JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 17, 297-312 (2001)

DOI:10.1688/JISE.2001.17.2.7

Practice and Experience: Interactive Rendering of the Colonic Environment on PC-platforms

TONG-YEE LEE, PING-HSIEN LIN AND CHAO-HUNG LIN
Visual System Laboratory
Department of Computer Science and Information Engineering
National Cheng-Kung University
Tainan, Taiwan 701, R.O.C.
E-mail: tonylee@mail.ncku.edu.tw

The virtual colonoscopy technique is a non-invasive technique used to generate high resolution video views of the colon interior structure. The colonic structure consists of a great number of triangle meshes that challenge rendering performance. In contrast to previous work, the proposed approaches allow interactive colonic surface rendering on low-cost PC systems. Our system takes advantage of the following concepts: PVS, LOD and image caching. Initially, the colon structure is partitioned into many cells based on curvature of the centerline. During rendering, only triangles contained in PVS are considered for rendering. To further improve rendering performance, each cell is represented by multi-resolutions of triangular mesh or cached images. We present an adaptive approach used to automatically switch LOD/image caching and yield interactive rendering performance with acceptable realism on low-cost PC platforms.

Keywords: virtual colonoscopy, interactive rendering, potentially visible set, level-of-detail (LOD), image caching

1. INTRODUCTION

Virtual environments are computer simulations of environments that either exist in the real world or are brought to existence through computer simulation. In medical applications, the virtual colonoscopy environment [1-3] gives physicians a way to enter, observe and interact with the colon structure in an artificial environment. In virtual colonoscopy, hundreds to millions of triangles are required to represent with good fidelity the colonic structure for detecting possible polyps. In such virtual environments, interactive rendering performance is very crucial. Achieving a sufficiently interactive frame rate is the major goal for creating a visually convincing presentation of the virtual colonoscopy environment for physicians. In this study, we experimentally evaluated different rendering techniques. We report our practice and experience in achieving interactive rendering rates in a complex colonic environment on low cost PC-platforms.

In the virtual colonscopy system, the triangular surface of the colon structure is first segmented based on the volume data and can be surface-extracted using a marching cube algorithm. In our experiments, one set of colonic volume data 512×512×318 in size was used. After surface extraction, there were approximate 500K triangles used to represent

Received March 25, 1999; revised August 30, 1999; accepted October 8, 1999.
Communicated by Yung-Nien Sun.

the entire colon structure. When rendering this number of triangles on a low-end PCs, it is hard to achieve interactive performance. However, since the colonic environment is a tube-like structure, at any instant, only a subset of the colon is visible to the observer. Therefore, there is great potential for exploiting this observation and the characteristics of the colonic structure to cull the number of triangles. In the past, several approaches have been proposed to speed up rendering of the colon structure [2, 4, 5]. In [2], Arie's approach [2] computes PVS (Potentially Visible Set) on the fly during rendering and uses a hardware-assisted visibility algorithm called the aggregate cull rectangle (ACR) algorithm. ACR efficiently culls unseen cells in near-to-far order. Lorensen [4] et al. exploited a triangle decimation method to globally eliminate the flat portion of the colonic surface prior to rendering. In this manner, the number of triangles is significantly reduced, however, this may lead to a loss of detail contained in the original surface. On the other hand, Yagel et al. [5] exploited a parallel technique to expedite the process of volume rendering. So far, as mentioned above, most work has been done on expensive high-end workstations or parallel architectures. In contrast, we report a practical solution based on virtual colonoscopy technology for use on low-end PC platforms.

The remainder of this paper is organized as follows. Section 2 overviews relevant work on rendering large geometric databases. Section 3 reports our practical solutions for accomplishing interactive rendering on PC-platforms. Experimental results and discussion are presented in Section 4. Finally, concluding remarks are given in Section 5.

2. INTERACTIVE RENDERING TECHNIQUES

To render complex virtual environments like the colonic structure at sufficient frame-rates, interactive rendering techniques can be exploited. This section gives an overview of the previous work in real-time computer graphics relevant to our work. PVS (Potentially Visible Set) was first proposed in [6] for application to the densely occluded environment. The key is to compute an appropriate partition of the environment. During rendering, only triangles in PVS are taken into consideration. This approach obtains a conservative estimate of the actual visible parts for a given region. Another approach to improving rendering performance is to replace a complicated mesh with a set of level-of-detail (LOD) approximations. Hoppe et al. utilized an energy function to obtain a mesh reduction framework [7]. Similarly, Hoppe's recent work [8] presented a new framework for selectively refining an arbitrary progressive mesh according to the viewing position. Schaufler [9] et al. and Shade [10] et al. independently developed similar techniques termed dynamic impostors and image caching, respectively. Object images are dynamically generated and re-used in the following frames as long as possible. The object images are mapped onto transparent polygons that are positioned and oriented properly with respect to the point of view in order to closely mimic the appearance of the objects [9]. In [11], we presented PVS with LOD technique to improve rendering speed. In this paper, our practical solutions exploit the concept of PVS in conjunction with LOD and the use of image caching techniques to improve rendering performance. We use a heuristics based on distance and viewing disparity to determine the transitions of LODs or image caching.

