A Multi-Criteria Approach to Camera Motion Design for Volume Data Animation

Wei-Hsien Hsu, Yubo Zhang, and Kwan-Liu Ma, Fellow, IEEE

Abstract—We present an integrated camera motion design and path generation system for building volume data animations. Creating animations is an essential task in presenting complex scientific visualizations. Existing visualization systems use an established animation function based on keyframes selected by the user. This approach is limited in providing the optimal in-between views of the data. Alternatively, computer graphics and virtual reality camera motion planning is frequently focused on collision free movement in a virtual walkthrough. For semi-transparent, fuzzy, or blobby volume data the detection free objective becomes insufficient. Here, we provide a set of essential criteria focused on controlling camera paths to establish effective animations of volume data. Our dynamic multi-criteria solver coupled with a force-directed routing algorithm enables rapid generation of camera paths. Once users review the resulting animation and evaluate the camera motion, they are able to determine how each criterion impacts path generation. In this paper, we demonstrate how incorporating this animation approach with an interactive volume visualization system reduces the effort in creating context-aware and coherent animations. This frees the user to focus on visualization tasks with the objective of gaining additional insight from the volume data.

Index Terms—Camera motion planning, volume rendering, visualization, animation

1 INTRODUCTION

Visualization has become a necessary tool that many scientists use to directly validate their studies, explore data, and present findings. Using current open-source or commercial visualization software, scientists are enabled to create vivid pictures and animations of their computer simulations or imaging results. Animations are especially effective at illustrating complex structures, ambiguous spatial relationships, and dynamic trends. However, the production of complex animations continues to be beyond individual scientists and requires professional animators and authoring tools. This is because most visualization systems offer rather basic animation support based on interpolation from user-specified keyframes. Using an interactive user interface, users pick desired views and other visual settings. The animation is then created via keyframe interpolation with the user having very minimal control of this process. Further adjustments to the animation, particularly in complicated parameters such as the actual camera path, require professional tools that are time consuming for scientists to easily use.

In this paper, we present a solution to the camera motion design and path generation problem for making volume data animations. Camera control is a nontrivial problem in computer graphics [9]. A simple camera model has at least six degrees of freedom, not to mention the difficulty of setting up a sequence of views to elaborately connect several points of interest (POIs). Many different methodologies have been introduced to meet specific needs in robotics, character animations, computer games, and other arenas of computer graphics. Surprisingly, little attention has been paid to camera motion control for making animations in scientific visualization. Since scientists are presently working with increasing large-scale datasets and need to inspect features in complex volume data, we argue that camera motion path planning for navigation and presentation of volume data is as important as volume classification, feature identification, and rendering.

Volume rendering has become a primary scientific visualization tool. It is especially effective when the goal is to reveal complex 3D features involving multiple materials or intensity values. During volume classification, different colors and opacities are assigned to features by defining transfer functions. After classification is complete and with the support of interactive tools, locating interesting features in the data and picking good views to further examine the volume are generally easy tasks for scientists. However, designing camera movements and creating animations to present volume features is a totally separate step. The interpolated views for transitioning from one feature to another in the traditional keyframe-based approach can become problematic because the views generated by direct interpolation from consecutive keyframes can lead to wandering in the spatial domain. This is especially true when two keyframes have very different positions and view directions. Furthermore, in volume visualization, camera paths may penetrate opaque volumetric regions or the view may end up pointing away from the main feature. This results in disorientation for viewers. These situations are presently remedied by subtle introduction of additional keyframes to support the presentation of complicated volume data. To accommodate this challenge, our main objective is to achieve steadily oriented movements along with smooth and context-aware camera transitions to create effective animations.

Previous research on automatic view selection for volume visualization [5] is relevant here, particularly if the user has little prior knowledge about the data. In our work, we assume that the user has sufficient domain knowledge to select primary views of essential features in the data. For most scientific datasets, features of interest are often occluded by surrounding materials, and can only be revealed by users prior knowledge. In this case, a system that assists the user in designing clear camera motion paths to present features with sufficient contextual information is desired. A number of volume data properties (e.g., opacity) and perceptual principles (e.g., camera panning versus tilting) should be taken into consideration to achieve such a camera motion path. Furthermore, users might consider different camera motions to address different datasets or visualization needs. Therefore, we aim to generate the best overall camera paths, based on a set of criteria concerning the visibility and optical properties of each feature in the data, along with the length of the animation.

A multi-criteria-based decision making system requires specification of weights for each criterion. This most often requires users to have extensive knowledge of the relationships among and effects of all criteria. This problem is exacerbated when users need to re-generate entire paths in order to evaluate new camera path settings. To address this, we present a dynamic camera motion planning mechanism that incorporates multiple criteria into a single system. The system simultaneously provides instant visual feedback while users are adjusting the weights of criteria. Our motion planning method is based on the construction of a roadmap for the free space of the volume data.

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

* Wei-Hsien Hsu is with UC Davis. E-mail: whhsu@ucdavis.edu.
* Yubo Zhang is with UC Davis. E-mail: ybzhang@ucdavis.edu.
* Kwan-Liu Ma is with UC Davis. E-mail: klnma@ucdavis.edu.

Index Terms—Camera motion planning, volume rendering, visualization, animation
The roadmap is a node-link graph that is created from the medial axis transformation. The initial path is computed via the A* search algorithm. We further refine the generated camera path by considering it as a mass-spring system, in which the various criteria are encoded as forces. The user is allowed to dynamically tune the weights of criteria, and the resulting camera motion path changes accordingly in a smooth animated fashion. We also develop the visualization to show criterion forces along the path. This guides the user in this fine-tuning process.

Our work aims to help scientists create expressive animations without mastering complicated animation techniques or acquiring cinematographic knowledge. Domain scientists simply focus on identifying the key aspects of their data. Our methods can be easily integrated into visualization systems that support keyframe animation. By offering scientists such animation support, we hope to greatly enhance their ability to communicate with colleagues and the general public about their work along with using animations for scientific storytelling.

