Multiple Contour Finding and Perceptual Grouping as a set of Energy Minimizing Paths

Laurent D. COHEN[1] and Thomas DESCHAMPS[2]

[1] CEREMADE, UMR 7534, Université Paris-Dauphine 75775 Paris cedex 16, France
cohen@ceremade.dauphine.fr
[2] Laboratoire d'Electronique Philips France,
thomas.deschamps@philips.com

Abstract. We address the problem of finding a set of contour curves in an image. We consider the problem of perceptual grouping and contour completion, where the data is a set of points in the image. A new method to find complete curves from a set of contours or edge points is presented. Our approach is an extension of previous work on finding a set of contours as minimal paths between end points using the fast marching algorithm. Given a set of key points, we find the pairs of points that have to be linked and the paths that join them. We use the saddle points of the minimal action map. The paths are obtained by backpropagation from the saddle points to both points of each pair.
We also propose an extension of this method for contour completion where the data is a set of connected components. We find the minimal paths between each of these components, until the complete set of these "regions" is connected. The paths are obtained using the same backpropagation from the saddle points to both components.
Keywords:Perceptual grouping, salient curve detection, active contours, minimal paths, fast marching, level sets, weighted distance, reconstruction, energy minimization, medical imaging.

1 Introduction

We are interested in perceptual grouping and finding a set of curves in an image with the use of energy minimizing curves.

Since their introduction, active contours [12] 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 of different objects, we need many active contours to be initialized on the image. The level sets paradigm [15, 1] allowed changes in topology. It enables to get multiple contours by starting with a single one. However, these do not give satisfying results when there are gaps in the data since the contour may propagate into a hole and then split to many curves where only one contour is desired. This is the problem encountered with perceptual grouping where a set of incomplete contours is given. For example, in a binary image like the ones in figure 1 with a drawing of a shape with holes and spurious edge points, human vision can easily fill in the missing boundaries, remove the spurious ones and form complete curves. Perceptual grouping is an old problem in computer vision. It has been approached more recently with energy methods [17, 11, 18]. These methods find a criteria for saliency of a curve component or for each point of the image. In these methods, the definition of saliency measure is based indirectly on a second order regularization snake-like energy ([12]) of a path containing the point. However, the final curves are obtained generally in a second step as ridge lines of the saliency criteria after thresholding. In [19] a similarity between snakes and stochastic completion field is reported. Motivated by this relation between energy minimizing curves like snakes and completion contours, we are interested in finding a set of completion contours on an image as a set of energy minimizing curves.

In order to solve global minimization for snakes, the authors of [6] used the minimal paths, as introduced in [14, 13]. The goal was to avoid local minima without demanding too much on user initialization, which is a main drawback of classic snakes [4]. Only two end points were needed. The numerical method has the advantage of being consistent (see [6]) and efficient using the Fast Marching algorithm introduced in [16]. In this paper we propose a way to use this minimal path approach to find a set of curves drawn between points in the image. In order to find a set of most salient contour curves in the image, we draw the minimal paths between pairs of points.

We are also interested in finding a set of curves in an image between a set of connected components. This extension of our perceptual grouping technique finds its application in the completion of tube-like structures in images. The problem is here to complete a partially detected object, based on a number of connected components that belong to this object.

In our examples, the potential P to be minimized along the curves is usually an image of edge points that represent simple incomplete shapes. These edge points are represented as a binary image with small potential values along the edges and high values at the background. Such a potential can be obtained from real images by edge detection (see [5]). 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. In these cases the chosen function has to be such that the potential is positive everywhere, and it has to be decreasing in order to have edge points as minima of the potential.

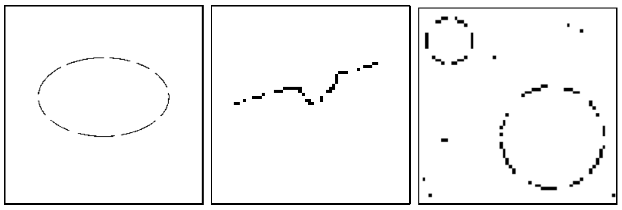

Fig. 1. Examples of incomplete contours

The potential could also be a grey level image as in [6]. It can also be a more complicated function of the grey level, as in [7] or in section 4.4.

