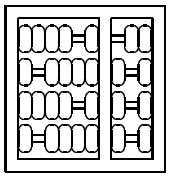

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

The Image Foresting Transform: Theory, Algorithms and Applications

A.X. Falcão *J. Stolfi* *R.A. Lotufo*

Technical Report - IC-03-04 - Relatório Técnico

April - 2003 - Abril

Abstract

The image foresting transform (IFT) is a graph-based approach to the design of image processing operators based on connectivity. It naturally leads to correct and efficient implementations, and to a better understanding of how different operators relate to each other. We give here a precise definition of the IFT, and a procedure to compute it—a generalization of Dijkstra's algorithm—with a proof of correctness. We also discuss implementation issues and illustrate the use of the IFT in a few applications.

Index terms: Dijkstra's algorithm, shortest-path problems, image segmentation, image analysis, regional minima, watershed transform, morphological reconstruction, boundary tracking, distance transforms, and multiscale skeletonization.

1 Introduction

The *image foresting transform* (IFT)[1], which we describe here, is a general tool for the design, implementation, and evaluation of image processing operators based on connectivity. The IFT defines a *minimum-cost path forest* in a graph, whose nodes are the image pixels and whose arcs are defined by an *adjacency relation* between pixels. The cost of a path in this graph is determined by an application-specific *path-cost function*, which usually depends on local image properties along the path— such as color, gradient, and pixel position. The roots of the forest are drawn from a given set of *seed pixels*. For suitable path-cost functions, the IFT assigns one minimum-cost path from the seed set to each pixel, in such a way that the union of those paths is an oriented forest, spanning the whole image. The IFT outputs three attributes for each pixel: its predecessor in the optimum path, the cost of that path, and the corresponding root (or some label associated with it). A great variety of powerful image operators, old and new, can be implemented by simple local processing of these attributes.

We describe a general algorithm for computing the IFT, which is essentially Dijkstra's shortest-path algorithm [1–3], slightly modified to multiple sources and general path-cost functions. Since our conditions on the path costs apply only to optimum paths, and not to all paths, we found it necessary to rewrite and extend the classical proof of the correctness of Dijkstra's algorithm. In many practical applications, the path costs are integers with limited increments and the graph is sparse; therefore, the optimizations described by Dial [4] and Ahuja et al. [5] will apply, and the running time will be linear on the number of pixels.

The IFT unifies and extends many image analysis techniques which, even though based on similar underlying concepts (ordered propagation, flooding, geodesic dilations, dynamic programming, region growing, A^* graph search, etc.), are usually presented as unrelated methods. Those techniques can all be reduced to a partition of the image into *influence zones* associated with a given seed set, where the zone of each seed consists of the pixels that are "more closely connected" to that seed than to any other, in some appropriate sense. These influence zones are simply the trees of the forest defined by the IFT. Examples are watershed transforms [6, 7] and fuzzy-connected segmentation [8–11]. The IFT also provides a mathematically sound framework for many image-processing operations that

[1] This paper was submitted to IEEE Trans. on Pattern Analysis and Machine Intelligence.

are not obviously related to image partition, such as morphological reconstruction [12], distance transforms [13, 14], multiscale skeletons [14], shape saliences and multiscale fractal dimension [15, 16], and boundary tracking [17–19].

By separating the general forest computation procedure from the application-specific path-cost function, the IFT greatly simplifies the implementation of image operators, and provides a fair testbed for their evaluation and tuning. For many classical operators, the IFT-based implementation [6] is much closer to the theoretical definition than the published algorithms [20, 21]. Indeed, many algorithms which have been used without proof [20–24] have their correctness established by being re-formulated in terms of the IFT [6, 12]. By clarifying the relationship between different image transforms [12], the IFT approach often leads to novel image operators and considerably faster (but still provably correct) algorithms [14, 25, 26].

The IFT definition and algorithms are independent of the nature of the pixels and of the dimension of the image; and therefore they apply to color and multispectral images, as well as to higher-dimensional images such as video sequences and tomography data [26].

Section 2 reviews the previous related work. We define the IFT and related concepts in Section 3. Section 4 presents the basic IFT algorithm, a proof of its correctness and optimization hints. Tie-breaking policies are discussed in Section 5. In Section 6, we illustrate the use of the IFT in a few selected applications. Section 7 contains the conclusions and current research on the IFT.

2 Related work

2.1 Shortest-path problems in general graphs

The problem of computing shortest paths in a general graph has an enormous bibliography [27]. Most authors dispose of the multiple-source problem by trivial reduction to the single-source version. Solutions to the latter were first proposed by Moore in 1957 [1] and Bellman in 1958 [2]. The running time of their algorithms was $O(mn)$, in the worst case, for a graph with n nodes and m arcs. An improved algorithm was given by Dijkstra in 1959 [3]; its running time of $O(n^2)$ can be reduced to $O(m + nlogn)$ through the use of a balanced heap data structure [5]. In 1969, Dial published a variant of Dijkstra's algorithm [4, 5], for the special case where the arc costs are integers in the range $[0..K]$, which uses bucket sorting to achieve running time $O(m + nK)$.

The standard path-cost function is the sum of arc costs. Extensions to more general cost models fall in two major classes. The algebraic or "semiring" model considers fixed arc costs that are combined by an associative operation, that distributes over the min function (or, more generally, over a second associative operator which is used to summarize all paths leading to a given node) [28, 29]. Another model does not assume associativity or fixed arc costs, but requires that the cost of a path be computed incrementally from the cost of its prefixes through a composition function that satisfies certain monotonicity constraints [30]. We note that, in both approaches, the constraints on the path-cost functions are required to hold for all paths, not just for optimum ones.

2.2 Image processing using graph concepts

In image processing, Montanari [31] and Martelli [32, 33] were the first to formulate a boundary finding problem as a shortest-path problem in a graph. Montanari [31] required the boundary to be star-shaped [34]. Martelli [32, 33] considered only a selected subset of the arcs, and therefore his algorithm may fail in some situations. These restrictions have since been lifted by Falcão et al. [17] and Mortensen et al. [35]. In region-based image segmentation, Udupa et al. [8, 9] proposed the fuzzy connectedness theory for object definition, which was efficiently implemented by Nyúl et al. using Dial's bucket queue [10]. Verwer et al. [36] and others [37] used the Dial's algorithm to compute weighted distance and chamfer distance transforms. Meyer extended their work to some variations of the watershed transform based on the eikonal equation [38]. Dial's bucket queue became the core of fast ordered propagation algorithms for various applications, including Euclidean distance transform [39, 40], watershed transforms [20, 21, 41], and morphological reconstructions [22–24].

2.3 The PDE approach

There is a certain formal resemblance between the IFT and the *partial differential equations* (PDE) approach to image processing [42], since both involve the sweeping of the image by a propagating front— which, in the IFT approach, is the boundary between the sets S and $\bar{S}$ of Dijkstra's original algorithm [5]. The eikonal problem, specifically, asks for the minimum-time path from the initial front to each point of the domain. The reciprocal of the local speed of propagation is analogous to the arc costs of the IFT model, and the Fast Marching method is seen as the analog of Dijkstra's algorithm [43].

To a large extent, the comparison of the two approaches is just another instance to the continuous vs. discrete question. The discreteness of the IFT allows rigorous proofs and predictable algorithms, free from approximation errors and numerical instabilities that often affect the PDE. The IFT model also allows the cost of extending a path to depend on the whole path, and not just on its current cost. On the other hand, the PDE approach allows the propagation front to move back and forth, or with speed that depends on its local shape—features that are used in contour smoothing, contour detection, and other applications. It is not obvious whether similar effects can be achieved in the IFT model, where the "shape of the front" is not a well-defined concept.

3 Notation and Definitions

An *image* $\mathbf{I}$ is a pair $(\mathcal{I}, I)$ consisting of a finite set $\mathcal{I}$ of *pixels* (points in $\mathbb{Z}^2$), and a mapping I that assigns to each pixel t in $\mathcal{I}$ a *pixel value* $I(t)$ in some arbitrary value space.

An *adjacency relation* $\mathcal{A}$ is an irreflexive binary relation between pixels of $\mathcal{I}$. Once the adjacency relation $\mathcal{A}$ has been fixed, the image $\mathbf{I}$ can be interpreted as a directed graph whose nodes are the image pixels and whose arcs are the pixel pairs in $\mathcal{A}$. In what follows, $n = |\mathcal{I}|$ is the number of nodes (pixels) and $m = |\mathcal{A}|$ is the number of arcs.

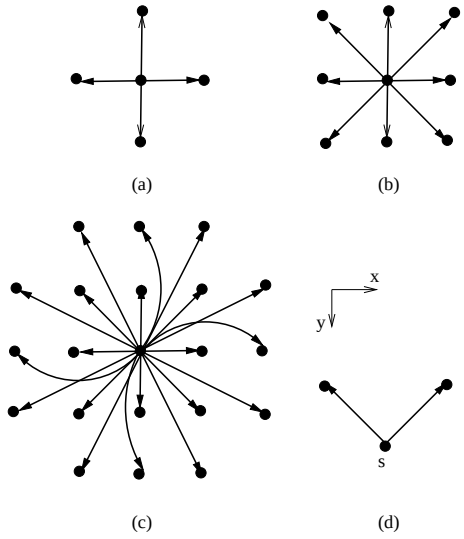

Figure 1: (a)-(c) Euclidean adjacency relations for $\rho = 1, \sqrt{2}$ and $\sqrt{5}$, respectively. (d) An asymmetric adjacency relation given by $\mathcal{M} = \{(-1,-1),(1,-1)\}$.

