# Automatic Centerline Extraction for Virtual Colonoscopy

Ming Wan, Zhengrong Liang*, Qi Ke, Lichan Hong, Ingmar Bitter, and Arie Kaufman

*Abstract*—**In this paper, we introduce a concise and concrete definition of an accurate colon centerline and provide an efficient automatic means to extract the centerline and its associated branches (caused by a forceful touching of colon and small bowel or a deep fold in twisted colon lumen). We further discuss its applications on fly-through path planning and endoscopic simulation, as well as its potential to solve the challenging touching and colon collapse problems in virtual colonoscopy. Experimental results demonstrated its centeredness, robustness, and efficiency.**

*Index Terms*—**Centerline extraction, distance mapping, flight-path planning, virtual colonoscopy (VC).**

## I. Introduction

VIRTUAL endoscopy is an integration of medical imaging and computer graphics technologies, leading to a computer-based alternative to the traditional fiberoptic endoscopy for examining the interior structures of human organs [17], [19], [24]. It has many advantages compared with the traditional endoscopy procedures, such as being noninvasive, cost-effective, highly accurate, free of risks and side effects (e.g., perforation and infection), and easily tolerated by the patient. Therefore, many prototype systems have been developed for a variety of clinical applications, including virtual colonoscopy (VC), virtual bronchoscopy, virtual angioscopy, and others. We have been developing a three-dimensional (3-D) VC system [12], [13] toward a fast and accurate computer-aided screening modality for early detection of colonic polyps, which are the major cause ($>90\%$) of colon cancer, the second leading cause of cancer deaths in the USA.

Our VC system takes a spiral computed tomography (CT) scan of the patient's abdomen after the colon is cleansed and distended with room air or $CO_2$ gas. Several hundred high-resolution slice images are rapidly acquired during a single breath hold by currently available CT technology, forming a volumetric abdomen dataset. A model of the entire colon is then extracted from the abdomen dataset, where the tagged (by contrast solutions) residual stool and fluid are electronically removed by image segmentation algorithms from the dataset [6], [14], [16]. A centerline of the colon model is then determined, where a potential field is built within the colon lumen. The colon model can be viewed by an automatic planned navigation following the centerline for a general overview [12], or by an interactive navigation for a more flexible and detailed study on suspicious regions [13], [26]. The navigation is based on volume rendering [25] and executed in real time on a personal computer (PC) platform [8].

A crucial component of our VC system is the extraction of the centerline, which not only provides a compact colon shape description, accurate colon geometry measurement, and precise polyp registration with fiberoptic colonoscopy or between supine and prone CT scans, but also supports automatic path planning for both planned and interactive navigations. An accurate and fast extraction of the centerlines from patient datasets has been a challenging problem, due to the complex structure of the colon. In Section II, we will review the centerline concepts and related algorithms and introduce our centerline definition and its extraction algorithm.

## II. Theory

The concept of *centerlines* (also known as *medial* or *symmetric axes*) was first introduced by Blum [3]. In a tubular object like the colon, there is normally only one centerline that spans it. In more general cases, an object may have a more complicated shape—such as airways in the lungs—and, therefore, there is a set of centerlines attaching to each other through the object, forming a topology similar to a skeleton. Hence, the set of connected centerlines in an object is also called a *skeleton*. (Skeleton and centerlines are used interchangeably in this paper.) A concise definition of "skeleton" was given in [3] as the locus of centers of maximal disks (in two dimensions) or balls (in three dimensions) contained in the shape. Extracting the skeleton has been a very challenging task in various applications, resulting in various modifications on the centerline definition, so that it can be extracted and utilized efficiently. In the following, we will focus on the colon object.

Based on previous reports [1], [2], [5], [7], [10]–[13], [18], [22], [23], an adequate colon centerline definition and extraction should meet the following requirements.

Manuscript received June 6, 2001; revised August 14, 2002. This work was supported by the National Cancer Institute under Grant NIH CA82402, by the National Science Foundation under Grant MIP9527694, and by Viatronix Inc. The Associate Editor responsible for coordinating the review of this paper and recommending its publication was W. Higgins. *Asterisk indicates corresponding author.*

M. Wan is with The Boeing Company, Seattle, WA 98124-2207 USA (e-mail: Ming.Wan@Boeing.com).

*Z. Liang is with the Department of Computer Science and the Department of Radiology, State University of New York at Stony Brook, Stony Brook, NY 11794-8460 USA (e-mail: jzl@clio.rad.sunysb.edu).

Q. Ke and I. Bitter are with the Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794 USA (e-mail: qke@cs.sunysb.edu; ingmar@cs.sunysb.edu).

L. Hong is with the Palo Alto Research Center, Palo Alto, CA 94304 USA (e-mail: hong@parc.com).

A. Kaufman is with the Department of Computer Science and the Department of Radiology, State University of New York at Stony Brook, Stony Brook, NY 11794 USA (e-mail: ari@cs.sunysb.edu).

Digital Object Identifier 10.1109/TMI.2002.806409

1) **Connectivity**. The definition of connectivity is closely related to the data presentation. In virtual endoscopy systems, the acquired CT or magnetic resonance imaging (MRI) dataset is usually presented on a 3-D grid, called a *volume*. Each grid element is called a *voxel*, which can have a cubic or rectangular shape. Two voxels are 6-, 18-, or 26-connected if at most one, two, or three of their 3-D coordinates differ by one. If two voxels are 6-, 18-, or 26-connected, they are said to be *directly connected*. Connectivity requires that an extracted centerline is a sequence of directly connected voxels. Evidently, a 26-connected *discrete centerline* is the smoothest one among the three direct connections and, therefore, is the closest one to the actual *continuous centerline* in the continuous 3-D space. Given a colon model $D$ segmented from a data volume, the interior region $R$ of $D$ and the centerline[1] $C$ spanning over $R$, a connected centerline can be described mathematically as follows:

$$\forall v_0 \in C \rightarrow \exists v_1 \in C \wedge v_1 \in \partial(v_0) \qquad (1)$$

where $\partial(v_0)$ is the set of voxels inside $R$ that are directly connected to voxel $v_0$, and $v_1$ is any voxel inside $R$.

2) **Centricity**. The centerline should stay away from the colon wall as much as possible. This requirement guarantees that the centerline is not only an accurately centered object shape descriptor, but also a safe navigation guide that prevents the navigator from penetrating the colon wall (i.e., always stays inside the colon lumen) and hugging the corners at sharp turns.

3) **Singularity**. The centerline should be a single path of one-voxel width, without any manifolds or self-intersection. Such a sequential single-voxel chain is required for both navigation path planning and quantitative measurement in endoscopic simulations. More precisely, given any two directly connected centerline voxels $v_0$ and $v_1$, i.e., $\forall v_0 \in C \wedge \forall v_1 \in C \wedge v_1 \in \partial(v_0)$, a one-voxel-wide single centerline $C$ should satisfy the following condition:

$$\forall v_2 \in R \wedge v_2 \in \partial(v_0) \wedge v_2 \in \partial(v_1) \rightarrow v_2 \notin C \qquad (2)$$

which requires that each centerline voxel has, at most, a direct connection to two other centerline voxels, where $v_2$ is any voxel inside $R$.

4) **Detectability**. Although a perfectly prepared colon has a single tubular shape without any branches or holes, colon touching with small bowel or by twist as well as colon collapse may occur, resulting in a complicated structure with holes and branches in the colon dataset. This problem becomes even more complicated when one or more colon segments collapse fully, which may separate the colon tube into multiple segments and cut loops into branches if it occurs on looping areas. Therefore, the algorithms for extracting the defined centerline shall tolerate and/or detect such looping and branching topology in each colon segment.

[1]Centerlines mentioned in this paper are, by default, the discrete 26-connected centerlines.

