A graph-based approach to glacier flowline extraction: an application to glaciers in Switzerland

Nicolas Le Moine[a,*], Pierre-Stéphane Gsell[a]

[a]*Sorbonne Universités, UPMC Univ. Paris 06, CNRS, EPHE, UMR Metis*

Abstract

In this paper we propose a new, graph-based approach to glacier segmentation and flowline extraction. The method, which requires a set of glacier contours and a Digital Elevation Model (DEM), consists in finding an optimum branching that connects a set of vertices belonging to the topological skeleton of each glacier. First, the challenges associated with glacier flowline extraction are presented. Then, the three main steps of the method are described: the skeleton extraction and pruning algorithm, the definition and computation of a travel cost between all pairs of skeleton vertices, and the identification of the directed minimum spanning tree in the resulting directed graph. The method, which is mainly designed for valley glaciers, is applied to glaciers in Switzerland.

Keywords: glacier flowline, skeleton, discrete curve evolution, optimum branching, directed minimum spanning tree

1. Introduction

1.1. Glacier morphology

Glaciers are moving ice bodies which flow under their own weight, due to the accumulation of solid precipitations on the higher slopes of a mountain range. As the strain rate increases, ice viscosity decreases and the accumulated ice literally 'flows' downslope. Bahr and Peckham (1996) first explicitly drew a parallel between rivers and valley glaciers (i.e. glaciers that are confined by topography, as opposed to ice caps). They showed that glaciers also

*Corresponding author
Email address: nicolas.le_moine@upmc.fr (Nicolas Le Moine)

exhibit branching topologies, and computed classical river network indices such as bifurcation and area ratios for glacier networks. This analogy stems from the fact that in most recent orogens where valley glaciers are found (the Alps, the Andes, the Himalayas, the Rocky Mountains, etc.), glacier inception took place in a topography previously shaped by fluvial erosion (Gsell et al., 2014). Bahr and Peckham also pointed out that self-similarity properties could provide a 'lever arm' for tackling glacier flow dynamics for complex geometries, just as these properties are used for treating subgrid, hydraulic propagation in complex river networks with concepts such as the Geomorphological Instantaneous Unit Hydrograph (Rodríguez-Iturbe and Valdés, 1979; Gupta et al., 1980). One of the reasons why this approach has not been given much attention is maybe the difficulty lying in the first step of identifying networks of glacier flowlines.

1.2. Limits of classical drainage network extraction methods

Figure 1 shows the downstream region of the Rhone glacier in Switzerland. In fluvial morphology, we typically find cross-sections such as A–A' with a concave topography in the talweg. This translates into V- or U-shaped (for former glacial valleys) elevation contours, with the lowest point roughly in the medial axis of the talweg. Hence, river network extraction from a DEM is relatively straightforward, except for problems such as flat areas or closed depressions (see e.g. Garbrecht and Martz, 1997; Martz and Garbrecht, 1998).

Things are more complicated for ice-covered areas. In the accumulation (higher) area of the glacier, where hillslopes as well as valley floors are ice-covered, the topography is still concave (B–B'): the surface of the ice is more or less homothetic to the bedrock surface (with lower roughness though). In contrary, in the ablation area the glacier is limited to a narrow ice tongue confined between lateral, ice-free hillslopes. Since ice thickness is maximum in the medial axis of the ice tongue, we have a convex cross-section (C–C') with seemingly two talwegs on each side of the glacier. Elevation contours in this area have the shape of a W with its two wedges pointing upstream, as opposed to the single wedge in concave topography. The central flowline of the glacier (i.e. the line of maximum ice thickness), which is also typically the line of the bedrock talweg, is a local maximum and not a local minimum of the ice surface (it looks like is a local water divide). Therefore, it cannot be extracted in a stable way from a DEM with classical algorithms.

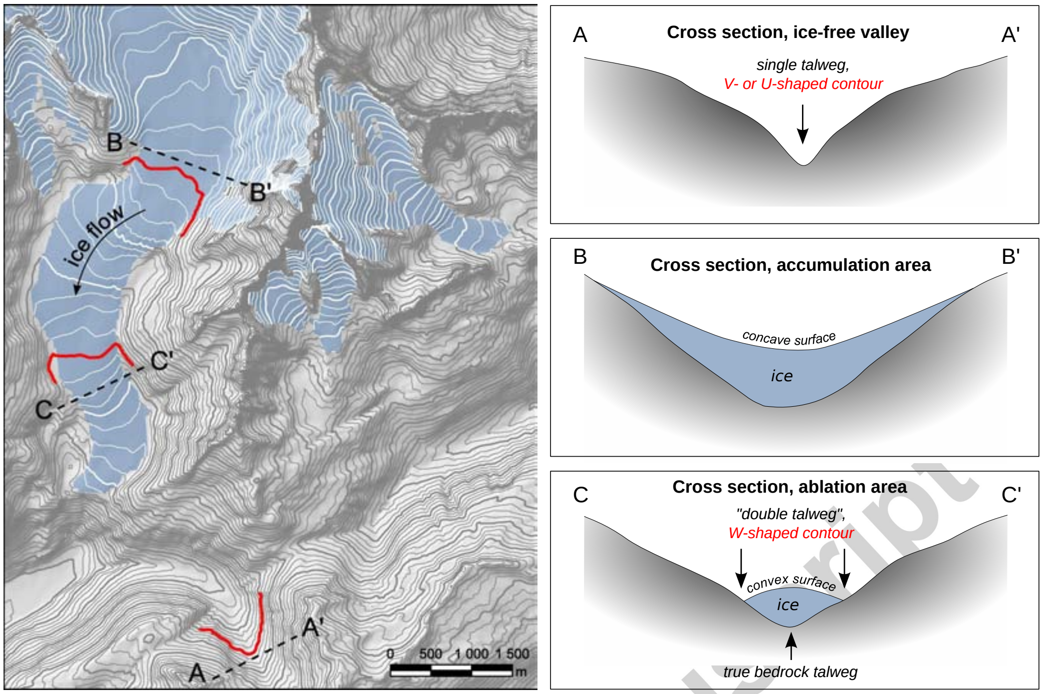


Figure 1: Illustration of the spurious 'double talweg' in glacial landscape. This feature mainly appears in the glacier's ablation area where a narrow ice tongue is confined in a valley (C–C'). On a map, elevation contours in this area have the shape of a W with its two wedges pointing upstream, whereas contours in classical (ice-free) valleys are V- or U-shaped with a single wedge pointing upstream (A–A').

1.3. Automatic methods for glacier flowline extraction

The problem of glacier flowline extraction has received some attention recently, due to the need of feeding glacier databases with attributes such as glacier length. A flowline or a set of flowlines has to essentially meet two requirements: (i) to stay as far as possible from the glacier boundary, and (ii) to cross elevation contours orthogonally. Le Bris and Paul (2013) propose to construct a set of waypoints located at the center of 'traverses' drawn perpendicular to a single, rectilinear 'main axis', and then connect them. However, the method can only extract one centerline per glacier. Kienholz et al. (2014) use a more complex approach based on a cost function which quantifies the trade-off between the two requirements; a set of flowlines is then extracted between glacier heads and a single snout (terminus) per glacier. Other methods apply alternate procedures in the accumulation and ablation zones (Machguth and Huss, 2014), also resulting in a large number of parameters.

1.4. Objectives of the study

In this paper, a new method is presented that aims at extracting glacier flowlines with an emphasis on preserving their tree-like structure, i.e. the structure of tributaries within the glacier.

As in Le Bris and Paul (2013), our method first identifies a set of waypoints (i.e., vertices of a graph) that are subsequently connected. However these waypoints are identified with a more general operation called skeletonization. Once these waypoints are identified (including special 'snout' vertices), we compute a travel cost between every pair of them: the cost function is designed so as to penalize displacements that deviate from the steepest slope direction. The main difference with Kienholz et al. (2014) is the formulation of an anisotropic cost function. The final step is to construct a directed minimum spanning tree (DMST) that allows to visit all waypoints at minimal cost, starting from a snout (root) vertex. Edges of this DMST meet the two requirements: they stay 'far' from the glacier's boundary (since they link waypoints belonging to the skeleton), and they deviate little from the steepest slope direction since they are least-cost paths with respect to the slope-dependent cost function. The overall procedure requires only 5 parameters, in contrast with other methods (e.g. 16 parameters in Kienholz et al., 2014 and 17 in Machguth and Huss, 2014).

The method is mainly designed for valley glaciers, as ice caps usually do not exhibit strong branching topologies. It is tested on a dataset of Swiss glaciers (Figure 2a), covering a total area of 1200 km^2 and mainly located in the headwaters of the Rhone, Rhine, and Danube rivers. The steps of the method are illustrated with a focus on a particular glacier complex in the Bernese Alps (Figure 2b), straddling the water divide between the Rhone and Rhine rivers.

2. Data

2.1. Digital Elevation Model

In this study we use the 25-meter Digital Elevation Model from the Swiss Federal Office of Topography (SwissTopo, 2004).