In most image-processing applications, the adjacency relation $\mathcal{A}$ is *translation-invariant*, meaning that $x\mathcal{A}y$ depends only on the relative position $y - x$ of the pixels. For example, one often takes $\mathcal{A}$ to consist of all pairs of distinct pixels $(s,t) \in \mathcal{I} \times \mathcal{I}$ such that $d(s,t) \leq \rho$, where $d(s,t)$ denotes the Euclidean distance and ρ is a specified constant. Figures 1a-c show the adjacent pixels of a fixed central pixel s in this *Euclidean adjacency relation*, when $\rho = 1$ (the 4-*connected* adjacency), $\rho = \sqrt{2}$ (*8-connected*) and $\rho = \sqrt{5}$, respectively.

More generally, we can take a finite subset $\mathcal{M}$ of $\mathbb{Z}^2 \setminus \{(0,0)\}$, for example $\mathcal{M} = \{(-1,-1),(1,-1)\}$, and define $\mathcal{A}$ as all pairs of pixels (s,t) where $t - s \in \mathcal{M}$. See Figure 1d. Note that the adjacency relation does not have to be symmetric.

A *path* is a sequence of pixels $\pi = \langle t_1, t_2, \ldots, t_k \rangle$ where $(t_i, t_{i+1}) \in \mathcal{A}$ for $1 \leq i \leq k-1$. We denote the *origin* t_1 and the *destination* t_k of π by $\mathrm{org}(\pi)$ and $\mathrm{dst}(\pi)$, respectively. The path is *trivial* if $k = 1$. If π and τ are paths such that $\mathrm{dst}(\pi) = \mathrm{org}(\tau) = t$, we denote by $\pi \cdot \tau$ the concatenation of the two paths, with the two joining instances of t merged into one.

3.1 Path costs

We assume given a function f that assigns to each path π a *path cost* $f(\pi)$, in some totally ordered set $\mathcal{V}$ of cost values. Without loss of generality, we can always assume that $\mathcal{V}$ contains a maximum element, which we denote by $+\infty$. Usually, the path cost depends on local properties of the image $\mathbf{I}$—such as color, gradient, and pixel position—along the path.

A popular example is the *additive* path-cost function, which satisfies

$$f_{\mathrm{sum}}(\langle t \rangle) \quad = \quad h(t),$$

$$f_{\text{sum}}(\pi \cdot \langle s,t \rangle) \quad = \quad f_{\text{sum}}(\pi) + w(s,t), \tag{1}$$

where $(s,t) \in \mathcal{A}$, π is any path ending at s, $h(t)$ is a fixed *handicap cost* for any paths starting at pixel t, and $w(s,t)$ is a fixed non-negative weight assigned to the arc (s,t).

Another important example is the *max-arc* path-cost function $f_{\max}$, defined by

$$\begin{aligned} f_{\max}(\langle t \rangle) &= h(t), \\ f_{\max}(\pi \cdot \langle s,t \rangle) &= \max\{f_{\max}(\pi), w(s,t)\}, \end{aligned} \tag{2}$$

where $h(t)$ and $w(s,t)$ are fixed but arbitrary handicap and arc weight functions. Note that f_{sum} and $f_{\max}$ are distinct models: since the incremental cost $f_{\max}(\pi \cdot \langle s,t \rangle) - f_{\max}(\pi)$ depends on the cost of π, one cannot redefine $f_{\max}$ as the sum of *fixed* weights $w'(s,t)$.

3.2 Seed pixels

In typical applications of the IFT, we would like to use a pre-defined path-cost function f but constrain the search to paths that start in a given set $\mathcal{S} \subseteq \mathcal{I}$ of *seed pixels*. We can model this constraint by defining a new path-cost function $f^{\mathcal{S}}(\pi)$, which is equal to $f(\pi)$ when $\text{org}(\pi) \in \mathcal{S}$, and $+\infty$ otherwise. In particular, for the f_{sum} and $f_{\max}$ functions, this is equivalent to setting $h(t) = +\infty$ for pixels $t \notin \mathcal{S}$.

3.3 Optimum paths

We say that a path π is *optimum* if $f(\pi) \leq f(\pi')$ for any other path π' with $\text{dst}(\pi') = \text{dst}(\pi)$, irrespective of its starting point. In that case, $f(\pi)$ is by definition the *cost* of pixel $t = \text{dst}(\pi)$, denoted by $\hat{f}(t)$. Note that a trivial path is not necessarily optimum: even for f_{sum} or $f_{\max}$, the handicaps of two pixels s and t may be such that a non-trivial path from s to t is cheaper than the trivial path $\langle t \rangle$.

3.4 Spanning forests

A *predecessor map* is a function P that assigns to each pixel t in $\mathcal{I}$ either some other pixel in $\mathcal{I}$, or a distinctive marker *nil* not in $\mathcal{I}$ — in which case t is said to be a *root* of the map. A *spanning forest* is a predecessor map which contains no cycles — in other words, one which takes every pixel to *nil* in a finite number of iterations. For any pixel $t \in \mathcal{I}$, a spanning forest P defines a path $P^*(t)$ recursively as $\langle t \rangle$ if $P(t) = \textit{nil}$, and $P^*(s) \cdot \langle s,t \rangle$ if $P(t) = s \neq \textit{nil}$. We will denote by $P^0(t)$ the initial pixel of $P^*(t)$ (see Figure 2a).

3.5 The Image Foresting Transform

The *image foresting transform* (IFT) takes an image $\mathbf{I}$, a path-cost function f and an adjacency relation $\mathcal{A}$; and returns an *optimum-path forest*— a spanning forest P such that $P^*(t)$ is optimum, for every pixel t. See Figures 2b-c.

Note that, in an optimum-path forest P for a seed-restricted cost function $f^{\mathcal{S}}$, any pixel t with finite cost $f^{\mathcal{S}}(P^*(t))$ will belong to a tree whose root is a seed pixel; however, some seeds may not be roots of the forest.

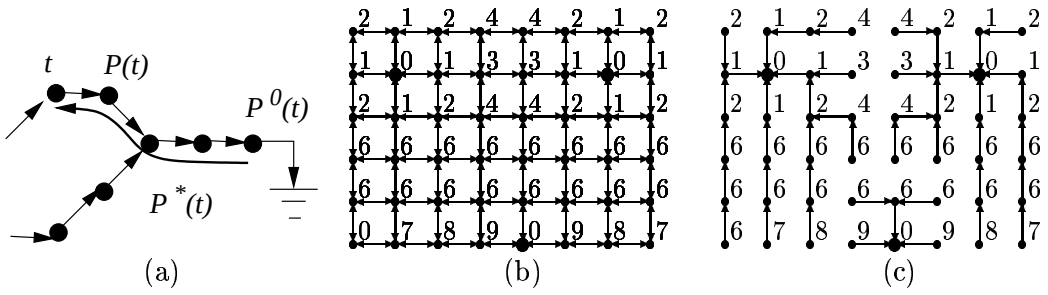

Figure 2: (a) The main elements in a spanning forest. (b) An image graph with 4-connected adjacency, where the integers are the image values $I(t)$. (c) An optimum-path forest for the path-cost function $f_{\max}$, where $h(t) = w(s,t) = I(t)$, restricted to the three seed pixels represented by bigger dots.

In general, there may be many paths of minimum cost leading to a given pixel, and many optimum-path forests; only the pixel costs $\hat{f}(t)$ are uniquely defined. Observe that, if we independently pick an optimum path for each pixel, the union of those paths may not be a forest. Indeed, certain graphs and cost functions may not even admit *any* optimum-path forest. Sufficient conditions for the existence of the IFT will be given in Section 4.3.

4 Algorithm

For suitable path-cost functions, the IFT can be computed by Algorithm 1 below— which is essentially Dijkstra's procedure for computing minimum-cost paths from a single source in a graph [3, 5], slightly modified to allow multiple sources and more general cost functions. This variant was chosen for maximum flexibility and to simplify the proof of correctness. In practice, Algorithm 1 can be optimized in a number of ways (see Section 4.5).

Algorithm 1 Input*: An image* $\mathbf{I} = (\mathcal{I}, I)$*; an adjacency relation* $\mathcal{A} \subset \mathcal{I} \times \mathcal{I}$*; and a path-cost function* f*.* Output*: An optimum-path forest* P*.* Auxiliary Data Structures*: Two sets of pixels* $\mathcal{F}$, $\mathcal{Q}$ *whose union is* $\mathcal{I}$*.*

1. Set $\mathcal{F} \leftarrow \{\}$, $\mathcal{Q} \leftarrow \mathcal{I}$.
   For all pixels $t \in \mathcal{I}$, set $P(t) \leftarrow nil$.
2. While $\mathcal{Q} \neq \{\}$, do
   2.1. Remove from $\mathcal{Q}$ a pixel s such that $f(P^*(s))$ is minimum, and add it to $\mathcal{F}$.
   2.2. For each pixel t such that $(s,t) \in \mathcal{A}$, do
      2.2.1. If $f(P^*(s) \cdot \langle s,t \rangle) < f(P^*(t))$, set $P(t) \leftarrow s$.

4.1 General properties

Some basic facts about Algorithm 1 are easily established by induction on the number of steps. Since every iteration of the main loop removes from $\mathcal{Q}$ exactly one pixel (which is never returned), and each arc of $\mathcal{A}$ is examined exactly once in step 2.2, it follows that

Lemma 1 *Algorithm 1 terminates in $O(n)$ iterations of the outer loop, and $O(m)$ total iterations of the inner loop.*

Moreover, each predecessor $P(t)$ is initially *nil*, and is modified only by the assignment $P(t) \leftarrow s$ in step 2.2.1 — when t is still in $\mathcal{Q}$ but s is in $\mathcal{F}$. It follows that

Lemma 2 *The predecessor map P computed by Algorithm 1 is always a spanning forest.*

Note that lemmas 1 and 2 hold for *any* path-cost function f. This is more of a curse than a blessing, because it tempts people into using Algorithm 1 even when its result is not an optimum-path forest. The optimality depends on f being sufficiently well-behaved.

