

# Image Foresting Transform: On-the-fly Computation of Segmentation Boundaries

Filip Malmberg

Centre for Image Analysis,  
Uppsala University, Sweden  
[filip.malmberg@cb.uu.se](mailto:filip.malmberg@cb.uu.se)  
<http://www.cb.uu.se>

**Abstract.** The Image Foresting Transform (IFT) is a framework for seeded image segmentation, based on the computation of minimal cost paths in a discrete representation of an image. In two recent publications, we have shown that the segmentations obtained by the IFT may be improved by refining the segmentation locally around the boundaries between segmented regions. Since these methods operate on a small subset of the image elements only, they may be implemented efficiently if the set of boundary elements is known. Here, we show that this set may be obtained on-the-fly, at virtually no additional cost, as a by-product of the IFT algorithm.

**Keywords:** Interactive Image Segmentation, Image Foresting Transform.

## 1 Introduction

Image segmentation, the process of identifying and separating relevant objects and structures in an image, is a fundamental problem in image analysis. Accurate segmentation of objects of interest is often required before further processing and analysis can be performed. Despite years of active research, fully automatic segmentation of arbitrary images remains an unsolved problem.

Seeded segmentation methods attempt to solve the segmentation problem in the presence of prior knowledge in the form of a partial segmentation. Given an image where a small subset of the image elements (called *seed-points*) have been assigned correct segmentation labels (e.g., object or background), an automatic algorithm completes the labeling for all image elements. The seed-points may be provided either by some automatic pre-processing algorithm, or by a human user in an interactive setting. Many different algorithms for seeded segmentation have been proposed ranging from classical seeded region growing [1, 11], through to the more recent minimal graph cuts [3], random walks [7], and image foresting transform (IFT) [6] approaches. Here, we focus on the IFT approach.

In the IFT, the image is represented by an edge-weighted graph. Each image element corresponds to a node in the graph, and adjacent image elements are connected by graph edges. Segmentation is performed by assigning to each node

the label of the closest seed-point, as determined by the *minimum cost path* from the node to the set of seed-points.

The IFT can be computed using Dijkstra’s algorithm [4], slightly modified to allow multiple seed-points [6]. An efficient implementation of Dijkstra’s algorithm, and so of the IFT, requires  $\mathcal{O}(|\mathcal{I}|)$  operations, where  $|\mathcal{I}|$  is the number of image elements, for a sparse graph with bounded integer path costs.

In interactive segmentation applications, a user often adds or removes seed-points to refine an existing segmentation. In [5], it was shown that seed-points can be added to, or removed from, an existing IFT solution, without recomputing the entire solution. This modified algorithm, called the differential IFT (DIFT), gives a significant reduction of the total time required for interactive segmentation. In a differential implementation, the computation time required for each editing operation is proportional to the number of image elements that are modified by the operation. This number is usually much smaller than  $|\mathcal{I}|$ . For typical segmentation scenarios, the DIFT reduces the computation time required for editing operations by a factor between 10 and 20, compared to the IFT [5]. The DIFT is described in detail in Section 3.

In two recent publications [9, 10], we show that the segmentations obtained by the IFT may be improved by performing local operations around the boundaries of the segmented regions. These methods are reviewed, and examples of their application are given, in Section 4.

The boundary of a segmentation is here defined as the set of image elements adjacent to at least one element with a different label. This set is usually much smaller than the set of image elements. Since the methods proposed in [9] and [10] operate only in a small region around the boundary, they may be computed efficiently if the set of boundary elements is known.

For any given image element, it is easy to check if the element is part of the segmentation boundary by comparing the label of the element to the labels of its neighbors. Thus, a trivial algorithm for obtaining the boundary is to iterate over all image elements and check if the element is on the boundary. This however, requires  $\mathcal{O}(|\mathcal{I}|)$  operations, and thus the advantage of the differential implementation is lost. Here, we show that the set of boundary elements may be computed on the fly, at virtually no additional cost, as a by-product of the DIFT algorithm<sup>1</sup>. This makes it possible to implement the methods proposed in [9, 10] efficiently in conjunction with the DIFT, thereby making them more attractive for interactive segmentation.

