

A Flexible Pinhole Camera Model for Coherent Nonuniform Sampling

Voicu Popescu and Bedrich Benes ■ *Purdue University*

Paul Rosen ■ *University of Utah*

Jian Cui ■ *Purdue University*

Lili Wang ■ *Beihang University*

Most computer graphics and visualization applications employ images computed with the planar pinhole camera (PPC) model. The PPC is a good approximation of the human eye, which makes it well suited for applications showing users what they would see during an actual exploration of the scene. However, in some applications, the PPC model's reduced field of view, single viewpoint, and uniform sampling rate create a severe disadvantage.

The flexible pinhole camera allows modulating the sampling rate over the field of view with fine granularity. It inexpensively renders complex datasets by projection followed by rasterization. The resulting image is a coherent nonuniform sampling of the dataset that matches the local variation of the dataset's importance.

To address the uniform-sampling-rate limitation, we developed the *flexible pinhole camera* (FPC), which lets users adjust the sampling rate according to the local importance or complexity of the imaged data. Like the PPC, the FPC is defined by a viewpoint (the center of projection or eye) and an image plane. However, the sampling locations are defined not by a uniform grid but by a sampling map that allows shifting sampling locations from one region of the image plane to another. The FPC image provides a *coherent non-*

uniform sampling (CoNUS) of the dataset. For example, the CoNUS image in Figure 1a samples the five faces at a higher rate. The underlying sampling map (see Figure 2) has the topology of a 32×32 regular rectangular mesh but is distorted to implement the sampling-rate modulation. (For an overview of other research on nonuniform sampling, see the sidebar at the end of the article.)

CoNUS images preserve the advantages of conventional images. A CoNUS image can be computed quickly with the help of GPUs. Data access is constant-time, with the small additional cost of the sampling-map indirection. A CoNUS image has good pixel-to-pixel coherence, and conventional image compression algorithms apply. Finally, a CoNUS image remains a single-layer 2D array of samples that defines connectivity implicitly.

The Flexible Pinhole Camera

The FPC must be flexible, to allow defining the desired sampling rate for individual image subregions. It must also be fast, to render the CoNUS image quickly from a variety of input data types.

The Flexible Pinhole Camera

The Camera Model

We implement the sampling-rate variation with a sampling map that defines a distortion of a regular



Figure 1. Working with coherent nonuniform sampling (CoNUS) images. (a) A CoNUS image that allocates more samples to the face regions. (b) An output frame reconstructed from the CoNUS image. (c) An output frame reconstructed from a conventional image of the same size.

2D mesh. The distorted mesh has the same topology as a regular 2D mesh, but with quadrilateral cells that are larger where we want a higher sampling rate (see Figure 2). We encode the sampling map as a 2D array of 2D points. Each point defines a node of the distorted mesh.

Given an (undistorted) image point (u, v) , we find the corresponding (distorted) CoNUS image point (u_d, v_d) by looking up the sampling map using bilinear interpolation (see Figures 3 and 4). We first convert the input point to sampling-map coordinates (u', v') (line 1 in Figure 3). Then, we compute the distorted point by bilinear interpolation of the four distorted mesh points stored in the sampling map at the 2×2 neighborhood containing (u', v') (line 2).

Camera model definition. We define the camera model *FPC* with a conventional planar pinhole camera *PPC* and a sampling map *SM* that distorts the *PPC* image according to the algorithm in Figure 3.

Projection. $FPC(PPC, SM)$ projects a 3D point P to its CoNUS image plane by first projecting P with *PPC* to obtain (u, v) and then distorting (u, v) to (u_d, v_d) (see Figure 3).

Camera rays. The $FPC(PPC, SM)$ ray through (u_d, v_d) is the *PPC* ray through (u, v) . So, to compute the camera ray, we must invert the distortion, which poses two challenges.

First, we must find the quadrilateral cell of the distorted mesh containing (u_d, v_d) . A naive approach would examine all quads. A better approach would be to use a hierarchical subdivision of the CoNUS image (for example, using a *k-d* tree or a binary-space-partitioning tree) to quickly find the quad containing (u_d, v_d) . However, constructing the subdivision is laborious.

Second, we must solve the quadratic equations of the inverse bilinear interpolation that computes x and y from (u_d, v_d) , $SM_{i,j}$, $SM_{i+1,j}$, $SM_{i,j+1}$, and $SM_{i+1,j+1}$.



Figure 2. The sampling map for Figure 1a. It has the topology of a 32×32 regular rectangular mesh but is distorted to implement the sampling-rate modulation.

```

Algorithm: FPC::Distort( $u, v$ ) // FPC distortion
input: undistorted image resolution ( $w, h$ ),
         undistorted location ( $u, v$ ), and
         sampling map SM of resolution ( $w_0, h_0$ )
output: distorted location ( $u_d, v_d$ )
1: ( $u', v'$ ) = ( $uw_0/w, vh_0/h$ )
2: ( $u_d, v_d$ ) = SM.BilinearLookup( $u', v'$ )

```

Figure 3. The flexible pinhole camera (*FPC*) distortion algorithm. Given an (undistorted) image point (u, v) , we find the corresponding (distorted) CoNUS image point (u_d, v_d) by looking up the sampling map using bilinear interpolation.

We bypass these challenges by leveraging two observations:

- CoNUS applications don't need to compute an individual ray of *FPC* but rather all rays iteratively.

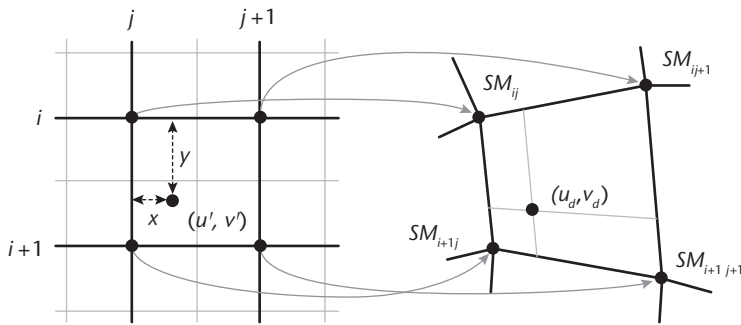


Figure 4. Piecewise bilinear image distortion using a sampling map. The sampling map provides a constant time mapping from the undistorted to the distorted domain.

- We can avoid bilinear-interpolation inversion by splitting the distorted mesh quads into two triangles and replacing the quad bilinear interpolation with two triangle barycentric interpolations.

This modification doesn't reduce the sampling-rate flexibility of FPC. We efficiently find all rays of the modified FPC (see Figure 5).

We find the FPC rays by rasterizing the distorted mesh QM defined by the sampling map (line 3 in Figure 5). QM has the topology of a 2D regular mesh, but its vertices are displaced according to the desired sampling-rate variation (see Figure 2). Each distorted mesh vertex carries its undistorted coordinates as texture coordinates (line 4).

