International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Combination of piecewise-geodesic paths for interactive segmentation - Supplemental file: appendix B

Julien Mille · Sébastien Bougleux · Laurent D. Cohen

Received: date / Accepted: date

B Implementation details

B.1 Fast Marching propagation and extraction of saddle points

As described in Section 3.1, admissible paths joining two source points $\boldsymbol{a}$ and $\boldsymbol{b}$ are constructed by extracting the saddles points of the combined action map $U_{\boldsymbol{a},\boldsymbol{b}}$, located on the medial set $\mathcal{M}_{\boldsymbol{a},\boldsymbol{b}}$. The minimal action map $U_{\boldsymbol{a},\boldsymbol{b}}$ can be estimated, at each point of the discrete image domain $\tilde{\mathcal{D}} = \mathcal{D} \cap \mathbb{Z}^2$, by solving the discrete derivative free Eikonal equation [10, 6, 8]

$$\begin{cases} \forall \boldsymbol{x} \in \tilde{\mathcal{D}} \setminus \{\boldsymbol{a},\boldsymbol{b}\}, \quad U_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{x}) = \mathcal{F}_{\boldsymbol{x}}\left(U_{\boldsymbol{a},\boldsymbol{b}}\right), \\ U_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{a}) = U_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{b}) = 0, \end{cases} \tag{1}$$

where

$$\mathcal{F}_{\boldsymbol{x}}(U_{\boldsymbol{a},\boldsymbol{b}}) = \min_{\boldsymbol{y} \in \partial N(\boldsymbol{x})} U_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{y}) + W_P(\boldsymbol{y},\boldsymbol{x})\|\boldsymbol{x}-\boldsymbol{y}\|, \tag{2}$$

and $\partial N(\boldsymbol{x}) \subset \mathcal{D}$ defines the boundary of a topological ball, usually the boundary of the convex hull of the 4-connected or 8-connected grid neighbors of $\boldsymbol{x}$. According to (2), the minimal action at $\boldsymbol{x}$ is approximated by the minimal combination of the Euclidean distance between $\boldsymbol{x}$ and a point $\boldsymbol{y} \in \partial N(\boldsymbol{x})$, weighted by potential P along segment $\overline{\boldsymbol{x}\boldsymbol{y}}$, and the minimal action at $\boldsymbol{y}$ obtained by affine interpolation of $U_{\boldsymbol{a},\boldsymbol{b}}$ from its two nearest grid points in $\partial N(\boldsymbol{x})$. Let $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ be these two points. Then, update operator (2) becomes

$$\mathcal{F}_{\boldsymbol{x}}(U_{\boldsymbol{a},\boldsymbol{b}}) = \min_{\overline{\boldsymbol{x}_i\boldsymbol{x}_j} \in \partial N(\boldsymbol{x})} \min_{t \in [0,1]} \alpha(t), \tag{3}$$

$$\alpha(t) = (1-t)U_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{x}_i) + tU_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{x}_j) + W_P(\boldsymbol{y},\boldsymbol{x})\|\boldsymbol{x}-\boldsymbol{y}\|,$$

where $\boldsymbol{y} = (1-t)\boldsymbol{x}_i + t\boldsymbol{x}_j$. Several interpolation approaches have been explored to compute the weight W_P (see [1] for recent comparisons). In our experiments we have simply used $W_P(\boldsymbol{x},\boldsymbol{y}) = P(\boldsymbol{x})$. Regardless of the choice of the neighborhood and the weight, the solution of (1) can be approximated by the Fast Marching algorithm, which propagates two discrete fronts simultaneously, one starting from $\boldsymbol{a}$ and the other from $\boldsymbol{b}$, until they meet to form a discrete version of the medial set $\mathcal{M}_{\boldsymbol{a},\boldsymbol{b}}$, as illustrated in Fig. 1. Algorithm 1 presents a non-optimized version of the Fast Marching. Fronts are represented by a set Q initialized with the source points. At each step (while loop), a point of the fronts having a minimal action is removed from Q, and the minimal action of its neighbors is updated in consequence by solving (3) (see [6, 1, 8] for details). In particular, a neighbor which has not yet been explored by the fronts becomes a new point of the fronts and is thus added to Q. This propagation process is iterated until Q is empty, which guaranties the computation of $U_{\boldsymbol{a},\boldsymbol{b}}$ at each point of $\tilde{\mathcal{D}}$. Other instructions are related to the computation of the medial set, explained in the following.

