractive surface, they may bend and alter their path. These refracted or reflected rays bundle together to form bright caustics on nearby scenes. In computer graphics, rendering caustics has been a challenging global illumination problem. Photon mapping and beam tracing have been two classical approaches to produce photorealistic caustics.

Photon mapping tracks photons emitted from the light source and stores their intersections with scene geometry [9]. These photons are then gathered on the receiving geometry and their density is estimated to compute the intensity of the caustics. However, to render photorealistic caustics, a large number of photons need to be traced and gathered. Distributed algorithms have been proposed to accelerate photon mapping [5]. Purcell et al [15] used a GPU-based ray tracer [14] to track photons stored on a uniform grid. Wand and Strasser [18] sampled each reflective object and treated each sample as a light source. They then gathered the contributions from each light sample using graphics hardware.

Recently, Wyman and Davis [20] proposed an image-space technique to efficiently emit and gather photons. Shah and Pattanik [16] developed a similar method using vertex tracing to construct a caustic map. They have also explored tracing fewer photons using fixed- or variable-sized photon splats. To maintain the rendering quality, they applied spatial and temporal filters to reduce the aliasing artifacts, but at the cost of additional computations [2, 19, 16].

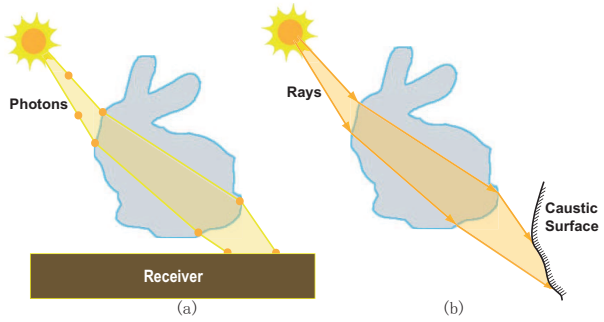

Figure 2. Caustics can be interpreted from one of two perspectives: (a) in terms of photons gathered on scene geometry, or (b) in terms of the caustic surfaces formed by light rays.

Caustics can also be rendered using beam tracing [7, 8]. Beam tracing computes the caustic polygons formed by the light rays on the receiver. To estimate the intensity for each caustic polygon, Nishita and Nakamae [11] used prism-shaped caustic volumes to calculate the energy flux passed onto the polygon. Ernest et al. [4] simplified this computation using a caustic volume warping. Their result show that high quality caustics can be produced with much fewer polygons than photons.

We present a third approach based on the geometry of caustic surfaces. In the literature, these surfaces represent an envelop of light rays [6]. In computer vision, such caustic surfaces have been used to guide the design of catadioptric imaging systems. Conventional catadioptric mirrors place a pinhole camera at the focus of a hyperbolic or parabolic surface to synthesize a different camera with a wider field of view. When the camera moves off the focus, the caustic surfaces quickly evolve into complicated shapes [17]. We have [22] proposed a ray space algorithm to estimate the reflection caustic surfaces of arbitrarily shaped mirrors. In this paper, we extend this ray space framework to compute the caustic surfaces formed by an arbitrary set of sampled rays using graphics hardware.

Before proceeding, we explain our notation. Superscripts, such as p^X, p^Y, and p^Z represent the x and y and z component of a point or vector. Subscripts, such as f_ξ and f_η represents the first-order partial derivatives of f with respect to ξ and η.

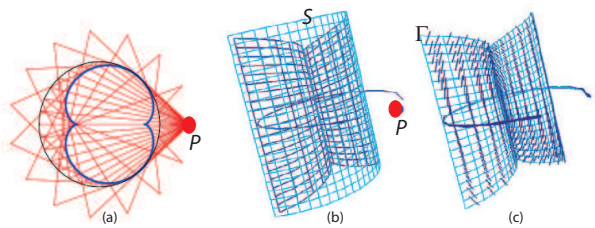

Figure 3. (a) The caustic surface represents the envelop (blue) of the rays (red). (b) The caustic surfaces (blue) formed by the rays emitted from a point light source (red) and reflected by a cylindric mirror (cyan). (c) The caustic surfaces (cyan) are ruled by the loci of the slits (blue).