The problems we solve in this paper are presented as follows:

- Minimal path between two points: The solution proposed in [6] is reviewed in Section 2.
- Minimal paths between a given set of pairs of points is a simple application of the previous one.
- Minimal paths between a given set of unstructured points: a way to find the pairs of linked neighbors and paths between them is proposed in Section 3.
- Minimal paths between a given set of connected components: we propose to find the set of minimal paths that link altogether this set of regions, and we show an example for a medical image in section 4.
- Minimal paths between an unknown set of point: In [2] we propose the automatic finding of key points among a larger set of admissible points and the drawing of minimal paths that leads to completed curves.

2 Minimal Paths and weighted distance

2.1 Global minimum for Active Contours

We present in this section the basic ideas of the method introduced in [6] to find the global minimum of the active contour energy using minimal paths. The energy to minimize is similar to classical deformable models (see [12]) where it combines smoothing terms and image features attraction term (Potential P):

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

where $C(s)$ represents a curve drawn on a 2D image and Ω is its domain of definition. The authors of [6] have related this problem with the recently introduced paradigm of the level-set formulation. In particular, its Euler equation is equivalent to the geodesic active contours [1]. The method introduced in [6] improves energy minimization because the problem is transformed in a way to find the global minimum.

2.2 Problem formulation

In [6], contrary to the classical snake model (but similarly to geodesic active contours), s represents the arc-length parameter, which means that $\|C'(s)\| = 1$, leading to a geometric energy form. Considering a simplified energy model without the second derivative term leads to the expression $E(C) = \int \{w\|C'\|^2 + P(C)\}ds$. Assuming that $\|C'(s)\| = 1$ leads to the formulation

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

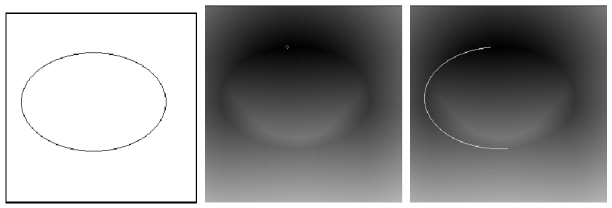

Fig. 2. Finding a minimal path between two points. On the left, the potential is minimal on the ellipse. In the middle, the minimal action or weighted distance to the marked point. On the right, minimal path using backpropagation from the second point.

The regularization of this model is now achieved by the constant $w > 0$ (see [6] for details).
We now have an expression in which the internal energy can be included in the external potential. Given a potential $P \geq 0$ that takes lower values near desired features, we are looking for paths along which the integral of $\tilde{P} = P + w$ is minimal. The surface of minimal action $\mathcal{U}$ is defined as the minimal energy integrated along a path between a starting point p_0 and any point p:

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

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. Assuming that potential $P \neq 0$ (this is always the case for $\tilde{P}$), the action map has only one local minimum which is the starting point p_0. The minimal path is found by a simple back-propagation, that is a gradient descent on the minimal action map $\mathcal{U}$ starting from p_1 until p_0 is reached. Thus, contour initialization is reduced to the selection of the two extremities of the path. We explain in the next section how to compute efficiently the action map $\mathcal{U}$.

2.3 Fast Marching Resolution

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

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

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 [16], was used in [6] noticing that the map $\mathcal{U}$ satisfies the Eikonal equation:

$$||\nabla \mathcal{U}|| = \tilde{P} \quad \text{and} \quad \mathcal{U}(p_0) = 0. \tag{5}$$

Classic finite differences schemes for this equation tend to overshoot and are unstable. An up-wind scheme was proposed by [16]. It relies on a one-sided

```
Algorithm for 2D Fast Marching
  – Definitions:
      • Alive set: all 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 Equation (6) 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: reduced to the starting point $p_0$, with $U(p_0) = \mathcal{U}(p_0) = 0$;
      • 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, with $\mathcal{U} = U = \infty$;
  – 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 (i.e. $\mathcal{U}(p) = U_{i_{min},j_{min}}$ is
        frozen);
      • For each neighbor $(i, j)$ (4-connexity in 2D) of $(i_{min}, j_{min})$:
          ∗ If $(i, j)$ is Far, add it to the Trial set and compute $U_{i,j}$ using Eqn.
            6;
          ∗ If $(i, j)$ is Trial, update the action $U_{i,j}$ using Eqn. 6.
```

Table 1. *Fast Marching* algorithm

derivative that looks in the up-wind direction of the moving front, and thereby avoids the over-shooting associated with finite differences:

$$\begin{aligned}&(\max\{u - \mathcal{U}_{i-1,j}, u - \mathcal{U}_{i+1,j}, 0\})^2 + \\ &(\max\{u - \mathcal{U}_{i,j-1}, u - \mathcal{U}_{i,j+1}, 0\})^2 = \tilde{P}^2_{i,j},\end{aligned} \tag{6}$$

giving the correct viscosity-solution u for $\mathcal{U}_{i,j}$. 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 (6).

