Interactive Visibility Retargeting in VR using Conformal Visualization

Kaloian Petkov, *Student Member, IEEE,* Charilaos Papadopoulos, *Student Member, IEEE,* Min Zhang, Arie E. Kaufman, *Fellow, IEEE* and Xianfeng Gu, *Member, IEEE*

Abstract—In Virtual Reality, immersive systems such as the CAVE provide an important tool for the collaborative exploration of large 3D data. Unlike head-mounted displays, these systems are often only partially immersive due to space, access or cost constraints. The resulting loss of visual information becomes a major obstacle for critical tasks that need to utilize the users' entire field of vision. We have developed a conformal visualization technique that establishes a conformal mapping between the full 360° field of view and the display geometry of a given visualization system. The mapping is provably angle-preserving and has the desirable property of preserving shapes locally, which is important for identifying shape-based features in the visual data. We apply the conformal visualization to both forward and backward rendering pipelines in a variety of retargeting scenarios, including CAVEs and angled arrangements of flat panel displays. In contrast to image-based retargeting approaches, our technique constructs accurate stereoscopic images that are free of resampling artifacts. Our user study shows that on the visual polyp detection task in Immersive Virtual Colonoscopy, conformal visualization leads to improved sensitivity at comparable examination times against the traditional rendering approach. We also develop a novel user interface based on the interactive recreation of the conformal mapping and the real-time regeneration of the view direction correspondence.

Index Terms—Virtual Reality, Conformal Visualization, Ricci flow, GPU, Immersive Cabin, CAVE, partially immersive.

1 Introduction

A number of visualization technologies have been developed for the immersive exploration of complex large-scale data. Prime examples are the CAVE [1] and Head-Mounted Displays (HMD) which provide a much larger field of view into a virtual environment compared to traditional desktop systems, and also use stereoscopic pairs of images to improve the perception of spatial relationships. While HMDs allow for arbitrary views in the virtual world, they are usually bulky, wired and easily lead to eye fatigue. At the same time, CAVEs provide a much more natural visualization without the need for a virtual avatar. Building a fully enclosed CAVE, however, remains a difficult task. Although such installations exist [2], they present an engineering challenge in terms of cost, facility access, as well as head and gesture tracking.

Immersion in the virtual data is a function of different factors, such as sensory perceptions, interaction techniques and realism of the visualization. We focus on *visual immersion*, defined by the field of view coverage afforded by the physical visualization platform, as well as the coverage of the virtual scene provided by the visualization software. The disadvantage of partially-immersive environments, such as CAVEs with at least one missing display surface, is that important visual information may be lost. While many applications may tolerate one or more missing projection screens, this partial loss of visual context adversely affects the general navigation capabilities of the user and becomes a critical issue in the exploration of medical data.

We have developed a visualization approach that utilizes conformal mapping to modify scene geometries or viewing rays at runtime, depending on the rendering modality. As a result, the full virtual environment can be displayed on a partially-immersive visualization platform, for example a 5-sided CAVE, without the artifacts typically associated with image-based retargeting approaches. In mathematics, the conformal map is an angle preserving function that describes a mapping between two Riemannian surfaces [3]. Intuitively, it allows us to map the geometry of the fully-immersive 6-sided CAVE to an arbitrary configuration of display surfaces that is topologically equivalent to a disk, such as a 5-sided CAVE or a non-planar arrangement of flat panel displays. This mapping is then used to transform the viewing directions during rendering with ray tracing, for example. The main advantage of using a conformal map to define the transformation is the guarantee that shapes will be preserved locally even though distances will not be. This is particularly beneficial for the exploration of medical data, such as in Virtual Colonoscopy (VC) where potentially cancerous polyps are detected by the radiologist based on their shape.

The following are the specific contributions of our work:

- We develop a conformal visualization technique that establishes a shape-preserving transformation between the 360° field of view and an arbitrary display configuration. We demonstrate applications to visibility retargeting for CAVEs with 3, 4 and 5 display surfaces, as well as for angled arrangements of flat panel displays.
- The conformal transformation of the viewing directions is applied during rendering with rasterization, direct volume rendering and ray-tracing. Compared to the traditional

- *Authors are affiliated with Department of Computer Science, Stony Brook University, Stony Brook, NY 11794.*
E-mail: [kpetkov,cpapadopoulo,mzhang,ari,gu]@cs.stonybrook.edu

image-based retargeting approaches, our technique does not introduce resampling artifacts and constructs accurate stereoscopic image pairs for all three rendering modalities using a unified transformation definition.

- Our system computes the conformal mapping at interactive speeds and regenerates the viewing ray correspondence on the GPU at real-time speeds allowing for a novel user interface for the dynamic manipulation of the visibility. The performance penalty for applying the conformal transformation during rendering is approximately 1% for ray tracing and volume rendering.
- Our approach is applied to the visualization of both mesh and volume data under 3 different retargeting scenarios. Our user study shows an improvement in the task of visual polyp detection in phantom colonoscopy datasets during Immersive Virtual Colonoscopy.

2 Related Work

Immersive visualization systems allow the user to explore data in novel ways that go beyond the standard $2D$ images on a workstation. These platforms provide a superior depiction of the information via a significantly wider field of view and enhanced depth and shape perception. The first such environment, the CAVE [1], offers an immersive experience using back-projected images on 3 walls and front projection on the floor. Other display arrangements have been proposed, including the 5-sided Immersive Cabin (IC) [4] and the 6-sided CAVE [2]. However, building fully-immersive facilities is expensive and introduces many challenges in terms of head tracking, sound systems and even air circulation. Our conformal visualization system uses the discrete Ricci flow [5] to compute the mapping between the full 360° spherical field of view and the partially-immersive platforms. The main advantage is that the local shape of the projected scene geometry and the features in volume data are preserved under the conformal transformation. The Ricci flow method has found a broad range of applications in the graphics and visualization fields, including optimal surface parameterization [6], [7], shape analysis [8] and surface matching [9].

Least Square Conformal Mapping (LSCM) is an alternative heuristic technique for computing a conformal mapping [10]. Unlike Ricci flow, which automatically produces seam-free global conformal parameterizations, LSCM and other computational approaches may require special algorithms or heuristic inputs when dealing with certain topologies [5]. The main disadvantage of LSCM is that it cannot control the boundary of the region. In our case, we require a mapping between the visibility boundary of the CAVE and the unit circle. Therefore, LSCM is not a viable technique for our application. Furthermore, unlike the approach by Springborn et al. [11], the Ricci flow is a stable algorithm with a theoretical assurance of convergence.

Raskar et al. have used the LSCM to display the output of an ad-hoc cluster of projectors onto an arbitrary set of surfaces [12]. Our conformal visualization solves a different problem: given a set of display surfaces and a visualization system that produces the correct projections (e.g., partially-immersive CAVE or an arrangement of LCD monitors), we compute a mapping between the full visibility sphere and the visibility area provided by the display system. The user is able to visualize the entire data at once on the target platform, while the properties of the mapping guarantee that shapes are preserved locally. Our application of conformal mapping is fundamentally different and requires the use of advanced rendering techniques beyond texture projection.

Focus and Context (F+C) techniques such as magic lenses [13], [14], [15] are designed for a single projection surface and do not translate directly to immersive environments with multiple non-planar displays. Illustrative deformations for data exploration have been proposed [16] which could be used to warp data that lies around the CAVE volume, however they do not provide any guarantee in regard to shape preservation. The technique presented by Lorenz and Döllner [17] handles piecewise approximation of non-planar perspective projections on the GPU. Non-planar projections can be used to define a projection surface that ”wraps” around the CAVE, including part of the ceiling, and thus recovering the non-visualized sections of the data. The technique can be applied in either image-space or geometry-space, however in the first case the sampling artifacts can impact visual quality significantly [18], while the geometry approach does not scale well with mesh density [17]. In contrast, our conformal visualization technique generates the final images directly onto the display surfaces or partially immersive environments with minimal overhead for volume rendering and ray tracing, and competitive performance for rasterization. Our GPU implementation of a dynamic visibility manipulation technique further provides a natural and efficient F+C interface for immersive environments.

Our conformal visualization is applicable to a number of visualization tasks, however in the user study of its effectiveness we focus primarily on medical visualization. Virtual Colonoscopy (VC) has been established as a non-invasive alternative to traditional Optical Colonoscopy (OC) for cancer screening [19], [20]. A VC session involves the acquisition of Computed Tomography (CT) scans of the patient’s abdomen and the extraction and visualization of the colon surface via segmentation and volume rendering techniques. Traditional VC covers only about 91% of the colon surface after full navigation in both the antegrade and the retrograde directions [21] and the percentage is significantly lower for a single direction. One shortcoming of existing partially-immersive configurations for Immersive Virtual Colonoscopy (IVC) is that the missing display surfaces may hide a significant amount of information. While this may be acceptable in certain applications, it becomes a critical issue for the exploration of medical data. Our approach allows the radiologist to examine the entire surface of the colon in a single pass and ensures that the shape of any colon polyps, which is crucial for the detection of colon cancer, is preserved in the conformal visualization.

The current work presents a number of significant improvements over our previous implementation of conformal visualization [22]. We have designed a more general conformal mapping pipeline based around templates for the source and the target visualization platforms. Our template for the

5-sided CAVE produces conformal maps that are virtually identical to the manually generated version from our previous work and we have also designed templates for the 4-sided CAVE, the 3-sided CAVE and for tiled arrangements of displays. Our new conformal visualization pipeline is also significantly more efficient and allows for regeneration of the conformal maps at interactive rates based on tracking data. We have also developed a novel user interface for realtime visibility manipulation. Finally, the rasterization pipeline has been redesigned to take advantage of the tessellations stage available in modern DirectX 11-compatible GPUs. Compared to the previous implementation, our current system does not introduce rendering artifacts during the application of the conformal map and supports scenes with sub-optimal initial tessellation.