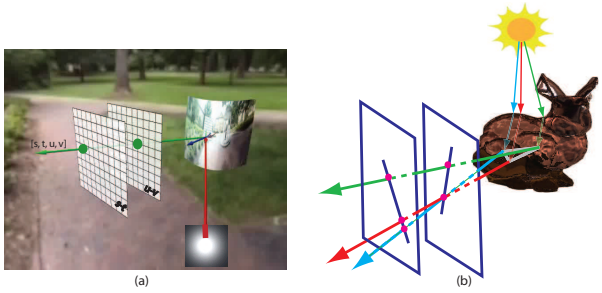

Figure 4. (a) At each point on the surface, the reflected (refracted) light ray is mapped into the ray space by intersecting with the two parametrization planes. (b) The neighboring ray triplets are constrained to pass simultaneously through two slits, which are parallel to the specified parametrization planes and rule the caustic surfaces.

3. Caustic Surfaces

Given a point light source P and a reflective or a refractive surface S, our goal is to compute the caustic surface Γ formed by the light rays that are transmitted through S, as shown in Figure 3. Assume S is parameterized in (ξ, η), we can represent each exiting light ray as $r = \dot{S}(\xi,\eta) + \lambda\vec{D}(\xi,\eta)$, where $\dot{S}(\xi,\eta)$ represents the origin of the ray and $\vec{D}$ represents the direction.

The caustic surfaces correspond to the envelop of the rays (Figure 3(a)) and they are tangential to each ray r along $\vec{D}$. Therefore, for some λ the caustic surface lies at

$$\Gamma(\xi,\eta) = \dot{S}(\xi,\eta) + \lambda\vec{D}(\xi,\eta) \tag{1}$$

Since $\vec{D}$ must be one of the tangent direction at $\Gamma(\xi,\eta)$, we must have:

$$\begin{vmatrix} S^X_\xi + \lambda\cdot D^X_\xi & S^X_\eta + \lambda\cdot D^X_\eta & D^X \\ S^Y_\xi + \lambda\cdot D^Y_\xi & S^Y_\eta + \lambda\cdot D^Y_\eta & D^Y \\ S^Z_\xi + \lambda\cdot D^Z_\xi & S^Z_\eta + \lambda\cdot D^Z_\eta & D^Z \end{vmatrix} = 0 \tag{2}$$

Solving for λ we get the caustic surfaces for each ray $r(\xi,\eta)$. This is often referred to the Jacobian method for computing the ray caustic surfaces [17, 12].

Equation (2), in general, is quadratic in λ and should have two solutions. This implies that caustic surfaces should appear in pairs except for the degenerate cases. Near the caustic surfaces, the light rays bunch up close together to form high energy flux and bright caustics.

3.1. Local Ray Parametrization

While the theory of caustic surfaces is well understood, little work has been done on estimating caustic surfaces of sampled rays. This is because accurately estimating the surface and direction differentials (e.g., $\dot{S}_\xi$, $\dot{D}_\xi$) from sampled rays can be difficult.

We present a new caustic surface estimation algorithm based on the recently proposed General Linear Camera (GLC) [21]. In the GLC framework, every ray is parameterized by its intersections with the two parallel planes, where $[u,v]$ is the intersection with the first and $[s,t]$ the second, as shown in Figure 4(a). This parametrization is often called a two-plane parametrization (2PP).

For each reflected/refracted ray r, we choose the uv plane to be perpendicular to r. We also orient the local frame to align the $z=0$ plane with the uv plane. We position the second st plane at $z=1$ parallel to the uv plane, as shown in Figure 4(a). Neighboring rays around each exiting light ray r can be represented by their intersections with the st and uv plane as $[s,t,1]$ and $[u,v,0]$, as shown in Figure 4(a).