This technique of considering at each step only the necessary set of grid points was originally introduced for the construction of minimum length paths in a graph between two given nodes in [8].

The algorithm is detailed in Table 1. An example is shown in Figure 2. The *Fast Marching* technique selects at each iteration the *Trial* point with minimum action value. In order to compute this value, we have to solve Equation (7) for each trial point, as detailed in table 2.

2.4 Algorithm for 2D Up-Wind Scheme

Notice that for solving Equation (6), only alive points are considered. This means that calculation is made using current values of $\mathcal{U}$ for neighbors and not estimate U of other trial points. Considering the neighbors of grid 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 $\mathcal{U}(A_1) \le \mathcal{U}(A_2)$, $\mathcal{U}(B_1) \le \mathcal{U}(B_2)$, and $\mathcal{U}(A_1) \le \mathcal{U}(B_1)$. Considering that we have $u \ge \mathcal{U}(B_1) \ge \mathcal{U}(A_1)$, the equation derived is

$$(u - \mathcal{U}(A_1))^2 + (u - \mathcal{U}(B_1))^2 = \tilde{P}^2_{i,j} \tag{7}$$

Computing the discriminant Δ of Equation (7) we have the steps described in table 2.

1. – If $\Delta \ge 0$, u should be the largest solution of Equation (7);
     - If the hypothesis $u \ge \mathcal{U}(B_1)$ is wrong, go to 2;
     - If this value is larger than $\mathcal{U}(B_1)$, this is the solution;

   – If $\Delta < 0$, B_1 has an action too large to influence the solution. It means that $u \ge \mathcal{U}(B_1)$ is false. Go to 2;

   Simple calculus can replace case 1 by the test:

1bis. If $\tilde{P}_{i,j} > \mathcal{U}(B_1) - \mathcal{U}(A_1)$,
   $u = \frac{\mathcal{U}(B_1) + \mathcal{U}(A_1) + \sqrt{2\tilde{P}^2_{i,j} - (\mathcal{U}(B_1) - \mathcal{U}(A_1))^2}}{2}$ is the largest solution of Equation (7)
   else go to 2;
2. Considering that we have $u < \mathcal{U}(B_1)$ and $u \ge \mathcal{U}(A_1)$, we finally have $u = \mathcal{U}(A_1) + \tilde{P}_{i,j}$.

Table 2. Solving locally the upwind scheme

Thus it needs only one pass over the image. To perform efficiently these operations in minimum time, the *Trial* points are stored in a min-heap data structure (see details in [16]). Since the complexity of the operation of changing the value of one element of the heap is bounded by a worst-case bottom-to-top proceeding of the tree in $O(\log_2 P)$, the total work is bounded $O(P \log_2 P)$ for the *Fast Marching* on a grid with P nodes.

3 Finding multiple contours from a set of key points p_k

The method of [6], detailed in the previous section allows to find a minimal path between two endpoints. We are now interested in finding many or all contours in an image. A first step for multiple contours finding in an image is to assume we have a set of points p_k given on the image and then find contours passing through these points. We assume the points are either given by a preprocessing or by the user. We propose to find the contours as a set of minimal paths that link pairs of points among the p_k's. If we also know which pairs of points have to be linked together, finding the whole set of contours is a trivial application of the previous section. This would be similar to the method in [10] which used a dynamic programming (non consistent, see [6]) approach to find the paths between successive points given by the user. The problem we are interested in here is also to find out which pairs of points have to be connected by a contour. Since the set of points p_k's is assumed to be given unstructured, we do not know in advance how the points connect. This is the key problem that is solved here using a minimal action map.

3.1 Main ideas of the approach

Our approach is similar to computing the distance map to a set of points 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 p_k. Instead of computing a minimal action map for each pair of points, as in Section 2, 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 points 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 point p_0. Since all points p_k are set with $\mathcal{U}(p_k) = 0$, we now have one front evolving from each of the starting points p_k. In what follows when we talk about front meeting, we mean either the geometric point where the two fronts coming from different p_k's meet, or in the discrete algorithm the first alive point which connects two components from different p_k's (see Figures 3 and 4).
Our problem is related to the approach presented at the end of [6] in order to find a closed contour. Given only one end point, the second end point was found as a saddle point. This point is where the two fronts propagating both ways meet. Here we use the fact that given two end points p_1 and p_2, the saddle point S where the two fronts starting from each point meet can be used to find the minimal path between p_1 and p_2. Indeed, the minimal path between the two points has to pass by the meeting point S. This point is the point half way (in energy) on the minimal path between p_1 and p_2. Backpropagating from S to p_1 and then from S to p_2 gives the two halves of the path. This is in fact an approximation, due to some discretization error in finding the meeting point S. If high precision is needed, a subpixel location of *saddle points* can be made based on the final energy map. In order to get the precise minimal path between the two points, we could also backpropagate from the second point to the first as in Section 2, but computation time would be then much increased.