5) **Robustness**. In addition to detectability, the algorithms shall be robust, i.e., they should perform consistently for clinically acquired CT colon datasets, regardless if it is a perfectly prepared tubular colon or a colon which is separated into multiple segments with branches and loops. Furthermore, the extracted centerline should not vary for the location of the start and/or end points, or any 3-D transformation on the colon volume such as translation or rotation.

6) **Automation**. In addition to robustness, the two end points of the colon shall be determined automatically, resulting in a fully automatic procedure, without any user participation.

7) **Efficiency**. The algorithms embedded in the fully automatic procedure shall be computationally efficient, so that the centerline extraction could be done in seconds on a PC platform (cost effectiveness).

Based on these requirements, we summarize existing centerline-extraction algorithms with a brief discussion on their strength and weakness. Then we propose a novel algorithm, which is capable of automatically delivering a colon centerline in a fast and robust manner and satisfies all the above requirements.

## *A. Brief Review of Existing Algorithms for VC*

There has been extensive work on centerline extraction. Most of these methods can be divided into three categories: manual extraction, topological thinning, and distance mapping.

*1) Manual Extraction:* This method requires the user to manually mark the center of each colon region on each image slice of several hundreds in a dataset. When enough points are selected, a centerline is obtained using a linear or higher order interpolation (e.g., [10]). This is time consuming and tedious, and the result is by no means more accurate than the results of other methods, for two reasons. First, a center point in a two-dimensional (2-D) slice may not lie along the medial axis in the 3-D space. Second, this does not guarantee that the interpolated line will always stay inside the colon lumen even though all the selected points are inside, as noted in [1], [5], and [23].

*2) Topological Thinning:* This technique is traditionally assumed to provide the *most accurate* result. It peels off a volumetric object layer by layer iteratively until there is only one central layer left, which is essentially the skeleton of the object. During the iterative process, "simple points" need to be detected so that the removal of these points does not affect the topology of the object [20]. Although the idea is very simple, the repetitive procedure is quite time consuming, especially for identifying the simple points. For example, Hong *et al.* [12] spent several hours extracting the centerline from a $512^3$ colon dataset on an SGI Power Challenge R10000 CPU. Fast thinning algorithms have recently been extensively investigated from the traditional sequential voxel-by-voxel thinning to the more efficient parallel layer-by-layer thinning, as discussed below.

Ge *et al.* [11] accelerated the topological thinning algorithm during colon centerline extraction using three strategies. First, a specific data structure is used to accelerate the procedure of checking topological constraints. Second, cavities within each colon region, which are caused by imperfect colon segmentation, are removed before thinning. Third, the colon volume is downsampled to a much smaller size. Their method took about 8 min on the original colon dataset and approximately 1 min on the downsampled colon volume on an SGI R10000 CPU. The removal of cavities may change the object topology, and the downsampling may affect the accuracy of the centerline [5].

Paik *et al.* [18] proposed a more reliable acceleration approach, which avoids the expensive evaluation of simple points by incorporating the shortest path method [9] into a parallel thinning procedure. Their method has three steps. First, it determines an initial shortest path along the outmost layer of the object. Second, it performs one step of a parallel unrestricted thinning and computes a new shortest path through the union of all the voxels on the newly exposed layer as well as those voxels in the previous path. Third, it repeats the second step until only one center path remains. This method was further extended to extract branches in the object. It took about 10 min to extract the centerline from a patient colon dataset on an SGI O2-R5000 CPU. Evidently, this method guarantees the connectivity of centerlines and inherits precision from the thinning algorithm. However, the manual detection of branch tips needs to be improved, in addition to the computing time.

*3) Distance Mapping:* This method, first used in robotic path planning [15], is considered to be the *fastest* one among the three categories. It has two phases. The first phase computes the distance from a user-specified source point to each voxel inside the 3-D object,[2] which is called a *DFS-distance* (i.e., the distance from the starting point) in this paper. Once this distance map is generated, the second phrase extracts the shortest path from any starting voxel to the source point by descending through the gradient of the distance map. The shortest path can be rapidly extracted by a simple backtracing rather than the steepest descent, if Dijkstra's shortest path technique [9] is used to create the distance map. Those centerline algorithms [1], [2], [5], [7], [10], [22], [23], [27], using distance mapping differ in how they specify the distances between orthogonal, 2-D-diagonal, and 3-D-diagonal neighboring voxels. The most accurate distance measure is the $1$–$\sqrt{2}$–$\sqrt{3}$ Euclidean metric [21]. Unfortunately, most algorithms use the approximate distance transformation metrics [4] in order to reduce computational expenses or to obtain some specific properties (such as clusters [27]) from the generated distance map. The popularly used distance metrics include the $1$–$\infty$–$\infty$ Manhattan metric [22] (considering only 6-connectivity), the 1–2–3 metric [27], the 3–4–5 Chamfer metric [10], and the 10–14–17 metric [5], where the distance errors decrease in the order as compared with the Euclidean distance.

The major advantages of distance mapping methods are their computational efficiency and guarantee of the extracted shortest paths staying inside the object. However, the shortest path has a tendency to hug the corners rather than following the medial axis around sharp turns. Therefore, a great effort has been made to push the shortest path toward the object center. For example, Samara *et al.* [22] proposed to initially calculate the shortest path twice from both ends, then merge these two paths to a mean centerline, and finally relocate each centerline voxel to the mass center of the grown region lying in the plane perpendicular to the tangent of the mean centerline at that voxel. However, such a refinement could not completely solve the centricity problem at the high curvature regions.

[2]A voxel inside a volumetric object is also called an *inside voxel* in this paper.

More recent algorithms considered the use of an additional distance from each inside voxel to the nearest object boundary, to improve the centricity of the shortest path. We call this the *DFB-distance* in this paper (i.e., the distance from the boundary). Similar to the situation with the DFS-distance, a variety of distance metrics with different accuracies exist to measure the DFB-distance.

Chen *et al.* [5] exploited the DFB-distance to relocate each voxel on the shortest path to the maximal DFB-distance voxel within the plane perpendicular to the shortest path. Unfortunately, a single correcting step in the 2-D plane does not yield an optimal centerline, and even the iterative refining is not guaranteed to find a global optimum. Zhou *et al.* [27] proposed a better solution that relocates each voxel on the shortest path to the maximal DFB-distance voxel within the voxel cluster of the same DFS-distance, rather than within a perpendicular 2-D plane. However, such relocation disconnects the path. This method took about 8 min to extract the centerline from a $512^3$ colon dataset on an SGI Power Onyx-R10000 CPU.

Bitter *et al.* [1] proposed a more efficient centerline algorithm using a different strategy, where the precision of the centerline is adjusted during the procedure of shortest path generation rather than afterward. In the distance map, the distance for each inside voxel is defined as a heuristic combination of the DFS-distance and the DFB-distance, which is called a penalty distance. However, the penalty distance could not completely prevent the shortest path from hugging the corners. Furthermore, this algorithm fails to extract a complete colon centerline when holes appear on the colon wall, because it has a higher priority to enter these holes rather than span through the entire colon. For further speedups, they identified the center region of the object and extracted object centerline within this region. Their method took about 5 min on the $512^3$ colon data on the SGI Power Challenge-R10000 CPU. Sato *et al.* [23] extended this algorithm to detect branch structures by the use of a simple strategy. Once the penalty distance field is generated as in [1], it finds the voxel with the maximal penalty distance and extracts a centerline by backtracing to the source point or an early encountered centerline voxel, followed by marking those inside voxels near this centerline and rolling an adaptive sphere along the path. It repeats this procedure until all the inside voxels are marked. This method guarantees the connection of all the detected centerlines. However, determining the covered area near each centerline is heuristic and expensive. Also, the "hugging-corner" problem remains, and the shape of the extracted skeleton is sensitive to the selection of the source point.