In this paper, we use a slightly different parametrization $[\sigma,\tau,u,v]$, where $\sigma = s-u$ and $\tau = t-v$ so that $[\sigma,\tau,u,v]$ represents the direction of the ray. Under this new parametrization, each ray maps to a point in a four-dimensional $[\sigma,\tau,1]$ *ray space*. Furthermore, since we have aligned the $z=1$ plane with the uv plane, neighboring rays around r can be parameterized as in x and y as:

$$r(x,y) = [\sigma(x,y), \tau(x,y), u(x,y), v(x,y)] \tag{3}$$

To compute the caustic surface at r, we substitute $\dot{S} = [u,v,0]$ and $\vec{D} = [\sigma,\tau,1]$ into the Jacobian method (2) as:

$$\begin{vmatrix} u_x + \lambda\sigma_x & u_y + \lambda\sigma_y & \sigma \\ v_x + \lambda\tau_x & v_y + \lambda\tau_y & \tau \\ 0 & 0 & 1 \end{vmatrix} = 0 \tag{4}$$

In [22], we have shown that Equation (4) can be alternatively formulated as:

$$\begin{vmatrix} u+\lambda\sigma & v+\lambda\tau & 1 \\ (u+u_x)+\lambda(\sigma+\sigma_x) & (v+v_x)+\lambda(\tau+\tau_x) & 1 \\ (u+u_y)+\lambda(\sigma+\sigma_y) & (v+v_y)+\lambda(\tau+\tau_y) & 1 \end{vmatrix} = 0 \tag{5}$$

Equation (5) is quadratic in λ as

$$A\lambda^2 + B\lambda + C = 0 \tag{6}$$

where

$$\begin{aligned} A &= \sigma_x\tau_y - \sigma_y\tau_x, \quad C = u_xv_y - u_yv_x \\ B &= \sigma_xv_y - \sigma_yv_x - \tau_xu_y + \tau_yu_x \end{aligned} \tag{7}$$

We call Equation (5) the *ray characteristic equation*. The two solutions to the quadratic equation correspond to the depth of the caustic surfaces at each ray.

Notice, the LHS of Equation (5) can be interpreted as the area of the triangle formed by the three neighboring rays r, $r + r_x$, and $r + r_y$ on the $z = \lambda$ plane. This implies that this ray triplet envelops at a *line slit* on each piece of the caustic surfaces, as shown in Figure 4(b). The loci of the slits provide an important ruling of the caustic surfaces (Figure 3(b)).

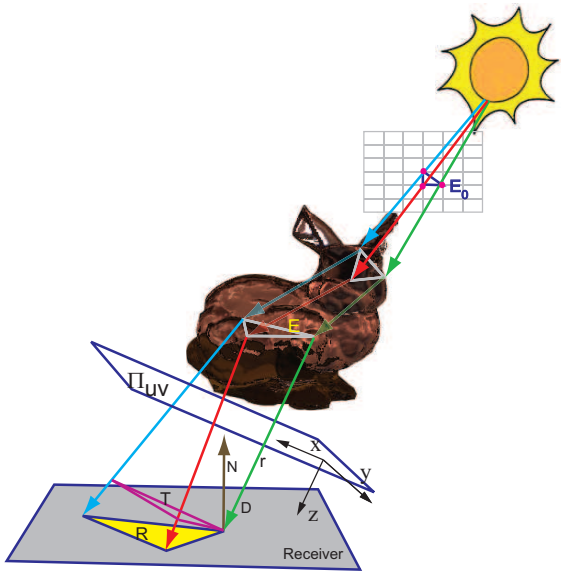

Figure 5. To compute the intensity for each ray (green), we map the neighboring ray triplet into the ray space under 2PP. We then use the ray characteristic equation to approximate the energy flex on the receiver.

4. Estimating Discrete Caustic Surfaces

The ray characteristic equation reveals that under the two plane parametrization, nearby light rays will pass through two slits that rule the two caustic surfaces. In this section, we show how to apply the local two-slit model to efficiently estimate the caustic surfaces from sampled rays.

Given a point light source and a reflective/refractive surface with vertices and vertex normals, we first compute the light rays exiting from the surface. For single refraction or refraction, we rasterize the vertices and normals of the front faces by treating the light source as a camera. We then use Snell's Law to calculate the exiting ray for each pixel using the fragment shader.

To handle multiple ray bounces for dielectric objects such as the crystal bunny (Figure 1), we employ Davis and Wyman's ray-depth map intersection technique [3]. We have implemented a similar iterative multi-pass rendering algorithm. In the first pass, we render the back faces with respect to the light source and store their normals and depths into two textures. In the second pass, we compute the rays refracted from the front face using the single bounce algorithm. To find the intersection point of each ray with the back surface, we apply a GPU-based binary search that iteratively estimates and corrects the intersection [3]. Finally, we use this intersection point to look up the normal from the back-face texture and use Snell's Law to compute the exiting ray. We store these exiting rays into two textures (one for position and one for direction).

