CEASAR: A Smooth, Accurate and Robust Centerline Extraction Algorithm

Ingmar Bitter Mie Sato Michael Bender Kevin T. McDonnell
Arie Kaufman Ming Wan
Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794–4400, USA *

Abstract

We present CEASAR, a **c**enterline **e**xtraction **a**lgorithm that delivers **s**mooth, **a**ccurate, and **r**obust results. Centerlines are needed for accurate measurements of length along winding tubular structures. Centerlines are also required in automatic virtual navigation through human organs, such as the colon or the aorta, as they are used to control movement and orientation of the virtual camera. We introduce a concise but general definition of a centerline, and provide an algorithm that finds the centerline accurately and rapidly. Our algorithm is provably correct for general geometries. Our solution is fully automatic, which frees the user from having to engage in data preprocessing. For a number of test datasets, we show the smooth and accurate centerlines computed by our CEASAR algorithm on a single 194 MHz MIPS R10000 CPU within five minutes .

1 Motivation

Many aspects of complicated 3D shapes are often better understood and handled when the shapes are reduced to their 1D centerlines. For example, automatic virtual navigation through a human colon [6] uses the colon centerline to control the movement and orientation of the virtual camera. Similarly, accurate length measurements and navigation through other tubular human organs such as the aorta require centerline computations. In addition, finding an optimal path of minimal collision-probability [5] through tubular structures in virtual engineering and architectural designs also poses the same centerline finding problem.

In this paper we find centerlines in binary discretized 3D occupancy maps of tubular structures. We use segmented medical CT and MRI scans as our input data. However, the techniques that we develop are general and may be readily applied to other domains, because our assumptions are not specific to the source of the data.

2 Overview of Centerline Algorithms

The intuitive notion of a centerline of a 3D object is the central path through that object. Surprisingly, even when restricting the shape to non-treelike tubular structures such as a human colon, it is challenging to construct a formal mathematical definition of its centerline. There has been extensive work on this topic. We summarize here traditional centerline algorithms along with their concepts of what a centerline should be.

All centerline algorithms assume that the data is presented as a 3D rectilinear grid called a *volume* [8] of volumetric sample points called *voxels* [8]. Two voxels are *6-connected* if at most one of their 3D coordinates differs by 1, *18-connected* if at most two coordinates differ by 1, and *26-connected* if all three coordinates are allowed to differ. A *6/18/26-connected* path through this data is a sequence of 6/18/26-connected voxels. A *discrete centerline* is such a path whereas a *continuous centerline* is an unrestricted continuous curve in 3D space.

There have been many methods for creating a continuous centerline from a discrete centerline by using interpolating and approximating curves [1, 3, 4, 7, 10]. All these methods employ regularly spaced discrete voxel positions as the control points. Consequently, a large number of control points is required to ensure the high accuracy needed to keep the centerline inside the colon in narrow colon regions. However, in wide colon sections larger differences between the continuous and discrete centerlines are acceptable. This arrangement is even preferred, if it uses fewer control points, and thus generates a smoother path and requires fewer computational resources.

2.1 Radiologist Markings

The most basic recipe for finding the centerline of a colon is to defer to the expertise of a radiologist [3]. In this least automated method, the radiologist is provided with a sequence of 2D cross sections of the colon. On each cross section the radiologist manually marks the "center" of the cross section. Then, all of the centers of cross sections are connected to form a path, which is then defined to be the centerline.

Unfortunately, this method has some severe practical and fundamental drawbacks. Practically, it is expensive to rely on the radiologist to click on the centers of all cross sections. Figure 1 explains the more fundamental drawback. Placing markings that are optimal in 2D cross sections is insufficient to create an optimal centerline in 3D. In fact, this method may lead to paths that are non-centered and even penetrate through the colon wall.

2.2 DSF and Dijkstra Shortest Path

Many centerline algorithms use the Dijkstra shortest path graph algorithm [2] as an intermediate step. The Dijkstra algorithm provably finds the global minimal weight path in an undirected weighted graph with non-negative weights. The algorithm has two phases. The first creates a **d**istance from **s**ource **f**ield (DSF) by labeling all graph vertices with the shortest distance from a single source to those vertices. The second phase creates the shortest path by tracing back to the source node. Note that this backtrace is not the same as the

*{ingmar, mie, bender, ktm, ari, mwan}@cs.sunysb.edu

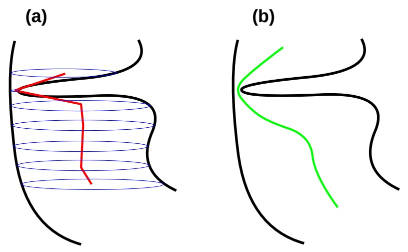


Figure 1: *(a) Colon penetrating centerline found by Radiologist markings. (b) A better centerline is shown that can not be found using only 2D context.*

steepest descent in the DSF. In order to apply the Dijkstra algorithm to our centerline algorithm, the volume data has to be transformed into a graph. We implicitly map voxels to graph vertices and voxel neighbor relations to graph edges (for more details see Section 3.2 and Figure 4a).

The centerline algorithms using steepest descent or Dijkstra's method differ in how they assign the weights corresponding to orthogonal, 2D-diagonal, and 3D-diagonal vertex neighbor relations. The algorithms employ the 1-0-0 Manhattan metric [15] (only orthogonal neighbors), the 1-2-3 metric [18] (also diagonal neighbors, but with higher distance weights), the 3-4-5 Chamfer metric [3], or the 10-14-17 metric [1]. The metrics are sorted by decreasing error when compared with the Euclidian distance between the voxel positions. It is most accurate to use a 1-$\sqrt{2}$-$\sqrt{3}$ Euclidian metric for isotropic volumes and one with axis specific corrections for anisotropic volumes. This is the approach adopted in this paper.

