1998 International Computer Symposium
Workshop on Computer Graphics and Virtual Reality
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

INTERACTIVE 3D VIRTUAL COLONOSCOPY SYSTEM*

Tong-Yee Lee[1], *Ping-Hsien Lin*[1], *Chao-Hung Lin*[1],
Yung-Nein Sun[1], *Xi-Zhang Lin*[2]

[1]Visual System Laboratory, Dept. of Computer Science and Information Engineering
National Cheng-Kung University, Tainan, Taiwan, R.O.C.
Email:tonylee@mail.ncku.edu.tw

[2]Department of Internal Medicine, National Cheng-Kung University Hospital
Tainan, Taiwan, R.O.C.

ABSTRACT

We describe a low-cost 3-D virtual colonoscopy system that is a non-invasive technique to examine the entire colon, and can assist the physicians in detecting polyps inside the colon. Using the helical CT data and proposed techniques, we can 3D 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 the automatic navigation, the flythrough path is determined ***a priori*** using the 3D 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. In comparison with the previous work, the proposed system can efficiently accomplish the required preprocessing tasks, and afford adequate rendering speeds on the low-cost PC system.

Keywords: Virtual Colonoscopy, Segmentation, Fly-through Path, Interactive Rendering, Potentially Visible Set

1. INTRODUCTION

Colon cancer is one of the most leading causes of cancer-related deaths per year in industrialized nations [1,2]. Most large-bowel malignancies arise from pre-exiting 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. In the optical colonoscopy, an optical probe is inserted into the colon and then is examining the colon interior structures. The physician can walk through inside the colon, and explore the regions of interest. However, there are several inherent drawbacks in this commonly used routine. To the contrary, 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 non-invasive, and, therefore, it does not require sedation or anesthesia to reduce patient's discomfort. Second, colonoscopy is possibly associated with a small of risk of perforation, and virtual colonoscopy can avoid this risk [3,4]. Third [4,5,6], 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 3D 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 3D reconstructed and visualized [7,8]. Using this scenario, it demonstrates many significant promises in several clinical applications such as surgery simulation and radiotherapy planning. Among them, the virtual colonoscopy and endoscopy are recently proposed applications that are still under development and investigation in several academic and medical institutes [9,10,11]. These researches were most conduct on the high-end graphics workstations [4,5,6] or expensive parallel architectures [12,13]. Therefore, one primary motivation of this paper is to develop a 3D virtual colonoscopy system on the 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 a very important factor to increase its popularity and to speed up its evolution.

The paper is organized as follows. Section 2 overviews relevant work. Section 3 describes our system components. Section 4 introduces methods on colon segmentation and Section 5 presents schemes to generate the flythrough path. Section 6 states our methods to achieve interactive rendering performance as well as to avoid navigating outside of the colon. Experimental results and discussions are presented in Section 7. Finally, some concluding remark and future work is given in Section 8.

2. PREVIOUS WORK

At Stony Brook, Professor Arie Kaufman [4,5,12] leads a team toward the virtual colonoscopy system. His team has substantial progress: automatically plan navigation, and achieve interactive rendering by a hardware-assisted visibility algorithm [5] or parallelization on the multi-computer architecture [12]. In [5], a potential field and

*This work is supported by NSC-88-2213-E-006-012,-Taiwan, ROC

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 colonic surface. However, the preprocessing stage is a terribly time-consuming process. For one data set, this process took a few hours on high-end SGI workstations. In their most recent work [12], a parallel version of software-assisted ray-casting scheme is exploited to visualize tissues beneath the colonic surface. In GE research group, Lorensen [6] et al develop 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 "Marching Cubes" algorithm [14]. To reduce the number of triangles generated and display rendering time, the triangle decimation algorithm [15] is used to eliminate the flat portion of 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 wave-front propagation algorithm is used [6]. Smooth flythrough animations along a planned camera trajectory is achieved through the key-frame technique. Roni Yagel et al [13] 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 the ESS. By far, for the related prototypes mentioned above, most of them were implemented either on high-end graphics workstations [4,5,6] or on multi-computer systems [12,13].

3. VIRTUAL COLONOSCOPY

In this section, we overview 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 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 in establishing such a system is lower than other proposed systems [4,5,6,12,13]. Figure 1 shows the process of the proposed system. The colon data is acquired by scanning 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 3D region growing method to automatically extract 3D colon volume from the original data. However, due to some technical difficulties in air inflation to the colon, in some place, there still exists ambiguity between the colon wall and lumen. In other words, there is no enough contrast between them, so it is difficult to segment the colon and the lumen using a simple threshold method. To remedy this problem, there is a 2D region growing sub-process allowing medical technicians to segment colon slice by slice manually.