To find the ray at the current pixel p , we first find (u, v) for p from its texture coordinates (s_p, t_p) (line 8). We then compute the PPC ray at the undistorted coordinates (line 9). We find the rays at the cost of rasterizing the $2 \times w_0 \times h_0$ triangles of the distorted mesh. This cost is small because the sampling map's resolution is much smaller than the CoNUS image's resolution. We compute the

rays on the GPU with a trivial fragment shader that executes lines 8 and 9.

The algorithm in Figure 5 provides the FPC rays, one at a time, at a small amortized cost. To render a CoNUS image from a regular image or volume data, we specialize the algorithm as described next.

Rendering CoNUS Images

The FPC renders CoNUS images efficiently from geometry, image, height field, and volume data.

Geometry data. Figure 6 shows the steps for rendering a CoNUS image from a 3D triangle mesh T . We project the vertices of T with FPC (PPC projection followed by distortion). Then, we conventionally rasterize the projected triangles. These triangles must be small enough such that conventional rasterization provides a good approximation of the nonlinear projection induced by the sampling map. Most datasets have small triangles, and conventional rasterization is acceptable without further subdivision. When subdivision is needed, we use an offline approach to avoid the performance bottleneck of issuing a large number of primitives in the geometry shader.

Image data. We render a CoNUS image from a conventional input image by modifying line 9 of Figure 5 (see Figure 7). Once we know (u, v) , we look up the input image to set (u_d, v_d) . A CoNUS image has fewer pixels than the original image. The original image provides the maximum resolution over the entire field of view, which is preserved in some regions of the CoNUS image. The other regions of the CoNUS image are at lower resolution.

```

Algorithm: FPC::Rays() // Computation of FPC rays
input: FPC of resolution  $w \times h$ , defined by PPC and by SM of resolution  $w_0 \times h_0$ 
output: FPC rays
1: Initialize 2D mesh  $QM$  of resolution  $w_0 \times h_0$ 
2: for all  $(i, j)$  where  $0 \leq i < w_0, 0 \leq j < h_0$  do
3:   Vertex coordinates  $QM.v_{i,j} = SM_{i,j}$ ;
4:   Texture coordinates  $QM.(s, t)_{i,j} = (i/w_0, j/h_0)$ ;
5: end for
6: for all triangles  $q$  in  $QM$  do
7:   for all pixels  $p$  covered by  $q$  do
8:      $(u, v) = (ws_p, ht_p)$ 
9:      $ray_p = PPC.GetRay(u, v)$ 
10:   end for
11: end for

```

Figure 5. The algorithm for efficiently finding all rays of the modified FPC. To render a CoNUS image from a regular image or volume data, we specialize the algorithm as needed.

Height field data. We similarly construct a CoNUS height field sampled orthogonally to the base plane. However, we set up the pixel by looking up the depth in the original height field instead of (or in addition to) looking up the color.

Volume data. We render a CoNUS image from volume data by tracing the *FPC* rays through the volume. We determine the rays using the algorithm in Figure 5.

Resampling a Regular Image from a CoNUS Image

Some applications, such as remote visualization, use the CoNUS image as an intermediate representation from which they must resample a conventional image to present to the user. We resample a regular image I_1 from a CoNUS image I_0 with the steps in Figure 8. A planar pinhole camera PPC_1 defines the rays that sample I_1 .

Given pixel (u_1, v_1) of I_1 , we compute the corresponding (u_d, v_d) in two steps. First, we compute the corresponding point (u_0, v_0) on the image plane of PPC_0 (lines 2 and 3 in Figure 8). We compute this correspondence by generating P corresponding to (u_1, v_1) by unprojection with PPC_1 and then by projecting P with PPC_0 . We combine the unprojection and following projection into a single matrix multiplication followed by perspective divides. Second, we compute the corresponding (u_d, v_d) by distortion, leveraging the algorithm in Figure 3.

Sampling-Map Construction

We construct sampling maps in one of three ways. The first uses an interactive physics-based 2D mass-spring system. We cover the image with regularly distributed particles connected with springs to form a quadrilateral mesh. All particles have the same mass, and all springs have the same resting length (set to 10 percent of the initial particle distance in our implementation). The user perturbs the system interactively by adding repulsive forces between particles with a circular brush (see Figure 9). The force magnitude decreases exponentially from the brush's center toward its periphery.

We compute the equilibrium state by tracking each particle's position over time until all particle velocity vectors have negligible magnitude. For each time step, we compute the forces on each particle. First, we use Hooke's equation for harmonic oscillators, $F_i = -kx_i$, where F_i is the force applied to the particle by spring i connected to it, k is the spring constant, and x_i is the particle displacement along the spring direction. Then, we

```

Algorithm: FPC::Render( $T$ ) // render from
                                geometry
input: FPC  $FPC$  and triangle mesh  $T$ 
output: CoNUS image  $I$ 
1: for all vertices  $v$  of  $T$  do
2:    $v' = FPC.Project(v)$ 
3: end for
4: for all projected triangles  $t'$  of  $T$  do
5:   Rasterize  $t'$ 
6: end for

```

Figure 6. The algorithm for rendering a CoNUS image from a 3D triangle mesh T . We project the vertices of 3D triangle mesh T with *FPC*. Then, we conventionally rasterize the projected triangles.

```

Algorithm: FPC::Render( $I$ ) // render from image
input: FPC  $FPC$  and image  $I$ 
output: CoNUS image  $I'$ 
// identical to Figure 5 except for line 9
...
9:  $I'(u_d, v_d) = I(u, v)$  // the difference
                                from Figure 5
...

```

Figure 7. The algorithm for rendering a CoNUS image from a conventional input image. The original image provides the maximum resolution over the entire field of view, which is preserved in some regions of the CoNUS image. The other regions of the CoNUS image are at lower resolution.

```

Algorithm: FPC::CoNUS2Regular( $I_0$ ) // Resampling
input: CoNUS image  $I_0$ ,  $FPC(PPC_0, SM)$ ,  $PPC_1$ 
output: Conventional image  $I_1$  for  $PPC_1$ 
1: for all pixels  $(u_1, v_1)$  in  $I_1$  do
2:    $P = PPC_1.Unproject(u_1, v_1)$ 
3:    $(u_0, v_0) = PPC_0.Project(P)$ 
4:    $(u_d, v_d) = FPC.Distort(u_0, v_0)$ 
// see Figure 3
5:    $I_1(u_1, v_1) = I_0(u_d, v_d)$ 
6: end for

```

Figure 8. The algorithm for resampling a regular image from a CoNUS image. We combine the unprojection and following projection into a single matrix multiplication followed by perspective divides.

update the particle velocity \mathbf{v} and displacement \mathbf{x} , using $\mathbf{v} = \mathbf{v} + \Delta t \mathbf{F}/m$ and $\mathbf{x} = \mathbf{x} + \Delta t \mathbf{v}$, where \mathbf{F} is the resultant force acting on the particle. A mesh of 256×256 particles updates at 30 fps, and we reach a stable state in less than 2 s. The particles' final position defines the sampling map, which can have a lower resolution than the particle mesh.