2.2. Randolph Glacier Inventory (RGI) glacier contours

Glacier outlines are taken from the Randolph Glacier Inventory (RGI, Arendt et al., 2012). The RGI provides a segmentation of glacier complexes (continuous ice bodies) into individual glaciers; we chose to dissolve (re-aggregate) these elements and to work with the complexes in order test the ability of our approach to identify multiple snouts in such complexes. The segmentation is not a prerequisite and is even a by-product of our method.

3. Skeleton extraction and pruning

3.1. Glacier flowlines as a topological skeleton

The method proposed by Le Bris et al. (2013), which consists in picking the midpoints of 'traverses' drawn orthogonally to the glacier's 'main axis', is actually related to a topological operation called skeletonization, or medial-axis transform. A first, intuitive definition is the analogy with a grassfire (Blum, 1967): if one 'sets fire' simultaneously at all points on the border of a grass field enclosed in the object's boundary, then the skeleton is the set of points where two or more firefronts meet (see Figure 3a). The result (Figure 3b) is a 'thinned' version of the object (i.e., a set of edges, namely a graph) that preserves its essential geometrical and topological features (in the example of Figure 3, the skeleton edges look like the veins of the leaf). We will see that this is a first interesting step to glacier flowline extraction.

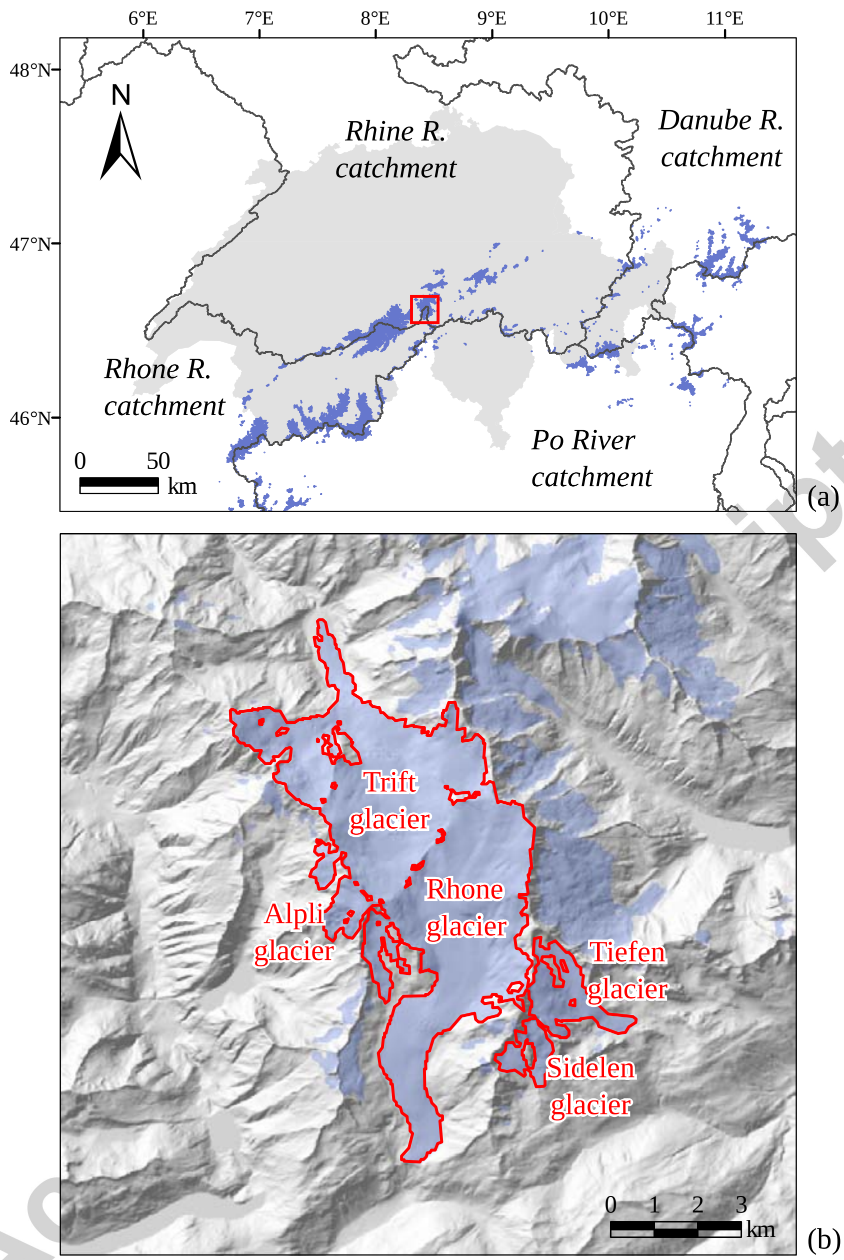
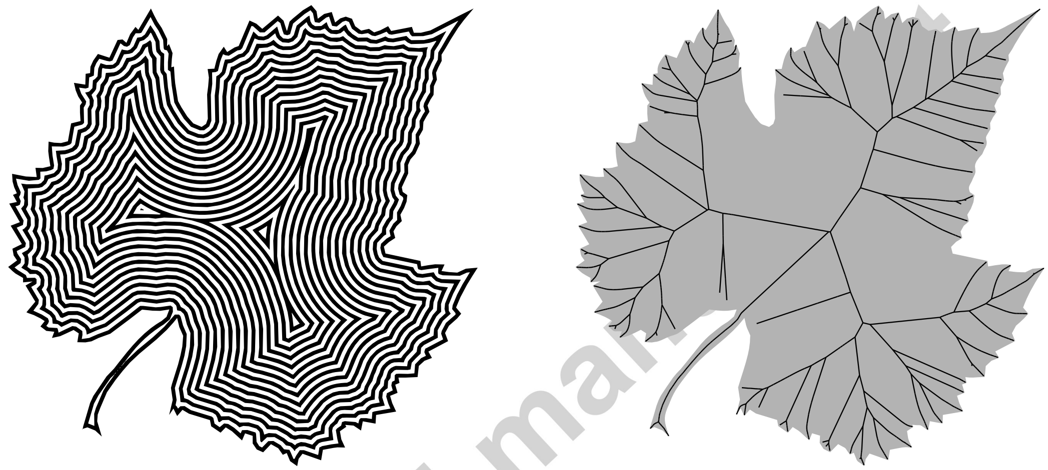


Figure 2: Area covered by this study. (a) Map of Switzerland with glacier contours in dark blue. (b) Zoom on the Rhone-Trift glacier complex illustrating the steps throughout the paper. This complex covers 35.8 km^2 and includes two large glaciers: the Rhone glacier (15.9 km^2) and the Trift glacier (16.6 km^2) as well as a number of smaller ones (e.g. Tiefen, Sidelen and Alpli glaciers).



(a) (b)

Figure 3: Grassfire analogy of the skeletonization: (a) propagation of the 'firefronts'; (b) resulting skeleton i.e. the set of points where two or more fronts meet.

3.2. Skeletonization using Voronoi tesselation

A second definition of the skeleton uses the concept of maximal disk (or maximal ball in dimensions higher than 2). A disk $\mathcal{B}$ is said to be maximal in set $\mathcal{D}$ if it is completely included in $\mathcal{D}$, and such that if it is contained in any other disc $\mathcal{B}'$ then $\mathcal{B}'$ is not completely included in $\mathcal{D}$. Mathematically,

$$\mathcal{B} \text{ is a maximal disk in set } \mathcal{D} \text{ iif } \begin{cases} \mathcal{B} \subseteq \mathcal{D} \\ \mathcal{B} \subset \mathcal{B}' \Rightarrow \mathcal{B}' \nsubseteq \mathcal{D} \end{cases} \tag{1}$$

A maximal disk $\mathcal{B}(\mathbf{s})$ centered at point $\mathbf{s}$ is entirely contained in $\mathcal{D}$ and is interiorly tangent to the boundary $\partial\mathcal{D}$ in at least two different points, called generating points: these are two locations where firefronts originate from in the grassfire analogy, and they meet at the center of a maximal disk. The skeleton $S(\mathcal{D})$ can be defined as the set of the centers of all maximal disks in $\mathcal{D}$:

$$S(\mathcal{D}) = \left\{ \mathbf{s} \quad / \quad \mathcal{B}(\mathbf{s}) \text{ is a maximal disk in } \mathcal{D} \right\} \tag{2}$$

Figure 4 illustrates this definition. $Gen(\mathbf{s}) \subset \partial\mathcal{D}$ denotes the generating points of the skeletal point $\mathbf{s} \in S(\mathcal{D})$.