2 Related Work

Camera control and motion planning have been widely studied since the beginning of cinematography. Recently, camera planning has received increased attention in diverse fields as the need for a better understanding of complex 3D space has increased. Many methodologies have been established for different purposes, in the fields of cinematography [17], robotics [19], computer animation [27], medical diagnostic systems [8], volume visualization [5], and game engines [16].

Fundamental camera control models and early approaches for interactive and automatic camera control were surveyed by Christie et al. [9]. Recent approaches can be roughly divided into two categories. The first category consists of approaches that are related to viewpoint selection, where a number of sampled views of a 3D object are evaluated and suggested by the system. Sokolov et al. [29] and Zhao et al. [39] presented a technique to evaluate the quality of a viewpoint for a scene, and described how this information can be used to generate exploration paths for virtual worlds. A unified information-theoretic framework has been proposed for viewpoint selection and mesh saliency [13]. Considering the Shannon entropy at a set of locations on a viewpoint sphere around an object, their framework efficiently searches for optimal viewpoints within a closed scene.

The second category consists of methods that take various constraints, such as geometric collisions and visual occlusions, into account to achieve globally optimal camera motion planning for complex and open scenes. Benhamou et al. [3] converted the motion planning problem into a nonlinear function optimization problem, and they proposed analytical methods which examine occlusions and collisions to efficiently obtain solutions. Sanjyal et al. [27] presented a framework for generating tours which focus on interesting objects. By constructing the visibility cells and considering the influence of various objects, their graph-theoretic optimization method can efficiently prune a large number of walkthrough paths and find an optimal solution.

The roadmap technique [19] that is commonly used in robotics and computer game engines for globally optimal path finding has also been utilized in many camera motion planning techniques. Drucker and Zeltzer [12] proposed a camera framework for navigating through virtual environments. Their approach exploited a constraint solver to find a collision free path, but they demonstrated the framework using a rather simple case, a virtual museum tour which was basically based on a 2D navigation map. Li and Ting [20] proposed an intelligent user interface for motion planning for 3D navigation. They adopted the probabilistic roadmap approach to help users avoid unnecessary maneuvers due to collisions with the environment. Andjújar et al. [2] constructed a cell-and-portal graph using a distance-to-geometry field over a 3D grid. The graph was then used to automatically generate guided tours for a walkthrough model. Ozaki et al. [23] used a 2D roadmap graph, an entropy map, and an occlusion map to achieve real-time viewpoint evaluation. Their method automatically generates smooth chase camera movements to follow either a subject, a user-controlled character, or a character with unpredictable behaviors in a 3D environment. Oskam et al. [22] presented a real-time global camera path planning algorithm for complex environments using a 3D discretized roadmap. Their method finds a coarse path first, then refines the initial path based on a sequence of occlusion maps computed on-the-fly in a dynamic environment. Hence, the algorithm ensures that the path followed by the camera is smooth in both space and time. In our work, instead of tracking objects in a dynamic virtual world, we aim to build a framework in which the roadmap is scalable and adaptable on a lattice grid and the motion planning is flexible to dynamically accommodate difficult visualization purposes.

In volume visualization, research has been conducted into developing a better animation support for exploring and revealing complex spatial and temporal structures in volume data. Akiba et al. [1] developed a template-based animation tool which directly transforms the results of volume data exploration into animation. Mühler and Preim [21] presented a technique for medical visualization that enables exploration results to be easily reused, and animations to be visually designed. Wu et al. [37] proposed a palette-style volume visualization interface that can be used to generate animations using a palette wheel. Yu et al. [38] presented a digital storytelling approach that generates automatic animations for time-varying data visualization using an event graph structure. Their focus was on alleviating the difficulty of creating visualization animations, especially for non-expert users. However, minimal research attention was devoted to addressing camera motion planning for volume visualization animations.

Advanced view selection techniques for ray-casting volume rendering have been proposed since 2005, when Bordoloi et al. [5] applied information theory to volume entropy calculation by evaluating noteworthy voxels in a given view. Ever since then, many other optimal view selection approaches using different constraints have been developed to serve different visualization purposes and datasets. Takahashi et al. [30] presented a method that locates optimal viewpoints by estimating the visibility and entropy of iso-surfaces based on a similar method for 3D surface meshes. Later, Viola et al. [33] introduced an information-theoretic framework for evaluating the importance distribution among features in a volume based on the mutual information measure. Vázquez et al. [32] used measures of multi-scale entropy and algorithmic complexity to achieve an adaptive method for representative view selection and exploration path generation. Ruiz et al. [25] developed the viewpoint information channel which evaluated the visibilities of voxels within volume data. The derived per-voxel information values can be used to select the most informative viewpoints. But all these techniques have the limit that the selected views and/or camera motion paths always surround the object. Other than the optimal view selection, certain types of medical datasets (e.g. cardiovascular or colorectal datasets) usually contain complex shapes and require a fly-through navigation within the data. Kang et al. [18] and Chen et al. [8] have proposed interactive and automatic flight path generation techniques based on the distance mapping from colon iso-surfaces for virtual colonoscopy systems. Wan et al. [34] also used a distance from boundary field to extract the centerline of volume structures to achieve interactive and automatic fly-through navigation in a volumetric environment. Diepenbrock et al. [11] presented an image-based navigation technique that aids user exploration of volume data by avoiding collisions with opaque material. However, these distance field- or image-based navigation methods are essentially designed for particular types of medical data and cannot be used in general volume data which usually contains sparse or semi-transparent regions.

In order to generalize the camera path optimization process and to apply it to animation generation for volume visualization, we utilize the medial-axis-based roadmap technique [36] since it has proven effective in capturing the shape of free space [15]. We also introduce several essential criteria for computing camera paths to create effective animations for general semi-transparent volume data. We develop a dynamic system in which we incorporate all the camera constraints such as opacity, occlusion, visibility, viewing direction, distance, and smoothness so that any adjustment in the weighted criteria can instantly be reflected in the visualization system. Our method finds a set of values that produce generally good camera motion paths, but we believe that with our interactive system, users can easily create a camera motion path that fits their needs.
3 DESIGN CONSIDERATIONS AND FRAMEWORK OVERVIEW

Most visualization systems use trackball navigation as the major camera manipulation metaphor, owing to its intuitive user interface for 3D view rotation. Due to its fixed center of focus, the trackball camera is most suitable for exploring a single object or providing an overview of a group of features. As data become more and more complex, scientists may find it necessary to closely investigate multiple POIs within the data. Although many feature selection methods have been developed to help users pick camera views for individual features, creating camera motion for connecting separate POIs still mainly relies on manual specification. In this section, we summarize a number of data-, vision-, and perception-driven principles and give an overview on our camera planning framework that incorporates these principles.

