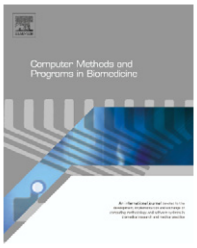
journal homepage: www.intl.elsevierhealth.com/journals/cmpb



An extension to 3D topological thinning method based on LUT for colon centerline extraction

M. Ding[*], *Ruof. Tong*, *Sheng-hui Liao*, *JinX. Dong*

State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, PR China

ARTICLE INFO

Article history:
Received 23 May 2007
Received in revised form
28 January 2008
Accepted 6 October 2008

Keywords:
3D topological thinning
Centerline extraction
LUT
Region growing
Virtual colonoscopy

ABSTRACT

Topological thinning is a valid but time-consuming method to calculate the centerline of human colon or other hollow organs accurately. An optimized 3D topological thinning method based on Look-up Table (LUT), which was proposed by Sadlier, proves to be effective in improving the efficiency on many occasions. However, it is still inefficient when processing some complex datasets. In this paper, we first analyze the reason causing the unstable performance, and then present an extension to Sadlier's method, which enables the rapid execution of the extraneous loops removing by avoiding unnecessary global connectivity testing. To reach this purpose, a min-heap structure is introduced to select a seed from the candidate voxels set of the final centerline, and region growing technique is used to find the voxels in the same branch with the seed. The comparison among the standard topological thinning, LUT method and the extension to LUT method indicates the extension achieves the most efficient performance.

1. Introduction

Accompanied with the rapid development of the computer and medical photograph technology, improvements in high-resolution computed tomography (CT) and MR acquisition together with advances in 3D reconstruction provided a large clinical potential for virtual endoscopy in the evaluation of any hollow organ system.

Virtual colonoscopy [1–3] is a type of interactive 3D medical imaging tool which combines the features of endoscopic viewing and cross-sectional volumetric imaging. It processes the 3D image data sets from CT or MRI scans using lots of computer methods and rebuilds the 3D structure of the scanned body as well. Thus doctors can operate the rebuilt 3D structure through moving, rotating or zooming (called navigation) to observe the internal structure of the organ. Since it is a non-invasion method, it can avoid the serious side effects such as perforation, infection and hemorrhage caused by real endoscopy.

However, manual navigation through a virtual reality model of colon is a very slow and tedious process. Accordingly people prefer an automated navigation which is performed as follow steps: First, a centerline is generated; Then this centerline can subsequently be used to guide the observer (or virtual camera) through the colon lumen and generate fly-through images of the inside colon.

In this paper, an extension of the LUT method given by Sadleir is presented. First, we summarize the approach to centerline extraction using Look-up Table (LUT) presented by Sadleir. Then we analyze the defect of the algorithm with LUT and describe our improvement as well as the data structure. We reveal implementation and results in Section 5 and finally conclude this paper with the result comparison among the standard way, LUT method and our extension in Section 6.

[*] *Corresponding author*. Tel.: +86 571 87934736.

0169-2607/$ – see front matter © 2008 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2008.10.002

2. Background

2.1. Centerline extraction

A centerline extraction algorithm is expected to calculate an approximation of the center path of the colon accurately in a reasonable time. Time constraint is a very important factor to evaluate the algorithm especially in a clinical practice.

Early centerline extraction algorithms were based on a technique called onion peeling [4] or topological thinning. In this method, the surface points of colon are peeled repeatedly until the centerline is obtained. Though the results from this standard algorithm are accurate, it is extremely inefficient. Therefore, researchers recently worked out other methods such as distance transform, minimum energy path etc.

Hassouna and Farag [19] presented a novel framework for computing centerlines for both 2D and 3D shape analysis. In their method, centerline is considered a point source (PS) transmitting a wave front that evolves over time and traverses the object domain. The front propagates at each object point with a speed proportional to its Euclidean distance from the boundary. The motion of the front is governed by a nonlinear partial differential equation whose solution can be computed efficiently using level set methods.

Van Uitert and Bitter [20] presented an automatic algorithm for computing subvoxel precise skeletons of volumetric data by using subvoxel precise distance fields as input and fast marching method to extract the skeleton. It can obtain the skeletons of those objects which are less than a single voxel thick.

There is a brief summary in Table 1, in which we classify centerline extraction methods into three categories. In each category, several representative works are listed.