3 Theoretical Background

We start by briefly describing the theoretical foundations of our conformal visualization approach. The overall goal is to develop a 1-to-1 mapping between two surfaces that represent the source and target visibility geometries. In both cases, the geometry is constructed as the section of the field of view that a visualization system can cover from a certain viewpoint. For our application, the source geometry is most often defined as the complete unit sphere around the viewpoint.

According to the Riemann mapping theorem, simply connected surfaces with a single boundary can be mapped onto the planar disk, such that the mapping is *angle-preserving*. Locally, the mapping is only a scaling and therefore it is also shape-preserving. This is advantageous in our rendering technique as it maps the viewing directions from the 6 original projections in the fully-enclosed CAVE to the 5-sided CAVE while preserving the local shapes. Our computational algorithm is based on the surface Ricci flow theorem [23].

3.1 Conformal Mapping

Let S_1 and S_2 be two surfaces with Riemannian metrics $\mathbf{g}_1$ and $\mathbf{g}_2$, and let $\phi : (S_1, \mathbf{g}_1) \to (S_2, \mathbf{g}_2)$ be a homeomorphism between them. We say ϕ is *conformal*, if the pull back metric tensor induced by ϕ on the source differs from the original metric by a scalar:

$$\phi^* \mathbf{g}_2 = e^{2\lambda} \mathbf{g}_1,$$

where $\lambda : S_1 \to \mathbb{R}$, is a function. The following Riemann mapping theorem plays a fundamental role in the current work:

Theorem 3.1 (Riemann Mapping) *Suppose a surface S is simply connected with a Riemannian metric. There exist conformal mappings $\phi : S \to \mathbb{D}$, where D is the unit disk on the complex plane and all such mappings differ by a Möbius transformation.*

A Möbius transformation of a point z on the complex plane is given by

$$z \to e^{i\theta} \frac{z - z_0}{1 - \bar{z}_0 z},$$

where θ and $z_0 \in \mathbb{C}$ are constants.

In order to compute the mapping ϕ, one can compute the pull back metric first, which can be achieved by the surface Ricci flow.

3.2 Ricci Flow

A surface Ricci flow is the process used to deform the Riemannian metric of the surface. The deformation is proportional to the Gaussian curvature so that the curvature evolves in a manner similar to heat diffusion. In mathematics, it is a powerful tool for finding a Riemannian metric satisfying the prescribed Gaussian curvature. Chow and Luo have described the theoretic foundation for the discrete Ricci flow on surfaces [23], and Jin et al. have developed an efficient computational algorithm [5].

Let $\Sigma = (V, E, F)$ be a triangular mesh embedded in $\mathbb{R}^3$, where V, E and F are respectively the sets of vertices, edges, and faces. A *discrete Riemannian metric* on Σ is a piecewise constant metric with cone singularities at the vertices. The edge lengths are sufficient to define a discrete Riemannian metric,

$$l : E \to \mathbb{R}^+, \tag{1}$$

as long as, for each face $f_{ijk} = [v_i, v_j, v_k], f_{ijk} \in F$, the edge lengths satisfy the triangle inequality: $l_{ij} + l_{jk} > l_{ki}$.

For simplicity, we use e_i to denote the edge against the vertex v_i, namely $e_i = [v_j, v_k]$, and l_i is the edge length of e_i in triangle f_{ijk}. The cosine laws are given by:

$$l_k^2 = l_i^2 + l_j^2 - 2 l_i l_j \cos\theta_k \tag{2}$$

The *discrete Gaussian curvature* K_i on a vertex $v_i \in \Sigma$ can be computed as the angle deficit,

$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \notin \partial\Sigma \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \in \partial\Sigma \end{cases} \tag{3}$$

where θ_i^{jk} represents the corner angle attached to vertex v_i in all the face $f_{ijk} \in F$ that share vertex v_i, and $\partial\Sigma$ represents the boundary of the mesh Σ.

The circle packing metric (see Fig. 1) was introduced to approximate the conformal deformation of metrics [3], [24]. The function $\Gamma : V \to \mathbb{R}^+$ assigns a radius γ_i to each vertex $v_i \in V$. Similarly, the weight function $\Phi : E \to [0, \frac{\pi}{2}]$ associates the acute angle Θ_{ij} with each edge $e_{ij} \in E$, where Θ_{ij} is defined as the angle of intersection of the circles at vertices v_i and v_j. The circle packing metric for Σ is the pair (Γ, Φ).

Let $u : V \to \mathbb{R}$ be the *discrete conformal factor*, which measures the local area distortion, where $u_i = \log \gamma_i$ for each vertex $v_i \in V$. Then, the discrete Ricci flow is described by the following:

$$\frac{du_i(t)}{dt} = (\bar{K}_i - K_i), \tag{4}$$

where $\bar{K}_i$ is the prescribed curvature at vertex v_i. The discrete Ricci flow can also be formulated in the variational setting, namely, it is the negative gradient flow of some special energy form:

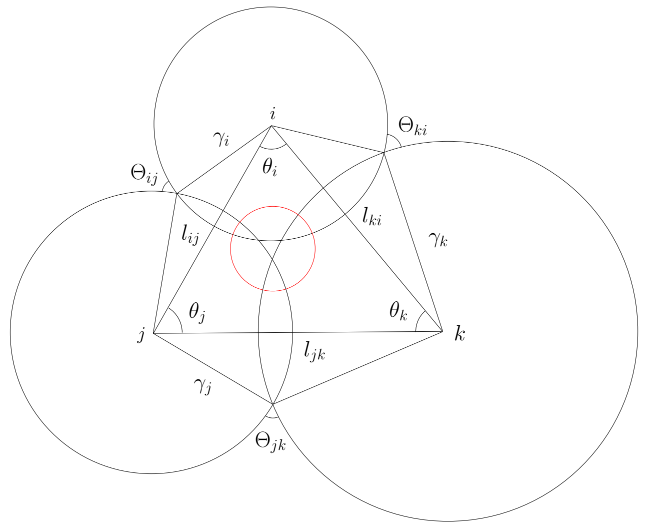


Fig. 1: Circle Packing Metric.

$$f(\mathbf{u}) = \int_{\mathbf{u_0}}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i, \quad (5)$$

where $\mathbf{u_0}$ is an arbitrary initial metric. The integration above is well-defined, and it is called the Ricci energy. Then, the discrete Ricci flow is the negative gradient flow of the discrete Ricci energy and the discrete metric which induces $\bar{\mathbf{k}} = (\bar{K}_1, \bar{K}_2, ..., \bar{K}_n)^T$ is the minimizer of that energy.

Computing the desired metric with prescribed curvature $\bar{\mathbf{k}}$ is equivalent to minimizing the discrete Ricci energy, which is strictly convex (namely, its Hessian is positive definite). The global minimum exists uniquely, corresponding to the metric $\bar{\mathbf{u}}$, which induces $\bar{\mathbf{k}}$. The discrete Ricci flow converges to this global minimum [23] and the global conformal parameterization for Σ can be computed in an automated and robust fashion.

3.3 Discrete Ricci flow

Suppose Σ is a triangle mesh embedded in $\mathbb{R}^3$. We associate each vertex v_i with a circle (v_i, γ_i) where γ_i equals the minimal length of any edge in the immediate neighborhood of v_i. Then we compute the intersection angle Θ_{ij} such that the circle packing metric is as close to the induced Euclidean metric as possible.

We compute the curvature at each vertex v_i and adjust the conformal factor u_i in proportion to the difference between the target curvature $\bar{K}_i$ and the current curvature K_i. Then, we update the metric, recompute the curvature, and repeat this procedure until the difference between the target curvature and the current curvature is less than the given threshold. Alg. 1 summarizes the computational steps and more details can be found in the work of Jin et al. [5].

3.4 Riemann Mapping

Fig. 2 illustrates the algorithm for computing the Riemann mapping. We remove a face from the mesh Σ to convert it to a topological cylinder (Fig. 2a), resulting in the creation of a new boundary γ_2. The target curvature for both the interior and the boundary vertices is set to zero. The Ricci flow described in Alg. 1 produces a flat cylinder that is periodically embedded in the complex plane (Fig. 2b). Each period of the embedding is a rectangle and the original boundaries γ_1 along the cut face and γ_2 are aligned with the imaginary axis in the complex plane. The cylinder is then mapped to the unit disk with the hole in the center of the image by the exponential map e^z (Fig. 2c). Fig. 2d illustrates the mapping by texture mapping a checkerboard pattern back on the cut mesh. As a final step, the removed face is inserted back into the mesh, yielding the conformal mapping on the original mesh.