## 2 Background

### 2.1 Images and graphs

An *image*  $\mathbf{I}$  is a pair  $(\mathcal{I}, I)$  consisting of a set  $\mathcal{I}$  of image elements and a mapping  $I$  that assigns to each image element  $p \in \mathcal{I}$  an element in some arbitrary set,

<sup>1</sup> We note that the concept of extracting segmentation boundaries while computing the DIFT was previously investigated by Audigier et al. [2], for the purpose of visualizing the segmentation results.



**Fig. 1.** Segmentation of the liver in a slice from an MR volume image. (Left) Original image. (Middle) Segmentation obtained with the IFT, from user defined seed-points shown in gray. (Right) Boundary of the segmentation, computed on-the-fly as a by-product of the IFT algorithm.

typically a subset of  $\mathbb{Z}^n$  or  $\mathbb{R}^n$  (e.g.,  $\mathcal{I} \subset \mathbb{Z}^n$  and  $I : \mathcal{I} \rightarrow [0, 255]$ ). We associate an image with an *adjacency function*  $\mathcal{N}$  that maps each image element  $p \in \mathcal{I}$  to a set  $\mathcal{N}(p) \subseteq \mathcal{I}$  of *adjacent* image elements. We require the adjacency function to be symmetric, so that  $p \in \mathcal{N}(q) \iff q \in \mathcal{N}(p)$  for all  $p, q \in \mathcal{I}$ . An image, together with an adjacency function, may be interpreted as a graph, whose nodes are the image elements and whose edges are all ordered pairs of image elements  $p, q \in \mathcal{I}$  such that  $q \in \mathcal{N}(p)$ .

For each ordered pair of adjacent image elements  $p$  and  $q$ , we assign a real valued, non-negative, *edge weight*  $w(p, q)$ . The edge weights represent local *dissimilarity*, i.e.,  $p$  and  $q$  are strongly connected if  $w(p, q)$  is close to 0. Typically, edge weights are computed from local image features such as intensity or gradient magnitude.

## 2.2 Paths and path costs

A *path*  $\pi = \langle p_1, p_2, \dots, p_k \rangle$  of length  $|\pi| = k - 1$  is a sequence  $p_1, p_2, \dots, p_k$  of image elements such that  $p_{i+1} \in \mathcal{N}(p_i)$ . We denote the *origin*  $p_1$  and the *destination*  $p_k$  of  $\pi$  by  $org(\pi)$  and  $dst(\pi)$ , respectively. If  $\pi$  and  $\tau$  are paths such that  $dst(\pi) = org(\tau)$ , we denote by  $\pi \cdot \tau$  the concatenation of the two paths.

The *cost* of a path is denoted  $f(\pi)$ . This cost is typically a function of the edge weights along the path, e.g., the sum of all the edge weights along the path or the maximum edge weight along the path. The maximum possible cost of a path is denoted  $+\infty$ .

A path  $\pi$  is a *minimum cost path* if  $f(\pi) \leq f(\tau)$  for any other path  $\tau$  with  $org(\tau) = org(\pi)$  and  $dst(\tau) = dst(\pi)$ . In general, a minimum cost path is not unique. The set of minimum cost paths between two image elements  $p$  and  $q$  is denoted  $\pi_{min}(p, q)$ .

The definition of a minimum cost path between two sets of image elements is analogous. For two sets  $A \subseteq \mathcal{I}$  and  $B \subseteq \mathcal{I}$ ,  $\pi$  is a path between  $A$  and  $B$  if  $org(\pi) \in A$  and  $dst(\pi) \in B$ . If  $f(\pi) \leq f(\tau)$  for any other path  $\tau$  between  $A$  and

$B$ , then  $\pi$  is a minimum cost path between  $A$  and  $B$ . The set of minimum cost paths between  $A$  and  $B$  is denoted  $\pi_{min}(A, B)$ .