After extracting the colon data, we 3D reconstruct the inner surface of it using the Marching Cubes algorithm. Our technique of automatically generating a flythrough animation consists of the following two steps. First, we use a 3D thinning technique to find the skeleton of the colon. The cross-sections of the colon have various sizes, so a 3D 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 centerline. Then, for each region, we compute its corresponding PVS (Potentially Visible Set). During rendering, only triangles in the current PVS will be rendered. Rendering performance can be further improved using LOD technique. For more details, see our companion paper "Interactive Walkthrough of Large Colonic Database"[26]. In this manner, we can achieve adequately interactive rendering on the proposed system.

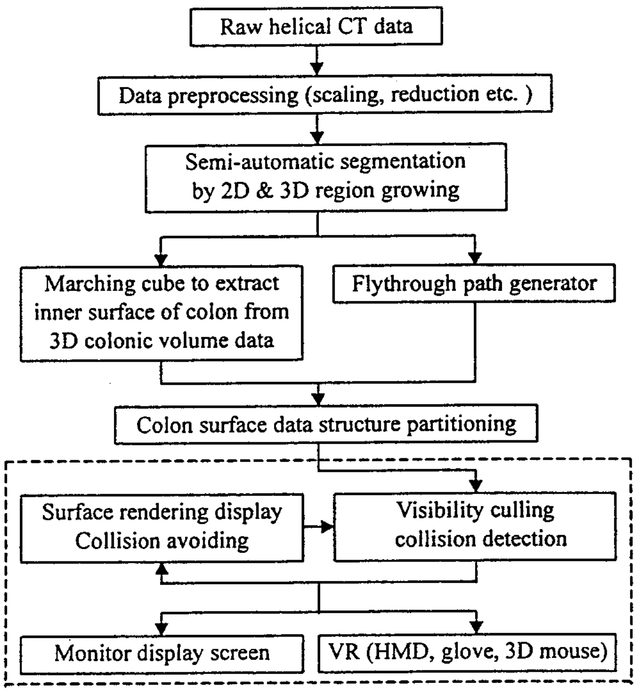
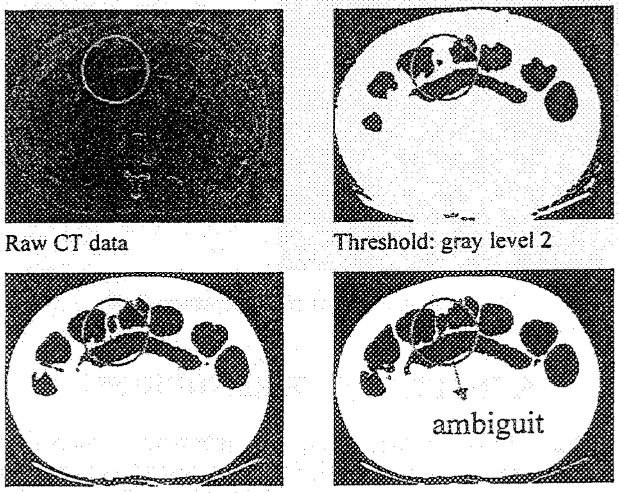

Raw helical CT data
Data preprocessing (scaling, reduction etc.)
Semi-automatic segmentation by 2D & 3D region growing
Marching cube to extract inner surface of colon from 3D colonic volume data
Flythrough path generator
Colon surface data structure partitioning
Surface rendering display Collision avoiding
Visibility culling collision detection
Monitor display screen
VR (HMD, glove, 3D mouse)

Figure 1. The process of interactive virtual colonoscopy system

To avoid viewer's eyes outside the colon (i.e., will abruptly deflect examiner's attention), we compute collision detection during navigation. For this purpose, a chain of bounding cylinders are used to constraint the movement of 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 inter-actively flythrough the colon by manipulating the viewing parameters employing a 3D 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.

4. COLON SEGMENTATION

At this stage, we exploit 3D region growing method to extract colonic volume from the original CT data. Three-dimensional region growing is a procedure that groups voxels or sub-volume into a larger volume. Our approach belongs to the voxel aggregation, which starts with a set of "seed" points and from these grows 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: use a threshold to segment colon data. Figure 2 shows the segmentation results using different thresholds. From this figure, because of poor contrast between colon wall and lumen, we can not successfully extract colon through a simple threshold method. Therefore, we might obtain only a small portion of colon (Figure 3(a)) or generate the mixture of the colon and lumen (Figure 3(b)).