3. INTERACTIVE RENDERING OF COLONIC STRUCTURE

3.1 PVS Determination

In our previous work [3, 11], we described a virtual colonscopy system. In that work, we concentrated on the preprocessing tasks in the proposed system, including segmentation, 3-D thinning and tracking, used to find the flythrough path. The whole colon is divided into several cells based on curvature of the centerline. In this paper, we concentrate on the issue of interactive rendering of the colonic structure. For more details about other preprocessing steps, please refer to [3, 11]. Here, we assume that partitioning of the colon has been accomplished, and that now, we can perform visible set pre-computation. Fig. 1 illustrates our method used to compute PVS. For each region, there are two cross sections (front and rear doors) shared with two neighboring regions (except for the first and last regions). Take region *F* as an example. Cells *E* and *G* are immediate neighbors of *F*, so *E* and *G* will be included in *F*'s PVS. For other non-immediate neighbors like *H*, we must determine if any point of *H* can be reached by a *sight-line* that originates inside cell *F*. This visibility can be determined as follows. We cast rays from those points on the front door of *F* to the points on the rear door of *H*. Considering two rays, say ray_i and ray_j, ray_i can reach cell *H*, but ray_j is blocked (since it hits the boundary before hitting *H*'s rear door). Therefore, cell *H* is visible to *F* and will be included in *F*'s PVS. However, if no visible ray (such as ray_i) is found, then cell *H* will not be included. If *H* is not visible to *F*, we will terminate the search of *F*'s PVS because cell *H* (invisible cell) will block the remaining cells. Otherwise, the search of PVS will continue until the end of the colon is reached or an invisible cell is found. The visibility computation for PVS is obtained by using sight-lines between two doors. Each door consists of a finite number of voxels on each door plane. We shoot each sight-line from one voxel in one door to the other voxel on the corresponding door. This is accomplished using the standard 3-D line drawing algorithm, called 3DDA. It takes about 10 seconds to compute the PVS between two cells. Due to the finite sampling involved in the sight-line process, there might be a problem with accuracy of PVS. However, this most often occurs at the ending portion of PVS. In our experience, it does not cause too much bad visual effect to influence surgical examination. One solution for decreasing the possibility of error is to increase the number of sight-lines per voxel. In our implementation, a double-linked list is maintained for fetching a cell's PVS. As the camera moves along the colon cell by cell, we also travel this list node by node and access the cells for the current PVS (as shown in Fig. 2). As an observer moves forward and backward along the colon, the pointer to the current PVS is updated forward and backward.

3.2 Adaptive Display of the Colonic Environment

To adaptively display colonic cells, we first create multi-resolution representations for each cell. Switching between different resolutions is primarily determined by the order (*near-to-far*) existing in the current PVS. In this study, we adopted Hoppe's progressive mesh approach [7] to create the LODs of each cell. Fig. 3 shows some results of mesh simplification obtained in our study. In this example, there are four versions available for one cell, and the number of triangles is approximately reduced by a factor of two

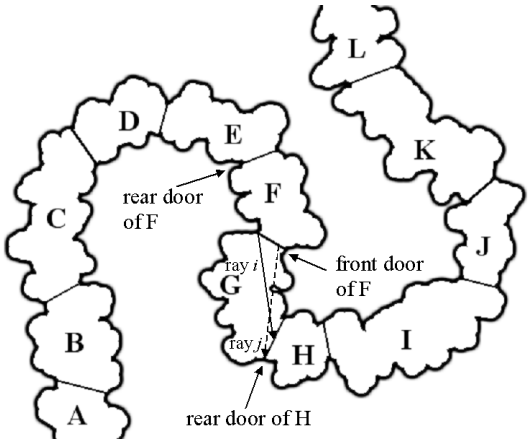


Fig. 1. Computation of PVS for each cell.

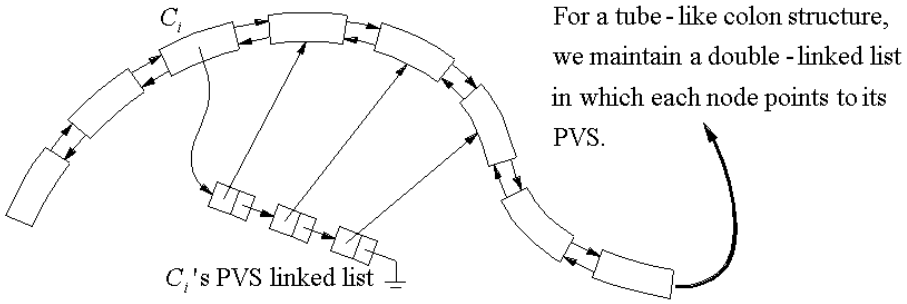


Fig. 2. The double-linked list used to maintain PVS.

between two consecutive levels (i.e., the maximum and minimum are 11470 and 997 triangles, respectively). However, the difference in appearance between them is small. Note that in the method [7], there is only a copy of original mesh accompanying with split-merging relation links among different versions. Transitions between different versions are computed during rendering. In our implementation, we adopt the method in [7] to simplify the mesh only, but we compute and store six versions for each cell in the preprocessing stage. During rendering, we only need to determine how to switch different versions instead of computing new versions as in [7]. Additionally, since different versions of mesh are created for two adjacent cells, there might be cracks between cells. To solve this problem, we can prevent triangular meshes on the boundary from being simplified by simply adjusting their weights (*i.e.*, very large values) in the energy function.

During rendering, we propose an adaptive selection of LOD algorithm described as follows.

We make the following assumptions (see Fig. 4):

- Each cell has *m* versions of levels-of-detail, $\ell_0, \ell_1, ... \ell_{m-1}$, where ℓ_0 is the original mesh and ℓ_{m-1} is the crudest version.
- The current camera is located at cell C_i, and its viewing direction and 3D position are denoted as V_{view} and *eye*, respectively.
- There are *n* cells in Ci 's PVS denoted as $\{p_1, p_2, \ldots, p_n\}$, where p_i is in near-to-far

order from C_i.

- The center and normal vector of the partition plane (rear door) for p_i are denoted as O_i and N_i, respectively.

