ELSEVIER
Theoretical Computer Science 207 (1998) 203–216
Theoretical Computer Science

An optimization problem in virtual endoscopy

Jie Wang[a,*,1], Yaorong Ge[b,2]

[a] *Department of Mathematical Sciences, University of North Carolina at Greensboro, Greensboro, NC 27412, USA*

[b] *Department of Mathematics and Computer Science, and Department of Medical Engineering of the Division of Radiological Sciences, Bowman Gray School of Medicine, Wake Forest University, Box 7388, Winston-Salem, NC 27109, USA*

Abstract

This paper studies a graph optimization problem occurring in virtual endoscopy, which concerns finding the central path of a colon model created from helical computed tomography (CT) image data. The central path is an essential aid for navigating through complex anatomy such as colon. Recently, Ge et al. (1998) devised an efficient method for finding the central path of a colon. The method first generates colon data from a helical CT data volume by image segmentation. It then generates a 3D skeleton of the colon. In the ideal situation, namely, if the skeleton does not contain branches, the skeleton will be the desired central path. However, almost always the skeleton contains extra branches caused by holes in the colon model, which are artifacts produced during image segmentation. To remove false branches, Ge et al. (1998) formulated a graph optimization problem for obtaining the central path. This paper presents a refined formulation and justifies that the solution of the refined optimization problem represents the accurate central path of a colon. We then provide a fast algorithm for solving the problem.

1. Introduction

Virtual endoscopy is a new medical technology that allows physicians to examine computer simulations of patients' anatomy rendered from CT scans. It combines medicine, clinical experience, radiology, image processing, computer algorithms, and applied mathematics to provide the public with alternative medical procedures that are less painful, less costly, and less risky compared to conventional endoscopic procedures. For instance, medical research has shown that small colon polyps, the precursor to colon cancer, can be detected with virtual endoscopy [9, 11].

* Corresponding author. E-mail: wang@uncg.edu.
[1] This work was supported in part by NSF under grant CCR-9424164.
[2] Supported in part by NSF under grant BES-9520388 and by ARPA under grant F41624-96-2-001.

0304-3975/98/$19.00
PII S0304-3975(98)00065-6

When applying virtual endoscopy to complex anatomy such as colon, users often find it difficult to keep track of their position and orientation inside the complex colon image. As a consequence, a part of colon lumen may be left without inspection. Hence, one would like to have a tour guide for traveling through a virtual colon. Finding the central path through the lumen of a colon provides a natural solution. The central path can be used, for example, to create a movie that displays the internal views of the colon lumen generated automatically along the path; and to guide the user to walk through, by using a computer mouse, the colon lumen without getting lost in the virtual space. In addition to aiding navigation, the central path also provides a vital component for more advanced processing and visualization of the virtual anatomy. For example, one can slice the virtual colon into segments of similar length and split each segment into two halves based on a curvilinear cutting plane that passes through the central path, which provides a clear visualization with a single view [10].

Several methods have been proposed to determine the central path of a virtual anatomy. One approach requires that users manually select a number of points along the pathway for constructing a central path [4, 8, 11]. Using this method, users often need to spend considerable amount of time to explore and understand the data volume prior to placing the points, which makes the method less desirable for routine clinical applications. Another approach offers automatic methods [5, 7], but these methods can only find a central path (or a path closed to the center) in the colon lumen for limited cases. These methods fail when the colons have complex shapes. In some cases, a part of the small bowel may also be included in the colon data, which makes the case even more difficult to analyze. To overcome these obstacles, Ge et al. [3] recently devised an efficient method based on the concepts of skeletons. A *skeleton* of a 3D object is the locus of the centers of the largest balls that can fit inside the object; such balls are referred to as *maximally inscribed balls* [1].

Ge et al.'s method consists of four steps. First, it generates a 3D image data volume of a colon from helical CT scans by image segmentation. Second, it generates a 3D skeleton of the colon image by using an improved thinning algorithm based on Li et al.'s thinning algorithm [6]. The thinning process preserves the topological constraints and the geometric constraints of the original object. (The geometric constraints are the two endpoints of the colon provided by the user.) This means that if the skeleton does not contain branches, then it is the desired central path. However, almost always the skeleton contains extra branches caused by holes in the object that are artifacts produced during image segmentation. The number of holes may be reduced by a finer segmentation; but no segmentation, however fine, seems likely to eliminate holes completely. Hence, false branches need to be removed, which is the task of the third step of the method. Note that the image segmentation process may also produce cavities, which corresponds to an isolated point on the skeleton, and hence can be removed easily. After the false branches have been pruned, the remaining skeleton provides the accurate central path. However, the path contains many abrupt direction changes due to the discrete nature of image data. The last step of the method computes a smooth representation of the central path by approximating the final skeleton with B-splines.

