Interactive 3-D Virtual Colonoscopy System

Tong-Yee Lee, Ping-Hsien Lin, Chao-Hung Lin, Yung-Nien Sun, and Xi-Zhang Lin

***Abstract*—We describe a low-cost three-dimensional (3-D) virtual colonoscopy system that is a noninvasive technique for examining the entire colon and can assist physicians in detecting polyps inside the colon. Using the helical CT data and proposed techniques, we can three-dimensionally reconstruct and visualize the inner surface of the colon. We generate high resolution of video views of the colon interior structures as if the viewer's eyes were inside the colon. The physicians can virtually navigate inside the colon in two different modes: interactive and automatic navigation, respectively. For automatic navigation, the flythrough path is determined *a priori* using the 3-D thinning and two-pass tracking schemes. The whole colon is spatially subdivided into several cells, and only potentially visible cells are taken into account during rendering. To further improve rendering efficiency, potentially visible cells are rendered at different levels of detail. Additionally, a chain of bounding volume in each cell is used to avoid penetrating through the colon during navigation. In comparison with previous work, the proposed system can efficiently accomplish required preprocessing tasks and afford adequate rendering speeds on a low-cost PC system.**

***Index Terms*—Adaptive selection of level of detail, flythrough path, interactive rendering, level-of-detail, potentially visible set, segmentation, virtual colonoscopy.**

I. Introduction

COLON CANCER is one of the leading causes of cancer-related deaths per year in industrialized nations [1], [2]. Most large-bowel malignancies arise from preexisting adenomas; this is substantiated by a decreased rate of cancer occurrence after colonoscopic polypectomy. Colonoscopy is a common diagnostic and surgical technique performed in hospitals for the detection of polyps or carcinoma of the colon. During an optical colonoscopy, an optical probe is inserted into the colon that examines the colon's interior structures. The physician can walk through the colon and explore the regions of interest. However, there are several inherent drawbacks in this commonly used routine. In contrast, virtual colonoscopy [3], [4] is an alternative technique to perform the same routine without inserting an optical probe into the human body. This promising method integrates medical imaging, computer graphics, and virtual reality technologies. Compared with the real colonoscopy, it has many attractive features. First, its examination is noninvasive, and, therefore, it does not require sedation or anesthesia to reduce the patient's discomfort. Second, colonoscopy is associated with a small of risk of perforation, and virtual colonoscopy can eliminate this risk [3], [4]. Third [4]–[6], it allows easy control of virtual camera parameters (i.e., provide wider viewing frustum) as well as guided flythrough paths. Furthermore, it is more convenient to localize the three-dimensional (3-D) positions of polyps using it.

With the increasing availability of high-speed medical scanners and new methods for medical image processing, a stack of medical images can be three-dimensionally reconstructed and visualized [7], [8]. Using this scenario, it demonstrates many significant promises in clinical applications such as surgery simulation and radiotherapy planning. Among them, virtual colonoscopy and endoscopy are recently proposed applications that are still under development and investigation in several academic and medical institutes [9], [10].[0] These research studies are generally conducted on the high-end graphics workstations [4]–[6] or expensive parallel architectures [11], [12]. Therefore, one primary motivation of this paper is to develop a 3-D virtual colonoscopy system on low-cost PC platforms. Our techniques aim at making virtual colonoscopy practical on an affordable system. We believe that a low-cost and popular platform is an important factor to increase its popularity and to speed up its evolution.

The paper is organized as follows. In Section II, we present an overview of relevant work. In Section III, we describe our system components. In Section IV, we introduce methods on colon segmentation, and in Section V we present schemes to generate the flythrough path. Section VI states our methods to achieve interactive rendering performance as well as to avoid navigating outside of the colon. We present experimental results and discussions in Section VII. Finally, we discuss conclusions and future work in Section VIII.

II. Previous Work

At the State University of New York at Stony Brook (SUNY Stony Brook), Prof. Arie Kaufman [4], [5], [11] leads a team on the development of the virtual colonoscopy system. His team has made substantial progress, with automatical plan navigation and interactive rendering by a hardware-assisted visibility algorithm [5] or parallelization on the multicomputer architecture [11]. In [5], a potential field and rigid body dynamics techniques are employed to fully control camera parameters while avoiding collision. Flight path is automatically generated and determined by the distance field from

Manuscript received September 14, 1998; revised November 3, 1998. This work is supported in part by the National Science Council, Taiwan, R.O.C., under Grant NCS-88-2213-E-006-012, Grant NCS-88-2213-E-006-037, and Grant NCS-88-2213-E-006-034.

T.-Y. Lee, P.-H. Lin, C.-H. Lin, and Y.-N. Sun are with the Visual System Laboratory, Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, R.O.C. (e-mail: tonylee@mail.ncku.edu.tw).

X.-Z. Lin is with the Department of Internal Medicine, National Cheng-Kung University Hospital, Tainan, Taiwan, R.O.C.

Publisher Item Identifier S 1089-7771(99)01792-6.

[0] Visualization Laboratory of the Department of Computer Science at the State University of New York at Stony Brook and is headed by Leading Professor Arie E. Kaufman, http://www.cs.sunysb.edu/~vislab/

the colonic surface. However, the preprocessing stage is an extremely time-consuming process. For one data set, this process took several hours on high-end SGI workstations. In their most recent work [11], a parallel version of software-assisted ray-casting scheme is exploited to visualize tissues beneath the colonic surface. In the research group at GE, Lorensen *et al.* [6] developed endoscopy techniques providing real-time high-resolution video views of the interior of hollow organs and cavities that exist within the human body. In this approach, the organ surface is extracted by the "Marching Cubes" algorithm [13]. To reduce the number of triangles generated and display rendering time, the triangle decimation algorithm [14] is used to eliminate the flat portion of the surface. However, this simplification could potentially lead to loss of details contained in the original highly curved surface. To generate a planned navigation inside the colon, a common wavefront propagation algorithm is used [6]. Smooth flythrough animations along a planned camera trajectory are achieved through the key-frame technique. Yagel *et al.* [12] describe new methods for real-time volume rendering, model deformation, interaction, and the haptic devices and demonstrate the utilization of these technologies in the real-world application of endoscopic sinus surgery (ESS) simulation. The "volume spatter" and "volume slicer" technologies are developed to provide interactive rendering performance for ESS. Most of the related prototypes mentioned above were implemented either on high-end graphics workstations [4]–[6] or on multicomputer systems [11], [12].

III. Virtual Colonoscopy

In this section, we present an overview of the proposed virtual colonoscopy system on the PC platform. To extract the inner surface of the patient colon, there are several preprocessing steps including patient preparation, air inflation into the colon, and surface extraction. More details about each step can be found in [4]. Our system is built on the PC Patium-2 dual-CPU server platform with a graphics acceleration card. The cost of establishing such a system is lower than other proposed systems [4]–[6], [11], [12]. Fig. 1 shows the process of the proposed system. The colon data is acquired by scanning a patient's abdomen using a helical CT scanner at the Hospital of National Cheng-Kung University. Since scanned data could be too large to be further processed on our system, data might be scaled down to a proper size. At the segmentation stage, we primarily exploit the 3-D region growing method to automatically extract 3-D colon volume from the original data. However, due to some technical difficulties in air inflation to the colon, in some places there still exists ambiguity between the colon wall and lumen. In other words, there is not enough contrast between them, so it is difficult to segment the colon and lumen using a simple threshold method. To remedy this problem, there is a two-dimensional (2-D) region growing subprocess allowing medical technicians to manually segment the colon slice by slice.

After extracting the colon data, we three-dimensionally reconstruct the inner surface of it using the Marching Cubes algorithm. Our technique of automatically generating a fly-through animation consists of the following two steps. First, we use a 3-D thinning technique to find the skeleton of the colon. The cross sections of the colon have various sizes, so a 3-D thinning technique will potentially generate many branches along the centerline of the colon. Next, we use a two-pass tracking method to trim these branches, and, therefore, it can yield a single planned path. It is not wise to pass all surface triangles into the graphics pipeline during rendering. To cull invisible triangles, we partition triangles into several regions based upon the curvature of the centerline. Then, for each region, we compute its corresponding potentially visible set (PVS). During rendering, only triangles in the current PVS will be rendered. One approach to improve rendering performance is to replace a complicate mesh by a set of level-of-detail (LOD) approximations. A detailed mesh is used when the object is close to the viewer, and coarser ones are substituted as the object recedes [15]. In this paper, each region is represented by a set of LOD and a method proposed to choose the appropriate level of detail in real time. In this manner, we can achieve adequate interactive rendering in the proposed system.

