Grouping connected components using minimal path techniques. Application to reconstruction of vessels in 2D and 3D images.

Laurent D. COHEN[1]
[1] CEREMADE, UMR CNRS 7534
Université Paris 9-Dauphine
75775 Paris cedex 16, France
cohen@ceremade.dauphine.fr

Thomas Deschamps[1,2]
[2] Medical Imaging Systems Group
Philips Research France
51 rue Carnot, B.P. 301, 92156 Suresnes, France
thomas.deschamps@philips.com

Abstract

We address the problem of finding a set of contour curves in a 2D or 3D image. We consider the problem of perceptual grouping and contour completion, where the data is an unstructured set of regions in the image. A new method to find complete curves from a set of edge points is presented. Contours are found as minimal paths between connected components, using the fast marching algorithm. We find the minimal paths between each of these components, until the complete set of these "regions" is connected. The paths are obtained using backpropagation from the saddle points to both components.

We then extend this technique to 3D. The data is a set of connected components in a 3D image. We find 3D minimal paths that link together these components. Using a potential based on vessel detection, we illustrate the capability of our approach to reconstruct tree structures in a 3D medical image dataset.

1 Introduction

Since their introduction, active contours [9] have been extensively used to find the contour of an object in an image through the minimization of an energy. In order to get a set of contours with T-junctions, we need many active contours to be initialized on the image. The level sets paradigm [13, 1] allows changes in topology. It enables to get multiple contours by starting with a single one. However, it does not give satisfying results when there are gaps in the data since the contour may propagate into a hole and then split into several curves where only one contour is desired. This is the problem encountered with perceptual grouping when we try to group a set of incomplete contours. For example, in a binary image like in figure 1 with a drawing of a shape with holes, human vision can easily fill in the missing boundaries and form complete curves. Perceptual grouping is an old problem in computer vision. It has been approached more recently with energy methods [16, 8, 18]. These methods find a criteria for saliency of a curve component or for each point of the image. This saliency measure is based indirectly on a second order regularization snake-like energy ([9]) of a path containing the point. However, the final curves are generally obtained in a second step as ridge lines of the saliency criteria after thresholding. Motivated by this relationship between energy minimizing curves like snakes and completion contours, we worked upon finding a set of completion contours on an image as a set of energy minimizing curves.

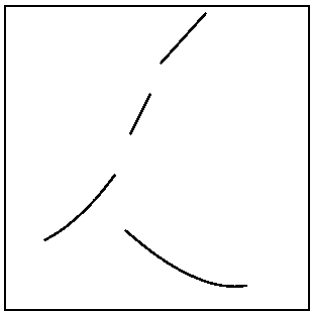

Figure 1: Examples of connected regions to be completed

In order to solve global minimization for snakes, the authors of [4] used the minimal paths, as introduced in [11, 10]. The goal was to avoid local minima without demanding too much on user initialization, which is a main drawback of classic snakes [3]. Only two end points were needed. The numerical method has the advantage of being consistent (see [4]) and efficient using the Fast Marching algorithm introduced in [15]. In [2], we proposed a way to use this minimal path approach to find a set of curves drawn from a set of points in the image. We also introduced a technique that automatically finds a set of *key* end points. In this paper, we extend the previous approach to connected components instead of end points. In order to obtain a set of most salient contour curves, we find a set of minimal paths between pairs of connected com-

ponents.

This approach is then extended for application in the completion of tube-like structures in 2D and 3D images. The problem is here to complete a partially detected object, based on some detected connected components that belong to this object.

For perceptual grouping, the potential P to be minimized along the curves is usually an image of edge points that represent simple incomplete shapes, as in figure 1. These edge points are represented as a binary image with small potential values along the edges and high values at the background. The potential could also be defined as edges weighted by the value of the gradient or as a function of an estimate of the gradient of the image itself, $P = g(\|\nabla I\|)$, like in classic snakes. The potential could also be a grey level image as in [4]. It could also be a more complicated function of the grey level. In our real examples of vascular structures in 2D and 3D, we use a potential based on a vesselness filter [7].

The paper is presented as follows. We first give a summary of minimal paths and fast marching in 2D and 3D images in section 2. We then present in Section 3 how to find a set of curves from a given set of unstructured points. Grouping the points in connected components, we propose a way to find the pairs of linked connected components and the paths between them. We then extend this approach to 3D and show an application in 3D medical images.