Independent of the choice of the metric, the resulting shortest path visits vertices of the graph, and is therefore guaranteed to reside inside the colon. Unfortunately, this path tends to cut the corners and travel along boundary voxels on the inside of sharp turns. Hence, this path generally does not qualify as centered.

A method for reducing this cutting of corners is to replace the Dijkstra backtrace path with a path along the centers of mass of clusters with similar DSF values [15]. This technique would work well if the "wave fronts" formed by cluster of voxels of the same DSF value were always perpendicular to the centerline. Unfortunately, near sharp turns the wave fronts tilt and can be even parallel to the intuitive centerline (for more details see Figure 11).

2.3 Distance from Boundary Field

A slight modification to the first phase of Dijkstra's algorithm is to replace the single source voxel with the set of all boundary voxels. The result is a distance field that stores for each voxel the length of its shortest discrete path to the boundary. Again a variety of distance metrics for edge weight assignments is possible.

This **d**istance from **b**oundary **f**ield (DBF) can be used to improve the centrality of the centerline by relocating centerline candidate points to the maximal point of the DBF within the plane perpendicular to the centerline [1]. However, a single correcting step does not yield an optimal centerline and even iterating this method is not guaranteed to find a global optimum.

A better approach is to relocate centerline point candidates to the maximal DBF voxel within the "wave front" of same DSF values [18]. However, this disconnects the candidate centerline and stitching it back together is based on local heuristics.

2.4 Topological Thinning

The technique that is traditionally considered to provide high quality results is called topological thinning or "onion peeling" [3, 6, 11, 12]. In this general strategy, one layer of voxels at a time is peeled off the colon until just the centerline remains. Multiple invariants should be maintained to avoid errors. The starting and ending voxels can not be removed and must remain part of the same connected component, and the topology must be preserved. No voxel can be removed that would cause these constraints to be violated.

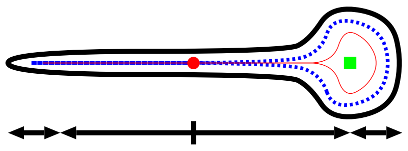


Figure 2: *Performance of topological thinning on a banjo-shaped colon. The displayed cross section of the colon depicts a long neck attached to a body. The onion peeling algorithm determines that the centerline traverses through the circle in the neck, whereas a more intuitive location would be through the square in the body itself.*

Unfortunately, onion peeling is computationally expensive. Much research effort has been devoted to the problem of how to speed up the basic topological thinning algorithm. The highest improvement stems from separating the thinning phase from the connectivity preserving considerations [11]. The main idea here is to first determine a rough candidate centerline, then perform one step of parallel, unrestricted thinning and finally computing the union of the remaining shape and the candidate centerline. This guarantees connectivity and, when iterated, finds a solution that is very close to or possibly identical to the normal onion peeling algorithm. However, there is no concise mathematical formulation of what the onion-peeled centerline should look like. In fact, there are examples, such as the banjo-shaped colon in Figure 2, in which the onion peeling algorithm does not find the intuitively desired centerline.

2.5 Centers of Maximally Inscribed Balls

Local maxima in the distance from boundary field can geometrically be viewed as centers of maximally inscribed balls. If the center points are moved, the balls must shrink in size to remain inside the colon. Intuitively, all of these points belong on the centerline. However, defining the centerline as the union of these points is insufficient, as they typically form a disconnected set. Ge et al. [3] therefore extended the class of voxels that can not be removed during topological thinning to include these centers of maximally inscribed balls. With this extra constraint, onion peeling does determine the intuitively desired centerline for a banjo-shaped colon cross section like in Figure 2, but, fails on other shapes. For example, Figure 3 depicts a colon with a flower shaped cross section that causes excessive winding of the centerline. Here the intuitive centerline would exclude the centers of maximally inscribed balls in the folds.

3 Formal Centerline Definition

In this paper we introduce a concise but general definition of a centerline. We then present an algorithm that can accurately and rapidly produce such a centerline. We provide a fully automatic solution, which frees the radiologist from having to participate in the data preprocessing. Our centerline algorithm is designed to be *provably robust*. It is guaranteed to perform correctly even for a winding, twisted colon with large folds and bulgings.

3.1 Formal Colon/Tubular Object Definition

It is important to specify exactly which assumptions are made about the shape of the colon or a more general tubular object. Once these assumptions are formally defined, we have a set of conditions under which the algorithm is guaranteed to run correctly. Our assumptions are minimal and apply to many other domains in which one might want to find a centerline.

A colon is assumed to have the following properties.

- The colon is a singly-connected component.
- The colon has genus zero, that is, there are no holes or tunnels through the colon.
- Intuitively, the colon is *long* and *narrow*. More formally, these conditions are described as follows. There exist two disjoint voxel sets: a start set $\mathcal{S}$ and an end set $\mathcal{E}$, such that:

  i *There are two disjoint ends that are far apart.*
  That is, for all voxels x and y in the colon of maximal shortest path length ℓ and a factor $\alpha \in (.9, 1)$, if the distance between x and y is sufficiently large, then x and y must be in $\mathcal{S} \cup \mathcal{E}$:
  $d(x,y) > \alpha\ell \Rightarrow x, y \in \mathcal{S} \cup \mathcal{E}$.

  ii *The ends $\mathcal{S}$ and $\mathcal{E}$ of the colon are narrow.*
  That is, $x, y \in \mathcal{S} \Rightarrow d(x,y) < \frac{\ell}{10}$, and analogous for $\mathcal{E}$.

  iii *The intermediate sections of the colon are narrow.*
  That is, for any x in the colon, for all y that maximize $d(x,y)$, $y \in \mathcal{S} \cup \mathcal{E}$.

This definition allows us to prove the correctness of the part of our centerline algorithm that automatically finds both colon ends (see Section 4.7).