Algorithm 1 Discrete Ricci Flow

```
Require: Triangular mesh $\Sigma$, target curvature for each vertex $\bar{K}_i$, error threshold $\varepsilon$.
Ensure: Discrete metric (edge lengths) satisfying the target curvature.
1:  $\mathbf{u} = [u_i], \mathbf{v} = [v_i]$ for $\mathbf{u}, \mathbf{v} \leftarrow 0$
2:  while true do
3:      Compute edge length $l_{ij}$ for edge $[v_i, v_j]$:
        $l_{ij} = e^{u_i} + e^{u_j} + 2\cos\phi_{ij} e^{u_i + u_j}$
4:      Compute the corner angle $\theta_i^{jk}$ in triangle $[v_i, v_j, v_k]$:
        $\theta_i^{jk} = \cos^{-1} \frac{l_{ij}^2 + l_{ki}^2 - l_{jk}^2}{2 l_{ij} l_{ki}}$
5:      Compute the curvature $K_i$ at $v_i$:
        $K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \notin \partial\Sigma \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \in \partial\Sigma \end{cases} \quad \mathbf{K} = [K_i]$
6:      if max $|\bar{K}_i - K_i| < \varepsilon$ then
7:          return the discrete metric $l_{ij}$
8:      end if
9:      Update $\mathbf{u}$:
        Compute the Hessian Matrix $H$, $H_{ij} = \frac{\partial K_i}{\partial u_j}$
        $\mathbf{u} \leftarrow \mathbf{u} - H^{-1}(\bar{\mathbf{K}} - \mathbf{K})$
10: end while
```

3.5 Practical Application

The previous sections introduce the theoretical foundation of the Discrete Ricci flow algorithm, which is the base algorithm in our conformal visualization technique. Using the 5-sided CAVE as an example, we consider all the possible view directions mapped onto a unit sphere for a reference position at the center of the environment. This sphere is cut at the position that corresponds to the center of the missing surface (Fig. 3c) and towards the four corners. The top edges of the CAVE are also mapped to the sphere (Fig. 3a) and they define the visibility boundary $\partial\Sigma$ for the central CAVE position. We apply the Discrete Ricci flow to compute the conformal maps for both spheres, which are then aligned to provide a 1-to-1 mapping between the two surfaces (see Fig. 3b and Fig. 3d). Using this mapping, the full set of viewing directions defined over the 6-sided CAVE is then projected onto the geometry corresponding to the 5-sided CAVE. We then encode the viewing directions into a cube map which can be sampled efficiently during rendering. Although our rendering framework can utilize conformal maps produced

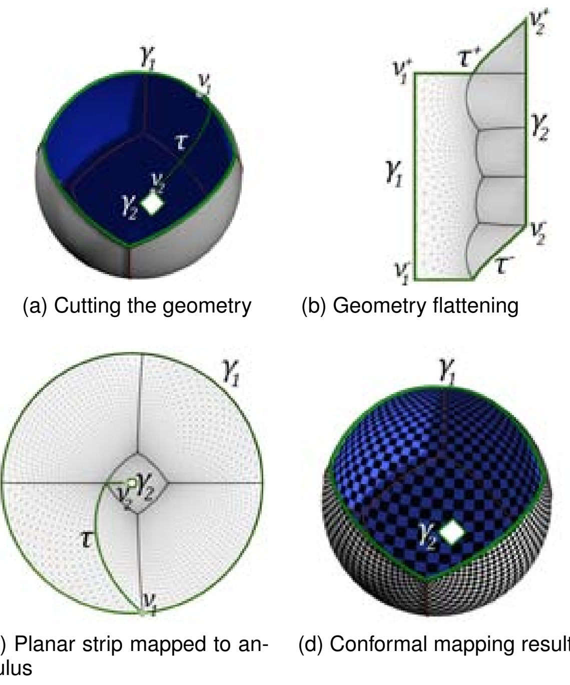


Fig. 2: Riemann Mapping Algorithm

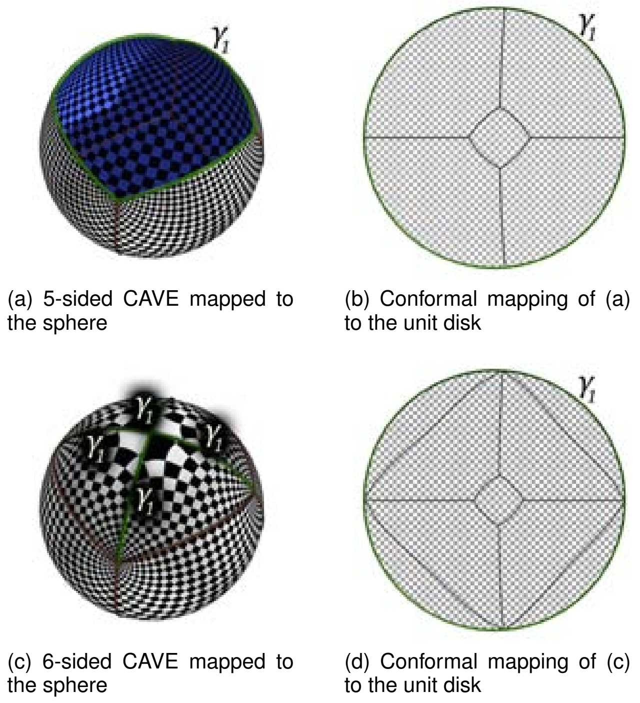

γ_1
(a) 5-sided CAVE mapped to the sphere
γ_1
(b) Conformal mapping of (a) to the unit disk
γ_1
γ_1
γ_1
γ_1
(c) 6-sided CAVE mapped to the sphere
γ_1
(d) Conformal mapping of (c) to the unit disk

Fig. 3: Algorithm for conformal mapping between a 5-sided CAVE and a 6-sided CAVE.

with any computational algorithms, the automation, robustness and efficiency of the Ricci flow provide an advantage for quickly generating the mappings between the different display topologies.

4 Implementation Details

Our conformal visualization utilizes an efficient pipeline for generating the conformal mapping at interactive speeds. The input is a user-specified rendering target, such as an n-sided CAVE or an arrangement of displays, as well as a set of parameters that control the accuracy of the generated maps.

4.1 Mesh Templates

Our mesh processing toolkit utilizes a half-edge data structure to explicitly represent the mesh connectivity information. We start by generating the templates for the source and target visibility meshes. Although the conformal mapping is performed over the visibility spheres, the geometries at this stage are simple cubes in order to facilitate a more intuitive definition of the visibility boundaries and the cuts. Fig. 4 and 6 illustrate the templates for the 5-sided and the 4-sided CAVE respectively. In both cases, the templates are parameterized with a reference point whose projection on the walls (shown as green spheres) define the intersection points of the cut.

A similar template can be defined for the 3-sided CAVE as well (Fig. 5a). However, because of the increased length of the cut, mapping to an arrangement of displays is not practical as the conformal visualization would introduce significant distortion artifacts. For a target mesh similar to Fig. 5c, we map only a hemisphere of the original viewing directions (Fig. 5b).

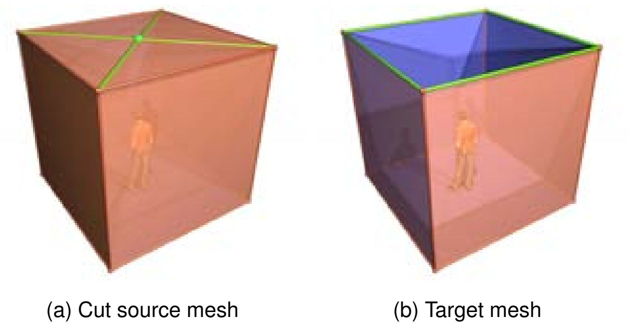


Fig. 4: Template meshes for a 5-sided CAVE.

4.2 Mesh Processing

The source and target meshes are processed independently and in parallel to obtain the conformal mapping. The first step is to triangulate and refine the template to the desired granularity so that the computations can be performed with sufficient accuracy. We achieve this by an edge split operation where a new vertex is iteratively introduced at the mid-point of each edge, doubling the number of triangles. The process is performed until the edge length falls below a user-specified threshold. In our experiments, edge sizes corresponding to $20cm$ in the real-world allow for sufficiently accurate conformal maps to be generated at interactive speeds, while granularity below $5cm$ yields only marginal improvements. In addition to the performance constraints, the edge threshold also depends on the resolution of target displays.

At this point, each mesh is finely tessellated and contains a single closed boundary. Next, we remove a triangle so that

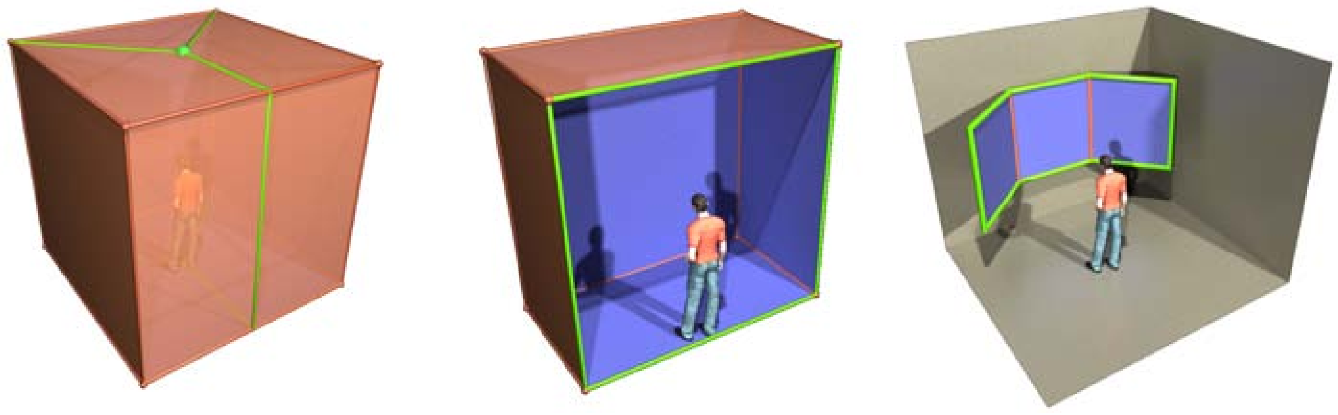


Fig. 5: Template meshes for a 3 screen target. The cut in (a) is suitable for the 3-sided CAVE. For an arrangement of flat-panel displays (c), the cut in (b) reduces the distortion effects.

