An Extension of the Differential Image Foresting Transform and its Application to Superpixel Generation

Marcos A.T. Condori, Fábio A.M. Cappabianco, Alexandre X. Falcão, Paulo A.V. Miranda

PII:	S1047-3203(19)30369-4
DOI:	https://doi.org/10.1016/j.jvcir.2019.102748
Reference:	YJVCI 102748
To appear in:	*J. Vis. Commun. Image R.*
Received Date:	3 May 2018
Revised Date:	24 August 2019
Accepted Date:	24 December 2019

An Extension of the Differential Image Foresting Transform and its Application to Superpixel Generation

Marcos A.T. Condori[a], Fábio A.M. Cappabianco[b], Alexandre X. Falcão[c], Paulo A.V. Miranda[a]

[a] *University of São Paulo, Institute of Mathematics and Statistics, São Paulo, SP, Brazil.*
[b] *Universidade Federal de São Paulo, Instituto de Ciência e Tecnologia, São José dos Campos, SP, Brazil.*
[c] *University of Campinas, Institute of Computing, Campinas, SP, Brazil.*

Abstract

The Image Foresting Transform (IFT) is a graph-based framework to develop image operators based on optimum connectivity between a root set and the remaining nodes, according to a given path-cost function. Its applications involve a variety of tasks, such as segmentation, boundary tracking, skeletonization, filtering, among others. The Differential Image Foresting Transform (DIFT) allows multiple IFT executions for different root sets and a same monotonically incremental path-cost function, making the processing time proportional to the number of modified nodes. In this paper, we extend the DIFT algorithm for non-monotonically incremental functions with root-based increases. This proposed extension, called Generalized DIFT (GDIFT), has been successfully used as the core part of some modern superpixels methods with state-of-the-art results. Experimental results show considerable efficiency gains over the sequential flow of IFTs for the generation of superpixels, also avoiding inconsistencies in image segmentation, which could occur with the regular DIFT algorithm.

Keywords: Image Foresting Transform, superpixels, Differential Image Foresting Transform

Email addresses: `mtejadac@vision.ime.usp.br` (Marcos A.T. Condori), `cappabianco@unifesp.br` (Fábio A.M. Cappabianco), `afalcao@ic.unicamp.br` (Alexandre X. Falcão), `pmiranda@vision.ime.usp.br` (Paulo A.V. Miranda)

1. Introduction

The Image Foresting Transform (IFT) algorithm is a generalization of Dijkstra's algorithm for multiple sources (root sets) and more general connectivity functions [1, 2]. It reduces connectivity-based image processing problems to the computation of an optimum-path forest in an image derived graph, followed by a simple postprocessing of the resulting forest attributes (e.g., cost, path's predecessor node, propagated label). Its applications involve morphological filtering [3], boundary tracking and curve tracing [4], multiscale shape skeletonization [5, 6], region-based segmentation [7], data clustering [8, 9], among others. Recently, the conditions for the optimality of the cost map were corrected in [2]. However, even when the paths of the resulting spanning forest are not optimal, the forest may still be optimal according to other optimality criteria (e.g., a cut measure in the image graph [10, 11]).

In applications that require multiple executions of the IFT algorithm for different seed sets (root candidates) over a same image and using a same path-cost function, such as seed-based interactive image segmentation, the Differential Image Foresting Transform (DIFT) algorithm [12] allows to update the optimum-path forest in time proportional to the size of the modified regions in the image (i.e., in sublinear time). However, it requires that the path-cost function be *monotonically incremental* (MI) — i.e., (a) the costs along the paths can only increase or remain the same and (b) the cost relation between paths with the same terminal node must not change when they are extended from that node. For a more detailed description of MI see Appendix B.

In recent years, non-monotonically incremental functions (NMI) have been employed in the IFT algorithm in order to improve the results for several applications, including segmentation by optimum cuts in graphs subject to high-level constraints [11, 13, 14, 15, 16], interactive segmentation with adaptive cost functions [17, 18], interactive repair [19, 20], boundary tracking and curve tracing [21, 22], and superpixel segmentation [23, 24, 25, 26].

Motivated by the great success of NMI functions, in this paper, we present a novel differential IFT algorithm, named *Generalized DIFT* (GDIFT), which extends the original DIFT algorithm to handle connectivity functions with root-based increases (which can be non-monotonically incremental), maintaining its same time complexity order. Figure 1 shows an example of interactive segmentation comparing the wrong DIFT with NMI functions with its corrected algorithm by GDIFT. In addition to the natural application of GDIFT in interactive segmentation of 3D volumes, allowing the efficient use of modern adaptive functions to deal with image inhomogeneity problems [17, 18], and to perform interactive repair to remove imperfections in automatic results [19], here we emphasize its recent successful application for the generation of superpixels. Indeed, GDIFT currently resides in the main core of some modern superpixels methods [24, 25, 26], being their ground basis to achieve state-of-the-art results. In this context, it avoids segmentation inconsistencies (e.g., disconnected superpixels), which could occur with the DIFT algorithm, as used in [23].

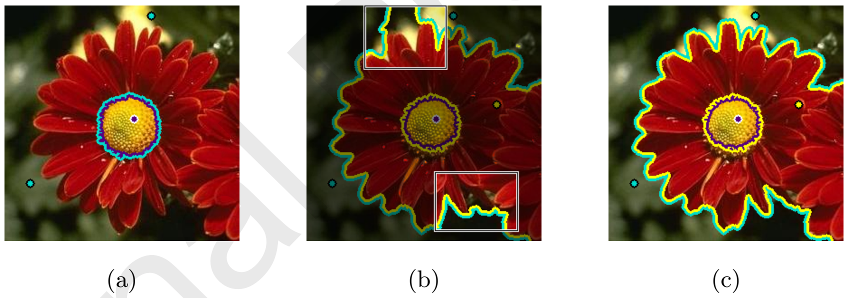


Figure 1: Interactive segmentation example with NMI functions. a) Result of the first interaction by IFT with three initial markers (two for the background and one for the object). b) The updated result after the addition of an object marker (in the flower petal) by the DIFT algorithm from [12] presents the highlighted segmentation inconsistencies. c) The correct result by the proposed GDIFT algorithm for the same addition of markers.

For the experimental comparison with various methods of superpixels, including Simple Linear Iterative Clustering (SLIC) [27], Linear Spectral Clustering (LSC) [28], Entropy Rate Superpixels (ERS) [29], Lazy Random Walks (LRW) [30] and Waterpixels [31], on six image datasets of natural and medical images with distinct object properties, the reader should refer to [24, 26]. Here we focus on the detailed technical description of the GDIFT algorithm which is missing in these other works.

A preliminary version of GDIFT, previously named as DIFTmod, was first presented in a conference paper [32]. In this extended work, we provide more details in the presentation of the method, including a formal definition of its supported functions and its revised algorithm. Moreover, we also present experiments involving the differential computation of the forest in the case of the Euclidean distance path-cost function.

The paper is organized as follows. Section 2 gives the related works, including the required background on IFT and DIFT algorithms. Section 3 discusses the problems of a first attempt to extend DIFT for connectivity functions with root-based increases. Based on this detailed analysis of problems from the previous section, the new algorithm, called GDIFT, is then proposed in Section 4. The experimental results are discussed in Section 5 and we conclude the paper in Section 6.

2. Related works

The fast incremental calculation of a new result with an updated starting condition from a previous one is a recurrent problem in different frameworks [33]. For instance, in the Random Walks segmentation algorithm, the fast editing of a segmentation by adding more seeds may be obtained by using the previous solution as the initialization of an iterative matrix solver [34], while in the Graph Cuts segmentation method, a maximum flow on a new graph is efficiently obtained starting from the previous flow without recomputing the whole solution from scratch, by increasing the edge capacity of terminal links connected to the new seeds [35]. In the context of IFT, the incremental computation leads to DIFT. In this section, we present these related methods, which are required to introduce the proposed extension, GDIFT. Given that GDIFT is important within the context of superpixel generation via the ISF framework [24], we also discuss it briefly.

2.1. Image Foresting Transform (IFT)

The Image Foresting Transform (IFT) algorithm [1] interprets an image as a graph $G = \langle \mathcal{I}, \mathcal{A} \rangle$, whose nodes in $\mathcal{I}$ are image elements (pixels, vertices, regions) and arcs in $\mathcal{A}$ are defined by a given *adjacency relation*. For instance, take the nodes as the pixels in the image domain $\mathcal{I} \subset \mathbb{Z}^2$ and the arcs as the ordered neighboring pixel pairs $\langle s, t \rangle \in \mathcal{A}$ (e.g., 4/8-neighborhood in 2D and 6/26-neighborhood in 3D). A *connectivity function* assigns a cost to any path in the graph, including the trivial ones formed by a single node. Initially, all nodes represent trivial paths and the minima of the initial cost map form a seed set (root candidates). The seeds compete among themselves by offering minimum-cost paths to the remaining nodes, such that the cost map is minimized as paths are propagated (extended) from the seed set and the graph is partitioned into an *optimum-path forest* rooted at the winner seeds. In the context of image segmentation, the nodes along the computed paths receives the label from their corresponding root nodes (winner seeds), such that objects are defined as sets of nodes sharing a same label.

The general IFT algorithm can run in time proportional to the number of nodes for most applications. It propagates paths with lower costs by using a priority queue. Hence, linear-time implementation is possible by bucket sorting, whenever the cost increments along the paths are integers and limited [1, 4]. When this is not the case, it executes in linearithmic time by using a binary heap as priority queue.

For the purpose of completeness in presentation, we include the IFT algorithm [1] in Appendix B, using the standardized notation of this work given in Appendix A.

2.2. Differential Image Foresting Transform (DIFT)

In interactive region-based segmentation from markers (i.e., set of seeds), the user can add markers to and/or remove markers from previous interactions in order to improve the segmentation results. In the context of IFT, instead of starting over the segmentation for each new set of seeds, DIFT can be employed to update the segmentation in a differential manner, by correcting only the wrongly labeled parts of the forest. This greatly increases efficiency, which is crucial to obtain interactive response times in the segmentation of large 3D volumes (Figure 2).