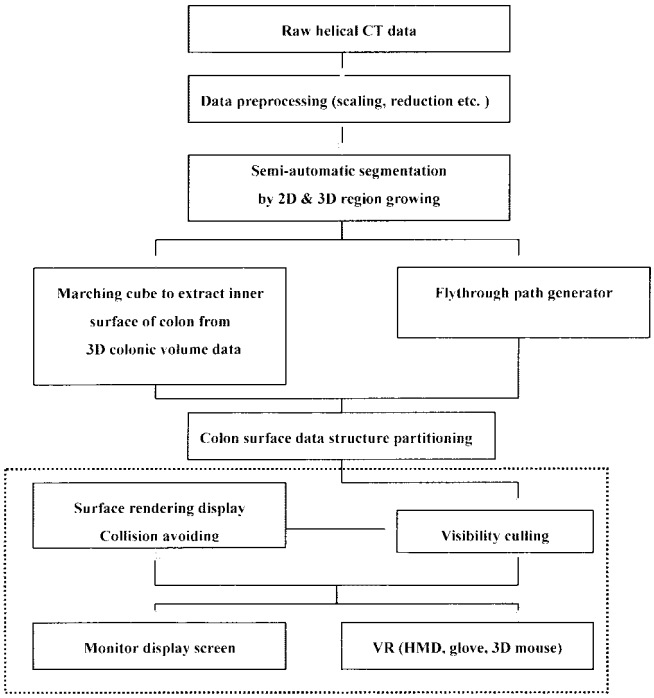
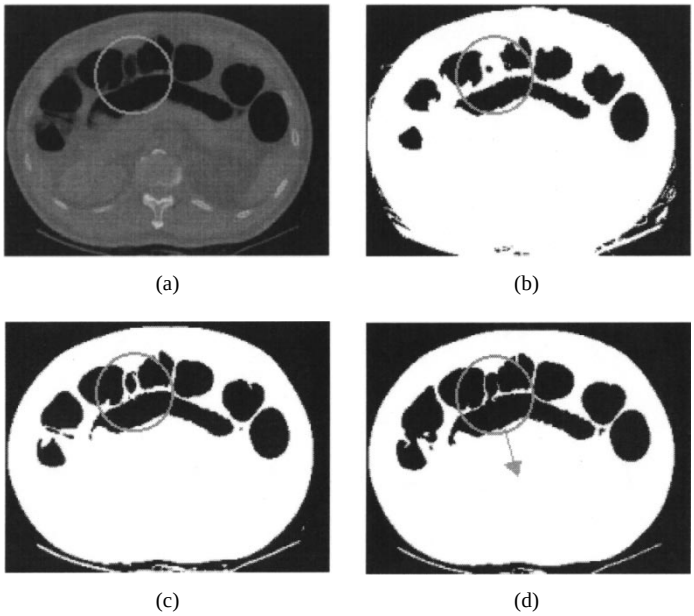


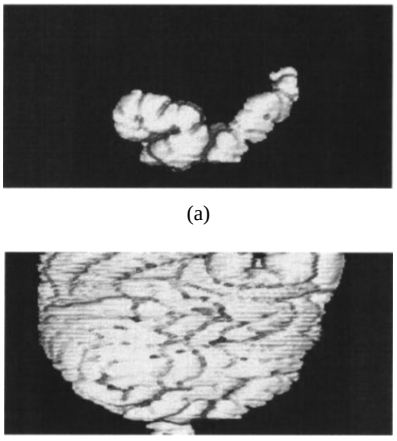
Fig. 1. The process of the interactive virtual colonoscopy system.

To avoid viewer's eyes straying outside the colon, we compute collision detection during navigation. For this purpose, a chain of bounding cylinders are used to constraint movement of the camera within each region. These cylinders can be determined *a priori* in the preprocessing stage. There are two examining modes provided in the proposed system. The physicians can interactively flythrough the colon by manipulating the viewing parameters employing a 3-D mouse or a Dataglove. Another navigating scenario is to automatically navigate through the colon using a planned flythrough path. This scenario can provide a quick overview of the colon structures.



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

Fig. 2. The segmented results using different thresholds. (a) Raw CT data. (b) Threshold: gray level 2. (c) Threshold: gray level 32. (d) Threshold: gray level 50.



(a) (b)

Fig. 3. (a) Incomplete colon as a small threshold is used. (b) The mixture of colon and lumen as a large threshold is used.

IV. Colon Segmentation

At this stage, we exploit the 3-D region growing method to extract colonic volume from the original CT data. Three-dimensional region growing is a procedure that groups voxels or subvolume into a larger volume. Our approach belongs to the voxel aggregation, which starts with a set of "seed" points and from these grow regions by appending to each seed point those neighboring voxels that have similar properties (such as gray level, gradient, and color). We primarily use the gray level of voxel as a growing property: we use a threshold to segment colon data. Fig. 2 shows the segmentation results using different thresholds. From this figure, because of poor contrast between the colon wall and lumen, we cannot successfully extract the colon through a simple threshold method. Therefore, we might obtain only a small portion of the colon [Fig. 3(a)] or generate the mixture of the colon and lumen [Fig. 3(b)].

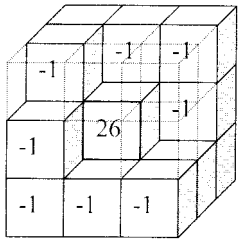


Fig. 4. A 3×3×3 convolution mask.

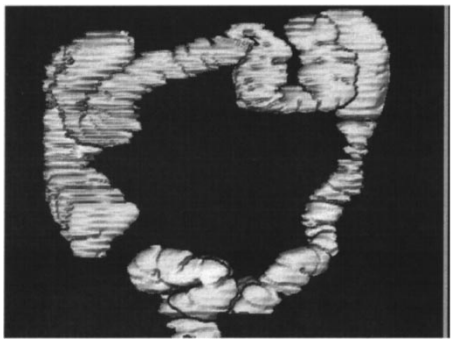
To solve the ambiguity existing between colon and lumen, we refine the primary growing property as follows. For a $N\times N\times N$ volume data, $[0\cdots N=1, 0\cdots N-1, 0\cdots N-1]$, and a convolution template with size 3×3×3 (see Fig. 4), $T[-1\cdots 1, -1\cdots 1, -1\cdots, 1]$. We use two selected thresholds p and q where $0<p<q$. We then use the ***growth*** procedure to determine whether or not a voxel $[i,j,k]$ will grow

procedure ***growth*** ( )
begin
If $[i,j,k]\le k]\le p, [i,j,k]$ will grow;
else if $[i,j,k]\ge q, [i,j,k]$ will not grow;
else {
Compute $c=\sum_{u=-1}^{1}\sum_{v=-1}^{1}\sum_{w=-1}^{1}\cdot V[i+u, j+v, k+w]\times T[u,v,w]$,
where if $u=v=w=0, T[u,v,w]=26$,
otherwise $T[u,v,w]=-1$;
if $c>0, [i,j,k]$ will grow;
}
end

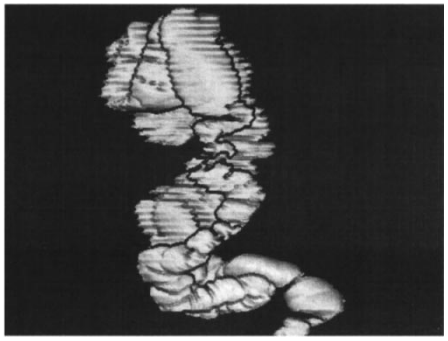
In the above procedure, we will compute a 3×3×3 convolution (c) of $[i,j,k]$ if its intensity is between p and q. If c is positive, this implies that most of the 26 neighbors are colon, so there is a good chance that it is colon also. Therefore, $[i,j,k]$ will grow. Fig. 5 shows extracted 3-D colonic volume after the segmentation process. In case the gray levels of the original data are too ambiguous to be automatically segmented by using the proposed method, we also provide a 2-D region growing dialog box allowing technicians to segment 2-D images slice by slice in an interactive manner (as shown in the center of Fig. 6). Furthermore, we can view currently segmented results using a 3-D rendering dialog box simultaneously to check its correctness. The 3-D rendering is accomplished by an optimum semiboundary (SB) rendering scheme [18] (as shown in the top and bottom of Fig. 6). A snapshot of our segmentation GUI is shown in Fig. 6. After the segmentation, we exploit the Marching Cubes algorithm to obtain the colon interior structures, as shown in Fig. 7.

