Enclosure Sphere Based Cell Visibility for Virtual Endoscopy

Jianfei LIU * Xiaopeng ZHANG #

Sino-French Lab LIAMA - National Lab NLPR, Institution of Automation, Chinese Academy of Sciences

ABSTRACT

Virtual Endoscopy is an interactive exploration inside human organs to detect polyps by combining medical imaging and computer graphics technologies. In order to realize rendering acceleration, we present a novel two-steps visibility algorithm, Enclosure Sphere Based Cell Visibility (ESBCV), which performs visibility computation between cells assisted by Z-buffer in pre-processing and eye-to-cell visibility through a simple numerical calculation of the intersection of a circle with the rendered image in navigation. Experimental results demonstrated virtual navigation with high image quality and interactive rendering speed.

CR Categories and Subject Descriptors: I.3.7 Three-Dimension Graphics and Realism

Keywords: Virtual Endoscopy, Visibility Analysis, Cell and Portal

1 INTRODUCTION

Virtual endoscopy [1] has been widely employed to examine the interior structures of human organs. Compared with conventional endoscopy, it is noninvasive, cost-effective, free of risks, and capable of viewing some locations and directions in which conventional endoscopy is limited.

Several papers on virtual endoscopy have focused on the performance of interactive rendering. Hong [2] applied an image-based portal technique to virtual colonscopy. The colon is partitioned into cells in preprocessing, and invisible cells are culled using the visibility determination assisted with Z-buffer at run-time. Extending from the single-tube-structure of organs to the tree-structure, Bartz [2] used an octree decomposition of the volume data and a hardware supported visibility algorithm to identify the visible voxels during rendering. However, data management of cells needs to be improved, in addition to the visibility computation efficiency in navigation.

A two-steps visibility algorithm, Enclosure Sphere Based Cell Visibility (ESBCV), is proposed to remove hidden polygons through conservative *cell-to-cell* visibility and *eye-to-cell* visibility with the help of Z-buffer and view frustum. In pre-processing, the model is decomposed into cells and portals along its centerline similar as [2]. Then we project enclosure sphere of each portal into the Z-buffer and compute *cell-to-cell* visibility by checking their intersections. During dynamic rendering, the *cell-to-cell* visibility can be further reduced into *eye-to-cell* visibility assisted with current view frustum. The original contribution of this paper is to introduce a novel visibility algorithm, ESBCV, to manage polygon data with visibility tree in preprocessing and decrease visibility computation cost at run-time.

2 ENCLOSURE SPHERE BASED CELL VISIBILITY

Enclosure Sphere Based Cell Visibility (ESBCV) contains three main steps: mesh decomposition, *cell-to-cell* visibility, and *eye-to-cell* visibility.

2.1 Mesh Decomposition

The dataset we deal with are BREP mesh along with Distance Contained Centerline acquired through [4]. Then a centerline-based scheme is employed to decompose the mesh into cells and portals to assist the visibility determination. During mesh decomposition, the point with high centerline curvature and minimum distance from the boundary in the centerline has the priority to be selected as the partition position. In figure 1(a), the circles shown in cyan represent different portals, and the mesh between any adjacent portals forms a cell.

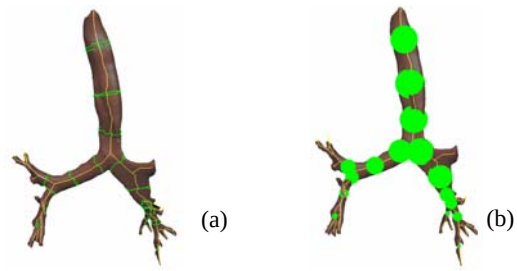

Figure 1: Centerline based mesh decomposition.

2.2 Cell-to-cell Visibility

In order to simplify the visibility computation and avoid the influence of anisotropy, the minimum enclosure sphere, shown with cyan color in figure 1(b), is applied to replace a portal, which is a nonplanar polygon. Hence, the visibility between portals can be simplified to the problem that how many spheres a line is able to stab using a depth-first search strategy.