2.2. Review on LUT method

Sadleir and Whelan [13] provide an optimized version of topological thinning. Considering the deletion of voxels is just a local process, they use Look-Up Table to register whether a voxel with respect to a neighborhood configuration can be deleted. LUT is established before the extraction of centerline and once established, it can be used for any dataset. When testing a voxel in topological thinning, an index of this voxel is generated first. Whether this voxel should be deleted can be determined immediately after querying the LUT. It can successfully avoid testing the local connectivity of every voxel so that the performance will be improved greatly.

For a $3 \times 3 \times 3$ space, the number of possible neighborhood configurations in LUT will be 2^{26}. Each of the configurations has a unique index I, which can be generated as Eq. (1):

$$I = \sum_{n=0}^{25} 2^n V_n \qquad (1)$$

where every bit of this integer V_n represents whether the corresponding voxel is a background one or otherwise. An example of index generation is showed in Fig. 1.

The value referenced by each index is assigned as 1 if this voxel can be deleted, which means that deleting this voxel does not break the local connectivity or introduce a new hole, and 0 otherwise. Fig. 2a shows the case that the connectivity is broken and Fig. 2b shows the case that a new hole is introduced by the deletion.

After reading the segmented result, the centerline extraction algorithm will firstly scan the solid object to specify the surface voxels and insert them into a vector. Then to each element in this vector, the algorithm generates the index using the method described above and checks the returned value from the Look-up Table referenced by this index. If the value is “1”, this element is deleted from the vector. This procedure is repeated until there are no more elements that can be deleted in this vector.

Unfortunately, now the elements in the vector are not the final centerline because it may have some extraneous loops due to the holes in the original segmented colon lumen, which are a common phenomenon associated with the Hasutral

Table 1 – Centerline extraction algorithms published previously.

Category	Authors	Year	Method
Euclidean distance coding/transform	Zhou et al. [5]	1998	Using a distance from surface field to extract skeleton
	Zhou and Toga [6]	1999	Using a distance from source field with cluster centers to find centerline points
	Bitter et al. [7–9]	2001	Using Dijkstra’s shortest path algorithm on a graph built with a combination of distance from a source node and distance from the boundary
Boundary peeling/erosion	Ge et al. [11,12]	1999	3D topological thinning and using graph search algorithm to remove extra loops and branches
	Sadleir et al. [13,14]	2005	Optimized 3D topological thinning using Look-up Table
Hybrid methods	Bouix et al. [15]	2005	Using modified the average outward flux based medial surfaces algorithm to extract the center path
	Deschamps and Cohen [16]	2001	Finding paths of least action in 3D intensity images

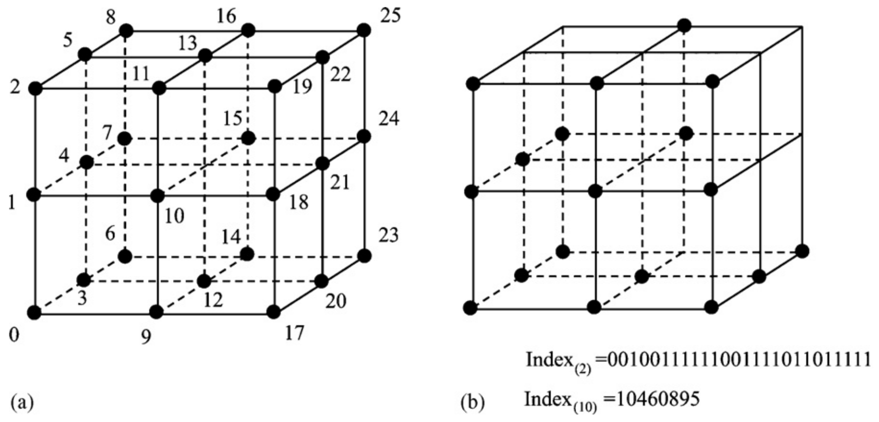


Fig. 1 – Index generation. (a) The order of 26 neighbor voxels. (b) An example of index generation.

folds [14]. Extraneous loops can be removed by testing whether there is global connectivity violation when deleting a centerline voxel closest to the surface. A voxel should not be removed if it causes global connectivity violation.

After removing the extraneous loop, the remaining elements in the vector compose the final centerline.

3. Optimization

3.1. Analysis

From the above discussion, we know that the running time of centerline extraction using topological thinning, ET_{Total}, can be expressed as

$$ET_{Total} = T_{Deletion} + T_{Refine},$$

where $T_{Deletion}$ represents the time consumed in deletion process and T_{Refine} represents time consumed in removing the extraneous loops. Clearly, $T_{Deletion}$ is a linear function of the data size n and the LUT reduces $T_{Deletion}$ by avoiding the local connectivity testing for every voxel in a segmented volumetric data set. However, T_{Refine} is not a linear function but a quadratic function of m which is the number of voxels in the vector after thinning process, because this process uses Deep First Search to perform the global connectivity test. It means that, T_{Refine} will increase dramatically if m cannot be reduced to a relative small scale.