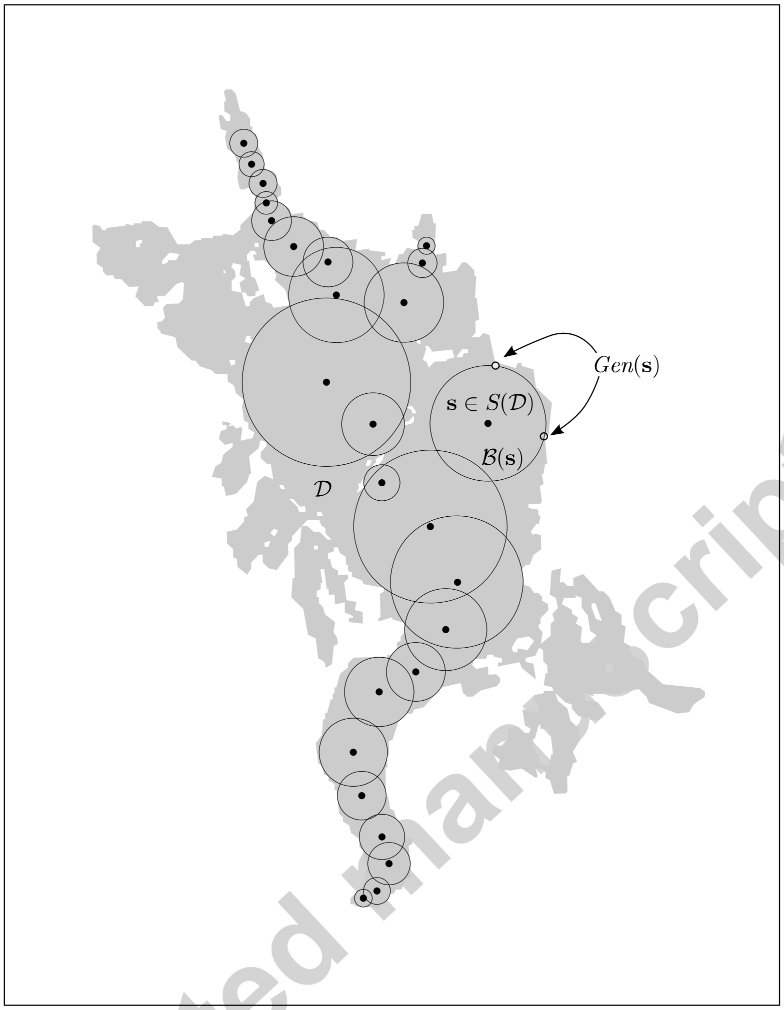


Figure 4: Definition of skeleton $S(\mathcal{D})$ as the set of centers of all maximal disks in $\mathcal{D}$ (gray shape). $Gen(\mathbf{s})$ denotes the generating points of the skeletal point $\mathbf{s}$.

One possibility to generate the skeleton is to implement a 'grassfire' algorithm. Given the previous definition, another method uses the Voronoi tesselation (see e.g. Brandt and Algazi, 1992) of a set of points (input sites) sampled along the boundary (see Figure 5). Each Voronoi region represents the area closest to a boundary input site, and is delimited by several edges: hence, each edge $\mathcal{E}$ is a segment of the perpendicular bisector of a *pair* of input sites. To show the similarity with the previous results, we call this pair the generating pair $Gen(\mathcal{E})$ of the edge. As the number n of input sites sampled on the boundary increases ($n \to \infty$), the set of edges of the tesselation that are contained in the domain $\mathcal{D}$ converges to the exact skeleton.

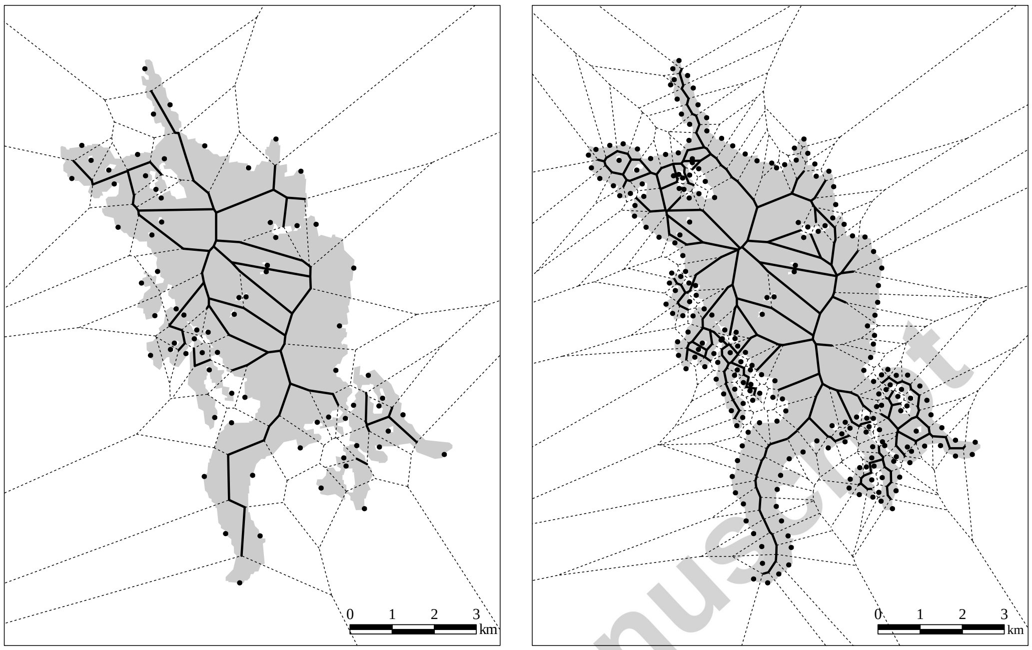


Figure 5: Skeletonization using the Voronoi tesselation of a set of boundary input sites. The set of internal edges of the tesselation (solid black lines) converges to the skeleton as the number of points on the boundary increases (left: one point every 2000 m ; right: one point every 500 m).

We used the Qhull library (Barber et al., 1996) to generate the tesselation. It allows to retrieve not only the Voronoi edges but also the pair of generating input sites (boundary points) for each edge, a highly valuable information for the subsequent steps. It is worth noting that the object may have inner boundaries ('holes' in the shape, such as inner rocks for a glacier), which will translate into cycles in the skeletal graph.

RGI glacier contours were first densified so as to have at least one point every 50 m along the boundary, before running the tesselation with Qvoronoi. All subsequent steps were then performed under Scilab (Scilab Entreprises, 2012), using the CLaRiNet library (Le Moine, 2013).

3.3. Skeleton pruning through Discrete Curve Evolution

As mentioned previously, the set of edges obtained with the Voronoi tesselation is only an approximation of the skeleton. Moreover, due to the noise in the boundaries, unsignificant boundary features can generate skeleton edges which do not reflect essential topological properties. Hence, we need to simplify, or 'prune', the skeleton i.e. remove many spurious edges.

In this study we use the pruning algorithm of Bai et al. (2007), who point out that not all n boundary points significantly contribute to the shape of the object: only a relatively small subset of these boundary (input) sites may be sufficient to describe the overall shape. Hence, if we select $k \leq n$ points on the boundary, we define a partition of the contour into k contour segments (subarcs). The pruning rule is to remove all skeleton edges whose generating points lie on a same contour segment, as illustrated in Figure 6. It is important to note that this pruning process is not equivalent to reducing the number of input sites in the Voronoi tesselation (i.e. moving from right to left in Figure 5): the pruning removes some edges but the geometry of the remaining ones is not altered.

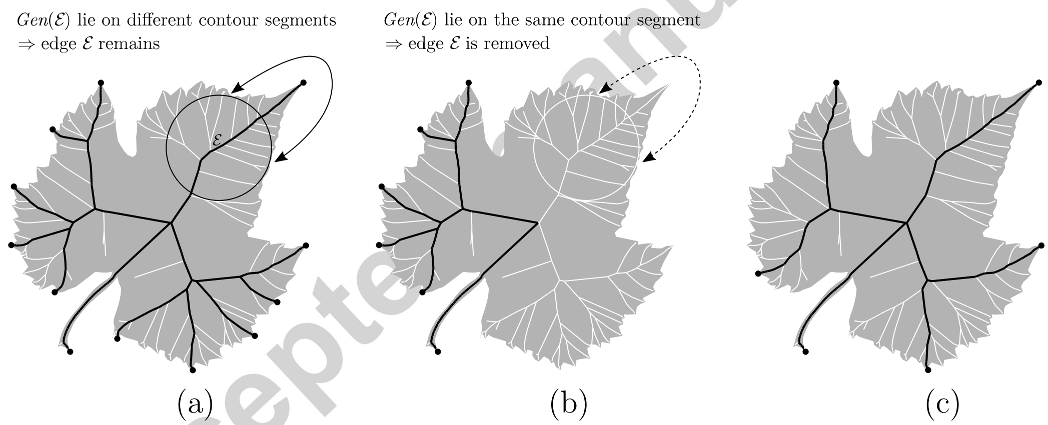


Figure 6: Pruning of the skeleton based on contour partitioning. Skeleton edges which have their generating points lying on a same contour segment are removed (in white). (a) Partition with $k = 12$ points; (b) $k = 6$ points; (c) $k = 6$ points but at different locations.

The problem is to select the k most relevant points; Bai et al. (2007) propose to reduce the number of boundary points from n to $k < n$ through Discrete Curve Evolution (DCE).