3.2 Some definitions

Here are some definitions that will be used in what follows.

- For a point p in the image, we note $\mathcal{U}_p$ the minimal action obtained by Fast Marching with potential $\tilde{P}$ and starting point p.

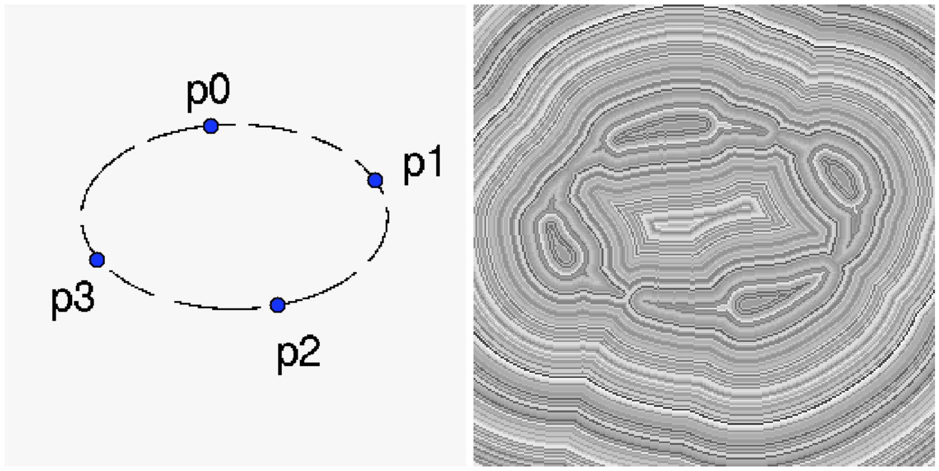

Fig. 3. Ellipse example with four points. On the left the incomplete ellipse as potential and four given points; on the right the minimal action map (random LUT to show the level sets) from these points.

- 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 and all their 4-connexity neighbors are *trial* points. This is easy to see that $\mathcal{U}_X = \min_{p \in X} \mathcal{U}_p$.
- The *region* R_k associated with a point p_k is the set of points p of the image closer in energy to p_k than to other points p_j. This means that minimal action $\mathcal{U}_{p_k} \leq \mathcal{U}_{p_j}, \forall j \neq k$. Thus, if $X = \{p_j, 0 \leq j \leq N\}$, we have $\mathcal{U}_X = \mathcal{U}_{p_k}$ on R_k and the computation of $\mathcal{U}_X$ is the same as the simultaneous computation of each $\mathcal{U}_{p_k}$ on each region R_k. These are the simultaneous fronts starting from each p_k.
- The *region index* r is $r(p) = k, \forall p \in R_k$. (Voronoi Diagram for weighted distance).
- A *saddle point* $S(p_i, p_j)$ between p_i and p_j is the first point where the front starting from p_i to compute $\mathcal{U}_{p_i}$ meets the front starting from p_j to compute $\mathcal{U}_{p_j}$; At this point, $\mathcal{U}_{p_i}$ and $\mathcal{U}_{p_j}$ are equal and this is the smallest value for which they are equal.
- Two points among the p_k's will be called *linked neighbors* if they are selected to be linked together. The way we choose to link two points is to select some *saddle points*. Thus points p_i and p_j are *linked neighbors* if their *saddle point* is among the selected ones.

3.3 Saddle points and Reconstruction of the set of curves

The main goal of our method is to obtain all significant paths joining the given points. However, each point should not be connected to all other points, but