Our experimental results confirm the above analysis. For those data sets with about 3–5 millions voxels, the ET_{Total} is no more than 20 s. However, when the scale of data set is about 10–15 Mega, the ET_{Total} increases to about 120 s, which is mainly caused by the increase of T_{Refine} from 5 s to 70 s. Actually, m is not only decided by n, but also depends on the complexity of the volumetric data, which means m may be large even though n is small.

To improve the performance, T_{Refine} must be reduced. There are two problems we should focus on. The first one is how to choose a voxel which is closest to the surface. The second is how to avoid global connectivity testing for each voxel in the candidate set.

3.2. Minimum heap

For the first problem, we introduce a minimum heap structure. It records the approximate distance of a particular voxel to the original surface. This distance can be simply calculated by assigning a relevant thinning order to each voxel as it is exposed in the thinning iteration. Using this minimum heap, a voxel which is closest to the surface can be easily chosen from the candidate set, because the top element of the heap is just the one which has minimum distance. After being selected, it is removed from the heap and the necessary adjustment will be performed to maintain the heap property.

3.3. Region growing

It is a little more complex to solve the second problem. Notice that, after the topological thinning process, all the voxels in the extraneous loop except its two endpoints should have only two connected neighbors. And to find these two neighbors is easy. It means that, if a voxel belonging to an extraneous loop is selected, its two connected neighbors which should be deleted as well can be obtained by a simple local $3 \times 3 \times 3$

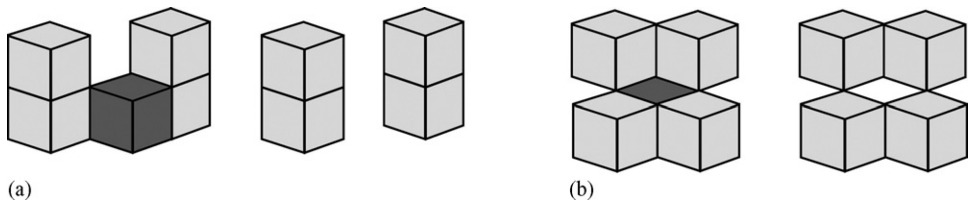


Fig. 2 – A local 3 × 3 × 3 test. (a) Connectivity will be broken when delete the center voxel. (b) The removal of a surface voxel introduces a new hole.