Raw CT data   Threshold: gray level 2

Threshold: gray level 32   Threshold: gray level 50

Figure 2. The segmented results using different thresholds

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, V [0..N-1. 0..N-1, 0..N-1], and a convolution template with size $3 \times 3 \times 3$ (see Figure 4), T [-1..1, -1..1, -1..1]. We use two selected thresholds, p and q, where $0 < p < q$. We then use ***growth*** procedure to determine if a voxel V [i, j, k] will grow or not.

```
procedure growth ( )
 begin
  if  $[i,j,k] \le p$,  $[i,j,k]$ will grow;
  else  if  $[i,j,k] \ge q$,  $[i,j,k]$ will not grow;
  else {
   Compute $c = \sum_{u=-1}\sum_{v=-1}\sum_{w=-1} V[i+u, j+v, k+w] \times T[u,v,w]$,
   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 $3 \times 3 \times 3$ convolution (c) of $[i,j,k]$ if its intensity is between p and q. If c is positive, implies that most of 26-neighbors are colon, so it has a high potential to be as colon, too. Therefore, $[i,j,k]$ will grow. Figure 5 shows extracted 3D colonic volume after segmentation process. In case that the gray levels of the original data are too ambiguous to be automatically segmented by using the proposed method, we also provide a 2D region growing dialog box allowing technicians to segment 2D image slice by slice in an interactive manner. Furthermore, we can view currently segmented results using a 3D rendering dialog box simultaneously to check its correctness. The 3D rendering is accomplished by an optimum SB (semi-boundary) rendering scheme [19]. A snapshot of our segmentation GUI is shown in Figure 6. After the segmentation, we exploit "Marching Cubes" algorithm to obtain the colon interior structures.

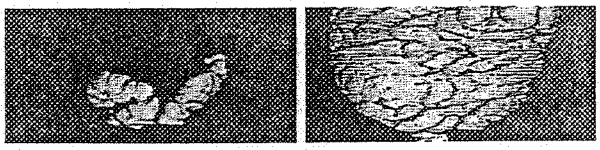


Fig. 3(a) Incomplete colon as a small threshold is used   Fig. 3(b) The mixture of colon and lumen as a large threshold is used

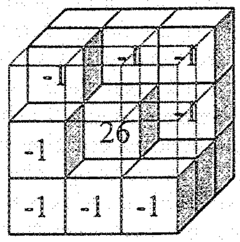


Figure 4. shows a 3 x 3 x 3 convolution mask

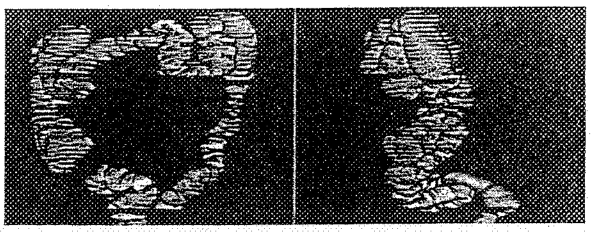


Figure 5. The extracted colon after the proposed segmentation process

5. AUTOMATICALLY-GENERATED FLIGHT PATH

A skeleton or centerline of 3D shapes is an efficient and compact shape descriptor, and it can give useful cues for a number of applications in visualization; these include automatic navigation, compression, shape description and abstraction, and tracking. In our case, we use a skeleton of 3D colon in the automatic navigation control, where it guides the virtual scopy to walkthrough the sinuous colon structures. Arie suggested a "peel-onion" technique [4] (i.e., a 3D thinning method) to extract the centerline of colon. The outermost layer of sub-volume 