Figure 9. A mass-spring system for defining sampling maps interactively. The user defines regions of higher resolution using the yellow circular brush.

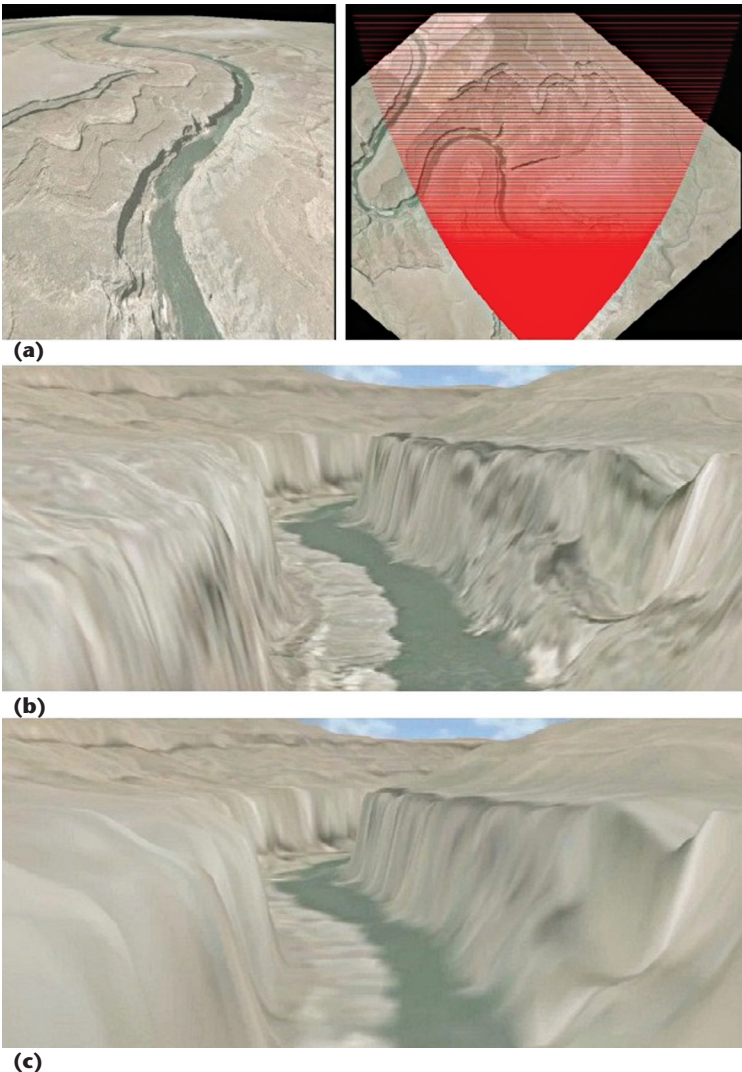


Figure 10. Remote visualization of a height field. (a) A CoNUS height field and its sampling pattern. (b) An output frame rendered from the CoNUS height field. (c) An output frame rendered from a conventional height field. If the server sends the CoNUS height fields instead of the conventional one, the output frame's fidelity increases considerably.

We can also construct sampling maps through a linear combination of the distortion vectors of existing sampling maps:

$$SM_{i,j} = SM_{i,j}^0 + \sum_k (s_k (SM_{i,j}^k - SM_{i,j}^0) + t_k),$$

where $SM_{i,j}$, $SM_{i,j}^0$, and $SM_{i,j}^k$ are elements (i, j) of the new sampling map, the undistorted sampling map, and the input sampling map k , respectively, and s_k and t_k are the scale factor and translation vector of k .

The third way eliminates the discrete representation and defines the distortion analytically, as we describe in the next section.

Applications

We've used CoNUS images for remote visualization, accelerating depth image rendering, and focus-plus-context visualization. (See also the video at <http://doi.ieeecomputersociety.org/10.1109/MCG.2014.21>. For an overview of related research on these topics, see the sidebar.)

Remote Visualization

Digital-camera resolution continues to increase faster than network bandwidth. In addition, workstation displays now have lower resolution than the simplest digital cameras attached to cellular phones (for example, Apple's 4-Mpixel 30" LCD versus the 8-Mpixel iPhone 5S camera). So, even if the image is transferred at full resolution, it will most likely be downsized for viewing.

Often, a digital image's pixels won't all have the same relevance for the application. For example, faces in a portrait photograph are more important than the room furnishings (see Figure 1). Moreover, digital cameras automatically find faces for focusing. In an online geographic atlas, pixels sampling famous locations or locations that other users have marked as interesting are more relevant. In remote scientific visualization, some image regions might be known to be of higher interest to scientists, such as regions showing receptors targeted in drug molecule design.

In such contexts, our approach could help reduce bandwidth requirements and improve interactivity. The server renders a CoNUS image that samples the ROIs at a higher rate (see Figure 1). Then, the server transfers that image to the client, which resamples it into a conventional image. The application tours the CoNUS image, showing the ROIs in detail.

We've also used our approach for remote terrain visualization (see Figure 10). Given a height field H at the server and a current view PPC at the

client, we want to resample H to a CoNUS height field that has all and only the samples needed to provide a quality visualization of the height field from views in the neighborhood of PPC .

First, we construct a reference view PPC_0 by enlarging the field of view of PPC to support view rotations and increasing the resolution to support zooming in and forward translation. Then, we construct a CoNUS height field CH with a sampling rate matching the requirements of PPC_0 . CH should have more samples close to the viewpoint and fewer at a distance (see Figure 10b). We construct CH with the analytical distortion function in Figure 11.

We look up (u_d, v_d) in H at location (u, v) , which we compute by intersecting the ray at (u_d, v_d) in PPC_0 with the ground plane $H.g$ of H . This construction applies the perspective foreshortening of PPC_0 while maintaining the orthogonal sampling of H . This avoids disocclusion errors that would occur if we actually rendered the geometry of H from PPC_0 . We send CH to the client, which transforms it into a 3D triangle mesh that's rendered for each frame. To convert a CH sample to a 3D triangle mesh vertex, we compute the ground plane point P corresponding to (u_d, v_d) (line 2 in Figure 11) and offset P by $CH(u_d, v_d)$ above the ground plane.