3.1 Design Considerations

Volume datasets are unique for its semi-transparent nature in the way it is rendered. During an animation production when scientists need to pick a set of POIs within a complex or cluttered dataset in order to illustrate a sequence of phenomena, viewers can possibly lose their focus due to inappropriate camera transitions between viewpoints, especially when targets are moved out of sight or the context fails to provide coherent clues about relative positions. Therefore, the visibility of interesting features is usually the primary consideration in constructing camera transition paths since the ultimate goal of a camera is to show subjects. And the visibility of the context is also important in conveying spatial relations during viewpoint switching.

In the camera motion planning problem, the quality of a generated camera motion path depends on the context of its evaluation. For a simple camera model with six degrees of freedom, a view is mainly defined by the camera position and the camera orientation. The camera position determines if the camera is placed somewhere that has high visibility of the target or contextual regions, while the camera orientation determines the actual view from that position. We come up with a number of criteria for determining camera positions and orientations to guarantee the visibility and stability along the camera motion paths.

3.1.1 Camera Position

We find that two critical properties, opacity and degree of occlusion, in volume data significantly impact the visibility for a camera view. The opacity of the volume directly determines if a camera can see through the regions, and the occlusion tells how much contextual information a view can provide. And since the visibility of the targets is the uppermost concern, it is desirable to move the camera toward the positions which have as high visibility as possible.

When users want to switch views from one focus to another, a Google Earth-like transition (zooming out from the current location, panning, then zooming in to the next location) can be used to reveal the surrounding context to provide extra clues for spatial relationships [31]. And how much the view needs to be turned or zoomed out also depends on the angular difference of the viewing directions.

The camera orientation should also be taken into consideration in determining the camera path. According to the cinematography, it is easier for human to track objects in camera panning (horizontal camera motion) than tilting (vertical camera motion) because of the wider aspect ratio of the field of vision of human eyes. In other words, viewers can receive more information in a horizontal camera motion than a vertical motion. As a result, it is preferable to avoid too much vertical movement to maintain its perceptibility. For the data which have clear upright orientations, it is desirable to align the camera up vector with world up. For the data without clear orientations (such as flow simulations), assigning orientations to objects can also help viewers recognize spatial relations [24].

Other criteria that affect the overall camera path include the length of the camera path and the smoothness of the curve.

3.1.2 Camera Orientation

Carefully designing camera orientation along its movement path is also crucial for providing quality camera transition. To determine the camera orientation, two straightforward design strategies can be used.

- The viewpoint follows the tangent of the camera path.
- The viewpoint focuses on the current or next POIs.
- The viewpoint follows the tangent of the camera path.

The first strategy works well when the data volume is less crowded or POIs are relatively close, and provides better contextual information. The second strategy works better for highly occluded volumes and longer paths, where it is hard to see the next POI before reaching it. In such a case, it is better to point the camera view to the movement direction so that the viewer can easily realize where they are going.

3.2 Framework Overview

Fig. 1 depicts the workflow of our multi-criteria camera motion planning framework. For volumetric data, we first define a configuration space based on an opacity threshold, and build a roadmap representing the free space where the camera is allowed to move. Based on the roadmap, we are able to generate globally optimal corridors connecting user-selected views. We develop a dynamic system that incorporates the aforementioned criteria and allows users to interactively refine the resulting camera motion path.
Fig. 2. Upper: Using the same number of sample nodes, the medial axis consistently captures a small tunnel structure that a probabilistic roadmap (PRM) sometimes does not. In addition, the medial axis produces higher quality roadmaps. Bottom: The adaptive octree partition requires a total of 124,072 sample nodes in order to represent very small structures (down to 8 voxels) in the \(500 \times 500 \times 100\) hurricane dataset, while the medial axis representation only needs 4,257 nodes.

### 3.2.1 Points of Interest

View selection for volumetric features is not a trivial problem, and, in most cases, is data-dependent. Automatic view selection for individual POIs is beyond the scope of this paper. In our system, we provide three interactive methods for users to specify POIs. First, we exploit the picking technique for volume visualization described in [35]. We use an additional user-specified radius to define the size of a region of interest, and the camera for this POI is determined in reference to the view being used in the picking process.

For some datasets with rather clear iso-surfaces, we allows users to set POIs by picking target surfaces based on an opacity threshold, as shown in Fig. 3. Our system then suggests a camera view for the selected region based on the surface normal information of the volume data. Furthermore, in addition to single static views, we can also produce extra views using rotational motion to present the same region of interest, because rotational motion provides very effective depth information for revealing complex spatial structures and/or possibly occluded features [24].

In some cases, experienced users may want to have the full control of the camera to set up the views for certain POIs. And our system allows for easy integration with the traditional keyframe-based animation system and lets the users set arbitrary key views.

### 3.2.3 Dynamic Path Refinement

The initial path, which is based on the roadmap graph, coarsely represents the globally optimal corridors that connect interesting points in the data. The next step is to turn the initial path into the final camera motion path by refining it using the aforementioned criteria. We develop a dynamic system that incorporates these criteria and transforms the multi-criteria optimization problem into a force balancing problem. The force-based method is used in many visual guidance techniques and is proven effective in many interactive applications such as motion planning [23] and model deformation [7]. We use the initial path to create a mass-spring system and convert the criteria into forces. The six essential criteria guarantee a generally good camera path. However, the dynamic system provides much more flexibility by allowing users to interactively adjust the balance of criteria, further tuning the resulting path. We also develop methods for visualizing criteria forces along the camera path, to help users better understand the effects of different criteria on the path generation process.