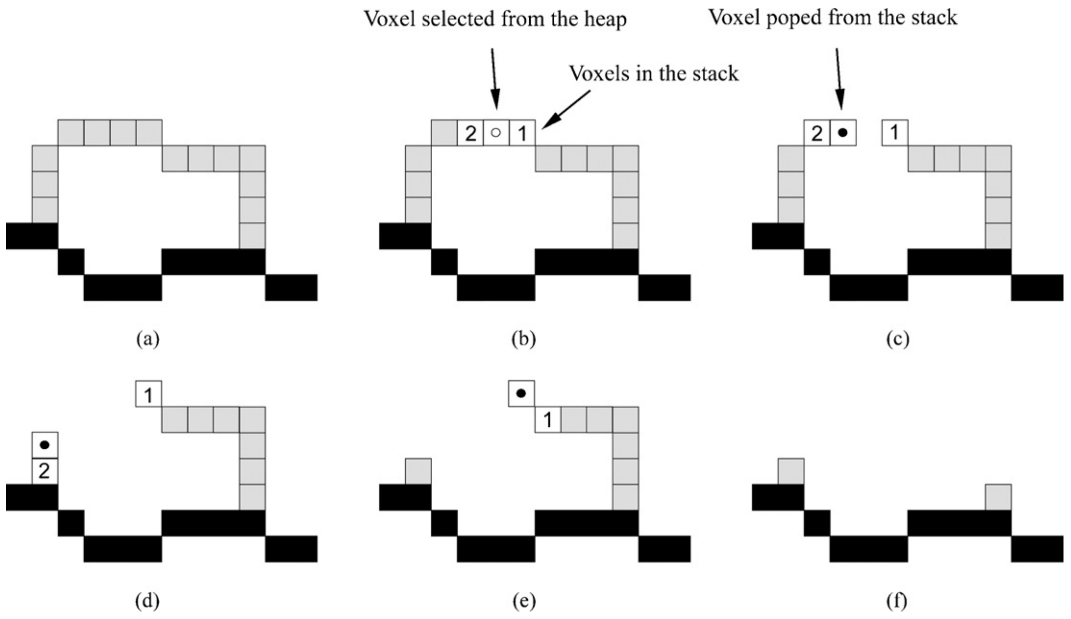


Fig. 3 – The operation of region growing when selected voxel does not belong to the final centerline. (a) The map of current vector after topological thinning. (b) Voxel selected and its two neighbors. The region growing will be performed. The voxels belonging to this loop will be deleted in the order simulated by (c–e). (f) The result after removing this extraneous loop. Black ones represent the centerline which has not been tested; while lightly gray ones represent the extraneous loop.

test. And so, global connectivity test on these two connected neighbors is unnecessary.

The process of extraneous loops removing begins after topological thinning. The procedure first gets the top element of the minimum heap, and test whether the global connectivity will be violated without it.

If the global connectivity is still maintained, it implies that the tested voxel belongs to an extraneous loop, thus it is marked as tested and deleted from the vector. Then we push its two neighbors into a stack and perform region growing procedure: a voxel will be popped from the stack first, and the procedure will check that whether this voxel has only one neighbor. If so, this neighbor should belong to the same extraneous loop also. We mark it as tested, delete it from the vector, and push its neighbor into the stack. Otherwise, the voxel must be one of the endpoints of this extraneous loop, so its

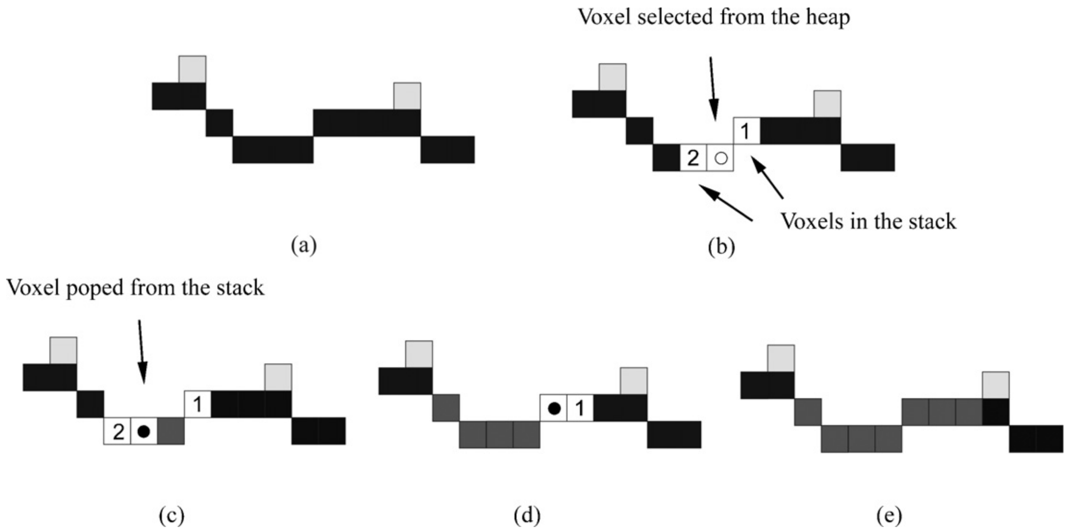


Fig. 4 – The operation of region growing when a voxel belongs to final centerline. (a) The map of current vector. (b) The selected voxel and its two neighbors. The region growing will be performed as well. The difference is that these voxels will not be deleted but marked as tested. Subparts (c–e) show the general process. Black ones represent the centerline which has not been tested yet, lightly gray ones represent the extraneous loop, and heavily gray ones represent those which belong to final centerline.