Let $\mathbf{c}_1, \ldots, \mathbf{c}_n$ be the n input sites sampled on the boundary, and let $\mathbf{u}_{i-1,i} = \mathbf{c}_i - \mathbf{c}_{i-1}$ and $\mathbf{u}_{i,i+1} = \mathbf{c}_{i+1} - \mathbf{c}_i$ be the vectors of the boundary edges incident to point $\mathbf{c}_i$ (Figure 7). The contribution of this site to the overall shape is (Latecki and Lakämper, 2002):

$$K(\mathbf{c}_i) \quad = \quad \frac{\theta_i \ \|\mathbf{u}_{i-1,i}\| \ \|\mathbf{u}_{i,i+1}\|}{\|\mathbf{u}_{i-1,i}\| + \|\mathbf{u}_{i,i+1}\|} \tag{3}$$

where

$$\theta_i = (\widehat{\mathbf{u}_{i-1,i}\,,\,\mathbf{u}_{i,i+1}}) = \arccos\left(\frac{\mathbf{u}_{i-1,i} \ \cdot \ \mathbf{u}_{i,i+1}}{\|\mathbf{u}_{i-1,i}\| \ \|\mathbf{u}_{i,i+1}\|}\right) \operatorname{sgn}\big(\det(\mathbf{u}_{i-1,i}\,,\,\mathbf{u}_{i,i+1})\big)$$

is the oriented turn angle at point $\mathbf{c}_i$ (trigonometric, i.e. measured counterclockwise). This contribution is usually defined in 2D (planar computation), but there is no difficulty in extending it to a 3D curve. However, even if glaciers are located on steep topography, they remain objects in the geographical space i.e. with a rather flat aspect ratio, so that the addition of the vectors' z-coordinate would not change the contributions dramatically.

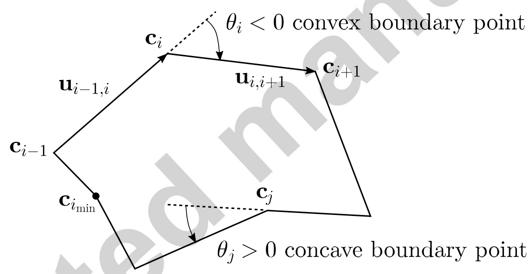


Figure 7: Definition of the turn angle at a boundary point.

Since contours are closed, we set $\mathbf{c}_{n+1} = \mathbf{c}_1$ and $\mathbf{c}_0 = \mathbf{c}_n$. If boundary points are sorted clockwise (i.e. with the 'inside' on the right and the 'outside' on the left when moving along the boundary, as in the ESRI® shapefile format), then:

- K is negative for a convex point (clockwise turn), and positive for a concave point (counterclockwise turn),
- the higher the absolute value $|K|$, the higher the contribution to the overall shape (through great segment lengths and/or large turn angle).

> A point lying on a straight contour portion, i.e. such that $\theta = 0$, will have a zero contribution (i.e. it can be removed without any loss of shape information).

We start the DCE with the n initial input sites and we removes the point $\mathbf{c}_{i_{\min}}$ having the lowest absolute value $|K(\mathbf{c}_{i_{\min}})|$, thus leading to a simplified contour called DCE level $n-1$. The metric K is again computed for the $n-1$ remaining points, and the procedure is repeated p times to obtain a hierarchical set of contour partitions, called DCE level $n-2, n-3, \ldots, k = n-p$. At each iteration, we remove the skeleton edges whose generating points lie on a same contour segment of DCE level $k = n - p$, and these edges are assigned the level k. Since we need at least two points on the contour to define a partition, the highest possible level is $k = 2$. Figure 6c actually represents DCE level 6 of the leaf, and in Figure 8 we show the final hierarchy of skeleton edges.

If the shape has one or several inner boundaries, we simply add a loop on the boundaries in order to find the least-contributing site at each iteration.

Figure 8: Hierarchy of skeleton edges obtained at the end of the pruning algorithm: we can stop at any level in this hierarchy.

4. Identification of snout vertices

In the previous section we illustrated the skeleton extraction and pruning with a simple shape (a leaf); we now apply these methods to RGI contours of Swiss glaciers. The main issues will be to decide the level k at which we should extract the skeleton for each glacier, and to use elevation data in order to identify snout and head vertices.

4.1. Choice of DCE level for each glacier

Consider a glacier with area A. Clearly, the larger the glacier, the more points will be needed on its boundary to correctly describe its shape or skeleton, i.e. the higher we will need to stay in the DCE hierarchy. Conversely, a small glacier with a single main axis without tributary could be correctly described at DCE level $k = 2$. A selection rule of thumb $k = f(A)$ was devised empirically upon visual appreciation:

$$(k-2) = \lfloor k_0 A^{\gamma} \rfloor$$

where $\lfloor\ \rfloor$ is the floor function. We used $k_0 = 13.5$ and $\gamma = 0.8$ with A in km^2. For example, the skeleton of Rhone-Trift glacier complex, with an area $A = 35.8$ km^2, is still correctly described at DCE level $k = 2 + 13.5(35.8)^{0.8} = 238$, i.e. with 238 sites on its boundary (Figure 9, left panel).

4.2. Hierarchical snout identification

The next step is the orientation of the skeletal edges in order to identify glacier snouts and heads. Indeed, skeletonization is a planar operation and we do not know if a 'leaf' vertex in the skeleton corresponds to the beginning of a flowline ('head' vertex), or to its end ('snout' vertex). Again, we will use the results of the DCE algorithm to create a hierarchy of potential snouts. A skeletal vertex $\mathbf{s}$ is said to be a level-k snout if it meets the three following requirements:

(i) $\mathbf{s}$ is a leaf vertex of the skeletal graph of DCE level k, i.e. having a degree (number of incident edges) $d = 1$,

(ii) $\mathbf{s}$ is at an elevation lower than its closest vertex $\mathbf{s}'$ of degree $d \neq 2$ in the level k skeleton graph. $\mathbf{s}'$ is such that all vertices on the path between $\mathbf{s}$ and $\mathbf{s}'$ have degree 2: in the following we will call such a sequence of vertices of degree 2 (except the two extremal vertices) a *stretch*.

(iii) the generating points $Gen(\mathcal{E}_{\mathbf{s}})$ of its unique incident edge $\mathcal{E}_{\mathbf{s}}$ (pendant edge) lie on either side of an input site which is a local topographic minimum in the DCE level k.

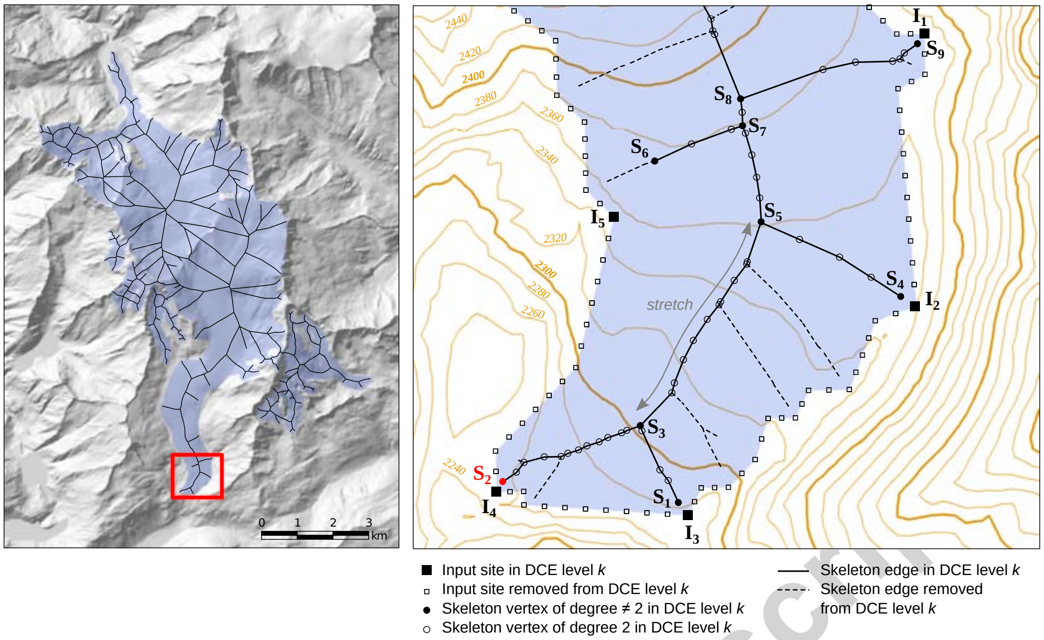


Figure 9: Definition of a snout vertex in the skeleton pruned at DCE level k ($k = 238$ here).

These seemingly complicated requirements translate the more intuitive notion that an edge is at a snout if it points downslope in a locally convex portion of the boundary that also points downslope, as shown in Figure 9.