Quality. The CoNUS image in Figure 1 allows rendering all five faces in great detail. The CoNUS height field produces frames comparable to those rendered from the original high-resolution height field (see Figure 10).

Performance. For Figure 1, once we know the FPC model, rendering the CoNUS image takes negligible time. We designed the FPC sampling map interactively, using the spring-mass system. For Figure 10, we used a CoNUS height field of $1,024 \times 1,024$ resolution, which was rendered at over 400 fps and used at over 100 fps.

Limitations. Our approach increases the ROI sampling rate at the expense of the rest of the image. When high frequencies are outside the ROIs, the undersampling can become noticeable (see Figure 12).

Our approach doesn't address occlusions. Occlusions don't occur for images or orthogonally sampled height fields. However, for our approach to support six-degree-of-freedom remote visualization of general 3D data, we'll have to integrate it with an occlusion alleviation scheme such as a nonpinhole camera.

```

Algorithm: HeightFieldCoNUS( $H, PPC_0$ )
input: Height field  $H$ , client reference view  $PPC_0$ 
output: CoNUS height field  $CH$ 
1: for all samples  $(u_d, v_d)$  in  $CH$  do
2:    $(u, v) = PPC_0.Ray(u_d, v_d) \cap H.g$ 
3:    $CH(u_d, v_d) = H(u, v)$ 
4: end for

```

Figure 11. The analytical distortion function. The constructed height field CH has a sampling rate matching the reference view's requirements.

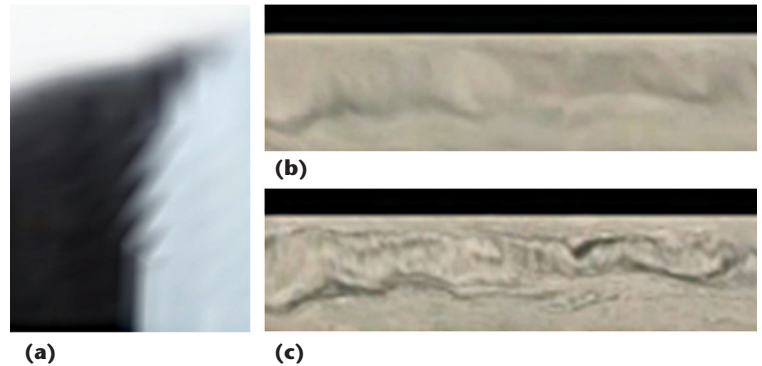


Figure 12. Limitations of our approach. (a) A sampling artifact outside the regions of interest (ROIs) in a frame reconstructed from the CoNUS image in Figure 1. (b) Undersampling of a distant mountain by the CoNUS height field. (c) The original height field. Our approach increases the ROI sampling rate at the expense of the rest of the image.

Accelerating Depth Image Rendering

A depth image can be computed quickly with the help of graphics hardware and can be quickly intersected with a ray. Because of these important advantages, depth images have been used to accelerate the rendering of complex effects such as specular reflection, refraction, ambient occlusion, and relief texture mapping. Eliminating the uniform-sampling-rate constraint of conventional depth images through our approach could benefit all these techniques, provided we preserve the efficiency of depth image construction and ray intersection. CoNUS depth images can be rendered efficiently from height field or geometry data using the FPC, as we discussed before.

To intersect a conventional depth image with a ray, the ray is projected to the depth image's plane, and the projection is traced with one-pixel steps until an intersection is found.¹ For a CoNUS depth image, the ray's projection is no longer a line segment but a curve segment. We can no longer project the ray solely by projecting its endpoints. Instead, we must subdivide it into segments and project each segment endpoint with the FPC. This preserves the fundamental advantage of

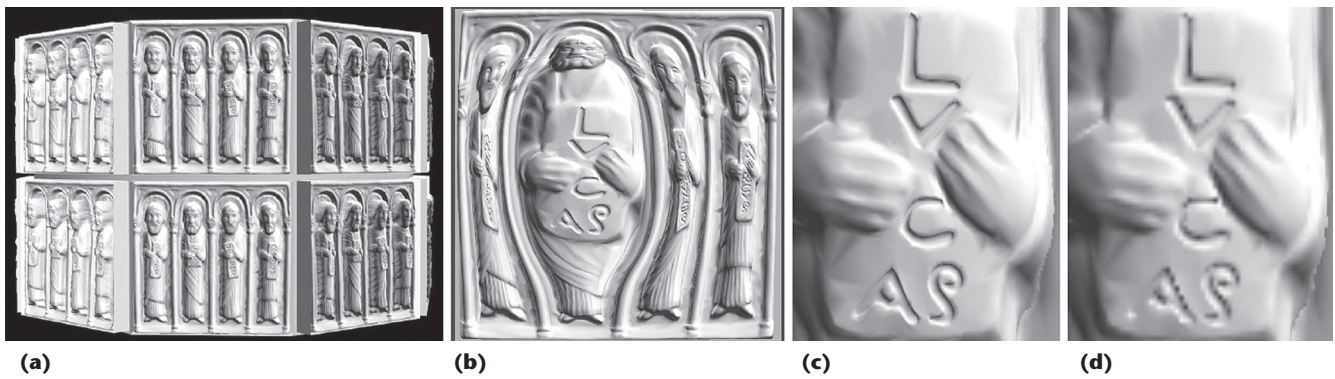


Figure 13. Using a CoNUS relief texture. (a) The object of interest. (b) A CoNUS relief texture that allocates more samples to the tablet. (c) A detail rendered with the CoNUS relief texture. (d) The same detail rendered with conventional relief textures of the same size.