To estimate the caustic surfaces for each exiting ray r, we use the fragment shader to fetch neighboring sets of three rays around r from the ray textures. We then choose a parametrization plane perpendicular to r and calculate the ray coordinates of the three rays under the 2PP. Finally, we compute the ray characteristic Equation (5), find its two solutions, and store the caustic surfaces.

Notice, the most computationally expensive step in our algorithm is to solve the quadratic ray characteristic equation. In Section 5 and 6, we show that this equation needs not to be solved when we render the caustics or estimate the mean and the Gaussian curvatures. Instead, we only need to store the A, B, and C coefficients of the ray characteristic equation.

5. Rendering Caustics

To render the caustics cast by the light rays, we first compute the intersection of each ray with the receiving geometry by reapplying the depth-map binary search algorithm. The only difference is that we treat the receiver as the back faces. Similar to [16], we store the resulting intersection points in a caustic map.

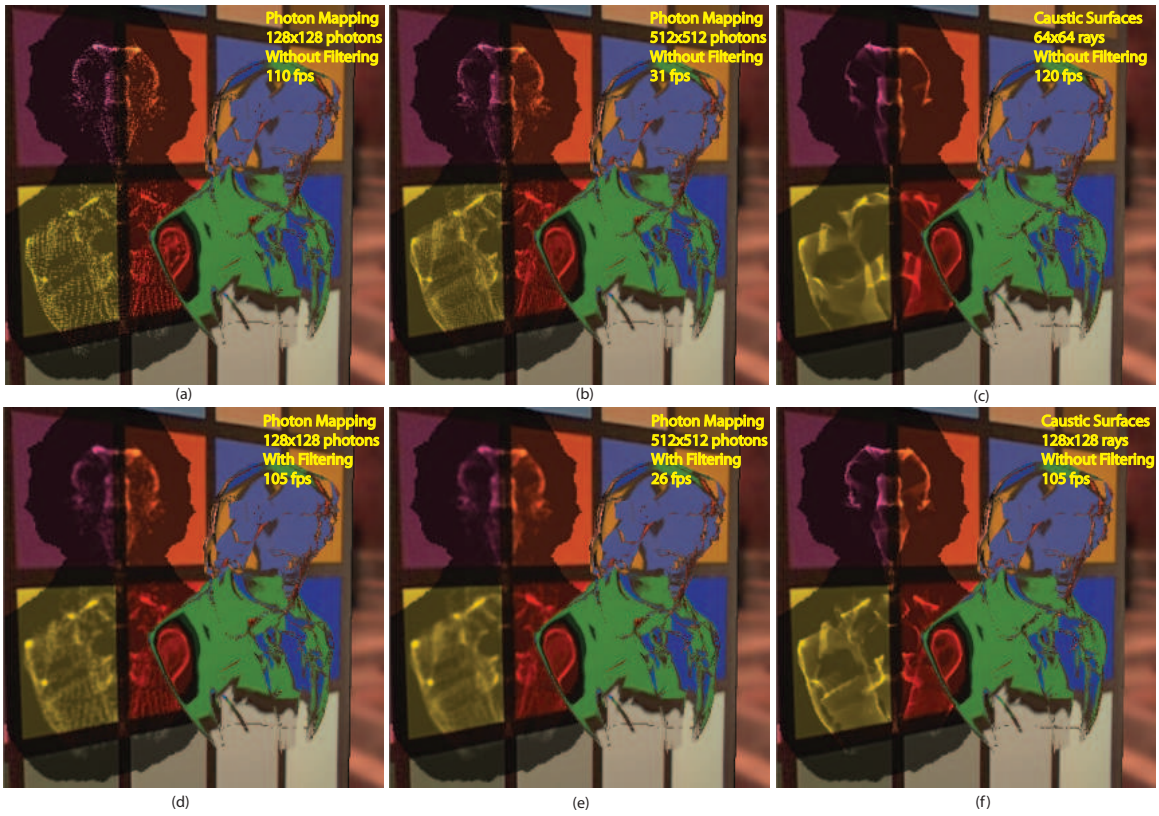

Figure 6. We compare our caustics surface algorithm with photon mapping.