3.2 Formal Centerline Definition

We now describe some basic properties a centerline should have. Most importantly, the centerline should be a simple path without any 2D manifolds extending from one end of

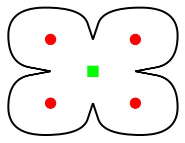


Figure 3: *A centerline that is forced to include all centers of maximally inscribed balls, does not only traverse the desired square in the center of the cross section of a flower shaped colon, but also the circles in the petals of the cross section. This causes excessive winding of the centerline.*

the colon to the other. The centerline should never leave the inside of the colon. More specifically, the centerline should tend to remain in the "center" of the colon. For winding and bulging colons, the concept of center may not be well defined. Intuitively, the centerline should be situated as far from the boundary as possible. On the other hand, it should also avoid too much winding because the centerline should be as short as possible within all other constraints. This suggests that our algorithm should find some kind of shortest path through the colon.

As pointed out in Section 2.2, the Dijkstra shortest path algorithm requires volume data to be mapped to graph vertices and graph edges. Figure 4a depicts a straightforward implicit mapping. Edges represent the 26-neighbor relations between voxels. As weights, we use the exact Euclidian distances between the voxels that correspond to the graph vertices at both ends of the edge. However, even when including corrections for anisotropic volumes, an unembellished shortest path through the colon has the defect that when it turns, it cuts the corners, instead of staying near the center. Therefore we enhance the implicit graph by adding more edges and vertices as depicted in Figure 4b to incorporate penalties for coming close to the colon boundary and to create a **p**enalized **d**istance from **e**nd **f**ield (PDEF). There are now 27 vertices per voxel: one center vertex and 26 penalty vertices that each share a penalty edge with the center vertex. The penalty edges have a weight equal to half the penalty associated with including that voxel into the path. Neighbor edges now always connect to penalty vertices. Since this modification results in a graph that is a singly connected component where edges have positive weights, it is guaranteed that the Dijkstra algorithm will find the globally minimal shortest path. The cost along that shortest path is the accumulated piecewise Euclidian distance of the path plus the sum of the penalties of all penalty edges visited along the path.

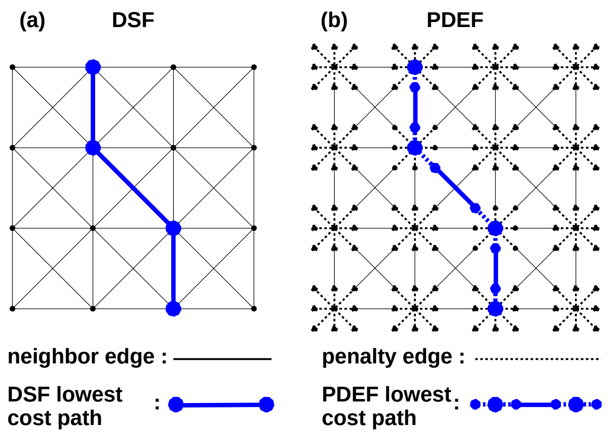


Figure 4: *2D top view of the implicit mapping of the voxel grid and neighbor relations to an undirected graph on which the Dijkstra algorithm can be applied. (a) A region of a "plain" distance field with a part of a shortest path. (b) The same region, but now with penalties. Solid neighbor edges have weights equal to the distance between the voxels. Dashed penalty edges have a weight equal to half the penalty assigned to the associated voxel.*

We define the centerline to be the minimum cost path found in the penalized distance field. This definition has the following concrete advantages: It is precise, rapidly computable, and suggests a provably correct algorithm. It does not require any specific geometry in order to run correctly. Naturally, there is a range of penalties that may be applied to the penalty edges defining a family of continuously varying centerlines. In Section 4.8 we suggest a choice of penalty function that yields a high-quality, centered path.

4 CEASAR Algorithm

The CEASAR algorithm works for any elongated or tubular structure. Examples of such structures are pipes, tunnels, blood vessel segments, and colons. In this paper, we concentrate on the colon example, because it is arguably the most complicated tubular shape. The algorithm consists of ten logical steps :

1. Read a binary segmented colon
2. Crop volume to just colon
3. DBF: Compute distance from boundary field
4. GVF: Compute gradient vector field
5. Flag Non-uniform gradient neighborhoods
6. Connect flagged voxels
7. DAF: Compute distance from any flagged voxel field
8. PDEF: Compute penalized distance from end voxel field
9. Extract minimum cost path from PDEF
10. Smooth centerline

We now describe these steps in detail.

4.1 Binary Segmented Colon

The input for our CEASAR algorithm is a binary mask, which labels the voxels belonging to the colon interior and colon wall. This mask is generated from the CT scan using a segmentation algorithm [9]. This algorithm ensures that the colon is one connected component, that there are no bubbles or holes in the colon, and that folds from different parts of the colon are separated by at least one voxel.

4.2 Crop Volume to Just Colon

One essential component of CEASAR it that it is computational efficient. Thus, in each step we strive to minimize the number of voxels that have to be processed. The first step in reducing the number of relevant voxels is to crop the volume automatically to just the bounding box of the labeled colon voxels. For our abdominal CT scans, this step reduces the volume size by 30-50%.

4.3 DBF: Compute Distance from Boundary Field

The Euclidian distance between an inside voxel and the colon boundary is recorded at each voxel. This forms the distance from boundary field (DBF). We use a four pass algorithm by Saito [14] that is linear in the number of voxels to compute the real Euclidian DBF accurately.

4.4 GVF: Compute Gradient Vector Field

The next two CEASAR algorithm steps combined deliver a large reduction in the number of voxels that have to be processed. The algorithm would work without these, but the subsequent steps would be slower.

For each volume position within the colon mask, we compute its central difference gradient, which requires reading of only six neighboring voxels. This forms the gradient vector field (GVF) of the DBF (see Figure 6). (We also experimented with the smoother 26-voxel neighborhood Sobel filter [16]. However, it is slower and does not affect the final centerline.)