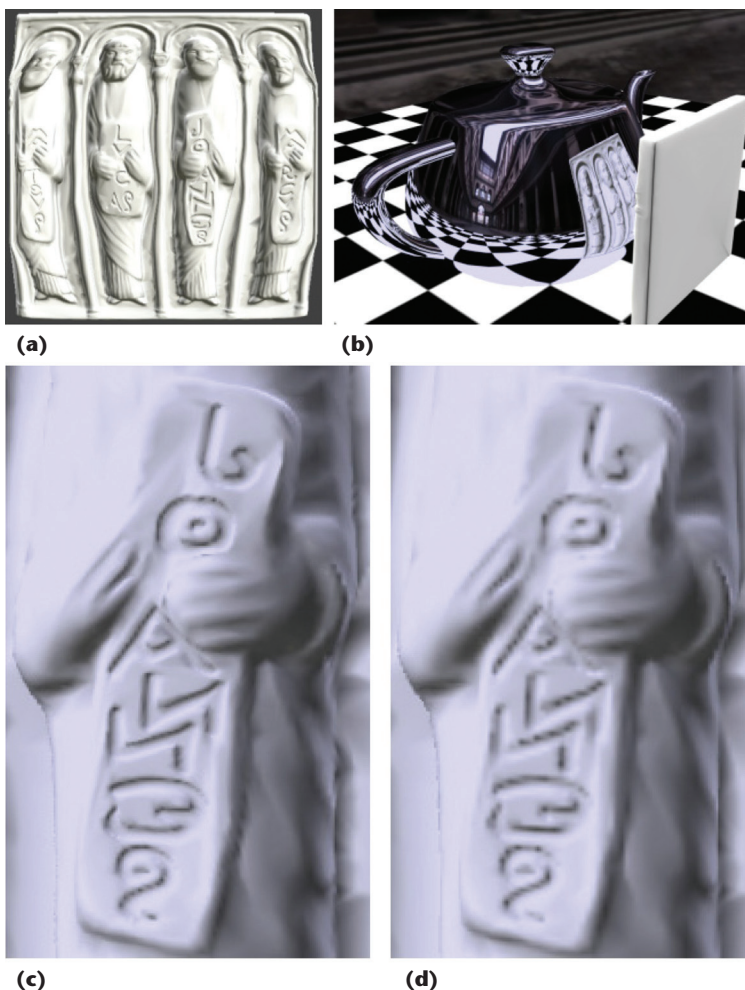


Figure 14. Using a CoNUS depth image. (a) A CoNUS depth image emphasizing all four engraved tablets (b) The scene setup. (c) Reflection details rendered with a CoNUS depth image. (d) Reflection details rendered with a conventional depth image.

depth images having a 1D intersection with a ray, at the cost of a slightly more complicated ray projection.

We've integrated CoNUS depth images into relief texture mapping (see Figure 13) and specular-

reflection rendering (see Figure 14), in which the CoNUS depth image intersects with eye rays and reflected rays, respectively. Relief texture mapping uses a depth image to enhance a surface with geometric detail. Specular-reflection rendering avoids environment-mapping approximation errors by modeling objects close to reflectors with depth images. Depth images accelerate these effects mainly because you can compute the intersection between a ray and a depth image faster than you can compute the intersection between a ray and the original geometry. A CoNUS depth image brings sampling flexibility without increasing the intersection's cost.

Quality. The sampling flexibility afforded by CoNUS depth images let us improve the clarity of the engraved tablets in Figure 13 and their reflection (see Figure 14).

Performance. For both conventional and CoNUS depth images, the performance bottleneck for relief texture mapping and specular-reflection rendering is computation of the intersection of the depth image and ray. Intersecting a ray with a CoNUS depth image incurs the additional cost of distorting a 2D point at every step along the ray. However, CoNUS distortion is fast; our average frame rate penalty was only 5 percent.

For applications in which the CoNUS depth image intersects with many rays, it might be advantageous to undistort the CoNUS depth image at the client into a higher-resolution conventional depth image, using the algorithm in Figure 8. This results in straight ray projections and avoids the cost of per-step distortion.

Limitations. CoNUS depth images inherit conventional depth images' occlusion limitations. The sampling tradeoff can lead to visual artifacts outside the ROIs.

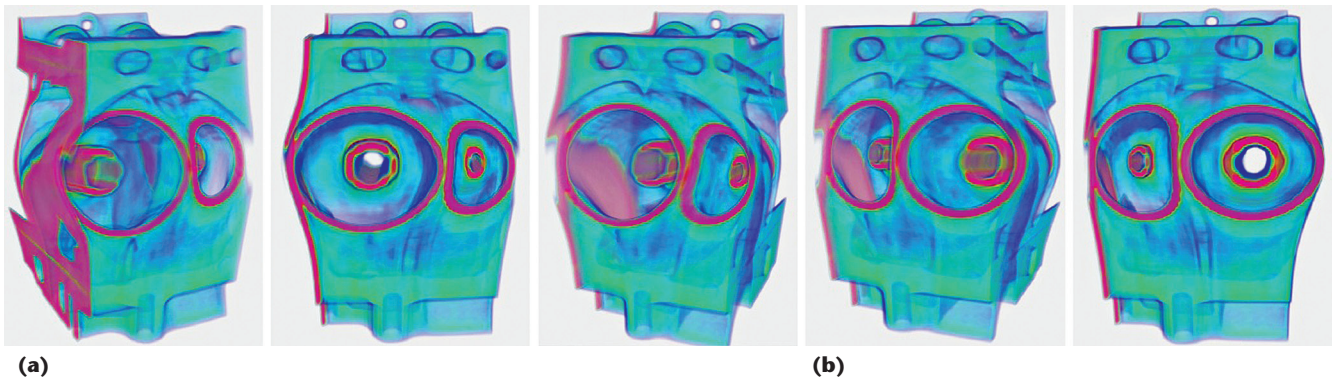


Figure 15. CoNUS focus-plus-context volume-rendering visualizations emphasizing the (a) left and (b) right cylinder housings of an engine. The FPC enables versatile focus-plus-context visualization that can handle any type of data and provides good control over the focus regions.

Focus-plus-Context Visualization

Our approach is well suited for focus-plus-context visualization for two reasons. First, it offers good control over the sampling rate, which allows precisely designing one or multiple focus regions. Second, as we mentioned before, CoNUS images can be rendered quickly, which supports dynamic scenes and interactive changing of focus region parameters. The CoNUS image is shown directly to the user, so no decoding is needed. The CoNUS image can be rendered efficiently from a variety of data, as we described before. The only remaining challenge is sampling-map construction.

Unlike the previous two applications, here we must construct the sampling map online, once for every output frame, which precludes using the mass-spring approach. We compose canonical circular sampling maps, one for every focus region. We’ve applied this approach for volume rendering (see Figure 15) in which the user manipulates the focus region and view parameters to examine a volume dataset. We’ve also applied it to a city scene modeled with triangle meshes (see Figure 16), where the focus regions track moving cars. We located the focus region by projecting the center of the tracked car in the output view.

Quality. In our approach, the focus regions have strong magnification and low distortion. As we mentioned before, their parameters can change; regions can merge and separate without abruptly changing the output visualization. Focus-plus-context visualization is particularly robust to undersampling outside the focus region. Users will likely focus on the region they selected as important, and the focus regions can shift interactively to visualize any region in more detail.