More recently, Bitter *et al.* [2] published an efficient penalized-distance volumetric skeleton algorithm, which combines their previous work on finding the centerline [1] and its branches [23] and improves the work by making two small corrections, but the basic ideas remained the same. The first correction was on their centerline algorithm [1]. They extracted the maximum-length path, instead of the maximum-penalty path, among all the minimum-cost paths in the penalized-distance field. Therefore, they could always extract the longest path as the centerline. Their second correction was for their branch-extraction algorithm [23]. To extract a new branch, they computed a new penalty-distance field from all the voxels on the centerline and the branches detected so far, instead of always using the same penalty-distance field from a fixed source voxel [23]. A comparison between this method and our algorithm will be given later in this paper.

### B. Description of New Algorithm

This work aims to overcome the limitations of the previous methods and to develop a framework satisfying the seven requirements described above. Our centerline definition and extraction are based on the DFB-distance field, which contains the exact Euclidian distance from each voxel inside the colon to the nearest colon boundary. Most of the existing centerline methods do not use such an accurate Euclidian DFB-distance because of the assumption that "an accurate calculation of the Euclidian distance is neither efficient nor algorithmically trivial, especially for large high-resolution 3-D volumetric data sets that include complicated objects" [27]. However, this assumption is not true. We have implemented an efficient algorithm, proposed by Saito *et al.* [21], which rapidly calculates the exact Euclidian DFB-distance field.

Before describing our new centerline-extraction algorithm, we will introduce a concise and concrete definition of the centerline based on the DFB-distance field. *The centerline is defined as the minimum-cost path spanning over the inversed-DFB-distance field inside the colon.* This definition has the following advantages. It is a concise and concrete definition, similar to the traditional description of the centerline given by Blum [3], but more practical for implementation. It is different from all of the previous explicit or implicit centerline definitions in the distance mapping methods that are based on the shortest path in the DFS-distance field or its variations. Solely based on the accurate DFB-distance field, our centerline definition, for the first time, specifies a highly centered centerline for a distance-mapping method, *without* the tendency to hug the corners. Furthermore, it suggests a theoretically based, efficient and robust algorithm to extract the centerline and its branches from a minimum-cost spanning tree (MST tree) in the DFB-distance field as described below. The algorithm consists of two steps: 1) constructing a MST tree in the DFB-distance field; and 2) extracting the colon centerline and its branches (if any exist) from the tree. It is a fully automatic approach.

*1) Construction of a MST tree:* This step can be divided into two stages. First, it converts the CT volume with DFB-distances to a 3-D directed weighted graph. Second, it builds up a MST tree from the weighted graph using a modified Dijkstra's shortest path technique. In the following, we assume that the colon interior is a 26-connected region. If two or more colon regions occur due to colon collapse, we will build one MST tree for each region (see Section II-D.4 for more details).

The conversion from a volumetric dataset to a 3-D directed weighted graph is depicted in Fig. 1. Each voxel forms a node in the graph. Edges represent the 26-neighbor relations between voxels. Each edge has two directions pointing toward its two end points, respectively. Each direction has its own weight as the inverse of the DFB-distance, say, $\mathrm{DFB}(A)$, of the end voxel $A$ to which it points.[3] To distinguish the DFB-distance from its inverse, we call the latter *DFB-cost*. Although our directed DFB-based graph looks more complicated than the traditional nondirected graphs used in [1] and [26], its implementation is provably simpler and faster by using our modified Dijkstra technique as described below.

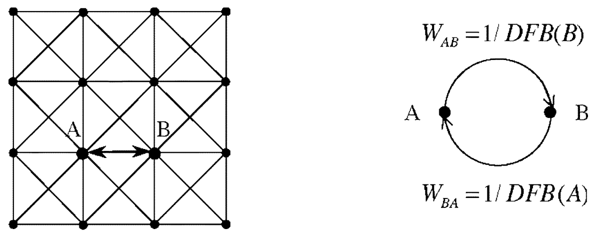


Fig. 1. Two-dimensional top view of the 3-D directed weighted graph.

The MST tree of the directed graph is defined as a tree that connects all the inside voxels at the minimum DFB-cost. In order to build up such a MST tree with minimal computations, we propose to modify the standard Dijkstra technique [9], because we only care about the DFB-cost at each individual node when we build the MST tree, so that it does not accumulate the weights of the nodes during the region-growing iterations.

Our modified Dijkstra technique consists of the following four steps, where a source point $S$ at one end of the colon is predefined by the user or automatically selected by our VC system. If the user does not specify the source point, our algorithm will automatically select the lowest (bottom) voxel among all the voxels that are segmented as inside voxels. (It will pick up the middle one if more than one voxel is there.)

Step 1) Mark source point $S$, define $S$ as the *current node*, set its *pathlink* to *NULL*, and let its $\mathrm{DFS}(S)$ be zero, where $\mathrm{DFS}(V)$ is the accumulated distance from voxel $V$ to the source point.

Step 2) Put each of the unmarked 26 neighbors $B$ of the current node $C$ into a sorted heap, set its pathlink to $C$, and calculate its $\mathrm{DFS}(B)$ as $\mathrm{DFS}(C)+\mathrm{DIS}(B, C)$, where $\mathrm{DIS}(B, C)$ is the Euclidian distance between $B$ and $C$.

Step 3) Remove the head of the heap, mark it, and set it as the current node.

Step 4) Repeat Step 2 and Step 3 until the heap is empty.

Fig. 2 shows the flowchart[4] of the above modified Dijkstra algorithm. A fast heap-sorting technique is adopted to find the node with the minimum DFB-cost in the current heap by $O(\log M)$ time scale, where $M$ is the number of elements in the current heap. Because we process each object only once, our algorithm is completed in $O(N \log M)$ time scale, where $N$ is the number of voxels inside the colon. Each inside voxel obtains a pathlink pointing toward its neighboring voxels, through which it reaches the source point with a minimum

[3]Of course, we could directly use the DFB-distance itself as the weight; then we would have to say "*maximum*-cost spanning tree."

[4]For simplicity, we implicitly assume that each voxel $V$ has all its 26 neighbors available at $V.neighbors$ in our flowcharts in this paper. In fact, if a neighbor is not available, we simply ignore it in our algorithm.

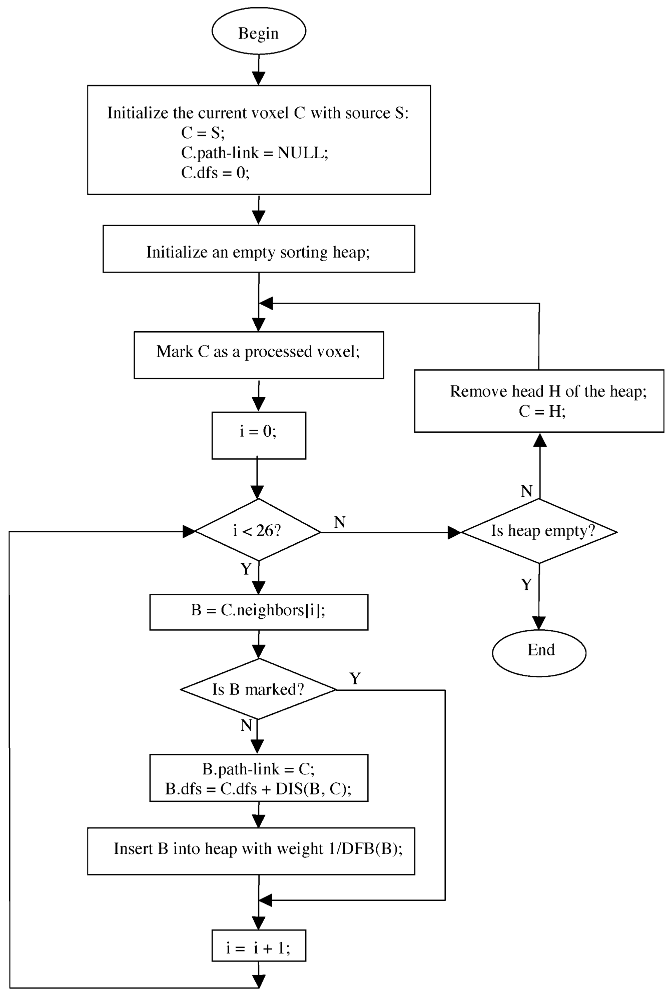

Begin
Initialize the current voxel C with source S:
C = S;
C.path-link = NULL;
C.dfs = 0;
Initialize an empty sorting heap;
Mark C as a processed voxel;
i = 0;
i < 26?
N
Y
B = C.neighbors[i];
Is B marked?
Y
N
B.path-link = C;
B.dfs = C.dfs + DIS(B, C);
Insert B into heap with weight 1/DFB(B);
i = i + 1;
Is heap empty?
N
Y
Remove head H of the heap;
C = H;
End

Fig. 2. Flowchart of constructing an MST tree from a source voxel. A fast heap-sorting technique is adopted so that the heap is sorted in $O(\log M)$ time when a new element is inserted, where $M$ is the number of elements in the current heap. As a result, the algorithm is completed in $O(N \log M)$ time, where $N$ is the number of voxels inside the colon.

