A Rendering Framework for Multiscale Views of 3D Models

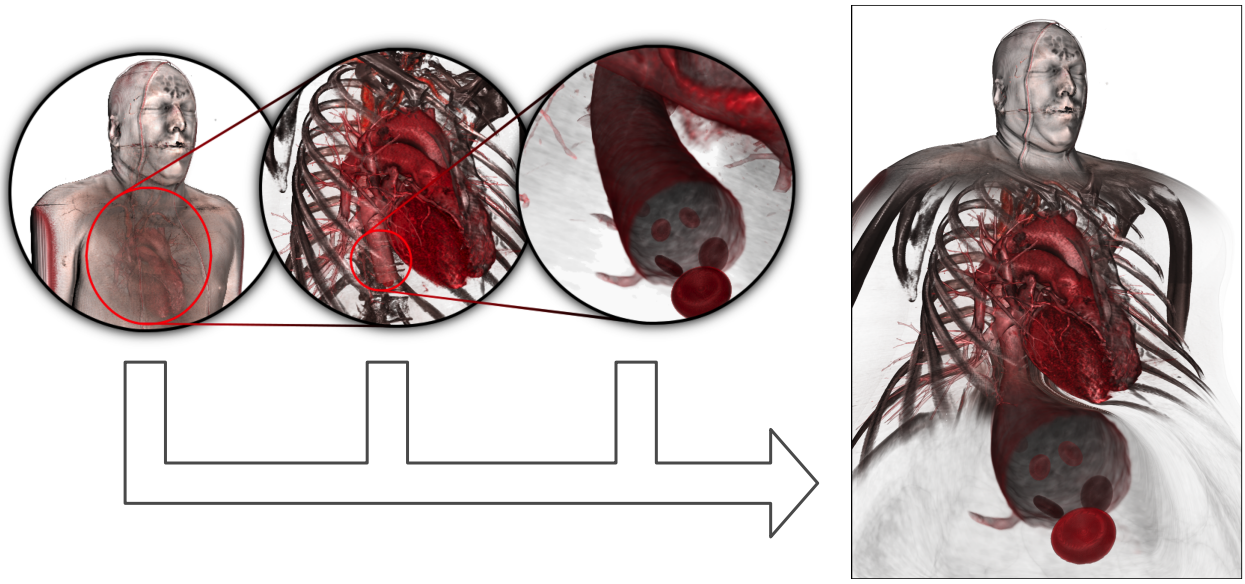


Figure 1: *A continuous multiscale view (right) of a volumetric human body dataset shows three different levels of detail (left three) in a single image. The image on the right is directly rendered with our multiscale framework.*

Abstract

Images that seamlessly combine views at different levels of detail are appealing. However, creating such multiscale images is not a trivial task, and most such illustrations are handcrafted by skilled artists. This paper presents a framework for direct multiscale rendering of geometric and volumetric models. The basis of our approach is a set of non-linearly bent camera rays that smoothly cast through multiple scales. We show that by properly setting up a sequence of conventional pinhole cameras to capture features of interest at different scales, along with image masks specifying the regions of interest for each scale on the projection plane, our rendering framework can generate non-linear sampling rays that smoothly project objects in a scene at multiple levels of detail onto a single image. We address two important issues with non-linear camera projection. First, our streamline-based ray generation algorithm avoids undesired camera ray intersections, which often result in unexpected images. Second, in order to maintain camera ray coherence and preserve aesthetic quality, we create an interpolated 3D field that defines the contribution of each pinhole camera for determining ray orientations. We have experimented with our camera model using both polygon and volumetric data sets. The resulting multiscale camera has three main applications: (1) presenting hierarchical structure in a compact and continuous manner, (2) achieving focus+context visualization, and (3) creating fascinating and artistic images.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms;

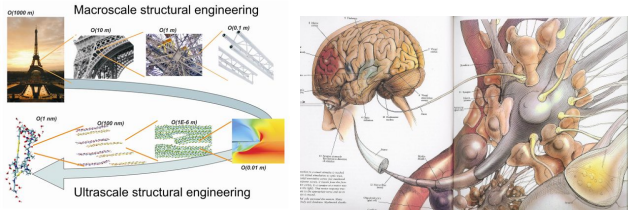
Keywords: multiscale views, camera model, levels of detail, visualization

1 Introduction

This project is motivated by an illustration created by artists at the Exploratorium in San Francisco. As shown in Figure 2, this illustration depicts both macro and micro perspectives of the human circulatory system in a continuous landscape across multiple scales. The seamless continuity between scales vividly illustrates how molecules form blood cells, how blood cells are distributed in a blood vessel, how the blood vessel connects to a human heart, and where the heart is located in a human body. The astonishment and fascination evoked by the illustration, along with its high educative value, won it first place in the illustration category of the 2008 US NSF and Science Magazine Visualization Challenge.

Figure 2: *Illustration "Zoom Into the Human Bloodstream" by Linda Nye and the Exploratorium Visualization Laboratory [Nye 2008].*

In scientific studies, it is often desirable to illustrate complex physical phenomena, organic structures, and man-made objects. Many of these physical structures are hierarchical in nature. Static multiscale illustrations are frequently used to convey hierarchical structures, such as the anatomy of organ systems and the design of engineered architectures, as shown in Figure 3. Large terrain data, on the other hand, is usually encapsulated in explorable, navigable interactive systems



(a) (b)

Figure 3: *Examples of multiscale illustrations. (a) A multiscale illustration showing the architecture of the Eiffel Tower using a zoom-in metaphor. (b) A hand-drawn image illustrating the hierarchical structure of the human nervous system.*

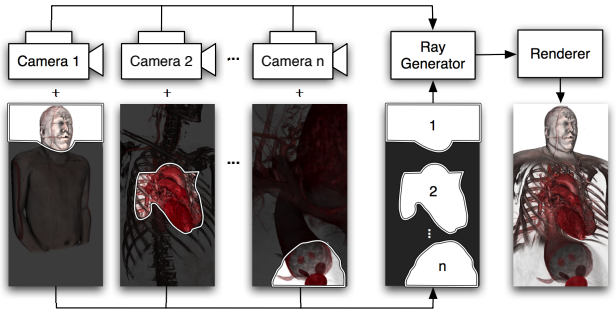


Figure 4: *Dataflow of multiscale rendering. The process starts by setting up separate pinhole cameras for different scales of view and image masks which indicate interesting regions in each view. Image masks are merged into a single image and passed to the camera ray generator, along with camera information. Non-linear bent rays are generated accordingly and used to sample the scene.*

[McCrae et al. 2009; Google 2010]. Animations are also helpful for presenting extremely large datasets which are difficult for users to navigate directly, such as those consisting of the solar system and the universe [National 1996; American 2009].