Performance. In our experiments, FPC volume rendering was on average 7 percent slower than

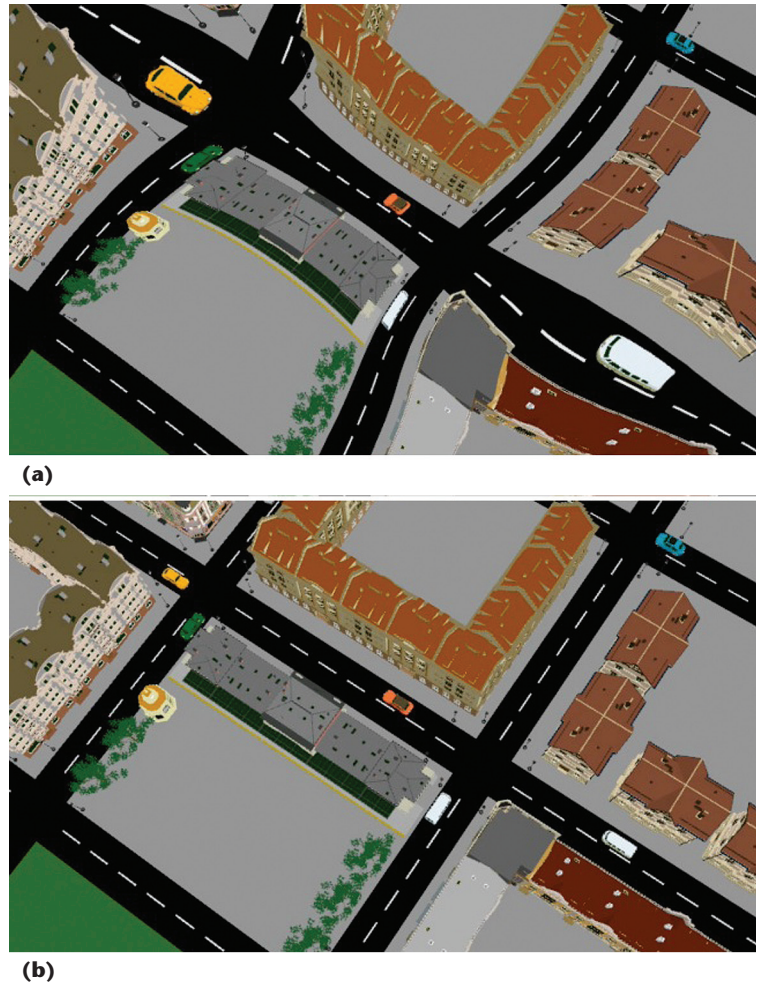


Figure 16. Visualization of a city scene modeled with triangle meshes. (a) A CoNUS visualization emphasizing the yellow and white cars. (b) A conventional image. The CoNUS images are rendered directly from the dataset using the FPC.

conventional volume rendering. Traversal of the volume dominates the cost of volume rendering by ray casting, so computing the perturbed rays for our approach doesn’t affect performance. We attribute the slight performance decrease to a

Related Work in Camera Models

First, we look at attempts to remove conventional images' uniform-sampling-rate constraint. Then, we review research on the areas in which we applied a flexible pinhole camera (FPC) to render coherent nonuniform sampling (CoNUS) images (see the main article).

Nonuniform Sampling Rate

Hierarchical spatial-partitioning approaches such as k -d trees improve representation efficiency by stopping subdivision in regions where data is sampled accurately. We could have defined the FPC sampling map with such an approach. These approaches support a wider range of sampling rates than our distorted-grid approach (see Figure 2 in the main article). However, they suffer from sampling-rate discontinuity, lack of contiguity, and more complex construction (rendering) and usage (lookup).

Images with a nonuniform sampling rate originally were a side effect of techniques for removing conventional images' field-of-view limitation. For spherical panoramas, the sampling-rate variation was an unwanted side effect; they were replaced by cube maps with a more uniform sampling rate. Recently, single-image panoramas have received renewed attention because programmable graphics hardware enables sampling patterns that avoid previous undersampling problems.¹ Researchers have also addressed conventional images' single-viewpoint limitation with camera-model-level innovations such as the general linear camera² and occlusion camera.³ Our approach complements these approaches, providing sampling-rate flexibility to panoramic and nonpinhole cameras.

Irregular sampling patterns have also occurred with image-based rendering by 3D warping⁴ and shadow antialiasing.⁵ Both approaches reproject depth images to novel views in which the forward-mapped samples are irregular. Our approach doesn't control sampling with sufficient granularity to sample the shadow map precisely at

the locations where the output image requires it, which would be needed to completely eliminate shadow aliasing. However, we could reduce shadow aliasing by using a CoNUS shadow map with a higher sampling rate in regions that are magnified in the output image.

The most general pinhole camera defines each ray independently with its own image plane point.⁶ Such a model theoretically has maximum generality, allowing for any sampling pattern given n rays, but has no practical use. First, the rays aren't organized in a 2D array, so the resulting image is an unsorted list of color samples that can't be easily displayed. Second, rendering such an image is expensive because it would require tracing each ray independently.

A practical implementation of the general pinhole camera restricts the sampling-rate variation to a rectangular region R of the image plane.⁶ A smaller rectangle r , concentric with R , provides a higher-resolution sampling of the scene. The region r is sampled with a planar pinhole camera and is thus distortion free (3D scene lines map to 2D image lines). This approach uses the region $R - r$ to transition from the low sampling rate outside R to the high sampling rate inside r . It chooses the sampling locations in $R - r$ with a quadratic or cubic function to achieve C^0 or C^1 continuity. It supports several disjoint regions of higher resolution.

The FPC is another specialization of the general pinhole camera model. Our approach has two fundamental advantages over the implementation in the previous paragraph. First, the FPC provides far greater flexibility in defining the sampling locations. Second, the FPC sampling map provides fine-grained control of the sampling rate while keeping constant the amortized cost of the fundamental image point distortion and undistortion operations. In contrast, a general planar pinhole camera with multiple rectangular regions of high resolution requires checking each region

larger output image footprint for the distorted volume and to more rays focusing on the center of the dataset, where volume-traversal distances are longer. The vertex distortion when rendering CoNUS images from triangle meshes had no measurable performance impact.

Limitations. Because our approach doesn't alleviate occlusions, tracked objects of interest can become hidden, and the user must change the view to reveal them. We'll examine changing the view automatically to keep tracked objects visible.

The sampling map is a powerful tool for assigning more pixels to some regions of the

image plane. For example, for the image in Figure 1, the maximum sampling-rate increase was 8.13 \times . We measured this by finding the sampling mesh's largest quadrilateral cell and dividing its area by an undistorted cell's area. The sampling map doesn't create new pixels—we increase the sampling rate by decreasing the sampling rate in less important regions.

For a sampling map of resolution $w_0 \times h_0$, with ROIs occupying k cells and a minimum sampling rate of the context regions of $c \times$, the upper bound for the sampling-rate increase is $z = w_0 h_0 (1 - c) / k + c$. For example, if $w_0 \times h_0 = 1,024$, $k = 64$, and $c = 1/2$, then $z = 8.5 \times$. If the application tolerates downsampling the context to $1/8$, z increases to 14.125 \times . If there's a single

for the distortion or undistortion of a point. This process doesn't scale with the number of regions.