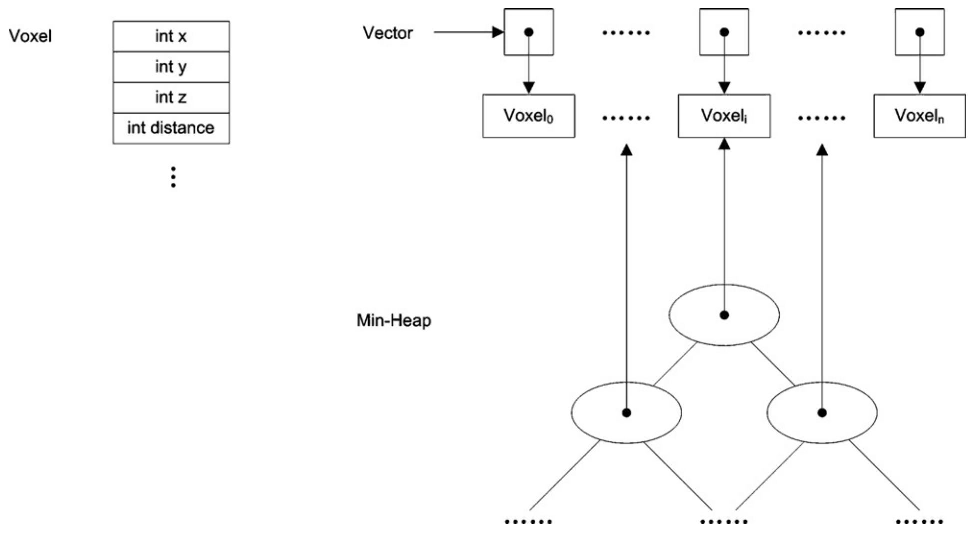


Fig. 5 – The data structure and its organization. Scattered memory spaces will be allocated for every voxel in the segmented data. The array "Vector", as well as the array "min_heap", is defined to save the addresses of these pieces of spaces.

neighbors should not be pushed into the stack. This process will be repeated until the stack is empty, which means one extraneous loop have been removed.

If the global connectivity is violated, it means that the tested voxel belongs to the final centerline. In this case, we just mark the voxel as tested but not delete it. After pushing the two neighbors of this voxel into the stack, the region growing will begin. The procedure will get the top element of the stack as well and check out whether this voxel has two neighbors. The difference from the first case here is that the voxel tested for global connectivity is not deleted. Thus, the popped voxel which has two neighbors, not one, also belongs to the final centerline. We mark it as tested and push its neighbor which has not been marked as tested into the stack. If the popped voxel has more than two neighbors, it must be an intersection where the final centerline crosses an extraneous loop, thus its neighbors should not be pushed into the stack. This process will be repeated until the stack is empty, which means a part of the final centerline has been determined.

When there are no more elements in the stack, an extraneous loop has been removed. The procedure goes back to the minimum heap to get its top element and perform the region growing again. If the minimum heap is empty, all the extraneous loops have been removed and those remaining elements in the vector are the voxels of the final centerline.

Figs. 3 and 4 illustrate the process of region growing. Formal description of the algorithm will be given in Section 4.

4. System description

4.1. *Data structures*

To implement our improved algorithm, some extra data structures are needed. We introduce a stack (or a queue) to store the neighbors of the deleted voxel. The structure of the stack can be simply defined to save the coordinates of a voxel only.

Besides, a new array Test[·][·][·] should be created to record whether a voxel specified by a triple (*x*, *y*, *z*) has been tested. This array is defined as bool type and initialed with FALSE. Fig. 5 shows the data structure and its organization in the algorithm. For convenience, the array "vector" and "min_heap" are defined as pointer type.

```
struct Voxel {

    int x;

    int y;

    int z;

    int distance;

};

Voxel* vector[n];

Voxel* min_heap[n];
```

4.2. *The algorithms*

The complete centerline extraction algorithm can be described as Table 2. Procedure *InitialSurfaceScan*() scans the volume data to find the surface voxels and puts them into vector. Procedure *GenerateIndex*() returns the index of a voxel as method mentioned in Section 2.2. Procedure *GetValue*() queries the LUT with the index and returns the result whether the voxel can be removed. The final centerline is smoothed by re-sampling and cubic spline fitting in procedure *Smooth*().

Table 3 shows how procedure *ExtraneousLoopRemove*() works. The procedure *establish _minimum_heap*() builds the minimum heap on the input array voxel[*n*]. The procedure *initialize_array*() initializes the array Test[·][·][·] with FALSE, and

Table 2 – Algorithm for centerline extraction.

```
Procedure CenterlineExtraction() begin
  Read the segmented raw volume data;
  InitialSurfaceScan();
  Repeat until vector is empty begin
    Index:=GenerateIndex(voxel);
    If GetValue(Index, LUT) is TRUE then
      delete_vector(voxel);
      Update the vector, add new surface voxels into vector;
      voxel_new.distance:=voxel_delete.distance+1;
    End If
  End repeat
  ExtraneousLoopRemove();
  Smooth();
End Procedure
```

initialize_stack() creates an empty stack. The procedure *heaptop*() returns the top element of the heap, deletes it from the heap and implements the necessary adjustment. The procedure *global_connectivity_test*() returns FALSE if there is no connectivity violation after deleting a voxel, or TRUE for yes.