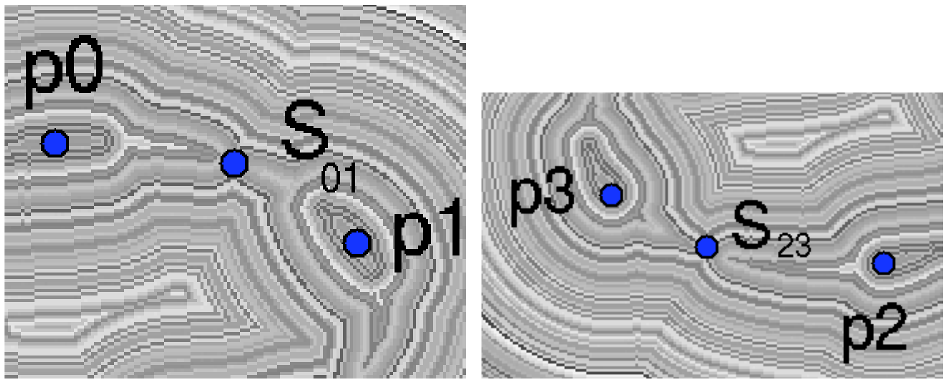

Fig. 4. Zoom on a *saddle point* between two key points.

only to those that are closer to them in the energy sense. In order to form closed curves, each point p_k should not have more than two *linked neighbors*. The criteria for two points p_i and p_j to be connected is that their fronts meet before other fronts. It means that their *saddle point* $S(p_i, p_j)$ has lower action $\mathcal{U}$ than the *saddle points* between these points and other points p_k. The fact that we limit each p_k to have no more than two connections makes it possible that some points will have only one or no connection. This helps removing some isolated spurious points or getting different closed curves not being connected together. We illustrate this in the example of Figure 6 where one of the p_k is not linked to any other point since all the other points already have two linked neighbors. In case we also need to have T-junctions, the algorithm can be used with a higher number of linked neighbors allowed for each endpoint or we may connect together all possible points as in Section 4.3. A non symmetric relation may also be used to link each point to the closest or the two closest ones, regardless of whether these have already two or more neighbors. In the exemple of Figure 6, such an approach would link the spurious points with the circles. Postprocessing would be needed to remove undesired links, based on high energy for example.

Once a *saddle point* $S(p_i, p_j)$ is found and selected, backpropagation relatively to final energy $\mathcal{U}$ should be done both ways to p_i and to p_j to find the two halves of the path between them. We see in Figure 5 this backpropagation at each of the four *saddle points*. 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 p_i, all points in a region different from R_i have their energy put artificially to ∞. This allows finding the good direction for the gradient descent towards p_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 neighbors*.

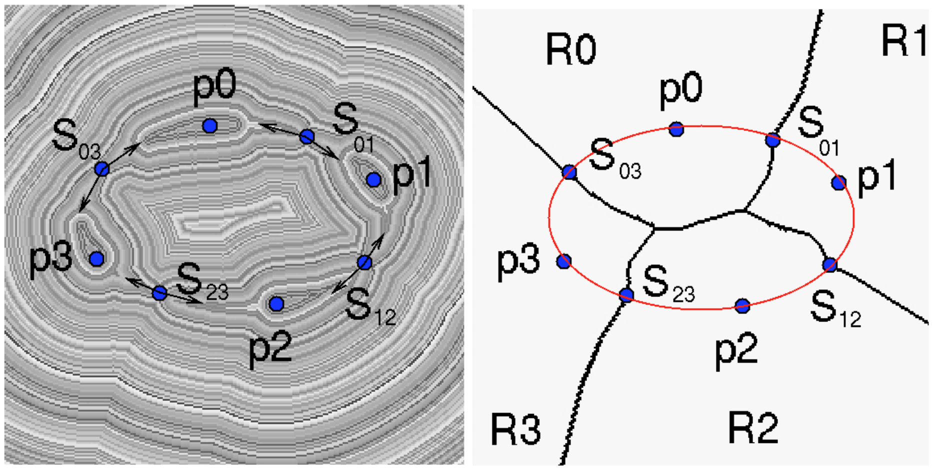

Fig. 5. Ellipse example with four points. On the left the *saddle points* are found, and backpropagation is made from them to each of the two points from where the front comes; on the right, the minimal paths and the Voronoi diagram obtained.

As defined above, the *region* R_k associated with a point p_k is the set of points p of the image such that minimal energy $\mathcal{U}_{p_k}(p)$ to p_k is smaller than all the $\mathcal{U}_{p_j}(p)$ to other points p_j. The set of such *regions* R_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*. For a point p on the boundary between R_j and R_k, we have $\mathcal{U}_{p_k}(p) = \mathcal{U}_{p_j}(p)$. The *saddle point* $S(p_k, p_j)$ is a point on this boundary with minimal value of $\mathcal{U}_{p_k}(p) = \mathcal{U}_{p_j}(p)$. This gives us a rule to find the *saddle points* during the fast marching algorithm.