Saddle points on the medial set $\mathcal{M}_{\boldsymbol{a},\boldsymbol{b}}$ are difficult to localize during the propagation, mainly due to discretization. It is more easy to compute a discrete medial set which is then traversed in order to approximate the saddle points. The medial set separates $\mathcal{D}$ into two regions, the first one composed of the points at least as close to $\boldsymbol{a}$ as to $\boldsymbol{b}$,

$$\mathrm{reg}_{\boldsymbol{a}} = \{\, \boldsymbol{x} \in \mathcal{D} \mid U_{\boldsymbol{a}}(\boldsymbol{x}) \leqslant U_{\boldsymbol{b}}(\boldsymbol{x}) \,\}, \tag{4}$$

and the other one, $\mathrm{reg}_{\boldsymbol{b}}$, defined similarly. The underlying partition of $\mathcal{D}$, called *Voronoi partition*, satisfies $\mathcal{D} = \mathrm{reg}_{\boldsymbol{a}} \cup \mathrm{reg}_{\boldsymbol{b}}$ and $\mathcal{M}_{\boldsymbol{a},\boldsymbol{b}} = \mathrm{reg}_{\boldsymbol{a}} \cap \mathrm{reg}_{\boldsymbol{b}}$. Together with this partition, we also define the *Voronoi map*

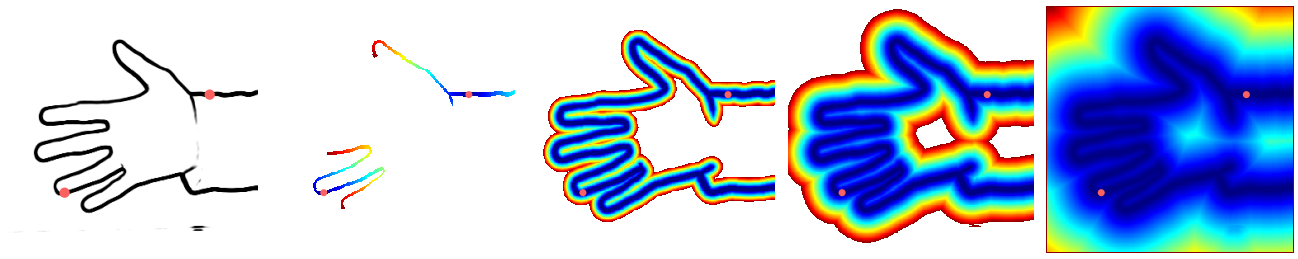
$$\mathrm{vor}(\boldsymbol{x}) = \left\{\, l(\boldsymbol{y}) \mid \boldsymbol{x} \in \mathrm{reg}_{\boldsymbol{y}},\ \boldsymbol{y} \in \{\boldsymbol{a},\boldsymbol{b}\} \,\right\}, \tag{5}$$

which provides for each point $\boldsymbol{x} \in \mathcal{D}$ the labels of its nearest source points, according to a label function $l : \{\boldsymbol{a},\boldsymbol{b}\} \to \{1,2\}$. Obviously, the medial set satisfies

J. Mille
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205
F-69622, Villeurbanne, France
E-mail: julien.mille@liris.cnrs.fr

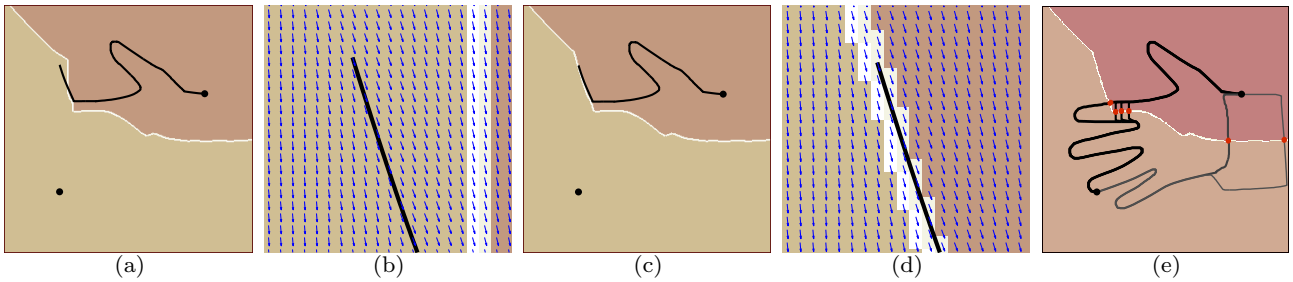
S. Bougleux
Université de Caen-Basse Normandie, GREYC, UMR6072
F-14050, Caen, France
E-mail: sebastien.bougleux@unicaen.fr