2 Minimal Paths in 2D and 3D

2.1 Global Minimum for Active Contours

We present in this section the basic ideas of the method introduced in [4] (see also [2]) to find the global minimum of the active contour energy using minimal paths. The energy to minimize is similar to classical deformable models (see [9]) where it combines smoothing terms and image features attraction term (Potential P):

$$E(C)=\int_\Omega \Big\{ w_1 \|C'(s)\|^2 + w_2 \|C''(s)\|^2 + P(C(s)) \Big\} ds \quad (1)$$

where $C(s)$ represents a curve drawn on a 2D image and Ω is its domain of definition. As explained in [4], we are lead to minimize

$$E(C) = \int_{\Omega=[0,L]} \{w + P(C(s))\} ds, \quad (2)$$

where s is the arclength parameter ($\|C'(s)\| = 1$). The regularization of this model is now achieved by the constant $w > 0$ (see [4] for details). Given a potential $P \geq 0$, the energy is like a distance weighted

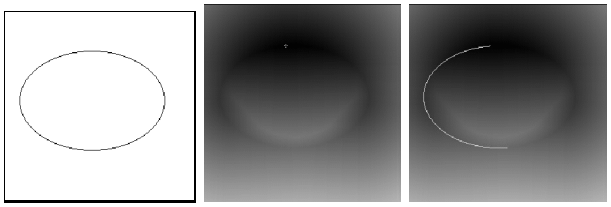

Figure 2: Finding a minimal path between two points. Left: potential; middle, minimal action to the marked point; right, minimal path from the second point.

by $\tilde{P} = P + w$. The minimal action $\mathcal{U}$ is defined as the minimal energy integrated along a path between a starting point p_0 and any point p:

$$\mathcal{U}(p) = \inf_{\mathcal{A}_{p_0,p}} E(C) = \inf_{\mathcal{A}_{p_0,p}} \left\{ \int_\Omega \tilde{P}(C(s)) ds \right\} \quad (3)$$

where $\mathcal{A}_{p_0,p}$ is the set of all paths between p_0 and p. The minimal path between p_0 and any point p_1 in the image can be easily deduced from this action map by a simple back-propagation (gradient descent on $\mathcal{U}$) starting from p_1 until p_0 is reached.

2.2 Fast Marching Resolution

In order to compute this map $\mathcal{U}$, a front-propagation equation related to Equation (3) is solved:

$$\frac{\partial C}{\partial t} = \frac{1}{\tilde{P}} \vec{n}. \quad (4)$$

It evolves a front starting from an infinitesimal circle shape around p_0 until each point inside the image domain is assigned a value for $\mathcal{U}$. The value of $\mathcal{U}(p)$ is the time t at which the front passes over the point p.

The *Fast Marching* technique, introduced in [15], was used in [4] noticing that the map $\mathcal{U}$ satisfies the Eikonal equation $\|\nabla \mathcal{U}\| = \tilde{P}$ and $\mathcal{U}(p_0) = 0$. Classic finite differences schemes for this equation tend to overshoot and are unstable. Therefore, an up-wind scheme was proposed by [15]:

$$\begin{aligned} (\max\{u - \mathcal{U}_{i-1,j}, u - \mathcal{U}_{i+1,j}, 0\})^2 \quad &+ \\ (\max\{u - \mathcal{U}_{i,j-1}, u - \mathcal{U}_{i,j+1}, 0\})^2 \quad &= \quad \tilde{P}_{i,j}^2. \end{aligned} \quad (5)$$

The improvement made by the *Fast Marching* is to introduce order in the selection of the grid points. This order is based on the fact that information is propagating *outward*, because the action can only grow due to the quadratic Equation (5). The algorithm is detailed in Table 1. An example is shown in Figure 2.

2.3 Algorithm for 2D Up-Wind Scheme

Notice that for solving Eqn. (5), only alive points are considered. Considering the neighbors of grid

Algorithm for 2D Fast Marching

- Definitions:
  - *Alive* set: grid points at which the action value $\mathcal{U}$ has been reached and will not be changed;
  - *Trial* set: next grid points (4-connexity neighbors) to be examined. An estimate U of $\mathcal{U}$ has been computed using Eqn. (5) from alive points only (i.e. from $\mathcal{U}$);
  - *Far* set: all other grid points, there is not yet an estimate for U;
- Initialization:
  - *Alive* set: start point $p_0, U(p_0) = \mathcal{U}(p_0) = 0$;
  - *Trial* set: reduced to the four neighbors p of p_0 with initial value $U(p) = \tilde{P}(p)$ ($\mathcal{U}(p) = \infty$);
  - *Far* set: all other grid points, $\mathcal{U} = U = \infty$;
- Loop:
  - Let $p = (i_{min}, j_{min})$ be the *Trial* point with the smallest action U;
  - Move it from the *Trial* to the *Alive* set;
  - For each neighbor (i, j) of (i_{min}, j_{min}):
    - If (i, j) is *Far*, add it to the *Trial* set;
    - If (i, j) is *Trial*, update $U_{i,j}$ with Eqn. (5).