V. Automatically Generated Flight Path

A skeleton or centerline of 3-D shapes is an efficient and compact shape descriptor, and it can give useful cues for a number of applications in visualization, including automatic navigation, compression, shape description and abstraction,



(a)



(b)

Fig. 5. The extracted colon after the proposed segmentation process. (a) Front view and (b) side view of the colon structure.

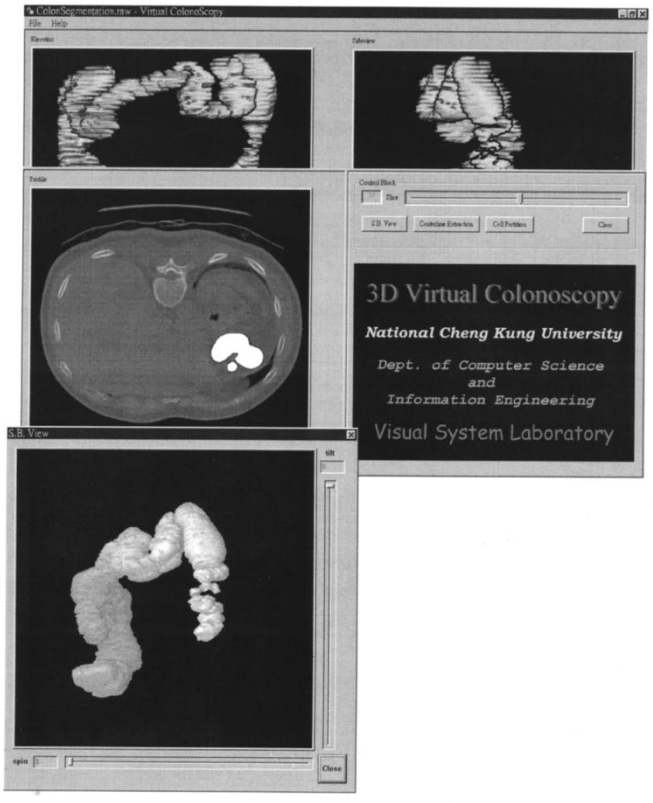
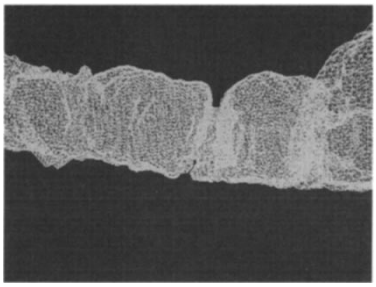


Fig. 6. A snapshot of segmentation GUI.

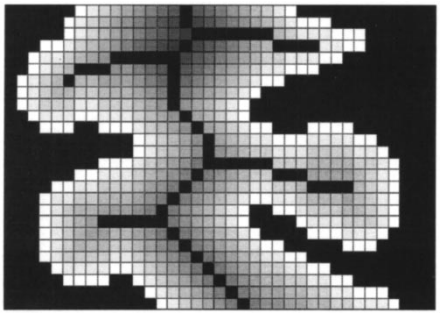
and tracking. In our case, we use a skeleton of 3-D colon in the automatic navigation control, where it guides the virtual scopy to walkthrough the sinuous colon structures. Prof. Kaufman of

(a)

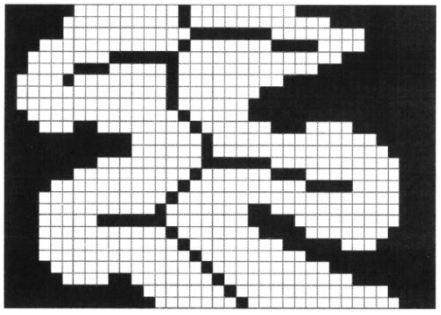


(b)

Fig. 7. Triangle surface generation using the Marching Cube algorithm.



(a)



(b)

Fig. 8. Cross sections with various sizes. (a) The skeleton of the colon is obtained by a "peel-onion" method (each layer is represented by a different color) but it has some branches along the skeleton. (b) The centerline of the colon.

SUNY Stony Brook suggested a "peel-onion" technique [4] (i.e., a 3-D thinning method) to extract the centerline of the colon. The outermost layer of subvolume will not be peeled off until there is only one layer of grid points left, which essentially is the skeleton of the colon [4]. During "peel-off," one must be careful in voxel deletion to avoid breaking the continuity of the desired path. In case the cross sections

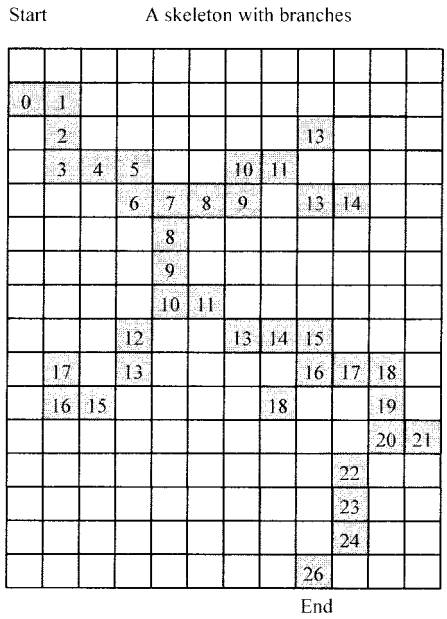


Fig. 9. The first pass in a two-pass tracking method.

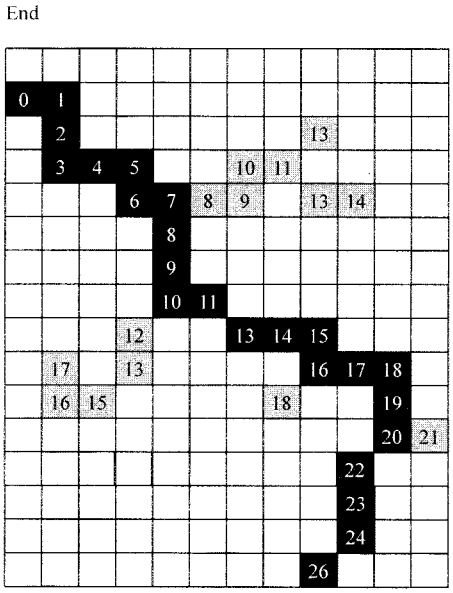


Fig. 10. The second pass in a two-pass tracking method.

of colon are ideally uniform as shown in [4] and [5], this technique poses no problems. We should note that the shapes of colon could become uniform if the patient suffers from a disease such as ileus or takes medication such as buscopam prior to CT scanning. (However, physicians should not suggest that the patient should take such medicine.) Therefore, in most clinical cases, the colon will have various sizes in its cross sections, as shown in Fig. 8(a). In this case, the "peel-onion" scheme will potentially lead to many branches according to its rule of preserving continuity [Fig. 8(a) and (b)]. To trim these redundant branches, we employ a simple two-pass tracking method. Before trimming the branches, the two-end points (starting and ending fly points) of the flight path are specified. At the first pass, we begin with labeling each voxel in an

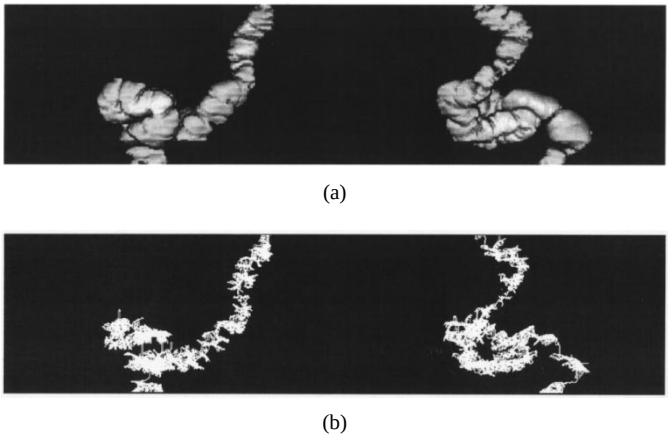


Fig. 11. (a) Two orthographic views of the colon before the "peel-onion" process. (b) Two orthographic views of the colon with branches after the "peel-onion" process.