DFB-cost. It also obtains a DFS-distance to record the length of this minimum DFB-cost path toward the source point. Our MST tree is represented by the collection of all the pathlinks. The DFS-distances will help us to find the other end of the centerline, and will also contribute to quantitative measurements, as discussed in Section II-D.2.

*2) Extraction of Colon Centerline and Branches:* The centerline-extraction algorithm contains two steps. First, if the user does not specify the end point $E$ of the colon centerline explicitly, the algorithm locates it to be the inside voxel with the maximum DFS-value as calculated in Section II-B.1. This strategy makes the centerline span the entire colon. Second, it finds the sequence of voxels on the centerline by tracing back from $E$ to $S$ according to the pathlinks (see Fig. 3). From the perspective of our MST tree, the centerline is the longest branch in the MST tree that starts from root $S$.

Branch extraction is optional, but clinically useful, for consideration of colon collapse and colon boundary touching problems. However, it is critical for more general virtual endoscopy applications with branch object structures, such as airways and blood vessels. We propose an efficient branch extraction algorithm, based on the same MST tree and the same centerline-extraction algorithm.

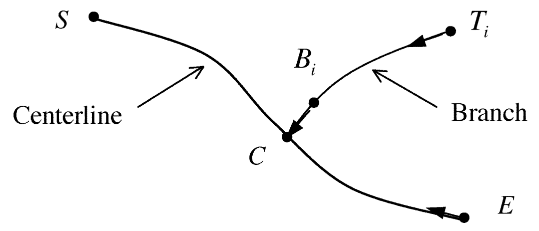


Fig. 3. Extraction of colon centerline and its branches. Voxels $S$ and $E$ are, respectively, the start and end points of the centerline. $C$ is an arbitrary centerline voxel. $T_i$ is the noncenterline voxel that can reach $C$ through pathlinks. The voxel chain from $T_i$ to $C$ through pathlinks is defined to be a branch of $C$. $B_i$ is the voxel directly connected to $C$ along this branch.

Before we describe our branch extraction algorithm, we give a formal definition of a *pathlinked* relationship between two inside voxels. Given two voxels $A$ and $B$ inside a continuous colon region, we say voxel $A$ is pathlinked to voxel $B$ if: 1) $A$ has a pathlink pointing to $B$; or 2) $A$ has a pathlink pointing to a third voxel $C$, which has a pathlink pointing to $B$; or 3) more generally, $A$ has a pathlink pointing to a voxel $D$, which is pathlinked to $B$ through one or more voxels. In the first case, we say $A$ is *directly* pathlinked to $B$; in the second and third cases, we say $A$ is *indirectly* pathlinked to $B$.

Our branch detection algorithm contains the following three steps (see Fig. 3).

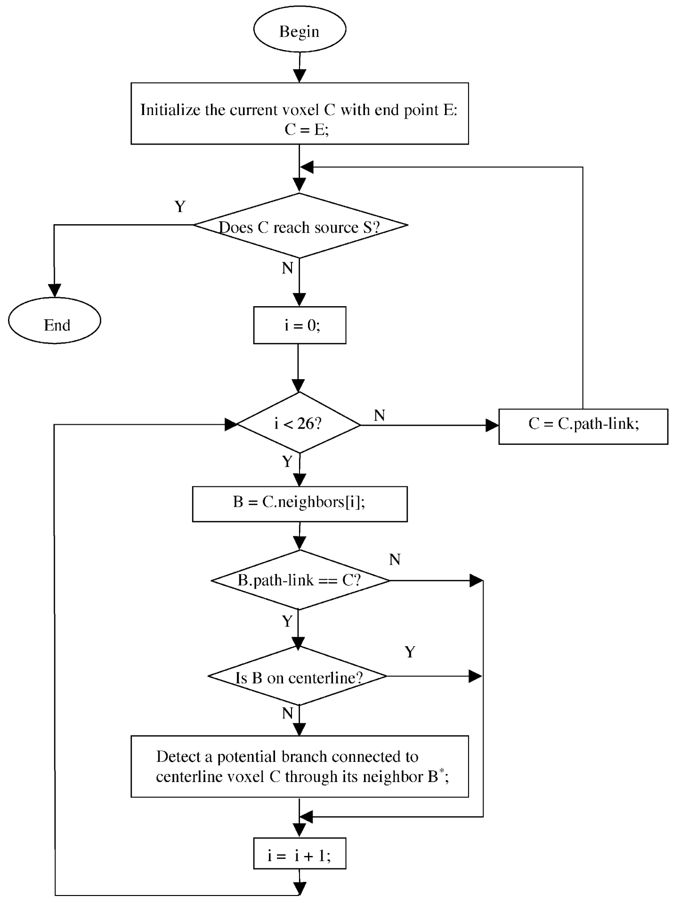
Step 1) Scan the centerline by tracing back from end point $E$ to source point $S$ along the pathlinks.

Step 2) For each centerline voxel $C$, check its 24 neighbors (that is, excluding its two neighbors on the centerline) and find those voxels $B_i$, each of which has a pathlink pointing to $C$.

Step 3) For each voxel $B_i$, search for all those voxels that are pathlinked (directly or indirectly) to it, record voxel $C$ to be the *closest centerline voxel* for them, and find the voxel with largest DFS-distance, $T_i$, among them. Store $T_i$ as the tip of a branch, which grows from the centerline voxel $C$ through its neighbor $B_i$ if $\mathrm{DFS}(T_i)$ is larger than a user-specified or system-default threshold $L$ of the branch length. The length of this branch is computed as $\mathrm{DFS}(T_i) - \mathrm{DFS}(C)$.

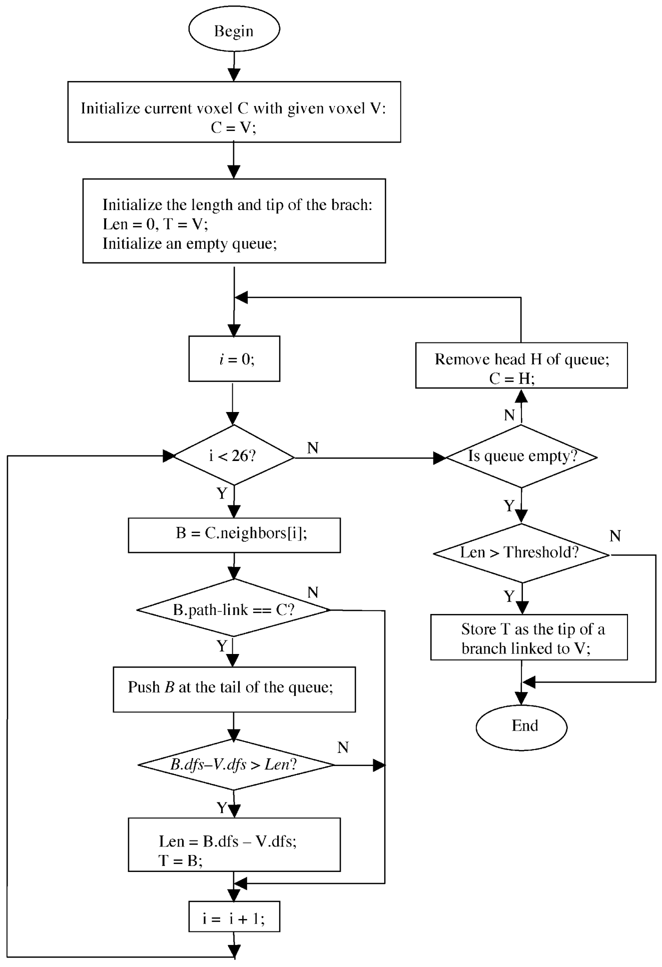
In step 2), we identify the branches at each centerline voxel through its 24 neighbors so that we are able to extract multiple branches that are connected to the centerline through the same centerline voxel. Fig. 4 shows the flowchart of the above branch-detection algorithm. Fig. 5 completes the flowchart of Fig. 4 by showing a simple but efficient implementation of step 3) utilizing a queue.