Table 1: *Fast Marching* algorithm

point (i, j) in 4-connexity, we note $\{A_1, A_2\}$ and $\{B_1, B_2\}$ the two couples of opposite neighbors such that we get the ordering $\mathcal{U}(A_1) \leq \mathcal{U}(A_2)$, $\mathcal{U}(B_1) \leq \mathcal{U}(B_2)$, and $\mathcal{U}(A_1) \leq \mathcal{U}(B_1)$. Considering that we have $u \geq \mathcal{U}(B_1) \geq \mathcal{U}(A_1)$, the equation derived is

$$(u - \mathcal{U}(A_1))^2 + (u - \mathcal{U}(B_1))^2 = \tilde{P}_{i,j}^2 \qquad (6)$$

Based on testing the discriminant Δ of Eqn. (6), one or two neighbors are used to solve it:

1. If $\tilde{P}_{i,j} > \mathcal{U}(B_1) - \mathcal{U}(A_1)$, solution of Eqn. (6) is $u = \frac{\mathcal{U}(B_1) + \mathcal{U}(A_1) + \sqrt{2\tilde{P}_{i,j}^2 - (\mathcal{U}(B_1) - \mathcal{U}(A_1))^2}}{2}$.
2. else $u = \mathcal{U}(A_1) + \tilde{P}_{i,j}$.

2.4 Minimal Paths in 3D

A 3D extension of the *Fast Marching* algorithm was presented in [5] and detailed in [5]. Similarly to previous section, the minimal action $\mathcal{U}$ is defined as

$$\mathcal{U}(p) = \inf_{\mathcal{A}_{p_0,p}} \left\{ \int_\Omega \tilde{P}(C(s))ds \right\} \qquad (7)$$

where $\mathcal{A}_{p_0,p}$ is now the set of all 3D paths between p_0 and p. The 2D scheme equation (5) is extended to 3D, leading to the scheme

$$\begin{aligned} (\max\{u - \mathcal{U}_{i-1,j,k}, u - \mathcal{U}_{i+1,j,k}, 0\})^2 &+ \\ (\max\{u - \mathcal{U}_{i,j-1,k}, u - \mathcal{U}_{i,j+1,k}, 0\})^2 &+ \\ (\max\{u - \mathcal{U}_{i,j,k-1}, u - \mathcal{U}_{i,j,k+1}, 0\})^2 &= \tilde{P}_{i,j,k}^2 \end{aligned} \qquad (8)$$

giving the correct viscosity-solution u for $\mathcal{U}_{i,j,k}$.

3 Finding Contours from a Set of Connected Components R_k

3.1 Minimal Path between two Regions

The method of [4], detailed in the previous section allows to find a minimal path between two endpoints. This is a straightforward extension to define a minimal path between two regions of the image. Given two connected regions of the image R_0 and R_1, we consider R_0 as the starting region and R_1 as a set of end points. The problem is then finding a path minimizing energy among all paths with start point in R_0 and end point in R_1. The minimal action is now defined by

$$\mathcal{U}(p) = \inf_{\mathcal{A}_{R_0,p}} E(C) = \inf_{p_0 \in R_0} \inf_{\mathcal{A}_{p_0,p}} E(C) \qquad (9)$$

where $\mathcal{A}_{R_0,p}$ is the set of all paths starting at a point of R_0 and ending at p. This minimal action can be computed the same way as before in table 1, with the alive set initialized as the whole set of points of R_0, with $\mathcal{U} = 0$ and trial points being the set of 4-connexity neighbors of points of R_0 that are not in R_0. Backpropagation by gradient descent on $\mathcal{U}$ from any point p in the image will give the minimal path that join this point with region R_0.

In order to find a minimal path between region R_1 and region R_0, we determine a point $p_1 \in R_1$ such that $\mathcal{U}(p_1) = \min_{p \in R_1} \mathcal{U}(p)$. We then backpropagate from p_1 to R_0 to find the minimal path between p_1 and R_0, which is also a minimal path between R_1 and R_0.

3.2 Minimal Paths from a Set of Connected Components

We are now interested in finding many or all contours in an image. We assume that from some preprocessing, or as data, we have an initial set of contours. We denote R_k the connected components of these contours. We propose to find the contours as a set of minimal paths that link pairs of regions among the R_k's. If we also know which pairs of regions have to be linked together, finding the whole set of contours is a trivial application of the previous section. The problem we are interested in here is also to find out which pairs of regions have to be connected by a contour. Since the set of contours R_k's is assumed to be given unstructured, we do not know in advance how the regions connect. This is the key problem that is solved here using a minimal action map.

3.3 Method

Our approach is similar to computing the distance map to a set of regions and their Voronoi diagram.

However, we use here a weighted distance defined through the potential P. This distance is obtained as the minimal action with respect to P with zero value at all points of regions R_k. Instead of computing a minimal action map for each pair of regions, as in Section 3.1, we only need to compute one minimal action map in order to find all paths. At the same time the action map is computed we determine the pairs of regions that have to be linked together. This is based on finding meeting points of the propagation fronts. These are *saddle points* of the minimal action $\mathcal{U}$. In Section 2, we said that calculation of the minimal action can be seen as the propagation of a front through equation (4). Although the minimal action is computed using fast marching, the level sets of $\mathcal{U}$ give the evolution of the front. During the fast marching algorithm, the boundary of the set of alive points also gives the position of the front. In the previous section, we had only one front evolving from the starting region R_0. Since all points p of regions R_k are set with $\mathcal{U}(p) = 0$, we now have one front evolving from each of the starting regions R_k. In what follows when we talk about front meeting, we mean either the geometric point where the two fronts coming from different R_k's meet, or in the discrete algorithm the first alive point which connects two components from different R_k's (see Figures 3 and 4).
We use the fact that given two regions R_1 and R_2, the saddle point S where the two fronts starting from each region meet can be used to find the minimal path between R_1 and R_2. Indeed, the minimal path between the two regions has to pass by the meeting point S. This point is the point half way (in energy) on a minimal path between R_1 and R_2. Backpropagating from S to R_1 and then from S to R_2 gives the two halves of the path.