L. Cohen
Université Paris-Dauphine, CEREMADE, UMR7534
F-75016, Paris, France
E-mail: cohen@ceremade.dauphine.fr



(a) (b) (c) (d) (e)

Fig. 1 (a) Potential and two source points. (b-e) Several steps of the computation of the minimal action map.



(a) (b) (c) (d) (e)

Fig. 2 (a) Voronoi map obtained by updating Voronoi labels locally [2, 8]: points are not always well labelized, particularly in regions of $\mathcal{D}$ where the gradient of the minimal action map and the medial set are aligned. (b) A zoom of (a) with $-\nabla U_{\boldsymbol{a},\boldsymbol{b}}$. (c) Voronoi map constructed using back-propogation to determine labels (by Algorithm 1): minimal paths are ensured to be located in only one Voronoi region. (e) Saddles (red dots) and associated minimal paths between the two source points.

Algorithm 1 Minimal action map and Voronoi map.

func VoronoiMap($\boldsymbol{a}$,$\boldsymbol{b}$,P,PATH2LABEL)
input
$\boldsymbol{a},\boldsymbol{b}\in\tilde{\mathcal{D}}$: source points
$P:\tilde{\mathcal{D}}\to\mathbb{R}^{+*}$: discrete potential
PATH2LABEL : a method solving (6)
variables
$U:\tilde{\mathcal{D}}\to\mathbb{R}^{+}$: minimal action map
$\nabla U:\tilde{\mathcal{D}}\to\mathbb{R}^{2}$: gradient of U
$\mathrm{vor}:\tilde{\mathcal{D}}\to\{0,1,2\}$: Voronoi map
Q: front set encoded as a priority queue

$U[\boldsymbol{x}]:=+\infty$, $\mathrm{vor}[\boldsymbol{x}]:=0$, $\nabla U[\boldsymbol{x}]:=0$, $\forall\boldsymbol{x}\in\tilde{\mathcal{D}}$
$U[\boldsymbol{a}]:=U[\boldsymbol{b}]:=0$, $\mathrm{vor}[\boldsymbol{a}]:=1$, $\mathrm{vor}[\boldsymbol{b}]:=2$
$Q:=\{\boldsymbol{a},\boldsymbol{b}\}$
while $Q\neq\emptyset$ **do**
  $\boldsymbol{x}:=\mathrm{argmin}\{U[\boldsymbol{x}]\mid\boldsymbol{x}\in Q\}$
  $Q:=Q\setminus\{\boldsymbol{x}\}$
  $\mathrm{vor}[\boldsymbol{x}]:=$ PATH2LABEL$(\boldsymbol{x},\nabla U,\mathrm{vor})$
  for $\boldsymbol{y}\in N(\boldsymbol{x})$ **do**
    if $U[\boldsymbol{y}]=+\infty$ **then**
      $Q:=Q\cup\{\boldsymbol{y}\}$
    end if
    $u_{\mathrm{new}}:=\mathcal{F}_{\boldsymbol{y}}(U)$
    if $u_{\mathrm{new}}<U[\boldsymbol{y}]$ **then**
      $U[\boldsymbol{y}]:=u_{\mathrm{new}}$
      update $\nabla U[\boldsymbol{y}]$
    end if
  end for
end while
return $(U,\nabla U,\mathrm{vor})$