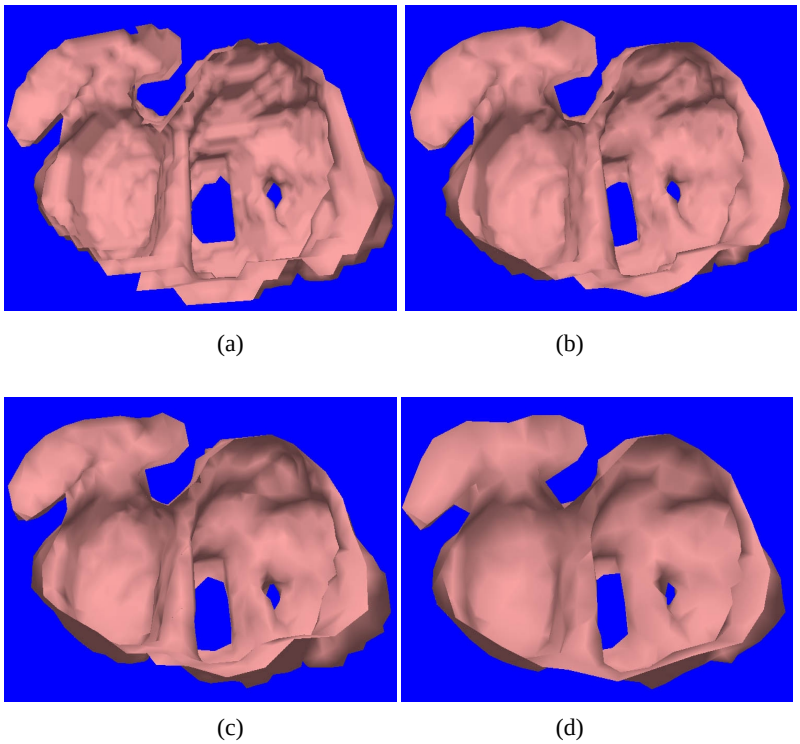


Fig. 3. Multi-resolution representations of a colonic cell ((a)-(d): high to low resolution).

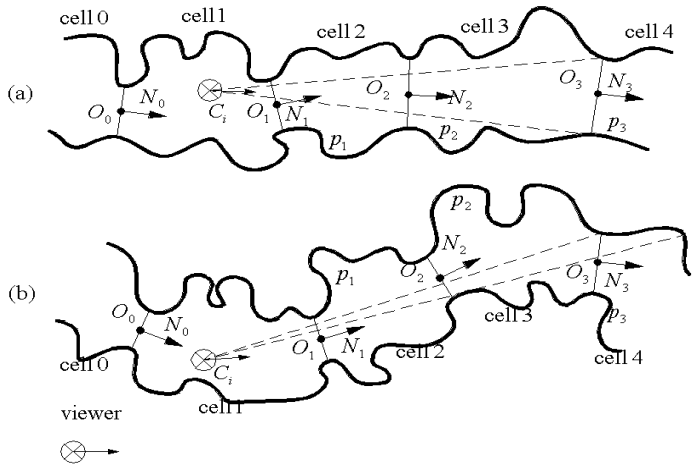


Fig. 4. The adaptive LOD selection scheme.

To automatically determine the transitions of LOD, we first normalize LOD selection based on several representative views within each cell in the preprocessing stage. Then, during rendering, we use this pre-computed selection mapping to determine the version of cell approximations. For each cell C_i, the ***normalized_selection()*** procedure computes three sel_{ji}s for three viewing points, the starting, middle and ending points, respectively, located on the centerline of cell C_i. In this manner, we compute sel_{ji}s for all cells and use their values as the approximate distribution for all possible views inside the colon. In the current algorithm, we only use three views to approximate the distribution of possible views inside the colon. It might not always be correct. One possible modification is to locate high curvature points along the centerline of the cell and to then use these points as the representative views. Next, we need to determine how to classify this distribution into *m* levels. For the purpose of classification, the *k-means* method [11] is exploited to find the range of sel_{ji} for each level of LOD. *K-means* is a well-known method used to determine *k* clusters among distribution. In our approach, each cluster represents a level of LOD. We can easily check the value of sel_{ji} to find out which cluster it belongs to. The advantage of the *k-means* method is that it groups most related distribution in one group. In this manner, we attempt to avoid jerk transitions. Therefore, during rendering, we compute a sel_{camera} from the current camera location to each possibly visible cell of C_i, and then use sel_{camera} to determine its version of LODs according to level classification in the preprocessing stage. Later, we will present detailed experimental results to illustrate use of our method.

procedure ***normalized_selection*** ( )
begin
  For each colonic cell do
  {
    For j = 1 to 3 do
    // three eye positions, starting, middle and ending points, respectively .
    {
    // For each cell, each PVS consists of *n* cells, and the center and normal vector of the partition plane (rear door) are denoted as $O_1,\ldots O_n$ and $N_1,\ldots N_n$, respectively.
      $dist = \|eye_j - O_1\|$; /*** currently, the viewer is located in cell $_k$)
      $\theta = \cos^{-1}(V_{view} \bullet N_1)$;
      For i= 1to *n* do /*** if there are *n* cells in PVS for the current cell $_k$
      // for all PVS cells from the current cell
      {
        $sel_{ji} = w_1 \times dist + w_2 \times \theta$; // w_1 and w_2 are user-specified weights
        $dist = dist + \|O_i - O_{i+1}\|$; // distance heuristics
        $\theta = \theta + \cos^{-1}(N_i \bullet N_{i+1})$; // viewing direction heuristics
      }
    }
  }
    Use the ***k-means*** method to classify all sel_{ji} s into *m* groups;
end

In the above algorithm, the parameters *dist* and θ are two key factors in determining the selection of levels-of-detail for each cell in the current PVS. In the preprocessing stage, we compute weights at three places (starting, middle and ending points) as a rough estimation used to determine transitions of LODs. When the above algorithm is used, the farther cell selects a coarser one. The parameter θ indicates the potential of being blocked by the nearer cell. Therefore, a larger θ will tend to select a coarser one, too. As shown in Fig. 4, for p_3, case (*a*) will tend to select a finer one than case (*b*).

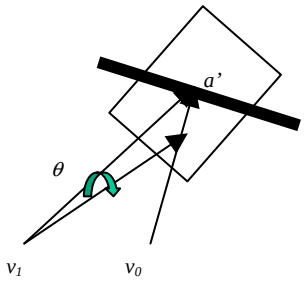


Fig. 5. The angular discrepancy, θ [10]. The cached image was computed based on v_0. Let point *a* be a point inside the cell and its correspondence on the cached image is *a'*.