4.2 Monotonic-incremental cost functions

When the path-cost function is additive, the correctness of Algorithm 1 is established by the standard proof of Dijkstra's method [5, 28]. Note that the extension to multiple starting pixels is trivial, since it is equivalent to adding extra arcs from a dummy starting pixel $u \notin \mathcal{I}$ to all pixels in $\mathcal{I}$, and setting $w(u,t) = h(t)$ for each new arc (u,t). This remains true even when the arc costs and handicaps are allowed to be $+\infty$.

In fact, as shown by Frieze [30], the original proof of Dijkstra's algorithm is easily generalized to *monotonic-incremental* (MI) path-cost functions, which satisfy

$$\begin{aligned} f(\langle t \rangle) &= h(t), \\ f(\pi \cdot \langle s,t \rangle) &= f(\pi) \odot (s,t), \end{aligned} \tag{3}$$

where $h(t)$ is an arbitrary handicap cost, and $\odot : \mathcal{V} \times \mathcal{A} \to \mathcal{V}$ is a binary operation that satisfies the conditions

(M1) $x' \geq x \to x' \odot (s,t) \geq x \odot (s,t)$,

(M2) $x \odot (s,t) \geq x$,

for any $x, x' \in \mathcal{V}$ and any $(s,t) \in \mathcal{A}$. An essential feature of this model is that $\odot$ depends only on the *cost* of π, and not on any other property of π. Both the additive cost f_{sum} (with non-negative arc weights) and the max-arc cost $f_{\max}$ are monotonic-incremental. It turns out that most image processing problems which we have successfully reduced to the IFT require MI path-cost functions, and therefore can be solved by Algorithm 1. In fact, condition (M2) can be weakened to $f(\pi \cdot \kappa) \geq f(\pi)$ for any cycle κ.

On the other hand, it is easy to find counterexamples of path-cost functions — not MI, of course — which cause Algorithm 1 to fail. A textbook counter-example is the additive

path-cost f_{sum} when the arc weights $w(s,t)$ are allowed to be negative. The algorithm may also fail for generalizations of f_{sum} or $f_{\max}$ where the arc weight $w(s,t)$ is allowed to depend on the path already chosen for s. Unfortunately, this applies to several path-cost functions that would seem reasonable for image processing. For example, in multi-seeded region-based segmentation, it may seem reasonable to use the function $f_{\text{abs}}(\pi)$, defined as the maximum of $|I(t) - I(\text{org}(\pi))|$ for any pixel t along the path π. Figure 3 shows a situation where Dijkstra's algorithm will fail (and, in fact, where the optimum paths do not form a forest). The same image graph is a counter-example when the path cost $f(\pi)$ is defined as the variance of the pixel values along π. Another counter-example is the region-growing criterion proposed by Adams et al. [44], where each candidate pixel is ranked by the absolute difference between its value and the mean value of all pixels in each region.

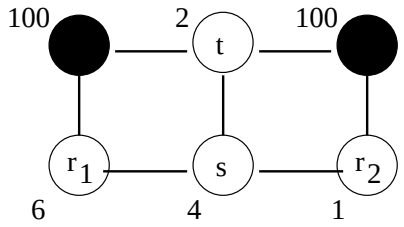

Figure 3: A 2×3 image where Algorithm 1 fails to produce an optimum path forest for $f^{\mathcal{S}}_{\text{abs}}$, with $\mathcal{S} = \{r_1, r_2\}$. We have $\hat{f}_{\text{abs}}(s) = f_{\text{abs}}(\langle r_1, s\rangle) = 2 < f_{\text{abs}}(\langle r_2, s\rangle) = 3$, but $\hat{f}_{\text{abs}}(t) = f_{\text{abs}}(\langle r_2, s, t\rangle) = 3 < f_{\text{abs}}(\langle r_1, s, t\rangle) = 4$.

4.3 Smooth path-cost functions

Algorithm 1 *does* work for certain cost functions which are not MI, or even monotonic. Consider for example the 4-connected adjacency and the function $f^{\mathcal{S}}_{\text{euc}}(\pi)$ defined as the Euclidean distance between the endpoints of π, restricted to paths that start at a given seed set $\mathcal{S}$. Algorithm 1 will correctly find an optimum-path forest as long as $|\mathcal{S}| \leq 2$, even though condition (M2) is violated when $|\mathcal{S}| = 1$, and both (M1) and (M2) are violated when $|\mathcal{S}| = 2$. (The algorithm may fail, however, if $|\mathcal{S}| \geq 3$; see Section 6.4.)

Examples like this one led us to search for conditions on the path-cost function f that are more general than (M1) and (M2) but still strong enough to ensure the correctness of Algorithm 1. Specifically, we claim that the algorithm will work if, for any pixel $t \in \mathcal{I}$, there is an optimum path π ending at t which either is trivial, or has the form $\tau \cdot \langle s, t\rangle$ where

(C1) $f(\tau) \leq f(\pi)$,

(C2) τ is optimum,

(C3) for any optimum path τ' ending at s, $f(\tau' \cdot \langle s, t\rangle) = f(\pi)$.

These conditions seem to capture the essential features of the path-cost function that are used in the classical proofs (cf. Bellman's optimality principle [45]). Observe that conditions

(C1)–(C3) are not required to hold for *all* paths ending at t, but only for *some* path π that is optimum. We say that a path-cost function f satisfying conditions (C1)–(C3) is *smooth*.

It can be checked that any MI path-cost function satisfies conditions (C1)–(C3). Also, if f is a MI cost function, then its restriction $f^{\mathcal{S}}$ to an arbitrary seed set $\mathcal{S}$ will be MI and hence smooth too. (Unfortunately, this is not necessarily true if f is smooth, but not MI.)

For an example of a smooth function that is not MI, let f be an MI function and define $f'(\pi) = f(\pi) + g(\pi)$, where $g(\pi)$ is zero if π is optimum for f, and an arbitrary positive value otherwise. The function f' satisfies conditions (C1)–(C3), even though it may fail (M1) for arbitrary paths. For a more realistic example, it can be checked that $f^{\mathcal{S}}_{\text{euc}}$ is smooth when $|\mathcal{S}| \leq 2$. Two key observations are that the influence zones of the seed pixels are 4-connected, and that any path π with minimum number of arcs connecting $\mathcal{S}$ to t satisfies conditions (C1)–(C3).

4.4 Proof of correctness

In order to prove the correctness of Algorithm 1 for a smooth path-cost function f, we require some auxiliary definitions. Let $\mathcal{B} \subseteq \mathcal{A}$ denote the set of all arcs of $\mathcal{A}$ that have already been considered in step 2.2. We say that a path is a $(\mathcal{B}, P)$-*path* if it is either a trivial path $\langle t \rangle$ with $t \in \mathcal{Q}$, or has the form $P^*(s) \cdot \langle s, t \rangle$, where $s \in \mathcal{F}$, $t \in \mathcal{Q}$, and (s, t) is an arc of $\mathcal{B}$. We also say that a path is $(\mathcal{B}, P)$-*optimum* if it is a $(\mathcal{B}, P)$-path, and has minimum cost among all $(\mathcal{B}, P)$-paths with the same final pixel. Note that a $(\mathcal{B}, P)$-optimum path may not be optimum overall, and therefore we cannot assume that it satisfies conditions (C1)–(C3).

The proof is based on the following invariants which, we claim, hold every time the algorithm reaches one of the steps 2.1, 2.2, or 2.2.1:

(H1) For every pixel $s \in \mathcal{F}$, the path $P^*(s)$ is optimum.

(H2) For every pixel $s \in \mathcal{F}$, and any path π that ends in $\mathcal{Q}$, we have $f(P^*(s)) \leq f(\pi)$.

(H3) For every pixel $t \in \mathcal{Q}$, the path $P^*(t)$ is $(\mathcal{B}, P)$-optimum.

The bulk of the proof lies in lemmas 3– 5 below, which are combined in theorem 6. The proofs of these lemmas are given in Appendix A.

Lemma 3 *Each execution of step 2.2.1 preserves invariants* H1–H3.

Corollary 4 *Each execution of step 2.2 preserves invariants* H1–H3.

Lemma 5 *If f is a smooth path-cost function, step 2.1 preserves invariants* H1–H3.

Theorem 6 *Algorithm 1 computes an optimum-path forest for any smooth cost function f.*

Proof: Invariants (H1–H3) are trivially valid just before step 2. By lemmas 3 and 5, they remain valid through the execution of the algorithm, which terminates by lemma 1. Upon termination, P is a spanning forest by lemma 2; and $\mathcal{F} = \mathcal{I}$ together with invariant (H1) imply that all paths $P^*(t)$ are optimum. ■

4.5 Efficient implementation

Algorithm 1 can be optimized in a number of ways, without affecting its correctness:

Reusing the path costs. One can avoid recomputing $f(P^*(t))$ at every iteration of step 2.2.1 by storing $f(P^*(t))$ in a table $C(t)$. Moreover, the cost $f(P^*(s) \cdot \langle s,t \rangle)$ can often be computed in $O(1)$ time from $w(s,t)$ and $f(P^*(s)) = C(s)$ (and, if f is not MI, from some other accumulated information about $P^*(s)$). At the end of the algorithm, $C(t)$ will be the minimum pixel cost $\hat{f}(t)$.