On Figure 9, skeleton edges at level k are drawn in solid black lines, while edges that have been pruned at earlier levels are drawn with dotted lines. There are several 'candidate' snouts in this area: skeleton vertices $\mathbf{S_1}$, $\mathbf{S_2}$, $\mathbf{S_4}$, $\mathbf{S_6}$ and $\mathbf{S_9}$, which are all leaf vertices at the current pruning level, i.e. meeting requirement (i). All these 5 vertices also satisfy requirement (ii), since we have:

$$\begin{array}{rcl} \text{Leaf vertex} & & \text{Nearest vertex of degree } \neq 2 \\ z(\mathbf{S_1}) & < & z(\mathbf{S_3}) \\ z(\mathbf{S_2}) & < & z(\mathbf{S_3}) \\ z(\mathbf{S_4}) & < & z(\mathbf{S_5}) \\ z(\mathbf{S_6}) & < & z(\mathbf{S_7}) \\ z(\mathbf{S_9}) & < & z(\mathbf{S_8}) \end{array}$$

However, Table 1 shows that only one candidate satisfies the third requirement:

Leaf vertex	Input site in DCE level k that separates the vertex's generating pair	Is requirement (iii) met ?
$\mathbf{S_1}$	$\mathbf{I_3}$	NO: $z(\mathbf{I_2}) > z(\mathbf{I_3}) > z(\mathbf{I_4})$
$\mathbf{S_2}$	$\mathbf{I_4}$	YES: $z(\mathbf{I_3}) > z(\mathbf{I_4}) < z(\mathbf{I_5})$
$\mathbf{S_4}$	$\mathbf{I_2}$	NO: $z(\mathbf{I_1}) > z(\mathbf{I_2}) > z(\mathbf{I_3})$
$\mathbf{S_6}$	$\mathbf{I_5}$	NO
$\mathbf{S_9}$	$\mathbf{I_1}$	NO

Table 1: Check for requirement (iii) in the definition of snout vertices.

Hence, the only snout at level k in this part of the glacier is vertex $\mathbf{S_2}$. As input sites are iteratively removed from the glacier's outline in the DCE process, the combination of requirements allows a robust identification of convex portions of the boundary that point downslope, i.e. glacier tongues/snouts. Finally, skeleton stretches which satisfy requirements (i) and (ii) but not requirement (iii) are further removed.

Hence, we obtain a snout hierarchy tied to the edge hierarchy: with a single threshold (the level $k = f(A)$ of the DCE), we can extract both a set of skeletal edges and a set of snouts for a given continuous ice body of area A.

4.3. Definition of skeleton waypoints

Let us have a closer look at the final pruned skeleton in Figure 10. The solid black lines are the remaining edges while the black dots are just a subset of the vertices in the pruned skeleton that we downsampled for simplicity (these vertices are *waypoints* distant from at least 1000 m). Clearly the skeleton edges are far from a set of flowlines, except maybe in the ablation area. The location of glacier heads and snouts looks satisfying but major flaws appear:

- many edges in the skeleton significantly deviate from the local steepest slope direction, i.e. do not at all cross elevation contours at a right angle,
- the skeleton has cycles (around inner rocks).

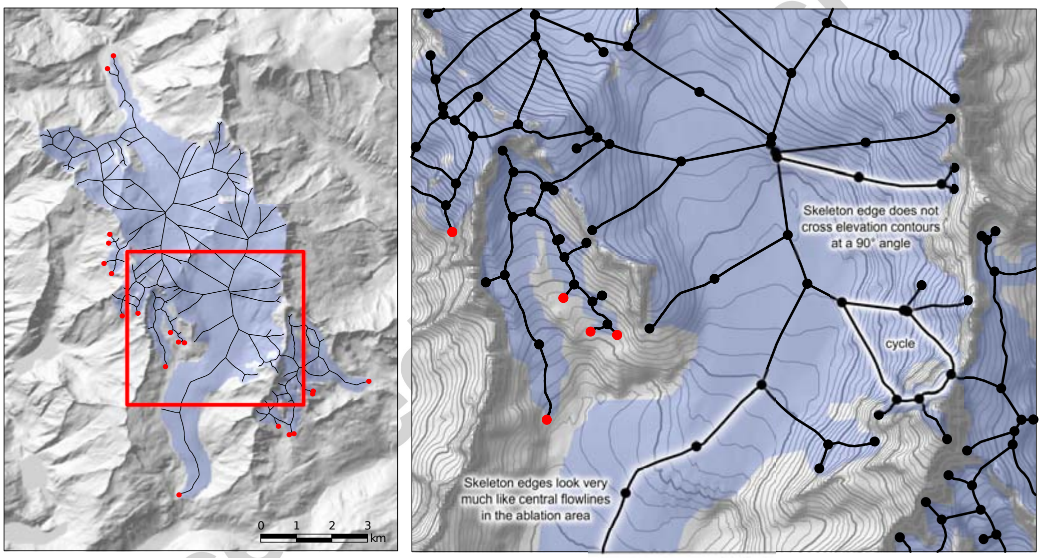


Figure 10: Example of flaws in the pruned skeletal graph, notably the deviation of skeleton edges from steepest slope directions and the presence of cycles. Black dots are a downsampled subset of skeleton vertices (waypoints), and red dots are snout vertices.

We will see that we can build another graph which links those same skeleton vertices/waypoints, but which do not exhibit these flaws, provided we can *quantify* what a 'significant deviation from the local steepest slope direction' is.

5. Construction of optimum branchings

In this section we define a *cost function* that will allow us to rate how much an edge between two vertices deviates from a slope line, and to select a suitable set of edges.

5.1. Rationale

Douglas (1994) recalls that computing the least-cost path (with respect to various factors such as slope, soil cover, etc.) between a point A and a point B involves three steps:

1. the definition of a *cost of movement* for an elementary movement from A to A' (close to A);
2. the computation of the *accumulated cost surface* spreading from the starting point, that sums the costs of all elementary movements;
3. the construction of the least-cost path as the *slope line* of the accumulated cost surface starting at B and ending at A (Warntz, 1957).

If we want a least-cost path to look like a steepest ascent or descent line, we can figure out what the corresponding cost function has to look like by reversing the three steps:

3. The least-cost path has to look like a topographical slope line (it will never be a true one though, unless B is strictly upslope or downslope from A)
2. This implies that the *accumulated cost surface* has to 'look like' the topographical surface z around A;
1. It means that the local *cost of movement* has to 'look like' a *differential* of the topographical surface, i.e. depend on its gradient $\boldsymbol{\nabla} z$.

The next section presents a cost function designed according to this rationale.

5.2. Definition of the cost function

Consider an elementary displacement $\boldsymbol{\delta}\mathbf{x} = (dx, dy)$ from $\mathbf{x} = (x, y)$ to $\mathbf{x}' = (x', y') = \mathbf{x} + \boldsymbol{\delta}\mathbf{x} = (x + dx, y + dy)$. The cost of this elementary displacement will be defined as:

$$\begin{aligned} C_{\uparrow}(\mathbf{x}\,,\,\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) &= \big[z_{\max} - z(\mathbf{x})\big] \quad - \quad \big[z(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) - z(\mathbf{x})\big] \quad + \quad \lambda\|\boldsymbol{\delta}\mathbf{x}\| \\ &= \underbrace{\|\boldsymbol{\nabla} z\|\|\boldsymbol{\delta}\mathbf{x}\|}_{(1)} \quad - \quad \underbrace{(\boldsymbol{\nabla} z)\cdot\boldsymbol{\delta}\mathbf{x}}_{(2)} \quad + \quad \underbrace{\lambda\|\boldsymbol{\delta}\mathbf{x}\|}_{(3)} \end{aligned}$$

where $\|\boldsymbol{\delta}\mathbf{x}\| = \sqrt{dx^2 + dy^2}$ is the length of the displacement, $\boldsymbol{\nabla} z$ is the gradient vector of the topographic surface at point $\mathbf{x}$, and λ is a friction factor (scalar) in $\mathrm{m}\cdot\mathrm{m}^{-1}$. This cost is expressed in meters of potential energy and is the sum of three terms:

(1) is the maximum elevation (relative to $z(\mathbf{x})$) we could reach with a displacement of length $\|\boldsymbol{\delta}\mathbf{x}\|$ starting at $\mathbf{x}$. By definition of the gradient, this maximum elevation is $z_{\max} = \|\boldsymbol{\nabla} z\|\|\boldsymbol{\delta}\mathbf{x}\|$, which is always positive.

(2) is the elevation we actually reached with the displacement $\boldsymbol{\delta}\mathbf{x}$, which is $z' = (\boldsymbol{\nabla} z)\cdot\boldsymbol{\delta}\mathbf{x}$ (note that z' is algebraic and may be negative). Consequently, the difference $(1)-(2)$, which is always positive whatever the displacement, is a measure of *how much higher* we could have gotten with a displacement of the same length. If the displacement is in the direction of the gradient (i.e. upslope) then $\boldsymbol{\nabla} z$ and $\boldsymbol{\delta}\mathbf{x}$ are colinear and $(\boldsymbol{\nabla} z)\cdot\boldsymbol{\delta}\mathbf{x} = \|\boldsymbol{\nabla} z\|\|\boldsymbol{\delta}\mathbf{x}\|$. Hence (1) and (2) cancel out in the case of a steepest *ascent*: the cost will be low. Conversely, if we move downslope, $(\boldsymbol{\nabla} z)\cdot\boldsymbol{\delta}\mathbf{x}$ is negative and $(1)-(2)$ is largely positive: the displacement is very costly.