To remove the false branches of the skeleton in Step 3, the skeleton is first converted to a connected graph with positive weights, referred to as a *skeletal graph*. Here by connected graph we mean that the graph does not contain isolated vertices. In the skeletal graph, a vertex corresponds to a point on the skeleton where either three or more branches join, or only one branch joins (in this case, the point is an endpoint of the colon), and an edge corresponds to a branch. The weight of each edge is determined as follows. Note that each point on a skeleton branch is associated with a maximally inscribed ball. It has been observed that holes are almost always near the surface [3], and so the extra branches that are induced by these holes usually pass through narrow segments. In other words, the desired central path lies in the center of the colon lumen with a relatively large diameter. Imagine that each point on the skeleton corresponds to a pipe whose size is the maximally inscribed ball. A skeletal branch can then be viewed as a connected sequence of pipes in which balls can roll back and forth. The smallest pipe along the branch determines the largest ball that can pass through this branch. Thus, the radius of the smallest ball along a skeleton branch is used as the weight of its corresponding edge in the graph. In case there are multiple edges between two vertices (this occurs when a skeletal branch splits into two to go around a hole), the edges with smaller weights are removed.

The skeletal graph of a colon has two endpoints. We want to find a path from one endpoint to the other endpoint in the skeletal graph such that its minimum weight is the maximum among all paths, and it avoids passing through any narrow passage whenever it is possible. Such a path corresponds to the true central path of the colon. Ge et al. [3] formulated this optimization problem as a problem of finding a path with the maximum *flow*, where the flow of a path is defined as the minimum weight of all edges on the path. However, in some cases, there may be more than one path with the maximum flow. In such a case, a heuristic was used in [3] for finding a central path among all paths with the maximum flow. We refine this formulation in this paper and fully justify that the solution of the refined optimization problem represents the accurate central path. These results are given in Section 2. In Section 3, we present a $O(n \log n)$-time algorithm for solving the optimization problem on skeletal graphs, as well as mathematical analysis of the algorithm, where n is the number of vertices in a graph. We then present a running example of the algorithm on a complex colon case in Section 4.

2. Finding the central path

We consider connected graphs with positive weights. Since an undirected graph can be simulated by a directed graph, where each edge is represented by two arcs traveling in opposite directions, we assume that all graphs are directed. Denote by $w(u, v)$ the weight of an edge from u to v.

Let G be a skeletal graph (i.e., the weighted graph converted from a skeleton as described in the second last paragraph in Section 1). Let s and t be the two endpoints

in G. In what follows, we will fix the usage of s and t to denote the two endpoints of the colon image. To search for the central path from s to t, we should avoid selecting branches with small weights whenever it is possible. There are two subtle issues in formulating a mathematical definition to capture the essential features of the central path, and we discuss them below.

Issue 1: False branches of large weights. In searching for the central path from s to t, one may want to use the following greedy strategy; namely, at a current vertex u, select the next vertex v with the largest $w(u,v)$. This strategy works fine if all false branches from vertex u have strictly smaller weights than the weight of the true branch from u (here by a true branch it means the branch on the central path). But this strategy fails if a false branch actually has a larger weight than that of the true branch on a particular segment, which may occur due to limitations of segmentation. Although such cases do not always occur, we have encountered such cases during our experiments. Hence, if this greedy strategy is used, then a false branch will be selected, which will then lead us to other false branches.

One may then perhaps suggest that, if such a case happens, one should try to run a finer segmentation and start the whole process over again. However, the segmentation process and the thinning algorithm are extremely time consuming because a large data volume is involved: Each helical CT volume may contain up to 200 MB of data. Hence, this approach is not desirable. Compared to the large data volume of the original image, the size of the skeletal graph is substantially smaller; naturally we should avoid re-running the whole process. Note that selecting a false branch with a large weight may lead to another false branch, which may lead to a false path with a smaller flow than the maximum flow. Hence, we want to find a path with the maximum flow. We note that there may be two or more such paths. If x is a common vertex occurred on such paths, the one that has the largest weight from s to x should be selected.

Let $p=\langle v_0, v_1, \ldots, v_k\rangle$ be a simple path. Denote by $f(p)$ the flow of p; namely,

$$f(p)=\min\{w(v_{i-1},v_i):\ 1\leqslant i\leqslant k\}.$$

We use $v_i \overset{p}{\rightsquigarrow} v_j$ $(j>i)$ to denote the portion of p from v_i to v_j, i.e., the path $\langle v_i,\ldots,v_j\rangle$.

Definition 2.1. Let G be a weighted graph. Let u and v be two vertices. A simple path p from u to v in G is a *largest path* if for any other path p' from u to v, the following two conditions hold.

(1) $f(p')\leqslant f(p)$.

(2) If x is a vertex shared by both p and p', then $f(u \overset{p'}{\rightsquigarrow} x)\leqslant f(u \overset{p}{\rightsquigarrow} x)$.