Because each inside voxel is accessed only once, our automatic branch-detection algorithm is rapidly completed in $O(N)$ time, where $N$ is the number of voxels inside the colon. It is capable of detecting all branches attached to the centerline, including those linked to the same centerline voxel. The information of the closest centerline voxel for each inside voxel is used later for endoscopic simulations (see Section II-D.2). From our experience, these first-level branches provide enough information for flight-path planning and endoscopic simulations



Begin
Initialize the current voxel C with end point E: C = E;
Does C reach source S?
Y
N
End
i = 0;
i < 26?
N
Y
C = C.path-link;
B = C.neighbors[i];
B.path-link == C?
Is B on centerline?
Detect a potential branch connected to centerline voxel C through its neighbor B*;
i = i + 1;

Fig. 4. Flowchart of the branch-detection algorithm along the centerline. A total of 24 neighbors are checked at each centerline voxel to detect all potential branches connected to the centerline voxel through its noncenterline neighbors.



Begin
Initialize current voxel C with given voxel V: C = V;
Initialize the length and tip of the brach: Len = 0, T = V; Initialize an empty queue;
i = 0;
Remove head H of queue; C = H;
i < 26?
Is queue empty?
N
Y
Len > Threshold?
Store T as the tip of a branch linked to V;
End
B = C.neighbors[i];
B.path-link == C?
Push B at the tail of the queue;
B.dfs–V.dfs > Len?
Len = B.dfs – V.dfs; T = B;
i = i + 1;

Fig. 5. Flowchart of detecting the tip of a branch linked to the centerline through a given voxel $V$. A queue is adopted to find all the voxels directly or indirectly pathlinked to $V$. The furthest voxel among them is identified as the tip of the branch if its distance to $V$ is larger than a user-specified threshold.

during the exploration of the tubular colon structure, although it is straightforward to extend the algorithm to detect all the higher level branches by recursively performing such a branch-detection procedure on each detected branch. Similar to the detection of the first-level branches, each object voxel is accessed only once for detection of higher level branches. Assuming that there are a total of $l$ levels of branches in a colon, the total branch detection time is then $O(lN)$.

## C. Properties of the New Algorithm

In this section, we discuss and summarize some important features of our centerline-extraction algorithm, including its precision, connectivity, simplicity, and computational efficacy. Finally, we compare our algorithm with Bitter *et al.*'s efficient penalized-distance volumetric skeleton algorithm [2].

*1) Connected Centered Singular Path:* The centerline algorithm provably delivers a 26-connected highly centered singular path with a 1-voxel width. Specifically, since the centerline is extracted from a 26-connected DFB-distance field inside the colon, it is guaranteed to be inside the colon with 26-connection, which is smoother than 18- or 6-connected centerlines.

The centeredness of our centerline is derived from the DFB-distance field. Therefore, the centerline is completely different from all the previous centerlines extracted from the conventional DFS-distance field [17] or its variations. No *ad hoc* adjustments [1], [2], [27] are needed to push the path back to the center.

Since our centerline and related branches are extracted from the MST tree in the DFB-distance field, they inherit the simplicity from the paths in a tree structure. In other words, the new algorithm generates centerlines of 1-voxel width without any 2-D manifolds or 3-D self-intersection.

*2) High Centeredness:* Our centerline not only has the correct tendency to stay at the center of the colon, but also approaches the colon center at the highest accuracy due to the following two factors.

First, the computation is based on an accurate DFB-distance field with the exact Euclidian values. Therefore, it is more accurate than all the approximate distance transformation metrics reported before, which suffer from the inaccuracy in distance measurement and sensitivity to translation and rotation, especially for large objects, since the approximation error increases with the object size [11]. In fact, our method can outperform the traditional thinning algorithms by exploiting the accurate, yet attractable, DFB-distance field. Theoretically, both methods are capable of finding the center region of the colon, either by directly detecting the center voxels according to their DFB-distances or by peeling off the object voxels layer by layer until the center region is exposed.

Second, our method performs global sorting of DFB-distance values for all the voxels inside the colon. Therefore, it is more accurate than algorithms that only consider a downsampled colon [5], [11] or part of the colon [1], [2], [22] for speedups. Furthermore, we perform a complete sorting rather than a partial sorting, for example, as used in [1] and [2], where if the difference of two costs is less than a default threshold, no sorting is conducted between them for speedups, and the sorting task is simply treated using an ordinary first-in-first-out (FIFO) queue.

*3) High Performance:* The new algorithm does not trade accuracy for speed, or *vice versa*. Its high performance is ascribed to its low computational complexity. It builds up the MST tree in $O(N \log N)$ time and then extracts the centerline and its first-level branches in $O(N)$ time. Its high performance is also ascribed to its full automation. Once a source point is given, the centerline and all its branches are extracted without any delay due to user participation.

*4) Robustness:* Our algorithm is robust in several aspects. Theoretically, its idea is simple due to the unique concise centerline definition. The algorithm is easy to understand, implement, and maintain. Furthermore, it is implemented based on a modified Dijkstra technique, so it inherits the efficiency and robustness of that technique. For example, it is not sensitive to rotations, and does not require any specific geometry in order to run correctly. Moreover, it works perfectly on both ideally segmented continuous colon tubes, as well as on clinical datasets with colon collapses and/or boundary touching.

*5) Comparison With Penalized-Distance Skeleton Algorithm:* The penalized-distance algorithm of Bitter *et al.* [2] is another efficient skeleton extraction algorithm that belongs to the distance-mapping category and has several similarities to our algorithm. For example, both methods are capable of rapid and automatic extraction of both the object centerline and its branches. Both exploit the fast Dijkstra's algorithm or its variation to extract central paths in a 26-connected distance field, and both utilize the DFB- and DFS-distance measurements for centerline extraction. However, there are several important differences between these two methods, which lead to different skeleton qualities and computational efficiencies.

First, these two methods have different definitions of the centerline, resulting in different approaches in utilizing the Dijkstra's region-growing strategy [9]. The original Dijkstra's algorithm uses the shortest distance from the source point as a voxel's weight; the penalized-distance algorithm computes each voxel's weight by adding a heuristic penalty factor into the Dijkstra's weight. Our method replaces the Dijkstra's weight by the voxel's DFB-distance. Therefore, these methods deliver different skeletons for the same object. The Dijkstra's algorithm delivers the shortest paths that hug the interior colon boundary at sharp turns. Similarly, because the shortest distance from the source affects the weight at each voxel, the extracted centerline by the penalized-distance algorithm has the tendency to approaching the interior colon boundary at sharp turns for a shortcut in the colon whenever possible. Furthermore, because the heuristic penalty factor is specified by the user, a different centerline can be extracted from the same colon dataset by choosing a different penalty parameter. On the contrary, based solely on the DFB-distance field, our method provides a nonheuristic solution and delivers a unique skeleton for each colon dataset. The accurate DFB-distance measurement delivers a *centered* path rather than a *shortest* one, so that our centerline stays as central as possible even at the sharp turns.