To render the caustic map, previous approaches either render the intersection points as splats [16, 19] or compute the intensity for each caustic polygon[11, 4]. We set out to compute the intensity for each intersection point. Our key idea is to trace the energy carried by each ray triplet from the light source to the receiver. Assume the initial energy of the triplet from the light source is E_0 as shown in Figure 5. We first compute the attenuated energy due to material absorption after it leaves the object as $E = E_0 e^{-K_a d}$, where K_a is the absorption coefficient and d is the average distance that the rays have traversed inside the object.

Next, we estimate the intensity of each exiting light ray r that arrives on the receiver R. Unlike previous methods that use the solid angle [20] or warped light volume [4], we directly use the ray characteristic Equation (5). Recall that the LHS of Equation (5) calculates the area of a triangle T formed by the ray triplet on $z = \lambda$ plane. It has been shown that each ray triplet (GLC) defines a convex 2D subspace of rays under 2PP [21] and all rays inside the triplet are completely encapsulated by T. Therefore, the initial energy E carried by the ray triplet is conserved on T. Since the energy flux is inverse proportional to the area of T, when the area of T approaches to zero, the energy flux will become high. Thus, near the caustic surfaces, bright caustics appear.

Assume λ_R corresponds to the depth of the intersection point of ray r with the receiver R under 2PP, we can compute the area formed by the ray triplet on plane $z = \lambda_R$ as $A\lambda_R^2 + B\lambda_R + C$. We then warp this area onto the receiver as:

$$Area_R \approx \frac{A\lambda_R^2 + B\lambda_R + C}{\vec{D} \cdot \hat{N}} \tag{8}$$

where $\vec{D}$ is the direction of ray r and N is the normal of R. Notice Equation (8) better approximates the area covered by the rays formed by the triplet than directly computing the triangular area formed by the three rays on the receiver.

Finally, we can compute the intensity of ray r as

$$I_r = \frac{E}{|Area_R|} \tag{9}$$

Since our algorithm is GPU-based, we modify Equation (9) by adding a regularization term to avoid singularity and to maintain precision as:

$$I_r = \frac{E}{|Area_R| + \epsilon} \tag{10}$$

where ϵ is a fixed positive number. In addition, instead of using one ray triplet, we average I_r using all four neighboring triplets sharing r.

We have compared our new caustic rendering algorithm with classical photon mapping. In Figure 6, we render the

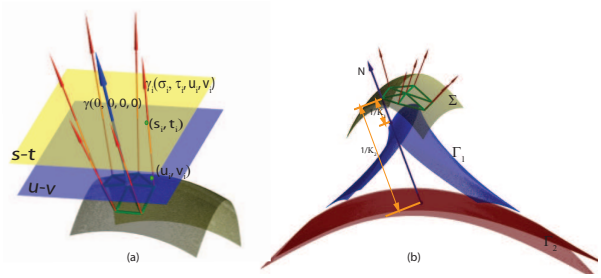

Figure 7. Surface normals can be represented as rays. (a) We orient the local frame to align the uv plane with the surface tangent plane. The neighboring normal rays can be parameterized as the intersections of the two planes as $[\sigma, \tau, u, v]$. (b) A smooth surface Σ has two sheets of normal ray caustic surfaces Γ_1 and Γ_2, each formed as the loci of the corresponding loci of principal curvature's radii.

caustics cast by the Beethoven model on a colored checkerboard. The Beethoven model consists of about 50K triangles. By sampling rays on a 128x128 grid, our method renders sharp and clear caustics at 105 fps on an NVidia GeForce7800. By tracing the same number of photons, photon mapping renders at approximately the same frame rate as our method. However, it produces noisy and temporally incoherent caustics. These artifacts are more noticeable in the animations. We have also increased the number of photons (Figure 6(b)) to improve the rendering quality. Although the resulting caustics reach a sharpness level similar to our method, they are still noisy and the frame rate drops to a quarter of our method. The noise level in photon mapping can be further reduced by using a Gaussian filter of size 7x7. However, applying these filters requires additional computations and results in an even lower frame rate (Figure 6(e)).