Issue 2: True branches of fluctuating weights. If there is only one largest path from s to t, then that path represents the accurate central path. However, in some cases, there are two or more largest paths. For example, imagine that a colon image contains some segments that are narrower than the other segments. Then the narrowest segment may determine the flow of the central path. If after this narrowest segment the colon image

(obtained from image segmentation) becomes substantially larger, then a false path may also have the same flow as the flow of the central path, because the maximum flow of all paths is already small. Several heuristics may be used to help identify the accurate central path among the largest paths. For example, one may suggest to find a largest path p from s to t such that for any other largest path p' from s to t, $f(p'_i) \leqslant f(p_i)$ for all i, where p_i (respectively, p'_i) represents the portion on p from the ith vertex to t. This heuristic works for some cases, but it may fail if the colon image has a narrow segment that is followed by a wide segment, and then followed by a narrow segment.

Let e_1 and e_2 be two edges on a path. Denote by $e_1 < e_2$ if e_1 is reached first before e_2 is reached. If e_1, e_2, and e_3 are three edges on a path with $e_1 < e_2 < e_3$ such that $w(e_1) < w(e_2)$, and $w(e_2) > w(e_3)$, then we say that the path has *fluctuating weights.*

The following is a possible heuristic to handle fluctuating weights; namely, find a largest path p from s to t such that for any largest path p' from s to t, if there is an edge e_1 on p and an edge e'_1 on p' with $w(e'_1) > w(e_1)$, then p must contain an edge $e_2 > e_1$ such that for some edge $e'_2 \geqslant e'_1$ on p', $w(e_2) > w(e'_2)$. This heuristic works for some cases, but it may fail in some other cases. For example, assume that e_1 and e_2 are the two edges at the end on the true central path, and e is an edge at the end of a false path such that $w(e) > w(e_1)$ and $w(e) > w(e_2)$, then the true central path will not be selected following this strategy.

To overcome these obstacles, let us imagine that the skeleton of a colon represents the density of the colon. In the ideal situation, namely, if at any point, all false branches have smaller weights than the true branch, then the central path has the heaviest *average weight.* An average weight of a path is the total weight of all edges divided by the number of edges on the path. Note that we cannot use the maximum total weight as a criterion because a false branch may be long and hence its total weight may be large. In a skeletal graph, almost all false branches have small weights. So even if on some segments of a colon, a false branch has a larger weight than that of the true branch, the central path must still have the heaviest average weight.

Let $p = \langle v_0, v_1, \ldots, v_k \rangle$ be a simple path. Denote by $W(p)$ the average weight of p; namely,

$$W(p) = \sum_{i=0}^{k-1} \frac{w(v_i, v_{i+1})}{k}.$$

Definition 2.2. Let G be a weighted graph. Let u and v be two vertices. A simple path from u to v in G is a *heaviest path* if for any other path p' from u to v, $W(p') \leqslant W(p)$.

In some cases a path with the heaviest weight may imply that it is also a largest path. But it is not always true because a heaviest path may contain an edge with very small weight. Hence, we want to find a path that is the heaviest among the largest paths. This gives rise to the following definition.

Definition 2.3. Let G be a weighted graph. Let u and v be two vertices. A simple path p from u to v in G is a *critical path* if p is a largest path from u to v, and for any largest path p' from u to v, $W(p') \leqslant W(p)$.

It is easy to see that if there is a path from u to v, then there must be a critical path from u to v. The following algorithm finds a critical path. First, the algorithm finds all paths with the maximum flow. Second, it finds all largest paths from these paths. Third, it finds a heaviest path from all largest paths. This proves the following lemma.

Lemma 1. *For any weighted graph G and any two vertices u and v, if there is a path from u to v, then there must be a critical path from u and v.*

Recall that for any vertex u in a skeletal graph G, almost always false branches from u have strictly smaller weights than the true branch from u. This implies that if a path is the heaviest among the largest paths, then it represents the accurate central path. Any reasonable segmentation guarantees that such a path is unique. (But we note that in a general weighted graph, there may be more than one critical path.) In practical terms, the central path can be found by solving the following optimization problem.

CENTRAL PATH PROBLEM
Input: A weighted graph G and two vertices s and t.
Output: A critical path from s to t.

We present a fast algorithm for solving this problem in the next section.