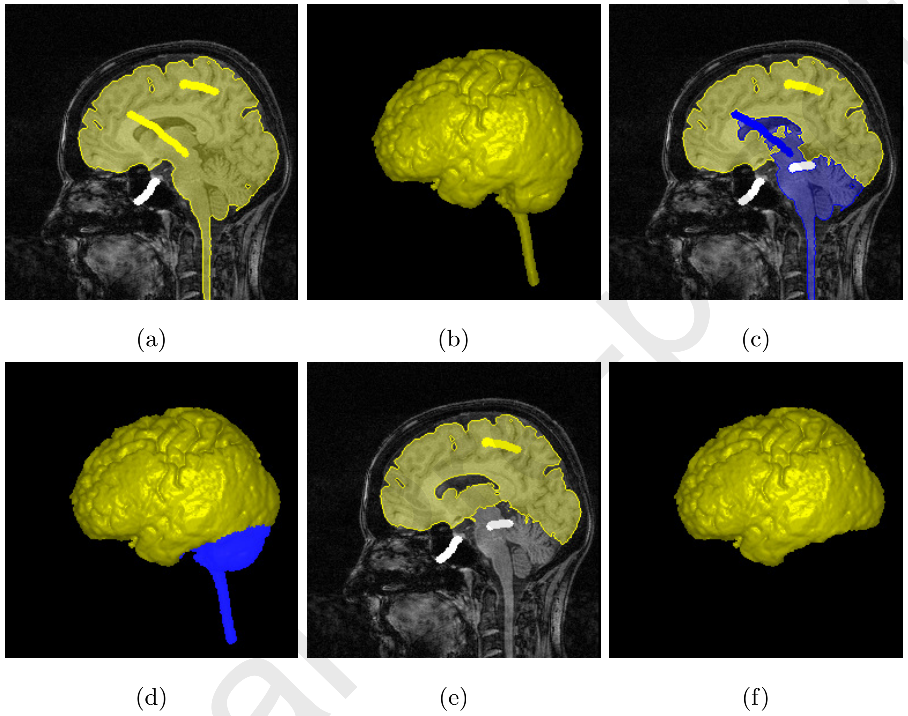


Figure 2: Example of DIFT in a 3D MR-T1 image of 256 × 256 × 143 voxels. By removing and adding markers, an initial 3D brain segmentation by IFT watershed from markers is gradually converted into a segmentation without the brain stem, cerebellum and ventricles. (a-b) The initial segmentation and its 3D rendition. (c-d) The object marker passing through the ventricle is marked for removal and the voxels from its optimum-path trees (in blue) become available for a new dispute among the remaining roots and the new selected background marker. (e-f) Final segmentation of cerebral hemispheres. The initial result took 4 sec, while the editing changes were made in only 292 ms in an Intel Core i3-5005U CPU 2.00GHz ×4.

For the purpose of completeness in presentation, we include the DIFT algorithm [12] for MI functions in Appendix C, using the standardized notation of this work given in Appendix A, in order to allow the discussion of its required modifications, aiming at its extension to other path-cost functions in the next sections.

2.3. Superpixel by Iterative Spanning Forest (ISF)

The over-segmentation of an image leading to a partition into k regions of similar and connected pixels is known as superpixels (supervoxels in 3D).

The Iterative Spanning Forest (ISF) framework [24] computes improved sets of connected superpixels by a sequence of image foresting transforms, where each superpixel structurally corresponds to a spanning tree rooted at a corresponding seed. Initially, k seeds are selected according to a *seed sampling strategy* (e.g., grid sampling, regional-minima-based sampling or mixed entropy-based seed sampling). Seed positions are then updated at each step in order to generate superpixels frontiers with better adhesion to the boundaries of objects in the image, following a *seed pixel recomputation procedure.* However, the connectivity functions leading to the best results in ISF are all NMI functions, so that their efficient implementation by the differential flow requires the GDIFT algorithm as proposed in this work. More details about ISF can be seen in the experiments involving superpixels in Section 5.1.

3. A first attempt to extend DIFT

From the information in Appendix B, we know that the state test $S(t) \neq 1$ is required for NMI functions in IFT (Algorithm 3). The original DIFT algorithm does not have the *state test* $S(t) \neq 1$, because it was developed only for MI functions. Given that we are interested in functions that are not MI, let's modify Algorithm 4 to include this test, by changing its condition at Line 14 to $S(t) \neq 1$, and name this new version as DIFT*.

Unfortunately, the introduction of the *state test* $S(t) \neq 1$ in Algorithm 4 is not sufficient to guarantee its correctness in the case of non-monotonically incremental functions (NMI). In fact, its introduction generates problems even in cases involving MI functions. Under certain circumstances it generates inconsistencies between the map of labels (and/or roots) and the map of predecessors, as defined in Appendix A.

- **Inconsistency in the root map:** A root map R is inconsistent if there is a node $s \in \mathcal{I}$ such that $\pi_s^P = \langle t_1, \ldots, t_n = s \rangle$ and $R(s) \neq t_1$.
- **Inconsistency in the label map:** A label map L is inconsistent if there is a node $s \in \mathcal{I}$ such that $\pi_s^P = \langle t_1, \ldots, t_n = s \rangle$ and $L(t_1) \neq L(s)$.

150 The detailed study of the reasons behind the occurrence of these inconsistencies, as presented in Sections 3.1 and 3.2, will be fundamental for the proposal of the new algorithm in Section 4.

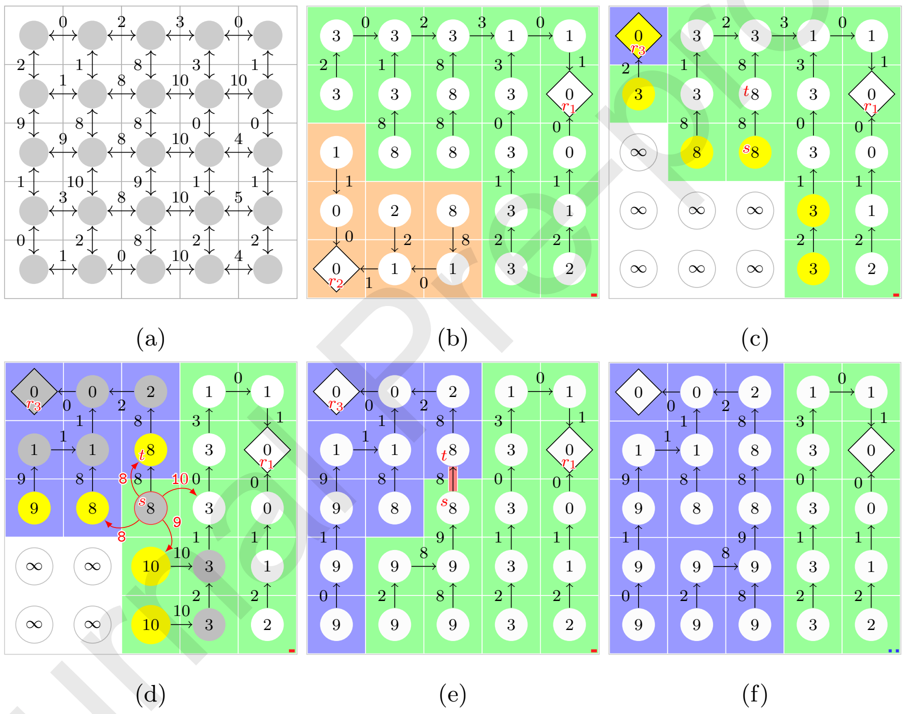


Figure 3: Execution of DIFT* algorithm, where the roots are indicated with diamond shape. Yellow pixels are in the priority queue Q and gray pixels have already been evaluated. a) A symmetric graph with 4-neighborhood adjacency and weights indicated in the arcs. b) First iteration $IFT_{(\mathcal{S}_1)}$ with f_{max} function (Eq. 1). c) Addition of a new marker (upper corner) and frontier set wraps the removed orange zone. d) Some steps later in DIFT*, the pixel s, indicated with red border, is evaluated, propagating a wrong label (green). Afterward, its predecessor t will be evaluated but it won't be able to fix the label of s due to the state test. e) The final result obtained by DIFT* with an inconsistent label map. f) The desired result.

3.1. DIFT* problems with MI functions

Figure 3 presents an example in which an inconsistency in the map of labels L happens after an execution of the DIFT* algorithm. It uses the cost function f_{max} in Equation 1, where π_s is a path (see Appendix A), $\delta(s,t)$ is the arc weight of $\langle s,t\rangle$ and $\mathcal{H}(s)$ is a handicap value (e.g., in Figure 3, $\mathcal{H}(s)$ is assumed to be zero for the seeds and infinity otherwise).

$$\begin{aligned} f_{max}(\langle s\rangle) &= \mathcal{H}(s) \\ f_{max}(\pi_s \cdot \langle s,t\rangle) &= \max\{f_{max}(\pi_s), \delta(s,t)\} \end{aligned} \tag{1}$$

Given the initial image graph in Figure 3(a), an initial segmentation $IFT_{(\mathcal{S}_1)}$ is executed producing the maps in Figure 3(b) for the first seed set $\mathcal{S}_1$ with two markers in green and orange. At the beginning of the second iteration $DIFT^*_{(\Delta^+_{\mathcal{S}_2},\Delta^-_{\mathcal{S}_2},\mathcal{C}_1)}$, shown in Figure 3(c), the orange marker and its subtree are removed and a blue marker is inserted (see Appendix C). The frontier pixels of the removed subtree and the inserted marker are show in yellow. Figure 3(d) shows the moment in which an inconsistency occurs in the label map during the execution of DIFT*. Pixel s leaves the queue Q before pixel t and changes its state, because of the employed First-In-First-Out (FIFO) tie-breaking policy. Later, when pixel t leaves the queue, it can not propagate its label to s, since s was already evaluated ($S(s) = 1$). Therefore, in the end of the DIFT* execution, t is the predecessor of s, but they have distinct labels (see Figure 3(e)). Figure 3(f) shows the expected result, which is consistent with the one by $IFT_{(\mathcal{S}_2)}$ using the same two markers.

Moya and Falcão were the first to identify this problem in DIFT*. In [36] the following alternative solutions were analyzed for MI functions.

1. Set the tie-breaking policy of Q as Last-In-First-Out (LIFO).
2. Consider the addition and removal operation as two independent operations.
3. Do not use the state test $S(t) \neq 1$ in the DIFT* algorithm.