4.5 Flag Non-Uniform Gradient Neighborhoods

Figure 7 depicts a section of a colon with the centerline and the distance from the boundary gradient vectors. We are interested in labeling the voxels on and next to the centerline that often have local neighborhoods of non-uniform GVF vectors. There are six classes of regions in the GVF:

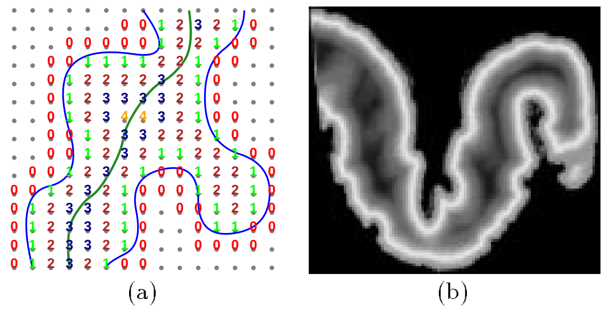


Figure 5: *(a) Explicit DBF values (rounded to integers). (b) DBF visualized through a rainbow color map.*

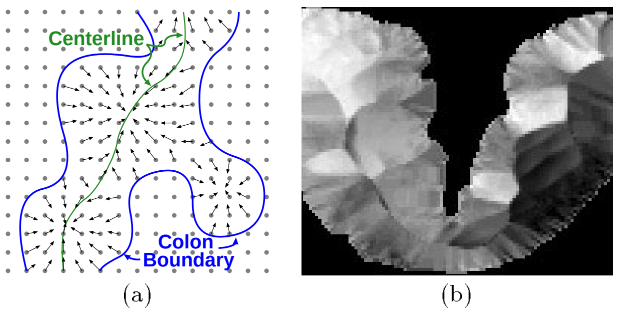


Figure 6: *GVF vectors (a) as arrows, and (b) as XYZ=RGB pixel components along a colon cross section.*

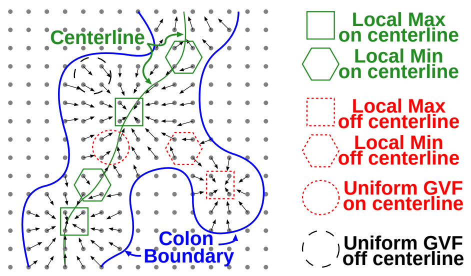


Figure 7: *Colon boundary and centerline in the GVF. Squares denote local maxima, hexagons point out local minima, and circles are in uniform GVF vector areas.*

(1) Local maxima on the centerline (depicted as solid squares in Figure 7). All the GVF vectors surrounding such a maximum, point towards it. This means each of these vectors has a different direction.
(2) Local minima on the centerline (depicted as solid hexagons). All the GVF vectors surrounding such a minimum, point away from it towards the neighboring maxima. In a tubular shape, for each minimum there must be at least two enclosing maxima. Thus, the GVF vectors at a minimum split in at least two groups of vectors pointing in opposite directions.
(3) Uniform areas off the centerline (depicted as black dashed circles). All positions close to the colon boundary have GVF vectors that point in about the same direction as their neighbors, because the steepest increase of distance from boundary is always perpendicular to the boundary, and because the boundary itself has only limited curvature.
(4) Uniform areas on the centerline (depicted as red dotted circles). Unfortunately, there are also areas along the centerline where the colon widens quickly. Hence, all GVF vectors neighboring the centerline point uniformly towards the center of the wide open area. Labeling all non-uniform GVF positions does not capture this area.
(5) Local maxima off the centerline (depicted as red dotted squares). There may exist folds of the colon that enclose local maxima of the DBF that are not on the centerline. The GVF vectors surrounding such a maximum have the same characteristics as the ones surrounding a local maximum on the centerline.
(6) Local minima off the centerline (depicted as red dotted hexagons). Between maxima on and off the centerline, there may also be local minima off the centerline that are indistinguishable from those on the centerline, as long as only local filters are applied.

We define a set of 8 GVF vectors in a 2^3 cell of voxel positions to be uniform, if and only if all 8 vectors differ by less than 90 degrees from their average direction. Thus, to label local non-uniform GVF vectors we:
(a) Compute the normalized average GVF vector for each 2^3 cell of voxel positions.
(b) Test if the dot-product between any of the 2^3 GVF vectors and the associated average vector is zero or negative.
(c) Label all 2^3 voxel cells that have non-positive dot-products as non-uniform.

It is possible that a voxel is exactly at a local DBF maximum and its GVF vector is a zero vector. Note that this leads to a zero-value dot-product which does cause labeling of that position. Thus, no special case handling for degenerate GVF vectors is necessary.

As each test involves only a small 8 voxel position neighborhood, it can be executed very quickly (14 s for an average colon) and it produces flagged voxels that are only 1% of the inside colon voxels (see Figure 8).

4.6 Connect Flagged Voxels

The previous CEASAR algorithm step results in a number of disconnected components of flagged voxels. We connect these components by applying the following procedure for each flagged voxel.
(a) Pick a voxel from the flagged source volume and flag the corresponding voxel in the destination volume.
(b) Starting from that flagged voxel traverse the source volume along the GVF vectors and flag all voxels along the path in the destination volume.
(c) Stop the traversal as soon as another flagged voxel is reached.