Out of the above scenarios, the creation of multiscale illustrations is the most challenging because there is no direct way to project a complex hierarchical 3D scene to a 2D image. In this paper, we focus on generating continuous multiscale images. Unlike traditional multiscale illustrations, in which separate images at different scales are connected by symbols (Figure 3(a)), a continuous multiscale image shows objects at several levels of detail with smooth object-space continuity between different scales. Although multiperspective rendering has received much attention in the literature [Yu et al. 2008], previous work has not specifically addressed seamless multiscale image rendering.

We introduce a rendering framework which generates and casts non-linearly bent rays into a 3D scene and projects multiple scales of interest onto a single image. Our multiscale rendering framework consists of a sequence of pinhole cameras, each of which captures a view of interest at a particular scale. The camera rays for each pixel in the final projected image are generated based on a user-defined image mask which specifies the interesting regions in each view. The rays are bent gradually from one scale to another to maintain object-space continuity as well as image-space smoothness. Our framework can be used on both polygon models and volumetric data. The resulting views are useful in many contexts. The most direct application is the presentation of complex hierarchical structures, as shown in Figure 1. Our multiscale camera can also achieve a focus+context effect, a technique frequently used in many visualization applications. Finally, we show that it can potentially create pictures that mimic artistic or impossible views of objects or scenes like the kind made by the artist M. C. Escher.

2 Related Work

2.1 Camera Models and Non-linear Projections

Camera projection is fundamental to computer graphics, since every 3D scene uses a projection to form a 2D image. As a result, various camera models have been developed for different scene modeling and rendering purposes. The General Linear Camera Model (GLC) developed by Yu and McMillan [2004b] uses three rays to define the affine combination and to generate other sampling rays. GLC is capable of modeling most linear projections, including perspective, orthogonal, push-broom, and crossed-slits projections, and was further extended in the Framework for Multiperspective Rendering [Yu and McMillan 2004a]. In this work, multiperspective rendering is achieved by combining piecewise GLCs that are constrained by an appropriate set of rules and interpolated rays that are weighted based on the distance to the closest fragment in a GLC region. Another type of camera model uses image surfaces to explicitly specify how sampling rays propagate in a scene [Glassner 2000; Brosz et al. 2007]. In Glassner's work, rays are defined by only two NURBS surfaces, whereas Brosz et al. use parametric surfaces to define a flexible viewing volume, and are thus able to accomplish non-linear projections such as fisheye or cylindrical projections. The Graph Camera developed by Popescu et al. [2009] introduces three basic construction operations on the frustum of a planar pinhole camera (PPC) to simultaneously show several regions of a 3D scene. Their General Pinhole Camera [Popescu et al. 2010] also provides focus+context views by manually specifying magnified central rectangular regions. Similar to the Graph Camera, Cui et al. [2010] introduced the curved ray camera, which generates a bent view frustum based on a sequence of PPC's and provides C^1 continuity at the connections between each PPC segment to circumvent occluders.

Much work has been done on non-linear projections of camera rays. The image-space constraint method is frequently used [Blinn 1988; Gleicher and Witkin 1992; Coleman et al. 2005]. Sudarsanam et al. [2008] created camera widgets which encapsulate specific aspects of non-linear perspective changes and allow users to manipulate both the widgets and their image-space views. Wang et al. [2005] presented a camera lens technique based on geometric optics for volume rendering. Camera rays are refracted according to the selected lens type at the image plane before being shot into the 3D volume. Instead of explicitly modifying camera rays, Mashio et al. [2010] introduced a technique that simulates non-perspective projection by deforming a scene based on camera parameters.

Multiperspective cameras have also been used in making panoramas for both single objects and complex 3D scenes. Wood et al. [1997] developed a system to create multiperspective panoramas for cel animation. In this work, camera paths are used to reconstruct a single panoramic image where local camera views have the appearance of a perspective projection. Rademacher and Bishop [1998] used a strip camera along with a continuous camera path to approach the multiple-center-of-projection image. Vertical push-broom cameras are, on the other hand, used in capturing facades along a street [Roman et al. 2004; Roman and Lensch 2006; Agarwala et al. 2006].

2.2 Other Types of Multiscale and Focus+Context Rendering

Multiscale rendering can also be achieved by using image-space or object-space deformation. Böttger et al. [2006] presented a technique for generating complex logarithmic views for visualizing flat vector data. A similar technique was later employed to visualize complex satellite and aerial imagery, showing details and context in a single seamless image [Böttger et al. 2008]. For 3D data, distortion that magnifies the focal region is also a common approach to achieve focus+context rendering [Carpendale et al. 1996; Wang et al. 2008; Wang et al. 2010].

3 Multiscale Image Composition

A continuous multiscale image is composed of two types of regions. The first type consists of the ordinary perspective views at each scale of interest, and the other type consists of the transitions that smoothly connect views at different scales. An intuitive way to create views at multiple scales is to use several pinhole cameras, with each camera capturing a perspective view at a different scale. In order to render transitions between scales, a naive approach is to separately render each view and then blend them together using either an illustration metaphor (as shown in Figure 3(a)), or a seamless image-stitching algorithm, such as the graph cut [Kwatra et al. 2003] or Poisson image editing [Pérez et al. 2003]. Although the images created by these methods appear to be seamless, the underlying content is not continuous in object-space, and thus can make it difficult for viewers to comprehend the true spatial relations between scales.