In a LIFO tie-breaking policy, all nodes to be conquered with the same cost would be attributed to the new seed, however, a LIFO policy does not generate a well-balanced treatment of tie zones, which is not desirable in practice. To handle a simultaneous addition and removal operation as two independent operations, it would be necessary to run the main loop of DIFT* twice, adding computational effort. The third solution corresponds to using the original DIFT algorithm from [12], which solves the aforementioned problem by updating each node multiple times, which is inefficient.

All of these alternatives do not work properly for more general functions, including non-monotonically incremental functions (NMI), which is the objective of this work.

3.2. DIFT* problems for functions with root-based increases

For non-monotonically incremental functions (NMI), the usage of the state test is mandatory in order to avoid the reprocessing of pixels and the occurrence of infinite loops. However, in this section, we show how DIFT* algorithm fails to output consistent cost, root and label maps for NMI functions from a particular class of functions, in which the cost increase of a path depends on some property of its root node.

Definition 1 (Connectivity function with root-based increases (RI))**.** *We define a connectivity function with increases based on its root, as a function in which, for the extension of a path $\pi_{r\rightsquigarrow s}$ by an arc $\langle s,t\rangle$, we have that $\Delta_C = f(\pi_{r\rightsquigarrow s} \cdot \langle s,t\rangle) - f(\pi_{r\rightsquigarrow s})$ is a non-negative function of only r, s and t. That is, $\Delta_C = g(r,s,t) \geq 0$.*

Note that the connectivity functions with root-based increases are a particular case of a more general class of functions with prefix-based increases.

Definition 2 (Connectivity function with prefix-based increases (PI))**.** *We define a connectivity function with increases based on its prefix, as a function in which, for the extension of a path $\pi_{t_{n-1}} = \langle t_1,\ldots,t_{n-1}\rangle$ by an arc $\langle t_{n-1},t_n\rangle$,* *we have that $\Delta_C = f(\pi_{t_{n-1}} \cdot \langle t_{n-1}, t_n\rangle) - f(\pi_{t_{n-1}})$ is a non-negative function of all pixels $t_1, \ldots, t_n$. That is, $\Delta_C = g(t_1, \ldots, t_n) \geq 0$.*

One example of a connectivity function with prefix-based increases can be seen in [22], where a new approach is proposed for curve tracing and boundary tracking.

In this section we are interested in connectivity functions with root-based increases. We will consider function f_{euc} in Equation 2 for the squared Euclidean distance transform and the function f_{abs_add} in Equation 3, which is a simplification of the function used in [23] for the generation of superpixels, where $I(t)$ denotes the image intensity at pixel t.

$$\begin{aligned} f_{euc}(\pi_t = \langle t\rangle) &= \begin{cases} 0 & \text{if } t \in \mathcal{S} \\ +\infty & \text{otherwise} \end{cases} \\ f_{euc}(\pi_{r\rightsquigarrow s} \cdot \langle s,t\rangle) &= \begin{cases} \|t-r\|^2 & \text{if } r \in \mathcal{S} \\ +\infty & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

$$\begin{aligned} f_{abs\_add}(\pi_t = \langle t\rangle) &= \begin{cases} 0 & \text{if } t \in \mathcal{S} \\ +\infty & \text{otherwise} \end{cases} \\ f_{abs\_add}(\pi_{r\rightsquigarrow s} \cdot \langle s,t\rangle) &= f_{abs\_add}(\pi_{r\rightsquigarrow s}) + |I(r) - I(t)| + 1 \end{aligned} \tag{3}$$

In the presented equations, the path extension depends on a root property (its pixel intensity or its relative position in the image). Note that function f_{euc}, from the theoretical point of view, is not a cost function with root-based increases, since $\Delta_C = f_{euc}(\pi_{r\rightsquigarrow s} \cdot \langle s,t\rangle) - f_{euc}(\pi_{r\rightsquigarrow s}) = \|t-r\|^2 - \|s-r\|^2$ may be negative. Δ_C is negative when the pixel t is closer to the root r than s. However, in practice, the IFT algorithm with f_{euc} only evaluates the cases where $\Delta_C \geq 0$, because t would already have been conquered by another shorter path. Therefore we also consider f_{euc} in our experiments.

As our purpose is to compute a sequence of IFTs, in a differential manner, next we show the inconsistencies that are generated by DIFT* for functions with root-based increases.

Inconsistency in the cost map: An inconsistency in the map of cost occurs when DIFT* generates a cost map that does not correspond to any valid cost map that can be generated by the IFT algorithm for the same graph and seeds.

DIFT* with function f_{euc} may generate an inconsistency in the cost map. Consider the IFT result with a single seed presented in Figure 4(b) over the input image graph of Figure 4(a). Now suppose that a blue marker (r') is inserted as shown in Figure 4(c). After a few steps by DIFT*, the pixel s offers to its neighboring pixel t a higher cost (i.e., 5) than its current one (i.e., 4), as shown in Figure 4(d), but by the *test of predecessor*, pixel t is conquered by the path $\pi_{r' \rightsquigarrow s} \cdot \langle s, t \rangle$, as shown in Figure 4(e). This leads to an inconsistent cost map since pixel t will have a higher cost compared to the result obtained by IFT for the final seed set (compare Figures 4(e) and (f)).

Applying DIFT* algorithm with path function f_{abs_add} described in Equation 3 may output a graph with cycles, which violates the fact that the IFT algorithm should result in a spanning forest. Consider for the the image graph in Figure 5(a), the forest in Figure 5(b) computed by IFT. Then, the orange marker and its tree are removed, and a new green marker is inserted, as shown in Figure 5(c). The yellow pixels indicate the frontier obtained after the tree removal. After a few steps by DIFT*, pixel s offers a higher cost to t than its current one (Figure 5(d)). Nevertheless, by the predecessor test, pixel t is conquered by the path $\pi_s \cdot \langle s, t \rangle$ (Figure 5(e)). Note that the pixel with red bound in Figure 5(e) leaves the queue Q before the pixel t, conquering it again, such that a cycle is generated, as shown in Figure 5(f). Note that the connected component to which t belongs does not even have a root node.

4. Proposal: The GDIFT algorithm

We propose here a novel algorithm, GDIFT (Algorithm 1), to reinforce the *predecessor test*. The objective is to continue using the *state test*, having a high performance solution and to be able to deal with the previously reported problems, including the issue reported in Section 3.1 and also the problems with

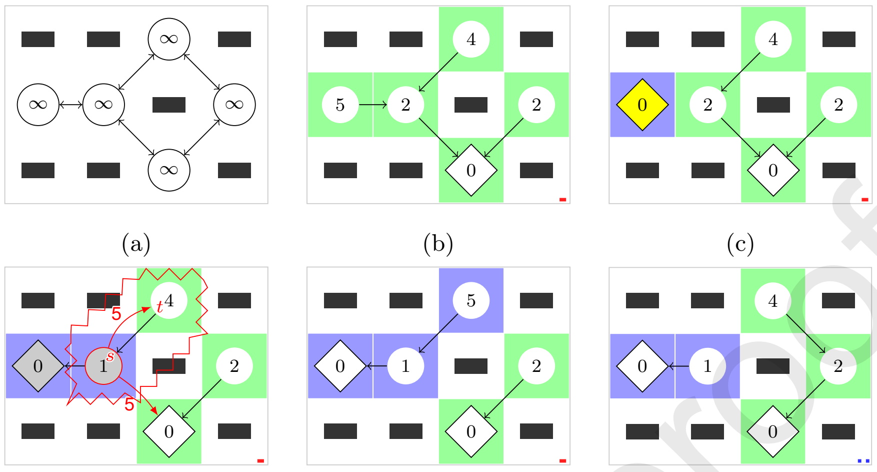


Figure 4: Example of inconsistency by DIFT* with f_{euc}. The roots are indicated in diamond shape, the yellow color is used to indicate pixels that are in the priority queue Q and gray pixels are used for nodes that have already been evaluated. a) Graph with 8-neighborhood adjacency and the initial cost map, before inserting the seed. b) The initial forest computed by IFT with f_{euc}. c) The addition of a new marker. d) After some steps by DIFT*, the pixel s offers a higher cost to t than its current cost in $V(t)$. Nevertheless, t is conquered by s due to the test of predecessor. e) Result obtained by DIFT*. f) The expected result.

the non-monotonically incremental functions presented in Section 3.2.

Moya's alternatives [36], presented in Section 3.1, do not support our objective. Therefore, we propose the following alternative: If the cost offered by path $\pi_{r' \rightsquigarrow s} \cdot \langle s, t \rangle$ to pixel t is equal to its current cost $V(t)$, both pixels have different roots $R(s) \neq R(t)$ and $s = P(t)$, then t is conquered by $\pi_{r' \rightsquigarrow s} \cdot \langle s, t \rangle$ and t is inserted in the priority queue Q at the beginning of its corresponding bucket (i.e., a LIFO tie-breaking policy is adopted only for this insertion, after which the priority queue gets back to work according to a FIFO tie-breaking policy, see Lines 20 and 25-27 of Algorithm 1). This prevents path suffixes of path $\pi_{r' \rightsquigarrow t}$, with the same value of $V(t)$ in Q to be evaluated before the pixel t, such that inconsistencies of labels and roots from Section 3.1 are eliminated.