$\mathcal{M}_{\boldsymbol{a},\boldsymbol{b}}=\{\boldsymbol{x}\in\mathcal{D}\,|\,\mathrm{card}(\mathrm{vor}(\boldsymbol{x}))>1\}$. In practice, the Voronoi map is only estimated at points of $\tilde{\mathcal{D}}$, when (1) is solved. This is usually realized locally, in the neighborhood $N(\boldsymbol{x})$ of each point $\boldsymbol{x}\in\tilde{\mathcal{D}}$, by assigning to $\boldsymbol{x}$ the label of the grid point closest to the optimal neighbor $\boldsymbol{y}\in\partial N(\boldsymbol{x})$. Using the Fast Marching algorithm, labels are thus propagated simultaneously to minimal actions, starting with $\mathrm{vor}(\boldsymbol{a})=1$ and $\mathrm{vor}(\boldsymbol{b})=2$ (see [2, 8]). By construction, considering only one optimal point $\boldsymbol{y}\in\partial N(\boldsymbol{x})$ implies that each point of $\tilde{\mathcal{D}}$, being or not on the medial set $\mathcal{M}_{\boldsymbol{a},\boldsymbol{b}}$, is always connected to exactly one source point by a unique minimal path. As illustrated in Fig. 2(a), the resulting Voronoi map is not always consistent with the definition of a Voronoi partition, as some minimal paths may connect two points having different labels. This leads to errors in the localization of the medial set[1]. To overcome this drawback, which is mainly due to discretization, neighborhoods used to determine labels must be extended. This can be done by explicitely solving back-propagation (6) at each step of the Fast Marching algorithm (function PATH2LABEL in while loop of Algorithm 1). Let $\boldsymbol{x}\in Q$ be the front point having a minimal action and which is going to be labelized. Starting from $\boldsymbol{x}$, the back-propagation is stopped when a source point, or a point surrounded by points of $\tilde{\mathcal{D}}$ having the same Voronoi label, is reached. The proposed *discrete Voronoi map* is finally defined by $\tilde{\mathrm{vor}}:\tilde{\mathcal{D}}\to\{1,2\}$,

$$\tilde{\mathrm{vor}}(\boldsymbol{x})=\tilde{\mathrm{vor}}\left(\tilde{\gamma}(t_l)\right),\quad\forall\boldsymbol{x}\in\tilde{\mathcal{D}},\tag{6}$$

[1] Note that computing $U_{\boldsymbol{a},\boldsymbol{b}}$ in one propagation also introduces errors on the minimal actions, on and near the medial set, comparing to the map $\min(U_{\boldsymbol{a}},U_{\boldsymbol{b}})$ obtained with two separate propagations. But this should not affect so much the localization of the Voronoi regions.

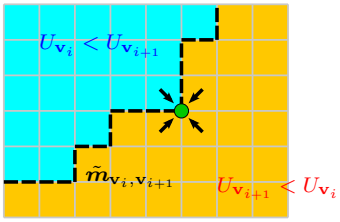


Fig. 3 Digital 4-connected medial curve on pixel edges. The combined action map on the medial point (pixel corner) is estimated using bilinear interpolation over the 4 neighboring pixels.

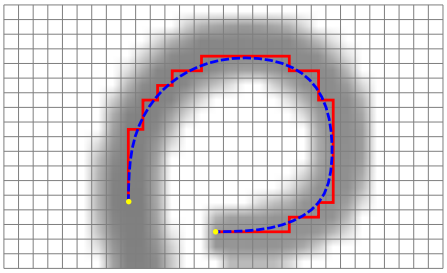


Fig. 4 Path with real coordinates with subpixel precision (dashed blue line) and its corresponding digital 4-connected path (plain red line) over the potential grid.

such that $\gamma \subset \mathcal{D}$ is the minimal path solving

$$\begin{cases} \forall t > 0, \quad \dfrac{d}{dt}\gamma(t) = -\dfrac{\nabla U_{\boldsymbol{a},\boldsymbol{b}}(\gamma(t))}{\left\|\nabla U_{\boldsymbol{a},\boldsymbol{b}}(\gamma(t))\right\|}, \\ \gamma(0) = \boldsymbol{x}, \end{cases} \tag{7}$$