Each time two fronts coming from p_k and p_j meet for the first time, we define the meeting point as $S(p_k, p_j)$. 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 p_k starts with index k. Each time a point is set as alive, it gets the same index as the points it was computed from in formula (6). 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 from the same *region*, except if (i, j) is on the boundary between two regions. If A_1 and B_1 are both alive and with different indexes 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* $S(p_k, p_l)$ between these regions. A point on the boundary between R_k and R_l is given the index of the neighbor point with smaller action A_1. At the boundary between two regions there can be a slight error on indexing. This error of at most one pixel is not important in our context and could be refined if necessary.

```
Algorithm with previously defined $p_k$

– Initialization:
    • $p_k$'s are given
    • $\forall k, V(p_k) = 0; r(p_k) = k;$  $p_k$ alive.
    • $\forall p \notin \{p_k\}, V(p_k) = \infty; r(p) = -1;$  $p$ is far except 4-connexity neighbors
      of $p_k$'s that are trial with estimate $U$ using Eqn. (6).
– Loop for computing $V = \mathcal{U}_{\{p_k, 0 \le k \le N\}}$:
    • 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 $r(p)$ with the same index as point $A_1$ in formula (6). If $r(A_1) \neq$
      $r(B_1)$ and we are in case 1 of table 2 where both points are used and if
      this is the first time regions $r(A_1)$ and $r(B_1)$ meet, $S(p_{r(A_1)}, p_{r(B_1)}) =$
      $p$ is set as a saddle point between $p_{r(A_1)}$ and $p_{r(B_1)}$. If these points
      have not yet two linked neighbors, they are put as linked neighbors and
      $S(p_{r(A_1)}, p_{r(B_1)}) = p$ is selected,
      For each neighbor $(i, j)$ (4-connexity) of $(i_{min}, j_{min})$:
        ∗ If $(i, j)$ is Far, add it to the Trial set and compute $U$ using Eqn. (6);
        ∗ If $(i, j)$ is Trial, recompute the action $U_{i,j}$, and update it.
– Obtain all paths between selected linked neighbors by backpropagation each
  way from their saddle point (see Section 3.3).
```

Table 3. Algorithm of Section 3

3.4 Algorithm and results

The algorithm for this section is described in Table 3 and illustrated in figures 3 and 5. When there is a large number of p_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 neighbors and *saddle points*. This may generate more conflicting neighbor points, and due to the constraint of having at most two linked neighbors, some gaps may remain between contours. The method can be applied to a whole set of edge points or points obtained through a preprocessing. However, choosing few key points simplifies the computation of *saddle points* and *linked neighbors* and the geometry of the paths. When there are few key points, they are not too close to each other. 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. In [2] we give a way to find automatically a set of key points from a larger set of "admissible points".

We show in figure 6 the results of the approach for the data given in figure 1. We show in figure 7 an application of our approach combined with the saliency map of [11]. In such an example, the given dots are too few to enable finding the ellipse as a minimal path. Indeed, taking two opposite points on the ellipse, the minimal path between them will not be along the ellipse but rather along a straight line. By passing through low potential points (in black) along the ellipse,

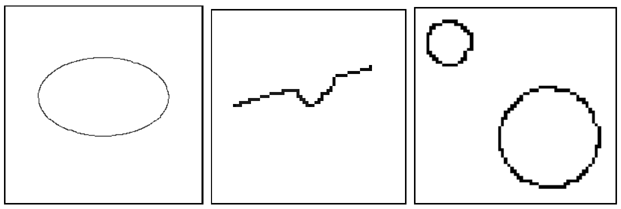

Fig. 6. Final paths obtained for images of Figure 1

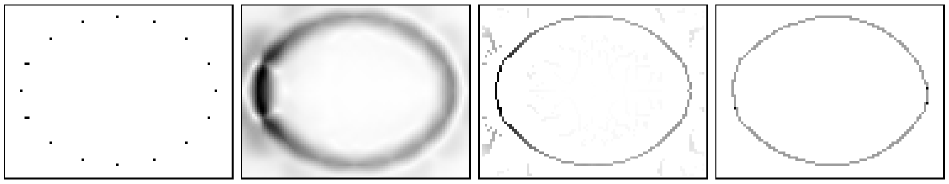

Fig. 7. Finding a set of minimal path using a saliency map as potential. From left to right, original data, saliency map, ridge lines and minimal paths.

the path will also pass through more points with high potential (background in white). Thus applying the method of [11] gives a saliency map that is much more dense than the original image. Taking the saliency map as potential, our approach allows finding the whole ellipse as a set of minimal paths between points determined automatically. The set of points to be linked here can be either the initial set of points of the original image or the set of points obtained by thresholding the saliency map. We can also find the ellipse by looking for ridge curves on the saliency map but there are many spurious ridge curves obtained.