Second, the computational efficiencies of these two methods are different for extraction of the centerline and its branches. During the centerline extraction, the penalized-distance algorithm uses a weight that considers both the shortest distance from the source and a penalty factor. Because the shortest distance from source is accumulated during the Dijkstra's region growing, the penalized-distance algorithm has to compute and update the weights of object voxels from time to time. In contrast, our DFB-cost is constant during the Dijkstra's region growing, resulting in a significant saving of computing time.

To extract a new branch, the penalized-distance algorithm needs to recompute a new penalized-distance field from all those voxels on both the centerline and the detected branches. The time scale for extracting a branch becomes $O(N \log N)$, where $N$ is the number of object voxels. If there are $k$ branches in an object, the computational complexity of branch detection would be $O(kN \log N)$. In contrast, our algorithm takes $O(N)$ time to extract all first-level branches and $O(lN)$ time to extract all branches within $l$ levels (see Section II-B.2). Even in the worst case where multiple levels of branches need to be extracted, our method is faster, because $l$ is always smaller than $k$.

### D. Use of the Centerline for Virtual Colonoscopy

Generally, centerlines have many important applications for compact shape description, data compression, automatic navigation, object tracking, etc. In this section, we focus on its specific applications in virtual colonoscopy.

*1) Fly-Through Path Planning:* Virtual navigation technique is critical in a VC system. This decides how the physician controls the movement of a virtual camera with six degrees of freedom (DOF) to examine the inner surface and identify colonic polyps. Traditional work focuses either on planned navigation or free navigation. In planned navigation [12], the camera automatically moves along a precomputed fly-through path from one end to the other to capture a sequence of navigation frames (where our centerline provides a centered smooth fly-through path). This method can provide a quick overview of the inner colonic surface, but it limits the flexibility of a more detailed study of suspicious regions due to the lack of interaction. In contrast, free navigation allows the user to take full control of the camera during navigation. Immediate visual feedback is required for efficient interactions. However, without any guidance inside a long and twisted colon, such navigation is time consuming and challenging. Recently, Hong *et al.* [13] proposed an interactive navigation technique by introducing a physically based camera-control model to overcome the problems of free navigation. A potential field was defined based on two distance fields (DFB-distance and DFS-distance) inside the colon, so the camera always receives two different forces, a repulsive force from the colon wall and an attractive force from the destination, generated by the potential fields.

The repulsive force prevents collision on the colon wall and the attractive force guides the navigation. Unfortunately, this camera model suffers from the local minimum problem [15], which is frustrating to the user. For example, when the camera is approaching a narrow region inside the colon, it would always be pushed back due to the rapidly increasing repulsive force near that region.

Since our centerline is based on the DFB-distance field, it enables a more efficient camera-control model that combines the benefits of both the planned and the interactive navigations, and successfully eliminates the local minimum problem [26]. Our camera model combines two navigation modes: *automatic* fly-through and *interactive* walk-through. In the automatic fly-through mode, our camera flies automatically from the current position (which can be an arbitrary place inside the colon) toward the destination (the end point) along the centerline, according to the pathlinks generated by our centerline algorithm. Whenever necessary, the user can take over the control at any time by clicking the mouse to switch to the interactive walk-through mode. The user can then freely adjust the camera movement in the interior colon region for a more detailed inspection. The camera only receives repulsive force when it is too close to the colon wall to avoid colliding with or penetrating the wall. When the user wants to return from the interactive walk-through—for instance, to pass a local-minimum area or just to speed up the exploration—the user simply releases the mouse button and lets the camera fly in the automatic mode. Our technique results in a smooth navigation by implementing a seamless switch between the two modes. The immediate visual feedback during navigation is guaranteed by our fast volume-rendering techniques [25], which have reached more than 10 Hz on a low-cost PC platform with a single processor, and more than 20 Hz with dual processors, as shown from our most recent experimental results [8].

*2) Quantitative Measurements for Endoscopic Simulations:* Another important use of the centerline in our VC system is to provide accurate measurements of abnormality locations. For example, once a polyp is detected during navigation, the physician immediately knows its location and distance from the rectum for surgery registration. It is noted that the DFS-distance at each voxel on the inner colon wall may not be the same distance from that of the voxel to the rectum, as compared to the measurement of optical colonoscopy, which is correlated to the measurement on the centerline. The distance from each voxel to the rectum must be "mapped" onto the centerline. Our centerline and its associated DFB-distance field provide an effective means for this mapping using the closest centerline point (voxel) at each voxel inside the colon.

Fig. 6 shows an example of how to accurately measure the distance (location) of a polyp from the rectum. Assume that $P$ is the voxel at the top of the polyp, $C$ is $P$'s closest centerline voxel [$C$ was identified and stored for $P$ in step 3) of Section II-B.2], and $C'$ is the voxel on the centerline whose DFS-distance equals $\mathrm{DFS}(P)$. If $P$ is very close to $C$, the physician will not have problem in finding the polyp at a distance of $\mathrm{DFS}(P)$ from the rectum, although the camera may pass $C$ and reach $C'$ on the centerline. However, if $P$ is at the tip of a long branch (a deep fold), at the $\mathrm{DFS}(P)$ distance the camera would be too far to see the polyp. An adequate measure on the distance should be $\mathrm{DFS}(C)$, rather than $\mathrm{DFS}(P)$ from the rectum. Then the polyp should be at a distance of $\mathrm{DFS}(P) - \mathrm{DFS}(C)$ from the camera, either on the colon wall near the centerline or inside a deep fold away from the centerline, where the closest centerline voxel for each inside voxel would have already been detected during our branch-extraction procedure, as described in Section II-B.2.

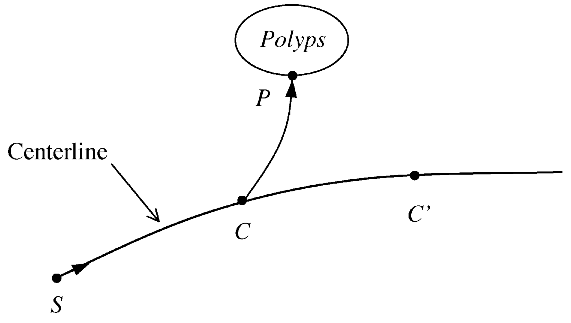


Fig. 6. Endoscopic measurements of a polyp in 3-D virtual colonoscopy. Voxel $S$ is the start point of the colon centerline. $P$ is an interior colon voxel at the surface of a polyp. $C$ is the closest centerline voxel to $P$. Centerline voxel $C'$ has the same distance from source $S$ as $P$ does. We use the distance from $C$ to $S$ rather than the distance from $P$ to $S$ as a conservative measurement to the location of the polyp.

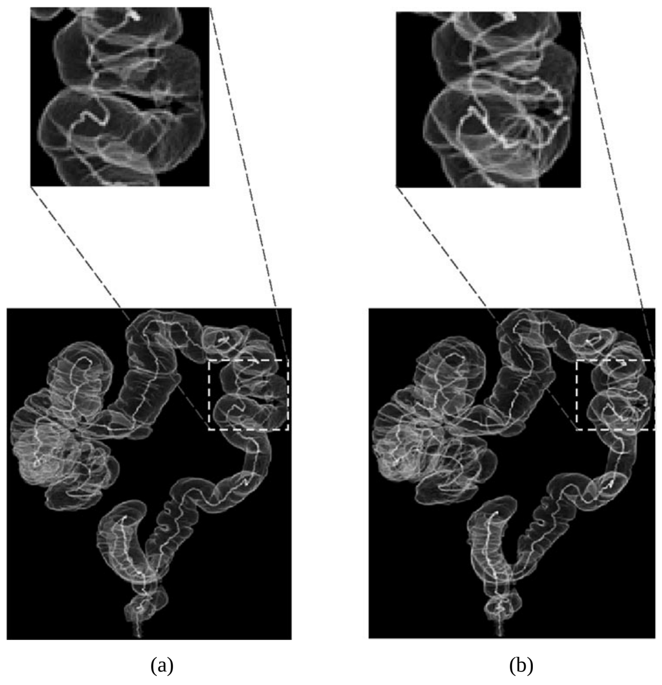


Fig. 7. Touching area detected by branch extraction.