Fig. 12. The centerline after trimming the branches using two-pass tracking.

incremental manner from the starting fly point. The result (2-D view) is shown in Fig. 9. In this example, we increase the label of a neighboring pixel by one and a diagonal neighboring pixel by two. There are several branches in this example. At the second pass, we start working back from the ending point in a decreasing manner, as shown in Fig. 10. During this secondary process, we will record those pixels we visited. The path consisting of these visited pixels in the second pass is the centerline we need. Fig. 11(a) and (b) shows the results of the "peeled onion." Many redundant branches are visible in this figure. We use a two-pass tracking scheme to generate a unique path, as shown in Fig. 12. As suggested in [4], we also employ cubic splines to smooth our generated path. Finally, we outline the "peeled onion" employed in this paper as follows:

repeat
  Mark every ***colon*** point that is *26-adjacent* to a ***background*** point;
  Delete all ***simplepoints***;
  Remark all undeleted marked points;
until no marked points can be delete;

For $N \times N \times N$ volume data, $[0 \cdots N-1, 0 \cdots N-1, 0 \cdots N-1]$, let p and q be two distinct voxels with coordinates (p_x, p_y, p_z) and (q_x, q_y, q_z), respectively. Then, p and q are

- *6-adjacent* if $|p_x - q_x| + |p_y - q_y| + |p_z - q_z| = 1$;
- *18-adjacent* if $1 \leq |p_x - q_x| + |p_y - q_y| + |p_z - q_z| \leq 2$ and $\max(|p_x - q_x|, |p_y - q_y|, |p_z - q_z|) = 1$;
- *26-adjacent* if $\max(|p_x - q_x|, |p_y - q_y|, |p_z - q_z|) = 1$.

After segmentation using 3-D region growing, each voxel V is classified as either a colon point or a background point. During "peel-off," we delete all ***simple*** points layer by layer

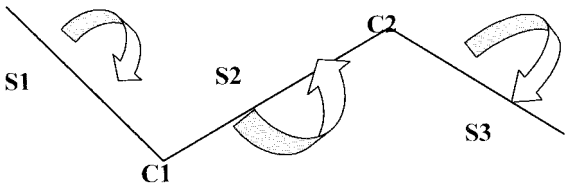


Fig. 13. A cutting is always made on the points with a large curvature.

starting from the outermost layer. A colon point p is said to be ***simple*** if it satisfies the following rules [16]:

- p is *26-adjacent* to only one colon object[1] in p's *26*-neighborhood;
- p is *6-adjacent* to only one background object in p's *18*-neighborhood.

VI. Interactive Rendering

For virtual colonoscopy, an interactive rendering performance is crucial. On a low-end system such as a PC platform, it is difficult to yield this performance by rendering all triangles of the colon surface using the traditional graphics pipeline. As pointed out in [5], given a camera position, only a small portion of the colon is visible to the observer. Previous work on the visibility issue has been explored [19]–[21]. One exploits a hierarchical Z-buffer to efficiently cull unseen objects [21]. The other approach in [17], [19] divides the whole scene into many cells. Each cell has a predetermined PVS. During rendering time, the current PVS is exploited to cull unseen cells; therefore, it can reduce the number of triangles submitted to a graphics pipeline. Kaufman's approach [5] computes PVS on the fly during rendering time as well as proposes a hardware-assisted visibility algorithm called the aggregate cull rectangle (ACR). The ACR efficiently culls unseen cells in near-to-far order. Our approach computes PVS off-line since our PC platform does not have enough computing power nor does it have a hardware-assisted board for visibility testing on the fly. In our approach, we first perform a region subdivision in which the colon structure is partitioned into several cells. We then perform a visibility precomputation in which the set of cells and surface triangles visible from each cell are computed. The results of these precomputations are stored in the display database for use during navigation.

A. Region Partitioning

The colon structure is partitioned into many cells. For tube-like colon structures, a good partition strategy must take the curvature of the centerline into account. As shown in Fig. 13, we can cut the colon structure into three parts at points C_1 and C_2. In this circumstance, triangles at S_3 will most likely be invisible to the triangles located at S_1. The optimum cutting points on the centerline should be the places where they have larger curvature on the centerline such as C_1 and C_2. Based on this observation, the proposed region partition is described as follows. Assume that we have a fixed tolerance ε. Our centerline consists of N points, $P_1, P_2, \cdots, P_N$, where

[1] Among its 26-adjacent points, there might be several disjoint colon objects. In this case, p will not be a simple point.

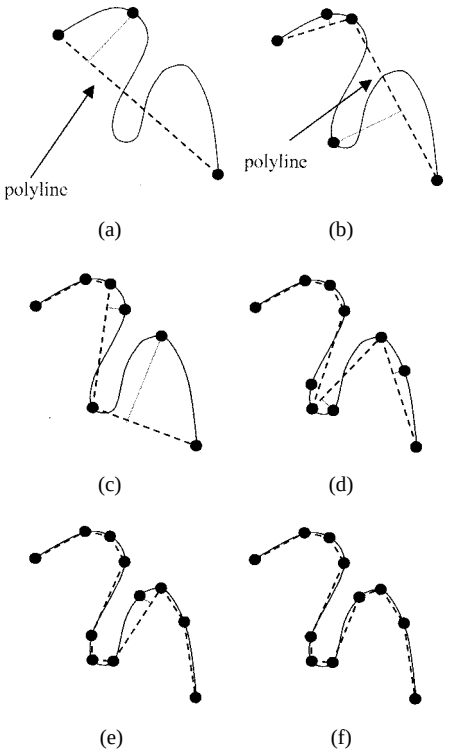


Fig. 14. The process of centerline partitioning. (a) The first cut is made at the position with the highest curvature. (b)–(e) Continuous cuts are made. (f) The final cut is made and the partitioning process is finished.

P_1 and P_N are two end points. Initially, we connect the straight line between P_1 and P_N, and termed L. We then use the ***cell_partition*** procedure to recursively partition the colon structures described below:

procedure ***cell***_mbipartition ( )
begin
1. For each point P_i of the centerline, compute the distance d_i from L;
2. Find largest d_i;
3. If $d_i < \varepsilon$, then return;
4. P_j with largest distance d_j will be a new cutting point, and replace the segment L with two new segments;
5. Call ***cell_partition*** ( );

end

Fig. 14 shows the process of partitioning. This process is similar to the process that finds an approximately polyline that most fits the curve of the centerline. In this manner, the cutting point is always made on the place where it has a larger curvature on the centerline. After determining the cutting points, we start partitioning the colon cell by cell. The normal vector of each cutting plane can be approximated by the summation of two vectors in two continuous polyline segments, as shown in Fig. 15.

Two additional criteria are implemented in our colon subdivision. The first criterion is to limit the accumulated number of triangles in each region. If it has reached a certain threshold, we will further divide this region into smaller subregions. The second criterion is to check if the width of the cross section (of

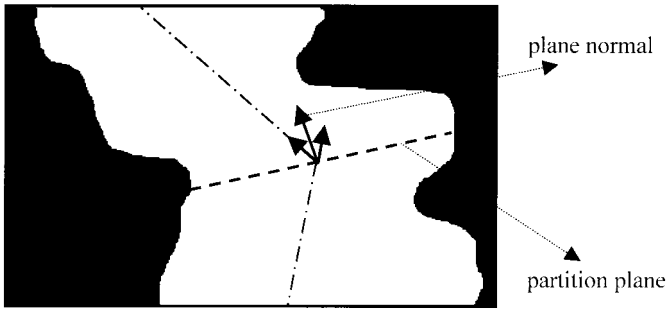


Fig. 15. The normal vector of a cross-section plane.