4.3. *Status report*

We divide the whole program into several modules which are I/O, 3D reconstruction, centerline extraction and navigation. The functions we have implemented include image segmentation, 3D reconstruction using Marching Cube, Centerline Extraction using our method as well as standard topological thinning algorithm and automated navigation. Fig. 6 shows the interface of virtual colonoscopy.

All algorithms are implemented using the C++, Visualization Toolkit 4.0 which is an open source, object-oriented software system for computer graphics, visualization and image processing.

5. Results and discussion

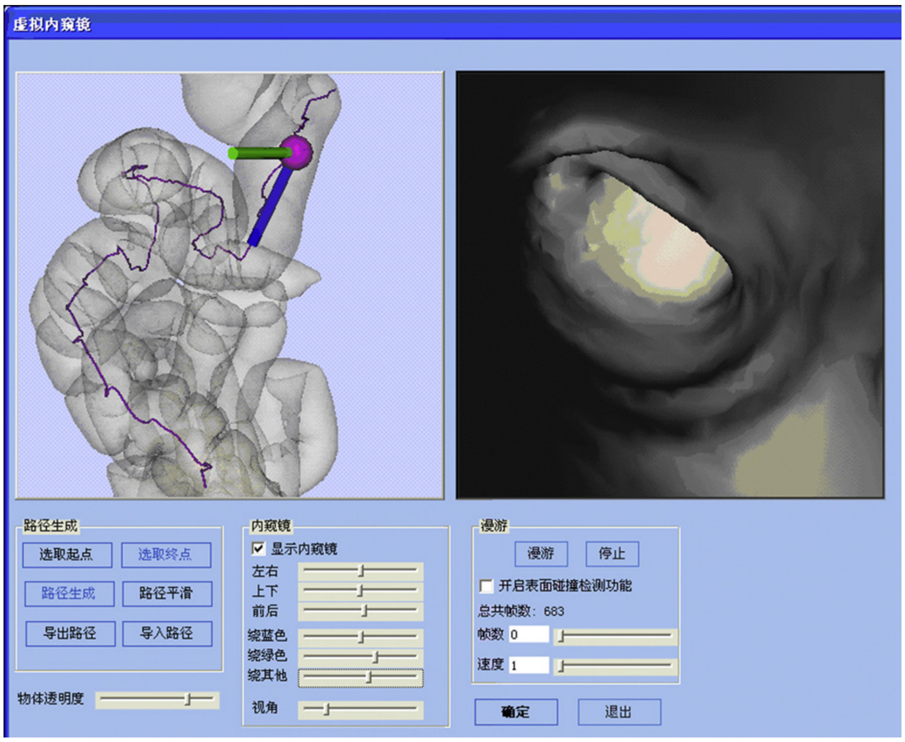
5.1. *Data acquisition and segmentation*

All CT scans were obtained from Sir Run Run Shaw Hospital, Hangzhou and Navy General Hospital, Beijing. The number of slices varies from 150 to 250 and each slice comprised of 512×512 pixels. Typical total size of the volumetric data is approximately 120 Mb.

We use an improved method which combined region growing algorithm [17,18] and double-threshold algorithm to segment raw volume data. Segmentation results of colon lumen are in a binary model, where “0” means background voxel, “1” means foreground voxel. The resulting solid binary object is the input of centerline extraction algorithm.

Table 3 – Algorithm for removing extraneous loops.

```
Procedure ExtraneousLoopRemove begin
  establish_minimum_heap(vector);
  initialize_array (Test[][][], FALSE);
  initialize_stack();
  Repeat until heap is empty begin
    voxel:=heaptop();
    If global_connectivity_test(voxel) is FALSE then
      push the two neighbors of voxel into stack if they have not been tested;
      delete_vector(voxel);
      Test[voxel.x][voxel.y][voxel.z]:=TRUE;
      Repeat until stack is empty begin
        voxel:=stack_pop();
        If voxel has only one neighbor then
          push the neighbor into stack if it has not been tested;
          delete_vector(voxel);
          Test[voxel.x][voxel.y][voxel.z]:=TURE;
        End if
      End Repeat
    Else
      push the two neighbors of voxel into stack if they have not been tested;
      Test[voxel.x][voxel.y][voxel.z]:=TRUE;
      Repeat until stack is empty begin
        voxel:=stack_pop();
        If voxel has two neighbor then
          push the neighbor into stack if it has not been tested;
          Test[voxel.x][voxel.y][voxel.z]:=TURE;
        End If
      End Repeat
    End If
  End Repeat
End Procedure.
```



☑ 显示内窥镜

☐ 开启表面碰撞检测功能

总共帧数: 683

帧数 0

速度 1

Fig. 6 – Interface of virtual endoscopy.