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", it must be careful in voxel deletion to avoid breaking continuity of the desired path. In case that the cross-sections of colon are ideally uniform as shown in [4,5], there is no problem in this technique. We should note that the shapes of colon could become uniform when patients have disease of ileus or take medicine of buscopam prior to CT scanning. However, physicians would not suggest that patient should take such medicine. Therefore, in most clinical cases, the colon will have various sizes in its cross-sections as shown in Figure 7(a). In such circumstance, the "peel-onion" scheme will potentially lead to many branches according to its rule of preserving continuity (Figure 7(a) and (b)). To trim these redundant branches, we employ a simple two-pass tracking method. Before trimming branches, the two-end points (starting and ending fly points) of flight path are specified. At the first pass, we begin with labeling each voxel in an incremental manner from the starting fly point. The result (2D view) is shown in Figure 8(a). In this example, we increase the label of a neighboring pixel by one, and a diagonal neighboring pixel by two, respectively. There are several branches in this example. At the second pass, we start walking back from the ending point in a decreasing manner (as shown in Figure 8(b)). 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. Figure 9(a) and (b) show the results of "peel-onion". Many redundant branches are visible in this figure. We use a two-pass tracking scheme to generate a unique path as shown in Figure 9(c). As suggested in [4], we also employ cubic splines to smooth our generated path. Finally, we outline the "peel-onion" employed in this paper as follows.

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

For a $N \times N \times N$ volume data, $V[0..N\text{-}1, 0..N\text{-}1, 0..N\text{-}1]$, 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 $1 \le |p_x-q_x|+|p_y-q_y|+|p_z-q_z| \le 2$ 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 3D region growing, each voxel of V is classified as either a colon point or a background point. During "peel-off", we delete all *simple* points layer by layer starting from the outermost layer. A colon point, p, is said to be *simple* if it satisfies the following rules [17]:

- 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.

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

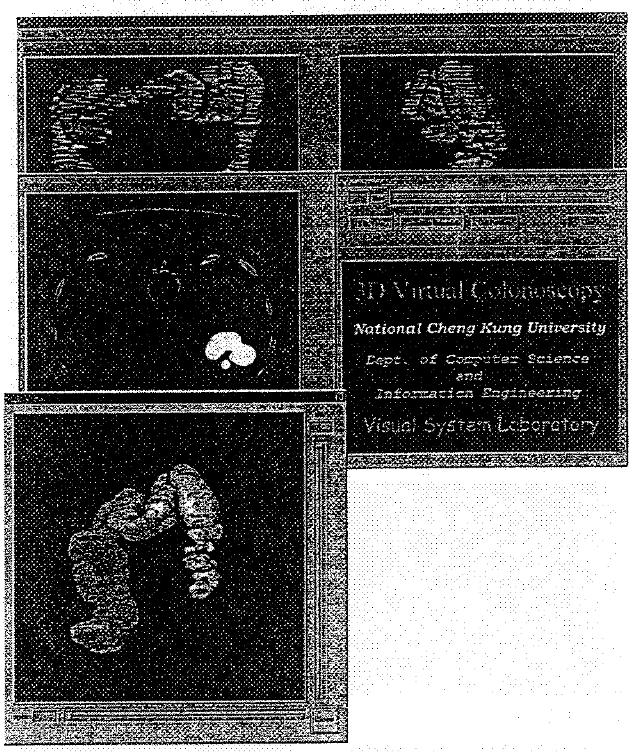

3D Virtual Colonoscopy
National Cheng Kung University
Dept. of Computer Science and Information Engineering
Visual System Laboratory

Figure 6. A snapshot of segmentation GUI

6. INTERACTIVE RENDERING