We introduce a multiscale rendering framework that generates camera rays which are cast non-linearly through all scales of interest. Since the camera rays in our model are bent coherently and march consistently, the objects projected on the image are hence continuous in both image-space and object-space. Our approach starts by setting up several pinhole cameras for viewing different scales of interest, utilizing most users' ease and familiarity with manipulating ordinary pinhole cameras. Each camera produces an image of its view. In order to combine all such views to form a single multiscale image, we use a user-specified image mask to indicate regions of each view that the user would like to show in the final image. In other words, every camera projects only part of its view onto the final image space, based on its image mask (Figure 4).

In order to ensure consistency while projecting different camera viewpoints onto different parts of the image, we force all camera rays to originate from the first camera, which is the one that has

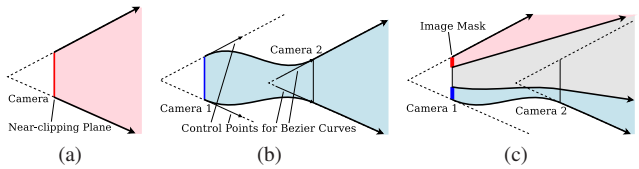


Figure 5: *(a) A pinhole camera and its view frustum. (b) The rays of Camera 2 are extended backward to couple with Camera 1 using Bezier curves. Small dotted arrows illustrate control points, which are the original ray orientations of the two views. (c) Different portions of the rays are assigned to different views based on the image mask. The red region is marked as a preserved viewing region for Camera 1, blue is for Camera 2, and the gray region is the transition between the two. As a result, interesting features of the two views can be seamlessly shown in the same image.*

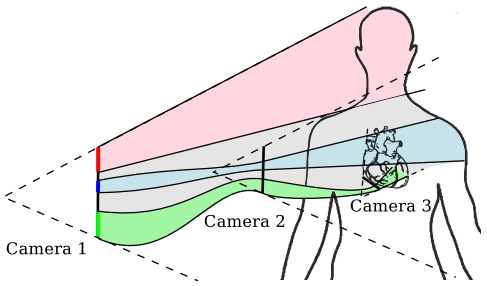


Figure 6: *Through careful image mask and camera placement, camera rays can cast through multiple scales to capture features of interest.*

the largest scale of view. The rest of the cameras define intermediate points that camera rays must pass through in order, from the largest to smallest scales. We use Bezier curves to connect the near-clipping planes of two successive cameras. Figure 5(a) illustrates a pinhole camera with its view frustum highlighted in red. Figure 5(b) shows Camera 2's rays extended at the near-clipped end to couple with Camera 1's rays to guarantee that every ray originates from Camera 1. Figure 5(c) depicts ray coupling with the application of an image mask. Note that a ray is curved to the next camera only when it originates in the region that is assigned to a descendant camera. In this figure, the red region indicates the preserved view for Camera 1, and the blue region indicates the preserved view for Camera 2. As a result, camera rays in the red region cast linearly, while rays in the blue region proceed from the near-clipping plane of Camera 1, march forward along Bezier curves, which are constructed based on the original ray directions, and arrive at Camera 2's clipping plane. Rays then continue to proceed linearly to capture the view of Camera 2. Figure 6 shows an example of three cameras simultaneously viewing at three different scales.

The gray regions between colored regions in Figure 5 and Figure 6 are transitional areas where camera rays need to gradually change between nearby colored regions to maintain good ray coherence. Two things are worth noting when dealing with camera ray bending in transitional regions:

Ray intersection. Camera rays that are bent at unequal degrees have a chance to intersect with one another. Simple ray direction interpolation such as linear interpolation between two nearest preserved rays can generate intersecting rays that might result in the same object being projected onto the image more than once.

Ray coherence. Adjacent camera rays must be coherent so as to avoid too much distortion in the resulting image. In other words, camera rays which are emitted from the pixels in a horizontal scanline should maintain their spatial relationship after bent.

The first issue is directly related to whether sampling rays can project the scene correctly. The second issue if properly addressed can minimize distortion and lead to aesthetically appealing view. We discuss these two issues and our solutions in the next section.

4 Camera Ray Generator

A key aspect of our design is the construction of a vector field, based on the multiple views specified by the user, to guide ray generation in the rendering process. Rays are derived by tracing streamlines in this vector field.

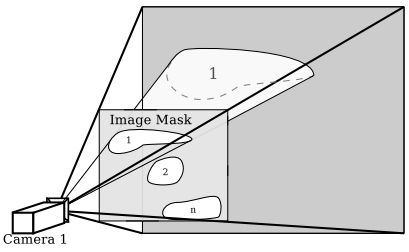


Figure 7: *When filling the mask scalar field, we trace only the camera rays in the corresponding restricted region. The above example illustrates the case where the mask value 1 is filled into the scalar field where the rays of Camera 1 pass through the corresponding region of the image mask.*

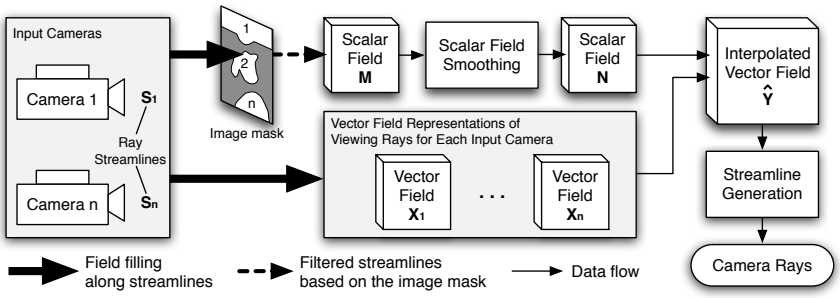


Figure 8: *The process of camera ray generation. Input camera rays are taken as streamlines and used to construct view vector fields $\boldsymbol{X}_1$ to $\boldsymbol{X}_n$. At the same time, a scalar field $\boldsymbol{M}$ is filled with the mask value along the filtered streamlines based on the image mask. Ray streamlines are generated upon the view vector field $\hat{\boldsymbol{Y}}$, which is derived by interpolating $\boldsymbol{X}_i$ according to the smoothed scalar field $\boldsymbol{N}$.*

4.1 Streamlines as Camera Rays

Streamlines are a set of curves which depict the instantaneous tangents of an underlying vector field. A streamline shows the direction that the corresponding fluid element travels in the field at any point in space. One characteristic of streamlines is that any two given streamlines would never intersect each other as long as no critical points are present [Fay 1994]. If we treat camera rays as streamlines, we can make use of this characteristic to ensure that no camera ray intersections can occur.