6. Estimating Curvatures

Finally, we show how to use the caustic surfaces to estimate differential geometry attributes on discrete surfaces. Our method is based on the observation that the vertices and normals of the original surface can be treated as rays, where each ray has its origin at a vertex and direction given by its normal as shown in Figure 7. We call these rays the *normal rays*.

To parameterize the normal rays, at each surface point we align the uv plane with the tangent plane and so that the normal corresponds to the z direction as shown in Figure 7(a). We can then compute the caustic surfaces using the ray characteristic Equation (5). In the literature, the foci of the normal rays are interchangeably referred to as evolutes, normal caustics, centro-surfaces, and focal surfaces [10]. We choose to use the term *normal caustics* to be consistent with ray caustics. The normal caustics have many important properties. For instance, they correspond to the loci of the principal curvatures' radii. In fact, the differential geometry of a smooth surface can be completely characterized from the perspective of normal caustics [12, 13].

Recall that the ray characteristic Equation (5) computes the depth of the caustic surfaces as λ_1 and λ_2. Since the normal caustics correspond to the loci of the surface's principal radii, we must have

$$\lambda_1 = -\frac{1}{\kappa_1}, \quad \lambda_2 = -\frac{1}{\kappa_2} \tag{11}$$

where κ_1 and κ_2 correspond to the min and the max curvature. Furthermore, since the coefficients of the characteristic equation must satisfy

$$\lambda_1 + \lambda_2 = -\frac{B}{A}, \quad \lambda_1 \cdot \lambda_2 = \frac{C}{A}, \tag{12}$$

we can re-derive the mean and the Gaussian curvature in terms of the coefficients of the ray characteristic equation without solving for λ_1 and λ_2:

$$\begin{aligned} K &= \kappa_1 \kappa_2 = \frac{1}{\lambda_1 \lambda_2} = \frac{A}{C} \\ 2H &= \kappa_1 + \kappa_2 = -\frac{1}{\lambda_1} - \frac{1}{\lambda_2} = \frac{B}{C} \end{aligned} \tag{13}$$

At the points where the Gaussian curvature K is zero, we must have $A = 0$ and the characteristic equation degenerates to a linear equation having at most one solution. These points correspond to the parabolic points of the base surface, therefore, only one normal caustics exists. If the mean curvature H is also zero, then we must have $A = 0$ and $B = 0$, and Equation (5) has no solution. In that case, the surface is locally flat, and there exists no normal caustic. Finally, we can compute the discriminant $\Delta = B^2 - 4AC$. If $\Delta = 0$, then the quadratic characteristic equation has double roots. This indicates the surface has two identical principal curvatures at the point, and thus, is umbilical [10].

6.1. Image-Space Curvature Estimation

We have implemented a multi-pass image-space curvature estimation algorithm by approximating the normal caustics on the GPU. In the first pass, the geometry is rasterized into two textures, one storing the position of the geometry per pixel, the second storing the normal of the geometry per pixel. In the second pass, we use the fragment shader to fetch neighboring sets of three pixels from both