Avoiding backward arcs. From invariants (H1) and (H2), it follows that step 2.2.1 may have an effect only if the current cost $f(P^*(t))$ of t is strictly higher than $f(P^*(s))$. Therefore, we can avoid computing the cost $f(P^*(s) \cdot \langle s,t \rangle)$ whenever $C(t) \leq C(s)$.

Handling of infinite costs. In many applications, the cost $f(P^*(t))$ is $+\infty$ for most pixels $t \in \mathcal{Q}$, through most of the execution of Algorithm 1. In such cases, we can save a large fraction of the running time by storing in $\mathcal{Q}$ only those pixels $t \in \mathcal{I} \setminus \mathcal{F}$ which have $f(P^*(t)) < +\infty$. In particular, if $f = f^{\mathcal{S}}$ for some set $\mathcal{S}$, then step 1 should initialize $\mathcal{Q}$ with $\mathcal{S}$ rather than $\mathcal{I}$. With this modification, any pixel t that is not reachable by any finite-cost path (in particular, any pixel which is not reachable from $\mathcal{S}$) will remain a single-node tree.

Infinite costs are also useful when we are only interested in paths whose costs do not exceed a specified threshold $C_{\max}$. In those cases, we can modify f by mapping any cost greater than $C_{\max}$ to $+\infty$. Note that the modified algorithm will terminate after enumerating all pixels t with finite cost $\hat{f}(t)$.

Efficient pixel queue structure. Asymptotically, the bottleneck of Algorithm 1 lies in step 2.1, the selection of the minimum-cost pixel $s \in \mathcal{Q}$. If $\mathcal{Q}$ is implemented as a balanced heap data structure, the total running time will be $O(m + n \log n)$.

In most applications, the path costs $f(\pi)$ are either $+\infty$ or integers in some range $[C_{\min} .. C_{\max}]$. In such cases, we can implement $\mathcal{Q}$ as a vector of C_Δ buckets, each pointing to a circular doubly linked list of pixels, where $C_\Delta = C_{\max} - C_{\min} + 1$. The insertion, deletion, and cost update of a pixel can then be done in $O(1)$ time. Finding the next minimum-cost pixel in $\mathcal{Q}$ may require skipping over several empty buckets; however, since the cost of the next pixel never decreases, the total time for step 2.1 is $O(C_\Delta)$. Algorithm 1 will then run in $O(m + n + C_\Delta)$ time and $O(n + C_\Delta)$ storage. Note that each pixel may appear in the queue at most once, and so we can avoid dynamic allocation by preallocating two links next(p) and prev(p) for each pixel p, which are then used to connect the pixels belonging to each bucket [18].

Moreover, in many of those applications, there exists a fairly small upper bound K to the incremental cost $f(\pi \cdot \langle s,t \rangle) - f(\pi)$ of extending an optimum path π by an arc $(s,t) \in \mathcal{A}$, and to the maximum difference $f(\langle t \rangle) - f(\langle t' \rangle)$ between the costs of trivial paths (excluding infinite values). Invariant (H3) then implies that, for all $t \in \mathcal{Q}$, the cost $f(P^*(t))$ is either $+\infty$ or an integer in the range $[C .. C+K]$, for some cost C that varies during the algorithm.

The bucket vector can then be replaced by a circular queue of $K + 1$ entries, reducing the storage cost to $O(n + K)$ [4, 5].

Furthermore, C_Δ will be bounded by K for $f_{\max}$, and by nK for general smooth cost functions, in the worst case. (In practice, an optimum path is unlikely to traverse a significant fraction of the image pixels, so its cost is usually bounded by a few times $K\sqrt{n}$.) Therefore, the algorithm will run in $O(m + nK)$ time. As noted by Ahuja et al. [5] there are more complicated data structures that achieve asymptotic cost $O(m + n\sqrt{\log K})$.

Propagating the root labels. Although not needed by the algorithm itself, many applications (and some path-cost functions) need to know the root pixel $P^0(t)$ associated with each pixel t — or, more generally, some attribute $\lambda(P^0(t))$ that was assigned to it. For example, in segmentation applications, one would assign the same label to all seed pixels that belong to the same target object, and treat the corresponding trees as a single region.

The root labels $\lambda(P^0(t))$ can be computed in linear time from the predecessor map P, for all $t \in \mathcal{I}$, as a post-processing step; but it is more convenient to compute them during Algorithm 1, as a *root label map* $L(t)$.

Optimized Algorithm. All these improvements are implemented in Algorithm 2 below:

Algorithm 2 Input*: An image* $\mathbf{I} = (\mathcal{I}, I)$*; an adjacency relation* $\mathcal{A} \subset \mathcal{I} \times \mathcal{I}$*; a labeling function* λ *defined on* $\mathcal{I}$*; and a smooth path-cost function* f*.* Output*: An optimum-path forest* P*, and the corresponding cost map* C *and root label map* L*.* Auxiliary Data Structures*: A priority queue* $\mathcal{Q}$*.*

1. For all $t \in \mathcal{I}$, set $P(t) \leftarrow nil$, $L(t) \leftarrow \lambda(t)$, and $C(t) \leftarrow f(\langle t \rangle)$.
   If $C(t) < +\infty$, insert t in $\mathcal{Q}$.
2. While $\mathcal{Q}$ is not empty, do
   2.1. Remove from $\mathcal{Q}$ a pixel s such that $C(s)$ is minimum.
   2.2. For each pixel t such that $(s, t) \in \mathcal{A}$ and $C(t) > C(s)$, do
      2.2.1. Compute $C' = f(P^*(s) \cdot \langle s, t \rangle)$. If $C' < C(t)$, then
         2.2.1.1. If $C(t) \neq +\infty$, remove t from $\mathcal{Q}$.
         In any case, set $P(t) \leftarrow s$, $C(t) \leftarrow C'$, and $L(t) \leftarrow L(s)$. Insert t in $\mathcal{Q}$.

5 Tie-breaking

As observed in Section 3.5, the optimum-path forest is not always unique. For example, if all paths have the same cost, then any spanning forest will be optimum. Since path costs are usually discrete, multiple solutions are actually quite common in practice.

Algorithms 1 and 2 already resolve some of those ambiguities. For instance, when a pixel t is reached by two optimum paths of equal cost, the algorithms will set $P^*(t)$ to the path that is found first. The only remaining ambiguity is the choice of the minimum-cost

pixel s in $\mathcal{Q}$, in case of ties. Usually, the most convenient choice is to pick the pixel that entered in $\mathcal{Q}$ first, i.e. ties are broken by using the *first-in first-out* (FIFO) policy.

In some cases, however, a *last-in first-out* (LIFO) policy may be more adequate. For consistency, in this case, the algorithm should also choose always the *last* optimum path found in case of ties. The optimized version of this algorithm is given below.

Algorithm 3 Input*: An image* $\mathbf{I} = (\mathcal{I}, I)$*; an adjacency relation* $\mathcal{A} \subset \mathcal{I} \times \mathcal{I}$*; a labeling function* λ *defined on* $\mathcal{I}$*; and a smooth path-cost function* f*.* Output*: An optimum-path forest* P*, and the corresponding cost map* C *and root label map* L*.* Auxiliary Data Structures*: A priority queue* $\mathcal{Q}$*, with LIFO tie-breaking.*

1. For all pixels $t \in \mathcal{I}$, set $P(t) \leftarrow nil$, $L(t) \leftarrow \lambda(t)$, $C(t) \leftarrow f(\langle t \rangle)$, and insert t in $\mathcal{Q}$.
2. While $\mathcal{Q}$ is not empty, do
   2.1. Remove from $\mathcal{Q}$ a pixel s such that $C(s)$ is minimum.
   2.2. For each pixel t such that $(s,t) \in \mathcal{A}$ and $t \in \mathcal{Q}$, do
      2.2.1. Compute $C' = f(P^*(s) \cdot \langle s,t \rangle)$.
      2.2.2. If $C' \leq C(t)$, then remove t from $\mathcal{Q}$,
      set $P(t) \leftarrow s$, $C(t) \leftarrow C'$, $L(t) \leftarrow L(s)$, and insert t in $\mathcal{Q}$.

The main difference between Algorithms 2 and 3, besides the queue policy, lies in step 2.2.2— where $P(t)$ and $L(t)$ must be updated, and t must be removed and reinserted in $\mathcal{Q}$, even when C' is equal to $C(t)$.

With the FIFO policy, any connected set of pixels that could be reached from two or more roots, at the same cost, will tend to get partitioned among the respective trees (see Figure 4a). With the LIFO policy, in contrast, those pixels will all get assigned to a single tree (see Figure 4b). The LIFO policy may be an advantage in some applications (see Section 6.1); in most cases, however, the shorter paths and more uniform trees produced by the FIFO policy are a better match to the users' expectations.

It must be noted, however, that the FIFO policy by itself does not ensure an unbiased partition. In the extreme case where any path from any given seed has the same cost, one might expect that the boundary between adjacent trees would follow the bisector of their roots, in the graph-theoretic metric (Figure 4a). In fact, one of the trees usually encroaches into the other, so the boundary may deviate quite sharply from the bisector (see Figure 4c). Thus, when the partition of ambiguous pixels is important, one must incorporate an appropriate tie-breaking criterion explicitly into the path-cost function.

6 Applications

In this section we illustrate a few selected applications of the IFT. Even though the examples given here use gray-scale images, many of these operators can be trivially applied to multispectral (color) images, by changing the path-cost function to use all bands.