To derive streamlines for use as multiscale camera rays, the fundamental question is what the underlying vector field should be. Since streamlines represent tracks along values in their vector field, the construction of the vector field determines the paths of the streamlines. The problem then becomes: given a set of pinhole cameras and an image mask, how can we construct a vector field whose streamlines are distributed identically to the original camera rays in preserved regions, and gradually transition between the interpolated regions? To achieve this goal, consider that we have n cameras and each camera has its own set of rays, we then have n sets of streamlines $\mathbf{S}_1$ to $\mathbf{S}_n$. We construct a preliminary vector field $\mathbf{X}(\mathbf{x} = (x, y, z)) = (u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x}))$ by filling in the tangent directions of $\mathbf{S}_i$ with the vector values along streamlines if the streamlines originate from the region assigned to camera i in the image mask, and zero elsewhere. The complete vector field $\mathbf{Y}(\mathbf{x})$ can be derived by solving an optimization problem that minimizes the following energy function [Xu et al. 2010]:

$$\varepsilon(\mathbf{Y}) = \int \varepsilon_1(\mathbf{Y}(\mathbf{x}), \mathbf{X}(\mathbf{x})) + \mu\varepsilon_2(\mathbf{Y}(\mathbf{x}))d\mathbf{x} \qquad (1)$$

where

$$\varepsilon_1(\mathbf{Y}(\mathbf{x}), \mathbf{X}(\mathbf{x})) = |\mathbf{X}(\mathbf{x})|^2 |\mathbf{Y}(\mathbf{x}) - \mathbf{X}(\mathbf{x})|^2$$
$$\varepsilon_2(\mathbf{Y} = (u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x}))) = |\nabla u(\mathbf{x})|^2 + |\nabla v(\mathbf{x})|^2 + |\nabla w(\mathbf{x})|^2$$

The term $\varepsilon_1(\mathbf{Y}(\mathbf{x}), \mathbf{X}(\mathbf{x}))$ guarantees that the resulting vector field $\mathbf{Y}$ has exactly the same value as the preliminary field $\mathbf{X}$ at points where $\mathbf{X}(\mathbf{x})$ is not zero, and the second term $\mu\varepsilon_2(\mathbf{Y}(\mathbf{x}))$ is minimized when the neighboring vectors are identical, thus resulting in smooth transitions. The energy equation can be solved by the *generalized diffusion equations* described in the fluid flow literature [Hall and Porsching 1990], and is further discussed in [Ye et al. 2005; Xu et al. 2010].

Unfortunately, although the resulting vector field satisfies our requirements that streamlines pass through all views in preserved regions, and smoothly transition between preserved regions, streamlines in the transitional regions produced by this method may not preserve ray coherence in all cases. To take a simple example, suppose we use only one camera, but that we only assign a small portion of the image mask to camera 1. Since only streamlines marked as originating from the image mask are used to fill the vector field, the remaining parts of the resulting smoothed vector field will have the same vectors as the boundary of the marked region and thus fail to project the expected view to the original camera. Figure 9 illustrates a case of two cameras. One can see that the preserved rays for Camera 1 in red only provide a small amount of view vector information. Thus, the neighboring vectors on top of the preserved views have false values that lead to a deviated perspective projection as shown in Figure 9(a). Figure 9(c) shows that the false vector values also cause a distorted distribution of the camera rays that often result in a twisted image.

4.2 Scalar Field Mask

Directly smoothing the vector field results in deviant camera rays. In order to obtain coherent rays and minimize distortion in the projected image, we construct the vector field using vector values sampled in the original camera views. Consider that we have n cameras and n sets of streamlines $\mathbf{S}_1$ to $\mathbf{S}_n$ representing camera rays. For each set of streamlines $\mathbf{S}_i$, we can construct a vector field $\mathbf{X}_i$ whose values are the tangents along $\mathbf{S}_i$ and zero elsewhere. In short, $\mathbf{X}_i$ represents the vector field of the camera rays of camera i. Then we construct a preliminary scalar field $\mathbf{M}(\mathbf{x} = (x, y, z)) = m(\mathbf{x})$ in the same way that $\mathbf{X}(\mathbf{x})$ was constructed, as shown in Figure 7. However, instead of filling in tangent directions, we fill the field with the mask values of the streamlines' origins. Again, the complete smoothed scalar field $\mathbf{N}$ can be derived by optimizing a 1-D version of Equation 1. Based on the smoothed scalar field mask $\mathbf{N}$ and the original view vector fields $\mathbf{X}_1$ to $\mathbf{X}_n$, we can construct a final view vector field using the following function:

$$\hat{\mathbf{Y}}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{X}_i(\mathbf{x})\omega(\mathbf{N}(\mathbf{x}), i)}{\sum_{i=1}^{n} \omega(\mathbf{N}(\mathbf{x}), i)}$$

The term $\omega(\mathbf{N}(\mathbf{x}), i)$ is a weighting function that determines the weight for each view vector field according to the mask scalar field. For example, the following weighting function performs linear interpolation between two neighboring view vectors:

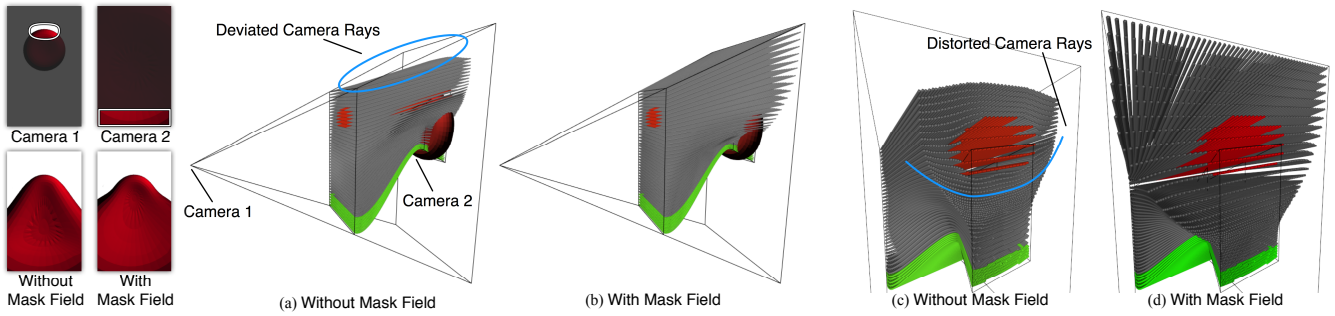


Figure 9: *Comparison of the results from direct vector smoothing and with the use of the scalar field mask. Two camera views with the corresponding masks are shown in the upper left images. The mask for Camera 1 contains only a small portion of the image area, which means only the camera rays in this preserved region are used to fill the view vector fields. The side view in (a) and the rear view in (c) point out that the curved rays generated by using direct vector field smoothing deviate from the expected perspective projection around Camera 1 (highlighted in the blue circle) and cause undesirable distortion (highlighted in the blue arc). (b) and (d) illustrate the coherent rays generated by interpolating the vector fields based on the smoothed scalar field mask. The resulting images are shown in bottom left.*

$$\omega(\mathbf{N}(\mathbf{x}), i) = \begin{cases} 1 - |\mathbf{N}(\mathbf{x}) - i| & \text{if } |\mathbf{N}(\mathbf{x}) - i| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Other types of interpolation, such as monotonic cubic interpolation, may be applied to increase the smoothness at the boundaries of transitional regions, but increasing the smoothness of boundaries implies a rapid change in the interior of regions.

4.3 Streamline Generation

The final step is to generate streamlines from the vector field $\hat{\mathbf{Y}}$. Since streamlines are used to simulate camera rays, each streamline should pass through a pixel on the projection plane (here we use the near-clipping plane of Camera 1). Therefore, we take the position of a pixel on the projection plane and use it as a seed point to trace a streamline using the Runge-Kutta method. Once we have all of the streamlines, we can render the final image using these streamlines as sampling rays. The entire streamline generation process is summarized in Figure 8.

5 Implementation Details

5.1 Multi-Resolution Vector Field

To implement vector field smoothing and streamline generation, we need to construct a 3D volume of the vector field that encloses the entire space, including all objects and camera frustums. This means we only have a finite number of vector field samples. As a result, the resolution of the volume directly affects the accuracy of generated streamlines. This resolution issue becomes especially essential when streamlines are used to cast through multiscale objects, and hence it is imperative to use multi-resolution volumes to achieve adaptive vector sampling.

5.2 Ray Tracing

Because our rendering framework requires modification of view vectors for the camera rays emitted from each pixel, we choose ray tracing as the rendering method. However, most popular raytracers only support linear sampling rays in ray-object intersection tests. In order to achieve the non-linear ray tracing, we divide the generated curved rays into line segments and performe piecewise linear ray-object intersection tests. As shown in Figure 10(f), all the sampling rays originate from the near-clipping plane of Camera 1, and then march through the scene to reach the corresponding scales. In this figure, the colored rays represent the rays in each camera view and are near-linear to approximate linear perspective in the regions of interest. Moreover, grayscale rays in the transitional regions are gradually bent to create continuous and smooth projection.

Our method is also capable of performing multisampling ray tracing to obtain high quality anti-aliasing images. We first divide all the rays into equal numbers of line segments for subpixel sampling. When rendering using multisampling, a subpixel sampling point is determined by averaging the four neighboring points at the same number of the line segments with random weights. The final color can be derived by averaging the sampled colors within a pixel. We extend the last line segments as infinite linear rays to cast through the rest of the scene. Since the ray directions are altered, some of the rays might intersect with each other at certain points. We think it is acceptable and causes little effect on the projection because the intersections only occur outside the regions of interest.

5.3 Volume Ray Casting

For volume data, we use ray casting to sample data values along with transfer functions (TFs) to achieve direct volume rendering. Ray casting, by nature, requires a traversal of the data structure, and thus simplifies the implementation of a non-linear ray marching algorithm. However, interesting features in volume datasets are usually hard to segment clearly, and in many cases important features are wrapped in opaque surrounding volumes. For example, in the human body dataset as shown in Figure 1, organs are embedded in the epidermis and other tissues. Even though the camera rays are bent to couple with the views at small scales, the first thing that the rays hit is the skin. In such cases, the multiscale sampling rays result in a magnified view of the outer skin, and the interesting features such as the heart and blood vessels are all occluded.

In order to clearly visualize the features at different scales, we apply different TFs to the same object while sampling rays with different mask values. Transfer functions define RGBA colors corresponding to data intensities. In other words, TFs define how color and transparency represent data on the screen. In our system, users are allowed to specify separate TFs for each camera viewpoint so as to obtain better views at different scales. During ray casting, TFs between different views are interpolated based on the smoothed mask values to achieve continuous color transitions. The left three images