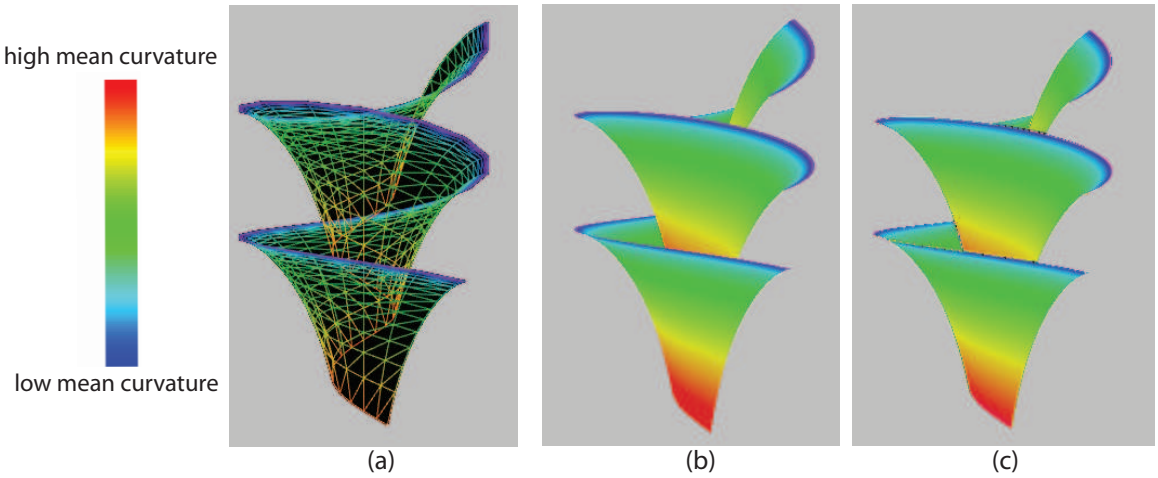

Figure 8. We have tested our image-space curvature estimation algorithm on a Dini surface. (a) We discretize the dini surface with a 32 by 32 mesh. We approximated the normal at each vertex by averaging the face normals of the triangles sharing the vertex. (b) The ground truth mean curvature of the Dini surface. (c) Our curvature estimation result using the GPU-based normal caustic surface approximation. Our method robustly computes per-pixel-based mean curvature at 110 fps on an NVidia GeForce7800.

textures. We then map them to the normal rays under 2PP, and compute the solution to their characteristic equation. If one needs to only compute the mean or the Gaussian curvature, then only the A, B, and C coefficients of the equation are required as shown in Equation (13).

We have experimented our new curvature estimation algorithm on both discretized analytical surfaces and scanned surfaces. Figure 8 shows a Dini surface that has format:

$$\begin{aligned} & x = \cos u \sin v, \quad y = \sin u \sin v \\ & z = \cos v + \ln[\tan(\frac{v}{2})] + 0.2u \\ & u \in [0, 4\pi], \quad v \in [0.2, 1.5] \end{aligned} \tag{14}$$

We sampled the surface to form a mesh of 1024 triangles (Figure 8a). We then apply our GPU-based algorithm to estimate the mean curvature from the mesh. At the rim of the Dini surface, the mean curvature evolves rapidly as shown in Figure 8(b). Our method faithfully captures these details at 110 fps with an image resolution of 512x512.

On complex scanned models such as the dragon, we approximated the normal at each vertex by averaging the face normals of the triangles sharing the vertex. Our normal caustics method produces highly smooth curvature fields as shown in Figure 9(a). Since the dragon model consists of 126,201 vertices and 250,000 faces, it is difficult to use traditional triangle-space curvature estimation algorithms to achieve real-time performance. Our GPU-based algorithm, on the other hand, approximates the curvatures in the image-space, and hence, is scalable to the complexity of the models. On the dragon model, our method computes the mean curvature at 71 fps at an image resolution of 512x512. The image-space nature of our method also provides a multi-resolution estimation of the curvature fields. As we zoom in towards the model, the curvature fields are re-estimated and more details are revealed as shown in Figure 9(b).

7. Conclusions and Discussions

We have developed a novel caustic surfaces estimation algorithm to render caustics and to compute the curvature fields on discrete surfaces. Our approach locally parameterizes the rays by their intersections with a pair of parallel planes. We have shown neighboring ray triplets are constrained to pass simultaneously through two slits, which are parallel to the specified parametrization planes and rule the caustic surfaces. These slits can be derived using the ray characteristic equation. We have derived a ray characteristic equation to compute the two slits, and hence, the caustic surfaces. Using the characteristic equation, we have developed a GPU-based algorithm to render the caustics. Our approach produces sharp and clear caustics using much fewer ray samples than the photon mapping method. Finally, we have presented a novel normal-ray surface representation that locally parameterizes the normals about a surface point as rays. Computing the normal ray caustic surfaces leads to a novel real-time discrete shape operator.

Our caustics rendering algorithm shares certain similarities with the recently proposed caustic map [16] and volume

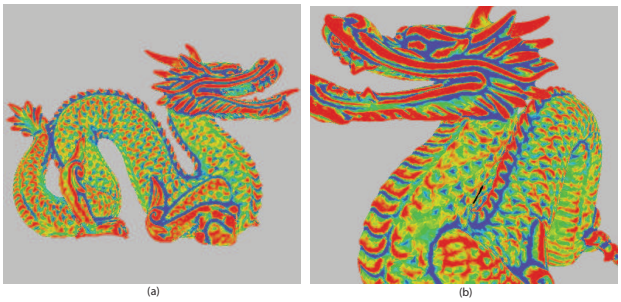

Figure 9. Our curvature estimation computes per-pixel-based curvatures and is scalable to the complexity of the model. (a) On a dragon model of 126201 vertices and 250000 triangles, our method runs at 71 fps. (b) shows the close-up view of the dragon. When zoomed in, our algorithm captures many fine curvature details.