*3) Colon Branch:* We have found that colon branches occur in patient datasets for two different reasons: 1) a deep colon fold presents at a twisted colon location; and 2) the colon walls sometimes come into contact when the colon is squeezed tightly at sharp turns or contacts the small bowel [8] so that shortcut tunnels appear through the touching area, resulting in loops in a tubular colon, as shown in Fig. 7.

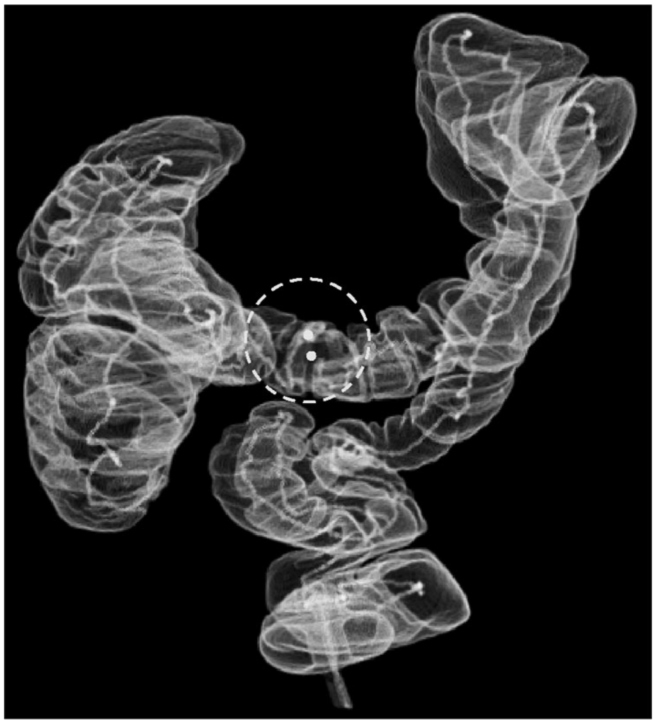


Fig. 8. Patient dataset with two disjointed segments due to colon collapse. The colon tube collapses at the area marked by the dashed circle at the center of the picture. Two dots inside the circle indicate two new end points from these two separated colon segments.

In the first case, our branch-extraction algorithm could ignore the branches by enlarging the user-specified threshold $L$ of the branch length [see Step 3) in Section II-B.2]. For diagnostic purposes, we detect these branches and label them to facilitate navigation.

In the second case, our centerline algorithm is highly likely to ignore these small artificial touching holes due to their high DFB-costs. However, in about 5% of our patient studies, the touching area becomes so large that our centerline takes a shortcut through the hole [see colon area inside the box in Fig. 7(a)]. In these rare cases, our experiment showed that our branch-extraction procedure had successfully detected these shortcuts and extracted the missed parts of the centerline in the skipped colon segments as branches [see Fig. 7(b)]. Since these branches are labeled, they can be easily corrected back to the centerline by inspecting the extracted colon model of Fig. 7. In summary, our centerline and branch-extraction algorithm is able to handle various colon-touching situations. In contrast, the previous centerline algorithms [1], [2], [17], [23], which are based on the shortest path, would always enter those holes regardless of whether they were small or large, because a shortcut always leads to a shorter path.

*4) Colon Collapse:* The colon may collapse due to poor distension or weak colon muscle, resulting in multiple disjointed colon segments. In order to generate the centerline correctly in such a special case, we apply our centerline algorithm on each colon segment and then link them together in an appropriate sequence. Fig. 8 shows a patient dataset with two disjointed colon segments. The two dots in the picture indicate the two ends of the two disjointed centerlines.

The procedure to detect these disjointed centerlines is given by the following three steps.

Step 1) Find the first or the next colon segment to be processed and the start point in that segment. This step distinguishes the first and subsequent segments to be processed. In the beginning, there is no colon segment being processed; we look for the start point of the colon centerline at the bottom of the colon region, as described in Section II-B.1. To locate the next segment, after the first one is processed, calculate the distances from the end point of the centerline of the latest processed colon segment to all the voxels in the remaining unprocessed segments. The voxel with the shortest distance is selected as the next start point, and the segment containing this voxel is the next segment to be processed.

Step 2) Generate a centerline from the start point in the current segment to be processed, then detect all the branches growing from that centerline, using our centerline-extraction algorithm as described in Sections II-B.1 and II-B.2.

Step 3) Go to the first step until no more segments remain.

Note that, although Step 1) is able to automatically detect the next colon segment, it does not guarantee that this would be the actual following segment if the colon tube did not collapse. However, this step is helpful to approach an optimal solution. To further verify the correctness of the extracted centerline segments, the colon model with the centerlines is provided for physician's inspection. The physician can edit it for a corrected sequence.

## III. Results

The presented centerline algorithm was implemented in our VC system on both a low-cost PC platform (with a single Intel Pentium 700-MHz processor and 655 MB of memory) and an SGI Power Challenge (with multiple R10000 processors and 4 GB of memory). The computing time on the PC was about 30% shorter than that on the SGI. We limit the following discussion to the results obtained on the PC platform. Our algorithm was validated, in terms of robustness, speed, and precision, by 44 human colon datasets. Its robustness is demonstrated by correctly extracting all the centerlines and their branches, including those cases with colon touching and collapse. In all the cases, the correct pathlink information about the touching and collapse was verified during automatic flythrough. These results are expected as the theory predicts. Its speed is shown in Table I, where four colon datasets were selected from our experimental results, based on their complexity. A well-prepared bowel has fewer branches and segments (e.g., 3–4), while a less prepared one has more branches and segments (e.g., 7–9). All these four datasets have the longest colons among the 44 tested cases. The computing time is at the second level, rather than the minute level of previously reported methods. Note that the centerline

TABLE I
EXPERIMENTAL RESULTS OF OUR CENTERLINE ALGORITHM ON FOUR COLON DATASETS

| **Data sets** | **Colon 1** | **Colon 2** | **Colon 3** | **Colon 4** |
|---|---|---|---|---|
| Colon size | 512x512x361 | 512x512x389 | 512x512x370 | 512x512x371 |
| Centerline length | 2,079 mm | 1,871 mm | 2,431 mm | 2,231 mm |
| Centerline time | 15.97 sec | 14.63 sec | 11.53 sec | 16.85 sec |
| Branches | 4 | 7 | 3 | 9 |
| Branch time | 3.74 sec | 3.41 sec | 2.66 sec | 4.03 sec |
| Total time | 19.71 sec | 18.04 sec | 14.19 sec | 20.88 sec |

time in Table I includes the time for building the MST tree, but excludes the time for creating the DFB-distance field.[5]

The fast speed of our centerline algorithm is mainly ascribed to its low computational complexity. As we mentioned before, it builds up the MST tree in $O(N \log N)$ time scale, and extracts the centerline and its branches in $O(N)$ time order. Although the typical size of a patient dataset can be as large as 100 million, the number $N$ of voxels inside the colon occupies only about 3%–5% of the total volume. Furthermore, due to the long and narrow shape of the human colon, $(\log N)$ approximates to a much smaller number than $(\log L)$, where $L$ is the maximum number of voxels that actually appear in the sorting heap. For example, $N$ in the Colon1 dataset is three million, but $L$ is 0.5 million—the maximum heap size. As a result, $O(N \log L)$ is only about a quarter of $O(N \log N)$ in this dataset. This results in a sizeable time reduction.