(3) is a friction term stabilizing the cost function: whatever the direction (upslope, downslope or along an elevation contour), there is always a cost λ for moving 1 meter across the surface. The effect is to smooth trajectories and to prevent paths from following too closely the small-scale irregularities on the surface.

We call $C_{\uparrow}$ the *upslope* cost function, which is obviously not symmetric:

$$C_{\uparrow}(\mathbf{x},\mathbf{x}') \neq C_{\uparrow}(\mathbf{x}',\mathbf{x})$$

The definition of $C_{\uparrow}$ looks unusual, as classical cost functions (see e.g. Kienholz *et al.*, 2014) are simply the product of a scalar impedance $I(\mathbf{x})$ by the length of the elementary displacement: $C(\mathbf{x}\,,\,\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) = I(\mathbf{x})\,\|\boldsymbol{\delta}\mathbf{x}\|$.

Hence, every displacement of length $\|\boldsymbol{\delta}\mathbf{x}\|$ around $\mathbf{x}$ has the same cost whatever the direction (C is necessarily isotropic). This kind of formulation is easily solved in classical GIS softwares (it only requires to specify the impedance raster, and the start and end points). However, we believe that our formulation is preferable for steepest ascent/descent problems, which are anisotropic in nature (see also Collischonn and Pilar, 2000).

Similarly, we can define a *downslope* cost function:

$$C_{\downarrow}(\mathbf{x}\,,\,\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) \quad = \quad \underbrace{+(\boldsymbol{\nabla} z)\cdot\boldsymbol{\delta}\mathbf{x}}_{(1)} \quad - \quad \underbrace{\big(-\|\boldsymbol{\nabla} z\|\|\boldsymbol{\delta}\mathbf{x}\|\big)}_{(2)} \quad + \quad \underbrace{\lambda\|\boldsymbol{\delta}\mathbf{x}\|}_{(3)}$$

This time, (2) is the *minimum* elevation we could reach with a displacement of length $\|\boldsymbol{\delta}\mathbf{x}\|$. The difference (1) − (2) is always positive and is a measure of *how much lower* we could have gotten with a displacement of the same length: (1) and (2) cancel out in the case of a steepest *descent*. Conversely, if we move upslope, (1) − (2) is large and the cost is high.

Finally, these cost functions satisfy the requirement that the steepest ascent path from $\mathbf{x}$ to $\mathbf{x}'$ is also the steepest descent path from $\mathbf{x}'$ to $\mathbf{x}$ (assuming the same friction factor λ):

$$C_{\uparrow}(\mathbf{x},\mathbf{x}') = C_{\downarrow}(\mathbf{x}',\mathbf{x})$$

In the following, we only use the upslope cost function $C_{\uparrow}$; we will see why it is more convenient to treat the problem from downslope up. Moreover, in order to improve flow convergence near the glacier boundary, we slightly modify the cost function by increasing the friction factor λ near the boundary (this will penalize flowlines wandering along the boundary):

$$C_{\uparrow}(\mathbf{x}\,,\,\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) = \|\boldsymbol{\nabla} z\|\|\boldsymbol{\delta}\mathbf{x}\| - (\boldsymbol{\nabla} z)\cdot\boldsymbol{\delta}\mathbf{x} + \underbrace{\left(\lambda_{\infty} + (\lambda_0 - \lambda_{\infty})e^{-\frac{D(\mathbf{x})}{D_{\lambda}}}\right)}_{\lambda(\mathbf{x})}\|\boldsymbol{\delta}\mathbf{x}\|$$

where $D(\mathbf{x})$ is the distance to the boundary at point $\mathbf{x}$, λ_{∞} is the friction cost far from the boundary, $\lambda_0 > \lambda_{\infty}$ is the friction cost at the boundary and D_{λ} is the scale of decrease with distance. Note that this modified cost is still anisotropic: $\lambda(\mathbf{x})$ is an impedance, but it is not the dominant term of the cost function. The values $\lambda_\infty = 0.035\ \mathrm{m}\cdot\mathrm{m}^{-1}$, $\lambda_0 = 5\lambda_\infty = 0.175\ \mathrm{m}\cdot\mathrm{m}^{-1}$, and $D_\lambda = 150$ m were found to give very good results on all glaciers.

Finally, since we use a raster DEM (i.e. a square lattice with only 8 possible elementary displacements from a given grid point $\mathbf{x}_{i,j}$), we define a discrete version of $C_\uparrow$:

$$C_\uparrow(\mathbf{x}_{i,j}\,,\mathbf{x}_{i+\delta i,j+\delta j}) = \left(z_{i,j} + s_{\max}\sqrt{\delta i^2+\delta j^2}\right) - z_{i+\delta i,j+\delta j} + \lambda(\mathbf{x}_{i,j})\sqrt{\delta i^2+\delta j^2}$$

where

$$s_{\max} = \max_{0<(\delta i^2+\delta j^2)\leq 2}\left\{\frac{z_{i+\delta i,j+\delta j} - z_{i,j}}{\sqrt{\delta i^2+\delta j^2}}\right\}$$

is the estimated upward slope (norm of the gradient) at pixel $\mathbf{x}_{i,j}$. If the above expression for $s_{\max}$ is negative, it is of course set to zero: there is a local maximum at $\mathbf{x}_{i,j}$.

5.3. Cost assignment for paths between waypoints

We use Dijkstra's algorithm (Dijkstra, 1959) to compute the least-cost path between each pair of waypoints. It uses the elementary cost function to produce an accumulated cost surface spreading around a start point **S**, and a backlink grid which gives the direction *opposite* to the gradient of the accumulated cost surface at each pixel. The least-cost path from **S** to any point is constructed in the reverse direction, starting from the destination and following the backlinks to **S**. In our case, the cost is defined from downslope up, so a backlink is defined from upslope down (see Figure 11).

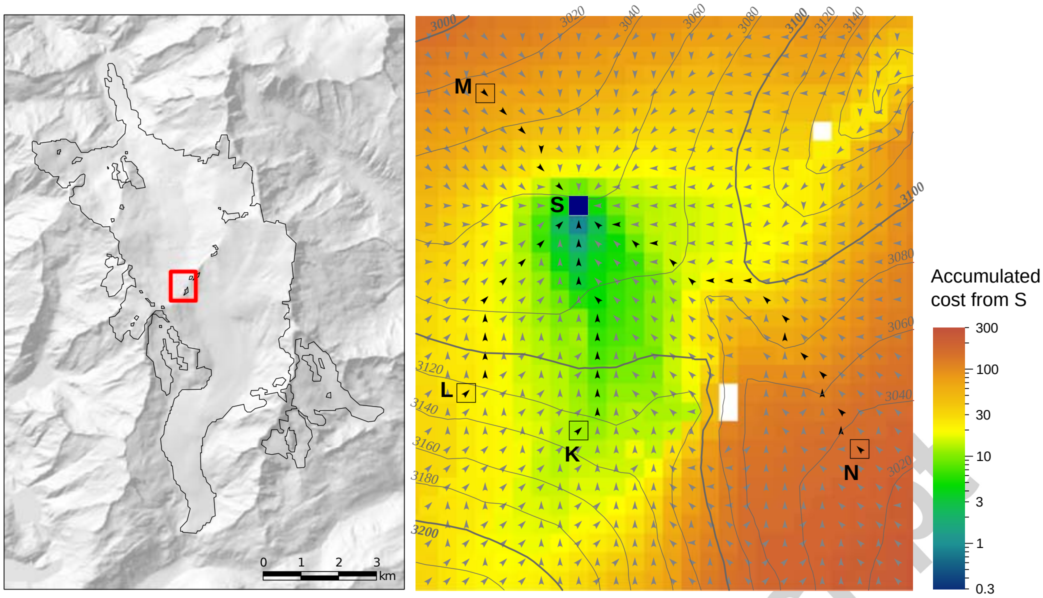


Figure 11: Example of accumulated cost surface, spreading from a start point **S**. Arrows indicate the backlinks: the least-cost path from **S** to any destination (**K**, **L**, **M**, **N**, etc.) is constructed from the destination to the start **S**, following the backlink grid. White pixels are ice-free zones (inner rocks) across which no displacement is possible.

Figure 11 illustrates the effects of the definition of $C_{\uparrow}$:

- destination **K** is located right upslope from start **S**, at an elevation of about 3130 m. Since edge (**S**,**K**) is almost a steepest ascent path, its cost is low: $C_{\uparrow}(\mathbf{S}, \mathbf{K}) = 11.2$
- destination **L** is located at about the same elevation, but one has to cross elevation contours at some angle to get there from **S**. The cost is higher: $C_{\uparrow}(\mathbf{S}, \mathbf{L}) = 25.3$
- destination **M** is located at an elevation lower than **S**, about 3030 m. Consequently, the edge from **S** to **M** is far from a steepest *ascent* (it is almost a steepest *descent*), and has a high cost: $C_{\uparrow}(\mathbf{S}, \mathbf{M}) = 80.5$ (note that this path would in contrary have a very low cost with respect to the downslope cost function $C_{\downarrow}$)
- finally, destination **N** is located on the other side of a ridge. In order to reach **N** from **S**, one must first climb to a pass at about 3100 m (low cost), but then go down (high cost). We have $C_{\uparrow}(\mathbf{S}, \mathbf{N}) = 169.3$

We use this algorithm to compute the costs between any pair of skeletal vertices/waypoints. In practice, we only compute the costs to the n nearest neighbors of each vertex with respect to the cost function, using a local run of Dijkstra's algorithm that is aborted as soon as n neighbor vertices have been visited (straightforward with an implementation based on a priority queue). The value $n = 30$ turned out to be largely sufficient for the following steps. We obtain the kind of graph shown in Figure 12. Each edge can be traversed in both directions, but at a different cost in each direction: the graph is *directed*.