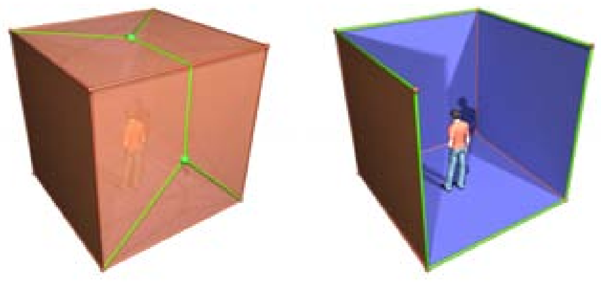


Fig. 6: Template meshes for a 4-sided CAVE.

the resulting mesh can be mapped to the complex plane (see Sec. 3.4). We select the triangle that is at the center of the mesh, or farthest from the original boundary. The same triangle is removed from both the source and target meshes. We also store a 3-vertex correspondence between the meshes, which will be used to align the planar projections of the conformal maps. These vertices are selected from among the common vertices on the original closed boundaries. We then map the cube geometry to a sphere by the *direction map*

$$p \to \frac{p-c}{|p-c|},$$

where p is a point on the cube and c is the center of the cube.

The following step computes the shortest path τ along the edges of the mesh between the original boundary γ_1 and the newly created boundary γ_2 at the center of the mesh. Since the embedding in the complex plane is periodic, the exact shape of τ is not important; however, for consistency, we enforce that the same cut is made on both the source and the target meshes.

Next, the discrete Ricci flow algorithm presented in Sec. 3.3 is used to compute the conformal mapping to the unit disc. This mapping is stored as UV coordinates at the vertices of the two meshes. Fig. 3 illustrates the results of this step. The Möbius transformation is then used to align the two conformal maps based on the vertex correspondence stored earlier. We use special Möbius transformations to map the point triplets to $1, i$ and -1 on the unit circle, which aligns the corresponding markers. Suppose $\{p,q,r\}$ are three markers on the unit circle and

$$\eta_1(z) = \frac{z-p}{z-q}\frac{r-q}{r-p}$$

maps them to $\{0,\infty,-1\}$. Let

$$\eta_2(z) = \frac{1+i}{2}\frac{z-1}{z-i},$$

then $\eta_2^{-1} \circ \eta_1$ is the desired Möbius transformation, which maps $\{p,q,r\}$ to $\{1,i,-1\}$. Fig. 3b and Fig. 3d show the result after the alignment.

4.3 Cubemap Generation

After the mesh processing, both the source and the target meshes contain aligned conformal mappings to the $2D$ complex plane stored in the UV channel at each vertex. All further processing is parallelized efficiently on the GPU using shaders developed in the NVIDIA Cg language.

In the next step, we utilize a vertex shader that computes the viewing direction through each vertex in the source mesh and stores the result in the vertex color attribute. The shader also flattens the mesh to the unit disc by coping the UV coordinates to the vertex position attribute. In the pixel shader we simply render the interpolated vertex colors to a high resolution floating-point RGB texture. This texture corresponds the circular map in Fig. 3d where each pixel encodes a viewing direction. The reference point for computing these directions is specified by the user and is not necessarily the same point used to define the cut on the mesh template.

At the final stage, we map the circular texture onto the target mesh and use a pixel shader to render the view directions onto the faces of a cubemap. This cubemap is a discretization of a conformal transformation T_{ray} that maps the viewing directions of the target visual environment to the corresponding directions on the full visibility sphere. By reversing the source and the target meshes, we also compute the T_{mesh} transformation from the set of all viewing directions to the visibility defined by the boundaries of the target mesh. The direction vectors for both the circular maps and the cubemaps are stored in RGB textures with 32-bit per-channel precision. We set the resolutions to 4096×4096 and 1024×1024 respectively, although smaller textures can be used with minimal loss of directional accuracy if GPU resources are limited. However, if the rendering is performed on older GPU hardware that does not support bilinear filtering of 32-bit floating point textures, the largest supported resolution should be used with nearest-neighbor sampling. Alternatively, the precision can be reduced to 16 bits per channel, although the resulting loss of accuracy in the direction vectors may lead to objectionable rendering artifacts.

5 Visualization Techniques

We have implemented the system described in Sec. 4 as a library that can be integrated into existing visualization applications. The cubemap generator class contains a minimal interface for updating parameters such as edge threshold and reference cut points, as well as for notifying the main application when the cubemaps are updated. The T_{ray} and T_{mesh} transformations that we compute allow us to support both forward and backward rendering pipelines. We demonstrate applications to rasterization, single-pass raycasting for Direct Volume Rendering (DVR) and real-time raytracing on the GPU. We have also developed a dynamic visibility manipulation technique based on a reference position in a tracked VR environment.

5.1 Rasterization

We first apply the conformal visualization to a rasterization pipeline. An efficient vertex-based transformation is used for rendering well-tessellated geometry with OpenGL, similar to the approach by Spindler et al. [25]. Intuitively, every vertex in the scene is transformed so that triangles that are projected on the top screen in the 6-sided CAVE configuration are instead projected on the 4 side screens in the 5-sided CAVE configuration. This transformation is defined by the following:

$$\begin{aligned} \mathbf{r_c} &= (M_{wc}^{-1})^T \cdot norm\left(\mathbf{p_w} - M_{wc}^{-1} \cdot [0,0,0,1]^T\right) \\ \mathbf{p_w} &\rightarrow M_{wc}^{-1} \cdot (T_{geom}(\mathbf{r_c}) \cdot |M_{wc} \cdot \mathbf{p_w}|) \end{aligned} \quad (6)$$

where $\mathbf{p_w}$ is the vertex position in world-space, $\mathbf{r_c}$ is the normalized view direction in CAVE space and M_{wc} is the world-space to CAVE-space transformation matrix. In our rendering framework, the head node for the visualization cluster emits the camera information to all the rendering clients and the view matrix V associated with that camera is the world-space to CAVE-space transformation matrix. Each visualization node then computes the final view and projection matrices based on the target projection surface (e.g., front, left, etc.). The geometry transformation is performed in a custom vertex shader that is bound to every primitive in the scene.

The simple vertex shader approach suffers from a number of shortcomings. In particular, the requirement for well-tessellated meshes limits the applicability of our technique for many large real-world datasets. We take advantage of the new hardware tessellation features available in DirectX 11-compatible GPUs to implement adaptive tessellation in our rendering pipeline. Another issue is that the conformal map for the forward rendering pipeline is not continuous by design. In certain cases, scene elements spanning the template cuts (see for example Fig. 4a) result in rendering artifacts that cannot be mitigated in the vertex shader. Instead, we move the computation of Eq. 6 to the geometry shader, which is available on DirectX 10-compatible GPUs and operates on the mesh primitives instead of the individual vertices.

The tessellation stage of the hardware pipeline operates on a new patch primitive and in a pre-processing step, we convert all triangles in the scene to triangular patches. In the Tessellation Control (TC) shader, we compute the edge and inner tessellation levels based on the size of the screen-space projections of the patches and the edges after applying the conformal map to the patch vertices. In the Tessellation Evaluation (TE) shader, new vertex attributes are computed using barycentric interpolation. Finally, a geometry shader operates on each triangle produces by the tessellation stage. We apply Eq. 6 to each vertex and detect triangles that span a discontinuity in the conformal map by a threshold on the area and the maximum edge length of the modified triangles. Detected triangles are simply not passed on to the rasterization stage. The fragment shader is not modified from our original implementation.

Compared to the vertex shader implementation, the improved image quality of using the adaptive tessellation incurs a more significant decrease in rendering performance on current generation GPUs. The relative speed reduction is less pronounced for 3D engines that already use GPU tessellation for effects such as displacement mapping or employ high-resolution meshes.

5.2 Direct Volume Rendering

The conformal transformation is applied in a similar fashion to volume rendering. Our visualization algorithm is based on single-pass ray-casting over 3D textures with support for advanced lighting and shadowing, as well as pre-integrated transfer functions [26]. Our framework integrates volume rendering tightly into the scene graph and we render out the volume-space positions of the front and back faces of the volume bounding box, modified by the depth of other scene geometry. One possible approach for incorporating the conformal map is to tessellate the bounding box and apply the T_{geom} transformation as described in Sec. 5.1. However, our target application is the exploration of the virtual colonoscopy data, in which case the camera is often within the volume and the starting positions of the rays are defined on the near clipping plane. It is more accurate to transform the positions on the near clipping plane and the back face of the bounding volume to world-space and then to apply the following transformation:

$$\begin{aligned} \mathbf{r_c} &= (M_{wc}^{-1})^T \cdot norm\left(\mathbf{p_w} - M_{wc}^{-1} \cdot [0,0,0,1]^T\right) \\ \mathbf{p_w} &\rightarrow M_{wc}^{-1} \cdot (T_{ray}(\mathbf{r_c}) \cdot |M_{wc} \cdot \mathbf{p_w}|) \end{aligned} \quad (7)$$

where, again, $\mathbf{p_w}$ is the vertex position in world-space, $\mathbf{r_c}$ is the normalized view direction in CAVE space and M_{wc} is the world-space to CAVE-space transformation matrix. The viewing vector is then constructed from the modified starting and ending positions. Note that this transformation is very similar to Eq. 6. The difference is that since this is a backward rendering pipeline, we are transforming the view directions directly and the T_{ray} cubemap is used.