For virtual colonoscopy, an interactive rendering performance is very crucial. On the low-end system like 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 [18,19,20,21,22]. One exploits a hierarchical Z-buffer to efficiently cull unseen objects [22]. The other approach in [18,20] divides the whole scene into many cells. Each cell has a predetermined potentially visible set (PVS). During rendering time, the current PVS is exploited to cull unseen cells; therefore, it can reduce the number of triangles submitted to graphics pipeline. Arie's approach [5] computes PVS on the fly during rendering time as well as proposes a hardware-assisted visibility algorithm called 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 has neither enough computing power nor 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 pre-computation in which set of cells and surface triangles visible from each cell are computed. The results of these pre-computations are stored in the display database for use during navigation.

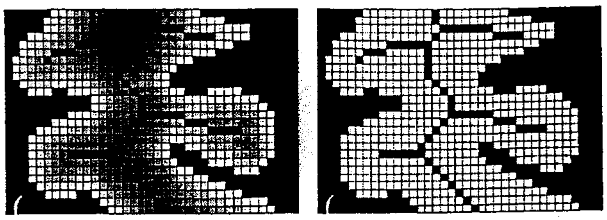


Figure 7 (a) and (b) show cross-sections with various sizes. In this figure (a), the skeleton of colon is obtained by "peel-onion" method (each layer is represented by a different color) but it has some branches along the skeleton. Figure (b) shows the centerline of the colon.

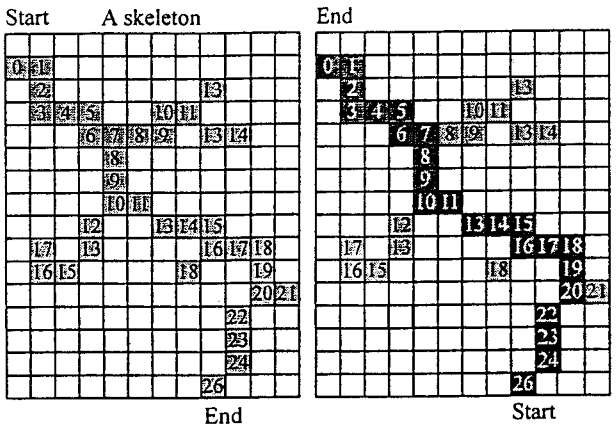


Figure 8(a) the first pass in two-pass tracking

Figure 8(b) the second pass in two-pass tracking

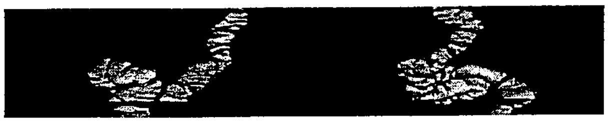


Figure 9 (a) shows two orthographic views of the colon before the "peel-onion" process

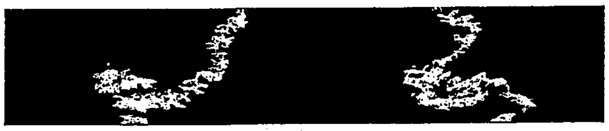


Figure 9 (b) shows two orthographic views of the colon with branches after the "peel-onion" process

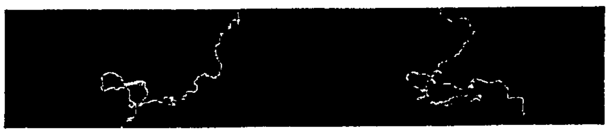


Figure 9 (c) shows the centerline after we trim the branches using two-pass tracking

6.1 Region Partitioning

The colon structure is partitioned into many cells. For the tube-like colon structures, a good partition strategy must take the curvature of the centerline into account. As shown in Figure 10, we can cut the colon structure into three parts at points C_1 and C_2, respectively. In such circumstance, triangles at S_3 will be very likely to be invisible to the triangles located at 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, $P_1, P_2, \ldots, P_N$, where P_1 and P_N are two end points. Initially, we connect the straight line between P_1 and P_N, and termed as L. We then use ***cell_partition*** procedure to recursively partition the colon structures described below.

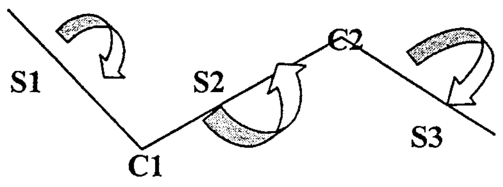


Figure 10. shows that cutting is always made on the points with a large curvature