3. A fast algorithm

Let $G=(V,E)$ be a connected graph with positive weights. Let s and t be two vertices. We want to find a critical path from s to t. Let

$$\Delta(u,v) = \begin{cases} \max\{f(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there is a path from } u \text{ to } v, \\ -1 & \text{otherwise,} \end{cases}$$

$$\Gamma(u,v) = \begin{cases} \max\{W(p) : u \overset{p}{\rightsquigarrow} v \text{ and } f(p) = \Delta(s,u)\} & \text{if there is a path from } u \text{ to } v, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 2. *Let G be a weighted graph. Let u and v be two vertices. Then a path p from u to v is a critical path if and only if for every vertex x on p, $f(u \overset{p}{\rightsquigarrow} x) = \Delta(u,x)$ and $W(u \overset{p}{\rightsquigarrow} x) = \Gamma(u,x)$.*

Proof. Let p be a critical path. If there is a vertex x on p such that $f(u \overset{p}{\rightsquigarrow} x) \neq \Delta(u,x)$ (i.e., $f(u \overset{p}{\rightsquigarrow} x) < \Delta(u,x)$), then it means that there is another path q from u to x with $f(u \overset{q}{\rightsquigarrow} x) = \Delta(u,x)$. Let p' be the path $u \overset{q}{\rightsquigarrow} x \overset{p}{\rightsquigarrow} v$. Then we have $f(p') \geqslant f(p)$ and $f(u \overset{p}{\rightsquigarrow} x) < f(u \overset{p'}{\rightsquigarrow} x)$, which violates the condition that p is a largest path from u to v. If there is a vertex x on p such that $W(u \overset{p}{\rightsquigarrow} x) \neq \Gamma(u,x)$ (i.e., $W(u \overset{p}{\rightsquigarrow} x) < \Gamma(u,x)$), then it means that there is another path q from u to x with $f(u \overset{q}{\rightsquigarrow} x) = \Gamma(u,x)$, and $f(q) = \Delta(u,x)$. Let p' be the path $u \overset{q}{\rightsquigarrow} x \overset{p}{\rightsquigarrow} v$. Then we have $f(p') \geqslant f(p)$ and $W(u \overset{p}{\rightsquigarrow} x) < W(u \overset{p'}{\rightsquigarrow} x)$, which violates the condition that p is a heaviest path among all the largest paths.

The other direction is straightforward. □

We observe that in a skeletal graph, the number of edges is in the same asymptotic order of the number of vertices. Hence, we use an adjacency list Adj to represent G for saving memory space. Note that although G does not contain isolated vertices, it does not mean that for any pair of vertices u and v, there always exists a path from u to v. But in a skeletal graph G, there is always a path from s to u for any vertex u. We present an algorithm that can also be used to handle non-connected graphs. For each vertex $u \in V$, we maintain four attributes $d[u]$, $\pi[u]$, $l[u]$, and $a[u]$ in the algorithm, where $d[u]$ represents the flow from s to u, $\pi[u]$ returns the predecessor of u on the current path, $l[u]$ represents the number of edges from s to u on the current path through the π attributes, and $a[u]$ represents the average weight of the current path from s to u.

```
CRITICAL_PATH(G, s, t)
1. for each vertex u ∈ V do
       d[u] ← −1, π[u] ← NIL, l[u] ← 0, a[u] ← 0
   endfor
   d[s] ← 0, S ← ∅, Q ← V
2. while Q ≠ ∅ do
   (a) u ← EXTRACT_MAX(Q)
       if d[u] ≠ −1 then S ← S ∪ {u} else goto Step 3
   (b) for each vertex v ∈ Adj[u] do
           if (d[v] < min{d[u], w(u,v)}) then
               d[v] ← min{d[u], w(u,v)}
               π[v] ← u, l[v] ← l[u] + 1
               a[v] ← (l[u] · a[u] + w(u,v))/l[v]
           if (π[v] ≠ NIL and min{d[u], w(u,v)} ⩾ d[v]) then
               if ((l[u] · a[u] + w(u,v))/(l[u] + 1) > a[v]) then
                   π[v] ← u, l[v] ← l[u] + 1
                   a[v] ← (l[u] · a[u] + w(u,v))/l[v]
       endfor
   endwhile
```

3. **if** $t \in S$ **then** output the path from s to t using the π attributes
   else there is no path from s to t

Part 1 of CRITICAL_PATH(G,s,t) is for initialization. The procedure EXTRACT_MAX(Q) in 2(a) finds an element $u \in Q$ that has the largest $d[u]$. For each vertex v in $Adj[u]$, if v has not been visited, namely, $\pi[v] = \text{NIL}$, then the first **if**-statement of 2(b) updates $d[v]$. If v has been visited, then the second **if**-statement of 2(b) checks whether the current path has a flow at least as large as the flow of the old path. If the answer is yes, the third **if**-statement of 2(b) checks whether the current path has a strictly larger average weight than that of the old path; if the answer is yes, the old path is flipped over to the current path by changing the π attribute. The formula $(l[u] \cdot a[u] + w(u,v))/(l[u]+1)$ calculates the average weight of the current path from s to v via u.

Lemma 3. *When the algorithm* CRITICAL_PATH(G,s,t) *terminates, we have* $d[u] = \Delta(s,u)$ *for all vertices* $u \in V - \{s\}$. *Moreover, if* $\pi[u] \neq \text{NIL}$, *the path* p *from* s *to* u *obtained from the* π *attributes is a largest path.*

Proof. Initially, $d[s] = 0$, and $d[u] = -1$ for all $u \in V - \{s\}$. Hence, vertex s must be the first to be chosen by the EXTRACT_MAX operation. For any vertex $u \in V - \{s\}$, if there is no path from s to u, then u is never inserted in S, and the value of $d[u]$ remains as -1, which is equal to $\Delta(s,u)$.