5.3 GPU Raytracing

A major limitation of the geometry transformation approach is that Eq. 6 needs to be computed for every vertex in the scene, including vertices generated by the tessellation shaders. As a result, the rendering performance scales poorly as the data sizes increase. Furthermore, while GPU adaptive tessellation and rejection of triangles spanning map discontinuities solves many of the rendering quality issues for non-uniformly tessellated meshes, those approaches affect the rendering performance and are more complex to implement. In contrast, the conformal visualization for a backward rendering pipeline transforms the viewing directions as opposed to the scene geometry itself. As a result, mesh tessellation is not an issue, the conformal map does not contain discontinuities and the scene shaders do not need to be modified.

A number of GPU-based ray-tracing algorithms have been proposed that can achieve interactive frame-rates even for large geometric models [27], [28], [29]. We integrate a ray-tracing renderer with our scene graph based on the NVIDIA OptiX engine. OptiX accelerates ray-tracing on the GPU by defining ray-generation, ray-scene intersections and shading programs in the CUDA language which access traditional acceleration structures that are also stored on the GPU. The programs are then intelligently scheduled on all CUDA-enabled GPUs in the system. Its features are comparable to other real-time ray-tracing approaches [28], [29]. For our conformal visualization, the ray transformation is applied at the ray-generation level, which is separate from the scene-graph and therefore much simpler to re-implement. The computation is also simplified:

$$\mathbf{d_w} \rightarrow M_{wc}^T \cdot \left(T_{ray}\left((M_{wc}^{-1})^T \cdot \mathbf{d_w}\right)\right) \quad (8)$$

Similarly to the volume rendering approach, we transform the world-space ray direction $\mathbf{d_w}$ to CAVE-space and fetch a new ray direction from the conformal map. CUDA (and by extension OptiX) does not currently support cube textures natively, therefore indexing the conformal map is implemented in the ray-generation program. We also augment the conformal map generation stage of our pipeline to copy the pixel data to a 2D texture in the vertical cross format.

5.4 Dynamic Visibility Manipulation

Our conformal visualization pipeline generates the ray transformations T_{ray} and T_{geom} based on a reference point inside the target mesh. In a tracked VR environment, the reference point is generally the head position in order to allows for more accurate visibility retargeting as the user moves through the environment. For example, as the user approaches a corner of the 5-sided CAVE, the visual information from the missing top projection would be presented mainly onto the 3 remaining visible surfaces. However, recomputing the mesh cut and the conformal mapping is an expensive operation that approaches interactive performance only for coarse tessellations of the source and target templates. In contrast, the GPU processing can be performed very efficiently even for large texture sizes on mid-range GPUs, allowing for realtime changes to the position of the reference point.

In our application, we decouple the head-tracking for rendering and computing the mesh templates from the tracking of the reference point for the viewing rays computations. This provides a novel user interface for controlling the conformal transformations T_{ray} and T_{geom} without recomputing the conformal maps. For a given conformal parameterization of the source and target meshes, moving the reference point changes the effective frustum of viewing rays that is mapped to each target display surface. As a result, zooming into the visual data is achieved by moving the reference point away from the given display surface. However, the interface presented to the user is actually the inverse as it is more natural to move a tracked device toward and area of interest. The conformal mapping established between the source and the target meshes ensures that all viewing directions are mapped onto the existing display surfaces, thereby maintaining the visual context around the focus point.

6 Results

6.1 Visualizing the Conformal Transformation

We first demonstrate the visual properties of our technique on a unit sphere with a checkerboard texture. In Fig. 7, the virtual camera is positioned at the center of the sphere so that the two poles of the texture are shown on the left and the right displays. Fig. 7b shows the results of our conformal visualization and demonstrates how the visual information originally shown on the top screen is instead rendered on the left, front, right and back screens. In Fig. 7c we apply the T_{ray} transformation computed for the 4-sided CAVE mesh target (see Fig. 6b). The result is that compared to the 5-sided case, the visual information from the back screen is pushed toward the bottom and the side screens.

We also visualize the conformal transformation for the scenario presented in Fig. 5 where the $180°$ field of view is mapped to an arrangement of workstation monitors. The monitor setup is modeled after the Samsung MD230X6 array of 3×2 high resolution thin bezel monitors. This represents a challenge for the conformal mapping since with an aspect ratio of $2.66:1$, the boundaries of the source and target mesh are significantly different. Nevertheless, our approach produces an

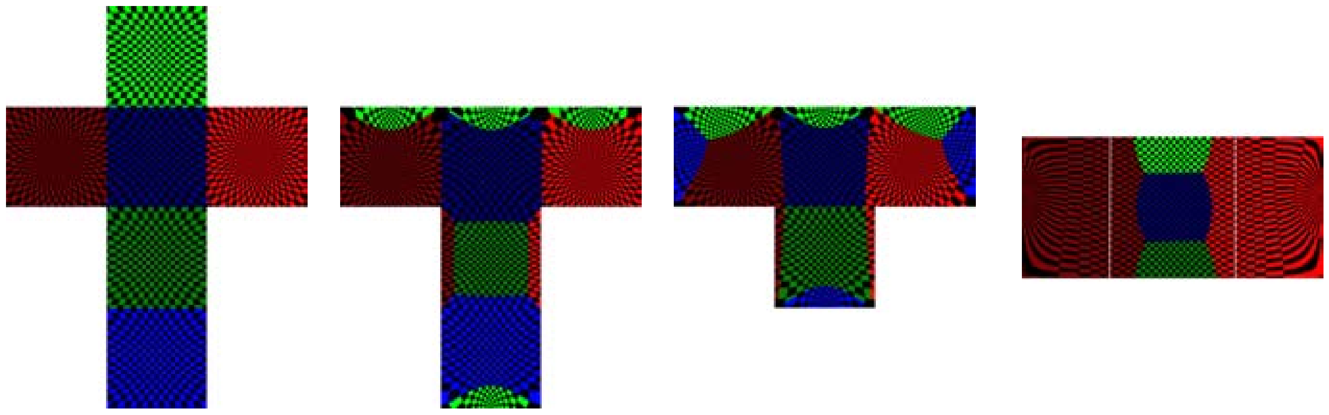

(a) 6-sided CAVE (b) 5-sided CAVE (c) 4-sided CAVE (d) 3-sided horizontal arrangement of displays

Fig. 7: Raytracing of a checkerboard sphere with conformal visualization on different display targets. The layout for (a)-(c) is in the standard vertical cross format and the color coding is defined for the original walls in the 6-sided CAVE configuration (blue for front/back, red for left/right, green for top/bottom). For (d), we show the output of all 3 displays side by side.

angle-preserving conformal transformation that enhances the field of view of the user.

User interactions with the virtual scenes often involve navigation in 3D space. We demonstrate this aspect of our visualization technique with a checkerboard tunnel. Fig. 8 illustrates the camera panning down during the navigation with the 5-sided CAVE rendering target. Note that compared to the straightforward front projection, the conformal rendering preserves the context of the tunnel's direction even when the camera is pointed downward as in Fig. 8d. The visual information corresponding to the original front projection is presented with minimal distortion, as also shown in Fig. 7b.

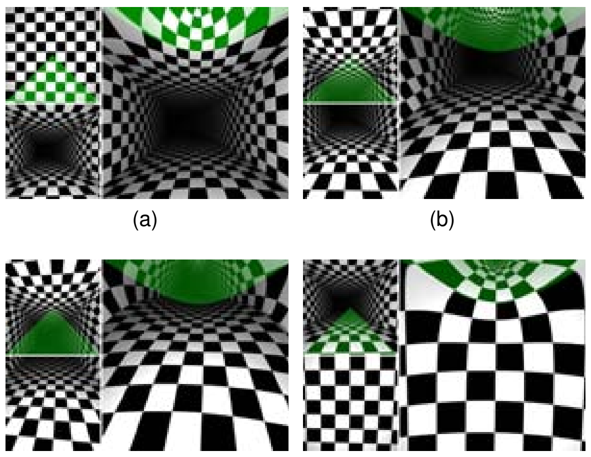

(a) (b) (c) (d)

Fig. 8: Navigation in the checkerboard tunnel with conformal visualization where the camera pans down in (a)-(d). Each triplet of images shows the original front view (lower-left), original top view (upper-left) and front view with conformal visualization (right).

6.2 Performance

The pipeline described in Sec. 4 is designed to execute most efficiently on a system with a modern multi-core CPU and at least one mid- to high-end GPU. The mesh processing for the source and the target templates is performed in parallel and we also utilize the PARDISO sparse linear solver from the Intel Math Kernel Library during the computation of the discrete Ricci flow. We measured the performance of a variety of systems with 2, 4 and 8 CPU cores and found that quad-core CPUs strike the best balance between cost and peformance. Although dual quad-core CPUs show an improvement with the PARDISO solver, the overall computation time is dominated by the mesh processing. In our current implementation, this is for the most part a serial operation that does not benefit from more than 2 CPU cores.

We also compare the performance of the GPU stage of the cubemap generation pipeline for a variety of hardware. Since our shaders are relatively simple, the only GPU requirement is the support for bilinear 32-bit floating point texture filtering and the appropriate texture sizes for the target display resolution. The oldest GPU we have tested is the NVIDIA Quadro 2500M, which can produce 4096×4096 textures for the circular map and a 1024×1024 cubemap texture at up to 20 iterations per second.