3.4 Notations and definitions

Here are some definitions that will be used in what follows. X being a set of points in the image, $\mathcal{U}_X$ is the minimal action obtained by Fast Marching with potential $\tilde{P}$ and starting points $\{p, p \in X\}$. This means that all points of X are initialized as alive points with value 0. All their 4-connexity neighbors that are not in X are *trial* points. This is easy to see that $\mathcal{U}_X = \min_{p \in X} \mathcal{U}_p$. X may be a connected component R or a set of connected components.

The *label* l at a point p is equal to the index k of the region R_k for p closer in energy to R_k than to other regions R_j. This means that minimal action $\mathcal{U}_{R_k}(p) \leq \mathcal{U}_{R_j}(p), \forall j \neq k$. We define the region $L_k = \{p / l(p) = k\}$. If $X = \cup_j R_j$, we have $\mathcal{U}_X = \mathcal{U}_{R_k}$ on L_k and the computation of $\mathcal{U}_X$ is the same as the simultaneous computation of each $\mathcal{U}_{R_k}$ on each region L_k. These are the simultaneous fronts starting from each R_k.

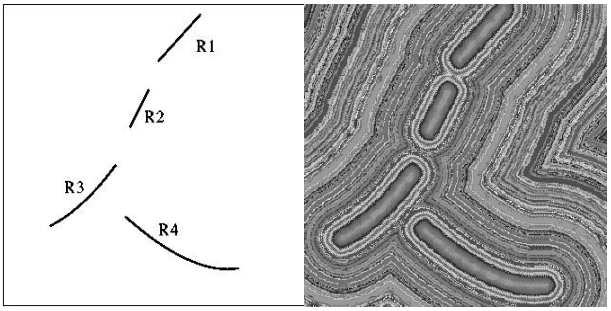

Figure 3: Minimal Action map from the four regions of the example of figure 1. On the right with a random LUT to show the level sets.

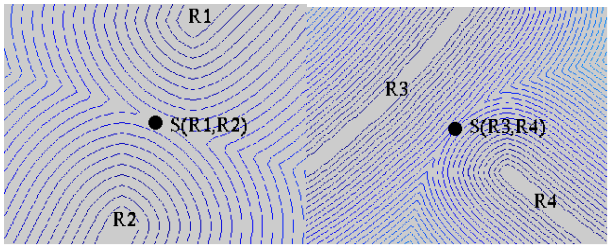

Figure 4: Zoom on *saddle points* between regions.

A *saddle point* $S(R_i, R_j)$ between R_i and R_j is the first point where the front starting from R_i to compute $\mathcal{U}_{R_i}$ meets the front starting from R_j to compute $\mathcal{U}_{R_j}$; At this point, $\mathcal{U}_{R_i}$ and $\mathcal{U}_{R_j}$ are equal and this is the smallest value for which they are equal.

Two different regions among the R_k's will be called *linked regions* if they are selected to be linked together. The way we choose to link two regions is to select some *saddle points*. Thus regions R_i and R_j are *linked regions* if their *saddle point* is among the selected ones.

A *cycle* is a sequence of different regions $R_k, 1 \leq k \leq K$, such that for $1 \leq k \leq K-1$, R_k and R_{k+1} are *linked regions* and R_K and R_1 are also *linked regions*.

3.5 Finding and Selecting Saddle Points

The main goal of our method is to obtain all significant paths joining the given regions. However, each region should not be connected to all other regions, but only to those that are closer to them in the energy sense. There are many possibilities for deciding which regions connect together depending on the kind of data and application. In some cases, the goal would be to detect closed curves and avoid forming branches, as in [2]. Then the criterion would be to constrain a

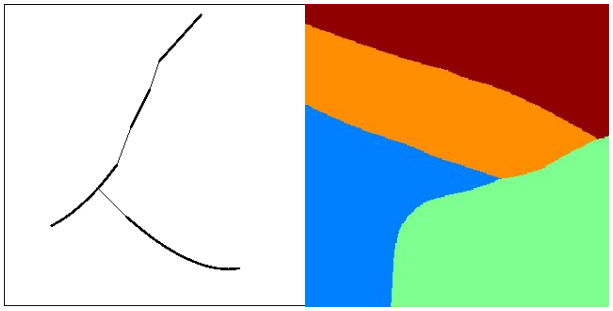

Figure 5: Example with four regions. On the left we show the minimal paths obtained by backpropagation from the three *saddle points* to each of the regions from where the front comes; on the right, and the Voronoi diagram obtained.

region to be linked to at most two other regions in order to make *cycles*. In our context, we are interested in detecting branches and avoiding closed curves. Therefore the criterion for two regions R_i and R_j to be connected is that their fronts meet without creating a "cycle".