We will show that for each vertex $u \in V - \{s\}$, we have $d[u] = \Delta(s,u)$ at the time when u is inserted into S and this equality remains true thereafter. We prove it by contradiction. Let u be the first vertex with $d[u] \neq \Delta(s,u)$ when it is about to be inserted in S. Then there must be a path from s to u; for otherwise, $d[u] = -1 = \Delta(s,u)$, which would violate our assumption that $d[u] \neq \Delta(s,u)$. Note that at this moment, u has not been inserted in S yet, and so $u \in V - S$. Since there is a path from s to u, there must be a path p from s to u such that for every v on p, we have $f(s \overset{p}{\rightsquigarrow} v) = \Delta(s,v)$.

We write $x \to y$ to denote an edge from x to y, and write $x \to y \in p$ to denote that $x \to y$ is an edge on p. Let y be the first vertex on p that is not in S. Let x be y's predecessor. Then path p can be decomposed into $s \overset{p_1}{\rightsquigarrow} x \to y \overset{p_2}{\rightsquigarrow} u$, where x may be equal to s, y may be equal to u, and path p_2 may or may not re-enter S.

We claim that $d[y] = \Delta(s,y)$ when u is inserted in S. If $x = s$, then it is obvious that $d[y] = \Delta(s,y)$ from the algorithm. Assume that $x \neq s$. Since y is the first vertex on p that is not in S, we have $x \in S$. Since u is the first vertex with $d[u] \neq \Delta(s,u)$ when it is inserted in S, this means that $d[x] = \Delta(s,x)$ when x was inserted in S. By assumption, the path $s \overset{p_1}{\rightsquigarrow} x \to y$ has flow $\Delta(s,y)$ and p_1 has flow $\Delta(s,x)$. Hence, the flow of $s \overset{p_1}{\rightsquigarrow} x \to y$ is equal to $\min\{\Delta(s,x), w(x,y)\}$. Since $d[x] = \Delta(s,x)$ and $d[y]$ is computed when x is inserted in S (see the first **if**-statement of Step 2(b)), where $d[y] = \min\{d[x], w(x,y)\}$, we have $d[y] = \Delta(s,y)$.

We now show that for any vertex $u \in V - \{s\}$, if there is a path from s to u, then the value of $d[u]$ will never be changed once $d[u]$ is assigned the value $\Delta(s,u)$. We note that $d[u]$ never decreases, so it suffices to show that $d[u] \leqslant \Delta(s,u)$. This

inequality is certainly true after the initialization. Suppose this inequality were not true during some of the steps of the algorithm. Let v be the first vertex with $d[v] > \Delta(s,v)$; namely, for any z that is visited before v, we have $d[z] \leqslant \Delta(s,z)$. We note that by definition, for any z with $z \to v$, $\Delta(s,v) \geqslant \min\{\Delta(s,z), w(z,v)\}$. By the algorithm, there must be a vertex z such that $d[v] = \min\{d[z], w(z,v)\}$. Since $d[z] \leqslant \Delta(s,z)$, we have $d[v] \leqslant \min\{\Delta(s,z), w(z,v)\} \leqslant \Delta(s,v)$, a contradiction.

Thus, $d[u] \leqslant \Delta(s,u) = \min\{w(a,b) : a \to b \in s \overset{p_1}{\rightsquigarrow} x \to y \overset{p_2}{\rightsquigarrow} u\} \leqslant \Delta(s,y) = d[y]$. On the other hand, since u was chosen before y, we have $d[u] \geqslant d[y]$, and so $d[u] = d[y] = \Delta(s,u)$, which contradicts the assumption that $d[u] \neq \Delta(s,u)$.

Once u is inserted, $d[u]$ remains unchanged, although its π attribute may be changed. But the π attribute is changed to a vertex u' only when $\min\{d[u'], w(u',u)\} \geqslant d[u]$. Hence, the final π attributes provides a path p' from s to u such that $f(p') = d[u] = \Delta(s,u)$. This completes the proof. □

Lemma 4. *When the algorithm* CRITICAL_PATH(G,s,t) *terminates, we have* $a[u] = \Gamma(s,u)$ *for all vertices* $u \in V - \{s\}$. *Moreover, if* $\pi[u] \neq$ NIL, *the path* p *from* s *to* u *obtained from the* π *attributes is a critical path.*