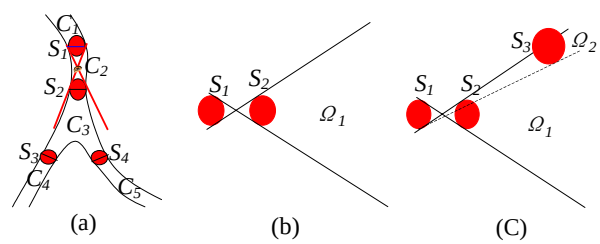

Figure 2: *Cell-to-cell* visibility analysis.

Let us take the tree-structure model shown in figure 2(a) for example. Suppose the source cell is C_1, and there is a portal, C_1/C_2, connecting cell C_1 with other cells. Each portal is replaced with a sphere, such as S_1 enclosing the portal C_1/C_2 in red circle. It is obvious that there must exist a sightline L intersecting S_1 emitting from C_1. After all these unobstructed sightlines have intersected with S_2, the area enclosing them is narrowed into a cone Ω_1 as shown in figure 2(b). When sightlines in Ω_1 encounter S_3, the visible area between C_1 and C_4 can be further reduced to $\Omega_2 = \Omega_1 \cap S_3$ shown in figure 2(c). If $\Omega_2 = \phi$, C_1 and C_4 are invisible; otherwise they are visible. Following the above procedure, the visibility between any two cells can be determined in preprocessing.

* e-mail: jfliu@liama.ia.ac.cn
# e-mail: xpzhang@liama.ia.ac.cn

Unfortunately, previous visibility analysis is a complex linear system and hard to solve through numerical calculation. We tackle it through the process of computer image synthesis assisted with Z-buffer. Using the previous model in figure 2 as the case, the collection of sightlines traversing S_1 will be restricted to a cone Ω_1, which can be chosen as the view frustum shown in figure 2(a) with two red intersection lines. Then we project S_2 onto the screen and generate a bounding box of the resulting vertices with the help of Z-buffer. This bounding box, called the *cull rectangle*, represents a conservative bound of the collection of sightlines stabbing S_2. As each sequential sphere is traversed, the aggregate *culling rectangle* can be shrunk in near-to-far order until the rectangle has fully degenerated. Figure 3 shows the results of *cell-to-cell* visibility along the forward direction, where the source cells are shown in cyan.

The visibility tree [5] is employed to store *cell-to-cell* visibility results in order to effectively swap the visible data in and out of memory. Each node or vertex of the binary tree corresponds to a cell visible from the source cell. Each edge of the binary tree corresponds to a portal connecting two cells.

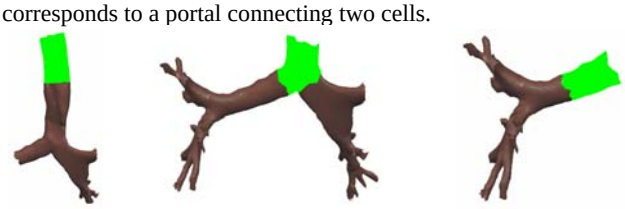

Figure 3: cell-to-cell visibility.

2.3 Eye-to-cell Visibility

During navigation, the cell containing the viewer is first determined, then its visibility tree is further pruned according to *eye-to-cell* visibility, finally the polygon data are delivered into the video card based on the current visibility tree. We perform the visibility tree culling by removing following two kinds of cells.

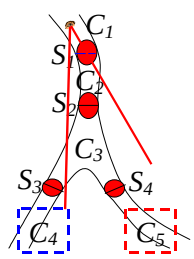

Figure 4: Two kinds of culling cell.

Exterior Cell: We maintain six flags for six planes of the view frustum, indicating whether an enclosure sphere is completely on the interior side of the plane with respect to the view frustum. If a cell's enclosure sphere is completely outside the view frustum, it is called exterior cell. In figure 4, C_4 is an exterior cell shown with blue dashed line.