We see in Figure 4 a zoom on the saddle points detected between regions R_1 and R_2 and R_3 and R_4. Once a *saddle point* $S(R_i, R_j)$ is found and selected, backpropagation relatively to final energy $\mathcal{U}$ should be done both ways to R_i and to R_j to find the two halves of the path between them. We see in Figure 5 this backpropagation at each of the three automatically selected *saddle points*. They link R_1 to R_2, R_2 to R_3 and R_3 to R_4. At a saddle point, the gradient is zero, but the direction of descent towards each point are opposite. For each backpropagation, the direction of descent is the one relative to each region. This means that in order to estimate the gradient direction toward R_i, all points in a region different from L_i have their energy put artificially to ∞. This allows finding the good direction for the gradient descent towards R_i. However, as mentioned earlier, these backpropagations have to be done only for selected *saddle points*. In the fast marching algorithm we have a simple way to find *saddle points* and update the *linked regions*.

As defined above, the region L_k associated with a region R_k is the set of points p of the image such that minimal energy $\mathcal{U}_{R_k}(p)$ to R_k is smaller than all the $\mathcal{U}_{R_j}(p)$ to other regions R_j. The set of such regions L_k covers the whole image, and forms the Voronoi diagram of the image (see figure 5). All *saddle points* are at a boundary between two regions L_k. For a point p on the boundary between L_j and L_k, we have $\mathcal{U}_{R_k}(p) = \mathcal{U}_{R_j}(p)$. The *saddle point* $S(R_k, R_j)$ is a point on this boundary with minimal value of $\mathcal{U}_{R_k}(p) = \mathcal{U}_{R_j}(p)$. This gives us a rule to find the *saddle points* during the fast marching algorithm.

Each time two fronts coming from R_k and R_j meet for the first time, we define the meeting point as $S(R_k, R_j)$. This means that we need to know for each point of the image from where it comes. This is easy to keep track of its origin by generating an index map updated at each time a point is set as alive in the algorithm. Each point of region R_k starts with label k. Each time a point is set as alive, it gets the same label as the points it was computed from in formula (5). In that formula, the computation of $U_{i,j}$ depends only on at most two of the four pixels involved. These two pixels, said A_1 and B_1, have to be with the same *label*, except if (i, j) is on the boundary between two labels. If A_1 and B_1 are both alive and with different labels k and l, this means that regions R_k and R_l meet there. If this happens for the first time, the current point is set as the *saddle point* $S(R_k, R_l)$ between these regions. A point on the boundary between R_k and R_l is given the label of the neighbor point with smaller action A_1. At the boundary between two labels there can be a slight error on labeling. This error of at most one pixel is not important in our context and could be refined if necessary.

3.6 Algorithm

The algorithm for this section is described in Table 2 and illustrated in figures 3 to 5. When there is a large number of R_k's, this does not change much the computation time of the minimal action map, but this makes more complex dealing with the list of linked regions and *saddle points* and testing for *cycles*.

The way we chose to test for cycles is as follows. Assume a saddle point between regions R_i and R_j is found. We then test if there is already a link between these regions through other regions. This means we are looking for a sequence of different regions $R_k, 1 \leq k \leq K$, with $R_1 = R_i$ and $R_K = R_j$, such that for $1 \leq k \leq K-1$, R_k and R_{k+1} are *linked regions*.

This kind of condition can be easily implemented using a recursive algorithm. When two regions R_i and R_j are willing to be connected - ie that their fronts meet - a table storing the connectivity between each region enables to detect if a link already exists between those regions. Having N different regions, we fill a matrix $M(N, N)$ with zeros, and each time two regions R_i and R_j meet without creating a cycle, we set $M(i, j) = M(j, i) = 1$. Thus, when two regions meet, we apply the algorithm detailed in table 3.

If two regions are already linked, the pixel where their fronts meet is not considered as a valuable can-