4 Extension to Finding multiple contours from a set of connected components

4.1 Minimal Path between two Regions

The method of [6], 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

$$\mathcal{U}(p) = \inf_{\mathcal{A}_{R_0,p}} E(C) = \inf_{p_0 \in R_0} \inf_{\mathcal{A}_{p_0,p}} E(C) \tag{8}$$

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.

4.2 Minimal Paths from a Set of Connected Components

Here, our approach is to compute the distance map to a set of unstructured set of points where connected points are considered as regions, using a weighted distance defined through the potential P. The set of paths is obtained with the minimal action with respect to P with zero value at all regions R_k. This method has the same possibilities as the one presented in section 3, as we only need to compute one minimal action map in order to find all paths. In the same way, we find the *saddle points* between each pair of propagating fronts from two regions R_1 and R_2. And we compute the minimal paths between two regions R_1 and R_2 by back-propagating from each selected saddle-point until we meet a pixel belonging respectively to regions R_1 and R_2. The notion of linked neighbors is extended to linked regions. More details can be found in [3].

4.3 Connecting all regions with minimal paths

The goal here is to connect all regions, but only to those that are closer to them in the energy sense. In section 3, we were looking for closed contours from a set of points, and had constraints on the maximal number of linked neighbors. In the application we show below, we are interested in connecting all initial given points through a path. In this case we also need to have T-junctions, for reconstructing tree-like structures, therefore the algorithm can be used with a higher number of linked regions allowed for each region, as said for the end points in subsection 3.3. The constraint we use now is to avoid creating a closed contour when connecting two regions. This is obtained through the definition of cycles and cycle tests.

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*. A cycle test 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 connection 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 for the first time, we set $M(i, j) = M(j, i) = 1$. Thus, when two regions meet, we apply the algorithm detailed in table 4.

If two regions are already connected, the pixel where their fronts meet is not considered as a valuable candidate for back-propagation.

The algorithm stops automatically when all regions are connected.

Once a *saddle point* between any region R_i and R_j is found and selected, back-propagation relatively to final energy $\mathcal{U}$ should be done both ways to p_i and to p_j, the first points belonging to respectively R_i and R_j, during the back-propagation algorithm.

```
**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 $k \neq i, k \neq j, k \neq l$ :
        if $M(k, j) = 1$, count += $Test(j, k, M, l)$;
      • return count;
```

Table 4. Cycle detection

4.4 Medical 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 8 an example of application for a hip medical image where we are looking for vessels. Potential P is defined using ideas from [9] on vesselness filter.

For this, we propose to extract valuable information from this dataset, computing a multiscale vessel enhancement measure, based on the work of [9] on ridge filters. Having extracted the two eigenvalues of the Hessian matrix computed at scale σ, ordered $|\lambda_1| \leq |\lambda_2|$, we define at each pixel a vesselness function

$$\nu(\sigma) = \begin{cases} 0, \text{ if } \lambda_2 \geq 0 \\ \exp \frac{-{R_B}^2}{2\beta^2}(1 - \exp \frac{-S^2}{2c^2}) \end{cases}$$

where $R_B = \frac{|\lambda_1|}{|\lambda_2|}$, and $S = \sqrt{{\lambda_1}^2 + {\lambda_2}^2}$. See [9] for a detailed explanation of the settings of each parameter in this measure. Using this information computed at several scales, we can take as new image the maximum of the response of the filter across all scales. In figure 8-top right you can observe the response of the filter, based on the Hessian information.

And we can easily give a very constrained threshold of this image, that will lead to sets of unstructured pixels that surely belong to the anatomical object of interest, as shown in figure 8-bottom left. Figure 8-bottom right shows the set of completion paths obtained that link all given connected components together.