Image caching is the concept of reusing a previously rendered image for the current or the following frames. In this study, dynamically generated cached images are integrated into our adaptive LOD selection scheme. Since the entire colon is divided into cells (i.e., continuous and order is predetermined) and the observer walks inside it, there is no need for a special data structure like the BSP tree [10] to determine the composition order (far-to-near). Therefore, the only task we need to perform is to classify each cell (in PVS) as being a candidate for an image caching or not, and whether a new caching must be generated or an existing one can be re-used. Similar to [10], we set up a parameter θ (see Figure 5), an error metric that measures the maximum angular discrepancy between a point inside a cell and the point that represents it in the cached image, to determine the life span of an image caching.

As mentioned above (Fig. 4), in the colonic environment, the parameters *dist* and θ are two key factors in determining the selection of levels-of-detail for each cell in the current PVS. Similarly, we adaptively determine θ for the life span of an image caching per cell using these two parameters. Principally, larger *dist* and θ will tolerate a larger θ error (kindly similar to the concept of LOD). Additionally, the projected area, *A*, of a cached image is also an interesting factor. Fig. 6 shows that a smaller *A* implies that it is located at the far end of the PVS. In this situation, the accuracy of its representation might be less significant. Therefore, we take it into account in determining θ for each cell of PVS. The method for selecting θ is exactly the same as the procedure ***normalized_selection*** () except that we use $sel_{ji} = w_1 \times dist + w_2 \times \theta + w_3 \times A$. By combining LODs and image caching, we hope to achieve the following: The nearer cells will be rendered using finer representations. On the other hand, the farther cells will be rendered using coarser representations, and we hope to use their cached images for several frames, thus saving rendering time. Basically, the hybrid methods mentioned above have the following arrangement:

- **Case 1**: The cell that is currently being visited by the viewer must be represented by the finest mesh version.
- **Case 2:** The cells located in the end portion of the PVS are represented by either the LOD or cached images. In particular, when a new cell becomes visible, it is first represented by the LOD. However, in the following frames, we use the cached image instead of the LOD as possibly as we could. Since these cells are located in the end portion, it is less significant and is always partially blocked by the other cells; therefore, this arrangement does very well. If the cached image is used, we represent each cell with a cached image and paint the cached images in a back-to-front manner. Note that if the current cached image for the cell can not satisfy angular discrepancy, we use the LOD instead.
- **Case 3:** The cells near the first cell (***i.e.***, the one currently being visited by the viewer) are always represented by the LOD.
- The determination of cases 2 and 3 is based on sel_{ji}.

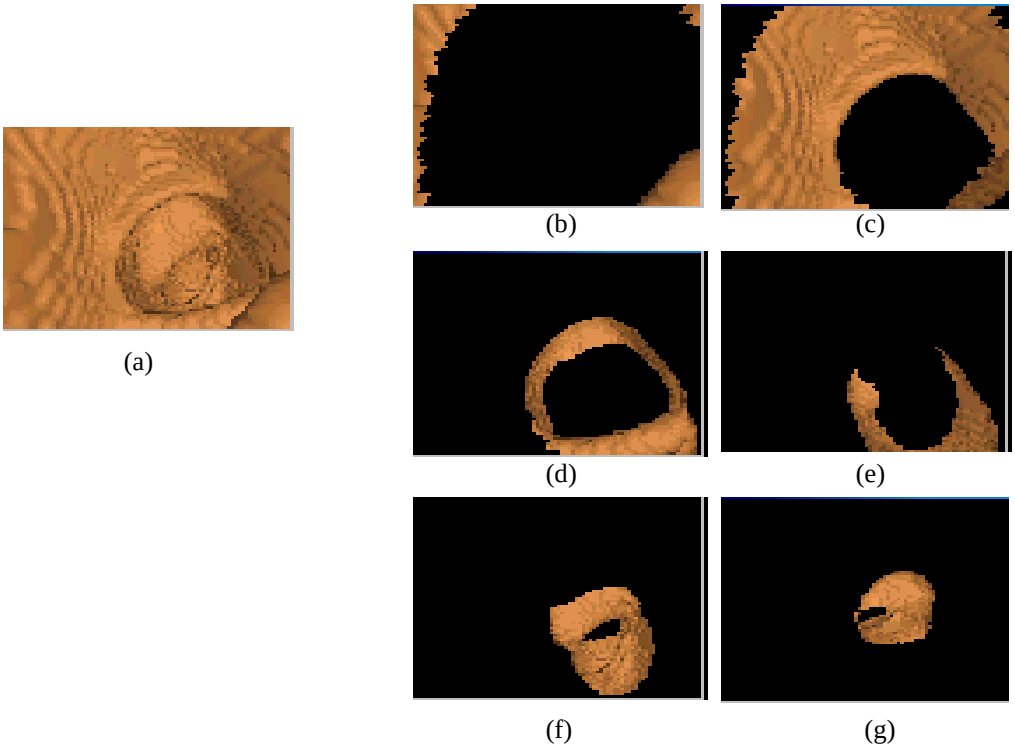


Fig. 6. The left image (a) is the final rendering result, and the other images ((b)~(g)) are the partial images for each cell from near-to-far in the PVS.