Minimal paths between Regions R_k

- Initialization:
  - R_k's are given
  - $\forall k, \forall p \in R_k, V(p) = 0; l(p) = k$; p *alive*.
  - $\forall p \notin \cup_k R_k, V(p) = \infty; l(p) = -1$; p is *far* except 4-connexity neighbors of R_k's that are *trial* with estimate U using Eqn. 5.
- Loop for computing $V = \mathcal{U}_{\cup_k R_k}$:
  - Let $p = (i_{min}, j_{min})$ be the *Trial* point with the smallest action U;
  - Move it from the *Trial* to the *Alive* set with V(p) = U(p);
  - Update $l(p)$ with the same index as point A_1 in formula (5). If $R(A_1) \neq R(B_1)$ and we are in case 1 of section 2.3 where both points are used and if this is the first time regions of labels $l(A_1)$ and $l(B_1)$ meet, $S(R_{l(A_1)}, R_{l(B_1)}) = p$ is set as a *saddle point* between $R_{l(A_1)}$ and $R_{l(B_1)}$. If adding a link between these regions does not create a *cycle*, they are set as *linked regions* and $S(R_{l(A_1)}, R_{l(B_1)}) = p$ is selected,
    For each neighbor (i, j) of (i_{min}, j_{min}):
    - If (i, j) is *Far*, add it to the *Trial* set;
    - If (i, j) is *Trial*, update action $U_{i,j}$.
- Obtain all paths between selected *linked regions* by backpropagation each way from their *saddle point* (see Section 3.5).

Table 2: Algorithm of Section 3

didate for back-propagation. The algorithm stops automatically when all regions are connected.

3.7 Application

The method can be applied to connected components from a whole set of edge points or points obtained through a preprocessing. Finding all paths from a given set of points is interesting in the case of a binary potential defined, like in Figure 3, for perceptual grouping. It can be used as well when a special preprocessing is possible, either on the image itself to extract characteristic points or on the geometry of the initial set of points to choose more relevant points. We show in Figure 6 an example of application to a medical image of the hip where the objects of interest are the vessels. Potential P is defined using ideas from [7] on vesselness filter (detailed later in section 4.2). About vessel detection, see also [17, 12].

4 Finding a Set of Paths in a 3D Image

4.1 Extension to 3D

We now extend our approach to finding a set of 3D minimal paths between regions in 3D images. All definitions and algorithms of section 3 are not affected

Algorithm for Cycle detection when a region R_i meets a region R_j: *Test*(i, j, M, i); *with* *Test*(i, j, M, l);

- if $M(l, j) = 1$, return 1;
- else
  - count=0;
  - for $k \in [1, N]$ with $k \neq i, k \neq j, k \neq l$: count $+=$ *Test*(k, j, M, l);
  - return count;

Table 3: Cycle detection

by changing the dimension of the image from 2D to 3D. The main changes are that 4-connexity in 2D is now 6-connexity in 3D and that we deal with minimal paths and minimal action in 3D images. We presented briefly in section 2.4 the fast marching extension to 3D and for more details on minimal paths in 3D images, we refer to [5].

4.2 Application to Real Datasets: a MR Image of the Aorta

The problem here is to complete a partially detected object. In figure 7 is shown a 3D MR dataset of the aorta, which presents a typical pathology: an abdominal aortic aneurysm. The anatomical object is made visible on the image by injecting a contrast product before the image acquisition.

We propose here to give a method for extracting from the grey level image a set of paths that will represent an approximate skeleton of the tree structure. This is based on extracting first a set of unstructured voxels or regions that belong to the object. Notice that [17, 12] give different methods to detect vessels but ours is much simpler and faster.

For this, we propose to extract valuable information from this dataset, computing a multiscale vessel enhancement measure, based on the work of [7] on ridge filters. Having extracted the three eigenvalues of the Hessian matrix computed at scale σ, ordered $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$, we define a vesselness function

$$\nu(s) = \begin{cases} 0, \text{ if } \lambda_2 \geq 0 \text{ or } \lambda_3 \geq 0 \\ (1 - \exp \frac{-{R_A}^2}{2\alpha^2}) \exp \frac{-{R_B}^2}{2\beta^2} (1 - \exp \frac{-S^2}{2c^2}) \text{ else} \end{cases}$$

where $R_A = \frac{|\lambda_2|}{|\lambda_3|}$, $R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}}$, and $S = \sqrt{{\lambda_1}^2 + {\lambda_2}^2 + {\lambda_3}^2}$. See [7] for a detailed explanation of the settings of each parameter in this measure.

In figure 8 you can observe the response of the filter, based on the Hessian information, at three different scales: $\sigma = 1, 5, 10$. Visualization is made with Maximum Intensity Projection (MIP). Using this information computed at several scales, we can take as

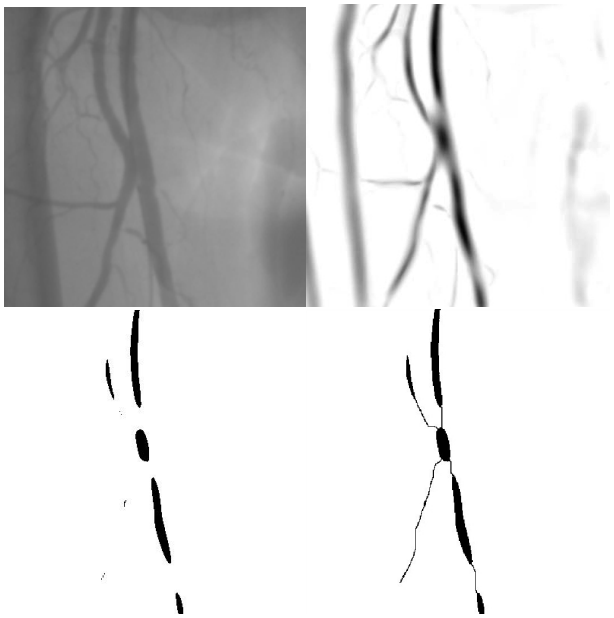

Figure 6: Medical Image. First line: original image and vesselness potential; Second line: from the set of regions obtained from thresholding of potential image, our method finds links between these regions as minimal paths with respect to the potential.