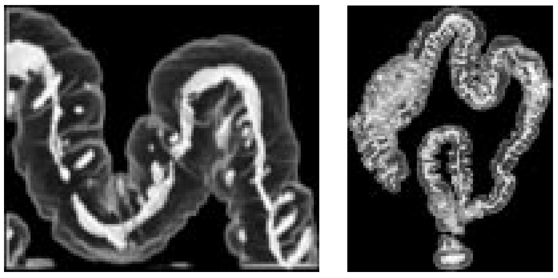

(a) (b)

Figure 8: *(a) Zoomed and (b) complete colon with highlighted voxels at non-uniform vector positions.*

Figure 9 shows an example, in which starting from a minimum, the path taken towards the next maximum results in a sequence of voxels along a path that is centralized in the DBF. In fact, the walk is self-correcting: each GVF vector can be viewed as a combination of the direction along the continuous centerline and the direction towards it. Due to the discrete nature of the path we might reach a voxel to the left of the continuous centerline. At that position the component of the GVF vector towards the centerline points to the right. Thus, the next step brings us back to the right. Any error caused by discretization is corrected in the next step.

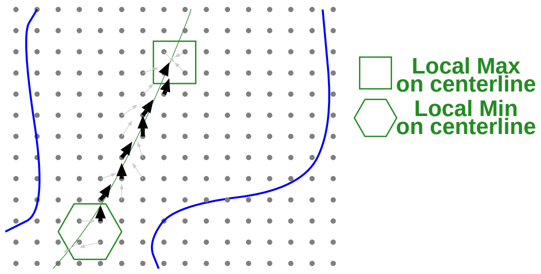


Figure 9: *Connecting flagged voxels. Starting from each flagged voxel, we follow along the GVF vectors until we reach another flagged voxel.*

Starting from flagged voxels near, but off the centerline, the followed path is directed towards the centerline and eventually merges with one of the path starting from a local minimum on the centerline. Consequently, this method also connects the off centerline minima and maxima to the set of flagged voxels on the centerline. At the end, we have labeled a set of voxels that includes all centerline voxels and totals just 15-30% of all colon voxels. All further operations are restricted to this set of voxels.

4.7 DAF: Compute Distance from Any Flagged Voxel Field

The following three CEASAR steps are an adaptation of the Dijkstra shortest path graph algorithm as outlined in Sections 2.2 and 3.2. This first step just computes the **d**istance from **a**ny flagged voxel **f**ield (DAF) with anisotropically correct Euclidian distance as weights in the implicit graph of Figure 4a. We assume that any well segmented colon has the properties listed in Section 3.1 (long and narrow). Then, independently of which colon voxel we select as a starting point, the furthest voxel is provably at one of the two ends of the colon. Figure 11 shows the resulting DAF of a colon.

4.8 PDEF: Compute Penalized Distance from End Voxel Field

Repeating the search for the furthest voxel from the end voxel found in the previous step, we would discover the voxel at the other end of the colon. However, during the second search we extend the mapping of the volume data to include a graph to incorporate penalties for coming close to the colon boundary as illustrated in Section 3.2, and thus create the penalized distance from end voxel field (PDEF).

The penalty p at a voxel v is assigned based on the DBF value at that voxel v and a global upper bound of all DBF values ($M > max(DBF)$). The penalty p is

$$p(v) = 5000 \cdot [1 - \frac{DBF(v)}{M}]^{16}.$$

Note that $\frac{DBF(v)}{M}$ is always in the range of [0,1]. Thus $[1 - \frac{DBF(v)}{M}]^{16}$ is in the same range, but with the maximal values for voxels close to the boundary. The factor 5000 is needed to ensure that the penalty overpowers the advantages of choosing a straight path. Choosing 5000 is a heuristic, that allows skeleton segments to be up to 3000 voxels long without exceeding floating point precision.

For our implementation we did not need to explicitly store all 26 penalty vertices and edges depicted in Figure 4, because the only way to incorporate a center vertex in the path is to go through two of its penalty vertices, and thus along the two penalty edges of equal penalty weight. Therefore, we can actually keep the implicit edges and vertices from the DAF generation method, but add the penalty directly to the computation of the accumulated distance at each voxel v:

$$dist(v_k) = dist(v_{k-1}) + dist(v_k, v_{k-1}) + penalty(v_k).$$

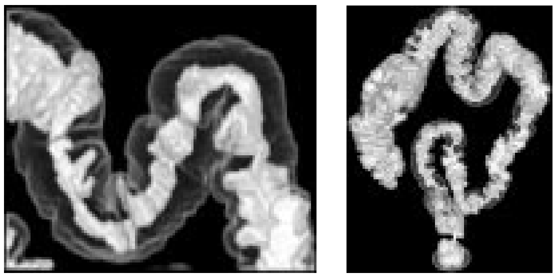

(a) (b)

Figure 10: *(a) Zoomed and (b) complete colon with all flagged voxels highlighted.*

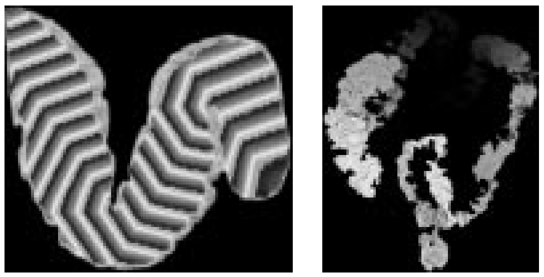

(a) (b)

Figure 11: *(a) Zoomed and (b) complete colon DAF views visualized through color rainbow mappings. (a) The fine structure of the DAF showing "same distace wave fronts" that "leak" around corners. This causes the centers of mass of same distances from source not to lie on the centerline. (b) The overall distance increases along the DAF of the complete colon. The starting point on the top right is colored black and colors become brighter with increasing distance.*

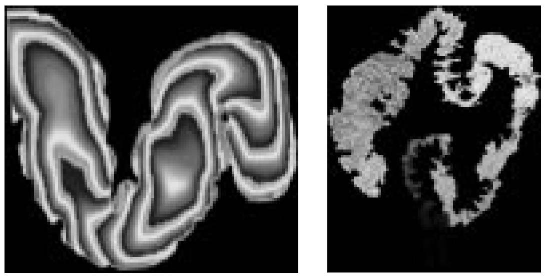

(a) (b)

Figure 12: *(a) Zoomed and (b) complete colon PDEF views visualized through color rainbow mappings. (a) The fine structure of the PDEF showing the "same distance wave fronts" progressing most rapidly in the center of the colon. (b) The overall distance increase along the PDEF of the complete colon.*

4.9 Minimum Cost Path

We choose the voxel with the largest PDEF value as starting voxels for the second phase of the Dijkstra algorithm. Because of our inclusion of strong penalties into the PDEF, this results in a global minimum path between both colon ends that is optimally centered, and follows maximal values of the DBF. This path is the discrete centerline as defined in Section 3.2 and depicted in Figure 13.