Proof. For any vertex $u \in V - \{s\}$, if there is no path from s to u, then u is never inserted in S, and the value of $a[u]$ remains as 0, which is equal to $\Gamma(s,u)$. If there is a path from s to u, we will show that $a[u] = \Gamma(s,u)$. We prove it by contradiction. Assume that $a[u] \neq \Gamma(s,u)$. Let p be the simple path from s to u returned from the π attributes. Note that when $a[u]$ is updated, $l[u]$ is also updated, which always returns the length of the path from s to u obtained from the π attributes. Let x be the first vertex on p from s such that $a[x] \neq \Gamma(s,x)$, namely $a[x] < \Gamma(s,x)$. This means that there must be another path q from s to x such that q is a largest path and $W(q) > W(s \overset{p}{\rightsquigarrow} x)$. Let x' be the predecessor of x on q. Then x' is not in p; for otherwise, since x is the first vertex on p with $a[x] < \Gamma(s,x)$, we have $a[x'] = \Gamma[s,x']$, which implies that $a[x] = \Gamma(s,x)$, a contradiction. If $a[x'] = \Gamma(s,x')$, then we have $\min\{d[x'], w(x',x)\} = d[x]$ and $(l[x'] \cdot a[x'] + w(x',x))/(l[x'] + 1) > a[x]$, which implies that $\pi[x] = x'$, a contradiction. So $a[x'] < \Gamma(s,x')$. We repeat the above argument, and obtain another vertex x'', where x'' is the predecessor of x', such that $a[x''] < \Gamma(s,x'')$ and $x'' \notin \{x,x'\}$. This process can be repeated and eventually some vertex y is reached with $a[y] < \Gamma(s,y)$, and y has a unique predecessor s, because there is only a finite number of vertices in G. But then, $a[y] = w(s,y) = \Gamma(s,y)$, a contradiction. Hence, for all $u \in V - \{s\}$, $a[u] = \Gamma(s,u)$. By Lemma 3, p is a largest path, and so p is a critical path from s to u. This completes the proof. □

It follows from Lemma 3 that the algorithm CRITICAL_PATH(G,s,t) returns a critical path from s to t. Namely, we have proven the following theorem.

Theorem 5. *Assume that there is a path from* s *to* t *in* G, *then when the algorithm* CRITICAL_PATH(G,s,t) *terminates, the path from* s *to* t *obtained from the* π *attributes is a critical path.*

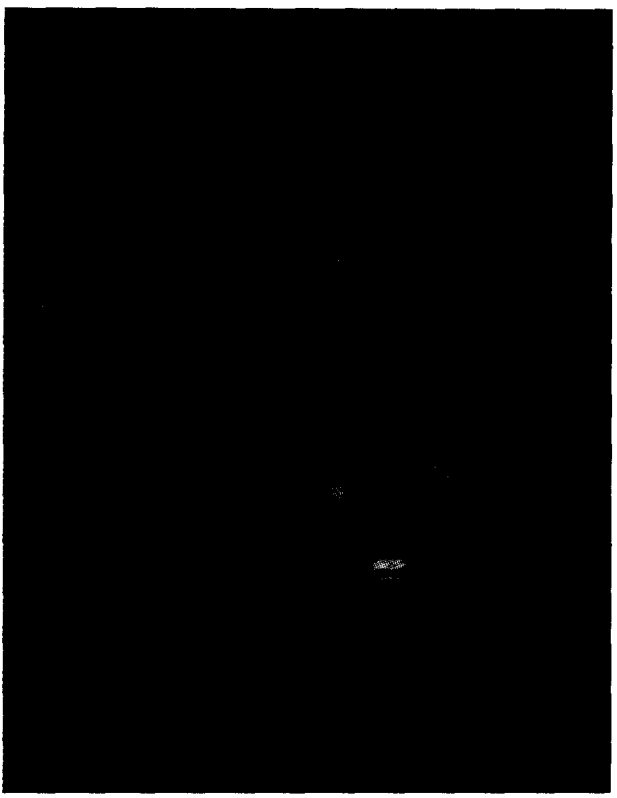

Fig. 1. Rendering of the segmented colon. Touching segments of colon and inclusion of a portion of small bowel are due to limitations in the segmentation process.

Next, we analyze the time complexity of the algorithm.

Theorem 6. *Depending on how the* EXTRACT_MAX *operation is implemented, the algorithm* CRITICAL_PATH(G,s,t) *runs in time* $O(n^2)$ *or* $O((n+|E|)\log n)$, *where* $|V|=n$. *Hence, when* $|E|=O(n)$, *the algorithm can be run in* $O(n\log n)$ *time.*

Proof. Suppose we implement the priority queue $Q=V-S$ as a linear array. Then EXTRACT_MAX operation takes time $O(|V|-|S|)\leqslant O(|V|)$. There are $|V|$ such operations, and so the total EXTRACT_MAX time is $O(|V|^2)$. Each vertex $v\in V$ is inserted in S exactly once, so each edge in the adjacency matrix list $Adj[v]$ is examined in the **for**-loop in Step 2(b) exactly once during the course of algorithm. Since the total number of edges in all the adjacency lists is $|E|$, there are at most $|E|$ iterations of this **for**-loop, with each iteration taking $O(1)$ time. The running time of the entire algorithm is therefore $O(|V|^2+|E|)=O(|V|^2)$.