```
procedure cell_partition ( )
  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
```

Figure 11 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.

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 sub-regions. The second criterion is to check if the width of the cross-section (of colon) varies significantly beyond some threshold in each region. If it does, we will re-partition this region into several smaller sub-regions, too. For each sub-region, we compute its PVS and also use a cylinder within this sub-region 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 sub-region. If collision occurs, the virtual camera will be forced to move back to be inside the bound volume, and, therefore, the camera viewing is always kept inside the colon. The purpose of this design is two folds. First, we are only interested in the interior regions of the colon and use it to avoid penetrating through colon. Second, it avoids passing all surface triangles of the colon to the graphics pipeline (i.e., save rendering time).

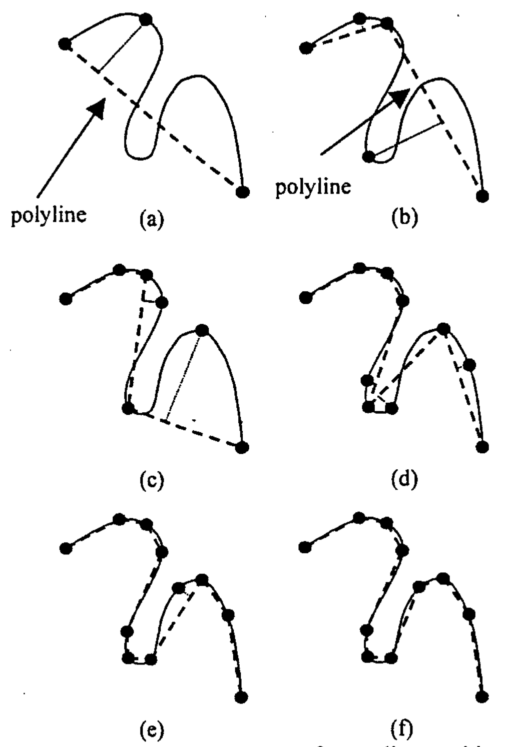


Figure 11. shows the process of centerline partitioning

6.2 Potentially Visible Set Determination

Once the colon partition has been accomplished, we perform a potentially visible set pre-computation in which the portion of colon structures visible from each cell is determined. Figure 12 illustrates our method to compute 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 non-immediate neighbors like *H*, we must determine if any point of *H* can be reached by a *sight-line* that originates inside the 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 boundary before hitting *H*'s rear door). Therefore, the cell *H* is visible to *F* and will be included in *F*'s PVS. However, if there is no visible ray (such as ray_i) found, then the cell *H* will not be included. In case that cell *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 util the end of the colon or an invisible cell found. For each ray traversal, it is efficiently performed voxel by voxel using a 3DDDA stepping algorithm [23]. In our implementation, a double-linked list is maintained for fetching cell's PVS. As 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 Figure 13).

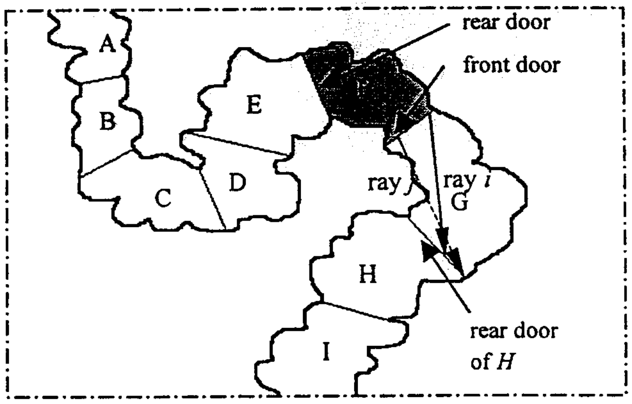
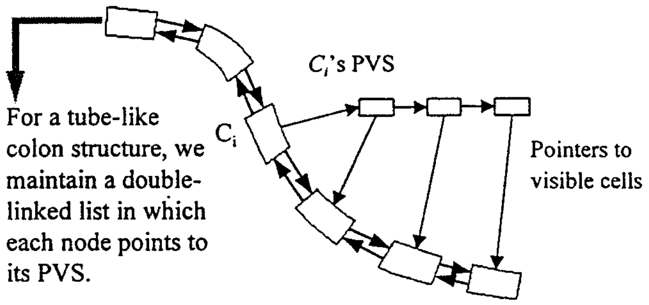


Figure 12. shows the computation of PVS for each cell


C_i's PVS
For a tube-like colon structure, we maintain a double-linked list in which each node points to its PVS.
C_i
Pointers to visible cells
Figure 13. shows the double-linked list used to maintain PVS.

7. EXPERIMENTAL RESULT AND DISCUSSION

To perform our experiments, we collaborate with the physicians at the Hospital of National Cheng-Kung University. The size of test colon CT data is 512 x 512 x 200[2]. Our initial experiments with our system have been tested against several patient data sets. The encouraging results to discover polyps are verified by the collaborating physicians. Figure 14, shows the user interface of navigation environment. In this environment, we provide a colonoscopy view and two colon's orthographic views (*xz* and *yz* directions). As the camera moves along the colon, one red spot (represents the current location of camera) will also move in these two orthographic views of 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 Figure 15). 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 the

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

average, there are 47944 (48 K) triangles contained 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.

Since our work is closely related to [5], detailed comparison is listed as follows.