Our primary testing machine contains a single Intel Xeon E5620 which is a quad-core CPU running at 2.4GHz and a single NVIDIA Geforce GTX 480. Fig. 9 summarizes the performance for a range of edge length thresholds that are reasonable for real-world applications. The threshold values are defined for the normalized vertex positions of the source and target meshes. At the 0.1 threshold, our conformal generation executes up to 2 iterations per second, allowing for interactive updates of the conformal maps. We expect that the most current CPU architecture running at higher clock speeds will double the performance of our current implementation. For higher quality images with a static conformal map, the 0.05 thresholds produces sufficient conformal map resolution for a 1400×1050 rendering target.

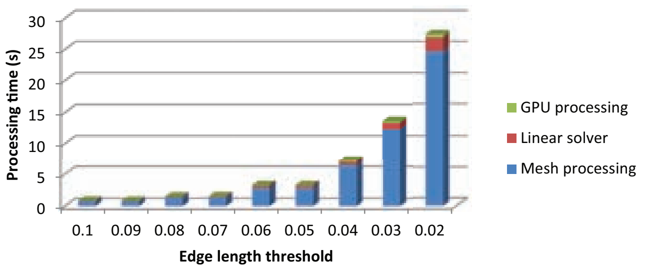


Fig. 9: Performance of the cubemap generation running on a single quad-core Intel Xeon E5620.

The GPU stage as used for the dynamic visibility manipulation can currently execute at minimum 60 iterations per second, although this is not indicative of real-world performance. As described in Sec. 5.3, the GPU raytracing requires that the cube texture be copied to a 2D format, which is an expensive operation that disrupts the GPU pipeline. The time for the copying operation is included in Fig. 9. The performance will increase significantly when such requirements are relaxed in the future. In practice, the time for generating the conformal maps on the GPU is negligible for both volume rendering and ray-tracing.

6.3 Applications

We first evaluate the geometry-based conformal visualization technique on a course mesh scene. Our rendering system uses OpenGL 2.0 rasterization and we have augmented the material system with the shaders described in Sec. 5.1. The other significant change is that triangle meshes in the scene are converted to the new OpenGL patch primitive with 3 control points per patch that match the original vertices of the triangles. The edge tessellation factors are computed based on a user specified edge length threshold so that silhouette edges in the scene appear smooth with the conformal visualization. The test scene used to generate the results in Fig. 10 is composed of 19 instances of the Stanford Bunny dataset where each instance contains $2,500$ triangles. Compared to the traditional pinhole projection camera, the conformal visualization significantly expands the user's field of view and allows the visual context of the scene to be maintained throughout the navigation. The angle-preserving property of the conformal maps ensures that the overall shape of the meshes is recognizable even through geometries directly above the user are scaled down, as illustrated in Fig. 7b.

The performance of the conformal visualization scales almost linearly with the number of triangles in the scene after the tessellation. Depending on the scene complexity, its initial tessellation, the edge length threshold and the position of the user in the scene, we have observed reductions in the framerate of up to 50%. For the example in Fig. 10, the observed performance was reduced by approximately 15% after enabling the conformal visualization.

Our system integrates full support for the rendering of volumetric data with particular focus on medical visualization such as IVC. The pair of images on the left in Fig. 11a

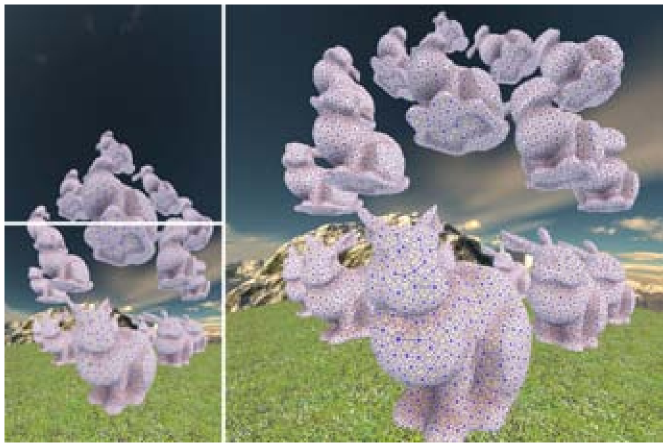
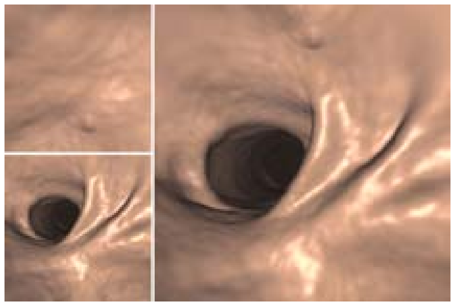
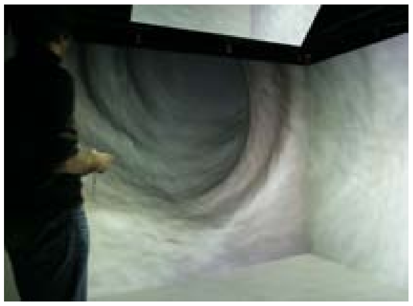


Fig. 10: Conformal visualization applied to the rasterization pipeline. The images show the original front view (lower-left), the original top view (upper-left) and the front view with conformal visualization (right) for the same camera position. Triangular patches are drawn in blue, tessellated triangles are in black. Our technique produces artifact-free results even for coarse scene geometries.

are the original perspective projections for the front and top screens in a fully-enclosed CAVE. The image on the right is the result of the conformal visualization described in Sec. 5.2 for the front screen in the 5-sided CAVE. Note that unlike in image-based retargeting approaches, the final result is rendered directly, and the original images are included here only for demonstration purposes. The source of the volume data is a $512\times512\times451$ 16-bit CT scan of a patient's abdomen after digital cleansing of the colon. A suspicious area is visible on the top screen, which would normally be missed in the CAVE due to the angle of the approach and the lack of a top screen. After the conformal transformation of the viewing rays, this area is projected onto the front CAVE screen and the shape of the polyp on the colon wall is preserved. Although conformal mapping is not distance-preserving and sizes are scaled, we provide tools to measure the actual distance in voxel-space. Since the conformal distortion map is computed in a pre-processing step, the cost of the initial transform of the ray direction is negligible compared to the ray integration. In practice, the performance drop is measurable but it is less than 1% on average.

Interactive ray-tracing is supported in our visualization software as a full replacement for the OpenGL renderer. As described in Sec. 5.3, the application of the conformal map is simplified in terms of both the computation and the integration with the existing scenes. The main benefit is that the rendering performance scales with the size of the viewport as opposed to the scene complexity. The ray-tracing approach also makes no assumptions regarding the tessellation of the data and provides high-quality conformal visualization results even in areas of low polygonal density. Our GPU raytracing pipeline is used in visualization experiments that involve large connected data structures. The conformal visualization is particularly suitable for scenes containing a significant number of elements when the exact shape of the connections is not relevant to


(a) Virtual Colonoscopy navigation with conformal visualization


(b) 5-sided CAVE without conformal visualization

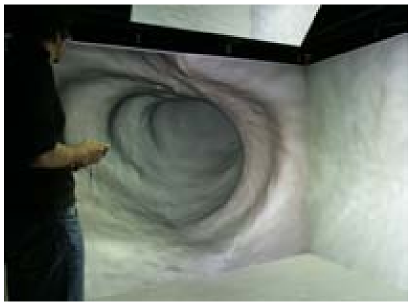
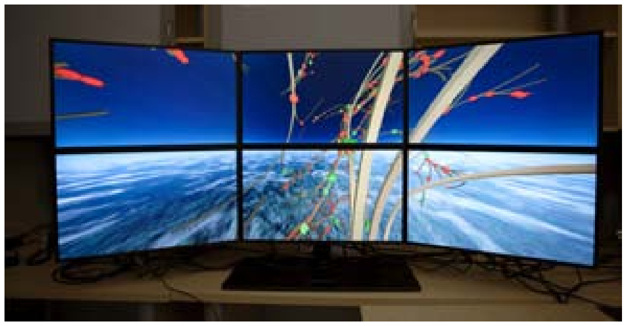

(c) 5-sided CAVE with conformal visualization

Fig. 11: (a) Snapshot of Immersive Virtual Colonoscopy in the 5-sided CAVE. The images show the original front view (lower-left), the original top view (upper-left) and the front view with conformal visualization (right). The shape of the polyp on the top view is preserved under the conformal visualization. Compared to the traditional volume rendering (b), the conformal visualization allows for a more thorough analysis of the dataset (c) and the entire surface of the colon is visible as the user negotiates a bend in the colon.

understanding the data, but the shape of the elements may be. The expanded field of view and the resulting visual context are particularly beneficial for exploration tasks in information visualization applications. Fig. 13 presents our test scene, which contains an abstract binary tree with 10,000 data elements and color-coded connections. Note that with conformal visualization for the 5-sided CAVE the missing visual information from the top screen is displayed near the 4 top edges of the CAVE. We further simulate a 4-sided CAVE by disabling the rendering to the back wall and applying the transformation depicted in Fig. 7c. As the checkerboard rendering shows, the missing information from the back wall is presented primarily on the back edges of the side screens, while the top is distributed over the front and side screens. In both cases, the visual transformation is angle-preserving and the nodes in the graph retain their shape.

We also render the scene with the mesh template described in Fig. 5c. Compared to the pinhole camera rendering in Fig. 12a, the conformal visualization in Fig. 12b increases the field of view while preserving the local shapes of the objects in the scene. Our current mesh template utilizes only a hemisphere of the user's original field of view, however, that can be increased by augmenting the tiled array with additional screens on the top and the bottom. The resulting boundary of the screen geometry would then more closely resemble the original visibility boundary, which in turn would provide more uniform conformal visualization results.