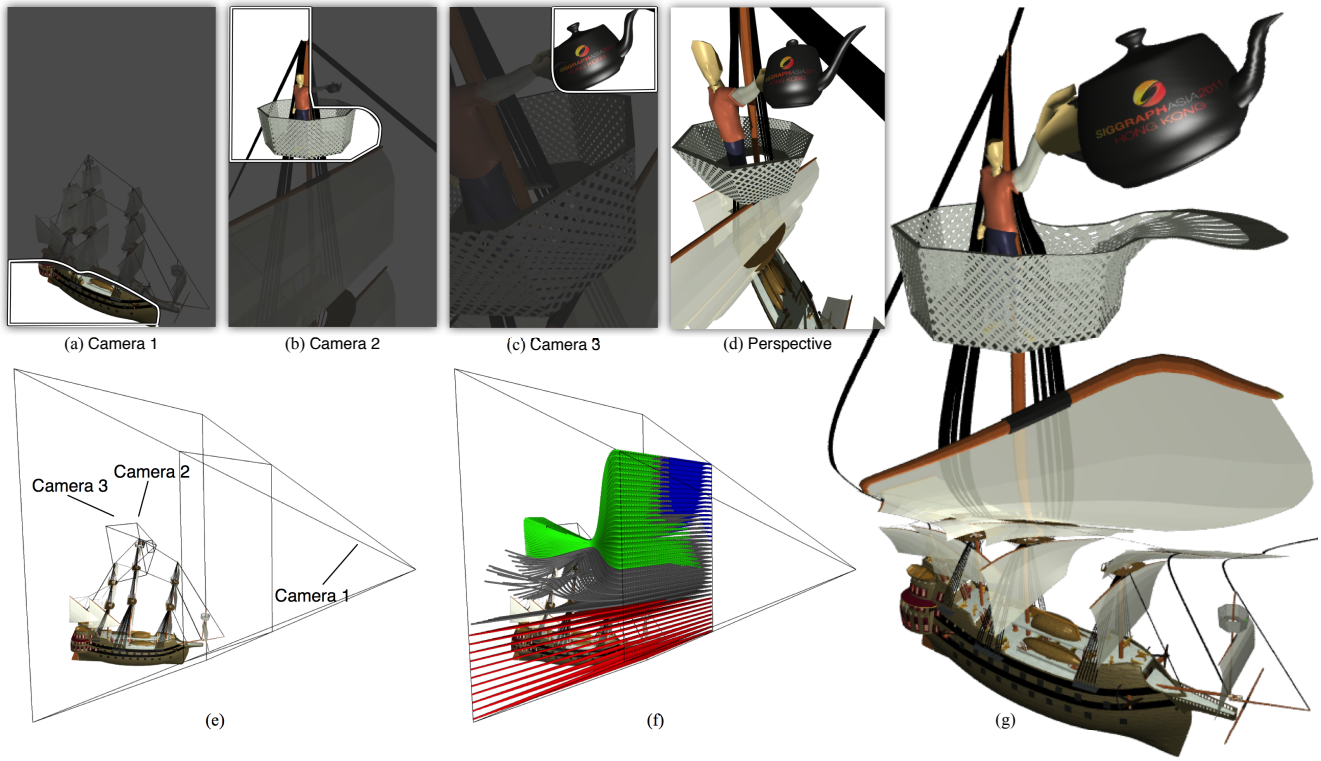


Figure 10: *Galleon dataset. (a), (b), and (c) on top show three camera viewpoints and the corresponding masks at different scales, from an overview of a Spanish galleon to a teapot held by a pirate on the galleon. The brightened parts of the three images are the user-specified preserved regions, whereas the remaining darkened regions are left as transitional regions. (e) shows the relative positions of the three cameras and the galleon in an ordinary perspective view, and (f) illustrates the generated non-linear camera rays in red, green, and blue for the three preserved views, respectively. The resulting multiscale image is shown in (g). (d) is a normal perspective viewpoint with a large field of view (FOV) which create a fisheye-like effect. One can see that even though the large FOV can contain parts of the galleon body, the viewer is still hard to tell the actual shape of the galleon due to its view angle.*

in Figure 1 show three camera views with different TFs applied to the same dataset.

6 Discussion

We have introduced a new capability for creating novel views of 3D models. Our current design demonstrates attractive results and suggests several interesting uses, but is not without limitations.

6.1 Limitations

Although our approach can generate non-linear sampling rays for direct multiscale rendering simply based on a set of conventional pinhole cameras and image masks, the quality and beauty of the final image are highly dependent on users' efforts in selecting image masks and camera viewpoints. First of all, objects projected in each view must have coherent spatial relationships to be able to create reasonable transitions. Second, design of image masks directly affects relative positions of each view as well as available space for transitional regions. As a result, even for the same set of camera views, resulting images can be quite different when different image masks are applied. Contradictory preserved views also result in undesirable images. For instance, rays in two preserved views that already intersect would cause unexpected problems while filling values in the scalar field. In such cases, users need to adjust the near-clipping and far-clipping distances of viewpoints to maintain the exclusivity of preserved views.

Adequate volume resolution for vector field calculation is also a crucial factor in producing high quality multiscale images using our method. Insufficient volume resolution will result in a lack of detail in streamlines at small scales, and noticeable errors in projecting preserved views. In Section 5.1 we mentioned the use of multiresolution volumes. However, discontinuous vector fields at junctions between different resolutions also introduce slight streamline perturbations, sometimes resulting in perceivable artifacts in projected images. In order to derive more coherent streamlines, higher-order vector sampling strategies at resolution junctions are needed to reduce streamline discontinuities.

Finally, our multiscale rendering framework supports both polygon models and volume datasets. However, most popular raytracers do not support non-linear ray tracing because ray-object intersection tests are much trickier for non-linear rays. Therefore, a custom raytracer is required in order to fully apply our framework to geometric scenes.

6.2 Applications

Presenting hierarchical structures. As mentioned earlier, multiscale images are especially useful in presenting objects with hierarchical structures. Showing continuous views at multiple scales in the same image helps viewers comprehend the spatial relation-

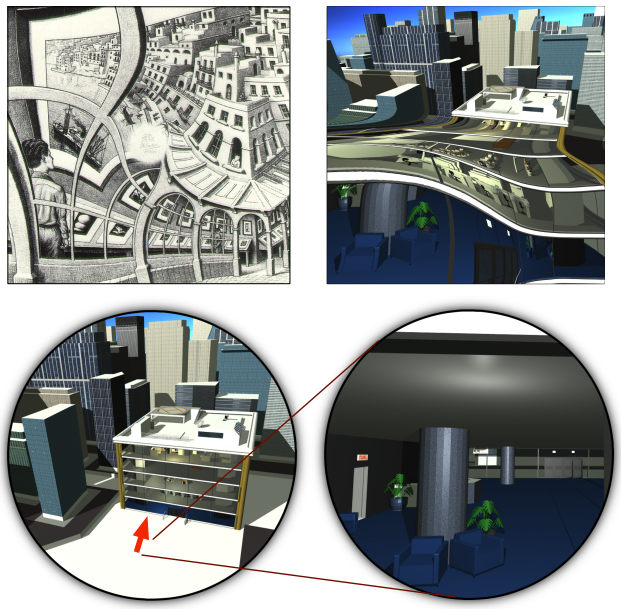


Figure 11: *Top Left: M. C. Escher's "Print Gallery." Top Right: our multiscale image of a 3D city model. Notice the similarity of the transitions from building outlines to indoor views in the two images. Bottom: The two camera views that show the city skyline and the building interior, respectively.*