Now, to prevent the problems related to functions which employ properties

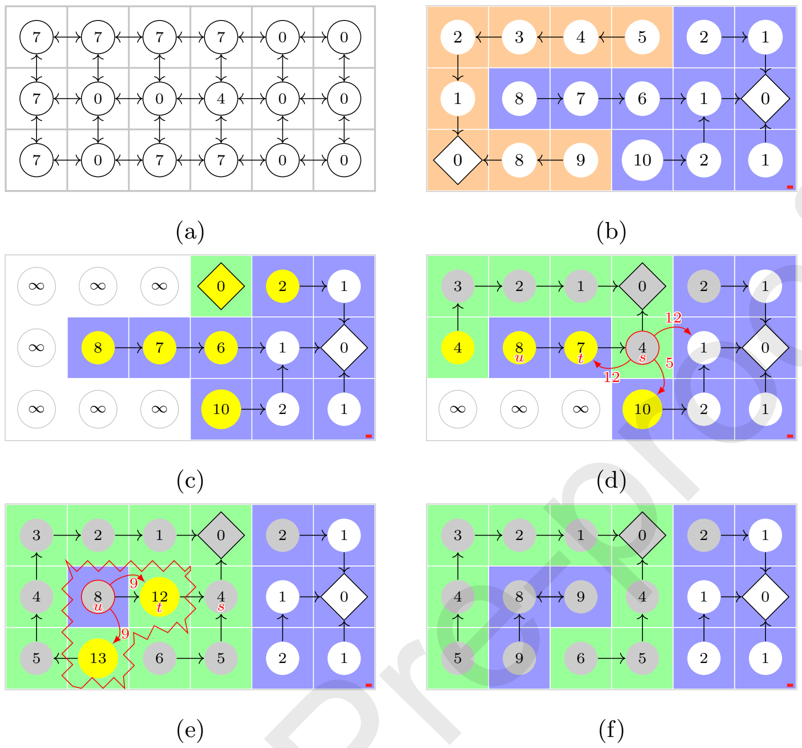


Figure 5: Example of inconsistency by DIFT* with function f_{abs_add}. The roots are indicated in diamond shape and gray pixels indicate pixels that have already been evaluated. a) Graph with 4-neighborhood adjacency and with the pixel intensities depicted in the nodes. b) The initial forest computed by IFT with f_{abs_add}. c) The removal of the orange tree and the addition of a new marker (green). The yellow pixels denote the frontier set. d) Some steps later by DIFT*, pixel s offers a higher cost to pixel t than its current one in $V(t)$, but by the test of predecessor t is conquered. The yellow pixels indicate the pixels in Q. e) The node marked with a red border leaves the queue before node t, generating a cycle in the graph. f) The incorrect resulting graph with a cycle obtained by DIFT*, instead of a spanning forest.

of roots, as explained in Section 3.2, we must include other conditions to the *predecessor test* of the DIFT* algorithm. The idea is to verify if the subtree of a pixel s, being evaluated, is in an inconsistent state. A subtree rooted in t is inconsistent if the cost offered to t by a new path π_s extended by $\langle s, t\rangle$ is greater than the current cost $V(t)$ and s is the predecessor of t. In this case, the procedure DIFT-RemoveSubTree must be executed (Line 23 of Algorithm 1) in order to release the inconsistent subtree for a new dispute among seeds and frontier pixels, avoiding the inconsistencies from Section 3.2.

As a final note, when it is not possible to have a priority queue with the two tie-breaking policies, the Lines 25-27 of Algorithm 1 could be replaced by a call to procedure DIFT-RemoveSubTree. This alternative solution generates the removal of the subtree releasing its region to a later dispute among seeds and frontier pixels, avoiding possible inconsistencies from Section 3.1.

Procedure DIFT-RemoveSubTree in Algorithm 2, releases the entire subtree, converting its pixels to trivial trees of infinite cost, and transforms all of its neighboring pixels into frontier pixels, inserting them in Q, assuming that the graph is symmetric.

Algorithm 1. – ALGORITHM GDIFT

INPUT:	Image graph $\langle \mathcal{I}, \mathcal{A} \rangle$, path-cost function f, the set $\Delta_{\mathcal{S}}^{+}$ of seeds for addition, set $\Delta_{\mathcal{S}}^{-}$ of seeds for removal, the maps L, R, V and P initialized with the result from the previous IFT/DIFT execution, and an initial labeling function $\lambda : \Delta_{\mathcal{S}}^{+} \rightarrow \{0, \ldots, l\}$ for the new seeds.
OUTPUT:	The updated maps L, R, V and P.
AUXILIARY:	Priority queue Q, variable tmp, the set of frontier pixels F, and an array of status $S : \mathcal{I} \rightarrow \{0, 1\}$, initialized as $S(t) = 0$ for all $t \in \mathcal{I}$, where $S(t) = 1$ for processed nodes and $S(t) = 0$ for unprocessed nodes.

```
1.  Set $Q \leftarrow \emptyset$.
2.  If $\Delta_{\mathcal{S}}^{-} \neq \emptyset$, then
3.      $(L, R, V, P, F) \leftarrow$ DIFT-TreeRemoval $(\langle \mathcal{I}, \mathcal{A} \rangle, \Delta_{\mathcal{S}}^{-}, L, R, V, P)$
4.      $F \leftarrow F \setminus \Delta_{\mathcal{S}}^{+}$
5.      For each $t \in F$, do insert $t$ in $Q$.
6.  For each $s \in \Delta_{\mathcal{S}}^{+}$, do
7.      If $f(\langle s \rangle) < V(s)$, then
8.          Set $V(s) \leftarrow f(\langle s \rangle)$, $L(s) \leftarrow \lambda(s)$, $R(s) \leftarrow s$, $P(s) \leftarrow nil$
9.          Insert $s$ in $Q$.
10. While $Q \neq \emptyset$, do
11.     Remove $s$ from $Q$ such that $V(s) = \min_{\forall t \in Q}\{V(t)\}$.
```

```
12.     Set $S(s) \leftarrow 1$.
13.     For each node $t$ such that $\langle s,t \rangle \in \mathcal{A}$, do
14.         If $S(t) \neq 1$, then
15.             Compute $tmp \leftarrow f(\pi_s^P \cdot \langle s,t \rangle)$.
16.             If $tmp < V(t)$, then
17.                 If $t \in Q$, then remove $t$ from $Q$.
18.                 Set $P(t) \leftarrow s$, $V(t) \leftarrow tmp$.
19.                 Set $R(t) \leftarrow R(s)$, $L(t) \leftarrow L(s)$.
20.                 Insert $t$ in $Q$ with FIFO tie-breaking policy.
21.             Else If $s = P(t)$, then
22.                 If $tmp > V(t)$, then
23.                     DIFT-RemoveSubTree($L, R, V, P, Q, S, t$)
24.                 Else If $R(t) \neq R(s)$, then
25.                     If $t \in Q$, then remove $t$ from $Q$.
26.                     $L(t) \leftarrow L(s)$, $R(t) \leftarrow R(s)$.
27.                     Insert $t$ in $Q$ with LIFO tie-breaking policy.
```

Algorithm 2. – PROCEDURE DIFT-REMOVESUBTREE

INPUT:	The maps L, R, V and P, the priority queue Q, the array of status $S : \mathcal{I} \rightarrow \{0,1\}$, and node t (root of the subtree to be removed).
OUTPUT:	The updated maps L, R, V and P, the updated priority queue Q and the updated array of status S.
AUXILIARY:	Queue J and a set F.

```
1.  Set $J \leftarrow \emptyset$, $F \leftarrow \emptyset$.
2.  Insert $t$ in $J$.
3.  While $J \neq \emptyset$, do
4.      Remove $s$ from $J$.
5.      Set $S(s) \leftarrow 0$.
6.      Set $P(s) \leftarrow nil$, $R(s) \leftarrow s$, $V(s) \leftarrow \infty$.
7.      If $s \in Q$, then remove $s$ from $Q$.
8.      For each node $t$ such that $\langle s,t \rangle \in \mathcal{A}$, do
9.          If $s = P(t)$, then insert $t$ in $J$.
```

```
10.             Else If $t \notin F$ and $V(t) \neq \infty$, then
11.                 Insert $t$ in $F$.
12. While $F \neq \emptyset$, do
13.     Remove $s$ from $F$.
14.     If $s \notin Q$ and $V(s) \neq \infty$, then
15.         Set $S(s) \leftarrow 0$ and insert $s$ in $Q$.
```

5. Experimental results

In this section, we evaluate the proposed differential algorithm in relation to the sequential application of IFTs, in order to illustrate its suitability to overcome all the problems presented in Sections 3.1 and 3.2. The gains of using GDIFT compared to DIFT* are presented in two applications: generation of superpixels and dynamic update of the Euclidean Distance Transform.

5.1. Application to generation of superpixels

Superpixels are the result of an over-segmentation of the image and have been used in a wide variety of applications of computer vision, including segmentation, object detection and 3D reconstruction [37]. One of the most well-known superpixel algorithms is the Simple Linear Iterative Clustering (SLIC) [27], which divides the image using an adaptive k-means clustering.

Here we discuss the results by the IFT-SLIC algorithm [23], which corresponds to a particular case of the ISF framework [24]. In order to compute k superpixels, IFT-SLIC considers a first initialization of k clusters in a similar way as SLIC and adapts the superpixels by calculating the IFT with a connectivity function that depends on the root pixel of each superpixel according to Equation 4. The algorithm is repeated a few times (e.g., 10), updating the central position of each superpixel.

$$\begin{aligned} f_D(\pi_t = \langle t \rangle) \; &= \begin{cases} 0 & \text{if } t \in \mathcal{S} \\ +\infty & \text{otherwise} \end{cases} \\ f_D(\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle) &= f_D(\pi_{r \rightsquigarrow s}) + (\|I(r) - I(t)\| \cdot \alpha)^{\beta} + \|t - s\| \end{aligned} \tag{4}$$

The differential mode of IFT-SLIC is possible by using DIFT in order to reduce the computation effort at each iteration, but since the path function used by IFT-SLIC is similar to f_{abs_add} (Equation 3), inconsistencies may occur. With the modifications presented in Section 4 though, it is possible to generate consistent results, capable of being reproduced by the execution of IFT. A demonstration is available at the author's website[1].

To compare the time reduction and the obtained results, we execute the algorithms for different values of the parameters. In Equation 4, we used 20 samples for α in [0.01,0.2] and β equals to 12 as recommended in [23].

In the experiment, we used the test set of 50 natural images of the public GrabCut dataset [38] on a 2.40 GHz Intel(R) i7-3630QM CPU. We compare DIFT* and GDIFT for the differential flow, taking the labeled superpixels by the IFT algorithm executed 10 times (IFT 10x) for the sequential flow as our gold standard. Since the number of superpixels is the same in all results, in order to compute the accuracy, we compare the resulting labels of the superpixels by the differential algorithms with the corresponding labels by the IFT 10x.