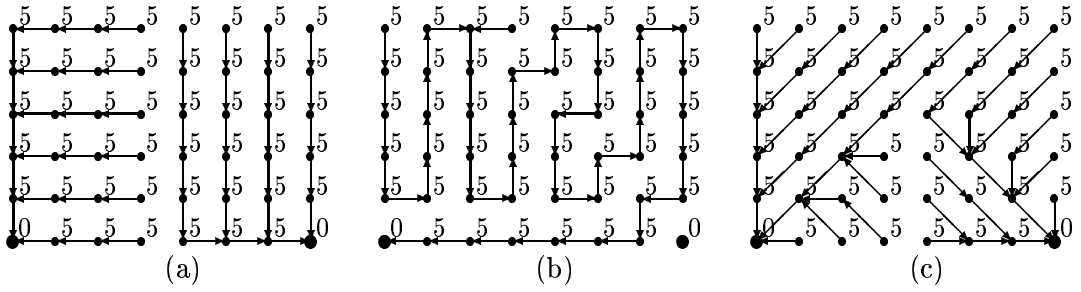

Figure 4: Three examples of tie-breaking for the same function of Figure 2, restricted to the two seed pixels represented by bigger dots. (a) FIFO policy and 4-connected adjacency. (b) LIFO policy and 4-connected adjacency. (c) FIFO policy and 8-connected adjacency.

6.1 Regional minima

A *regional minimum* is a maximal connected set $\mathcal{X} \subseteq \mathcal{I}$, such that $I(s) \leq I(t)$ for any $s \in \mathcal{X}$ and any arc $(s,t) \in \mathcal{A}$ [23]. We can compute the regional minima with the cost function

$$\begin{aligned} f_{\text{ini}}(\langle t \rangle) &= I(t), \text{ for all } t \in \mathcal{I}, \\ f_{\text{ini}}(\pi \cdot \langle s,t \rangle) &= \begin{cases} f_{\text{ini}}(\pi), & \text{if } I(s) \leq I(t), \\ +\infty, & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

The function f_{ini} is MI, and therefore smooth. Note that any optimum path starts at a regional minimum and never goes downhill. Since the regional minima are contained in the set $\mathcal{S}$ of pixels s such that $I(s) \leq I(t)$ for all $(s,t) \in \mathcal{A}$, we can use $f_{\text{ini}}^{\mathcal{S}}$ instead of f_{ini}.

With FIFO tie-breaking, the roots of the IFT will be the union of all regional minima. With LIFO tie-breaking, in contrast, we will get *exactly* one root r in each regional minimum. In this case, the full extent of each regional minimum is obtained by truncating the tree rooted at r to those pixels t with $I(t) = I(r)$.

6.2 The watershed transform and morphological reconstruction

Consider image $\mathbf{I} = \{\mathcal{I}, I\}$ as a 3D surface, where the altitude of each pixel is its gray value and the objects of interest are delimited by ridges which are higher than any other hills inside or outside the objects. This situation arises, for example, when $\mathbf{I}$ is some kind of gradient of the original image, where the strongest discontinuities occur at the object boundaries. In this case, we can separate the objects by choosing one or more seed pixels in each object (including the background), assigning a distinct label to all seeds within each object, and using the cost function f_{peak}, defined as the maximum pixel intensity along the path. This is a special case of $f_{\max}$, where $w(s,t) = I(t)$. The label map $L(t)$ will give the result of the segmentation (*catchment basins*).

This formulation captures the essential features of the *watershed transform* and several of its variants [6, 12, 20, 21]. There is no unique and precise definition for a watershed transform in the literature [41]. It is informally described as a flooding process on the

image surface, with a source of water at each seed; a barrier (*watershed line*) is erected wherever two bodies of water coming from distinct sources meet. However, the position of the watershed lines is not precisely defined in many situations, such as on plateaus and when two or more bodies of water overflow into an adjacent basin at the same time. In the IFT approach, the position of the barrier is determined by the tie-breaking policy. The FIFO policy usually leads to a fair (if not symmetrical) partition of plateaus and flooded basins among competing sources (see Figures 4a and c).

By restricting f_{peak} to a seed set $\mathcal{S} \subset \mathcal{I}$, the IFT computes a *watershed-from-markers* transform (WMT) (see Figures 5a-c). If we set $h(t) > I(t)$ for all $t \in \mathcal{S}$, we get the WMT without marker imposition [21]— that is, some seeds may not become roots of the optimum-path forest [12]. Marker imposition is achieved by assigning $h(t) = 0$ (or $h(t) = I(t)$) for all $t \in \mathcal{S}$ [6]; this solution is more efficient than the standard one based on homotopy change [21].

The *threshold decomposition* of an image $\mathbf{I}$ is the set $\mathcal{X}(\mathbf{I})$ of all connected components of pixel subsets of the form $\{ t \in \mathcal{I} : I(t) \geq v \}$, for all $v \in \mathcal{V}$ [12, 23]. *Morphological reconstruction* operators are robust filters whose only effect is to eliminate or merge connected components from $\mathcal{X}(\mathbf{I})$. If we use f_{peak} with $h(t) > I(t)$ for all $t \in \mathcal{I}$, then $C(t)$ will be the *superior morphological reconstruction* of $\mathbf{I}$, which only merges components selected by the regional minima of the *marker function* $h(t)$; and $L(t)$ will be the catchment basins of the watershed transform of the reconstructed image $C(t)$ [12]. The *inferior reconstruction* is the dual operation, which only eliminates components from $\mathcal{X}(\mathbf{I})$. (See Figures 5d-f.)

The *local superior reconstruction* is a variant introduced by Falcão et al. [12] which fills up one or more basins, selected by a seed set $\mathcal{S}$, up to levels specified by given handicaps $h(t) > I(t)$ for $t \in \mathcal{S}$. The parameters $\mathcal{S}$ and $h(t)$ can be given by the user or selected automatically, as in Vincent's area closing operator [22]. The required cost function f_{lrec} is defined as:

$$\begin{aligned} f_{\text{lrec}}(\langle t \rangle) &= h(t), \text{ if } t \in \mathcal{S}, \text{ and } +\infty \text{ otherwise,} \\ f_{\text{lrec}}(\pi \cdot \langle s, t \rangle) &= \begin{cases} f_{\text{lrec}}(\pi), & \text{if } f_{\text{lrec}}(\pi) > I(t), \\ +\infty, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Morphological reconstruction is the building block of many other image operations, such as *h-minima*, *h-maxima*, the *leveling operator*, *area opening* and *closing*, *alternate sequential filters by reconstruction*, *hole closing* and *dome removal* [12, 22–24].

A peculiar property of f_{peak} (and f_{lrec}) is that the cost of a pixel never needs to be updated once it has been inserted in $\mathcal{Q}$ [6]. Classical implementations of the watershed transform use this property to simplify the queue structure [20, 21]. Note that some generalizations of watershed may not have this property.

6.3 Boundary tracking

In the *boundary tracking* approach to image segmentation, the goal is to find an optimum curve that is constrained to pass through a given sequence $\langle \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k \rangle$ of k landmarks (pixel sets) on the object's boundary, in that order, starting in $\mathcal{T}_1$ and ending in $\mathcal{T}_k$. (For instance, each set $\mathcal{T}_i$ could be a short stroke drawn by the user across the presumed object

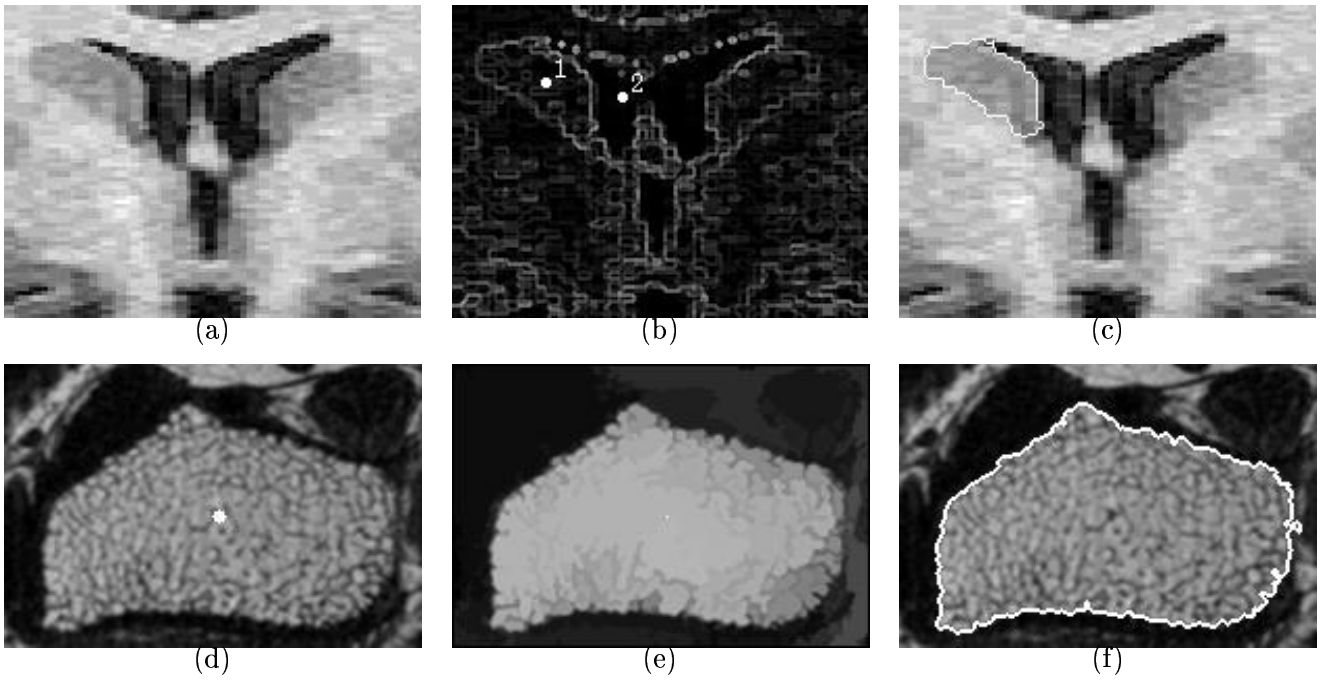

Figure 5: Examples of segmentation using IFT-based watershed transform and morphological reconstruction. (a) A magnetic ressonance (MR) image of a brain. (b) The gradient of image I' obtained from (a) by $I'(t) = 255\exp[(I(t) - 140)^2/(2 \times 30^2)]$, with a seed selected inside the left caudate nucleus (1), and another outside it (2). (c) The segmentation derived from the label map $L(t)$ of the IFT-watershed. (d) An MR image of a wrist with one seed pixel selected inside the bone. (e) The cost map $C(t)$ resulting from inferior morphological reconstruction of (d) with the given seed, followed by superior reconstruction with the image border pixels as seeds. (f) The bone is segmented by thresholding of (e).