potential the maximum of the response of the filter across all scales (Fig. 9-left). And we can easily give a very constrained threshold of this image, that will lead to sets of unstructured voxels that surely belong to the anatomical object of interest, as shown in figure 9-middle.

Based on this set of regions, we apply our algorithm of section 3, using the 3D version of the Fast-Marching algorithm presented briefly in section 2.4 and which is detailed in [5]. We find the set of paths that connect altogether all the seed regions in our image, leading to the representation shown in figure 9-right.

5 Conclusion

We presented a new method that finds a set of contour curves in an image. It was applied to perceptual grouping to get complete curves from a set of edge regions with gaps. The technique is based on finding minimal paths between two end points [4]. However, in our approach, start and end points are not required as initialization. Given a unstructured set of regions, the pairs of regions that had to be linked by minimal paths are automatically found. Once *saddle points* between pairs of regions are found, paths are drawn on the image from the selected *saddle points* to both points of each pair. This gives the minimal paths between selected pairs of regions. The whole set of paths completes the initial set of contours and allows to close these contours. We applied this method in order to reconstruct vascular structures, and we showed examples for 2D vascular image and 3D medical dataset of the aorta. In case a refinement is needed, this method could be an efficient way to initialize geodesic contours.

References

[1] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.

[2] L. D. Cohen. Multiple contour finding and perceptual grouping using minimal paths. *Journal of Mathematical Imaging and Vision*, 14(3), 2001.

[3] Laurent D. Cohen. On active contour models and balloons. *CVGIP:IU*, 53(2):211–218, March 1991.

[4] Laurent D. Cohen and R. Kimmel. Global minimum for active contour models: A minimal path approach. *IJCV*, 24(1):57–78, August 1997.

[5] T. Deschamps and L. D. Cohen. Minimal paths in 3D images and application to virtual endoscopy. In *Proc. ECCV'00*, Dublin, Ireland, June 2000. Long paper to appear in *Medical Image Analysis*.

[6] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math.*, 1:269–271, 1959.

[7] A. Frangi and W. Niessen, Multiscale Vessel Enhancement Filtering. MICCAI'98, Cambridge.

[8] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *IJCV*, 20(1/2) Oct. 1996.

[9] M. Kass, A. Witkin and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, Jan. 1988.

[10] R. Kimmel, A. Amir, and A. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE* PAMI-17(6):635–640, June 1995.

[11] R. Kimmel, N. Kiryati, and A. M. Bruckstein. Distance maps and weighted distance transforms. *JMIV*, 6:223–233, May 1996.

[12] L.M. Lorigo, O.D. Faugeras, W.E.L. Grimson, R. Keriven, R. Kikinis, A. Nabavi, and C.F. Westin. Codimension-two geodesic active contours for the segmentation of tubular structures. In *CVPR00*, pages I:444–451, 2000.

[13] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE PAMI*, 17(2):158–175, february 1995.

[14] A. Sarti, R. Malladi, and J.A. Sethian. Subjective surfaces: A method for completing missing boundaries. *PNAS*, 12(97):6258–6263, 2000.

[15] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences.* Cambridge Univ. Press, 1996.

[16] A. Shaashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proc. ICCV'88*, Dec. 1988.

[17] A. Vasilevskiy and K. Siddiqi. Flux maximizing geometric flows. In *ICCV01*, pages I: 149–154, 2001.

[18] L. R. Williams and D. W. Jacobs. stochastic completion fields: a neural model of illusory contour shape and salience. In *Proc. ICCV'95*, June 1995.