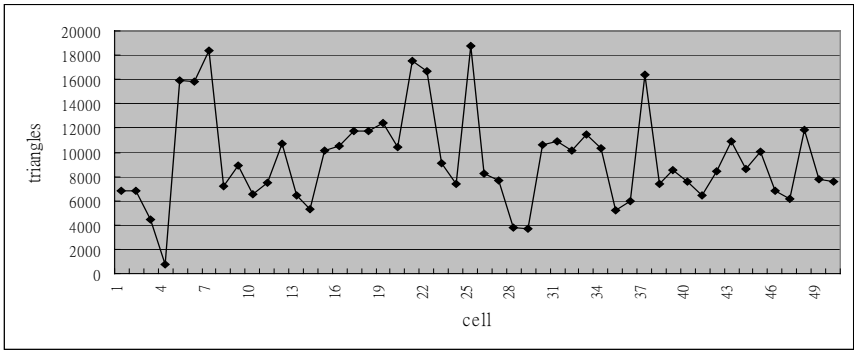
Image-caching for each colonic cell is implemented as follows. First, we project the eight corners of the bounding box of each cell. Then, we find the screen space bounding rectangle of eight points (***i.e.***, corners). Next, we project the center point of the bounding box and replace the projected z of the four corner points found with the projected z of the center point. Finally, we inversely project these four points into 3D space. If we use the

standard OpenGL graphics library, we can easily achieve this through Unproject() API. After the above steps are completed, we define a 3D quadrilateral (***i.e.***, a billlboard) that passes through the center of the bounding box, onto which we can texture map an image of triangles contained in this cell.

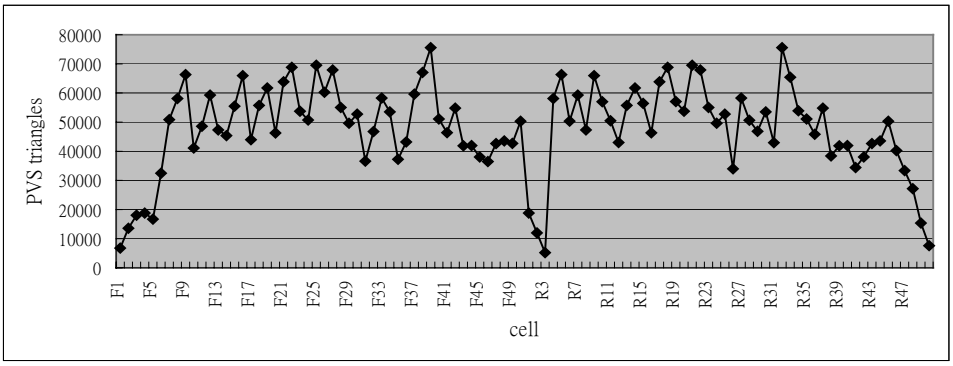
4. EXPERIMENTAL RESULTS AND DISCUSSION

We divide the entire colon structure into 50 cells. Plot 1 shows the number of triangles contained in each cell, and Plot 2 shows the number of triangles in the PVS of each cell. On average, there are 47944 (48 K) triangles in the PVS per cell. In case that we do not exploit PVS strategy, there are about 470K triangles required to pass the rendering pipeline. Therefore, the PVS can significantly reduce the number of triangles.

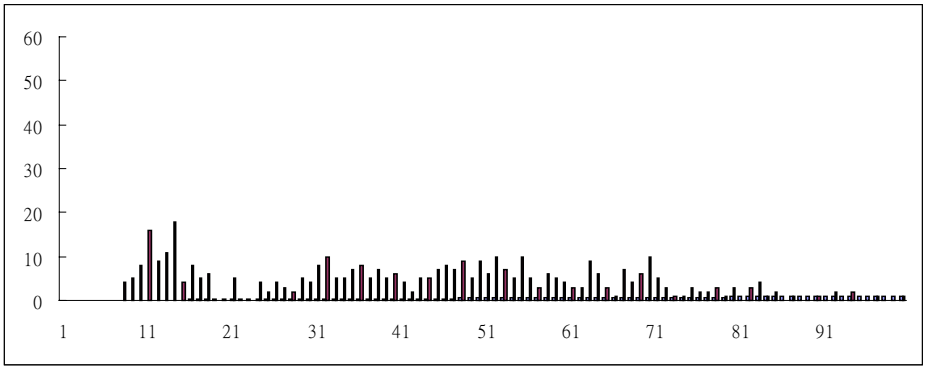
Next, we will show that the PVS strategy can be further improved in conjunction with the proposed adaptive-selection-of-LOD scheme. In this experiment, we used 6 levels-of-detail for each cell of the colon structures, and the weights for *w1* and *w2* were 1 and 2, respectively. Plot 3 shows the distribution of sel_{ji} for the whole colon pre-computed under this configuration, and Plot 4 shows six groups after *k-means* classification. Note that all the computed sel_{ji} were scaled to a range of integers (0,110). In our plots, the X-axis represents the scaled sel_{ji}, and the Y-axis the number of views with the same scaled sel_{ji} integer. Each classification represents a range of *sel* for a specific level of LOD. Plot 5 shows the average number of rendered triangles per frame during automatic navigation (a total of 6001 frames). On average, it rendered about 24K triangles. This number is a small portion of the colonic surface (470K triangles). On a PC-based platform, it is easy to achieve an interactive rendering rate of 24 K triangles. The plot shows that a larger or longer PVS may require that more triangles be rendered than a smaller or shorter one. We have achieved adequate rendering speed for an image resolution of 512×512 on a dual-CPU Pentium-2 PC server system. We should point out that [4] also used triangle reduction to improve rendering. However, they did not adaptively select appropriate version of colon cells based on distance and view direction.



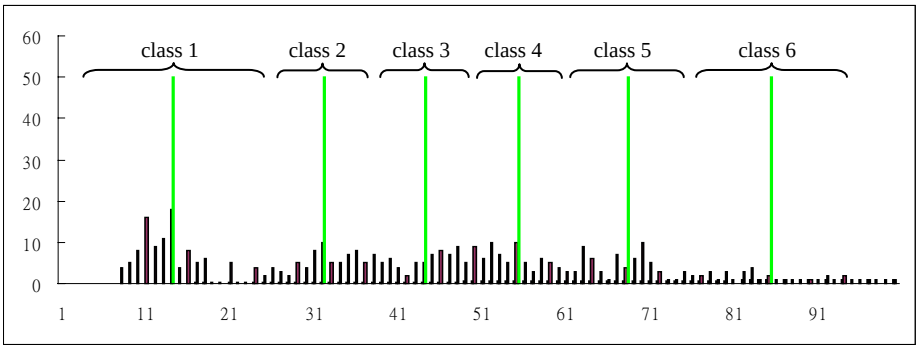
Plot 1. The number of triangles contained in each cell.