Suppose we implement the priority queue Q with a binary heap. Then each EXTRACT_MAX operation takes time $O(\log|V|)$. As before, there are $|V|$ such executions.

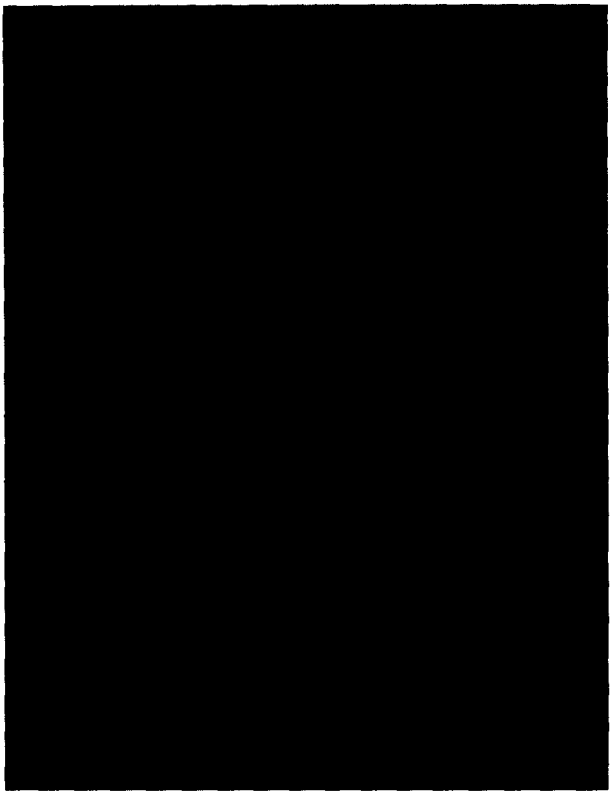

Fig. 2. The initial skeleton preserves the original colon topology. The many extra branches that deviate from the center of the colon are caused by holes in the original object.

The time to build the heap is $O(|V|)$. The assignment $d[v] \leftarrow \min\{d[u], w(u,v)\}$ can be accomplished by increasing the key for v in the heap, which takes time $O(\log|V|)$. The other operations each take $O(1)$ time on a single execution. There are at most $|E|$ such operations. Hence, the total running time is $O((|V| + |E|)\log|V|)$. When $|E| = O(|V|)$, this yields $O(|V|\log|V|)$ time, resulting a large saving than the previous implementation.

Since in a skeletal graph G, $|E| = O(|V|)$, it follows from Theorems 5 and 6 that finding the central path of the colon lumen can be carried out in $O(n\log n)$ time, where $|V| = n$.

Remark. The algorithm CRITICAL_PATH can also be applied to non-connected graphs with non-negative weights. Also, on a general weighted graph, if we implement the priority queue Q with a Fibonacci heap, then the algorithm CRITICAL_PATH can be run in time $O(|V|\log|V| + |E|)$. The reference [2] provides details of Fibonacci heap implementation.

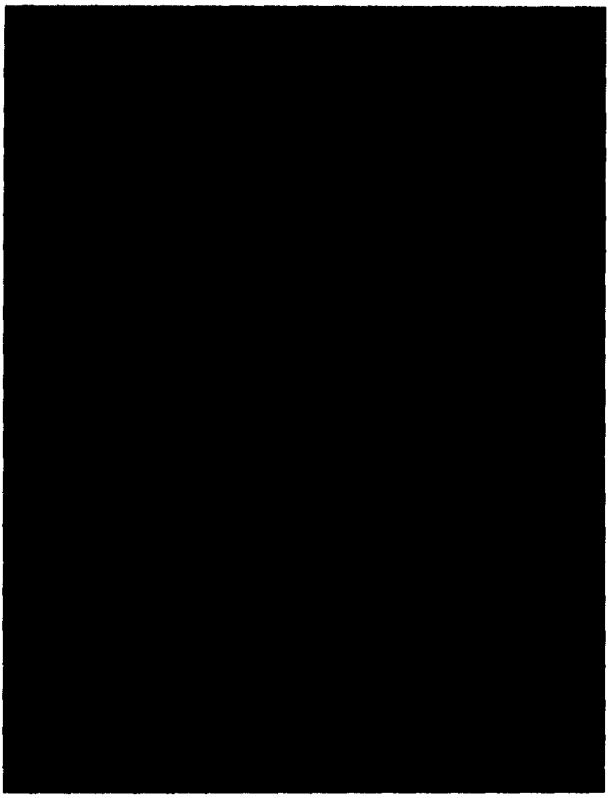

Fig. 3. The central path and its smooth B-spline approximation. The dotted line represents the central path from the original skeleton and the solid line is its B-spline approximation.