warping [4] approaches. The main difference here is that we interpret the caustics from the standpoint of the caustic surfaces and derive the energy flux from the ray characteristic equation. As for future work, since our algorithm can estimate the caustic surfaces of arbitrary set of rays, we plan to explore how to decompose environment lighting into combinations of special subspaces of rays so that the caustics can be rendered by summing up the contributions from individual caustic surfaces.

Finally, it has long been recognized that higher-order geometric attributes are desirable for surface modeling and physical-based animations. The caustic surfaces provide an enormous wealth of such geometric insights. As our GPU-based curvature estimation algorithm runs at the rendering stage, it can be easily integrated into many existing modeling and rendering systems to provide useful caustic surface geometries.

References

[1] J. Arvo. Backwards ray tracing. *Developments in Ray Tracing*, pages 259–263, 1986.

[2] C. Dachsbacher and M. Stamminger. Splatting indirect illumination. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 93–100, New York, NY, USA, 2006. ACM Press.

[3] S. Davis and C. Wyman. Interactive refractions with total internal reflections. In *Proceedings of Graphics Interface*, May 2007.

[4] M. Ernst, T. Akenine-Möller, and H. W. Jensen. Interactive rendering of caustics using interpolated warped volumes. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, pages 87–96, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[5] J. Günther, I. Wald, and P. Slusallek. Realtime caustics using distributed photon mapping. In *Proceedings of the Eurographics Symposium on Rendering*, pages 111–121, 2004.

[6] W. R. Hamilton. Theory of systems of rays. *Transactions of the Royal Irish Academy*, 15:69–174, 1828.

[7] P. S. Heckbert and P. Hanrahan. Beam tracing polygonal objects. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 119–127, New York, NY, USA, 1984. ACM Press.

[8] K. Iwasaki, Y. Dobashi, and T. Nishita. A fast rendering method for refractive and reflective caustics due to water surfaces. In *Proceedings of EUROGRAPHICS2003 (Computer Graphics Forum)*, volume 22, pages 601–609, 2003.

[9] H. W. Jensen. Realistic image synthesis using photon mapping. In *AK Peters*, 2001.

[10] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, 1994.

[11] T. Nishita and E. Nakamae. Method of displaying optical effects within water using accumulation buffer. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 373–379, New York, NY, USA, 1994. ACM Press.

[12] I. R. Porteous. *Geometric Differentiation for the Intelligence of Curves and Surfaces*. Cambridge University Press, Cambridge, 1994.

[13] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer, 2001.

[14] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan. Ray tracing on programmable graphics hardware. *ACM Trans. Graph.*, 21(3):703–712, 2002.

[15] T. J. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 258, New York, NY, USA, 2005. ACM Press.

[16] M. A. Shah, J. Konttinen, and S. Pattanaik. Caustics mapping: An image-space technique for real-time caustics. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):272–280, 2007.

[17] R. Swaminathan, M. D. Grossberg, and S. K. Nayar. Non-Single Viewpoint Catadioptric Cameras: Geometry and Analysis. *International Journal of Computer Vision*, 66(3):211–229, Mar 2006.

[18] M. Wand and W. Strasser. Real-time caustics. In *Computer Graphics Forum*, pages 611–620, 2003.

[19] C. Wyman and C. Dachsbacher. Improving image-space caustics via variable-sized splatting. *Journal of Graphics Tools*, to appear.

[20] C. Wyman and S. Davis. Interactive image-space techniques for approximating caustics. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games*, pages 153–160, March 2006.

[21] J. Yu and L. McMillan. General linear cameras. In *In ECCV (2) (2004)*, pages 14–27. Lecture Notes in Computer Science 3022 Springer, 2004.

[22] J. Yu and L. McMillan. Modelling reflections via multiperspective imaging. In *CVPR '05 - Volume 1*, pages 117–124, Washington, DC, USA, 2005. IEEE Computer Society.