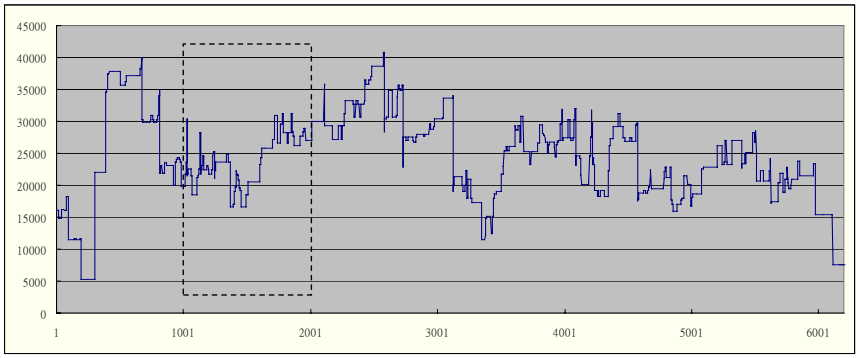
Plot 2. The number of triangles which are visible to each cell (F: forward viewing, R: backward viewing).



Plot 3. The estimated distribution of *sel* for the various possible views inside the colon. Note that the X-axis represents the scaled sel_{ji} integer range (0,110), and the Y-axis the number of views with the same scaled sel_{ji} integer.

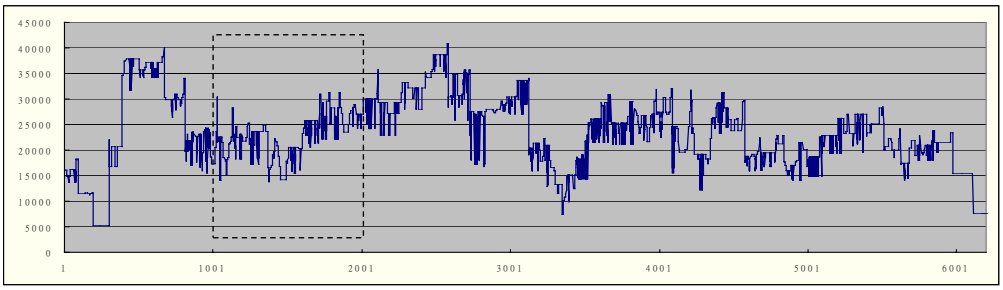


Plot 4. The classification of *sel* using the *k*-means method.



Plot 5. The number of triangles rendered using adaptive selection of LOD. On average, 24K triangles per frame were required during 6001 frames.

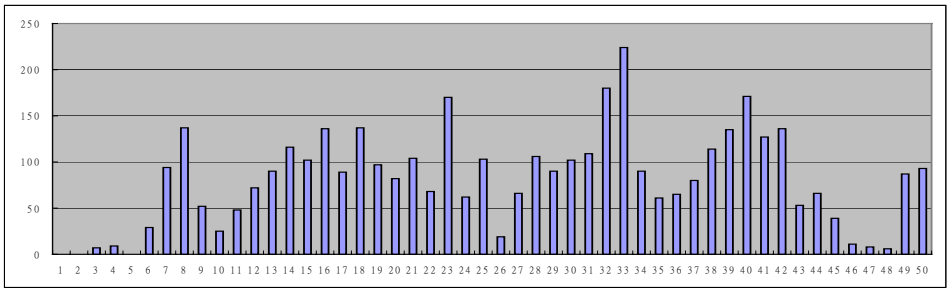
Plot 6 shows the results for adaptive LOD in conjunction with the image caching scheme. To prevent perceptible visual artifacts, our implementation used a small error threshold: θ selected by about 2 pixels on the image plane. This combination achieved about 20K triangles per frame. It did not offer significantly improvement over adaptive LOD. There are some reasons for this. Each PVS might contain several potentially visible cells, ranging from 4 to 7 in our case. For outdoors scenes [10], it can achieve greater improvement. However, we must examine further some frame intervals shown in Plots 5 and 6. Plots 8 and 9 shows their comparison between 1001 and 2001 frames. Adaptive LOD with image caching performed better. Plot 7 shows the number of cached images that could be re-used for each cell. Some cells, such as cells 1 and 2, did not re-use rendered images, since it is at beginning of walkthrough. In plot 6, we see frame rates going up and down. When the number of re-used images increased, the frame rate increased. However, when more cells invisible in the previous view became visible in the current view, the frame rate dropped. Therefore, rendering could not stay constant and could produce some annoying jerks. In the near future, we will study ways to maintain a constant frame rate. Figs. 7 (a) and (b) show two extra views of the colonic structure obtained in the course of navigation. Finally, we list below some points of comparison between our method and a recent work [2].



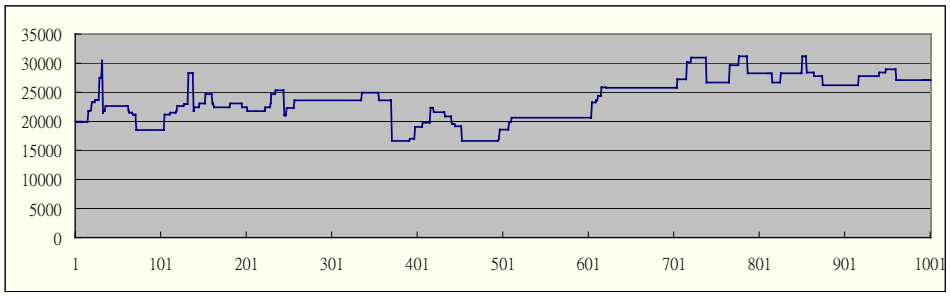
Plot 6. The rendering rate was about 20K triangles per frame using LOD with image caching during 6001 frames

- Our method is intended for PC platforms, while that in [2] was run on a high-end graphics workstation. No special graphics hardware is required for the proposed method; the method in [2] requires the use of special hardware to cull invisible cells.
- Visibility determination in our scheme is computed a ***priori***; the method in [2] determines it during rendering.
- To speed up rendering, multiple levels of mesh representations are adaptively used or cached-images are used. Furthermore, when PVS is employed, the proposed rendering rate is superior to that achieved using the method in [2], which only exploits the original mesh.