Finally, one notable feature of the graph dataset presented in Fig. 13 and Fig. 12 is that it contains a large number of dense node clusters, which are difficult to explore using the traditional navigation paradigms. In Fig. 14 we demonstrate our dynamic visibility manipulation technique for a stationary position of the user in both the physical space of the 5-sided CAVE and the virtual space of the 3D scene. The user manipulates the reference point used to generate the GPU textures for conformal visualization as described in Sec. 5.4 in order to focus on the cluster at the back wall (Fig. 14a) and the distant clusters displayed on the front wall (Fig. 14c) while maintaining the visual context. In both cases, the reference point was moved to the extreme positions on the back and the front walls respectively. Fig. 14b shows the default conformal visualization result where the reference point matches the user's viewing position. As discussed in Sec. 6.2, the visibility manipulation is implemented entirely on the GPU, and as a result, the zooming interface appears very smooth to the user.


(a) No retargeting

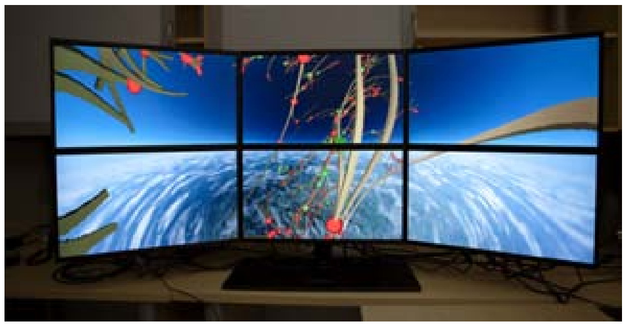

(b) With conformal visualization

Fig. 12: Conformal visualization results for a large tiled display. (a) The standard pinhole camera model with wide FOV leads to significant distortions near the periphery of the image. (b) In contrast, conformal visualization allows for 180° horizontal and vertical FOV while locally preserving the shapes in the data.

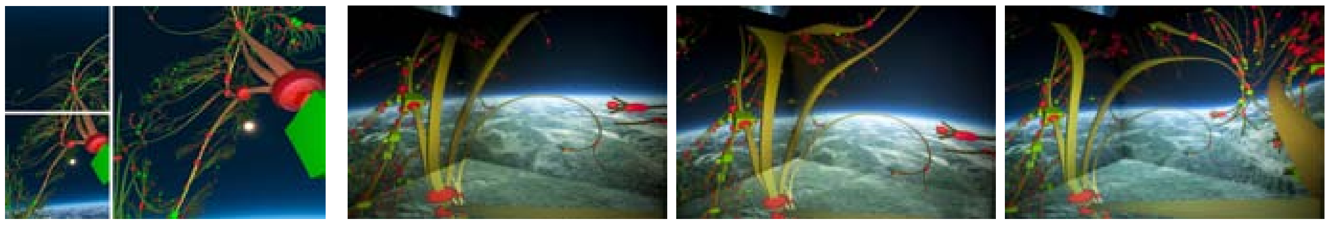

(a) Navigation of densely connected data with conformal visualization. (b) No retargeting (c) Retargeting to 5-sided CAVE (d) Retargeting to 4-sided CAVE

Fig. 13: Conformal visualization results. (a) Each triplet of images shows screen captures of the original front view (lower-left), the original top view (upper-left) and the front view with conformal visualization (right). The photographs (b)-(d) illustrate views from inside the CAVE where the front wall is on the left side of the image. Missing visual information from (c) the top wall and (d) the top and back walls is presented on the available display surfaces. The 4-sided CAVE is simulated by disabling the back wall.

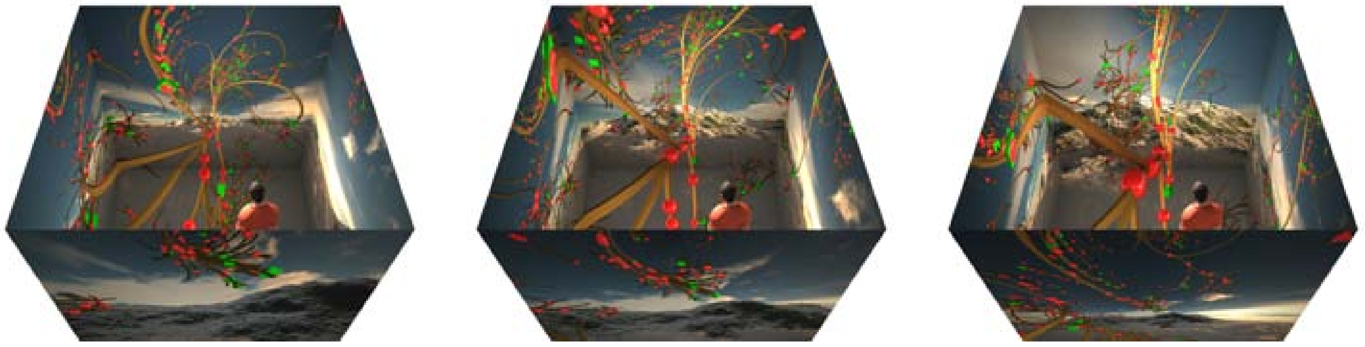

(a) Reference point at the back wall (b) Reference point at the center of the CAVE (c) Reference point at the front wall

Fig. 14: Conformal visualization results for the GPU raytracing pipeline with dynamic visibility manipulation. The standard conformal parameterization for the 5-sided CAVE is used, while the reference point for computing the GPU textures moves between the extreme points at the center of (a) the back wall and (c) the front wall. The recomputation executes at realtime speeds and allows for a context-preserving zoom operation (e.g. toward the dense cluster of nodes at the front wall).

7 Evaluation

Our conformal visualization technique is evaluated informally in the 5-sided CAVE for the visual detection of polyps in phantom colon datasets. The evaluation focuses on the core retargeting properties of the technique, and therefore we use static mesh templates for a user position at the center of the CAVE without the dynamic visibility manipulation. In particular, we are interested in how the additional visual information affects the users' navigation experience and their accuracy in identifying the simulated polyps. The phantom data are generated based on sections of the colon centerline from a real patient who had undergone Virtual Colonoscopy and each volumetric scene contains between 25 and 29 synthetic hemispherical polyps.

The user control scheme in the CAVE uses an analog gamepad with the following mapping: view control with the left analog nub, forward/backward motion with the right analog numb, camera roll with the shoulder pads. The users are given time to study the navigation scheme for both regular and conformal visualization, after which they are asked to find as many of the simulated polyps as possible. The initial camera roll is selected randomly at the beginning of each experiment and the visualization modality order is pre-selected for each user. The examiner records the number of detected polyps, the number of false positives and the time for each fly-through.

Our informal study involved 12 participants and on average, the examination time was consistent between the regular and conformal visualization. The conformal visualization improved the non-biased detection sensitivity for polyps from 91% to 93% overall with individual differences in the range $[-2\%, +10\%]$. On average this translates to the detection of up to 0.9 additional polyps over the baseline per datasets with 29 polyps. The false positive rate for both visualization modalities was on average 0.1%. During each evaluation, the examiner monitored the visualization on a separate computer and marked polyps that were projected only onto the missing ceiling surface. If we isolate only those polyps from our resuts, the detection rates were virtually identical to the average for the datasets examined with conformal visualization, compared to 0% for the regular volume rendering.

In our follow-up questionnaire, the users indicated that stereoscopic vision remained effective in all sections of the display surfaces, including the areas at the interfaces with the missing display surface where the visual information is compressed. They also developed natural strategies for utilizing the additional visual information with minimal input from the examiner (the users were told during the short training session that the visualization at the top edge was generated from data directly above them). Almost all the users indicated that the additional information helped them navigate bends in the colon that point upward. A significant number of users developed the habit of rotating the virtual camera in order to examine polyps they had spotted in the compressed areas of the visualization. We received similar comments from a professional radiologist who informally reviewed our system with real colon data. Our preliminary experiments indicate that the dynamic visibility manipulation can reduce the need for camera rotation by allowing the user to temporarily zoom into area of interest while navigating the dataset.

8 Conclusions and Future Work

We have presented a visualization technique based on conformal mapping for partially immersive visualization platforms that allow the user to visually explore the full virtual environment. The discrete Ricci flow is used to compute the conformal mapping and the theoretical foundation for our approach guarantees that the transformation is angle-preserving and therefore conformal visualization preserves shapes locally.

We have demonstrated applications of our approach to rasterization, volume rendering and GPU-based ray-tracing pipelines under a variety of retargeting scenarios. In particular, we show applications to angled arrangement of flat panel displays which are currently significantly cheaper to build and maintain, and are therefore more widely deployed. We avoid many of the challenges related to image-based retargeting approaches by rendering the final results directly onto the output surfaces, which does not produce resampling artifacts. Our user study simulates the task of finding cancerous polyps in Virtual Colonoscopy and we demonstrate improved sensitivity compared to the traditional visualization at similar examination times and false positive rates. Based on the real-time performance of the GPU stage in our conformal mapping pipeline, we have also developed a novel focus+context interaction technique based on the manipulation of the viewing ray correspondence.

The mesh templates we describe in Sec. 4.1 are not uniquely defined and we plan to fully explore alternative geometries and cuts in the future. We also want to extend our mesh generation system so that the templates are created automatically by a description of the visualization system or by the end-user of the visualization application. Furthermore, the template cuts we have presented are constructed to account for a tracked reference point in the environment, which is not necessarily optimal in terms of the quality of the conformal visualization.