The high centeredness of our centerlines is theoretically derived from the DFB-distance field and was further confirmed by both visual inspection during navigation through the tested 44 cases and mathematical measurements using the DFB-distances. We also compared our new centerlines with the previously generated centerlines using the accurate topological thinning algorithm [12]. The offsets along all the centerlines are less than 0.9 voxel width.

## IV. CONCLUSION AND FUTURE WORK

Based on an analysis of existing centerline algorithms with variable approximations to the well-defined skeleton [3], including tedious manual extraction, accurate topological thinning, and fast distance mapping, we introduce an alternative concise and concrete definition of the centerline, which is the minimum-cost path spanning across the DFB-distance field inside the colon for VC applications. This definition is equivalent to the traditional description of a skeleton [3]. It specifies a highly centered path based on the exact Euclidian distances, and suggests a provably efficient and robust algorithm to extract the colon centerlines without the tendency to hug the corners. Our new automatic centerline algorithm has been demonstrated to be more than ten times faster than the previously reported fastest distance-mapping approaches, while retaining the same (if not better) accuracy as those of the most accurate topological thinning approaches. We described its application to automatic fly-through path planning, endoscopic simulation, and removal of the challenging colon branch and collapse obstacles.

[5]It took about 20 s to create a Euclidean DFB-distance field for a typical colon dataset using the algorithm in [21] on a PC III platform.

We plan to conduct more research on virtually opening the collapsed colon segments by considering a pair of prone and supine CT-scanned colon datasets and using the two centerlines for registration. We will also extend our centerline algorithm to study more complicated human organs with tree structures such as the airways and blood vessels.

## ACKNOWLEDGMENT

The authors would like to thank S. Lakare, K. Kreeger, F. Dachille, M. Sato, D. Chen, and M. Wax for their contribution to the virtual colonoscopy system. The patients' data sets were provided by the University Hospital of the State University of New York at Stony Brook.

## REFERENCES

[1] I. Bitter, M. Sato, M. Bender, K. McDonnel, A. Kaufman, and M. Wan, "CEASAR: A smooth, accurate and robust centerline-extraction algorithm," in *Proc. Visualization 2000*, pp. 45–52.

[2] I. Bitter, A. Kaufman, and M. Sato, "Penalized-distance volumetric skeleton algorithm," *IEEE Trans. Vis. Comput. Graph.*, vol. 3, pp. 195–206, July/Sept. 2001.

[3] H. Blum, "A transformation for extracting new parameter of shape," in *Models for the Perception of Speech and Visual Form*. Cambridge, MA: MIT Press, 1967.

[4] G. Borgefors, "Distance transformations on digital images," *Comput. Vis. Graph. Image Processing*, vol. 34, pp. 344–371, 1986.

[5] D. Chen, B. Li, Z. Liang, M. Wan, A. Kaufman, and M. Wax, "A tree-branch searching multiresolution approach to skeletonization for virtual endoscopy," *Proc. SPIE Med. Imag.*, vol. 3979, pp. 726–734, 2000.

[6] D. Chen, Z. Liang, M. Wax, L. Li, B. Li, and A. Kaufman, "A novel approach to extract colon lumen from CT images for virtual colonoscopy," *IEEE Trans. Med. Imag.*, vol. 19, pp. 1220–1226, Dec. 2000.

[7] R. Chiou, A. Kaufman, Z. Liang, L. Hong, and M. Achniotou, "Interactive fly-path planning using potential fields and cell decomposition for virtual endoscopy," *IEEE Trans. Nucl. Sci.*, vol. 46, pp. 1045–1049, Aug. 1999.

[8] F. Dachille, K. Kreeger, M. Wax, A. Kaufman, and Z. Liang, "Interactive navigation for PC-based virtual colonoscopy," *Proc. SPIE Med. Imag.*, vol. 4321, pp. 500–504, 2001.

[9] E. Dijkstra, "A note on two problems in connexion the graphs," *Numer. Math.*, vol. 1, pp. 269–271, 1959.

[10] Y. Ge, D. Stelts, and D. Vining, "3-D skeleton for virtual colonoscopy," in *Proc. 4th Int. Conf. Visualization in Biomedical Computing (Lecture Notes in Computer Science 0302-9743m)*, 1996, pp. 449–454.

[11] Y. Ge, D. Stelts, J. Wang, and D. Vining, "Computing the centerline of a colon: A robust and efficient method based on 3-D skeleton," *J. Comput. Assist. Tomogr.*, vol. 23, no. 5, pp. 786–794, 1999.

[12] L. Hong, A. Kaufman, Y. Wei, A. Viswambharn, M. Wax, and Z. Liang, "3-D virtual colonoscopy," in *Proc. Symp. Biomedical Visualization*, 1995, pp. 26–32.

[13] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, "Virtual voyage: Interactive navigation in the human colon," in *Proc. SIGGRAPH'97*, pp. 27–34.

[14] S. Lakare, M. Wan, M. Sato, and A. Kaufman, "3-D digital cleansing using segmentation rays," in *Proc. Visualization 2000*, pp. 37–44.

[15] J. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[16] Z. Liang, F. Yang, M. Wax, J. Li, J. You, A. Kaufman, L. Hong, H. Li, and A. Viswambharan, "Inclusion of *a priori* information in segmentation of colon lumen for 3-D virtual colonoscopy," in *Proc. IEEE Nuclear Science Symp.*, vol. 2, 1997, pp. 1423–1427.

[17] W. Lorensen, F. Jolesz, and R. Kikinis, "The exploration of cross-sectional data with a virtual endoscope," in *Interactive Technology and the New Medical Paradigm for Health Care*, R. Satava and K. Morgan, Eds. Washington, DC: IOS Press, 1995, pp. 221–230.

[18] D. Paik, C. Beaulieu, R. Jeffery, G. Rubin, and S. Napel, "Automatic flight path planning for virtual endoscopy," *Med. Phys.*, vol. 25, no. 5, pp. 629–637, 1998.

[19] T. Parkins, "Computer lets doctors fly through the virtual colon," *J. Nat. Cancer Inst.*, vol. 86, pp. 1046–1047, 1994.

[20] T. Pavlidis, "A thinning algorithm for discrete binary images," *Comput. Graph. Image Processing*, vol. 13, pp. 142–157, 1980.

[21] T. Saito and J. Toriwaki, "New algorithm for euclidean distance transformation of an $N$-dimensional digitized picture with applications," *Pattern Recognit.*, vol. 27, no. 11, pp. 1551–1565, 1994.

[22] Y. Samara, M. Fiebrich, A. Dachman, J. Kuniyoshi, K. Doi, and K. Hoffmann, "Automatic calculation of the centerline of the human colon on CT images," *Acad. Radiol.*, vol. 6, pp. 352–359, 1999.

[23] M. Sato, I. Bitter, M. Bender, A. Kaufman, and M. Nakajima, "TEASAR: Tree-structure extraction algorithm for accurate and robust skeletons," in *Proc. 8th Pacific Conf. Computer Graphics and Applications*, 2000, pp. 281–289.

[24] D. Vining, D. Gelfand, R. Bechtold, E. Scharling, E. Grishaw, and R. Shifrin, "Technical feasibility of colon imaging with helical CT and virtual reality," in *Proc. Annu. Meeting American Roentgen Society*, 1994, p. 104.

[25] M. Wan, W. Li, A. Kaufman, Z. Liang, D. Chen, and M. Wax, "3-D virtual colonoscopy with real-time volume rendering," *Proc. SPIE Med. Imag.*, vol. 3978, pp. 165–170, 2000.

[26] M. Wan, F. Dachille, K. Kreeger, S. Lakare, M. Sato, A. Kaufman, M. Wax, and Z. Liang, "Interactive electronic biopsy for 3-D virtual colonoscopy," *Proc. SPIE Med. Imag.*, vol. 4321, pp. 483–488, 2001.

[27] Y. Zhou and A. Toga, "Efficient skeletonization of volumetric objects," *IEEE Trans. Vis. Comput. Graph.*, vol. 5, pp. 196–209, July/Sept. 1999.