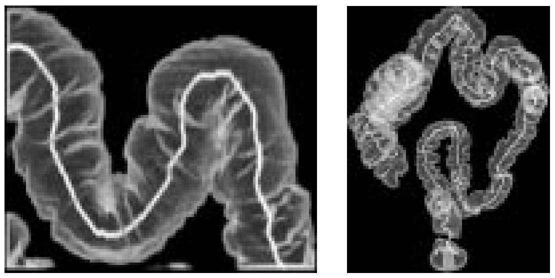

(a) (b)

Figure 13: *(a) Zoomed and (b) complete colon with centerline.*

4.10 Smoothing

For some applications an optimal discrete centerline is not the most desired centralized path. For example, in guided virtual endoscopy, the camera is moved along the centralized path. To maintain a steady view, a smooth curve is favored over a discrete "stair step" path. Given the optimal discrete path and the distance from the colon boundary at each centerline voxel, we can compute an approximating spline [13, 17] with adaptive error tolerance. In very narrow areas the allowed error should be very small, while in wide openings a little larger divergence is acceptable. This can be elegantly expressed as a percentage of the distance from boundary. Any percentage below 50% guarantees that the

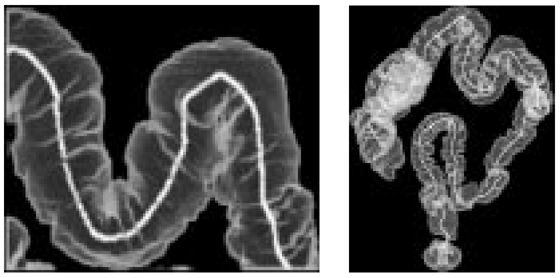

(a) (b)

Figure 14: *(a) Zoomed and (b) complete colon with the centerline after smoothing.*

centerline is always closer to the center than to the boundary. This additional freedom to place the continuous centerline is then used to minimize curvature along the centerline. Specifically, we use a B-spline curve that interpolates the first and last voxel and approximates the ones in between. The control points are placed close to centerline voxels at non-uniform intervals. We apply a number of heuristics (min/max curvature, min/max DBF, maximal control point separation) to minimize the number of control points that is needed to achieve the desired accuracy. In the example of Figure 14 an error tolerance of 35% requires 17% of the discrete centerline voxels as control points while a 50% tolerance needs 13% and results in a smoother centerline.

5 Results

We tested our CEASAR algorithm on three colon datasets and one aorta dataset. Tables 1 and 2 list the details about the dataset sizes and how many CEASAR steps changed the number of processed voxels. In all cases the discrete centerlines were placed right in the center according to visual inspection by virtual colonoscopy and according to mathematical measures such as the DBF.

Table 3 lists the platform and timings of all the CEASAR algorithm steps for each test dataset. The complete algorithm time was always below 5 minutes for the colon datasets and just 26 seconds for the aorta. Figure 13 depicts the final centerlines computed with their associated volumes. It includes a frame from our virtual navigation through a colon that compares the centerline found by "onion peeling" with the more smooth and more central centerline found by CEASAR. Finally, it also demonstrates the results of applying CEASAR to more complicated data such as an aorta and a lobster.

Table 1: *Colon dataset sizes and the reduction of voxels that have to be processed during the execution of the* CEASAR *algorithm.*

dataset	study4	study5	study8
original size X	514	514	514
original size Y	514	514	514
original size Z	363	360	372
auto cropped size X	416	409	406
auto cropped size Y	398	339	379
auto cropped size Z	363	363	375
colon CT voxels	96M	95M	98M
cropped colon voxels	59M	50M	58M
inside colon voxels	3.2M	1.5M	2.4M
flagged as % of colon	1%	2%	1%
connected as % of colon	27%	18%	15%
centerline voxels	1644	1814	1967

Table 2: *Reduction of voxels that have to be processed during the execution of the* CEASAR *applied to an aorta dataset.*

dataset ($256 \times 256 \times 211$)	aorta
aorta CT voxels	13M
cropped aorta voxels	6.7M
inside aorta voxels	230K
flagged as % of aorta	1.8%
connected as % of aorta	14%
centerline voxels	351

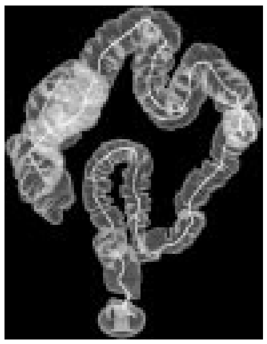
Table 3: *Time spent in each of the* CEASAR *algorithm steps. (All tests were done on an SGI Challenge with 4GB memory running IRIX 6.5 using a single MIPS R10000 CPU running at 194 MHz.)*

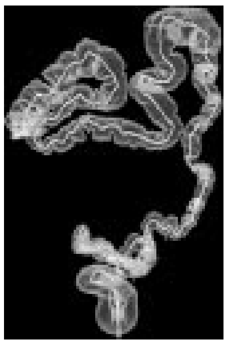
dataset	study4	study5	study8	aorta
cropping	16s	13s	15s	2s
DBF	106s	86s	97s	11s
GVF	44s	41s	43s	5s
flagging	18s	10s	14s	1s
connecting	17s	13s	15s	2s
DAF	9s	6s	7s	1s
PDEF	74s	19s	28s	2s
centerline	3s	3s	3s	1s
B-spline	11s	87s	27s	1s
total	299s	278s	250s	26s