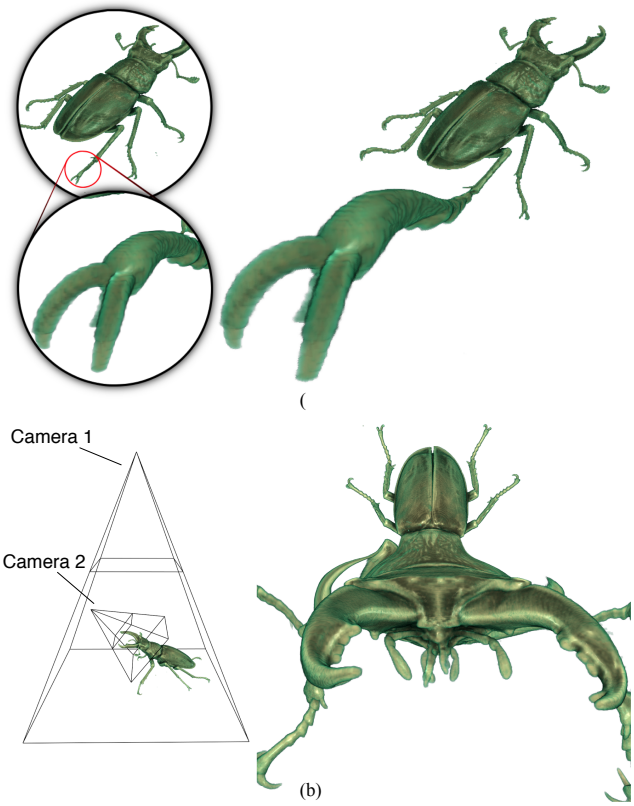


Figure 12: *Stag beetle volume dataset. (a) The image on the right creates a focus+context effect, simultaneously and seamlessly showing both an overview of the stag beetle and a closeup view of one of its limbs. The multiscale view is created using the two camera viewpoints shown on the left. (b) Another focus+context view that combines two different view angles, illustrates the body of the stag beetle from a bird's-eye view and zooms to its head in a front view.*

ship between scales. As illustrated in Figure 4, the three different scales: the upper body, the heart, and the aorta in a human body volume dataset can be revealed by the three different viewpoints. The brightened regions in the left three images are the corresponding preserved areas that are specified by the user in each view. Notice that the transfer functions applied to the dataset in each of the three views are slightly different to show interesting features more effectively. The resulting multiscale image is shown in Figure 1.

Focus+context effect. Most camera models capable of focus+context rendering (such as the Magic Volume Lens [Wang et al. 2005]) redirect a portion of camera rays in order to achieve partial magnification. Since ray direction alterations in these models are based on single cameras, views of magnified portions are similar to views of context. However, features at different scales usually require different viewpoints to best show interesting details. In our multiscale camera model, users can easily specify desired viewpoints for each level of scale. In addition, our model is also able to handle multilevel focus+context rendering, since users can set up more than two cameras, with each of them capturing different levels of detail.

Figure 10 shows an example of focus+context rendering. In this 3D model, a pirate is standing in the crow's nest on top of the mast of the galleon holding a teapot. The top three images (a)-(c) in Figure 10 are three different camera viewpoints showing different levels of detail in the scene. The side view in Figure 10(e) provides an overview of the scene that depicts the relative positions and scales of the three cameras and the model. Figure 10(f) illustrates the generated non-linear rays, where red rays are assigned to Camera 1, green to Camera 2, blue to Camera 3, and gray rays to transitional regions. Based on the three camera views, the resulting image in Figure 10(g) clearly shows a continuous view from the whole galleon to the pirate on the mast and the teapot in his hand. Despite the significant difference in scales, our multiscale camera model makes it possible to show an overview of the galleon as well as a detailed view of the text on the teapot in a single continuous image. Figure 12(b) demonstrates focus+context rendering with greatly differing view angles between focus and context views. Camera 1 provides a bird's-eye view directly above the stag beetle, while Camera 2 shows a closeup view of the front of the beetle. The resulting image on the right combines both an overview of the beetle's body and a focused view of the beetle's oral cavity in a single, continuous image.

Artistic rendering. Multiscale rendering techniques may also be applied in many works of art in order to create novel views. In addition to showing different levels of detail, multiple scales in a single illustration help guide the viewer's eye towards interesting parts of the image. Many works by the mathematician and artist M. C. Escher are good examples of multiscale images. Figure 11 compares his well-known illustration *Print Gallery* to a multiscale image of a city model. To generate this image, we combined views of the city skyline and the building interior to create a smooth, curved transition between indoor and outdoor areas, mimicking Escher's style. We placed image masks in the upper portion of the the first viewpoint (Figure 11, bottom left) to provide a perspective view of the surrounding buildings, and in the bottom left corner of the second viewpoint (Figure 11, bottom right) to capture interior details. Our framework then generates a smooth transition between the two viewpoints accordingly.

7 Conclusion and Future Works

We present a novel rendering framework model for creating continuous multiscale views of 3D geometric models and volumetric data. Almost all appealing multiscale illustrations are handcrafted by skilled artists. As we have shown, it is attractive to blend multiscale views of complex structures in a single static picture. Furthermore, even for a single object such as the stag beetle shown in Figure 12, multiscale views can be applied to achieve focus+context rendering and to highlight interesting parts of the data. Finally, our multiscale rendering framework may also be used to emulate multiscale illustrations created by artists for objects and scene physically impossible or expensive to create.