- [5] was implemented on the SGI R10000 processor with InfiniteReality and ours was on a dual-CPU Pentium 2 PC server system. Both exploited region growing method to perform colon segmentation. However, no detailed information provided in [4] and [5,12]. In this paper, we propose schemes to solve ambiguity between colon and lumen. Additionally, two dimensional region growing and 3D rendering dialog boxes are provided to assist in performing segmentation. For our test data, it took 3-4 minutes to accomplish this task. Both used "Marching Cube" algorithm to generate the inner surface of the colon.
- For the centerline extraction, [5] used a single-source shortest path algorithm and we first exploit 3D thinning (same as their early work [4]) to extract the skeleton of the colon and then use a two-pass tracking scheme to trim branches. In our case, the proposed scheme took 10 minutes. To the contrary, [4,5], segmentation and centerline extraction took several hours.
- For interactive rendering, [5] partitioned colon into cells based on centerline in the preprocessing stage, and exploited a hardware-assisted visibility algorithm to determine cell's visibility during rendering. The distance of centerline and the number of triangles are criteria during partitioning. On the other hand, we exploited an approximating polyline method to partition colon. The curvature of 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 in the preprocessing time and was calculated using an efficient ray traversal scheme between two cross-sections. On the average, [5] the number of triangles per frame ranges from 146 K to 53 K for three data sets and our reported number is about 47K. Both systems achieve adequate interactive speeds for image size 512 x 512.
- In both systems, there are two types of navigation: automatic (through centerline) and interactive modes, respectively. [5] presented a physically-based camera control model to control the movement of camera. This interesting scheme computed potential field to achieve collision detection, while it is expensive in computation. To be affordable on PC system, we simply achieved collision detection using a chain of cylinders as bounding volume. Although it will expense some flexibility of camera movement, it is very fast.

8. CONCLUSION AND FUTURE WORK

The virtual colonoscopy is an emerging non-invasive technique to detect polyps. It eliminates patient's

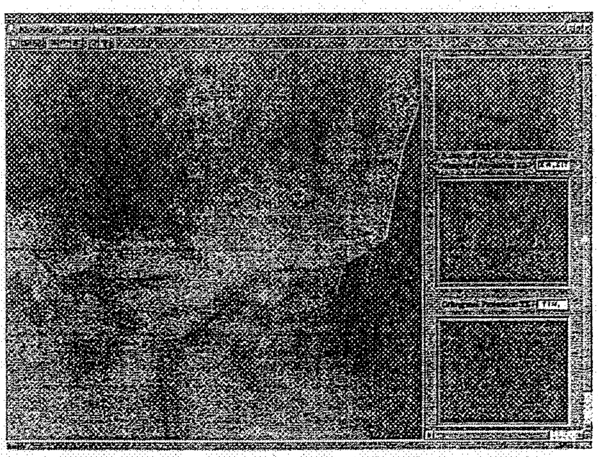


Figure 14 shows the user interface of navigation environment

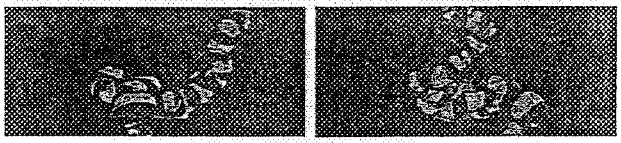
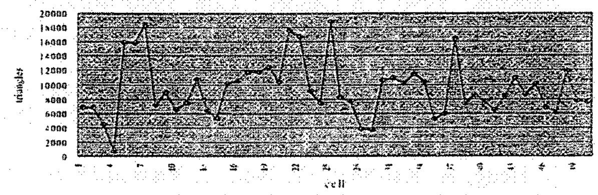
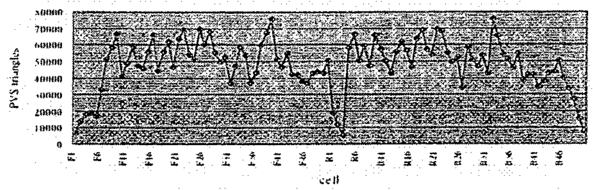


Figure 15. shows the colon with 50 cell partitions



Plot 1. shows the number of triangles contained in each cell



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

discomfort caused in the conventional colonoscopy surgery. In this work, we present a PC-based colonoscopy system. Compared with other related work [5], our work is attractive due to its low cost, while the presented system has similar capabilities and performance. The primary contributions in this work are summarized as follows. We proposed a colon segmentation scheme that has a solution to the ambiguity existing between colon and lumen. A 3D thinning and two-pass tracking schemes are exploited to generate a centerline of the colon. To achieve interactive rendering, the approximating polyline partitioning and visibility pre-computation was performed in the preprocessing. With these schemes, a small portion (10%) of the total triangles is passed to the graphics pipeline. Additionally, a simple collision detection is exploited to avoid penetrating through colon surface. In general, the preprocessing stages takes less than one hour on our PC system. It includes generating surface, extracting centerline, partitioning cells and determining PVS, and forming a chain of cylinders as bounding volume in each cell. In contrast, [4,5] took a few hours to accomplish their preprocessing stage.