### 2.3 Spanning forests

A *predecessor map* is a mapping  $P$  that assigns to each image element  $p \in \mathcal{I}$  either an element  $q \in \mathcal{N}(p)$ , or  $\emptyset$ . For any  $p \in \mathcal{I}$ , a predecessor map  $P$  defines a path  $P^*(p)$  recursively as

$$P^*(p) = \begin{cases} \langle p \rangle & \text{if } P(p) = \emptyset \\ P^*(P(p)) \cdot \langle P(p), p \rangle & \text{otherwise} \end{cases} .$$

We denote by  $P^0(p)$  the first element of  $P^*(p)$ . A *spanning forest* is a predecessor map that contains no cycles, i.e.,  $|P^*(p)|$  is finite for all  $p \in \mathcal{I}$ . If  $P^*(p) = \emptyset$ , then  $p$  is a *root* of  $P$ .

### 2.4 Image Segmentation

A *segmentation* of an image  $\mathbf{I}$  is a mapping  $\mathcal{L}$  that assigns to each image element  $p \in \mathcal{I}$  an element in some arbitrary set of *labels*, e.g.,  $\mathcal{L} : \mathcal{I} \rightarrow \{\text{object}, \text{background}\}$ . The *boundary*  $\partial\mathcal{L} \subseteq \mathcal{I}$  of a segmentation is defined as

$$\partial\mathcal{L} = \mathcal{I} \setminus \{p \mid \mathcal{L}(p) = \mathcal{L}(q) \text{ for all } q \in \mathcal{N}(p)\} ,$$

i.e., an image element belongs to the boundary if at least one of its neighbors has a different label.

### 2.5 The Image Foresting Transform

Given an image  $\mathbf{I}$ , a path cost function  $f$ , an adjacency function  $\mathcal{N}$  and a set of seed-points  $S \in \mathcal{I}$  with corresponding labels, the IFT computes a spanning forest  $P$  such that  $P^*(p) \in \pi_{min}(p, S)$  for all image elements  $p \in \mathcal{I}$ . During this process, a *cost map*  $C$  and a segmentation  $\mathcal{L}$  are built, such that  $C(p) = f(\pi_{min}(p, S))$  and  $\mathcal{L}(p) = \mathcal{L}(P^0(p))$ . A triple  $(P, C, \mathcal{L})$  that satisfies these properties is a *solution* of the IFT with respect to  $S$ .

## 3 The Differential Image Foresting Transform

The DIFT allows seed-points to be added to, or removed from, an existing IFT solution in an efficient way. Given a solution of the IFT with respect to a set of seed-points  $S$ , the DIFT algorithm computes a solution with respect to another set of seed-points  $S'$ . The difference between  $S$  and  $S'$  can be written in terms of the following two sets:  $S^+ = S' \setminus S$  and  $S^- = S \setminus S'$ . It holds that  $S' = (S \setminus S^-) \cup S^+$ , i.e.,  $S'$  can be obtained from  $S$  by adding all elements in  $S^+$  and removing all elements in  $S^-$ .

**Algorithm 1: DIFT**


---

**Input:** Image  $I$ , adjacency function  $\mathcal{N}$ , path cost function  $f$ , solution  $\{C, P, \mathcal{L}\}$ , set  $S$  of old seed-points, and set  $S'$  of new seed-points.  
**Output:**  $C, \mathcal{L}, P$ , and  $B$ .  
**Auxiliary:** Three sets of image elements  $Q, T$ , and  $V$ .