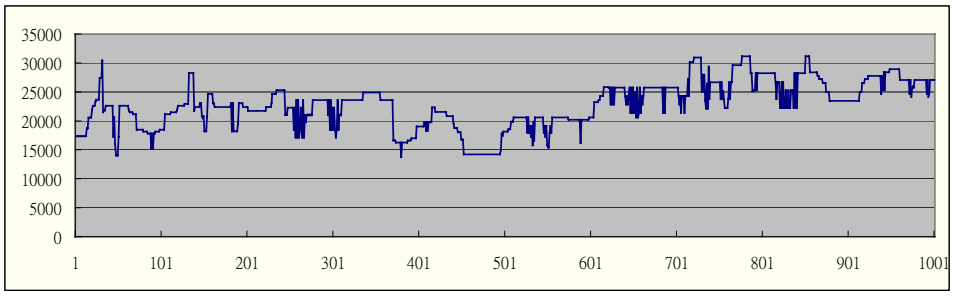
For more detailed comparison between our system and that in [2], please refer to our previous paper [12].



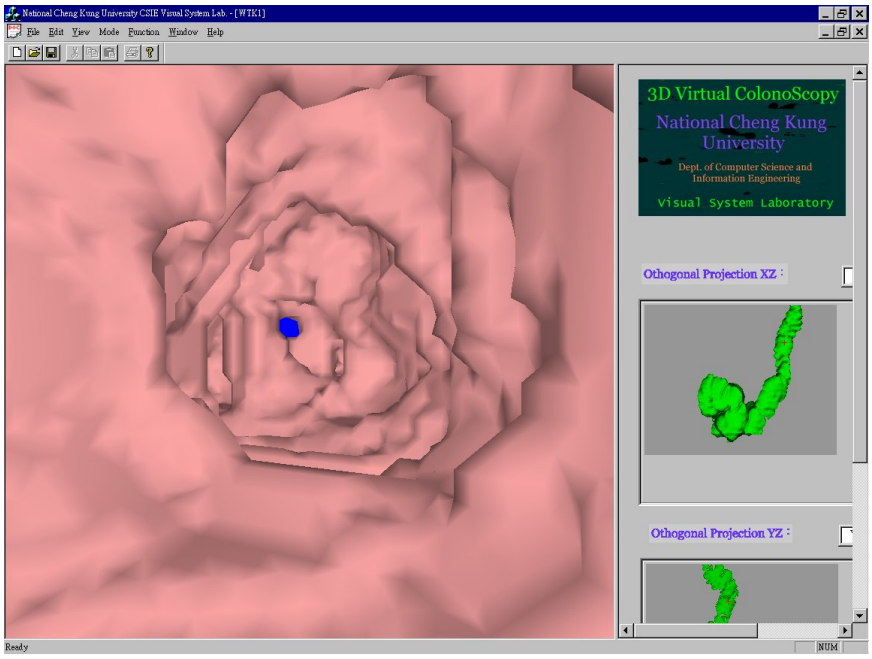
Plot 7. The number of images reused for each cell during 6001 frames



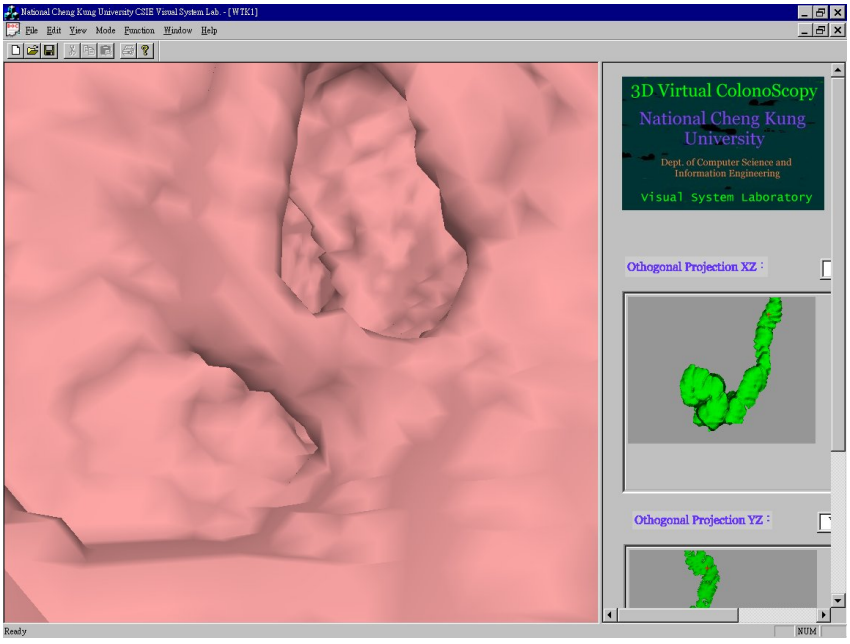
Plot 8. The average number of triangles required using LOD between frames 1001 and 2001.



Plot 9. The average number of triangles required using LOD with image caching between frames 1001 and 2001.



7 (a)



7(b)

Fig. 7. (a) and (b) show two interior views of the colonic structure.