There are several extensions that can be done in near future. First, we plan to design a cost-effective camera model to navigate the colon in a realistic manner like [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. So, second, we are collaborating with physicians to correctly visualize the surface appearance with combination of other image modalities such as ultrasonic image. 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 fiber scopy.

REFERENCES

[1] Silverberg E., Boring CE, Squires TS, "Cancer statistics," 1990, CA Cancer J. Clin. 1990: 40:9-26

[2] Eddy DM, "Screening for colorectal cancer," Ann. Intern Med. 1990: 113:373-384.

[3] H. M. Fenlon and J.T. Ferrucci, "Virtual Colonoscopy : What Will be Issues Be," AJR: 169, Aug. 1997, 453-458

[4] L. Hong, A. Kaufman, Y. Wei, A. Viswambharn, M. Wax, and Z. Liang, "3D Virtual Colonoscopy," Proc. 1995 Symposium on Biomedical Viusualization, 1995, 26-32.

[5] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, Virtual Voyage: Interactive Navigation in the Human Colon, Proc. ACM SIGGRAPH '97, 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, ISO Press, WA D.C., 1995, 221-230

[7] M. Levoy, "Display of Surfaces from Volume Data," IEEE Computer Graphics and Applications, Vol. 8, No. 5, May 1988, 29-37.

[8] A. Kaufman, Volume Visualization, IEEE Computer Society Press, Los Alamitos, CA, 1991

[9] 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/

[10] The Volume Graphics Research Group in the Department of Computer and Information Science in The Ohio State University, http://www.cis.ohio-state.edu/volviz/.

[11] GE Research and Development, Computer Graphics and System Program, http://www.crd.ge.com/esl/cgsp/.

[12] 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, Proc. IEEE Visualization '97, pp. 433-436.

[13] G.J. Wiet, R. Yagel, D. Stredney, P. Schmalbrock, D.J. Sessanna, Y. Kurzion, L. Rosenberg, M. Levin, and K. Martin, " Volumetric Approach to Virtual Simulation of Functional Endoscopic Sinus Surgery" Medicine Meets Virtual Reality 5, San Diego, CA, January 1997.

[14] Lorensen, W. E. and Cline, H. E., "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm," Computer Graphics, Vol. 21, No. 3, pp. 163-169, July 1987

[15] Schroeder, W. J., Zarge, J., and Lorensen, W. E., "Decimation of Triangle Meshes," Computer Graphics, Vol. 26, no. 2, pp.65-70, Aug. 1992.

[16] H. Hoppe, T. DeRose, T. Duchmap, J. McDonald, W. Stuetzle, "Mesh Optimization," Computer Graphics (Proceedings of SIGGRAPH), pp. 19-26, 1993.

[17] C. Min Ma and Milan Sonka, "A Fully Parallel 3D Thinning Algorithm and its Applications," Computer Vision and Image Understanding, Vol. 64, No. 3, Nov., pp. 420-433, 1996

[18] S. Teller, C. Sequin, "Visibility Preprocessing For Interactive Walkthroughs," Computer Graphics (Proceedings SIGGRAPH), Vol. 25, No. 4, pp. 61-69 , 1991

[19] S. Coorge, S. Teller, "A Spatially and Temporally Coherent Object Space Visibility Algorithm," Tech. Report TM-546, Lab. For Computer Science, MIT, 1996.

[20] Yagel R., Ray W., "Visbility Computation for Efficient Walkthrough of Complex Environment," Presence, Vol. 5, No. 1, pp. 45-60, 1995.

[21] O. Sudarsky, C. Gotsman, "Output-sensitive Visibility Algorithms for Dynamic Scenes with Applications to Virtual Reality," Proceedings of EUROGRAPHICS'96, pp. 294-258, 1996.

[22] N. Green, M. Kass and G. Miller, "Hierarchical Z-buffer Visibility," SIGGRAPH'93 Conference Proceedings, pp. 231-236, Aug. 1993.

[23] Fujimoto A., Tanaka T. and Iwata K., "ARTS: Accelerated ray tracing systems," IEEE Computer Graphics and Applications, 6(4), pp. 16-26, 1986

[24] H. Hoppe, "Progressive mesh," Proceedings of SIGGRAPH'96, pp. 99-108.

[25] MacQueen, J., "Some methods for classification and analysis of multivariate observations," in Proc. Fifth Berkeley Symposium on Math. Stat. And Prob., pp. 281-297, 1967.