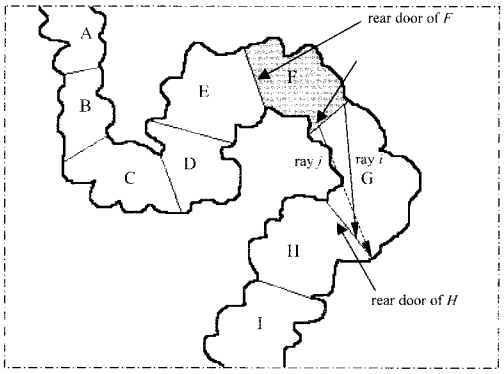


Fig. 16. The computation of PVS for each cell.

the colon) varies significantly beyond some threshold in each region. If it does, we will repartition this region into several smaller subregions also. For each subregion, we compute its PVS and also use a cylinder within this subregion as its bound volume for the purpose of collision detection. While walking through the colon, we continually test if the virtual camera is outside the bound volume of the current subregion. If a collision occurs, the virtual camera will be forced to move back inside the bound volume, and, therefore, the camera viewing is always kept inside the colon. The purpose of this design is twofold. First, we are only interested in the interior regions of the colon and want to avoid penetrating the sides of the colon. Second, the design avoids passing all surface triangles of the colon to the graphics pipeline (i.e., saves rendering time).

B. PVS Determination

Once the colon partition has been accomplished, we perform a PVS precomputation in which the portion of colon structures visible from each cell is determined. Fig. 16 illustrates our method of computing PVS. For each region, three are two cross sections (front and rear doors) shared with two neighboring regions (except for the first and last regions). Let's take the 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 nonimmediate 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, the ray_i can reach at cell H, but ray_j is blocked (since it hits the boundary before hitting H's rear door). Therefore, cell

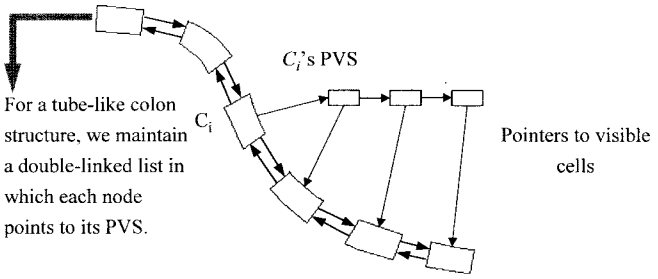


Fig. 17. The double-linked list used to maintain the PVS.

H is visible to F and will be included in F's PVS. However, if tno visible ray (such as ray_i) is found, then cell H will not be included. In case cell H is not visible to F, we will terminate the search for F's PVS because cell H (invisible cell) will block the remaining cells. Otherwise, the search for the PVS will continue until the end of the colon or an invisible cell is found. For each ray traversal, it is efficiently performed voxel by voxel using a 3-D DDA stepping algorithm [22]. 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 can access the cells in the current PVS, as shown in Fig. 17.

C. Adaptive Display of Colon Triangular Meshes

In Fig. 17, we assume that the PVS of C_i is $\{A, B, C, D\}$ in *near-to-far* order and now the camera is located at C_i. In this situation, the physician's attention is concentrated on the interior shapes of C_i. Therefore, it will be attractive to represent other cells (A, B, C, D) by a sequence of approximations at various levels-of-detail. The crudest approximations of a cell are used when the camera is far from that cell, while more detailed versions are substituted as the camera approaches. We are likely to display C_i and A using the original triangular meshes but B, C, D with lower resolution versions (for quick display) varying with its distance from C_i. While the camera moves to cell A, cells B, C, and D then will progressively improve the display quality as more detailed approximations are rendered. To support this progressive display of cells, we create multiresolution representations for each cell. Switching between different resolutions is primarily determined by the order (*near-to-far*) in the current PVS.

Hoppe [23] introduced an approach to minimize an energy function that is explicitly used to in the reduction of the number of vertices in an initially dense mesh of triangles. This approach tried to maintain the competing desires of conciseness of representation and fidelity to the original mesh of triangles. In this approach, the edge energy E_{energy} will be assigned to each edge E. During mesh simplification, those edges with the smallest energy will be the candidates to be collapsed (***ecol***, shown in Fig. 18). In this paper, we adopt this approach and define our energy function as follows.

- $E_{\text{energy}} = W_1 * E_{\text{spring}} + W_2 * E_{\text{sharp}}$, where W_1 and W_2 are user-specified weights.
- $E_{\text{spring}}(M) = \Sigma_{\{i,k\} \in k}\ k\|v_j - v_k\|^2$, where (v_j, v_k) is an edge of the mesh and k is a spring constant.

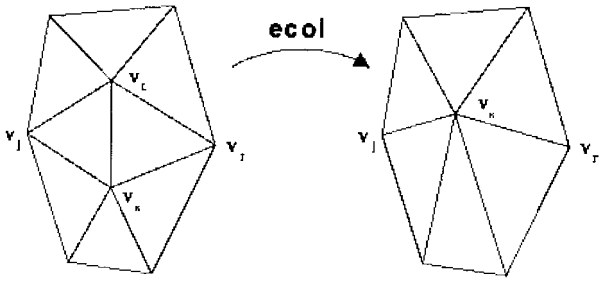


Fig. 18. An example of the edge collapse (i.e., ***ecol***).

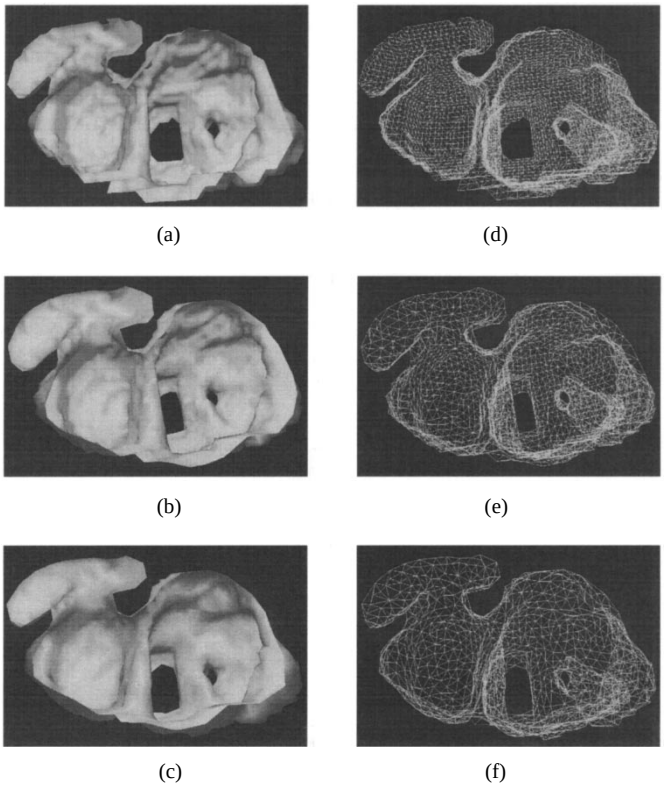


Fig. 19. Multiresolution representations of a colonic cell. (a) Initial mesh. (b), (c) Various levels-of-detail. (d)–(f) Wireframe representations for (a)–(c).

- An edge $\{v_j, v_k\}$ is termed as *sharp edge* if it belongs to one of the following:
  a) $\{v_j, v_k\}$ is a boundary edge of the mesh;
  b) an angle θ between normal vectors at vertices v_j and v_k is larger than a user-specified threshold.