$\tilde{\gamma}(t_l) \in \tilde{\mathcal{D}}$ is the nearest grid neighbor of $\gamma(t_l) \in \mathcal{D}$, and $t_l \geqslant 0$ satisfies

$$\tilde{\gamma}(t_l) \in \{\boldsymbol{a}, \boldsymbol{b}\}, \text{ or } \tilde{\text{vor}}(\tilde{\gamma}(t_l)) = \tilde{\text{vor}}(\boldsymbol{y}), \ \forall \boldsymbol{y} \in N_8(\tilde{\gamma}(t_l)),$$

where $N_8(\tilde{\gamma}(t_l)) \subset \tilde{\mathcal{D}}$ represents the 8-connected neighbors of $\tilde{\gamma}(t)$. Such a neighborhood ensures $\tilde{\gamma}(t_l)$ to be located in the interior of a Voronoi region being propagated. Remark that the gradient of the combined action map, involved in (7), can be estimated during the propagation from the updated minimal action (for loop of Algorithm 1). The proposed Voronoi labelling approach allows to preserve the computational advantages of the Fast Marching algorithm, while guarantying discrete Voronoi maps, and associated medial sets, consistent with the extraction of minimal paths, as observed in Fig. 2(c).

Based on the discrete Voronoi map, an approximation of the medial curve can now be defined as a sequence of connected pixel corners, as illustrated in Fig. 3. Since the two Voronoi regions are 4-connected sets, the discrete medial curve is 4-connected and simple. Hence, it can be extracted by a basic edge-linking procedure. Starting from any pixel corner on the boundary, pixel edges incident to pixels having different Voronoi labels are followed. Two situations may arise, whether the medial curve is open or closed. In the first situation, the two Voronoi regions are simply connected, i.e. without hole, and the medial curve reaches the boundaries of the image domain (see Fig. 2(e)). The edge-linking procedure should be run twice, taking the two possible directions starting from the initial pixel corner. In the second situation, one of the Voronoi region is entirely surrounded by the other, making the medial curve closed. Quite simply, the edge-linking procedure is stopped when the initial pixel corner is met again. Note that the orientation of the curve is not critical for the extraction of saddle points, thus the edge-linking step may be performed arbitrarily clockwise or counter-clockwise.

Let us shorten $U_{\boldsymbol{a},\boldsymbol{b}}$ to U, and let $\tilde{\boldsymbol{m}}$ be the discrete medial curve approximating $\boldsymbol{m}_{\boldsymbol{a},\boldsymbol{b}}$. We chose to estimate U at each pixel corner on $\tilde{\boldsymbol{m}}$ by bilinear interpolation. Trivially, the combined action at a pixel corner is taken as the average of the combined actions at the four neighboring pixels. To account for possible noise, values of the combined action

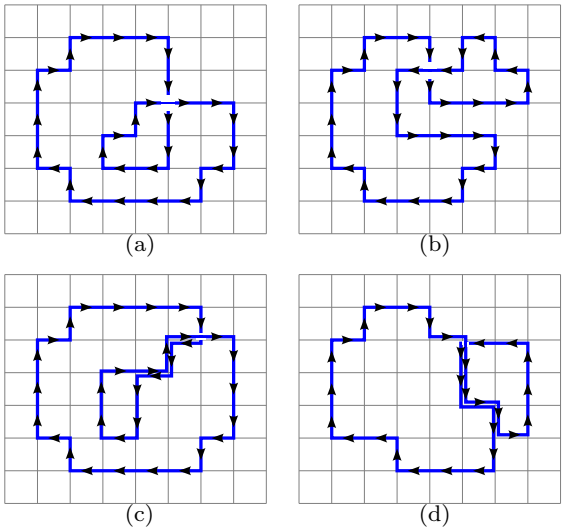


Fig. 5 Various types of simples loops on digital 4-connected curves: (a) Positive crossing with no overlapping section (b) Negative crossing with no overlapping section (c) Positive crossing with overlapping section (d) Negative crossing with overlapping section.

are smoothed along $\tilde{\boldsymbol{m}}$ by Laplacian regularization. In order to extract robust saddle points in spite of the effect of discretization, local minima are considered up to second order. A point $\tilde{\boldsymbol{m}}_j$ is marked as a saddle point of U if

$$U(\tilde{\boldsymbol{m}}_{j-2}) < U(\tilde{\boldsymbol{m}}_{j-1}) < U(\tilde{\boldsymbol{m}}_j) < U(\tilde{\boldsymbol{m}}_{j+1}) < U(\tilde{\boldsymbol{m}}_{j+2}).$$

Actually, not all saddle points are kept as starting points for path construction. In practice, we limit the number of admissible paths per set so that it does not exceed a user-defined threshold $K_{\max}$. If the number of detected local minima is greater than the threshold, they are sorted by increasing combined action and only the $K_{\max}$ lowest saddle points are taken as initial points for admissible paths.