Texture-mapping implementations have pursued non-uniform sampling through compression, atlas, enhancement with explicitly modeled high-frequency features (for example, edges), and distortion. We discuss only the last two approaches because they're the closest to our research. Textures enhanced with edges modeling shadow silhouettes⁷ or abrupt color changes⁸ are more robust to magnification. This approach is compatible with CoNUS textures. Edges derived from vector graphics primitives must undergo sampling-map distortion (see Figure 2 in the main article), and long edges must be split. For texture-derived edges, the CoNUS texture can be used directly. Space-optimized textures⁹ distort textures with a mechanism similar to our sampling map. However, our research extends nonuniform sampling to more types of data and applications.

Remote Visualization

As the size of acquired and computed datasets continues to increase, so will the importance of remote visualization of remote datasets for clients with no high-end storage or visualization capabilities. One approach reduces the dataset on the server to a size that can be transmitted to and visualized by the client. You can use many techniques for this, including data compression,¹⁰ feature extraction,¹¹ and level of detail.¹² Another approach computes the visualization at the server and sends images to the client.¹³ The client needs only a simple terminal that can display images, but network bandwidth limits the visualization resolution and frame rate.

A hybrid approach transfers from the server to the client images having more data than what's needed for the client's current frame. Such an enhanced image should be sufficient for a quality reconstruction of a sequence of frames at the client, without additional data from the

server. Researchers have used images enhanced with per-pixel depth¹⁴ and additional samples at the center⁶ to allow translating and zooming in at the client.

Another hybrid approach transfers CoNUS images. A CoNUS image that samples known regions of interest (ROIs) in greater detail anticipates the user's intention to zoom in on those regions. A CoNUS height field that samples the ground plane orthogonally, yet at a higher rate close to the user, supports six-degree-of-freedom navigation at the client in the current view's neighborhood.

Accelerating Depth Image Rendering

Depth images are powerful geometry approximations used for acceleration in many contexts. However, we limit this discussion to relief texture mapping and specular reflection, which we used in the main article to illustrate our approach's benefits.

Relief texture mapping adds geometric detail to surfaces. It produces correct silhouettes and correct interactions between relief geometry and other relief and nonrelief geometry (for example, intersections and casting and receiving shadows).¹⁵ The relief texture is a depth image attached to a base box. Rendering the box triggers the computation of an eye ray–depth image intersection at every pixel covered by the box. The intersection computation projects the ray onto the depth image and follows the ray projection until it finds the first intersection.

Specular reflection is challenging for the feed-forward 3D graphics pipeline because computing the image plane projection of reflected vertices isn't easy. We group specular-reflection rendering techniques into four categories: ray tracing,¹⁶ approximations of the projection of reflected vertices,¹⁷ image-based rendering,¹⁸ and approximations of the reflected scene. We discuss only the fourth category because CoNUS specular-reflection rendering falls in it.

Cont. on page 40


ROI that fits in one cell (that is, $k = 1$), then even for a negligible downsampling of the context regions by $c = 0.95\times$, the ROI's sampling rate can reach $z = 52.15\times$.

Possible future research includes

- exploring other uses of CoNUS images (for example, for geometric simplification and accelerating other rendering effects),
- investigating the cost–benefit tradeoff of higher-order interpolation of the sampling map to achieve C^1 sampling-rate continuity, and
- developing automatic sampling-map constructors.

We're particularly interested in tightly coupling our approach with automatic techniques for de-

termining what to sample in more detail, such as automatic geometric-complexity analysis, object recognition, eye tracking, and saliency maps.

We foresee that FPC-rendered CoNUS images will have wide applicability because they're compatible with virtually all contexts in which images are used. 

Acknowledgments

We thank the anonymous reviewers and associate editor, who helped improve this article. We also thank the US National Science Foundation for supporting Voicu Popescu and Jian Cui's research through grant 1217215, as well the National Natural Science Foundation of China for supporting Lili Wang's

Related Work in Camera Models (Cont.)

Environment mapping performs the most drastic approximation; it assumes that the reflected scene is infinitely far from the reflector.¹⁹ Environment-mapped reflections are incorrect for objects close to the reflector. Approximating these objects with billboards or depth images¹⁸ improves reflection accuracy. Using CoNUS depth images as relief textures or approximations of reflected objects provides sampling flexibility without considerably increasing the cost of ray or depth image intersection.

Focus-plus-Context Visualization

The visualization of complex scenes can benefit from highlighting the scene region that's most important in the application's context. Such focus-plus-context visualization has a multiple-stage pipeline, including

- finding the ROIs;
- finding the best viewpoint for an ROI; and
- highlighting the ROI by assigning it a salient color, assigning it more pixels, or managing occlusions through cut-away, transparency, or nonpinhole-camera techniques.

For example, the best viewpoint can be found automatically through analysis of the region feature distribution in an information-theoretic framework.²⁰ Stefan Bruckner and his colleagues surveyed state-of-the-art methods for the focus-plus-context pipeline stages.²¹ Here, we only discuss highlighting the ROI by allocating more pixels to it, which is how the FPC contributes to focus-plus-context visualization.

An important challenge stems from the fact that displays have a uniform pixel resolution (except for special focus-plus-context screens²²). So, a focus-plus-context

image can't be displayed directly and must be mapped to displays with uniform resolution by introducing distortions between the focus and context regions. Focus-plus-context visualization is typically applied to 2D data (for example, hierarchies,²³ graphs,²⁴ and maps²⁵). You can apply it to 3D data by either distorting the dataset and visualizing it with a conventional camera²⁶ or distorting the camera model.^{6,27} FPC focus-plus-context visualization falls in the second category. Like the general pinhole camera, the volume lens defines one or a few ROIs with higher resolution.²⁷ The employed ray perturbation doesn't provide closed-form projection, and the method is restricted to volume rendering and ray tracing.

References

1. J.-D. Gascuel et al., "Fast Nonlinear Projections Using Graphics Hardware," *Proc. 2008 Symp. Interactive 3D Graphics and Games*, 2008, pp. 107–114.
2. J. Yu and L. McMillan, "General Linear Cameras," *Computer Vision—ECCV 2004*, LNCS 3022, Springer, 2004, pp. 14–27.
3. C. Mei, V. Popescu, and E. Sacks, "The Occlusion Camera," *Computer Graphics Forum*, vol. 24, no. 3, 2005, pp. 335–342.
4. L. McMillan and G. Bishop, "Plenoptic Modeling: An Image-Based Rendering System," *Proc. Siggraph*, 1995, pp. 39–46.
5. G.S. Johnson et al., "The Irregular Z-buffer: Hardware Acceleration for Irregular Data Structures," *ACM Trans. Graphics*, vol. 24, no. 4, 2005, pp. 1462–1482.
6. V. Popescu et al., "The General Pinhole Camera: Effective and Efficient Nonuniform Sampling for Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 16, no. 5, 2010, pp. 777–790.
7. P. Sen, M. Cammarano, and P. Hanrahan, "Shadow Silhou-

research through projects 61272349, 61190121, and 61190125.