Here, we show some details of mesh reduction for a cell in our implementation. Fig. 19 shows the results of mesh simplification and Table I gives details of each version. In this table, there are three versions available for each colonic cell. The number of triangles is approximately reduced by a factor of two between two consecutive levels. However, the difference in appearance between them is small. In our implementation, E_{sharp} is more weighted than E_{spring}. There are two reasons for this choice. First, the shapes of polyps will potentially have larger θ; we can keep them as undeleted as possible during mesh simplification. Second, we do not want to collapse the boundary edges because we want to maintain

TABLE I
CHARACTERISTICS AT VARIOUS LEVELS-OF-DETAIL

A colonic cell (512*512*30)		
Levels of detail	*#triangles*	*#Collapse(ecol)*
(a) · (d)	11470	0
(b) · (e)	4147	2000
(c) · (f)	2177	3000

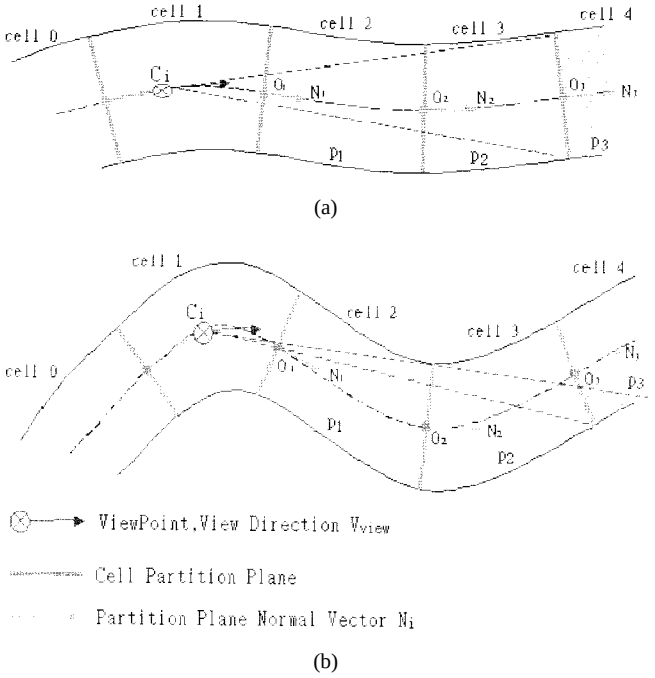


Fig. 20. The adaptive LOD selection scheme.

edge connections between two neighboring cells as well as its topology of shape.

During rendering, we propose an adaptive selection of the LOD algorithm.

We make the following assumptions (see Fig. 20).

- Each cell has m versions of levels-of-detail, $\ell_0, \ell_1, \cdots \ell_{m-1}$, 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 3-D position are denoted as V_{view} and *eye*, respectively.
- There are n cells included in C_i's PVS, and denoted as $\{p_1, p_2, \cdots p_n\}$, where p_i is ordered near-to-far 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.

To automatically determine transitions of LOD, we first normalize LOD selection based on several representative views within each cell at the preprocessing stage. Then, during rendering time, we use this precomputed 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} for all cells and use the values

as the approximate distribution for all possible views inside the colon. Next, we need to determine how to classify this distribution into m levels. For the purpose of classification, the *k-means* method [24] is exploited to find the range of *sels* for each level of LOD. 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 LOD's according to level classification in the preprocess. Later, we will present detailed experimental results to illustrate our method:

```
procedure normalized_selection ( )
begin
  For all colonic cells do
  {
    For j = 1 to 3 do
    // three eye positions, starting, middle and
       ending points, respectively,
    {
      dist = ||eye_j − O_1||;
      θ = cos^{-1}(V_view · N_1);
      For i = 1 to n do
      // for all PVS cells from the current cell
      {
        sel_ji = w_1 × dist + w_2 × θ; // w_1 and w_2
        are user-specified weights
        dist = dist + ||O_i − O_{i+1}||;
        // distance heuristics
        θ = θ + cos^{-1}(N_i · N_{i+1});
        // viewing direction heuristics
      }
    }
  }
    Use k-means method to classify all
    sel_jis into m groups;
end
```

In the above algorithm, the parameters, *dist* and V are two key factors in determining the selection of LOD for each cell in the current PVS. During preprocessing, we compute weights at three places (starting, middle, and ending points) as a rough estimation to determine transitions of LOD's. Using the above algorithm, the farther cell will select a coarser one. The parameter V indicates the potential of being blocked by the nearer cell. Therefore, the larger θ will tend to select a coarser one also. In Fig. 20, for p_3, case (a) will tend to select a finer one than case (b).

VII. Experimental Results and Discussion

To perform our experiments, we have collaborated with the physicians at the Hospital of National Cheng-Kung University. The size of test colon CT data is 512×512×200.[2] Our initial experiments with our system have been tested against several patient data sets. The encouraging results of discovering polyps are verified by the collaborating physicians. Fig. 21 shows the user interface of the navigation environment. In this environment, we provide a colonoscopy view and the orthographic views of two colons (xz and yz directions). As the camera moves along the colon, one red spot (representing the current location of the camera) will also move in these two orthographic views of the colon. In case polyps are detected, we can record the coordinates of these two views for further clinical evaluation.

Using our partitioning scheme, we divide the colon into 50 cells, as shown in Fig. 22. Fig. 23 shows the number of triangles contained in each cell, and Fig. 24 shows the number of triangles in the PVS of each cell. On average, there are 47 944 (48 K) triangles contained in the PVS per cell. In case we do not exploit PVS strategy, there are approximately 470 K triangles required to pass the rendering pipeline. Therefore, the PVS can significantly reduce the number of triangles.

Next, we will show that PVS strategy can be further improved in conjunction with the proposed adaptive selection of the LOD scheme. In this experiment, we use six levels-of-detail for each cell of the colon structures, and weights for w_1 and w_2 are 1 and 2, respectively. Fig. 25 shows the distribution of precomputed sel_{ji} for the whole colon under this configuration, and Fig. 26 shows six groups after *k-means* classification. Note that all computed sel_{ji}'s are scaled to a range of integers $(0, 110)$. In our plots, the X axis represents the scaled sel_{ji} and the Y axis represents the number of views with the same scaled sel_{ji} integer. Each classification represents a range of *sel* for a specific level of LOD. Fig. 27 shows the number of rendered triangles for each cell during automatic navigation. On the average, it renders 25 K triangles. Therefore, using adaptive selection of the LOD method, the number of rendered triangles is reduced from 48 K to 25 K. This number is a small portion of the colonic surface (470 K triangles). On a PC-based platform, it is easy to achieve an interactive rate to render 25 K triangles. We have achieved adequate rendering speed for image resolution at 512×512 on a dual-CPU Pentium-2 PC server system. We should point out that [6] also used triangle reduction to improve rendering. However, their work does not adaptively select an appropriate version of colon cells according to distance and view direction. Since our work is closely related to [5], a detailed comparison is listed as follows.

- The work in [5] was implemented on the SGI R10000 processor with InfiniteReality and our work was done on a dual-CPU Pentium 2 PC server system. Both exploited the region growing method to perform colon segmentation. However, no detailed information was provided in [4] and [5], [11]. In this paper, we propose schemes to solve the ambiguity between the colon and lumen. Additionally, 2-D region growing and 3-D rendering dialog boxes are provided to assist in segmentation. For our test data, it took 3–4 min to accomplish this task. Both works used the "Marching Cube" algorithm to generate the inner surface of the colon.
- For the centerline extraction, the study in [5] used a single-source shortest path algorithm; we first exploit 3-D thinning (the same as their early work [4]) to extract the

[2] Originally, we acquired 512×512×384 colon data. However, the number of surface triangles is beyond our memory capacity. Therefore, in our experiments, we only process triangles generated from a portion of the original colon data (i.e., 512×512×200).