5.4. Identification of minimum spanning branchings

The creation of a dense, redundant graph between the skeletal waypoints looks like a step backwards, compared to the simplicity of the skeleton. However, graph theory provides powerful tools which will help simplify this graph again, in the way we want.

Each waypoint (vertex) has several incoming and outgoing edges in the new directed graph. The set of flowlines we are looking for is a *branching* or, synonymously, a *directed spanning tree* i.e. a graph $\mathcal{T}$ such that:

- $\mathcal{T}$ contains no cycle,
- each vertex has *one and only one* incoming edge: each vertex is visited (the tree is *spanning*), but no two edges of $\mathcal{T}$ enter the same vertex (i.e., each vertex has only one downstream pixel, since we build the flowlines from downstream towards upstream).

Of course, such a tree $\mathcal{T}$ has to be rooted at a snout, i.e. a special vertex without an incoming edge. Moreover, we want the edges of this tree to be —as much as possible— steepest climb routes, i.e. to have low costs with respect to our cost function. Extracting a subset of edges (a subgraph) which satisfies all these requirements (being a tree, being spanning, and having a minimal cost) is a classical problem called Directed Minimum Spanning Tree (DMST) extraction. It is efficiently solved with the Chu-Liu/Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967): in this study we use the implementation of Tofigh (2009), based on the Boost Graph Library (Siek et al., 2002).

Since there are several snouts in a glacier complex, the subgraph may actually be a *forest* i.e. a set of trees, each one rooted at a different snout. The single-root version of the Edmonds algorithm is easily extended to the multiple-root case: a virtual 'master' root is created, and zero-cost edges are added from this root to each snout (see Figure 12).

Figure 13 shows the output of the algorithm for the Rhone-Trift glacier complex. It is worth noting that the segmentation of the complex into individual glaciers is a by-product of the method: each snout 'drains' a set of upstream vertices which form its catchment.

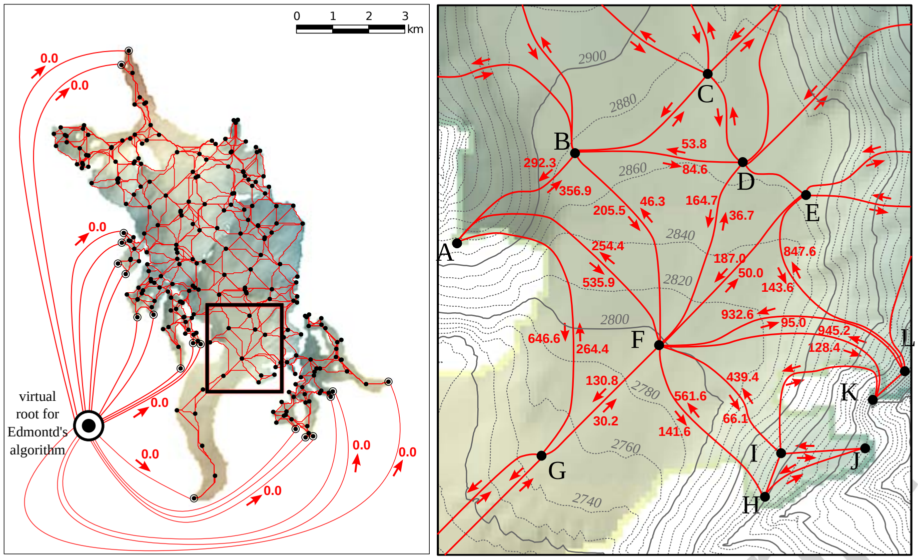


Figure 12: Edges and associated costs between skeleton waypoints in the Rhone-Trift glacier complex. Left: example of initial graph with numerous edges linking pairs of waypoints. For clarity, the plot displays only the edges between natural (planar) neighbors but a much denser graph can be created. Right: zoom on a region. Figures in red are the costs of each edge, in both directions.

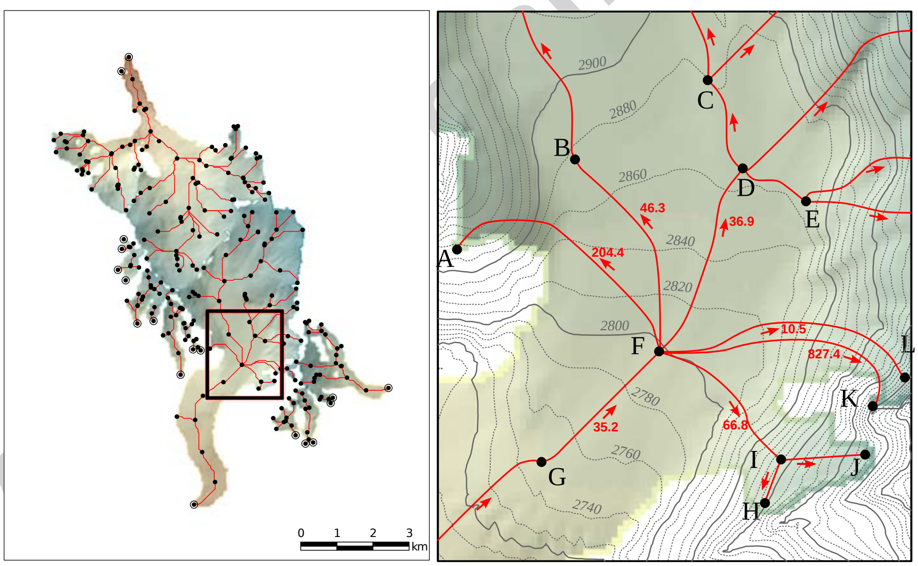


Figure 13: Output of the Chu-Liu/Edmonds algorithm with the set of edges and costs of Figure 12. The minimum branching (Directed Mininum Spanning Tree/Forest) allows to visit all waypoints, starting from a snout and always moving as upslope as possible.

Figure 14 is a zoom on a pass across the topographical divide between Rhone and Trift glaciers (Undri Triftlimi). It explains why no edge in the DMST/DMSF should cross any major topographical ridge. Let us suppose that the ridge-crossing edge (**K**,**L**) is part of the DMSF, denoted $\mathcal{F}$ and of total cost $C_{\uparrow}(\mathcal{F})$. In this case, point **L** ultimately drains to Trift glacier's snout. However, the other incoming edge to **L**, (**M**,**L**), has a lower cost than (**K**,**L**): hence if we remove (**K**,**L**) from $\mathcal{F}$ and add (**M**,**L**), we obtain a forest $\mathcal{F}'$ that is still spanning (**L** is visited i.e. has an incoming edge) and has a lower total cost $C_{\uparrow}(\mathcal{F}') = C_{\uparrow}(\mathcal{F}) + C_{\uparrow}(\mathbf{M}, \mathbf{L}) - C_{\uparrow}(\mathbf{K}, \mathbf{L}) < C_{\uparrow}(\mathcal{F})$. This means that $\mathcal{F}$ was not a DMSF in the first place (it is spanning but not of minimum cost): point **L** has to flow to Rhone glacier's snout, and no edge should go through the pass. Hence, our procedure is efficient not only for flowline extraction, but also for glacier segmentation.