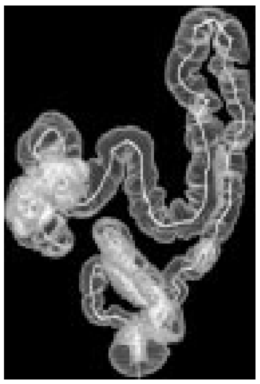
6 Conclusions and Future Work

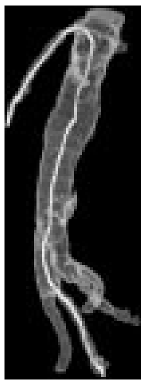
Based on an analysis of prior centerline definitions, we introduced new, mathematically sound definitions of a colon and a centerline. We then showed that CEASAR — our centerline extraction algorithm that delivers smooth, accurate and robust results — always fully automatically finds the two ends of the colon and always computes a provably connected centerline that is optimal with respect to length as well as centrality. We explained in detail our CEASAR implementation and reported results that not only empirically verifyed the correctness of the centerline, but also showed the superior speed of the CEASAR algorithm, that is, less than 5 minutes for all our colon dataset studies. Finally, we also demonstrated that CEASAR can be applied to a variety of colon shapes as well as to other tubular structures such as an aorta.

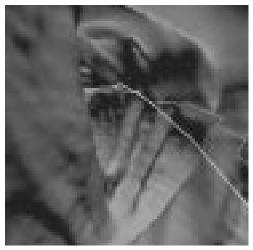
We plan to reduce the running times of the CEASAR algorithm further by employing more cache coherent data layouts and data traversals as well as by parallelizing it for efficient use on multiple CPU computers. We also will extend the algorithm to enable handling of tree structures such as the lungs.


Colon study 4


Colon study 5


Colon study 8


Aorta


Onion Peeling (green) vs. CEASAR (white)

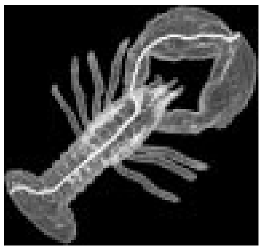

Lobster

Figure 15: *Six datasets with centerlines computed by the* CEASAR *algorithm. (See color plates)*

7 Acknowledgments

This work has been supported by grants from NIH CA79180, ONR N000149710402, E-Z-EM Inc., Hughes Research Laboratory, and Viatronix Inc. The patients' data sets were provided by the University Hospital of the State University of New York at Stony Brook.

References

[1] D. Chen, B. Li, Z. Liang, M. Wan, A. Kaufman, and M. Wax. A tree-branch searching, multiresolution approach to skeletonization for virtual endoskopy. In *SPIE's International Symposium on Medical Imaging 2000*, San Diego, CA, Feb 2000.

[2] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerishe Mathemetik*, 1:269–271, 1959.

[3] Y. Ge, D. R. Stelts, and D. J. Vining. 3D Skeleton for Virtual Colonoscopy. *Lecture notes in Computer Science 0302-9743*, pages 449–454, 1996.

[4] Y. Ge, D. R. Stelts, J. Wang, and D. J. Vining. Computing the Centerline of a Colon: A Robust and Efficient Method Based on 3D Skeletons. *Journal of Computer Assisted Tomography*, 23(5):786–794, 1999.

[5] T. He and A. Kaufman. Collision detection for volumetric models. *IEEE Visualization 97*, pages 27–34, Oct. 1997.

[6] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual Voyage: Interactive Navigation in the Human Colon. *Proc. SIGGRAPH '97*, pages 27–34, 1997.

[7] F. A. Jolesz, W. E. Lorensen, H. Shinmoto, H. Atsumi, S. Nakajima, P. Kavanaugh, Saiviroonpron, S. E. Seltzer, S. G. Silverman, M. Philips, and R. Kikinis. Interactive Virtual Endoscopy. *AJR*, 169:1229–1235, Nov 1997.

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

[9] S. Lakare, M. Wan, M. Sato, and A. Kaufman. 3D Digital Cleansing using Segmentation Rays. Visualization 2000, 2000.

[10] E. G. McFarland, G. Wang, J. A. Brink, D. M. Balfe, J. P. Heiken, and M. W. Vannier. Spiral Computed Tomographic Colonography: Determination of the Central Axis and Digital Unraveling of the Colon. *Acad Radiol*, (4):367–373, 1997.

[11] D. S. Paik, C. F. Beaulieu, R. B. Jeffery, G. D. Rubin, and S. Napel. Automated Flight path Planning for Virtual Endoscopy. *Medical Physics*, 25(5):629–637, 1998.

[12] T. Pavlidis. A thinning algorithm for discrete binary images. *Computer Graphics and Image Processing*, 13:142–157, 1980.

[13] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag, Berlin, second edition, 1997.

[14] T. Saito and J. ichiro Toriwaki. New Algorithms for Euclidian Distance Transformation of an n-Dimensional Digitized Picture with Applications. *Patern Recognition*, 27(11):1551–1565, 1994.

[15] Y. Samara, M. Fiebrich, A. Dachman, J. Kuniyoshi, K. Doi, and K. R. Hoffmann. Automated Calculation of the Centerline of the Human Colon on CT Images. *Acad Radiol*, 6:352–359, 1999.

[16] I. Sobel. An Isotropic 3x3x3 Volume Gradient Operator. Hewlett-Packard's Voxelator V-3 CD-ROM, Aug. 1996. Online Information at http://www.hp.com/info/voxelator (Paper is only on CD, not online).

[17] F. Yamaguchi. *Curves and Surfaces in Computer Aided Geometric Design*. Springer-Verlag, Berlin, 1988.