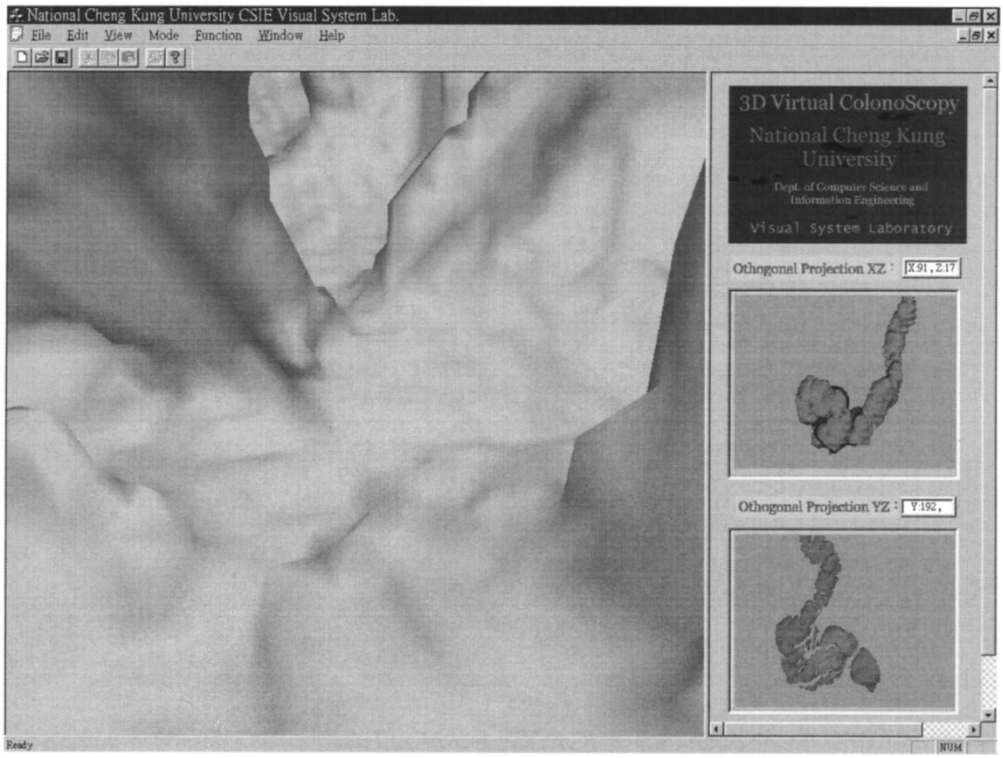
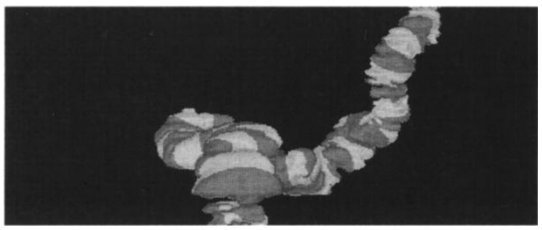
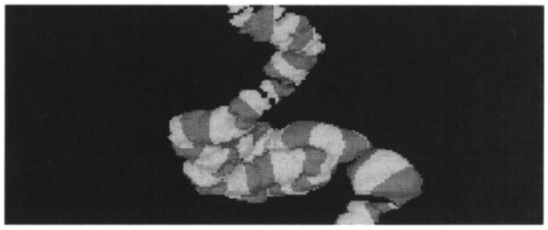


Fig. 21. User interface of navigation environment.



(a)



(b)

Fig. 22. Colon with 50 cell partitions.

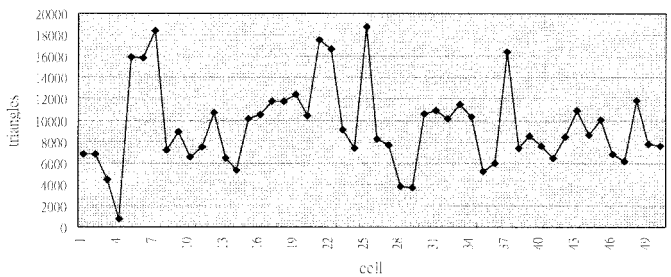


Fig. 23. The number of triangles contained in each cell.

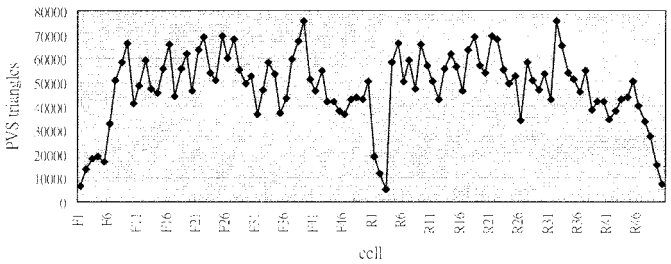


Fig. 24. The number of triangles visible to each cell (F: forward viewing; R: backward viewing).

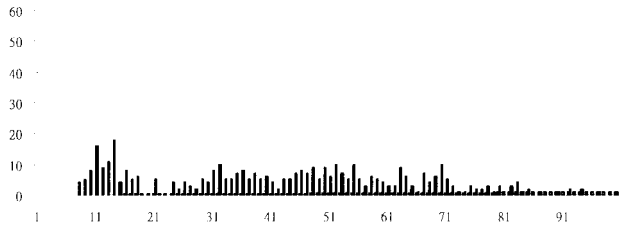


Fig. 25. Estimated distribution of *sel* for the possible views inside the colon. Note that the X axis represents the scaled sel_{ji} integer range $(0, 110)$ and the Y axis represents the number of views with the same scaled sel_{ji} integer.

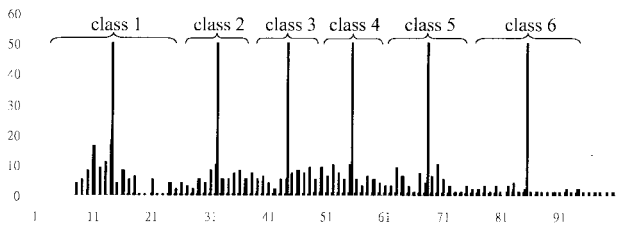


Fig. 26. The classification of *sel* using the *k-means* method.

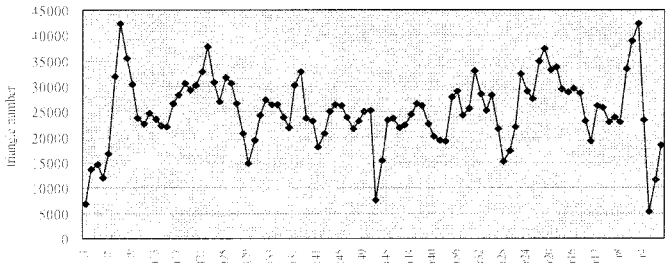


Fig. 27 The number of triangles rendered using an adaptive selection of LOD. On average, 25 K triangles are required.

skeleton of the colon and then use a two-pass tracking scheme to trim branches. In our case, the proposed scheme took 10 min. On the contrary, segmentation and centerline extraction took several hours [4], [5].

- For interactive rendering, the workers in [5] partitioned the colon into cells based on the centerline in the preprocessing stage and exploited a hardware-assisted visibility algorithm to determine cell's visibility during rendering. The distance of the centerline and the number of triangles are criteria during partitioning. On the other hand, we exploited an approximating polyline method to partition the colon. The curvature of the centerline is the first priority, and the distance and number of triangles are the secondary criteria to be taken into account during partitioning. Our PVS was determined during preprocessing and was calculated using an efficient ray traversal scheme between two cross sections. Furthermore, to reduce the number of rendered triangles, we adaptively select one version of each cell representations based on distance and sight-line. On average, the number of triangles per frame in [5] ranged from 146 K to 53 K for three data sets, while our reported number is about 25 K. Both systems achieve adequate interactive speeds for image size 512×512.
- In both systems, there are two types of navigation: automatic (through the centerline) and interactive modes. In [5], a physically-based camera control model was presented to control the movement of the camera. This interesting scheme computed a potential field to achieve collision detection, which was an expensive computation. To be affordable on a PC system, we simply achieved collision detection by using a chain of cylinders as bounding volume. While it sacrifices some flexibility of camera movement, this method is very fast.

VIII. Conclusion and Future Work

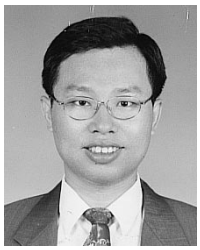
The virtual colonoscopy is an emerging noninvasive technique to detect polyps. It eliminates the patient's discomfort caused by conventional colonoscopy surgery. In this paper, we presented a PC-based colonoscopy system. Compared with other related work [5], our method is attractive due to its low cost, while the presented system has similar capabilities and performance. The primary contributions in this paper are summarized as follows. We proposed a colon segmentation scheme that provides a solution to the ambiguity existing between the colon and lumen. Two-pass tracking and 3-D thinning schemes are exploited to generate a centerline of the colon. To achieve interactive rendering, the approximating polyline partitioning and visibility precomputation was performed during preprocessing. With these schemes, a small portion (10%) of the total triangles is passed to the graphics pipeline. Furthermore, we propose an adaptive selection of LOD to further reduce the number of rendered triangles. As a results on average, only 25 K triangles are required for rendering. Rendering this amount of triangles is affordable on PC systems at interactive rate. Additionally, a simple collision detection is exploited to avoid penetrating the colon's surface. In general, the preprocessing stages take less than 1 h on our PC system, which include generating surface, extracting the centerline, partitioning cells and determining the PVS, and forming a chain of cylinders as bounding volume in each cell. In contrast, the methods in [4], [5] took several hours to accomplish the preprocessing stages.