B.2 Using digital curves

Discrete computation of region integrals involved in the region term, in Eq. (16), as well as the amount of self-tangency and twisting in Eqs. (9) and (11) is performed over digital 4-connected curves. Gradient descent over the minimal action

map generates admissible paths which are sampled continuous curves, i.e. sequences of points in $\mathbb{R}^2$ separated by an arbitrary step. Approximating region integrals and detecting intersections is more convenient on curves with integer coordinates and constant spacing between points, thus each admissible path is discretized as an open digital 4-connected curve (see Fig. 4). The resulting digital assembled contour $\tilde{\Gamma}$ is a sequence of $\tilde{n}$ points in $\mathbb{Z}^2$,

$$\tilde{\Gamma} = \langle \tilde{\Gamma}_1, \tilde{\Gamma}_2, \ldots, \tilde{\Gamma}_{\tilde{n}} \rangle,$$

such that $\left\| \tilde{\Gamma}_{i+1} - \tilde{\Gamma}_i \right\| = 1, \ \forall 1 \leq i \leq \tilde{n} - 1$, i.e. successive points are distinct and differ by only one coordinate. Moreover, the sequence is closed, hence $\left\| \tilde{\Gamma}_1 - \tilde{\Gamma}_{\tilde{n}} \right\| = 1$. We apply the discrete Green's theorem [4, 9] on $\tilde{\Gamma}$ to efficiently approximate the region integrals.

We now focus on the implementation of the measures described in sections 4.1 and 4.2. One should be aware that detecting intersections and corresponding loops may be done on continuous sampled curves, by finding intersections between line segments [3, 5]. A part of the literature deals with this non-trivial task for self-intersection prevention and topology changes in active contours, as in [7]. However, self-intersections and self-tangencies are more conveniently studied on digital 4-connected curves. When such a curve crosses or overlaps itself, it necessarily contains several occurences of the same point, which is straightforward to detect.

Instead of a naive implementation of the self-tangency measure derived from Eq. (9), which would yield a $O(\tilde{n}^2)$ complexity, it is computed in a single pass by storing visited pixels in the discretized image domain $\tilde{\mathcal{D}}$ and detecting edges between adjacent pixels that were already taken. Simultaneously, we check that the curve does not contain points with multiplicity greater than 2. If such case happens, the curve is most likely an irrelevant candidate combination and its energy is set to $+\infty$. In this way, we make sure that the upcoming twisting term can be safely computed. This simple method is summarized in Algorithm 2.

As regards the twisting measure, its computation basically implies to detect positive and negative crossings. Ideally, a self-crossing involves a single point - put another way, two curve positions - as depicted in Fig. 5(a) and 5(b). For a positive (resp. negative) crossing, the curve arrives from the right (resp. left) and leaves to the left (resp. right) at the same point. However, due to the effect of discretization and the fact that the curves rarely crosses itself orthogonally, intersections with overlapping sections are more common, as depicted in Fig. 5(c) and 5(d). Since only areas of inverted loops should be considered to compute the twisting measure, regardless of the overlap created by these inverted loops, we focus solely on the events along the curve that helps to detect inverted single and double loops, as previously shown in Fig. 10. Let us consider that curve points are marked as long as the curve is traveled. Two basic events can be detected:

- **entrance**: the curve comes from an unmarked area and "enters" on a marked curve point
- **leaving**: the curve "leaves" a marked point and goes into an unmarked area.

The entrance is kept as the event meaning that a loop is created. It turns out that detecting *left-sided events*, i.e. the curve leaves to the left or enters from the left, is sufficient to detect simple and double inverted loops. At this stage, we safely assume that the curve contains only points of multiplicity ≤ 2.