Figure 6 shows one example. The borders of each superpixel by IFT 10x and GDIFT are shown in Figures 6(a-b). The divergence composed of non-matching labeled pixels is shown in Figure 6(d). Note that the greatest divergence occurs at the borders of each superpixel. This divergence occurs in areas with little information to define good borders (absence of contrast), which are related to tie zones (Figure 6(c)). Given that it is common to have thin tie zones at the boundaries of the objects, we give a tolerance of 1 pixel at the borders and

[1]`http://www.vision.ime.usp.br/~mtejadac/gdift.html`

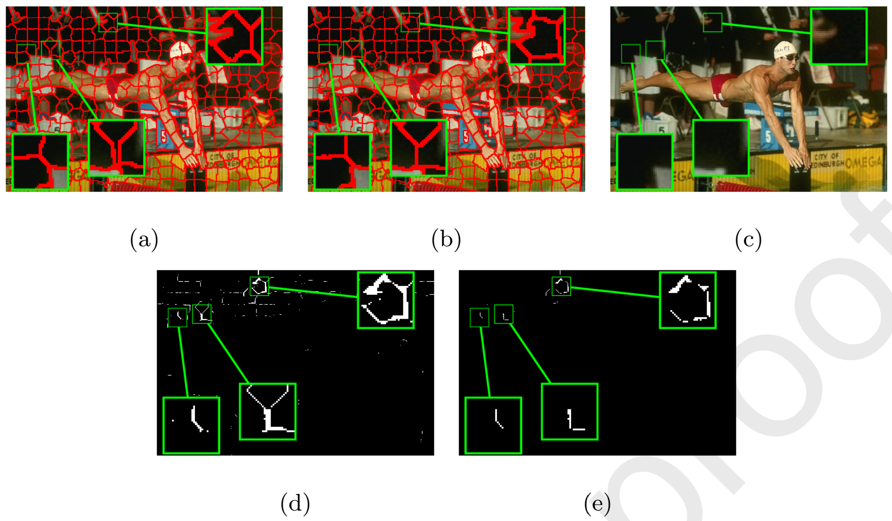


Figure 6: Graphical visualization of the accuracy of the algorithms. a-b) The borders of the superpixels obtained by the algorithms IFT x10 and GDIFT respectively, are shown with parameters $k = 450$ and $\alpha = 0.06$. The divergent areas are highlighted with zoom. c) The corresponding regions in the original image have low contrast to define precise edges. d) Divergent areas between results of IFT x10 and GDIFT algorithms. Note that most of the differences occur in areas with low contrast, which are related to tie zones. e) Divergent areas with 1-pixel tolerance (adjacency 4) at the edges of superpixels. White pixels represent notable differences in the algorithm results.

consider the result of Figure 6(e). At the end of the process, Figure 6(e) shows the largest differences between the result by IFT x10 and GDIFT. We repeat this process also comparing DIFT* and IFT 10x, in order to evaluate which differential method gets the higher agreement with IFT 10x. The percentage of divergence observed in this experiment is shown in Figure 7.

For the DIFT* algorithm, the divergence grows as we increase the value of the parameter α, indicating a higher probability of having inconsistencies in its results. On the other hand, for the GDIFT algorithm the divergence remains almost constant, presenting low values ($< 0.05\%$) as compared to the total number of pixels in the image. Figure 8 shows the visual results of DIFT* and GDIFT.

Regarding the processing time, we compare the running times of the se-

quential flow and differential flows by DIFT* and GDIFT. Figure 9 shows the processing time for different values of k (the number of superpixels). The time of
345 the first IFT execution, which is common for the three algorithms, is indicated in the blue zone. The range of recommended alpha values ($\alpha \in [0.04, 0.08]$ as indicated in [23] and [19]) is also highlighted in the figures. Clearly, the computation of superpixels by IFT-SLIC using the proposed GDIFT algorithm leads to a low computational cost and high precision results compared to IFT x10.

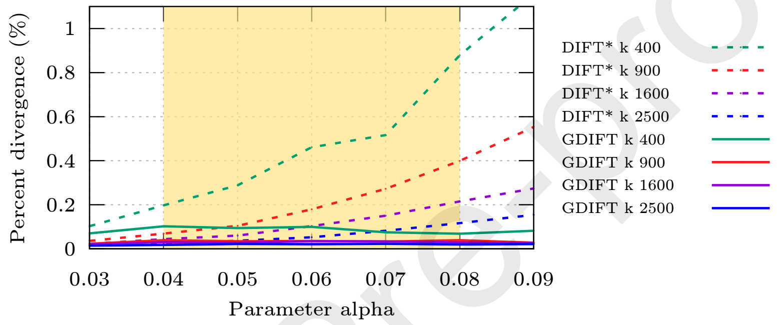


Figure 7: Percentage of pixels divergence between the results by IFT x10 and the results of differential flows by DIFT* and GDIFT, for different numbers of superpixels k.

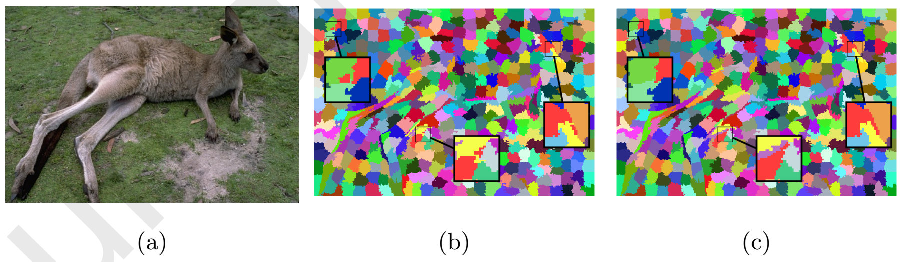
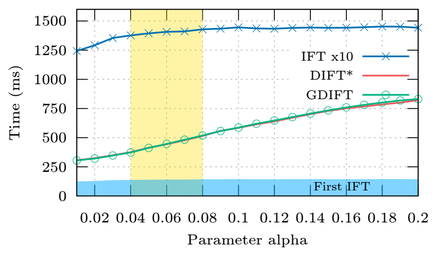
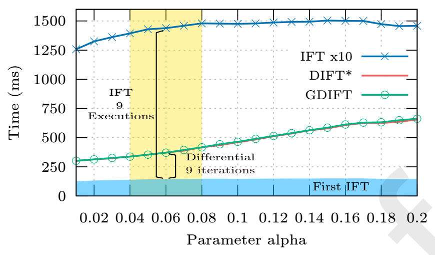


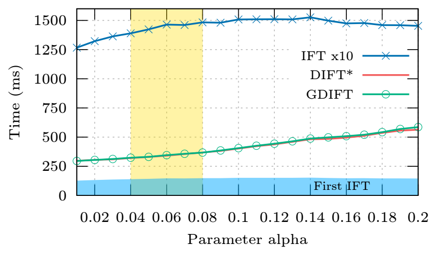
Figure 8: Inconsistencies generated by DIFT* and the result by GDIFT for parameters $k = 500$ and $\alpha = 0.1$. a) Original image. b) Image with superpixels labels by DIFT* presenting inconsistencies (disconnected regions), with focus on zoomed red pixels. c) Image with superpixels labels by GDIFT with no label inconsistencies.



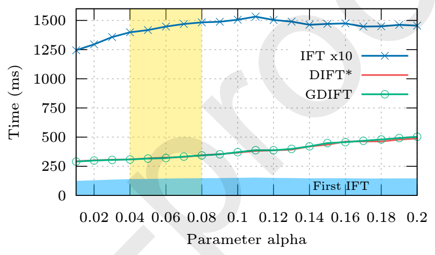
(a) k equal to 400



(b) k equal to 900



(c) k equal to 1600



(d) k equal to 2500

Figure 9: The mean running time of the methods for different values of k.

5.2. *Application to Euclidean Distance Transform*

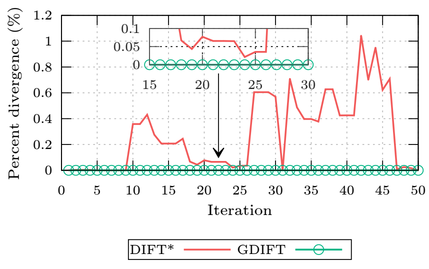
Experiments with f_{euc} (Equation 2) to compute the Euclidean Distance Transform (EDT) in IFTs sequences [1] were carried out to compare the values of the cost V and label L maps obtained through the differential algorithms and the results of the sequential flow.

First we calculate the Voronoi diagram in an image of size 512×512 with 40 random points, then in each iteration, some seeds are marked for removal and other novel seeds are added[2], on average 10% of modifications are made on the number of markers in each iteration. These iterations are performed differentially by DIFT* and GDIFT algorithms and sequentially by IFT.

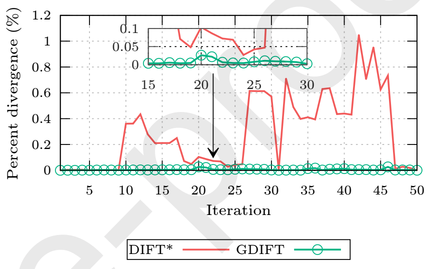
Figure 10(a) shows the percentage of divergence, in each of the 50 iterations,

[2]One possible application would be to dynamically update the voronoi diagram of moving objects (e.g., robots, players, vehicles) to monitor the distances between them or for the terrain analysis in real-time strategy games [39].

between the cost maps of the differential algorithms and the cost map of the IFT. No divergence in the cost map of GDIFT was found. However, for DIFT* the divergence is considerable and varies between iterations. The divergence of labels is shown in Figure 10(b), where GDIFT has a divergence of less than 0.05% (caused by tie-zones at the borders). In the DIFT* algorithm, cost and label maps divergences follow a similar pattern, where the cost is different, the more likely is the label to be different.



(a) Divergence of the cost map V

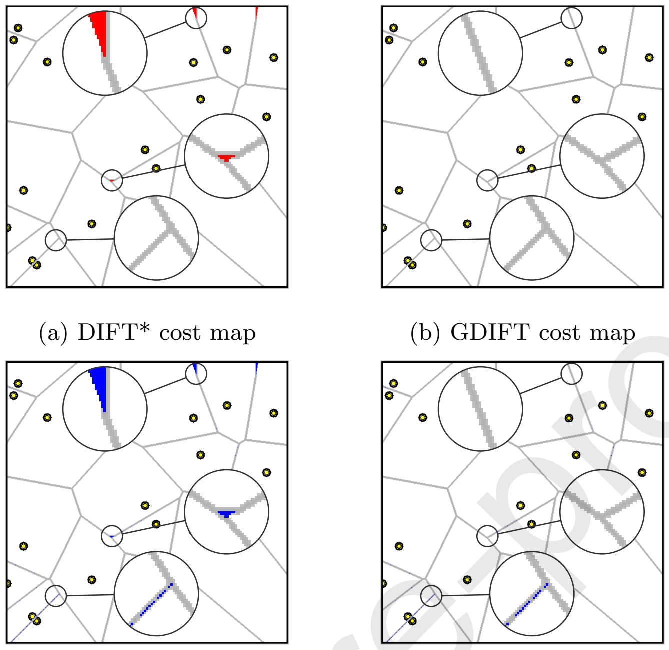


(b) Divergence of the label map L

Figure 10: Percentage of pixels divergence in cost and label maps between results of differential algorithms and IFT in each iteration.