5.2. *Experiment results*

We evaluated standard topological thinning, original LUT and our improved centerline extraction algorithm using 3 groups of data sets classified by the size of voxels. Only datasets with intact colon were selected. The two endpoints were chosen through man–machine interaction before centerline extraction.

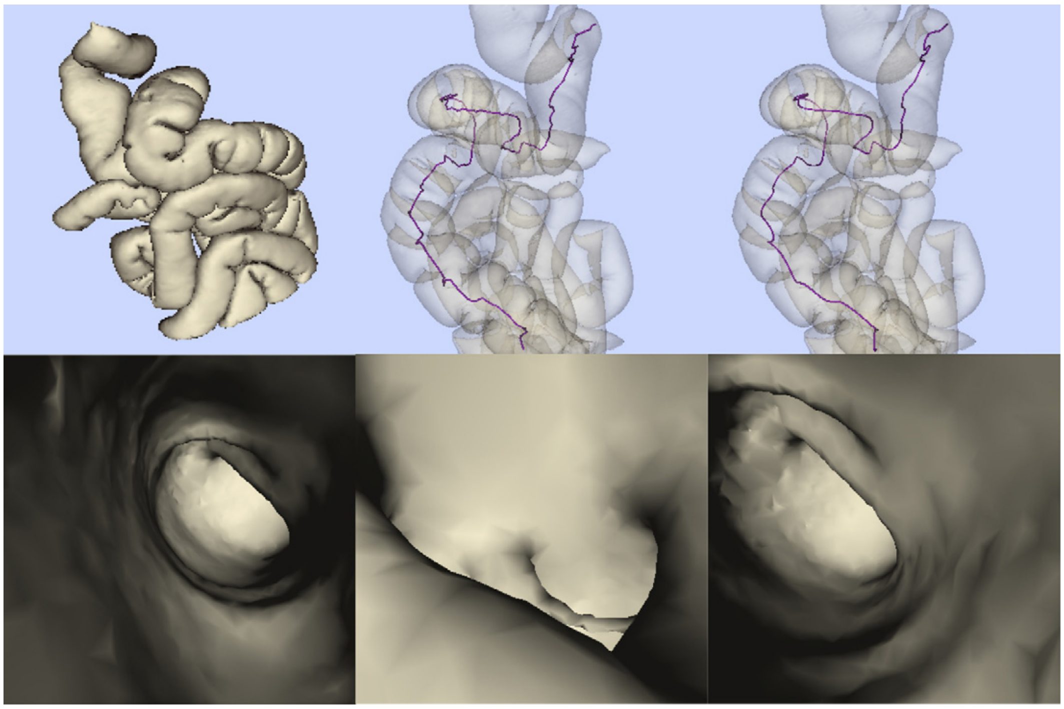


Fig. 7 – Three pictures in top show solid segmented data, phantom model with centerline and smoothed centerline from left to right respectively. Three pictures in bottom show some inside views.

Table 4 – Comparison between the standard one, LUT table and our improved version.

	Standard topological thinning	With Look-up Table	Our improved version
Group 1 voxel size 3 millions	147.453s	20.314s	18.051s
Group 2 voxel size 4 millions	185.413s	30.242s	26.168s
Group 3 voxel size 13 millions	389.619s	158.904s	50.281s

Examples of the centerline obtained are illustrated in Fig. 7. Table 4 shows the running time of different algorithm for different datasets. All experiments are performed on a P4 1.6 GHz CPU with 1G Mb RAM running Microsoft Windows XP professional (Microsoft Corp.).

5.3. *Discussion*

Standard topological thinning algorithm is a very time consuming process. Sadleir et al. have improved the standard one by utilizing a Look-Up Table to significantly reduce the computational cost. We describe an extended version of topological thinning based on LUT, which can avoid unnecessary global connectivity testing in extraneous loops removing stage. We obtained the running times that are an average of 7 times faster than the standard topological thinning algorithm. Comparing with original LUT method, when the size of volume data is relative small, this improvement is slight. However, when the size is large, the improvement is impressive. This can be explained by the fact that large dataset contains more complex topological structure of segmented volume data, which may have more extraneous loops after topological thinning. Considering time complexity of removing extraneous loops is $O(n^2)$, our extension will have a good performance on this situation since we decrease amount of voxels concerned with global connectivity test.