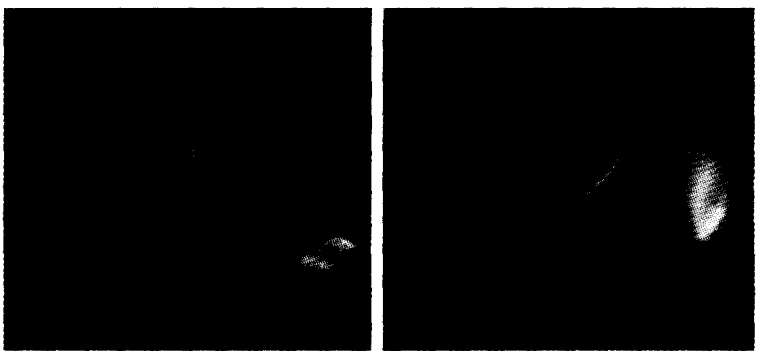

Fig. 4. Two internal renderings of the colon lumen as seen from positions along the central path. The central path is projected into the internal views for illustration.

4. Running examples

We have tested our algorithm on a number of virtual endoscopy colon cases. Each helical CT volume consists of up to 500 images with 512×512 pixels per image. We present in this section a running example of our algorithm on a complex colon case.

Fig. 1 shows a colon that has been segmented from a CT volume and rendered for visualization. Notice that the segmentation result is inexact. A large portion of small bowel has been segmented in addition to the colon. We use this example to demonstrate the robustness of our algorithms.

The skeleton resulting from the 3D thinning algorithm is shown in Fig. 2, superimposed on the original colon rendering. Notice that multiple colon segments touch each other, and that a portion of the small bowel touches the transverse colon. These touching segments result in segmentation artifacts known as holes and causes many extra branches in the skeleton.

Fig. 3 shows the true central path extracted from the initial skeleton (dotted line) and its B-spline approximation (solid line).

Fig. 4 shows two internal renderings of the colon lumen from two positions on the central path. Note that these are views of the colon surface from inside. The small holes that we see in these views are not the holes in the segmented colon object. Rather, they are artificial tunnels connecting colon segments that create holes visible from external views. These tunnels are where the extra skeletal branches pass through in order to go around the holes in the object.

The time required to execute our algorithms varies with the type of platform, the size of input volume, and the complexity of the colon itself. On an SGI System with R10000 (Silicon Graphics, Inc., Mountain View, CA), the time required to convert a skeleton to a graph and to search for the true central path never exceeded 15 seconds.

Acknowledgements

We thank Dr. David Vining at Wake Forest University School of Medicine for providing the clinical context of this research and for supplying the data sets to verify our results. We also thank Xiangjun Wu and Xiangliang Zha for implementing part of the algorithm presented in this paper.

References

[1] H. Blum, Biological shape and visual science: part I, J. Theoret. Biol. 38 (1973) 205–287.
[2] T.H. Cormen, C.L. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.
[3] Y. Ge, D.R. Stelts, X. Zha, J. Wang, D.J. Vining, Computing the central path of colon lumen in Helical CT images, SPIE International Symposium on Medical Imaging 1998: Image Processing. In press.
[4] A. Hara, C. Johnson, J. Reed, R. Ehman, D. Ilstrup, Colorectal polyp detection with CT colography: two- versus three-dimensional techniques, Radiology 200 (1996) 49–54.

[5] L. Hong, A. Kaufman, Y.-C. Wei, 3D virtual colonoscopy, in: Proc. 1995 IEEE Biomedical Visualization Symp., pp. 1995, 26–32.
[6] T.-C. Lee, R. Kashyap, C.-N. Chu, Building skeleton models via 3D medial surface/axis thinning algorithms. CVGIP: Graphical Models and Image Processing 56 (1994) 462–478.
[7] W. Lorensen, F. Jolesz, R. Kikinis, United states patent: virtual internal cavity inspection system, Tech. Rep. 5611025, 1997.
[8] R. Shahidi, V. Argiro, S. Napel, L. Gray, H. McAdams, G. Rubin, C. Beaulieu, R. Jeffrey, A. Johnson, Assessment of several virtual endoscopy techniques using computed tomography and perspective volume rendering, in: Karl Heinz Hohne, Ron Kikinis (Eds.), Lecture Notes in Computer Science: Visualization in Biomedical Computing, vol. 1131, Springer, Berlin 1996, pp. 521–528.
[9] D. Vining, D. Gelfand, R. Bechtold et al., Technical feasibility of colon imaging with helical CT and virtual reality, Amer. J. Roentgenology 162 (1994) 104.
[10] D. Vining, D. Stelts, G. Hunt, D. Aho, Y. Ge, P. Hemler, Technical improvement in virtual colonoscopy, Radiology 201 (1996) 524.
[11] D. Vining, R. Shifrin, Virtual reality imaging with helical CT, Amer. J. Roentgenology 162 (1994) 188.