Algorithm 2 Self-tangency measure

func selfTangency($\tilde{\Gamma}$) : $\mathbb{R}$
input
  $\tilde{\Gamma}$: digital 4-connected curve, sequence of $\tilde{n}$ points in $\mathbb{Z}^2$
variables
  $L \in \mathbb{R}$: length of self-tangent parts
  $M : \tilde{\mathcal{D}} \mapsto \mathbb{N}$: array of visited positions
  $D : \tilde{\mathcal{D}} \mapsto \{\text{TRUE}, \text{FALSE}\}$: array indicating double points

$L := 0$
for all $\boldsymbol{x} \in \tilde{\mathcal{D}}$ **do**
  $M[\boldsymbol{x}] := 0$
  $D[\boldsymbol{x}] :=$ FALSE
end for
for $i := 1$ to $\tilde{n}$ **do**
  if $M[\tilde{\Gamma}_i] \neq 0$ **then**
    if $D[\tilde{\Gamma}_i] =$ FALSE **then**
      $j := M[\tilde{\Gamma}_i]$
      $D[\tilde{\Gamma}_i] :=$ TRUE
      if $\tilde{\Gamma}_{j+1} = \tilde{\Gamma}_{i+1}$ or $\tilde{\Gamma}_{j-1} = \tilde{\Gamma}_{i+1}$ **then**
        $L := L + 1$
      end if
    else
      return $+\infty$
    end if
  end if
  $M[\tilde{\Gamma}_i] := i$
end for
return $\frac{L}{\tilde{n}}$

This property was previously checked during the computation of the self-tangency term in Algorithm 2. Suppose the curves enters a previously marked curve point from the left and denote the crossing by (i, j) such that $\tilde{\Gamma}_i$ is the "intersected" point (the position where the curve was visited first) and $\tilde{\Gamma}_j$ is the "intersecting" point (the current position). The curve is traveled again from index i until j is met again (in such case, the loop is simple) or a previous left-sided leaving is met (in such case, the loop is double). Whether an event is left-sided or not depends on the sign of the dot product between the approximated normal at $dcurve_i$ and tangent at $\tilde{\Gamma}_j$,

$$(\tilde{\Gamma}_j - \tilde{\Gamma}_{j-1}) \cdot (\tilde{\Gamma}_i - \tilde{\Gamma}_{i-1})^{\perp},$$

which should be positive for an entrance and negative for a leaving. The extraction of sets of single loops SL and double loops DL is detailed in Algorithm 3. The twisting measure is then computed by applying the discrete Green's theorem to Eq. (11).

B.3 Computation of the best combination of paths

The optimization procedure described in Section 5.2 is formalized in Algorithm 4. Variables $e_{\text{min-iter}}$ and e_{min} are respectively the current local minimum - the best combination found at the current iteration - and the current global minimum, i.e. the best configuration found since the beginning of the procedure. A special case arises when $e_{\text{min-iter}}$ is infinite at the end of the iteration, which happens when all tested curves have a negative area or an infinite simplicity term. In this case, the combination with the smallest area is selected as the starting sequence for the next iteration. This allows to

Algorithm 3 Extraction of single and double loops

func singleAndDoubleLoops($\tilde{\Gamma}$)
input
$\tilde{\Gamma}$: digital 4-connected path, sequence of $\tilde{n}$ points in $\mathbb{Z}^2$
output
SL: set of pairs of indices $\subset \mathbb{N}^2$
DL: set of double pairs of indices $\subset (\mathbb{N}^2)^2$
variables
onCurve $\in \{\text{TRUE}, \text{FALSE}\}$
loopFound $\in \{\text{TRUE}, \text{FALSE}\}$
$M : \tilde{\mathcal{D}} \mapsto \mathbb{N}$: array of visited positions
Le $\subset \mathbb{N}^2$: set of indices indicating left-sided events

onCurve := FALSE
SL := $\emptyset$, DL := $\emptyset$, Le := $\emptyset$
for all $\boldsymbol{x} \in \tilde{\mathcal{D}}$ **do**
$M[\boldsymbol{x}] := 0$
end for
for $j := 1$ to $\tilde{n}$ **do**
if onCurve = FALSE **then**
if $M[\tilde{\Gamma}_j] = 0$ **then**
$M[\tilde{\Gamma}_j] := j$
else
onCurve := TRUE ; $i := M[\tilde{\Gamma}_j]$
if $(\tilde{\Gamma}_j - \tilde{\Gamma}_{j-1}) \cdot (\tilde{\Gamma}_i - \tilde{\Gamma}_{i-1})^{\perp} > 0$ **then**
Le := Le $\cup \{(i, j)\}$
$k := i + 1$; loopFound := FALSE
while loopFound = FALSE **do**
if $k = j$ **then**
loopFound := TRUE
SL := SL $\cup \{(i, j)\}$
else
if $\exists l$ s.t$(k, l) \in$ Le **then**
loopFound := TRUE
DL := DL $\cup \{((i, j), (k, l)\}$
end if
end if
$k := k + 1$
end while
end if
end if
else
if $M[\tilde{\Gamma}_j] = 0$ **then**
$M[\tilde{\Gamma}_j] := j$; onCurve := FALSE ; $i := M[\tilde{\Gamma}_{j-1}]$
if $(\tilde{\Gamma}_j - \tilde{\Gamma}_{j-1}) \cdot (\tilde{\Gamma}_i - \tilde{\Gamma}_{i-1})^{\perp} < 0$ **then**
Le := Le $\cup \{(i, j)\}$
end if
end if
end if
end for