The revised method described above can deal with any colons, i.e. intact colons or part of a completed colon, where there is an unobstructed path between the selected two endpoints. This extension is not restricted to applications in CT and can be used in other areas of virtual endoscopy where a single path is required for a hollow object. In the cases of collapsed colon segments which are often present in CT colonoscopies, this method should be executed for each individual section.

6. Conclusion

We present an extension to Sadleir's colon centerline extraction method based on LUT. This extension maintains a min-heap structure to get the closest voxel to the surface from the vector, and uses region growing to remove the extraneous loops. It avoids unnecessary global connectivity testing in the candidate set for the final centerline, thus can observably reduce the running time in extraneous loops removing stage when the number of elements in the vector is large after topological thinning.

Conflict of interest

We claim that there is no conflict of interest in the submission of this manuscript.

Acknowledgement

This work was supported by National Basic Research Program of China (Project Number 2006CB303106) and Doctoral Subject Special Scientific Research Fund of Education Ministry of China (Project Number 20070335074).

References

[1] C.D. Johnson, A.K. Hara, J.E. Reed, Virtual endoscopy: what's in a name? Am. J. Roentgenol. 171 (1998) 1201–1202.
[2] A. Oto, Virtual endoscopy, Eur. J. Radiol. 42 (2002) 231–239.
[3] D.J. Vining, D.W. Gelfand, R.E. Bechtold, E.S. Scharling, E.K. Grishaw, R.Y. Shifrin, Technical feasibility of colon imaging with helical CT and virtual reality, Am. J. Roentgenol. 162S (1994) 104.
[4] Y.F. Tsao, K.S. Fu, A parallel thinning algorithm for 3D pictures, Comput. Graph Image Proc. 17 (1981) 315–331.
[5] Y. Zhou, A. Kaufman, A.W. Toga, Three-dimensional skeleton and centerline generation based on an approximate minimum distance field, Vis. Comput. 14 (1998) 303–314.
[6] Y. Zhou, A.W. Toga, Efficient skeletonization of volumetric objects, IEEE Trans. Vis. Comp. Graph 5 (1999) 196–209.
[7] I. Bitter, A. Kaufman, M. Sato, Penalized-distance volumetric skeleton algorithm, IEEE Trans. Vis. Comp. Graph 7 (2001) 195–206.
[8] I. Bitter, M. Sato, M. Bender, A. Kaufman, M. Wan, M. Wax, Automatic, accurate and robust colon centerline algorithm, Radiology 217 (S) (2000) 705.
[9] I. Bitter, M. Sato, M. Bender, K. McDonnell, A. Kaufman, M. Wan, CEASAR: a smooth, accurate and robust centerline extraction algorithm, Proc. IEEE Visual. (2000) 45–52.
[11] Y. Ge, D.R. Stelts, J. Wang, D.J. Vining, Computing the central path of the colon from CT images, Proc. SPIE Med. Imaging: Image Process. 3338 (1998) 702–713.
[12] Y. Ge, D.R. Stelts, J. Wang, D.J. Vining, Computing the centerline of a colon: a robust and efficient method based on 3D skeletons, J. Comput. Assist. Tomogr. 23 (1999) 786–794.
[13] R.J.T. Sadleir, P.F. Whelan, Colon centerline calculation for CT colonography using optimised 3D topological thinning, in: First International Symposium on 3D Data Processing, Visualization and Transmission, 2002, pp. 800–803.
[14] Robert J.T. Sadleir, P.F. Whelan, Fast colon centreline calculation using optimized 3D topological thinning, Comput. Med. Imaging Graph. 29 (2005) 251–258.
[15] S. Bouix, K. Siddiqi, A. Tannenbaum, Flux driven fly throughs, Comput. Vision Pattern Recognit. 1 (2003) 449–454.
[16] T. Deschamps, L.D. Cohen, Fast extraction of minimal paths in 3D images and applications to virtual endoscopy, Med. Image Anal. 5 (2001) 281–299.

[17] M. Sato, S. Lakare, M. Wan, A. Kaufman, M. Wax, An automatic colon segmentation for 3D virtual colonoscopy, IECE Trans. Inf. Syst. E84 (1) (2000) 201–208.

[18] C.L. Wyatt, Y. Ge, D.J. Vining, Automatic segmentation of the colon for virtual colonoscopy, Comput. Med. Imaging Graph. 24 (2000) 1–9.

[19] M.S. Hassouna, A.A. Farag, Robust centerline extraction framework using level sets, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 458–465.

[20] R. Van Uitert, I. Bitter, Subvoxel precise skeletons of volumetric data based on fast marching methods, Med. Phys. 34 (2) (2007) 627–638.