Hidden Cell: We need to find two points $\overrightarrow{P_{1,2}}$ in the enclosure sphere, whose projected points in the image plane compose the diameter of a circle entirely covering the projection area, an ellipse, of the enclosure sphere. Let $\vec{V}$ denote viewing direction, $\vec{O}$ be the center of the enclosure sphere, R is the radius, and $\vec{N}$ is the normal vector of the image plane. The point locating formula is defined as follows:

$$\overrightarrow{P_{1,2}} = (1-\lambda^2)d\vec{a} \pm \sqrt{1-\lambda^2}\,d\vec{b} + \vec{V} \qquad (1)$$

where $\vec{a} = \overrightarrow{VO} / \|\overrightarrow{VO}\|$, $\vec{b} = (\vec{a} \times \vec{n}) / \|\vec{a} \times \vec{n}\|$, $d = \|\overrightarrow{VO}\|$, and $\lambda = R/d$.

Hence, we can iteratively use equation (1) to find two points in the enclosure spheres from the root cell in the visibility tree, and compare intersection area between different circles in the image plane. If the aggregate circle of previous cells in the visibility tree doesn't intersect with the projection circle of the current cell's enclosure sphere, this cell is called hidden cell. In figure 4, S_4 is obstructed by the intersection of S_1 and S_2, so C_5 is a hidden cell illustrated with red dashed line. Figure 5 shows the results of visible cells and their corresponding rendering images in navigation, where the red line represents current viewing direction.

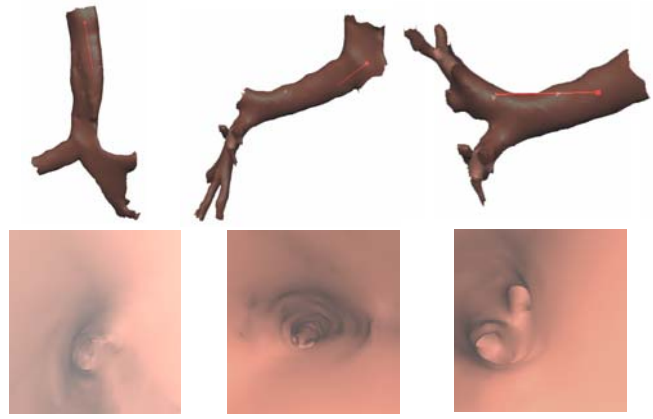

Figure 5: eye-to-cell visibility.

3 Implementation

All experiments have been implemented in Visual C++ and run on a P4 PC with 16M video memories and 512M RAM. Details of experiments are listed in table 1.

Table 1: Experimental details.

Dataset	Complex airway	Simple airway	Synthetic colon data
Triangle number	535025	134895	768000
Cell number	17	4	17
Preprocessing time (s)	62.24	43.35	54.735
Second/frame	0.078	0.094	0.094
Second/frame (Hong)	0.138	0.135	0.132

4 Conclusion

In this paper, we have presented a new two-steps visibility algorithm to compute *cell-to-cell* visibility assisted with Z-buffer as well as efficiently manage the mesh data using the visibility tree. During rending, it can be further pruned by culling exterior cells and hidden cells with the help of view frustum.

Future work will be focused on reducing the invisible cells in the preprocessing in virtue of their border and extending ESBCV algorithm to volume rendering.

Reference

[1] D. Bartz. Virtual Endoscopy in Research and Clinical Practice. In *Computer Graphics Forum*, Vol. 24, No. 1, 2005.

[2] L. Hong, S. Muraki, A. Kaufman, D. Bartz, T. He. Virtual Voyage: Interactive Navigation in the Human Colon. In *Proc. ACM SIGGRAPH'97*, pp.27-31, 1997.

[3] D. Bartz, M. Skalej, D. Welte, W. Straßer, and F. Duffner. Interactive Exploration of Extra- and Intracranial Blood Vessels. In *Proc. IEEE Visualization'99*, pp. 389-394, 1999.

[4] Jianfei Liu, Xiaopeng Zhang, Frédéric Blaise, Distance Contained Centerline for Virtual Endoscopy, In *Proc. ISBI 2004*: pp.261-264, 2004

[5] S. Teller and C. Sequin, "Visibility Preprocessing for Interactive Walkthroughs", In *Proc. ACM SIGGRAPH'91*, pp. 61-69, 1991.