### 4 Technical Approaches

#### 4.1 Roadmap Construction

The medial axis is a shape descriptor introduced by Blum [4] and has many applications, including shape recognition, topological analysis, and collision detection. The definition of a medial axis is a set of centers of maximally-sized spheres that fill a shape. By taking the configuration space as the target shape, the medial axis can be used to construct high quality roadmaps for motion planning [28]. However, medial representations in 3D actually consist of 2D sheets instead of simple 1D “axes”, so it is difficult to extract clear 3D skeletons.

The sphere fitting algorithm is a common approach to approximating skeletons for 3D shapes. In practice, for binary images, the discrete medial axis can be efficiently extracted from a distance transformation, and both can be computed in \(O(n)\) time [14]. For a semi-transparent 3D volume dataset, we define free space using an opacity threshold. Next, we perform the squared Euclidean distance transform (\(DT\)) described in [14] for the configuration space. Theoretically, the medial...
axis is at the local minima of the second derivative of the distance transform. Thus, we compute the discrete medial axis \((\mathcal{MA})\) representation by applying the Laplace transform to \(\mathcal{DT}\). We use “squared” Euclidean \(\mathcal{DT}\) to construct \(\mathcal{MA}\) so that the distance information is encoded in \(\mathcal{MA}\), which can then be used to ensure that the sphere-tree is traced in a sparseness-first order.

To build the sphere-tree, we first find the position with the global minimum in \(\mathcal{MA}\) and use it as the root. With a threshold \(\varepsilon\) for maximal \(\mathcal{MA}\) values, we trace the sphere-tree by recursively connecting tree nodes to candidate nodes on the sphere surface in ascending order of \(\mathcal{MA}\). The algorithm terminates after the entire configuration space is covered, which guarantees \(O(n)\) runtime. We outline our sphere-tree tracing algorithm in Algorithm 1.

### Algorithm 1: BuildSphereTreeFromMA(\(\mathcal{DT}\), \(\mathcal{MA}\), \(\varepsilon\))

**Output:** tree \(T(V_T, E_T)\)

1. Queue \(Q\) {sorted by element’s \(\mathcal{MA}\) value}
2. CoveredMap \(M: \mathbb{R}^3 \rightarrow\) bool {initialized to false}
3. ParentsMap \(P: \mathbb{R}^3 \rightarrow \mathbb{R}^3\)
4. \(r \leftarrow\) the position with the global minimum in \(\mathcal{MA}\)
5. \(P[r] \leftarrow r\) {the root of the tree}
6. \(Q\).push\(\(r\)
7. while \(Q\) is not empty do
   8. \(n \leftarrow Q\).pop()
   9. if \(M[n] = \text{false}\) then
      10. \(V_T \leftarrow V_T \cup n\)
      11. \(E_T \leftarrow E_T \cup \text{edge}(P[n], n)\)
      12. \(D \leftarrow \mathcal{DT}(n)\)
      13. for all \(x\) | distance\((x, n) < D\) do \(M[x] \leftarrow\) true
      14. for all \(x\) | distance\((x, n) = D\) and \(\mathcal{MA}(n) \leq \varepsilon\) do
         15. \(Q\).push\(n\)

The output from Algorithm 1 is a tree-structured graph. To make it usable as a roadmap, we add additional edges to the graph. We use the criterion \(K \times \text{distance}(v, v') < T(v, v')\) described in [15] to determine if an extra edge would be useful. If so, we use \(D \times \min_{x}(v) < \text{distance}(v, v')\), where \(\min_{x}(v)\) is the distance from \(v\) to its closest neighbor, to determine if the edge \((v, v')\) should be added to the graph.

### 4.2 Coarse Path Finding

Given a starting camera position and a destination position, we first connect the two positions to their \(k\) nearest and visible nodes. Then, we use the A* search algorithm to compute a path from the source to the destination in the graph. This algorithm is known as “best-first search” and is widely used in computer games for its high efficiency and sufficiently good accuracy. The use of a heuristic function greatly accelerates searching a graph that explicitly lies in a coordinate system. The heuristic function is \(f(n) = g(m) + h(n)\), where \(g(m)\) is the cost evaluation function for the edge \(m\), and \(h(n)\) is the heuristic estimate function (usually the Euclidean distance from the current node \(n\) to the destination node). Here, different evaluation functions \(g(m)\) can be used to produce different initial paths. For example, consider the Euclidean length of the edge \(m\) (i.e. \(g(m) = \text{distance}(m)\)); we have the shortest path as an initial path. Normally, if no straightforward connection exists, the shortest path between two points may follow an opaque surface in the volume data. This is undesirable because such regions are usually semi-transparent or highly occluded. Since our roadmap is built upon the medial axis of the free space, our approach ensures that even the shortest path maintains a reasonable distance from opaque regions. Fig. 5 shows different initial paths based on other evaluation functions such as opacity and occlusion.

### 5 Path Refinement

The initial path derived from the roadmap consists of a set of piecewise linear segments connecting the source point and destination point. Such a path represents a globally optimal corridor for the two points but is a \(C^0\) path and only considers one criterion such as the shortest path. Our next step is to use multiple criteria to further refine this path.

#### 5.1 Criterion Forces

To solve a multi-criteria problem, a common approach is to define objective functions for different criteria and find solutions by optimizing the weighted sums. However, in camera motion planning problem, setting up objective functions for the criteria would be rather difficult because preferable camera paths vary case by case. Our solution is a dynamic multi-criteria solver in which we adopt the mass-spring system and convert the criteria into different forces. The major advantage is that users are able to get visual feedback of the changing path while tuning the weight of a criterion. We introduce the six criterion forces in the following sections.