boundary.) In particular, if $\mathcal{T}_1 = \mathcal{T}_k$ is a single pixel, then the result will be an optimum *closed* path that satisfies those restrictions (see Figure 6).

If the path-cost function f is MI, then the optimum path that satisfies those constraints consists of $k-1$ segments π_i, $i = 1, 2, \ldots, k-1$, where each segment π_i is a f-optimum path connecting $\mathcal{T}_i$ to $\mathcal{T}_{i+1}$. Therefore, we can solve this problem by $k-1$ executions of the IFT. Each execution i uses $f^{\mathcal{T}_i}$ as the cost function; except that, for $i > 1$, the handicap $h(t)$ is set to the cost $C_i(t)$, computed in the previous execution, if $t \in \mathcal{T}_i$, or $+\infty$ otherwise. Each stage can be terminated as soon as the last pixel of the target set $\mathcal{T}_{i+1}$ is removed from the queue $\mathcal{Q}$. The optimum path can be obtained from the predecessor maps $P_i(t)$, $i = k-1, k-2, \ldots, 1$.

The function f should favor paths that go through boundary-like regions — for example, regions of high gradient, for objects with sharp high-contrast edges. The f_{sum} cost function is usually preferable to $f_{\max}$, because it favors shortest-distance jumps across regions where the boundary is not well defined.

We can also devise path costs that favor a specific orientation (clockwise or counterclock-

wise) for the boundary. Such a feature is useful to avoid nearby boundaries with similar contrast but opposite orientation [17]. For example, if the object is darker than the background, we can use $w(s,t) = K - \max\{G(s,t) \cdot \eta(s,t), 0\}$, where $G(s,t)$ is a gradient-like vector estimated at the midpoint of arc (s,t); $\eta(s,t)$ is the arc (s,t) rotated 90 degrees counter-clockwise; and K is an upper bound for $|G(s,t) \cdot \eta(s,t)|$.

Boundary tracking is usually formulated in terms of heuristic graph search [32, 33, 46] or dynamic programming [31, 46]. The IFT approach covers both formulations and several interesting interactive extensions— such as *live-wire* [17–19, 35] and *live-lane* [17].

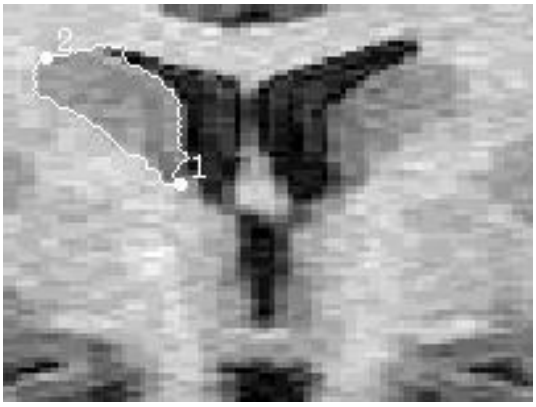

Figure 6: Boundary tracking of the left caudate nucleus in an MR image of a brain, with single-pixel landmarks $(1, 2, 1)$. The cost function is f_{sum} with the directional arc weights $w(s,t)$ described in Section 6.3, where $G(s,t)$ is the gradient of image I' used in Figure 5.

6.4 Euclidean distance transform and related operators

The *Euclidean distance transform* (EDT) of a given seed set $\mathcal{S}$ assigns to each image pixel t a value $C(t)$ which is the minimum Euclidean distance from t to $\mathcal{S}$. (Actually, in order to avoid irrational numbers, the square of the distance is stored instead.)

The EDT is related to the concept of *discrete Voronoi diagram* (DVD), which is a partition of the image pixels into *discrete Voronoi regions*. For any non-empty subset $\mathcal{X}$ of $\mathcal{S}$, we define the discrete Voronoi region of $\mathcal{X}$ as the set $\mathcal{R}(\mathcal{X})$ of all image pixels t such that t is equidistant from all the seeds in $\mathcal{X}$, and is strictly closer to those seeds than to any other seed. We also define the *extended discrete Voronoi region* $\mathcal{R}^*(\mathcal{X}) = \bigcup\{\, \mathcal{R}(\mathcal{Y}) : \mathcal{Y} \subseteq \mathcal{X} \,\}$. In particular, for any seed $r \in \mathcal{S}$, $\mathcal{R}(\{r\})$ consists of all image pixels whose centers lie in the interior of the exact (geometric) Voronoi polygon of $\mathcal{S}$ [34]; whereas $\mathcal{R}^*(\{r\})$ consists of the pixels inside or on the boundary of that polygon.

The obvious implementation of the EDT (or DVD) runs in time proportional to $n\,|\mathcal{S}|$. This large cost motivated a search for faster propagation-style algorithms. Ragnemalm [39] proposed an algorithm for an approximate EDT, similar to Algorithm 2 with 8-connected adjacency and the Euclidean path-cost function $f^{\mathcal{S}}_{\text{euc}}(\pi)$ defined in Section 4.3. When $|\mathcal{S}| \geq 3$, this algorithm may not output the exact EDT, because the regions of the discrete Voronoi

diagram may not be 8-connected, as observed by Danielsson [47]. (Indeed, the path-cost function $f^{\mathcal{S}}_{\text{euc}}$ is not smooth in this case.) On the other hand, Cuisenaire and Macq [40] observed that the exact EDT *can* be computed by Algorithm 2 if the adjacency radius is large enough. Lemma 7 below captures this idea:

Lemma 7 *For any image diameter D, there is a radius $\rho \geq \sqrt{2}$ such that any pixel $t \notin \mathcal{S}$ can be reached by a path $\pi = \tau \cdot \langle s,t \rangle$ that satisfies* (E1) *$r = \text{org}(\pi)$ is one of the seeds closest to t in the Euclidean distance,* (E2) *all pixels along τ belong to $\mathcal{R}(\{r\})$,* (E3) *all arcs in π are directed stricly away from r,* (E4) *τ is 8-connected, and* (E5) *$d(s,t) \leq \rho$.*

The lemma is proved by taking $\rho \geq \rho^*(D)$, the maximum Hausdorff distance between $\mathcal{R}^*(\{r\})$ and the 8-connected component of $\mathcal{R}(\{r\})$ that contains r, for any $r \in \mathcal{S}$. According to Cuisenaire and Macq [40], $\rho^*(D) \ll D$ (e.g. $\rho^* = 16$ for a 768×768 image).

Lemma 8 *If $\mathcal{A}$ is the Euclidean adjacency with radius $\rho \geq \rho^*(D)$, then $f^{\mathcal{S}}_{\text{euc}}$ is smooth.*

Proof: For each pixel t, if $t \in \mathcal{S}$ then the path $\pi = \langle t \rangle$ is optimum for $f^{\mathcal{S}}_{\text{euc}}$ and satisfies the smoothness condition trivially. If $t \notin \mathcal{S}$, let $\pi = \tau \cdot \langle s,t \rangle$ be a path from some seed r to t that satisfies the conditions (E1)–(E5) of Lemma 7. Because of conditions (E4) and (E5), the path π is $\mathcal{A}$-connected. Because of condition (E1), π is optimum for $f^{\mathcal{S}}_{\text{euc}}$. Because of condition (E2), s lies in $\mathcal{R}(\{r\})$ and therefore τ also is optimum for $f^{\mathcal{S}}_{\text{euc}}$, thus satisfying (C2). Because of condition (E3), $d(r,s) \leq d(r,t)$, and therefore $f^{\mathcal{S}}_{\text{euc}}(\tau) \leq f^{\mathcal{S}}_{\text{euc}}(\pi)$, satisfying (C1). Finally, let τ' be any optimum path leading to s; since $s \in \mathcal{R}(\{r\})$, we must have $\text{org}(\tau') = r$, and therefore $f^{\mathcal{S}}_{\text{euc}}(\tau' \cdot \langle s,t \rangle) = d(r,t) = f^{\mathcal{S}}_{\text{euc}}(\pi)$, which is (C3). ■

Cuisenaire and Macq also pointed out that one can correct Ragnemalm's approximate EDT by a post-processing stage which considers all arcs whose Euclidean length is less than or equal to $\rho^*(D)$, and whose origin lies in a certain subset of the leaf pixels of the computed influence zones. Note, however, that propagation methods will hardly compete with other approaches based on the exact geometric Voronoi diagram, which can compute the exact EDT in linear time [48]. The algorithm by Cusinaire and Macq will require $\Omega(n\,|\mathcal{A}|) = \Omega(n(\rho^*(\sqrt{n}))^2)$ time in the worst case, since the image diameter D is at least $\Omega(\sqrt{n})$, and it is possible that $\Omega(n)$ pixels will become seeds of the second stage. The asymptotic growth of $\rho^*(D)$ is not known, but from the table in reference [40] it seems to be at least $\Omega(\sqrt{D})$, which would imply total cost $\Omega(n^{3/2})$.