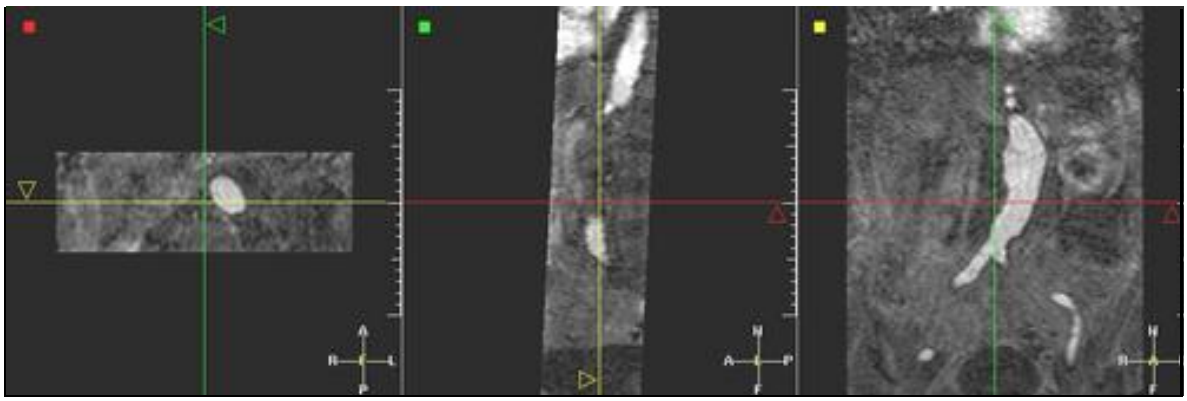

Figure 7: Three orthogonal slices of the aorta MR dataset

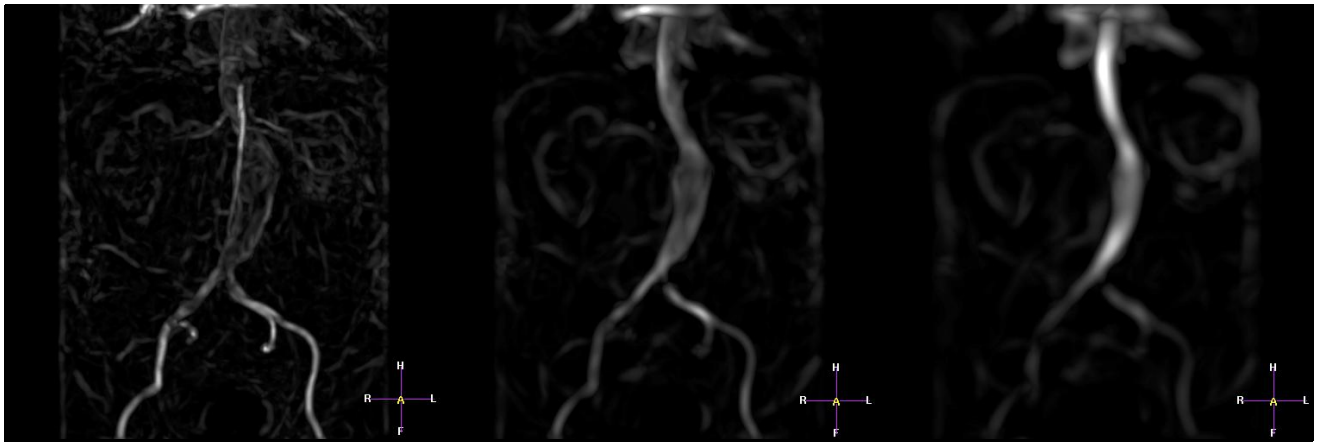

Figure 8: Ridge detection at three different scales ($\sigma = 1, 5, 10$) (MIP visualization of the 3D images)

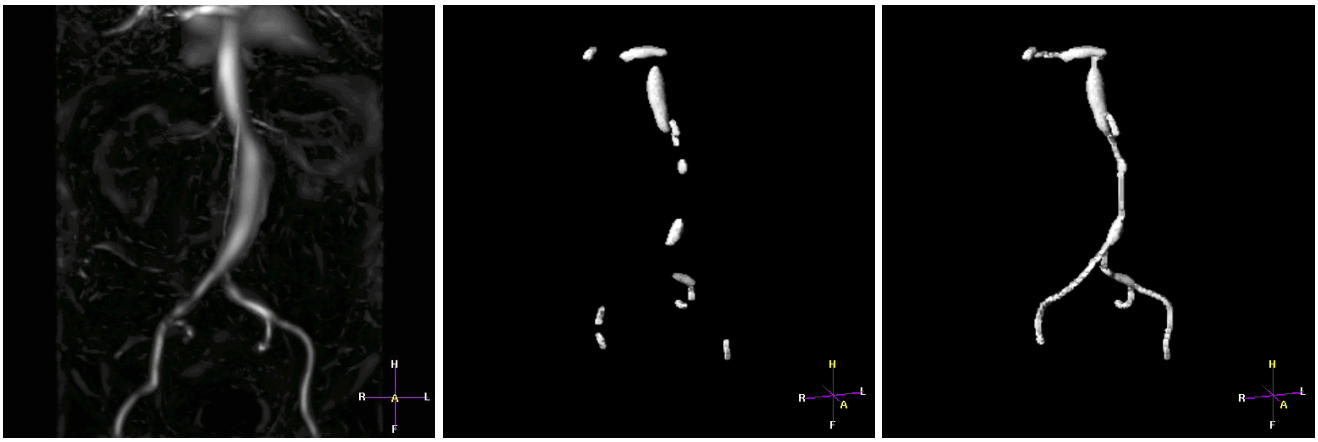

Figure 9: Perceptual Grouping in the aorta of figure 7: from left to right, visualization of the 3D potential (MIP view) obtained from the different scale of previous figure; a rough detection of the aorta; the Reconstructed aorta.