##### 5.1.1 Spring Force

The first step is to convert the initial path into a mass-spring system. We subdivide the initial path into \(M\) pieces based on a small distance interval and connect consecutive sub-nodes with springs. According to Hooke’s law, we can define the force for the \(m\)th spring as \(F_{\text{spring}}(m) = -K_s(m)\bar{x}\), where \(K_s(m)\) is the spring constant and \(\bar{x}\) is the displacement of the spring’s end from its equilibrium position. Thus, to calculate the spring force, we need to define the spring constant and the relaxed length of the spring. We divide the spring constant into a global component and a local component in order to gain more controllability, i.e. \(K_s(m) = K_s-\text{global}K_s-\text{local}(m)\), and the relaxed length of a spring is the equal division of the straight-line distance of the source and destination, \(l_0 = \text{distance}(S, T)/M\). For each sample node \(n\) along the camera path, the node is pulled by the two connected springs.

\[
\bar{F}_{\text{spring}}(n) = \frac{F_{\text{spring}}(m)}{2} \hat{V}_{n-1,n} + \frac{F_{\text{spring}}(m+1)}{2} \hat{V}_{n+1,n} \quad (1)
\]

where \(m\) and \(m+1\) are the two connected springs, and \(\hat{V}_{n-1,n}\) and \(\hat{V}_{n+1,n}\) are the unit vectors of the directions to the two neighbors.

Since the spring forces for a sample node are computed according to its two neighbor nodes, the resulting path is guaranteed \(C^1\) continuity. Furthermore, the spring forces along the entire path basically tend to stretch into a straight line to create a shortest path.
5.1.2 Opacity Gradient
In volume visualization, data classification is usually done by specifying a transfer function which maps data intensities to colors and opacities. With the technique, direct volume rendering can create unique semi-transparent pictures to reveal inner structures of scientific data. However, when the camera flies through such semi-transparent regions, the visibility certainly decreases, and the occluded view can suddenly disorient the viewer. Since the configuration space of our roadmap is defined with an opacity threshold, there is possibility that some regions in the free space are, although not fully opaque, semi-transparent with the opacity slightly below the threshold. So the ability to bypass such regions can generally ensure better camera views during the view transition. We encode the force as

\[
\vec{F}_{\text{opacity}}(n) = -K_O(n) \left( \alpha(n) + |\nabla \alpha(n)| \right) \frac{\nabla \alpha(n)}{|\nabla \alpha(n)|}
\]

(2)

where \(K_O(n)\) is the criterion weight for node \(n\), \(\alpha(n)\) is the opacity at the location of sample node \(n\), and the direction of the force is determined by the inverse of the gradient of the opacity. The intensity of the force is defined by the magnitude of the gradient plus the opacity value because the path passing through a high-opacity but low-gradient region should also be pushed away from the region.

5.1.3 Occlusion Gradient
Another important property of volume data is occlusion. Occlusion, especially ambient occlusion, has been frequently used in volume visualization for enhancing structural perception and realism. Considering occlusion information proves to be effective in revealing spatial structure of volumetric features. In camera motion planning, occlusion information can be used to prevent a path from going too close to severely occluded region, where the cameras placed within these regions have high possibilities to have limited views. Distance transform can achieve a similar effect, but DT does not take semi-transparency into account, and \(\nabla(DT)\) only leads to the medial axis. Unlike traditional ambient occlusion, which only considers the neighbors in the hemisphere defined by the normal of a point, we use a more general notion of occlusion described in [10], which also takes into account the effects of the neighbors in the other hemisphere to derive the occlusion information in all directions. This can be expressed as:

\[
O(x) = \int_{\Omega} \int_{\Delta} (1 - \alpha(x, \alpha, \delta)) W(\delta) d\delta d\omega
\]

(3)

where \(\Omega\) is the sphere of directions, \(\Delta\) is the maximum distance from \(x\), \(\alpha(x, \alpha, \delta)\) is the opacity at the position \(\delta\) away from \(x\) along \(\omega\) direction, and \(W(\delta) = e^{-\beta \delta^2}\) is an exponential weighting function [26]. So the force can be encoded as:

\[
\vec{F}_{\text{occlusion}}(n) = -K_O(n) \left( O(n) + |\nabla O(n)| \right) \frac{\nabla O(n)}{|\nabla O(n)|}
\]

(4)

Similar to the opacity criterion, the intensity of the force consists of the magnitude of the gradient and the occlusion value. And the force tends to push the path away from convex surfaces and prevent from passing through a concave region so as to give the camera a better sight.

5.1.4 Visibility Gradient
During a camera translation, an important mission is to tell the viewer where the camera is heading to so that the viewer can expect the move- ments and understand the surrounding structures better. Thus it is desirable to show the target as early as possible. But in some cases, the POI can be occluded and can only be revealed through certain angles. In such cases, the visibility of the POI along the camera motion path can be used to guide the camera positions. And we define its force as:

\[
\vec{F}_{\text{visibility}}(n) = K_A(n) \left( \Lambda(n) + |\nabla \Lambda(n)| \right) \frac{\nabla \Lambda(n)}{|\nabla \Lambda(n)|}
\]

(5)

where \(K_A(n)\) is the weight, \(\Lambda(n) = 1 - \Gamma(n)\), is the visibility from \(n\) to the next target, and \(\Gamma(n)\) is the accumulated opacity along the straight connection from \(n\) to the next focal point. Notice that there is no negative sign in the equation, which means the force tends to push the path to high visibility regions.

5.1.5 Vertical Penalty
As mentioned in Sec. 3.1, less vertical movement can increase the stability of the camera view and enhance the comprehensibility of the spatial recognition. To achieve this, we introduce an additional penalty for vertical translations to the spring system based on the up vector \(\hat{U}\) of camera orientations. The vertical force can be defined as

\[
\vec{F}_{\text{vertical}}(n) = \frac{K_v(m) \text{Length}(m) (\hat{V}_{n-1,n} \cdot \hat{U}(n))}{2} \hat{U}(n) + \frac{K_v(m + 1) \text{Length}(m + 1) (\hat{V}_{n+1,n} \cdot \hat{U}(n))}{2} \hat{U}(n)
\]

(6)

where \(K_v(m)\) is the weight for the vertical penalty for the spring \(m\) and \(\hat{U}(n)\) is the unit vector of the camera up direction at node \(n\). In the case when an upright orientation is applied to the object, we can simply let \(\hat{U}(n) = \hat{U}\), a constant global up vector.