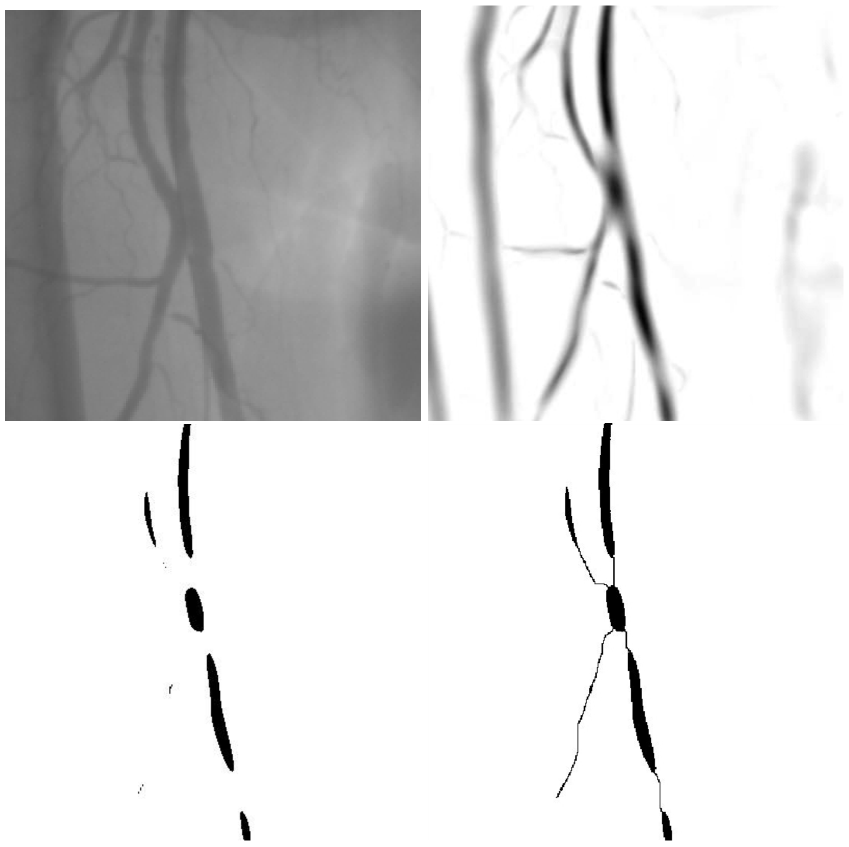

Fig. 8. Medical Image. First line: original image and vesselness potential; Second line: from the set of regions obtained from thresholding of potential image, our method finds links between these regions as minimal paths with respect to the potential.

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 noisy contours or edge points with gaps. The technique is based on previous work of finding minimal paths between two end points [6]. However, in our approach, we do not need to give the start and end points as initialization. In a first method, given a set of key points, we found the pairs of key points that had to be linked by minimal paths. Once *saddle points* between pairs of points 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 points. The whole set of paths completes the initial set of contours and allows to close these contours. In a second method, we compute the whole set of paths between unstructured regions in the image. We illustrate this approach with a medical image application.

References

1. V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 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), May 2001. CEREMADE TR 0101, Jan 2001. To appear.
3. L. D. Cohen and T. Deschamps. Grouping connected components using minimal path techniques. Technical report, CEREMADE, 2001. To appear.
4. Laurent D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, March 1991.
5. Laurent D. Cohen and Isaac Cohen. Finite element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-15(11):1131–1147, November 1993.
6. Laurent D. Cohen and R. Kimmel. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision*, 24(1):57–78, August 1997.
7. T. Deschamps and L.D. Cohen. Minimal paths in 3D images and application to virtual endoscopy. In *Proc. sixth European Conference on Computer Vision (ECCV'00)*, Dublin, Ireland, 26th June - 1st July 2000.
8. E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematic*, 1:269–271, 1959.
9. A. Frangi, W. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. In *Proc. Medical Image Computing and Computer-Assisted Intervention, MICCAI'98, Cambridge*, pages 130–137, 1998.
10. D. Geiger, A. Gupta, L. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):294–302, March 1995.
11. G. Guy and G. Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1/2):113–133, October 1996.
12. Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.
13. R. Kimmel, A. Amir, and A. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-17(6):635–640, June 1995.
14. R. Kimmel, N. Kiryati, and A. M. Bruckstein. Distance maps and weighted distance transforms. *Journal of Mathematical Imaging and Vision*, 6:223–233, May 1996. Special Issue on Topology and Geometry in Computer Vision.
15. R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. on PAMI*, 17(2):158–175, February 1995.
16. J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences.* Cambridge Univ. Press, 1996.
17. A. Shaashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proc. Second IEEE International Conference on Computer Vision (ICCV'88)*, pages 321–327, December 1988.
18. L. R. Williams and D. W. Jacobs. stochastic completion fields: a neural model of illusory contour shape and salience. In *Proc. Fifth IEEE International Conference on Computer Vision (ICCV'95)*, pages 408–415, Cambridge, USA, June 1995.
19. L. R. Williams and D. W. Jacobs. Local parallel computation of stochastic completion field. In *Proc. IEEE CVPR'96*, San Francisco, USA, June 1996.