On the other hand, propagation algorithms seem mandatory in applications where it is more important to have 8-connected influence zones than exact distance values [14–16]. Examples include the computation of one-pixel-wide and connected multiscale skeletons (see Figure 7), and the skeleton by influence zones (SKIZ) [14]. The former are used by Torres et al. to compute the salience points of a given contour [16].

In such applications, one could run Algorithm 2 with cost function $f^{\mathcal{S}}_{\text{euc}}$ and $\rho = \sqrt{2}$ (8-connected adjacency). Because of Lemma 2, the result will always be *some* partition of the image into 8-connected influence zones, even though the forest may not be optimum. However, because of condition (E4) of Lemma 7, it can be shown that the influence zone of each seed $r \in \mathcal{S}$ will contain the subset of $\mathcal{R}(\{r\})$ which is 8-connected to r. So, in this

sense, this "quasi-IFT" is an approximation of the EDT/DVD, which may be good enough for many applications.

Another alternative is to compute a *chamfer distance transform* (CDT), where the Euclidean distance is replaced by a piecewise-linear approximation. One can obtain an exact CDT by applying Algorithm 2 with the f_{sum} cost function and arc weights $w(s,t)$ that depend only on the vector $t-s$ [36]. For most applications it suffices to use 8-connected adjacency with weights 5 for axial arcs and 7 for diagonal arcs [49]. One drawback of the CDT is that the resulting skeletons are more affected by rotations than the quasi-EDT above.

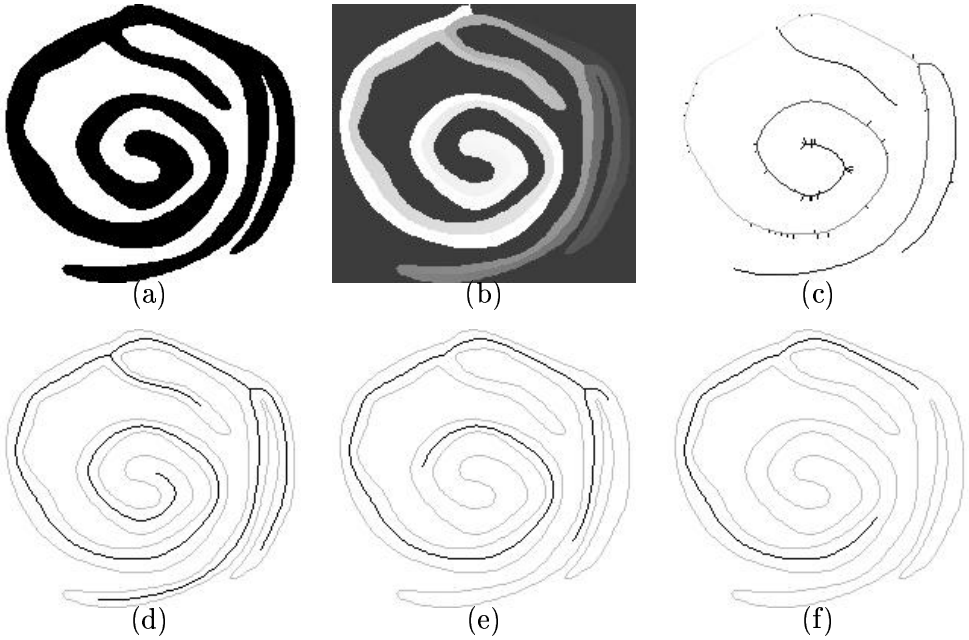

Figure 7: (a) A binary image. (b) The label map $L(t)$ resulting from the quasi-IFT with $f^{\mathcal{S}}_{\text{euc}}$, where the seeds are the boundary pixels with labels $\lambda(t)$ that increase sequentially along the contour. (c) A multiscale skeleton computed from (b) by local differencing. (d-f) Skeletons computed by thresholding (c) at three different spatial scales.

7 Conclusions and current research

We gave a precise definition of the image foresting transform, a proof of correctness of its basic algorithm (for a fairly general class of cost functions), and an efficient implementation (Algorithm 2). We pointed out the importance of the tie-breaking policy, especially for the watershed transform, and introduced the LIFO variant (Algorithm 3), which led to an efficient way of locating regional minima. We also presented cost functions for watershed transforms, morphological reconstructions, boundary tracking, and EDT-related operators.

The IFT provides a common and well-founded framework for the design of image processing operators, which leads to correct, efficient, and flexible implementations. Algorithms 2

and 3 together with the examples of this paper are available through the internet [50].

We are currently exploiting Algorithm 2 and variants to improve the efficiency of watershed transforms [25]. We are also developing novel IFT-based algorithms for automatic image segmentation [51], and paradigms for 3D segmentation of medical images at interactive speeds, such as the *differential IFT* [26, 52]. Finally, we are investigating parallel and hardware implementations of the IFT, and its possible application to 3D multiscale skeletonization.

Acknowledgments

We thank the MIPG, University of Pennsylvania, and the Dept. of Neurology, University of Campinas for the medical images used in the examples. This work was partially support by CNPq (Procs. 302966/02-1 and 301016/92-5).

A Proofs of the lemmas

By induction on the length of π, conditions (C1)–(C3) can be extended to arbitrary prefixes of optimum paths. Namely, if f is a smooth path-cost function, then for any $t \in \mathcal{I}$ there is an optimum path π to t, such that, for any τ and δ such that $\pi = \tau \cdot \delta$

(C1*) $f(\tau) \leq f(\pi)$,

(C2*) τ is optimum,

(C3*) for any optimum path τ' with same terminus as τ, $f(\tau' \cdot \delta) = f(\pi)$.

Lemma 3 *Each execution of step 2.2.1 preserves invariants* H1–H3.

Proof: In this proof, for any variable or preposition V, we will denote by $\triangleleft V$ the value of V *before* a generic execution of step 2.2.1, and by $\triangleright V$ its value *after* it. Thus, the effect of step 2.2.1 is to change the predecessor map from $\triangleleft P$ into $\triangleright P$; and we must show that, as a consequence, $\triangleleft$(H1 $\wedge$ H2 $\wedge$ H3) implies $\triangleright$(H1 $\wedge$ H2 $\wedge$ H3).

By definition, step 2.2.1 adds the arc (s,t) to $\mathcal{B}$ (i.e., $\triangleright\mathcal{B} = \triangleleft\mathcal{B} \cup \{(s,t)\}$). That step may change the predecessor map P for pixel t, but will have no effect on $P(r)$, for any pixel $r \neq t$. Since step 2.2.1 has no effect on $\mathcal{Q}$ or $\mathcal{F}$, or on $P^*(s')$ for any $s' \in \mathcal{F}$, invariants (H1) and (H2) are trivially preserved.

Now observe that, for any r and at any point in the algorithm, $P(r)$ is either *nil* or a pixel of $\mathcal{F}$; so $P^*(r)$ is always in $\mathcal{F} \cup \{r\}$. Since t is in $\mathcal{Q}$, it follows that $\triangleright P^*(r) = \triangleleft P^*(r)$ for any $r \neq t$. Moreover, step 2.2.1 only changes $P(t)$ if $f(\triangleright P^*(t)) < f(\triangleleft P^*(t))$. We conclude that $f(\triangleright P^*(t')) \leq f(\triangleleft P^*(t'))$, for any $t' \in \mathcal{Q}$.

Now suppose that (H3) is violated by step 2.2.1; that is, there exists some pixel $t' \in \mathcal{Q}$ and some $(\triangleright\mathcal{B}, \triangleright P)$-path π ending at t' with $f(\pi) < f(\triangleright P^*(t'))$. By the reasoning above, we should have also $f(\pi) < f(\triangleleft P^*(t'))$. On the other hand, given $\triangleleft$H3, the path π cannot be a $(\triangleleft\mathcal{B}, \triangleleft P)$-path. However, since $\triangleright P^*(t') = \triangleleft P^*(t')$ for all $t' \neq t$, it follows that π must use the arc (s,t), which is the only difference between $\triangleleft\mathcal{B}$ and $\triangleright\mathcal{B}$. Then π must be

the same path $P^*(s) \cdot \langle s, t\rangle$ considered in step 2.2.1. However, that step guarantees that $f(\pi) \geq f(\triangleright\, P^*(t'))$, a contradiction. We conclude that invariant (H3) is preserved, too. ■

Lemma 5 *If f is a smooth path-cost function, step 2.1 preserves invariants* H1–H3.

Proof: In this proof, we will denote by $\triangleleft V$ the value of V *before* a generic execution of step 2.1, and by $\triangleright V$ its value *after* it. Again, we must show that $\triangleleft(\text{H1} \wedge \text{H2} \wedge \text{H3})$ implies $\triangleright(\text{H1} \wedge \text{H2} \wedge \text{H3})$.