[18] Y. Zhou and A. W. Toga. Efficient Skeletonization of Volumetric Objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1999.

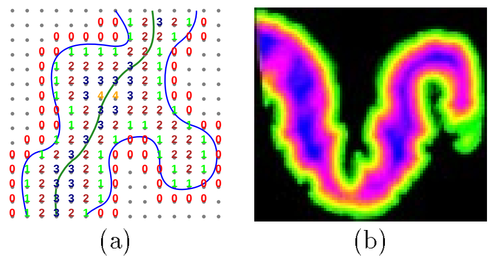


Figure 5: *(a) Explicit DBF values (rounded to integers). (b) DBF visualized through a rainbow color map.*

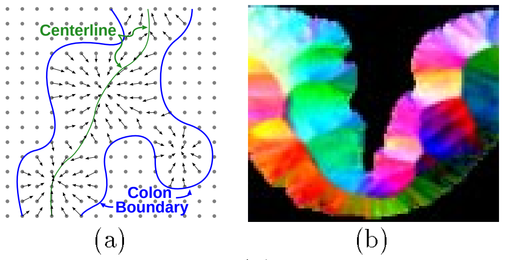


Figure 6: *GVF vectors (a) as arrows, and (b) as XYZ=RGB pixel components along a colon cross section.*

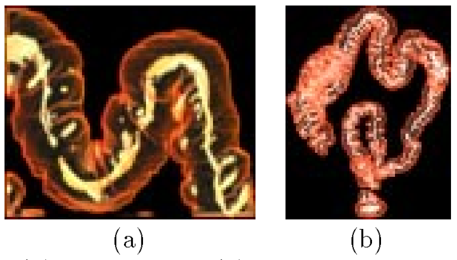


Figure 8: *(a) Zoomed and (b) complete colon with highlighted voxels at non-uniform vector positions.*

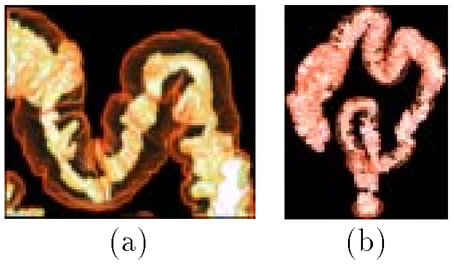


Figure 10: *(a) Zoomed and (b) complete colon with all flagged voxels highlighted.*

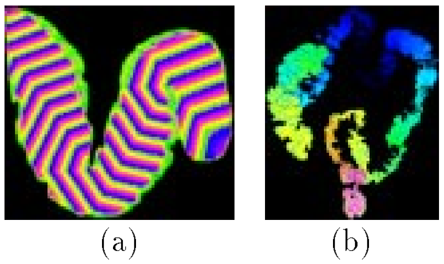


Figure 11: *(a) Zoomed and (b) complete colon DAF views visualized through color rainbow mappings.*

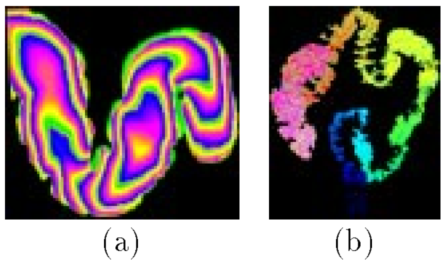


Figure 12: *(a) Zoomed and (b) complete colon PDEF views visualized through color rainbow mappings.*

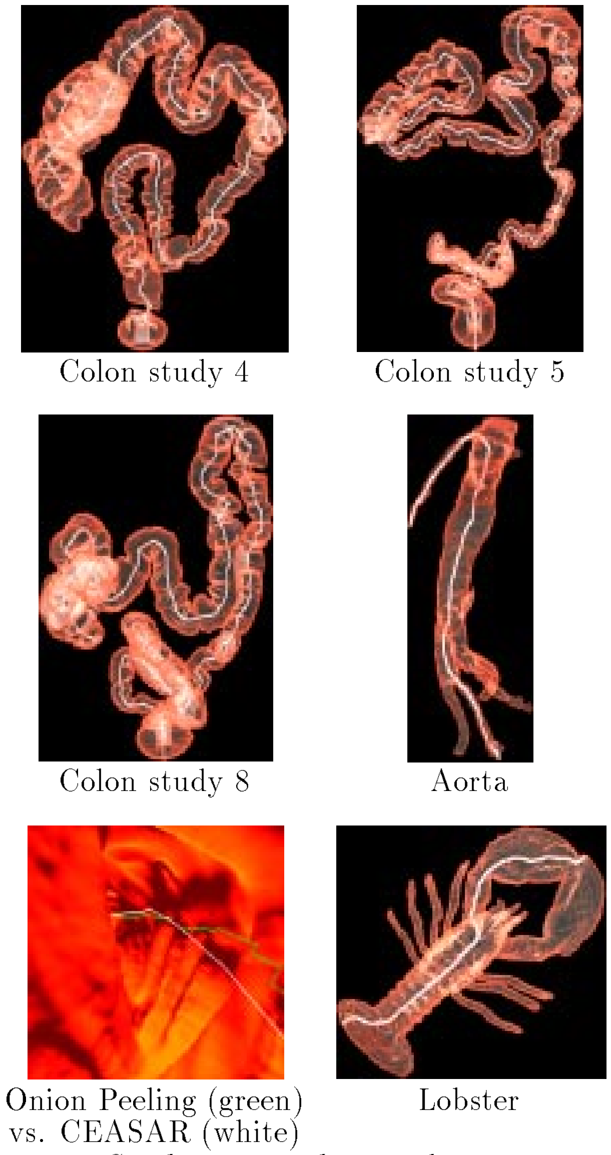


Figure 15: *Six datasets with centerlines computed by the* CEASAR *algorithm.*