There are a number of directions for future work especially to increase the usability and robustness of this new rendering technology. First, the implementation of a custom raytracer would fully enable multiscale rendering of geometric scenes using our camera model. Automatic viewpoint placement is another interesting topic to explore. Instead of specifying viewpoints and corresponding image masks, users select features of interest with respect to properties such as shape, size, location, etc. With proper constraints, camera views could be automatically determined using the properties and spatial relations of user-specified features, without resulting in undesirable artifacts. Finally, it is also possible to create animated views by allowing selected view points to move.

References

AGARWALA, A., AGRAWALA, M., COHEN, M., SALESIN, D., AND SZELISKI, R. 2006. Photographing long scenes with multiviewpoint panoramas. *ACM Trans. Graph. 25*, 3, 853–861.

AMERICAN, 2009. The known universe. Available at `http://www.amnh.org/news/2009/12/the-known-universe/`.

BLINN, J. 1988. Where am i? what am i looking at? *Computer Graphics and Applications, IEEE 8*, 4, 76–81.

BÖTTGER, J., BALZER, M., AND DEUSSEN, O. 2006. Complex logarithmic views for small details in large contexts. *IEEE Transactions on Visualization and Computer Graphics 12*, 5, 845–852.

BÖTTGER, J., PREISER, M., BALZER, M., AND DEUSSEN, O. 2008. Detail-in-context visualization for satellite imagery. In *Proceedings Eurographics, Eurographics Association*.

BROSZ, J., SAMAVATI, F. F., SHEELAGH, M. T. C., AND SOUSA, M. C. 2007. Single camera flexible projection. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, NPAR '07, 33–42.

CARPENDALE, M. S. T., CARPENDALE, T., COWPERTHWAITE, D. J., AND FRACCHIA, F. D. 1996. Distortion viewing techniques for 3-dimensional data. In *INFOVIS '96: Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, IEEE Computer Society, Washington, DC, USA, 46.

COLEMAN, P., SINGH, K., BARRETT, L., SUDARSANAM, N., AND GRIMM, C. 2005. 3d screen-space widgets for non-linear projection. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ACM, New York, NY, USA, 221–228.

CUI, J., ROSEN, P., POPESCU, V., AND HOFFMANN, C. 2010. A curved ray camera for handling occlusions through continuous multiperspective visualization. *IEEE Transactions on Visualization and Computer Graphics 16* (November), 1235–1242.

FAY, J. A. 1994. *Introduction to fluid mechanics*. MIT Press, Cambridge, MA.

GLASSNER, A. S. 2000. Cubism and cameras free-form optics for computer graphics. Tech. Rep. MSR-TR-2000-05, Microsoft.

GLEICHER, M., AND WITKIN, A. 1992. Through-the-lens camera control. *SIGGRAPH Comput. Graph. 26*, 2, 331–340.

GOOGLE, 2010. Google earth. `http://earth.google.com`.

HALL, C. A., AND PORSCHING, T. A. 1990. *Numerical Analysis of Partial Differential Equations*. Prentice Hall, Engelwood.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3, 277–286.

MASHIO, K., YOSHIDA, K., TAKAHASHI, S., AND OKADA, M. 2010. Automatic blending of multiple perspective views for aesthetic composition. In *Proceedings of the 10th international conference on Smart graphics*, Springer-Verlag, Berlin, Heidelberg, SG'10, 220–231.

MCCRAE, J., MORDATCH, I., GLUECK, M., AND KHAN, A. 2009. Multiscale 3d navigation. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, 7–14.

NATIONAL, 1996. Cosmic voyage. `http://www.imdb.com/title/tt0115952/`.

NYE, L., 2008. Zoom into the human bloodstream. Available at `http://www.nsf.gov/news/special_reports/scivis/winners_2008.jsp`.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3, 313–318.

POPESCU, V., ROSEN, P., AND ADAMO-VILLANI, N. 2009. The graph camera. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, ACM, New York, NY, USA, 1–8.

POPESCU, V., ROSEN, P., ARNS, L., TRICOCHE, X., WYMAN, C., AND HOFFMANN, C. M. 2010. The general pinhole camera: Effective and efficient nonuniform sampling for visualization. *IEEE Transactions on Visualization and Computer Graphics 99*, RapidPosts.

RADEMACHER, P., AND BISHOP, G. 1998. Multiple-center-of-projection images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 199–206.

ROMAN, A., AND LENSCH, H. P. A. 2006. Automatic multiperspective images. 537–544.

ROMAN, A., GARG, G., AND LEVOY, M. 2004. Interactive design of multi-perspective images for visualizing urban landscapes. In *VIS '04: Proceedings of the conference on Visualization '04*, IEEE Computer Society, Washington, DC, USA, 537–544.

SUDARSANAM, N., GRIMM, C., AND SINGH, K. 2008. Non-linear perspective widgets for creating multiple-view images. In *NPAR '08: Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, 69–77.

WANG, L., ZHAO, Y., MUELLER, K., AND KAUFMAN, A. 2005. The magic volume lens: an interactive focus+context technique for volume rendering. 367–374.

WANG, Y.-S., LEE, T.-Y., AND TAI, C.-L. 2008. Focus+context visualization with distortion minimization. *IEEE Trans. Visualization and Computer Graphics (Proceedings of IEEE Visualization 2008) 14*, 6.

WANG, Y.-S., WANG, C., LEE, T.-Y., AND MA, K.-L. 2010. Feature-preserving volume data reduction and focus+context visualization. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints.

WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. 1997. Multiperspective panoramas for cel animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 243–250.

XU, L., LEE, T.-Y., AND SHEN, H.-W. 2010. An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 1216–1224.

YE, X., KAO, D., AND PANG, A. 2005. Strategy for seeding 3d streamlines. *Visualization Conference, IEEE 0*, 60.

YU, J., AND MCMILLAN, L. 2004. A framework for multiperspective rendering. In *Rendering Techniques*, 61–68.

YU, J., AND MCMILLAN, L. 2004. General linear cameras. In *ECCV (2)*, 14–27.

YU, J., MCMILLAN, L., AND STURM, P. 2008. Multiperspective modeling, rendering, and imaging. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses*, ACM, New York, NY, USA, 1–36.