The effect of step 2.1 is only to set $\triangleright\,\mathcal{F} = \triangleleft\,\mathcal{F} \cup \{s\}$, $\triangleright\,\mathcal{Q} = \triangleleft\,\mathcal{Q} \setminus \{s\}$. The step has no effect on the set $\mathcal{B}$, the forest P, or the paths P^*. Therefore, to show that invariant (H1) is preserved, we must show only that $P^*(s)$ is optimum.

Let π be an optimum path that ends at s. Since f is a smooth path-cost function, we can assume, without loss of generality, that π satisfies conditions (C1*)–(C3*).

Let's consider first the case when $\text{org}(\pi) = r$ is in $\triangleleft\,\mathcal{Q}$. In that case, by the choice of s, we must have $f(P^*(s)) \leq f(P^*(r))$. Moreover, $\langle r\rangle$ is a $(\triangleleft\,\mathcal{B}, P)$-path; by invariant $\triangleleft$(H3), it follows that $f(P^*(r)) \leq f(\langle r\rangle)$. On the other hand, by condition (C2*), the path $\langle r\rangle$ is optimum; by condition (C1*), we must have $f(\langle r\rangle) \leq f(\pi)$. It follows that $f(P^*(s)) \leq f(\pi)$, meaning that $P^*(s)$ is optimum.

Let's now consider the case where $\text{org}(\pi) \in \triangleleft\,\mathcal{F}$. Let τ be the longest prefix of π that is entirely contained in $\triangleleft\,\mathcal{F}$; let (r', s') be the following arc, and δ the remainder of π, from s' to s. By invariant (H1), the path $\tau' = P^*(r')$ is optimum; by condition (C3*), the path $\tau' \cdot \langle r', s'\rangle \cdot \delta$ is also optimum. Also, by condition (C1*), we must have

$$f(\tau' \cdot \langle r', s'\rangle) \leq f(\pi). \tag{6}$$

Now, since $r' \in \triangleleft\,\mathcal{F}$ and $s' \in \triangleleft\,\mathcal{Q}$, the arc (r', s') must have been added to $\triangleleft\,\mathcal{B}$ in some previous execution of step 2.2 (right after r' was removed from $\mathcal{Q}$). It follows that $\tau' \cdot \langle r', s'\rangle$ is a $(\triangleleft\,\mathcal{B}, P)$-path; and, by invariant $\triangleleft$(H3),

$$f(P^*(s')) \leq f(\tau' \cdot \langle r', s'\rangle). \tag{7}$$

Furthermore, by choice of s, we have $f(P^*(s)) \leq f(P^*(s'))$. Combining this fact with equations (6) and (7), we conclude that $f(P^*(s)) \leq f(\pi)$, meaning that $f(P^*(s))$ is indeed optimum. ■

References

[1] E.F. Moore. The shortest path through a maze. In *Proc. Intl. Symp. on the Theory of Switching*, pages 285–292. Harvard University, Apr 1959.

[2] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[3] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[4] R.B. Dial. Shortest-path forest with topological ordering. *Comm. of the ACM*, 12(11):632–633, Nov 1969.

[5] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[6] R.A. Lotufo and A.X. Falcão. The ordered queue and the optimality of the watershed approaches. In *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, pages 341–350. Kluwer, Jun 2000.

[7] R.A. Lotufo, A.X. Falcão, and F. Zampirolli. IFT-Watershed from gray-scale marker. In *Proc. of XV Brazilian Symp. on Computer Graphics and Image Processing*, pages 146–152. IEEE, Oct 2002.

[8] J.K. Udupa and S. Samarasekera. Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *Graphical Models and Image Processing*, 58:246–261, 1996.

[9] P.K. Saha and J.K. Udupa. Relative fuzzy connectedness among multiple objects: theory, algorithms, and applications in image segmentation. *Computer Vision and Image Understanding*, 82:42–56, 2001.

[10] L.G. Nyúl, A.X. Falcão, and J.K. Udupa. Fuzzy-connected 3D image segmentation at interactive speeds. *Graphical Models*, 64(5):259–281, 2003.

[11] B.S. Cunha. Projeto de operadores de processamento e análise de imagens baseados na transformada imagem-floresta. Master's thesis, Universidade Estadual de Campinas, Instituto de Computação, Ago 2001.

[12] A.X. Falcão, B. S. da Cunha, and R. A. Lotufo. Design of connected operators using the image foresting transform. In *Proc. of SPIE on Medical Imaging*, volume 4322, pages 468–479, Feb 2001.

[13] R.A. Lotufo, A.X. Falcão, and F.A. Zampirolli. Fast Euclidean distance transform using a graph-search algorithm. In *Proc. of the XIII Brazilian Symp. on Computer Graphics and Image Processing*, pages 269–275. IEEE, Oct 2000.

[14] A.X. Falcão, L.F. Costa, and B.S. da Cunha. Multiscale skeletons by image foresting transform and its applications to neuromorphometry. *Pattern Recognition*, 35(7):1571–1582, Apr 2002.

[15] R.S. Torres, A.X. Falcão, and L.F. Costa. Shape description by image foresting transform. In *14th Intl. Conf. on Digital Signal Processing*, pages 1089–1092, Santorini, Greece, Jul 2002.

[16] R.S. Torres, A.X. Falcão, and L.F. Costa. A graph-based approach for multiscale shape analysis. Technical Report IC-0303, Institute of Computing, University of Campinas, Jan 2003.

[17] A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsch, and R.A. Lotufo. User-steered image segmentation paradigms: Live-wire and live-lane. *Graphical Models and Image Processing*, 60(4):233–260, Jul 1998.

[18] A.X. Falcão, J.K. Udupa, and F.K. Miyazawa. An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly. *IEEE Trans. on Medical Imaging*, 19(1):55–62, Jan 2000.

[19] A.X. Falcão and J.K. Udupa. A 3D generalization of user-steered live wire segmentation. *Medical Imaging Analysis*, 4(4):389–402, Dec 2000.

[20] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(6), Jun 1991.

[21] S. Beucher and F. Meyer. The morphological approach to segmentation: The watershed transformation. In *Mathematical Morphology in Image Processing*, chapter 12, pages 433–481. Marcel Dekker, 1993.

[22] L. Vincent. Morphological area opening and closings for greyscale images. In *Shape in Picture'92 - NATO Workshop*. Springer, Sep 1992.

[23] L. Vincent. Morphological grayscale reconstruction in image analysis. *IEEE Trans. on Image Processing*, 2(2):176–201, Apr 1993.

[24] F. Meyer. The levelings. In *4th Intl. Symp. on Mathematical Morphology*, pages 190–207. Kluwer, 1998.

[25] R.A. Lotufo, A.X. Falcão, and F. Zampirolli. IFT-Watershed from gray-scale marker. Technical Report IC-0212, Institute of Computing, University of Campinas, Dec 2002.

[26] A.X. Falcão and F.P.G. Bergo. The iterative image foresting transform and its application to user-steered 3D segmentation. In *Proc. of SPIE on Medical Imaging*, volume 5032, Feb 2003. to appear.

[27] N. Deo and C. Pang. Shortest-path algorithms: Taxonomy and annotation. *Networks*, 14:275–323, 1984.

[28] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT, 1990.

[29] M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. of Automata, Languages, and Combinatorics*, 7(3):321–350, 2002.

[30] A. Frieze. Minimum paths in directed graphs. *Operational Research Quarterly*, 28(2,i):339–346, 1977.

[31] U. Montanari. On the optimal detection of curves in noisy pictures. *Comm. of the ACM*, 14(5):335–345, 1971.

[32] A. Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1:169–182, 1972.

[33] A. Martelli. An application of heuristic search methods to edge and contour detection. *Comm. of the ACM*, 19(2):73–83, 1976.

[34] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer, 1985.

[35] E.N. Mortensen and W.A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60:349–384, 1998.

[36] B.J.H. Verwer, P.W. Verbeek, and S.T. Dekker. An efficient uniform cost algorithm applied to distance transforms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(4):425–429, 1989.

[37] M.Y. Sharaiha and N. Christofides. A graph-theoric approach to distance transformations. *Pattern Recognition Letters*, 15:1035–1041, Oct 1994.

[38] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38:113–125, 1994.

[39] I. Ragnemalm. Neighborhoods for distance transformations using ordered propagation. *CVGIP: Image Understanding*, 56(3):399–409, 1992.

[40] O. Cuisenaire and B. Macq. Fast Euclidean distance transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding*, 76(1):163–172, Nov 1999.

[41] J.B.T.M. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2000.

[42] J.A. Sethian. Curvature and the evolution of fronts. *Comm. in Mathematical Physics*, 101:487–499, 1985.

[43] J.A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. of the National Academy of Science*, 93(4):1591–1595, 1996.

[44] L. Bischof R. Adams. Seeded region growing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[45] R. Bellman. *Dynamic Programming*. Princeton University, 1957.

[46] M. Sonka, R. Boyle, and V. Hlavac. *Image Processing, Analysis and Machine Vision*. ITP, 1999.

[47] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[48] Jr. C.R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[49] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.

[50] A. Falcão, J. Stolfi, and R. Lotufo. The IFT project. Available: `http://www.ic.unicamp.br/~afalcao/ift.html`, 2003.

[51] G. Castellano, R.A. Lotufo, A.X. Falcão, and F. Cendes. Characterization of the human cortex in MR images through the image foresting transform. In *Proc. of IEEE Intl. Conf. on Image Processing (ICIP)*, Sep 2003. submitted.

[52] F.P.G. Bergo and A.X. Falcão. Interactive 3D segmentation of brain MR-images with the image foresting transform. In *Proc. of the 6th Intl. Conf. on Medical Image Computing & Computer Assisted Intervention (MICCAI)*, Toronto, CA, Nov 2003. submitted.