have the slowest dilation of the contour and hence reduces the risk of skipping combinations that could be relevant. With a view to conciseness, the energy of the current combination of selected admissible paths $E[\gamma_{1,x_1} \uplus \gamma_{2,x_2} \uplus \dots \gamma_{n,x_n}]$ is shortened to $E(x_1, \dots, x_n)$. Similarly, the inner area of this combination is shortened to $A(x_1, \dots, x_n)$.

Algorithm 4 Greedy search to minimize E over the set of combinations of admissible paths

$(x_1, \dots, x_n) := (1, \dots, 1)$
$e_{\text{min}} := E(x_1, \dots, x_n)$
$\mathcal{S}_{\text{min}} := \{x_1, ..., x_n\}$
while $\exists i, 1 \leq i \leq n,$ s.t. $x_i < K_i$ **do**
$e_{\text{min-iter}} := +\infty$
for $i := 1...n$ **do**
if $x_i < K_i$ **then**
$e := E(x_1, ..., x_i + 1, ..., x_n)$
if $e < e_{\text{min-iter}}$ **then**
$e_{\text{min-iter}} := e$
$\mathcal{S}_{\text{min-iter}} := (x_1, ..., x_i + 1, ..., x_n)$
end if
if $e < e_{\text{min}}$ **then**
$e_{\text{min}} := e$
$\mathcal{S}_{\text{min}} := (x_1, ..., x_i + 1, ..., x_n)$
end if
end if
end for
if $e_{\text{min-iter}} := +\infty$ **then**
$a_{\text{min-iter}} := +\infty$
for $i := 1...n$ **do**
if $x_i < K_i$ **then**
$a := A(x_1, ..., x_i + 1, ..., x_n)$
if $a < a_{\text{min-iter}}$ **then**
$a_{\text{min-iter}} := a$
$\mathcal{S}_{\text{area-min-iter}} := (x_1, ..., x_i + 1, ..., x_n)$
end if
end if
end for
$(x_1, ..., x_n) := \mathcal{S}_{\text{area-min-iter}}$
else
$(x_1, ..., x_n) := \mathcal{S}_{\text{min-iter}}$
end if
end while

References

1. V. Appia and A. Yezzi. Fully isotropic fast marching methods on cartesian grids. In *European Conference on Computer Vision*, volume 6311 of *LNCS*, pages 73–85. Springer-Verlag, 2010.
2. F. Benmansour and L. Cohen. Fast object segmentation by growing minimal paths from a single point on 2D or 3D images. *Journal of Mathematical Imaging and Vision*, 33(2):209–221, 2009.
3. J.L. Bentley and T.A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979.
4. S. Brlek, G. Labelle, and A. Lacasse. The discrete Green theorem and some applications in discrete geometry. *Theoretical Computer Science*, 346(2-3):200–225, 2005.
5. B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39(1):1–54, 1992.
6. Q. Lin. *Enhancement, extraction, and visualization of 3D volume data*. PhD thesis, Linkopings Universitet, 2003.
7. A. Nakhmani and A. Tannenbaum. Self-crossing detection and location for parametric active contours. *IEEE Transactions on Image Processing*, 21(7):3150–3156, 2012.
8. G. Peyré, M. Pechaud, R. Keriven, and L.D. Cohen. Geodesic methods in computer vision and graphics. *Foundations and Trends in Computer Graphics and Vision*, 5(3-4):197–397, 2010.
9. G.Y. Tang. A discrete version of Green's theorem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(3):242–249, 1982.

10. J.N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.