Reference

1. F. Policarpo and M.M. Oliveira, "Relief Mapping of Non-Height-Field Surface Details," *Proc. 2006 Symp. Interactive 3D Graphics and Games*, 2006, pp. 55–62.

Voicu Popescu is an associate professor in Purdue University's Computer Science Department. His research interests include computer graphics, computer vision, and visualization. His current projects include camera model design, remote visualization, aggressive and exact visibility computation, and computer graphics applications in education. Popescu received a PhD in computer science from the University of North Carolina at Chapel Hill. Contact him at popescu@purdue.edu.

Bedrich Benes is an associate professor in Purdue University's Department of Computer Graphics Technology, a Purdue Faculty Scholar, and a director of Purdue's High Performance Computer Graphics Laboratory. His research is primarily in procedural modeling, real-time rendering, and 3D computer graphics. Benes received a PhD in computer science from the Czech Technical University. Contact him at bbenes@purdue.edu.

Paul Rosen is a research assistant professor at the University of Utah with appointments in the Scientific Computing and Imaging Institute and the School of Computing. His research interests include scientific visualization, such as vector field and uncertainty visualization, and information visualization, such as parameter space and software performance visualization. Rosen received a PhD in computer science from Purdue University. Contact him at prosen@sci.utah.edu.

Jian Cui is a PhD candidate in computer science at Purdue University. His research interests span computer graph-

- ette Maps," *ACM Trans. Graphics*, vol. 22, no. 3, 2003, pp. 521–526.
8. G. Ramanarayanan, K. Bala, and B. Walter, *Feature-Based Textures*, tech. report, Cornell Univ., 2004.
 9. L. Balmelli, G. Taubin, and F. Bernardini, "Space-Optimized Texture Maps," *Computer Graphics Forum*, vol. 21, no. 3, 2002, pp. 411–420.
 10. L. Lippert, M.H. Gross, and C. Kurmann, "Compression Domain Volume Rendering for Distributed Environments," *Computer Graphics Forum*, vol. 16, no. 3, 1997, pp. C95–C107.
 11. Y. Livnat, S.G. Parker, and C.R. Johnson, "Fast Isosurface Extraction Methods for Large Image Data Sets," *Handbook of Medical Imaging*, Academic Press, 2000, pp. 731–745.
 12. S.P. Callahan et al., "Interactive Rendering of Large Unstructured Grids Using Dynamic Level-of-Detail," *Proc. IEEE Visualization 2005 (VIS 05)*, 2005, pp. 199–206.
 13. S. Stegmaier, M. Magallón, and T. Ertl, "A Generic Solution for Hardware-Accelerated Remote Visualization," *Proc. 2002 Symp. Data Visualisation (VISSYM 02)*, 2002, pp. 87ff.
 14. E.J. Luke and C.D. Hansen, "Semotus Visum: A Flexible Remote Visualization Framework," *Proc. IEEE Visualization 2002 (VIS 02)*, 2002, pp. 61–68.
 15. F. Policarpo and M.M. Oliveira, "Relief Mapping of Non-Height-Field Surface Details," *Proc. 2006 Symp. Interactive 3D Graphics and Games*, 2006, pp. 55–62.
 16. T. Whitted, "An Improved Illumination Model for Shaded Display," *ACM Siggraph 2005 Courses*, 2005, p. 4.
 17. E. Ofek and A. Rappoport, "Interactive Reflections on Curved Objects," *Proc. Siggraph*, 1998, pp. 333–342.
 18. L. Szirmay-Kalos et al., "Approximate Ray-Tracing on the GPU with Distance Impostors," *Computer Graphics Forum*, vol. 24, no. 3, 2005, pp. 695–704.
 19. J.F. Blinn and M.E. Newell, "Texture and Reflection in Computer Generated Images," *Comm. ACM*, vol. 19, no. 10, 1976, pp. 542–547.
 20. I. Viola et al., "Importance-Driven Focus of Attention," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, 2006, pp. 933–940.
 21. S. Bruckner et al., "Illustrative Focus+Context Approaches in Interactive Volume Visualization," *Scientific Visualization: Advanced Concepts*, Dagstuhl Publishing, 2010, pp. 136–162.
 22. P. Baudisch, N. Good, and P. Stewart, "Focus plus Context Screens: Combining Display Technology with Visualization Techniques," *Proc. 14th Ann. ACM Symp. User Interface Software and Technology (UIST 01)*, 2001, pp. 31–40.
 23. J. Lamping and R. Rao, "The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies," *J. Visual Languages & Computing*, vol. 7, no. 1, 1996, pp. 33–55.
 24. N. Wong, S. Carpendale, and S. Greenberg, "EdgeLens: An Interactive Method for Managing Edge Congestion in Graphs," *Proc. 2003 IEEE Symp. Information Visualization (INFOVIS 03)*, 2003, pp. 51–58.
 25. E. Pietriga and C. Appert, "Sigma Lenses: Focus-Context Transitions Combining Space, Time and Translucence," *Proc. 2008 SIGCHI Conf. Human Factors in Computing Systems (CHI 08)*, 2008, pp. 1343–1352.
 26. M.S.T. Carpendale, D.J. Cowperthwaite, and F.D. Fracchia, "Distortion Viewing Techniques for 3-Dimensional Data," *Proc. 1996 IEEE Symp. Information Visualization (InfoVis 96)*, 1996, pp. 46–53.
 27. L. Wang et al., "The Magic Volume Lens: An Interactive Focus+Context Technique for Volume Rendering," *Proc. IEEE Visualization 2005 (VIS 05)*, 2005, pp. 367–374.

ics and computer vision, focusing on image generalization through camera model design to overcome conventional images' single-viewpoint and uniform-sampling-rate limitations. Cui received his BS in computer science from the Harbin Institute of Technology. Contact him at cui9@purdue.edu.

Lili Wang is an associate professor at Beihang University's School of Computer Science and Engineering and a researcher with the State Key Laboratory of Virtual Reality Technology and Systems. Her interests include real-time rendering, realistic rendering, global illumination, and soft-shadow and texture synthesis. Wang received a PhD in computer science from Beihang University. Contact her at wanglily@buaa.edu.cn.

cn Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

Take the CS Library wherever you go!

IEEE Computer Society magazines and Transactions are now available to subscribers in the portable ePub format.

Just download the articles from the IEEE Computer Society Digital Library, and you can read them on any device that supports ePub. For more information, including a list of compatible devices, visit

www.computer.org/epub

IEEE IEEE computer society