Figure 11 shows the region where the greatest divergence of labels occurs for GDIFT (20th iteration of Figure 10(b)). For the same region, it also shows the divergence of labels and costs for both GDIFT and DIFT*, where the red and blue regions represent the divergences in the cost and label maps respectively.

As a result of the experiment, we present the mean percentage of divergence of 30 executions (of 50 iterations each) as shown in Figure 12(a-b). The execution time of IFT along the iterations has a constant behavior, since all the nodes of the graph are always modified (Figure 12(c)). However the differential algorithms (DIFT* and GDIFT) have a better performance, since only a subset of the nodes in the image domain is reached by the seeds in $\Delta^+_{\mathcal{S}}$ and $\Delta^-_{\mathcal{S}}$.



(a) DIFT* cost map (b) GDIFT cost map

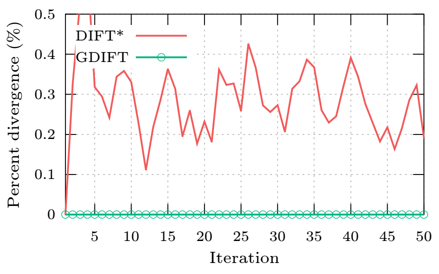
(c) DIFT* label map (d) GDIFT label map

Figure 11: Cost and label maps for the 20th iteration of Figure 10(b)

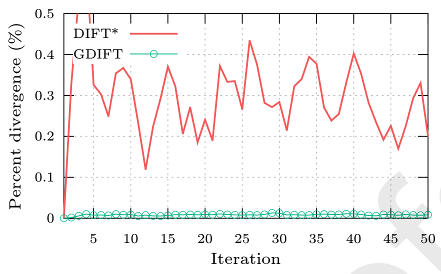
6. Conclusion

In this paper, we presented an extension of the DIFT algorithm, named GDIFT, to support connectivity functions with root-based increases, demonstrating the importance of exploring other types of connectivity functions in the IFT framework. The experiments show results compatible with multiple sequential IFT executions, but with reduced running times. The proposed algorithm is being used in several parallel works, with recent publications for state-of-the-art superpixel segmentation [24, 25, 26].

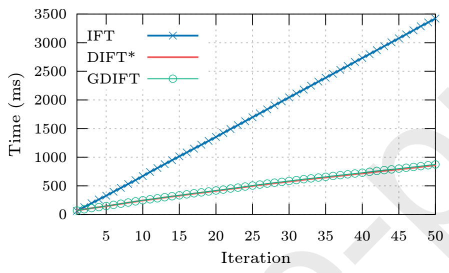
As future work, we intend to explore the differential computation of the *Oriented Image Foresting Transform* (OIFT) [11, 14] in 3D images by the GDIFT algorithm, by adding the support for other types of NMI functions, and to explore its usage with layered graphs in the HLOIFT method [16], to incorporate hierarchical constraints.



(a) Divergence of the cost map V



(b) Divergence of the label map L



(c) The mean accumulated time

Figure 12: a-b) Mean of the percentage of pixels divergence in cost and label maps between results of differential algorithms and IFT for 30 executions. c) The mean accumulated time along the iterations of the methods.

Appendix A. Definitions

A path $\pi = \langle t_1, \ldots, t_n \rangle$ is a sequence of adjacent pixels (i.e., $\langle t_i, t_{i+1} \rangle \in \mathcal{A}$, $i = 1, 2, \ldots, n-1$). The notation $\pi_t = \langle t_1, \ldots, t_n = t \rangle$ indicates a path with terminus at a pixel t and the notation $\pi_{r \rightsquigarrow t} = \langle t_1 = r, \ldots, t_n = t \rangle$ indicates a path from a pixel r (origin/root) to a pixel t (destination node). A path is also said *trivial* if it is composed of a single node, that is, when $\pi_t = \langle t \rangle$. A path $\pi_t = \pi_s \cdot \langle s, t \rangle$ indicates the extension of a path π_s by an arc $\langle s, t \rangle$. $\Pi_t(G)$ indicates the set of all paths with terminus at a node t in a graph G and $\Pi(G) = \bigcup_{t \in \mathcal{I}} \Pi_t(G)$ indicates the set of all possible paths in G.

A *predecessor map* is a function $P : \mathcal{I} \rightarrow \mathcal{I} \cup \{nil\}$ that assigns to each pixel t in $\mathcal{I}$ either some other adjacent pixel or a distinctive marker $nil \notin \mathcal{I}$ — in which case t is said to be a *root* of the map. A *spanning forest* is a predecessor map which contains no cycles. For any pixel $t \in \mathcal{I}$, a spanning forest P defines a path π_t^P recursively as $\langle t \rangle$, when $P(t) = nil$, or $\pi_s^P \cdot \langle s, t \rangle$, when $P(t) = s \neq nil$.

A *root map* $R : \mathcal{I} \rightarrow \mathcal{I}$ associates each pixel $s \in \mathcal{I}$ with the origin of the path π_s^P in the forest generated by IFT. That is, $\pi_s^P = \langle R(s), \ldots, P(P(s)), P(s), s \rangle$ and $P(R(s)) = nil$. The root map takes the form $R : \mathcal{I} \rightarrow \mathcal{S}$, when we constrain the computed paths to start from a given set $\mathcal{S}$ of seed pixels.

A *label map* is a function $L : \mathcal{I} \rightarrow \{0, \ldots, l\}$, where $l + 1$ is the number of image objects. In image segmentation from markers by IFT, we have a seed set $\mathcal{S} \subset \mathcal{I}$ and a partial labeling function $\lambda : \mathcal{S} \rightarrow \{0, \ldots, l\}$, which defines the initial label of each seed. This partial image labeling is then propagated to all the remaining pixels in $\mathcal{I} \backslash \mathcal{S}$, which can be initialized with an arbitrary labeling, since their labels will be overwritten in L. At the end, the label map L associates each pixel $s \in \mathcal{I}$ with the label of its root $r = R(s)$ (i.e., $\lambda(r) = L(r) = L(s)$).

A *connectivity function* $f : \Pi(G) \rightarrow \mathcal{V}$ computes a value $f(\pi_t)$ for any path π_t, in some totally ordered set $\mathcal{V}$ of cost values. A path π_t is *optimum* if $f(\pi_t) \leq f(\tau_t)$ for any other path τ_t in G.

Appendix B. IFT algorithm

In this section we present the IFT algorithm in detail using the standard notation from Appendix A. The IFT algorithm is a generalization of Dijkstra's algorithm for multiple sources and more general connectivity functions [1, 2]. As shown by Frieze [40], the original proof of Dijkstra's algorithm can be applied to monotonic-incremental (MI) connectivity functions — i.e., functions $f(\pi_s \cdot \langle s, t \rangle) = f(\pi_s) \odot \langle s, t \rangle$, where $f(\langle t \rangle)$ is given by an arbitrary handicap cost and $\odot : \mathcal{V} \times \mathcal{A} \rightarrow \mathcal{V}$ is a binary operation that satisfies two conditions:

- $x' \geq x \Rightarrow x' \odot \langle s, t \rangle \geq x \odot \langle s, t \rangle$ and
- $x \odot \langle s, t \rangle \geq x$,

for any x, $x' \in \mathcal{V}$ and any $\langle s, t \rangle \in \mathcal{A}$. An essential feature of MI functions is that $\odot$ depends only on the cost $f(\pi_s)$ of π_s and not on any other property of this path.

For applicable connectivity function f [2], the *image foresting transform* (IFT) takes an image graph $G = \langle \mathcal{I}, \mathcal{A} \rangle$ and returns an *optimum-path forest* P — i.e., a spanning forest where all paths π_t^P for $t \in \mathcal{I}$ are optimum. The computed costs of paths π_s^P in the computed forest P for each pixel $s \in \mathcal{I}$ are stored in a path-cost map $V : \mathcal{I} \rightarrow \mathcal{V}$ — i.e., $V(s) = f(\pi_s^P)$.

The general IFT algorithm is presented in Algorithm 3. The IFT ensures that the computed cost map V converges to $V_{opt}(t) = \min_{\pi_t \in \Pi_t(G)}\{f(\pi_t)\}$ for MI and non-MI (NMI) functions that satisfy the conditions in [2]. Algorithm 3 is also optimized for handling infinite costs, by storing only the nodes with finite-cost path in the priority queue Q, assuming that $V_{opt}(t) < +\infty$ for all $t \in \mathcal{I}$. In the case of NMI functions, the *state test* $S(t) \neq 1$ (Line 11) is required to ensure that each pixel will be removed from Q (Line 8) just once during the execution of the algorithm. In the case of MI functions, this state test is considered only a performance improvement [1] to avoid the path-cost computation in Line 12.