5.1.6 Repulsion from Focuses
Additional criteria can be added into the system by introducing other objective functions. Here we demonstrate a camera navigation technique that is frequently used in many navigation systems such as Google Earth, for switching from one camera viewpoint to another. The main idea is that when a camera view is leaving or approaching interesting points, we want to provide sufficient contextual information before the viewpoint starts to transit to the next or from the previous focus. This leads to a zoom-out or zoom-in camera operations when leaving or entering a focus view no matter which direction the actual motion transition is. The criterion can be described as a repulsive force and formulated as follows.

\[
\vec{F}_{\text{view}}(n) = \begin{cases} 
K_v(n)(w_1 \hat{v}_{11}(n) + (1 - w_1) \hat{v}_{12}(n)) & \text{if } n < n_1 \\
K_v(n)(w_2 \hat{v}_{21}(n) + (1 - w_2) \hat{v}_{22}(n)) & \text{if } n > n_2 \\
0 & \text{otherwise}
\end{cases}
\]

(7)

Here we define two supporting nodes \(n_1\) and \(n_2\) which are two intermediate nodes between the source and destination, and are the points where the camera should enter or leaves the camera transition trajectory, respectively. All the nodes before \(n_1\) or after \(n_2\) should be affected by the force. The four \(\hat{v}_{ij}\)s are defined by the directions to the source, destination, and intermediate points (e.g. \(\hat{v}_{11}(n) = \hat{n} - \hat{s}\) is the source focus point, \(\hat{v}_{12}(n) = \hat{n} - \hat{n}_1\), and vice versa for \(\hat{v}_{21}\) and \(\hat{v}_{22}\)). \(w_1(n)\) and \(w_2(n)\) are the weights for the vectors and can be simply defined linearly as \(w_1 = n = 1 / n_1\). With the capability to add extra objective functions, our framework can be easily extended to resolve further users’ needs on different camera motion needs.

Besides the six aforementioned forces, we introduce an additional damping force, \(\vec{F}_{\text{damping}}(n) = -K_d(n) \vec{x}(n)\) where \(\vec{x}(n)\) is the velocity of node \(n\), to stabilize the dynamic system. The net force determines the acceleration of a node. By considering the mass to be 1 for every node, the particle mass can then be omitted in the calculation. So the dynamic system can be updated by using the formulas: \(\dot{x} = \ddot{x}\) and \(x = x\). And the motion path is updated iteratively and is optimized when the system reaches a steady state.

Fig. 7. The colored line segments attached to the camera path represent the effect of a particular criterion. In the left image, the path runs too close to the semi-transparent corner where the opacity is high. By increasing the weight for the opacity force, the path is pushed away from the corner and stays in the clear area, as shown in the right image.
we apply a low-pass filter to the camera orientations are first calculated using the formula, and then the target, the camera angle should point to the target. In each iteration, defined in Sec. 5.1.4. For those nodes where the camera can see the \( \vec{v} \), where \( \vec{v} \) represent the intensity of the force (long and red lines indicate strong force). In this case, by increasing the weight for the force). In this case, by increasing the weight for the criterion forces at each sample point, and the color and length of the line are visualized as colored line segments that are attached to the camera path. The lines are pointing to the directions of the force at each sample point, and the color and length of the line represent the intensity of the force (long and red lines indicate strong force). In this case, by increasing the weight for the occlusion force, the path is pushed away from the opaque convex so the effect of the force also decreases. We use discrete samples and line segments to illustrate forces because we want to preserve as much contextual information as possible while visualizing the forces. Visualizing those forces as continuous curved planes could block too much surrounding region. And semi-transparent planes could confuse the viewers if the surrounding volume data are also rendered semi-transparently.

5.2 Camera Orientation Determination

We talk about the two design strategies for determining camera orientations in Sec. 3.1.2. In practice, in order to take advantage of the both strategies, we use the equation below to calculate the camera orientations along the motion path:

\[
\vec{V}(n) = \begin{cases} 
\vec{V}_{\text{source}}(n) & \text{if } n < n_1 \\
\vec{V}_{\text{target}}(n) & \text{if } n \geq n_1 \text{ and } \Lambda(n) > \varepsilon_v \\
\vec{V}_{\text{tangent}}(n) & \text{otherwise} 
\end{cases}
\]

where \( \vec{V}_{\text{source}}(n) = -w_1 \vec{v}_{11} - (1 - w_1) \vec{v}_{12}, \vec{V}_{\text{target}}(n) = -\vec{v}_{21}, \) which \( n_1, \vec{v}_{11}, \vec{v}_{12}, \) and \( \vec{v}_{21} \) are defined in Sec. 5.1.6, and \( \vec{V}_{\text{tangent}}(n) \) is the path tangent at node \( n \). \( \Lambda(n) \) is the visibility of the next focal point as defined in Sec. 5.1.4. For those nodes where the camera can see the target, the camera angle should point to the target. In each iteration, the camera orientations are first calculated using the formula, and then we apply a low-pass filter to \( \vec{V}(n) \) where \( \Lambda(n) \leq \varepsilon_v \), to avoid sudden view angle change during the transition.

5.3 Interactive Tuning and Visual Feedback

The dynamic system updates the camera motion path in real-time. This allows the user to interactively change the weight for each criterion to subtly tune the motion path. The weights for the criterion forces at a sample node are divided into global values and local values. Users can select a subset of the path and assign local weights to the selected nodes. Global weights control the overall trends, whereas the local weights provide minor adjustments for specific segments.

In order for users to more easily tune the path, we visualize the affecting forces along the path. Examples can be seen in Fig. 7, in which the occlusion force is visualized as colored line segments that are attached to the camera path. The lines are pointing to the directions of the force at each sample point, and the color and length of the line represent the intensity of the force (long and red lines indicate strong force). In this case, by increasing the weight for the occlusion force, the path is pushed away from the opaque convex so the effect of the force also decreases. We use discrete samples and line segments to illustrate forces because we want to preserve as much contextual information as possible while visualizing the forces. Visualizing those forces as continuous curved planes could block too much surrounding region. And semi-transparent planes could confuse the viewers if the surrounding volume data are also rendered semi-transparently.