5. CONCLUSIONS AND FUTURE WORK

In this work, we have reported our practice and experience in rendering of the colonic environment. In this study, several interactive rendering techniques were experimentally evaluated. These techniques consist of PVS computation, LOD and image caching. We have proposed an adaptive approach to automatically switching between LODs and image caching. Compared to previous works, the proposed scheme offers comparable performance on low-end PC systems. There are several possible directions for future work. Uniform frame rates are needed to achieve better simulation. Assuming that LODs and image caching are used, we hope to achieve optimal rendering quality but with a uniform frame rate. Additionally, the view-dependent LOD approach is also very promising for producing smooth LOD switching. We plan to investigate it for our system in the near future. Our current approach only displays the internal surface of the colon. We are planning to visualize the surface beneath the tissue via computationally intensive volume rendering. Therefore, other kinds of expediting techniques, such as parallel processing might be investigated. Finally, the aim of our proposed technique is to develop 3-D virtual colonoscopy and the make its practical on commonly used PC platforms. There is still much work to be done, such as to develop a realistic camera model and to achieve realistic appearance of colon morphological movement and real colonoscopy simulation.

ACKNOWLEDGEMENTS

This research was supported by the National Science Council, Republic of China, under contract NSC-88-2213-E-006-012.

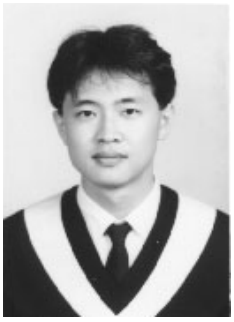
REFERENCES

1. L. Hong, A. Kaufman, Y. Wei, A. Viswambharn, M. Wax, and Z. Liang, “3D virtual colonoscopy,’’ in *Proceedings of 1995 Symposium on Biomedical Visualization*, 1995, pp. 26-32.
2. L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, “Virtual voyage: interactive navigation in the human colon,” in *Proceedings of ACM SIGGRAPH '97*, pp. 27-34.
3. T.-Y. Lee, B.-H. Lin, C.-H. Lin, Y.-N. Sun, and X.-Z. Lin, “Interactive 3D virtual colonoscopy system,” in *Proceedings of International Computer Symposium*, 1998, pp. 23-30.
4. W. Lorensen, F. Jolesz, and R. Kikinis, “The exploration of cross-sectional data with a virtual endoscope,’’ R. Satava and K. Morgan (eds.), *Interactive Technology and the New Medical Paradigam for Health Care*, ISO Press, 1995, pp. 221-230
5. G. J. Wiet, R. Yagel, D. Stredney, P. Schmalbrock, D.J. Sessanna, Y. Kurzion, L. Rosenberg, M. Levin, and K. Martin, “A volumetric approach to virtual simulation of functional endoscopic sinus surgery,” *Medicine Meets Virtual Reality*, Vol. 5, 1997, pp. 167-179.
6. Airey, J., Rohlf, J. and F. Brooks, “Towards image realism with interactive update

rates in complex virtual building environment," in *Proceedings of ACM Symposium on Interactive 3D Graphics*, Vol. 24, No. 2, 1990, pp. 41-50.
7. H. Hoppe, "Progressive mesh,'' in *Proceedings of SIGGRAPH'96*, pp. 99-108.
8. H. Hoppe, "View-dependent refinement of progressive meshes," in *Proceedings of SIGGRAPH'97*, pp. 199-208.
9. Shade, Jonathan, D. Lischinski, D. H. Salesin, T. DeRose, and J. Snyder, "Hierarchical image caching for accelerated walkthroughs of complex environments," in *Proceedings of SIGGRAPH '96 Proceedings*, 1996, pp. 75-83.
10. MacQueen, J., "Some methods for classification and analysis of multivariate observations," in *Proceedings of Fifth Berkeley Symposium on Math. Stat. and Prob.*, 1967, pp. 281-297.
11. T.-Y. Lee, P.-H. Lin, C.-H. Lin, Y.-N. Sun, and X.-Z. Lin, "Interactive 3-D virtual colonoscopy system," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 3, No. 2, 1999, pp. 139-150.

Tong-Yee Lee (李同益) was born in Tainan county, Taiwan, Republic of China, in 1966. He received his B.S in computer engineering from the Tatung Institute of Technology in Taipei, Taiwan, in 1988, his M. S. degree in computer engineering from National Taiwan University in 1990, and his Ph.D. degree in computer engineering from Washington State University, Pullman, in May 1995. Now, he is an Associate Professor in the department of Computer Science and Information Engineering at National Cheng-Kung University in Tainan, Taiwan, Republic of China. He was with WSU as a Visiting Research Professor at the School of EE/CS during the summer of 1996. He serves as a guest Associate Editor for IEEE Transactions on Information Technology in Biomedicine in 2000 and 2001. He has been working on parallel rendering and computer graphics since 1992, and has published more than 70 technical papers in referred journals and conferences. His current research interests include parallel rendering design, computer graphics, image-based rendering, visualization, virtual reality, surgical simulation, distributed & collaborative virtual environments, parallel processing and heterogeneous computing.

Ping-Hsien Lin (林炳賢) was born in Taipei, Taiwan, Republic of China, in 1970. He received his B.S. degree in mechanical engineering and M.S. degree in computer engineering from National Cheng-Kung University, Taiwan, Republic of China, in 1993 and 1998, respectively. Now, he is working toward his Ph.D. degree in the Department of Computer Science and Information Engineering, National Cheng-Kung University. Mr. Lin's research interests include computer graphics, image processing, virtual reality, visualization and image-based rendering.



Chao-Hung Lin（林昭宏) was born in Kaoshiung, Taiwan, Republic of China, in 1973. He received his B.S. degree in computer science/engineering from Fu-Jen University and M.S. degree in computer engineering from National Cheng-Kung University, Taiwan, Republic of China, in 1997 and 1998, respectively. Now, he is pursuing his Ph.D. degree in the Department of Computer Science and Information Engineering, National Cheng-Kung University. Mr. Lin's research interests include computer graphics, image processing, virtual reality, visualization and interactive rendering.