Algorithm 3. – IFT ALGORITHM

INPUT:	Image graph $\langle \mathcal{I}, \mathcal{A} \rangle$, path-cost function f and an initial labeling function $\lambda : \mathcal{I} \rightarrow \{0, \ldots, l\}$.
OUTPUT:	The label map $L : \mathcal{I} \rightarrow \{0, \ldots, l\}$, root map $R : \mathcal{I} \rightarrow \mathcal{I}$, path-cost map $V : \mathcal{I} \rightarrow \mathcal{V}$ and the spanning forest $P : \mathcal{I} \rightarrow \mathcal{I} \cup \{nil\}$.
AUXILIARY:	Priority queue Q, variable *tmp* and an array of status $S : \mathcal{I} \rightarrow \{0, 1\}$, where $S(t) = 1$ for processed nodes and $S(t) = 0$ for unprocessed nodes.

```
1. For each $t \in \mathcal{I}$, do
2.     Set $S(t) \leftarrow 0$.
3.     Set $P(t) \leftarrow nil$ and $V(t) \leftarrow f(\langle t \rangle)$.
4.     Set $R(t) \leftarrow t$ and $L(t) \leftarrow \lambda(t)$.
5.     If $V(t) \neq +\infty$, then
6.         insert $t$ in $Q$.
7. While $Q \neq \emptyset$, do
8.     Remove $s$ from $Q$ such that $V(s) = \min_{\forall t \in Q}\{V(t)\}$.
9.     Set $S(s) \leftarrow 1$.
```

```
10.   For each node $t$ such that $\langle s,t\rangle \in \mathcal{A}$, do
11.       If $S(t) \neq 1$, then
12.           Compute $tmp \leftarrow f(\pi_s^P \cdot \langle s,t\rangle)$.
13.           If $tmp < V(t)$, then
14.               If $t \in Q$, then remove $t$ from $Q$.
15.               Set $P(t) \leftarrow s$ and $V(t) \leftarrow tmp$.
16.               Set $R(t) \leftarrow R(s)$ and $L(t) \leftarrow L(s)$.
17.               Insert $t$ in $Q$.
```

Appendix C. DIFT algorithm

Let a sequence of IFTs be represented as $\langle IFT_{(\mathcal{S}_1)}, IFT_{(\mathcal{S}_2)}, \ldots, IFT_{(\mathcal{S}_n)}\rangle$, where n is the total number of IFT executions on the image. At each execution, the seed set $\mathcal{S}_i$ is modified by adding, removing, or by combining the addition and removal of seeds to obtain a new set $\mathcal{S}_{i+1}$. We define a scene $\mathcal{C}_i$ as the set of maps $\mathcal{C}_i = \{P_i, V_i, L_i, R_i\}$, resulting from the *ith* iteration in a sequence of IFTs. In applications, $\mathcal{S}_{i+1}$ is usually defined based on the previous scene output $\mathcal{C}_i$, so that we can not know beforehand the final set of seeds $\mathcal{S}_n$.

The DIFT algorithm [12] allows to efficiently compute a scene $\mathcal{C}_i$ from the previous scene $\mathcal{C}_{i-1}$, a set $\Delta^+_{\mathcal{S}_i} = \mathcal{S}_i \setminus \mathcal{S}_{i-1}$ of new seeds for addition, and a set $\Delta^-_{\mathcal{S}_i} = \mathcal{S}_{i-1} \setminus \mathcal{S}_i$ of seeds marked for removal. In the execution flow by DIFT, after the first execution of $IFT_{(\mathcal{S}_1)}$, we have that the scenes $\mathcal{C}_i$ for $i \geq 2$ are calculated based on the scene $\mathcal{C}_{i-1}$, taking advantage of the trees that were computed in the previous iteration, thus reducing the processing time. Hence, we have the following differential flow: $\langle IFT_{(\mathcal{S}_1)}, DIFT_{(\Delta^+_{\mathcal{S}_2},\Delta^-_{\mathcal{S}_2},\mathcal{C}_1)}, DIFT_{(\Delta^+_{\mathcal{S}_3},\Delta^-_{\mathcal{S}_3},\mathcal{C}_2)}, \ldots, DIFT_{(\Delta^+_{\mathcal{S}_n},\Delta^-_{\mathcal{S}_n},\mathcal{C}_{n-1})}\rangle$.

The differential flow must obtain a valid result, obtainable by $IFT_{(\mathcal{S}_n)}$. That is, $IFT_{(\mathcal{S}_n)}$ can have several possible outcomes depending on the order of processing of the nodes and arcs in case of draws. The result of the differential flow should correspond to one of them, so differences in the final labeling should be restricted only within the tie zones. As a consequence, when the result of

$IFT_{(\mathcal{S}_n)}$ has as output a unique cost map V_n (optimal or not optimal), the differential flow must reproduce exactly the same cost map. Since for MI functions, the IFT algorithm generates an optimal cost map that is unique, the differential flow for MI functions should generate the same optimal cost map.

The DIFT algorithm was originally designed exclusively for MI functions [12]. Its insertion and removal cases are explained below. In all cases, the maps of the current scene $\mathcal{C}_i = \{P_i, V_i, L_i, R_i\}$, for $i > 1$, are initialized as being the maps from the previous scene $\mathcal{C}_{i-1}$.

- **Addition of markers:** The insertion of new seeds into an existing optimal path forest can be done by inserting the new seeds $r' \in \Delta^+_{\mathcal{S}_i}$ into the priority queue Q (currently empty) with its trivial path cost and executing the main loop of Algorithm 3. Let's initially consider the same rules of Algorithm 3 (Lines 12-17) to process the extension of a path. Suppose the insertion of a new seed r' during an iteration, such that $\pi_{r' \rightsquigarrow s}$ becomes an optimum path to a pixel s (possibly $s = r'$). If the new path through s extended by the arc $\langle s, t\rangle$ offers a lower cost $f(\pi_{r' \rightsquigarrow s} \cdot \langle s, t\rangle)$ to t than its current one in $V_i(t)$, then the path π^P_t will be updated as $\pi_{r' \rightsquigarrow s} \cdot \langle s, t\rangle$ and t will be inserted in Q. However, if the offered cost $f(\pi_{r' \rightsquigarrow s} \cdot \langle s, t\rangle)$ is the same as its current value in $V_i(t)$ the segmentation label $L_i(t)$ will not be updated and t will not be reinserted in Q. A problem arises when in the *(i-1)th* iteration of the DIFT algorithm, we have $P_{i-1}(t) = s$. In this case, we will have an inconsistency of the maps of roots ($r' = R_i(P_i(t)) \neq R_i(t) = R_{i-1}(t)$) and labels ($L_i(r') = L_i(P_i(t)) \neq L_i(t) = L_{i-1}(R_{i-1}(t))$), because these maps will be outdated referring to an old root $R_{i-1}(t)$ of the previous execution. Hence, in order to fix this problem, a predecessor test $P_i(t) = s$ (Line 16 of Algorithm 4) was proposed in [12] to ensure that the maps will be updated.

- **Removal of markers:** In the case of a root $r \in \Delta^-_{\mathcal{S}_i}$ marked for removal, the entire tree of the seed r must be removed, creating a set of frontier pixels $F = \{t : \langle t, s\rangle \in \mathcal{A} \wedge r = R(s) \wedge R(t) \neq R(s)\}$. These frontier pixels will be inserted in the priority queue to start a new dispute for the conquest of the removed area.

- **Simultaneous addition and removal of markers:** In this case, the removal is executed first, defining the frontier pixels. The new markers are then initialized, removing from F all new markers that eventually also belong to it.

Therefore we have the following algorithms.

Algorithm 4. – DIFT ALGORITHM

INPUT:	Image graph $\langle \mathcal{I}, \mathcal{A}\rangle$, path-cost function f, the set $\Delta^+_{\mathcal{S}}$ of seeds for addition, set $\Delta^-_{\mathcal{S}}$ of seeds for removal, the maps L, R, V and P initialized with the result from the previous IFT/DIFT execution, and an initial labeling function $\lambda : \Delta^+_{\mathcal{S}} \rightarrow \{0, \ldots, l\}$ for the new seeds.
OUTPUT:	The updated maps L, R, V and P.
AUXILIARY:	Priority queue Q, variable *tmp*, the set of frontier pixels F, and an array of status $S : \mathcal{I} \rightarrow \{0, 1\}$, initialized as $S(t) = 0$ for all $t \in \mathcal{I}$, where $S(t) = 1$ for processed nodes and $S(t) = 0$ for unprocessed nodes.

```
1.  Set $Q \leftarrow \emptyset$.
2.  If $\Delta^-_{\mathcal{S}} \neq \emptyset$, then
3.      $(L, R, V, P, F) \leftarrow$ DIFT-TreeRemoval $(\langle \mathcal{I}, \mathcal{A}\rangle, \Delta^-_{\mathcal{S}}, L, R, V, P)$
4.      $F \leftarrow F \setminus \Delta^+_{\mathcal{S}}$
5.      For each $t \in F$, do insert $t$ in $Q$.
6.  For each $s \in \Delta^+_{\mathcal{S}}$, do
7.      If $f(\langle s\rangle) < V(s)$, then
8.          Set $V(s) \leftarrow f(\langle s\rangle)$, $L(s) \leftarrow \lambda(s)$, $R(s) \leftarrow s$, $P(s) \leftarrow nil$
9.          Insert $s$ in $Q$.
10. While $Q \neq \emptyset$, do
11.     Remove $s$ from $Q$ such that $V(s) = \min_{\forall t \in Q}\{V(t)\}$.
12.     Set $S(s) \leftarrow 1$.
13.     For each node $t$ such that $\langle s, t\rangle \in \mathcal{A}$, do
```

```
14.         If $S(t) \leq 1$, then
15.             Compute $tmp \leftarrow f(\pi_s^P \cdot \langle s,t \rangle)$.
16.             If $tmp < V(t)$ or $s = P(t)$, then
17.                 If $t \in Q$, then remove $t$ from $Q$.
18.                 Set $P(t) \leftarrow s$ and $V(t) \leftarrow tmp$.
19.                 Set $R(t) \leftarrow R(s)$ and $L(t) \leftarrow L(s)$.
20.                 Insert $t$ in $Q$.
```

The condition $S(t) \leq 1$ at Line 14 of Algorithm 4, which is always true, is used only to mark a point for modifications as discussed in Section 3. The following auxiliary algorithm for marker removal (Algorithm 5) assumes that the graph is symmetric. A graph G is symmetric if for all $\langle s,t \rangle \in \mathcal{A}$, the pair $\langle t,s \rangle$ is also an arc of G.