We plan to further explore the focus+context paradigm in relation to our conformal visualization technique. In particular, we plan to integrate some of the traditional magic lens approaches in the creation of our conformal maps, which can augment the existing real-time manipulation of the viewing ray correspondence.

Our implementation can be further optimized on both the CPU and the GPU. Although there exist GPU-based sparse linear solvers, the biggest performance gains would be from moving additional stages of the mesh processing pipeline to the GPU. Our selection of Ricci flow is based on the stability and convergence properties. In the future, we also plan to investigate additional conformal mapping algorithms in terms of their potential for GPU implementation.

Acknowledgments

This work has been supported by NSF grants CCF-0448399, CCF-0702699, CNS-0959979, CNS-1016829, IIS-0916235, IIS-0916286, NIH grant R01EB7530 and ONR grant N000140910228. The volumetric colon datasets have been obtained through the NIH, courtesy of Dr. Richard Choi, Walter Reed Army Medical Center.

References

[1] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The cave: Audio visual experience automatic virtual environment," *Communications of the ACM*, vol. 35, no. 6, pp. 64–72, 1992.

[2] A. Hogue, M. Robinson, M. R. Jenkin, and R. S. Allison, "A vision-based head tracking system for fully immersive displays," *Proceedings of the Workshop on Virtual Environments*, pp. 179–187, 2003.

[3] W. P. Thurston, "Geometry and topology of three-manifolds," *Princeton Lecture Notes*, 1976.

[4] F. Qiu, B. Zhang, K. Petkov, L. Chong, A. Kaufman, K. Mueller, and X. D. Gu, "Enclosed five-wall immersive cabin," *Proceedings of the 4th International Symposium on Advances in Visual Computing (ISVC)*, pp. 891–900, 2008.

[5] M. Jin, J. Kim, F. Luo, and X. Gu, "Discrete surface ricci flow," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 5, pp. 1030–1043, 2008.

[6] Y.-L. Yang, J. Kim, F. Luo, S.-M. Hu, and X. Gu, "Optimal surface parameterization using inverse curvature map," *Transactions on Visualization and Computer Graphics*, vol. 14, no. 5, pp. 1054–1066, 2008.

[7] Y. Wang, X. Yin, J. Zhang, X. Gu, T. F. Chan, P. M. Thompson, and S.-T. Yau, "Brain mapping with the ricci flow conformal parameterization and multivariate statistics on deformation tensors," *2nd MICCAI Workshop on Mathematical Foundations of Computational Anatomy*, pp. 36–47, 2008.

[8] M. Jin, W. Zeng, F. Luo, and X. Gu, "Computing Teichmüller shape space," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 3, pp. 504–517, 2009.

[9] X. Li, X. Gu, and H. Qin, "Surface matching using consistent pants decomposition," *ACM Solid and Physical Modeling Symposium*, pp. 125–136, 2008.

[10] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 162–170, 2002.

[11] B. Springborn, P. Schröder, and U. Pinkall, "Conformal equivalence of triangle meshes," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–11, 2008.

[12] R. Raskar, J. v. Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines, "ilamps: geometrically aware and self-configuring projectors," *ACM SIGGRAPH 2005 Courses*, 2005.

[13] Y. Yang, J. X. Chen, and M. Beheshti, "Nonlinear perspective projections and magic lenses: 3d view deformation," *Computer Graphics*, vol. 25, no. 1, pp. 76–84, 2005.

[14] E. Pietriga, O. Bau, and C. Appert, "Representation-independent in-place magnification with sigma lenses," *Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 455–467, 2010.

[15] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman, "The magic volume lens: An interactive focus + context technique for volume rendering," *Proc. IEEE Visualization*, pp. 367–374, 2005.

[16] C. Correa, D. Silver, and M. Chen, "Illustrative deformation for data exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1320–1327, 2007.

[17] H. Lorenz and J. Döllner, "Real-time piecewise perspective projections," *International Conference on Computer Graphics Theory and Applications (GRAPP)*, pp. 147–155, 2009.

[18] M. Trapp and J. Döllner, "Generalization of single-center projections using projection tile screens," *Advances in Computer Graphics and Computer Vision (VISIGRAPP)*, 2008.

[19] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, "Virtual voyage: interactive navigation in the human colon," *SIGGRAPH*, pp. 27–34, 1997.

[20] C. D. Johnson and A. H. Dachman, "CT colonography: The next colon screening examination?" *Radiology*, vol. 216, no. 2, pp. 331–341, 2000.

[21] W. Hong, J. Wang, F. Qiu, A. Kaufman, and J. Anderson, "Colonoscopy simulation," *Proc. SPIE Medical Imaging*, vol. 6511, 2007.

[22] K. Petkov, C. Papadopoulos, M. Zhang, A. E. Kaufman, and X. Gu, "Conformal visualization for partially-immersive platforms," *IEEE Virtual Reality*, pp. 143–150, 2011.

[23] B. Chow, "The ricci flow on the 2-sphere," *Journal of Differential Geometry*, vol. 2, no. 33, pp. 325–334, 1991.

[24] K. Stephenson, *Introduction To Circle Packing*. Cambridge University Press, 2005.

[25] M. Spindler, M. Bubke, T. Germer, and T. Strothotte, "Camera textures," *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (GRAPHITE)*, pp. 295–302, 2006.

[26] M. Hadwiger, J. M. Kniss, C. Rezk-salama, and D. Weiskopf, *Real-time Volume Graphics*. A K Peters, 2006.

[27] B. Liu, L.-Y. Wei, X. Yang, Y.-Q. Xu, and B. Guo, "Nonlinear beam tracing on a GPU," Mcrosoft Research, Tech. Rep. MSR-TR-2007-34, 2007.

[28] S. Popov, J. Günther, H.-P. Seidel, and P. Slusallek, "Stackless kd-tree traversal for high performance GPU ray tracing," *Computer Graphics Forum*, vol. 26, no. 3, pp. 415–424, 2007.

[29] M. Zlatuska and V. Havran, "Ray tracing on a GPU with CUDA – comparative study of three algorithms," *WSCG*, 2010.

Kaloian Petkov received the BA degree in computer science and mathematics from Lake Forest College in 2006. He is currently working toward the PhD degree in computer science at Stony Brook University. His research focuses on computer graphics, volume rendering, natural phenomena modeling and visualization, and general-purpose computing on graphics hardware. For more information, see http://www.cs.stonybrook.edu/~kpetkov.

Charilaos Papadopoulos graduated with a BSc in Informatics from the Athens University of Economics and Business, Athens, Greece in 2009. He is currently a PhD candidate in Computer Science at Stony Brook University, New York working on the RealityDeck, the first effort to create an immersive, giga-pixel display. He has also been employed at Microsoft Corporation as a software engineer intern, working on the web graphics group for Internet Explorer. His research interests include large scale visualization and virtual reality environments as well as user interfaces that enable fluid interactions with them. For more information, see http://www.cs.stonybrook.edu/~cpapadopoulo.

Min Zhang received her Bachelor degree in Mathematics and Applied Mathematics from Zhejiang University, China. She is a Ph.D Student in the Department of Computer Science in the State University of New York at Stony Brook University. Her research interests include computer graphics, visualization, and geometric modeling.

Arie E. Kaufman is a Distinguished Professor and Chair of the Computer Science Department, Chief Scientist of the Center of Excellence in Wireless and Information Technology (CEWIT), and the Director of the Center for Visual Computing (CVC) at Stony Brook University (aka State University of New York at Stony Brook). He is an IEEE Fellow, an ACM Fellow, and the recipient of IEEE Visualization Career Award (2005). He further received the IEEE Outstanding Contribution Award (1995), ACM Service Award (1998), IEEE CS Meritorious Service Award (1999), member of the European Academy of Sciences (since 2002), State of New York Entrepreneur Award (2002), IEEE Harold Wheeler Award (2004), and State of New York Innovative Research Award (2005). Kaufman was the founding Editor-in-Chief of *IEEE Transactions on Visualization and Computer Graphics* (TVCG), 1995-1998. He has been the co-founder, papers/program co-chair, and member of the steering committee of *IEEE Visualization Conferences*; co-founder/chair of *Volume Graphics Workshops*; co-chair for *Eurographics/SIGGRAPH Graphics Hardware Workshops*, and the Papers/Program co-chair for *ACM Volume Visualization Symposia*. He previously chaired and is currently a director of IEEE CS Technical Committee on Visualization and Graphics. Kaufman has conducted research and consulted for over 40 years specializing in volume visualization, graphics architectures, algorithms, and languages, virtual reality, user interfaces, medical imaging and their applications. He received a BS (1969) in Mathematics and Physics from the Hebrew University of Jerusalem, Israel, an MS (1973) in Computer Science from the Weizmann Institute of Science, Rehovot, Israel, and a PhD (1977) in Computer Science from the Ben-Gurion University, Israel. For more information, see http://www.cs.stonybrook.edu/~ari.

Xianfeng David Gu received his PhD degree in computer science from Harvard University in 2003. He is an associate professor of computer science and the director of the 3D Scanning Laboratory in the Department of Computer Science in the State University of New York at Stony Brook University. His research interests include computer graphics, vision, geometric modeling, and medical imaging. His major works include global conformal surface parameterization in graphics, tracking and analysis of facial expression in vision, manifold splines in modeling, brain mapping and virtual colonoscopy in medical imaging, and computational conformal geometry. He won the US National Science Foundation CAREER Award in 2004. He is a member of the IEEE.