6 EXPERIMENTAL RESULTS

6.1 Case Studies

We verified our camera model using three real volume datasets, including the temperature information of a computer room dataset, the MRI scan of a human brain tumor, and the velocity field of a hurricane simulation. For each dataset, Fig. 8 illustrates the original data visualization, the roadmap representation of the free space, the interesting views shown as yellow cameras and the generated motion path with gray cameras representing transitioning views, and the snapshot of one of the interesting views.

Computer Room Dataset has a dimension of 417 × 345 × 60. As shown in Fig. 8, it contains the air temperature information in a server room. The blue regions represent the cool air blew from the surrounding air conditioners of the room. From green to red, color indicates how much the temperature is higher than the normal room temperature. In the left side of the room where heat accumulates near the computers, red cloud occupies the aisles and blocks the views. Moreover, obstacles in the room also make camera path design a difficult task for the traditional keyframe animation. In our experiment, we pick five locations that have certain heat accumulation. And as Fig. 8 shows, the generated camera motion artfully routes around the obstacles and avoid penetrating opaque colorful cloud to ensure clear views. The altitude force is subtly used to help reduce too much vertical camera movement while showing the details of the temperature distributions.

Fig. 8. We demonstrate our camera motion planning algorithm using three datasets. From top to bottom: the temperature volume of a computer room dataset, an MRI brain, and the temperature volume of a hurricane simulation. Images from left to right represent the overview of each volume, the roadmaps for motion planning, a set of interesting views and the corresponding camera motion paths generated by using our method, and samples of interesting views along the actual camera paths.
Tumor Dataset is a $512 \times 512 \times 176$ 3D volume reconstructed from T1-weighted MR images. In this dataset, a tumor is deeply surrounded by the gray and white matter, blood vessels, and skull. In practice, such images from MR and other modalities are carefully examined and segmented by professional radiologists to produce high quality and accurate visualization for neurosurgical planning. An important task in preoperative imaging is to clearly examine the spatial relation of the brain tumor and the surrounding cerebral arteries and functional areas. In our experiment, we filter out the inner matter and only show the tumor and blood vessels to create enough navigation space. We pick a few views from outside and close views to the tumor. The results show that our approach is able to bypass those regions with either dense blood vessels or cluttered noises. The repulsion force in this case is particularly useful because it guides the camera to reveal more contextual regions while transitioning from one close view to another in the dense vascular area.

Hurricane Dataset contains a $500 \times 500 \times 100$ velocity field from a hurricane simulation. The high velocity regions are colored red with high opacity, while low velocity regions are colored green with low opacity. The opaque vortex structure around the hurricane center is surrounded by a large semi-transparent region. To better navigate around this data, we define a configuration space slightly larger than its volume size so that the roadmap can comprehensively cover the space and is able to produce reasonable initial paths. As illustrated in Fig. 2, our roadmap precisely extract the tunnel structure of the central vortex. As a result, we can create a fly-through camera motion by simply placing POIs at the both ends. The opacity force in this dataset plays an important role to avoid the view passing through semi-transparent cloud which immediately decreases its visibility.

The experimental results show that our technique is flexible to handle a diverse range of visualization scenarios. User interaction is an important part in our method in meeting different visualization needs. For the same set of POIs, our system generates several initial paths based on different evaluation functions. Users can choose a path that best fits their needs. The path is then further refined with the multi-criteria dynamic system. In this step, users are allowed to interactively tune the camera path by using several sliders to change the weight for each criteria. The outcome of the new setting is instantly updated in the visualization view. The visualized forces which indicate the effects of the criteria are also helpful for guiding users in tuning process.

In our implementation, the six controllable criteria are distance, opacity, occlusion, visibility, altitude, and repulsion. The distance criterion indicates desired length of the resulting path. For datasets with many semi-transparent regions, the opacity criterion plays an important role. Changing the weight associated with the opacity criterion allows users to transition between two extremes: a shorter path with higher opacities or a longer path with lower opacities along the way. For complex volume data with highly occluded regions, it is crucial to find a path with unobstructed views, i.e. not occluded by nearby objects along the path, to reveal more contextual information. For the vortex field data as shown in Fig. 9, we set the occlusion criterion to high priority to avoid occluded areas even when the opacity is low.

6.2 Comparisons

We compare our camera motion planning algorithm with two other methods: the cubic interpolation that is frequently used in keyframing animation, and the collision-free path which is based on a roadmap. The cubic interpolation takes each interesting view as a control point and smoothly interpolates intermediate frames in-between. The collision-free method finds the shortest path that avoids any obstacles based on a roadmap and is widely used in many 3D path planning systems. The path is smoothed using spline interpolation once the shortest path is found. We choose these two methods for comparison because they are most commonly used in camera motion planning.

We compare our method with the cubic interpolation method in Fig. 9. Without considering the contextual information, it is obvious that the cubic interpolated path can easily penetrate the opaque vortices, whereas our path shown in the left image bypasses the opaque regions. Fig. 11 shows another comparison with the cubic interpolation method using the rat dataset. In this data, the MRI of a rat neck is visualized with PET segmented carotid arteries to illustrate the atherosclerotic lesions in one of the arteries. Creating animations for such multimodal medical data can provide cardiologists a form of virtual endoscopy for further inspections. But in traditional keyframing animation, at least eight camera keyframes are needed to be carefully placed inside the injured artery to create a smooth navigation path. By setting the configuration to the artery, our method is able to accurately extract its hollow space and generate a fly-through camera motion.

Fig. 10 illustrates a comparison of our camera path and the collision-free shortest path in the brain tumor dataset. In order to have a close look of the tumor, the camera has to pass through many semi-transparent layers and opaque blood vessels so as to fly the view into the volume. The collision-free path here is found via the roadmap based on the medial axis transformation and thus mainly lies at the middle of the free space. However, the cluttered structures in the tumor dataset contain little space and allow only limited views of the surrounding areas. In such a case, the visibility and repulsion forces in our approach have the advantage of a better use of the space. By pushing the camera slightly away from POIs, it creates a wider field of view around the interesting regions and provides richer contextual information along the camera motion. Furthermore, the medial axis transformation is very sensitive to small structures of the opaque regions. Thus, without further refinement, the collision-free path can result in a longer route in noisy areas.