There are several extensions that can be done in the near future. First, we plan to design a cost-effective camera model to navigate the colon in a realistic manner as in [5]. At present, the functionality of the colonoscopy is to visualize the shapes of the inner colon surface. It is still difficult to present the correct appearance (i.e, color and texture) of the colon surface using a single CT image modality. Therefore, we are collaborating with physicians to correctly visualize the surface appearance with a combination of other image modalities such as ultrasonic images. Finally, we would like to implement a colonoscopy simulator allowing polyp removal (surgical operations) and control (back and forward, twist and rotate) of a tube-like fiberscopy.

References

[1] E. Silverberg, C. E. Boring, and T. S. Squires, "Cancer statistics," *CA Cancer J. Clin.*, vol. 40, pp. 9–26, 1990.
[2] D. M. Eddy, "Screening for colorectal cancer," *Ann. Intern Med.*, vol. 113, pp. 373–384, 1990.
[3] H. M. Fenlon and J. T. Ferrucci, "Virtual colonoscopy: What will be issues be?" *AJR*, vol. 169, pp. 453–458, Aug. 1997.
[4] L. Hong, A. Kaufman, Y. Wei, A. Viswambharn, M. Wax, and Z. Liang, "3D virtual colonoscopy," in *Proc. 1995 Symp. Biomedical Viusualization*, 1995, pp. 26–32.
[5] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, "Virtual voyage: Interactive navigation in the human colon," in *Proc. ACM SIGGRAPH '97*, 1997, pp. 27–34.
[6] W. Lorensen, F. Jolesz, and R. Kikinis, "The exploration of cross-sectional data with a virtual endoscope," in R. Satava and K. Morgan, Eds., *Interactive Technology and the New Medical Paradigam for Health Care*. Washington, DC: ISO Press, 1995, pp. 221–230.
[7] M. Levoy, "Display of surfaces from volume data," *IEEE Computer Graphics and Applications*, vol. 8, no. 5, pp. 29–37, May 1988.
[8] A. Kaufman, *Volume Visualization*. Los Alamitos, CA: IEEE Computer Society Press, 1991.
[9] The Volume Graphics Research Group, Dept. Computer and Information Science, The Ohio State University, http://www.cis.ohio-state.edu/volviz/
[10] GE Research and Development, Computer Graphics and System Program, http://www.crd.ge.com/esl/cgsp/
[11] S. You, L. Hong, M. Wan, K. Junyaprasert, A. Kaufman, S. Muraki, Y. Zhou, M. Wax, and Z. Liang, "Interactive volume rendering for virtual colonoscopy," in *Proc. IEEE Visualization'97*, pp. 433–436.
[12] 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*, San Diego, CA, vol. 5, Jan. 1997.
[13] W. E. Lorensen and H. E. Cline, "Mar.ing cubes: A high resolution 3D surface reconstruction algorithm," *Comput. Graph.*, vol. 21, no. 3, pp. 163–169, July 1987.

[14] W. J. Schroeder, J. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," *Comput. Graph.*, vol. 26, no. 2, pp. 65–70, Aug. 1992.
[15] H. Hoppe, T. DeRose, T. Duchmap, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Proc. SIGGRAPH*, 1993, pp. 19–26.
[16] C. M. Ma and M. Sonka, "A fully parallel 3D thinning algorithm and its applications," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 420–433, Nov. 1996.
[17] S. Teller and C. Sequin, "Visibility preprocessing for interactive walkthroughs," in *Proc. SIGGRAPH*, vol. 25, no. 4, pp. 61–69, 1991.
[18] T.-Y. Lee, T.-L. Weng, and Y.-N. Sun, "Optimized semi-boundary (SB) rendering scheme," *J. Inform. Sci. Eng.*, to be published.
[19] R. Yagel and W. Ray, "Visbility computation for efficient walkthrough of complex environment," *Presence*, vol. 5, no. 1, pp. 45–60, 1995.
[20] O. Sudarsky and C. Gotsman, "Output-sensitive visibility algorithms for dynamic scenes with applications to virtual reality," in *Proc. EUROGRAPHICS'96*, 1996, pp. 294–258.
[21] N. Green, M. Kass, and G. Miller, "Hierarchical Z-buffer visibility," in *Proc. SIGGRAPH'93*, Aug. 1993, pp. 231–236.
[22] A. Fujimoto, T. Tanaka, and K. Iwata, "ARTS: Accelerated ray tracing systems," *IEEE Computer Graphics and Applications*, vol. 6, no. 4, pp. 16–26, 1986.
[23] H. Hoppe, "Progressive mesh," in *Proc. SIGGRAPH'96*, pp. 99–108.
[24] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Fifth Berkeley Symp. Math. Stat. and Prob.*, 1967, pp. 281–297.



Tong-Yee Lee was born in Tainan county, Taiwan, R.O.C., in 1966. He received the B.S. degree in computer engineering from Tatung Institute of Technology, Taipei, Taiwan, in 1988, the M.S. degree in computer engineering from National Taiwan University in 1990, and the Ph.D. degree in computer engineering from Washington State University (WSU), Pullman, in May 1995.

He is presently an Assistant Professor in the Department of Computer Science and Information Engineering at National Cheng-Kung University, Tainan, Taiwan. He was with WSU as a Visiting Research Professor at the School of Electrical Engineering/Computer Science during the summer of 1996. He has been working on parallel rendering and computer graphics since 1992 and has published more than 50 technical papers in referred journals and conferences. His current research interests include parallel rendering design, computer graphics, visualization, virtual reality, surgical simulation, distributed and collaborative virtual environments, parallel processing, and heterogeneous computing.

Ping-Hsien Lin was born in Taipei, Taiwan, R.O.C., in 1970. He received the B.S. degree in mechanical engineering and the M.S. degree in computer engineering from National Cheng-Kung University, Taiwan, in 1993 and 1998, respectively. He is currently working toward the Ph.D. degree at Department of Computer Science and Information Engineering, National Cheng-Kung University.

His research interests include computer graphics, image processing, virtual reality, visualization, and image-based rendering.

Chao-Hung Lin was born in Koushung, Taiwan, R.O.C., in 1973. He received the B.S. degree in computer science/engineering from Fu-Jen University and the M.S. degree in computer engineering from National Cheng-Kung University, Taiwan, in 1997 and 1998, respectively. He is currently working toward the Ph.D. degree at the Department of Computer Science and Information Engineering, National Cheng-Kung University.

His research interests include computer graphics, image processing, virtual reality, visualization, and interactive rendering.

Yung-Nien Sun received the B.S. degree from National Chiao Tung University, Hsin-chu, Taiwan, R.O.C., in 1978 and the M.S. and Ph.D. degrees from University of Pittsburgh, Pittsburgh, PA, in 1983 and 1987, respectively.

He was an Assistant Scientist with the Brookhaven National Laboratory, NY, from 1987 to 1989. Since 1989, he has joined the faculty of the Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan, as an Associate Professor and became a Professor in 1993. He has been working on image processing and computer vision since 1982 and has published more than 70 papers, half of them in referred journals. His current research interests are in medical and industrial applications of computer vision technologies.

Dr. Sun is a member of Sigma Xi, the Chinese Association of Image Processing and Pattern Recognition, and the Chinese Association of Biomedical Engineering.

Xi-Zhang Lin received the M.D. degree from the University of Taiwan and completed internal medicine resident training at Taipei Veteran General Hospital (1983–1986).

He completed a gastroenterology fellowship at Taiwan University Hospital and then joined National Cheng Kung University, Tainan, Taiwan (1988). He is the Chief of te Division of Gastroenterology at the National Cheng Kung University Hospital and an Associate Professor at the Medical College. His main research interests include medical imaging on hepatology and gastroenterology, especially in the quantitation of the information from medical imagings, such as imaging analysis of liver echotexture, three-dimensional reconstruction of the organs, and image-guided therapy.