Algorithm 5. – DIFT-TreeRemoval Procedure

Input:	Image graph $\langle \mathcal{I}, \mathcal{A} \rangle$, the set of seeds to be removed $\Delta_{\mathcal{S}}^{-}$, and the maps L, R, V and P.
Output:	The updated maps L, R, V and P, and the set of frontier pixels F.
Auxiliary:	FIFO Queue T.

```
1.  Set $F \leftarrow \emptyset$.
2.  For each $r \in \Delta_{\mathcal{S}}^{-}$, do
3.      Set $V(r) \leftarrow +\infty$ and $P(r) \leftarrow nil$.
4.      Insert $r$ in $T$.
5.  While $T \neq \emptyset$, do
6.      Remove $s$ from $T$.
7.      For each node $t$ such that $\langle s,t \rangle \in \mathcal{A}$, do
8.          If $s = P(t)$, then
9.              Set $V(t) \leftarrow +\infty$ and $P(t) \leftarrow nil$.
10.             Insert $t$ in $T$.
11.         Else If $R(t) \notin \Delta_{\mathcal{S}}^{-}$, then $F \leftarrow F \cup \{t\}$.
```

Acknowledgment

The authors would like to thank CNPq (308985/2015-0, 313554/2018-8, 486083/2013-6, 303808/2018-7, 486988/2013-9, 166631/2018-3, 465446/2014-0, FINEP 1266/13), São Paulo Research Foundation (FAPESP 2011/50761-2, 2016/21591-5, 2014/50937-1), NAP eScience - PRP - USP, and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Finance Code 001 for funding (88887.136422/2017-00). This research is also part of the FAPESP Thematic Research Project (proc. 2014/12236-1).

References

[1] A. X. Falcão, J. Stolfi, R. de Alencar Lotufo, The image foresting transform: Theory, algorithms, and applications, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (1) (2004) 19–29.

[2] K. Ciesielski, A. Falcão, P. Miranda, Path-value functions for which dijkstra's algorithm returns optimal mapping, Journal of Mathematical Imaging and Vision 60 (7) (2018) 1025–1036. `doi:10.1007/s10851-018-0793-1`.

[3] A. Falcão, B. da Cunha, R. Lotufo, Design of connected operators using the image foresting transform, in: Proceedings of SPIE on Medical Imaging, Vol. 4322, 2001, pp. 468–479.

[4] A. X. Falcão, J. K. Udupa, F. K. Miyazawa, An ultra-fast user-steered image segmentation paradigm: live wire on the fly, IEEE Transactions on Medical Imaging 19 (1) (2000) 55–62. `doi:10.1109/42.832960`.

[5] A. Falcão, L. da F. Costa, B. da Cunha, Multiscale skeletons by image foresting transform and its application to neuromorphometry, Pattern Recognition 35 (7) (2002) 1571 – 1582. `doi:10.1016/S0031-3203(01)00148-0`.

[6] A. X. Falcão, C. Feng, J. Kustra, A. Telea, Chapter 2 - multiscale 2D medial axes and 3D surface skeletons by the image foresting transform, in: P. K. Saha, , G. Borgefors, , G. S. d. Baja (Eds.), Skeletonization, Academic Press, 2017, pp. 43 – 70. doi:10.1016/B978-0-08-101291-8.00003-1.

[7] R. A. Lotufo, A. X. Falcão, F. A. Zampirolli, IFT-watershed from gray-scale marker, in: Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on, IEEE, 2002, pp. 146–152.

[8] L. Rocha, F. Cappabianco, A. X. Falcão, Data clustering as an optimum-path forest problem with applications in image analysis, International Journal of Imaging Systems and Technology (IJIST) 19 (2009) 50–68.

[9] F. Cappabianco, A. X. Falcão, C. Yasuda, J. Udupa, Brain tissue MR-image segmentation via optimum-path forest clustering, Computer Vision and Image Underst. 116 (10) (2012) 1047–1059.

[10] P. A. V. Miranda, A. X. Falcão, Links between image segmentation based on optimum-path forest and minimum cut in graph, Journal of Mathematical Imaging and Vision 35 (2) (2009) 128–142. doi:10.1007/s10851-009-0159-9.
URL http://dx.doi.org/10.1007/s10851-009-0159-9

[11] P. A. Miranda, L. A. Mansilla, Oriented image foresting transform segmentation by seed competition, IEEE Transactions on Image Processing 23 (1) (2014) 389–398. doi:10.1109/TIP.2013.2288867.

[12] A. X. Falcão, F. P. Bergo, Interactive volume segmentation with differential image foresting transforms, IEEE Transactions on Medical Imaging 23 (9) (2004) 1100–1108.

[13] L. A. C. Mansilla, P. A. V. Miranda, Image segmentation by oriented image foresting transform with geodesic star convexity, in: Computer Analysis of Images and Patterns, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 572–579.

[14] L. A. C. Mansilla, P. A. V. Miranda, Image segmentation by oriented image foresting transform: Handling ties and colored images, in: 2013 18th International Conference on Digital Signal Processing (DSP), 2013, pp. 1–6. doi:10.1109/ICDSP.2013.6622806.

[15] L. A. C. Mansilla, P. A. V. Miranda, F. A. M. Cappabianco, Oriented image foresting transform segmentation with connectivity constraints, in: 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 2554–2558. doi:10.1109/ICIP.2016.7532820.

[16] L. Leon, P. Miranda, Multi-object segmentation by hierarchical layered oriented image foresting transform, in: Conference on Graphics, Patterns and Images (SIBGRAPI), 2017, pp. 79–86. doi:10.1109/SIBGRAPI.2017.17.

[17] L. A. Mansilla, P. A. V. Miranda, F. A. Cappabianco, Image segmentation by image foresting transform with non-smooth connectivity functions, in: 2013 XXVI Conference on Graphics, Patterns and Images, IEEE, 2013, pp. 147–154.

[18] R. Strand, K. Ciesielski, F. Malmberg, P. Saha, The minimum barrier distance, Computer Vision and Image Understanding 117 (2013) 429–437.

[19] A. A. Tavares, P. A. Miranda, T. V. Spina, A. X. Falcão, A supervoxel-based solution to resume segmentation for interactive correction by differential image-foresting transforms, in: Mathematical Morphology and Its Applications to Signal and Image Processing, 2017, pp. 107–118. doi:10.1007/978-3-319-57240-6.

[20] A. Tavares, H. Bejar, P. Miranda, Seed robustness of oriented image foresting transform: Core computation and the robustness coefficient, in: Mathematical Morphology and Its Applications to Signal and Image Processing, Springer, 2017, pp. 119–130.

[21] P. A. V. Miranda, A. X. Falcão, T. V. Spina, Riverbed: A novel user-steered image segmentation method based on optimum boundary tracking, IEEE Transactions on Image Processing 21 (6) (2012) 3042–3052. `doi:10.1109/TIP.2012.2188034`.

[22] M. A. Condori, L. A. Mansilla, P. A. Miranda, Bandeirantes: A graph-based approach for curve tracing and boundary tracking, in: Mathematical Morphology and Its Applications to Signal and Image Processing, 2017, pp. 95–106. `doi:10.1007/978-3-319-57240-6`.

[23] E. B. Alexandre, A. S. Chowdhury, A. X. Falcão, P. A. V. Miranda, IFT-SLIC: A general framework for superpixel generation based on simple linear iterative clustering and image foresting transform, in: 2015 28th SIBGRAPI Conference on Graphics, Patterns and Images, 2015, pp. 337–344. `doi:10.1109/SIBGRAPI.2015.20`.

[24] J. E. Vargas-Muñoz, A. S. Chowdhury, E. B. Alexandre, F. L. Galvão, P. A. V. Miranda, A. X. Falcão, An iterative spanning forest framework for superpixel segmentation, IEEE Transactions on Image ProcessingTo appear. `doi:10.1109/TIP.2019.2897941`.

[25] F. L. Galvão, A. X. Falcão, A. S. Chowdhury, Risf: Recursive iterative spanning forest for superpixel segmentation, in: 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2018, pp. 408–415. `doi:10.1109/SIBGRAPI.2018.00059`.

[26] F. de C. Belém, S. J. F. Guimarães, A. X. Falcão, Superpixel segmentation by object-based iterative spanning forest., in: 23rd Iberoamerican Congress on Pattern Recognition (CIARP), 2018, to appear.

[27] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Ssstrunk, SLIC superpixels compared to state-of-the-art superpixel methods, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (11) (2012) 2274–2282. `doi:10.1109/TPAMI.2012.120`.

[28] J. Chen, Z. Li, B. Huang, Linear spectral clustering superpixel, IEEE Trans. on Image Processing 26 (7) (2017) 3317–3330.

[29] M. Liu, O. Tuzel, S. Ramalingam, R. Chellappa, Entropy rate superpixel segmentation, in: Proc. IEEE International Conf. on Computer Vision and Pattern Recognition, (CVPR), Colorado Springs, CO, USA, 2011, pp. 2097–2104.

[30] J. Shen, Y. Du, W. Wang, X. Li, Lazy random walks for superpixel segmentation, IEEE Trans. Image Processing 23 (4) (2014) 1451–1462.

[31] V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, E. Decencière, Waterpixels, IEEE Trans. on Image Processing 24 (11) (2015) 3707–3716.

[32] M. A. T. Condori, F. A. M. Cappabianco, A. X. Falcão, P. A. V. D. Miranda, Extending the differential image foresting transform to root-based path-cost functions with application to superpixel segmentation, in: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2017, pp. 7–14. `doi:10.1109/SIBGRAPI.2017.8`.

[33] D. J. Silva, W. A. L. Alves, A. Morimitsu, R. F. Hashimoto, Efficient incremental computation of attributes based on locally countable patterns in component trees, in: 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3738–3742. `doi:10.1109/ICIP.2016.7533058`.

[34] L. Grady, Random walks for image segmentation, Trans. on PAMI 28 (11) (2006) 1768–1783.

[35] Y. Boykov, M.-P. Jolly, Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images, in: International Conference on Computer Vision (ICCV), Vol. 1, 2001, pp. 105–112.

[36] N. Moya, Interactive segmentation of multiple 3D objects in medical images by optimum cuts in graph, Master's thesis, Institute of Computing, University of Campinas, Brazil (May 2015).

[37] L. Duan, F. Lafarge, Image partitioning into convex polygons, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3119–3127. doi:10.1109/CVPR.2015.7298931.

[38] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, ACM Trans. Graph. 23 (3) (2004) 309–314. doi:10.1145/1015706.1015720.

[39] L. Perkins, Terrain analysis in real-time strategy games: An integrated approach to choke point detection and region decomposition, in: Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE'10, AAAI Press, 2010, pp. 168–173.

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

Highlights

- We extend the DIFT algorithm for non-monotonically incremental functions with root-based increases.
- Generalized DIFT has been successfully used as the core part of some modern superpixels methods with state-of-the-art results.
- GDIFT shows considerable efficiency gains over the sequential flow of IFTs for the generation of superpixels, also avoiding inconsistencies in image segmentation.