6.3 User Feedback

We demonstrated the animation support to our collaborator, a radiologist in UC Davis School of Medicine. "The way users in radiology actually look at the things is first to get oriented," he said, when we show a short animation presenting eroded bones in the corpus of a wrist CT dataset, "I like this of not going through (the bones). At least from what I saw, I have the anatomical contexts, as to how I am moving and which direction I am moving to." He also pointed out that the support would be especially effective in illustrating complex spatial structures in tumor datasets. "Tumors have shapes that are dependent on how cancerous the tumor is. So the shape is really critical to know..."
whether it is just a benign tumor or it’s a highly perfused tumor. So techniques like this actually will be helpful in orienting yourself to where the things are.”

When we explained how the camera path can be interactively adjusted such that more or less contextual information can be put in the field of view, he agreed with the flexibility but commented that more contexts in the view are not necessarily always better. “And I think this probably is object dependent because most of the time people who will be looking at this will have some knowledge of the anatomy. So I just need at least some contexts whether I am moving left or right and which of bones I am moving from one to the next.” The radiologist also mentioned that in some cases, for example, when looking at the knee joint, a broader view is preferred to examine the surface topography.

7 DISCUSSIONS

View selection for volume features is a challenging problem, especially for scientific data in which feature identification, in most cases, involves certain domain knowledge. In this work, our objective is to generate effective camera motion for presenting a sequence of user-selected POIs. We focus on the framework for multi-criteria camera motion optimization and develop a dynamic system allowing easy fine-tuning based on users’ intention. Although we aim to address the problem for general volume visualization, for a small volume object that an orbit camera is sufficient for showing most of its interesting features, our method can only bring little benefits over other view selection methods. Nonetheless, we show that in more complex cases when camera motion that flies through volume data is necessary, our method provides a great help to ease the animation production process.

As already mentioned in Sec. 4.1, the complexity for calculating $\mathcal{MA}$ and building a sphere tree is $O(n)$, where $n$ is the number of sample points in data. Constructing a roadmap requires $O(v^2)$ time, where $v$ is the number of nodes in the sphere tree representation and is usually much smaller than the data points. With the use of the heuristic function, searching for an initial path can be done in $O(v)$ time. And the path refinement involves iterative calculation but can be easily hardware-accelerated to achieve interactive responses.

Our experiments are carried out on a test system with Intel Core i7-3930K CPU, 16 GB RAM, and NVIDIA GeForce GTX 690. For the tumor dataset ($512 \times 512 \times 176$), it takes 10833 ms to calculate $\mathcal{MA}$, and 1872 ms to build a sphere tree with an $\mathcal{MA}$ threshold $\varepsilon = -200$. It results in 679 nodes, and with $K=2.2$ and $D=2.2$, computing the roadmap takes 6 ms. Depending on the number of POIs and the distances between each pair of POIs, searching for a coarse path takes less than 10 ms in most cases. The damping force has strong impact on the convergence time of the dynamic system. With a proper damping force, the system can reach a steady state within a few seconds after changes in criterion weights are made.

Limitations

Since we build our roadmap by tracing the medial axis from the sparsest position in the configuration space, the resulting sphere-tree is always a connected graph in the continuous free space. For dense data such as medical anatomical images, if there are spatially separated spaces in the volume, feature points in different spaces cannot reach to one another, and the system would fail to find an initial path for them. Or if there is not enough room in the free space inside the volume, it is harder to generate smooth camera motion paths to traverse through feature points that reside in the volume. In such cases, decreasing the opacity of contextual regions or increasing the opacity threshold of DT can create more free space for camera paths. Other techniques such as cut-away views or advanced filtering can also be introduced to reveal the occluded features but are out of the scope of this paper.

The medial axis transform is very sensitive for small perturbations in the volume data as already mentioned. So for some very noisy volume datasets, the medial axis sometimes does not lead to very good roadmaps. Noise in the volume can also result in undesired shakes in the refined motion path if the opacity force suddenly becomes too strong at a point. Slightly adjusting the transfer function or applying a Gaussian filter to suppress noise and make the visualization less clutter can relieve the situation.

Critical points in the force fields attract or repulse surrounding sample nodes in all directions. Repulsive critical points are local maxima such as the most opaque regions in the space, and nearby nodes are always pushed away from the points to reach lower energy states. However, attractive critical points could be a problem because if the attractive forces become too strong, the nearby nodes tends to converge on the points and distort the surrounding paths. As a result, it is crucial for the system to have balanced net forces to reach a steady state. The spring force thus plays an important role in the force balancing.

The roadmap is generated based on the optical properties of volume data and needs a recompute when the visual representation changes such as transfer functions, time steps, variables, or other visual parameters. Thus, our method cannot directly generate camera motion for connecting features in different visual states. To create complicated animations, generating multiple roadmaps and force fields for different portions of the camera motion path is a feasible solution but will increase the space complexity. For example, to move the camera between time-varying features requires sampling multiple force fields in different time steps along the path. Nonetheless, our method can easily be integrated into an existing keyframe-based visualization system to produce an improved and enhanced animation production tool.

8 CONCLUSION

We have introduced a solution to address the problem of the generation of desirable camera motion for producing volume data animations. There is the misconception that most visualization tools adequately support animation production. We argue that several aspects of the animation task can use some improvement. Our ultimate goal is to make animation production an effortless task for domain scientists. Our design is extensible to meet different application needs by adopting different path criteria such as preferred length, opacity tolerance, and cinematographic principles. It is also straightforward to incorporate our design into visualization systems that support keyframe-based animation. Here, we have demonstrated the difference our current design can make.

In future work, we plan to address the limitations of our current approach. We intend to explore additional criteria such as aperture and temporal coherence for supporting demanding volume visualization applications, including complex flow with embedded geometric objects and dynamic features in time-varying or multi-field data.

ACKNOWLEDGMENTS

This research has been sponsored in part by the National Science Foundation through grants OCI-0905008, OCI-0850566, OCI-0749227, IIS-1255237, and CCF-0811422, and also the Department of Energy through grants DEFC02-06ER25777, DE-CS0005334, and DE-FC02-12ER26072.
REFERENCES