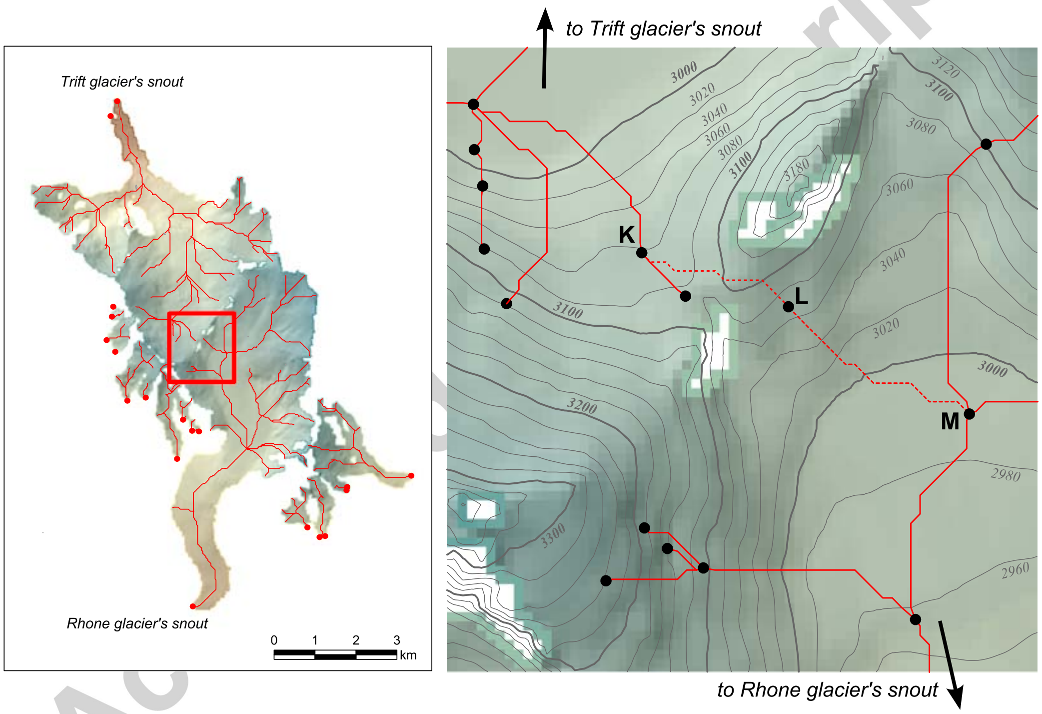


Figure 14: Zoom on the minimum branching in the vicinity of Undri Triftlimi, a pass on the divide between Rhone and Trift glaciers. Vertex **L**, which is located south of the ridge, has to be visited from **M** and not from **K** in the Directed Minimum Spanning Tree/Forest: no edge should go across any major divide.

6. Conclusion and perspectives

In this paper we proposed a new method to extract glacier flowlines, based on Voronoi skeletonization of glacier boundaries, skeleton pruning using Discrete Curve Evolution (DCE), and the construction of a Directed Minimum Spanning Tree between skeletal vertices with respect to an anisotropic, upslope cost function $C_\uparrow$.

The application of the method requires limited parameter tweaking: it only requires a selection rule $k = f(A)$ for the level k of the DCE as a function of glacier area A (2 parameters), and 3 parameters for the friction law (i.e. the isotropic term of the cost function). Table 2 sums up the values used on the whole domain of Figure 2a (1200 km^2 of glaciers, the largest contiguous icefield having an area of 261 km^2).

DCE level selection $k = 2 + \lfloor k_0 A^\gamma \rfloor$	k_0	13.5
	γ	0.8
Cost function $C_\uparrow(\mathbf{x}\,,\,\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) = \lVert\boldsymbol{\nabla} z\rVert \lVert\boldsymbol{\delta}\mathbf{x}\rVert - (\boldsymbol{\nabla} z)\cdot\boldsymbol{\delta}\mathbf{x} + \lambda(\mathbf{x})\lVert\boldsymbol{\delta}\mathbf{x}\rVert$ with $\lambda(\mathbf{x}) = \lambda_\infty + (\lambda_0 - \lambda_\infty)e^{-\frac{D(\mathbf{x})}{D_\lambda}}$	λ_∞	$0.035\ \mathrm{m}\cdot\mathrm{m}^{-1}$
	λ_0	$0.175\ \mathrm{m}\cdot\mathrm{m}^{-1}$
	D_λ	150 m

Table 2: Parameters of the method.

The method currently lacks a quality assessment, though this could be done on a small subset of glaciers by comparison with manually-extracted flowlines. The large-scale visual assessment is however very satisfying, and the resulting network can be used to compute indices such as Strahler orders, bifurcation ratios, etc. Many scaling properties of glaciers, such as volume-are scaling (Bahr et al., 1997) or power-law behavior of accumulation basin areas (Gsell et al., 2014), originate in glacier branching topology: such properties could act as a 'lever arm' for tackling the problem of catchment-scale glacier flow dynamics, much like other fundamental symmetries (plane-strain or radial), as advocated by Bahr and Peckham (1996). We hope that this study will foster ideas in this field.

7. Acknowledgments

This work has been supported by an EC2CO grant from the French CNRS (INSU). The authors acknowledge the Swiss Federal Office of Topography (SwissTopo) for providing the 25-meter DEM.

8. References

[1] Arendt, A., Bolch, T., Cogley, J. G., Gardner, A., Hagen, J. O., Hock, R., Kaser, G., et al., 2012. Randolph Glacier Inventory – A Dataset of Global Glacier Outlines: Version 3.2. Global Land Ice Measurements from Space, Boulder, CO, USA. Digital Media.
URL http://www.glims.org/RGI/

[2] Bahr, D. B., Meier, M., Peckham, S. D., 1997. The physical basis of glacier volume-area scaling. J. Geophys. Res. 102, 20355–20362.

[3] Bahr, D. B., Peckham, S. D., 1996. Observations and analysis of self-similar branching topology in glacier networks. J. Geophys. Res. 101, 25511–25521.

[4] Bai, X., Latecki, L. J., Liu, W.-Y., 2007. Skeleton pruning by contour partitioning with discrete curve evolution. IEEE Trans. Pattern Anal. Mach. Intell. 29 (3), 449–462.

[5] Barber, C. B., Dobkin, D. P., Huhdanpaa, H. T., 1996. The Quickhull algorithm for convex hulls. ACM Trans. on Mathematical Software 22 (2), 469–483.
URL http://www.qhull.org

[6] Blum, H., 1967. A Transformation for Extracting New Descriptors of Shape. In: Wathen-Dunn, W. (Ed.), Models for the Perception of Speech and Visual Form. MIT Press, Cambridge, pp. 362–380.

[7] Brandt, J. W., Algazi, V. R., 1992. Continuous skeleton computation by Voronoi diagram. CVGIP: Image Understanding 55 (3), 329–338.

[8] Chu, Y. J., Liu, T. H., 1965. On the shortest arborescence of a directed graph. Science Sinica 14, 1396–1400.

[9] Collischonn, W., Pilar, J. V., 2000. A direction dependent least-cost-path algorithm for roads and canals. International Journal of Geographical Information Science 14 (4), 397–406.

[10] Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271.
URL http://dx.doi.org/10.1007/BF01386390

[11] Douglas, D. H., 1994. Least-cost Path in GIS Using an Accumulated Cost Surface and Slopelines. Cartographica 31 (3), 37–51.

[12] Edmonds, J., 1967. Optimum branchings. J. Res. Natl. Bur. Stand. 71B, 233–240.

[13] Garbrecht, J., Martz, L. W., 1997. The assignment of drainage direction over flat surfaces in raster digital elevation models. Journal of Hydrology 193 (1-4), 204–213.

[14] Gsell, P.-S., Le Moine, N., Moussa, R., Ribstein, P., 2014. Identifying the probabilistic structure of drained areas as a function of hypsometry in river networks. Hydrological Processes.
URL http://dx.doi.org/10.1002/hyp.10296

[15] Gupta, V. K., Waymire, E., Wang, C. T., 1980. A representation of an instantaneous unit hydrograph from geomorphology. Wat. Resour. Res. 16 (5), 855–862.

[16] Kienholz, C., Rich, J. L., Arendt, A. A., Hock, R., 2014. A new method for deriving glacier centerlines applied to glaciers in Alaska and northwest Canada. The Cryosphere 8 (2), 503–519.

[17] Latecki, L. J., Lakämper, R., 2002. Application of Planar Shape Comparison to Object Retrieval in Image Databases. Pattern Recognition 35, 15–29.

[18] Le Bris, R., Paul, F., 2013. An automatic method to create flow lines for determination of glacier length: A pilot study with Alaskan glaciers. Computers & Geosciences 52, 234–245.

[19] Le Moine, N., 2013. CLaRiNet: Catchment, Landform, and River Network analysis – A geomorphological toolbox for Scilab, User Manual (draft version). UPMC.
URL http://www.sisyphe.upmc.fr/~lemoine/resources.htm

[20] Machguth, H., Huss, M., 2014. The length of the world's glaciers – a new approach for the global calculations of center lines. The Cryosphere 8, 1741–1755.

[21] Martz, L. W., Garbrecht, J., 1998. The treatment of flat areas and depressions in automated drainage analysis of raster digital elevation models. Hydrological Processes 12 (6), 843–855.

[22] Rodríguez-Iturbe, I., Valdés, J. B., 1979. The geomorphological structure of the hydrologic response. Wat. Resour. Res. 15, 1409–1420.

[23] Scilab Enterprises, 2012. Scilab: Free and Open Source software for numerical computation. Scilab Enterprises, Orsay, France.
URL http://www.scilab.org

[24] Siek, J. G., Lee, L.-Q., Lumsdaine, A., 2002. The Boost Graph Library: User Guide and Reference Manual. Addison-Wesley, Boston, Massachusetts.
URL http://www.boost.org/doc/libs/release/libs/graph/

[25] SwissTopo, 2004. DHM25, The digital height model of Switzerland, Product Information. Federal Office of Topography, Wabern, CH.

[26] Tofigh, A., 2009. Optimum Branchings and Spanning Arborescences. Accessed on SourceForge.
URL http://sourceforge.net/projects/edmonds-alg/

[27] Warntz, W., 1957. Transportation, social physics, and the law of refraction. The Professional Geographer 9 (4), 2–7.
URL http://dx.doi.org/10.1111/j.0033-0124.1957.094_2.x