- 1 Update the labels of all elements in  $S'$ , according to the user input;
- 2 **foreach**  $p \in S^+$  **do**
- 3    $C(p) \leftarrow 0, P(p) \leftarrow \emptyset$ ;
- 4  $(C, P, F) \leftarrow \text{RemoveSeeds}(C, P, \mathcal{N}, S^-)$ ;
- 5  $Q \leftarrow F \cup S^+, T \leftarrow \emptyset, V \leftarrow \emptyset$ ;
- 6 **while**  $Q \neq \emptyset$  **do**
- 7   Remove  $p$  from  $Q$  such that  $C(p)$  is minimum;
- 8    $B \leftarrow B \setminus \{p\}, V \leftarrow V \cup \{p\}, T \leftarrow T \setminus \{p\}$ ;
- 9   **foreach**  $q \in \mathcal{N}(p)$  **do**
- 10      $cost \leftarrow f(P^*(p) \cdot \langle p, q \rangle)$  ;
- 11     **if**  $q \notin V$  **then**
- 12        $T \leftarrow T \cup q$ ;
- 13     **if**  $q \in V$  and  $\mathcal{L}(p) \neq \mathcal{L}(q)$  **then**
- 14        $B \leftarrow B \cup \{p, q\}$ ;
- 15     **if**  $cost < C(q)$  or  $P(q) = p$  **then**
- 16        $Q \leftarrow Q \cup \{q\}$ ;
- 17        $P(q) \leftarrow \{p\}, \mathcal{L}(q) \leftarrow \mathcal{L}(p)$ , and  $C(q) \leftarrow cost$ ;
- 18 **foreach**  $p \in T$  **do**
- 19    $B \leftarrow B \setminus \{p\}$ ;
- 20   **foreach**  $q \in \mathcal{N}(p)$  **do**
- 21     **if**  $\mathcal{L}(p) \neq \mathcal{L}(q)$  **then**
- 22        $B \leftarrow B \cup \{p, q\}$ ;

---

The procedure for computing the DIFT is given in Algorithm 1. This algorithm is essentially the same as that presented in [5]. However, in addition to the solution  $(C, P, \mathcal{L})$ , Algorithm 1 computes a set  $B$  (for *boundary*) such that  $B = \partial\mathcal{L}$ . See Figure 1. To see that  $B$ , as computed by Algorithm 1, equals the boundary of  $\mathcal{L}$ , we observe that if a node is not inserted into  $Q$  during Algorithm 1, then the label of that node is not changed. Thus, to compute the correct boundary, we only need to update the nodes that pass through  $Q$  during the algorithm, and their neighbors.

Each node is inserted into and removed from  $Q$  at most once. Thus, when a node is removed from  $Q$ , it has already been given its final label. The set  $V$  (for *visited*) is used to keep track of this – each time a node is removed from  $Q$ , it is inserted into  $V$ . On line 13, it holds that  $p \in V$ . Thus, if  $q \in V$ , we can safely compare  $\mathcal{L}(p)$  and  $\mathcal{L}(q)$  to check if  $p$  and  $q$  belong to the boundary.

The set  $T$  (for *touched*) is used to keep track of elements that are adjacent to at least one element in  $V$ , but are not in  $V$  themselves. All nodes that remain in

---

**Procedure RemoveSeeds**

---

**Input:** Cost map  $C$ , predecessor map  $P$ , adjacency function  $\mathcal{N}$ , and set  $S^-$  of seed-points to be removed.

**Output:**  $C, P$ , and set  $F$  of frontier image elements.

**Auxiliary:** FIFO queue  $U$ , and set  $W$  of visited elements.

```

1  $F \leftarrow \emptyset, W \leftarrow \emptyset;$ 
2 foreach  $p \in S^-$  do
3    $\lfloor$  Insert  $p$  in  $U$ ;
4 while  $U$  is not empty do
5   Remove  $p$  from  $U$ ;
6    $C(p) \leftarrow +\infty, P(p) \leftarrow \emptyset;$ 
7    $W \leftarrow W \cup \{p\}, F \leftarrow F \setminus \{p\};$ 
8   foreach  $q \in \mathcal{N}(p)$  do
9     if  $q \notin W$  then
10     $\lfloor F \leftarrow F \cup \{q\};$ 
11    if  $P(q) = p$  then
12     $\lfloor$  Insert  $q$  in  $U$ ;

```

---

$T$  after the termination of the *while*-loop on lines 6-17 also need to be checked for possible inclusion in  $B$ , as is done on lines 18-22.

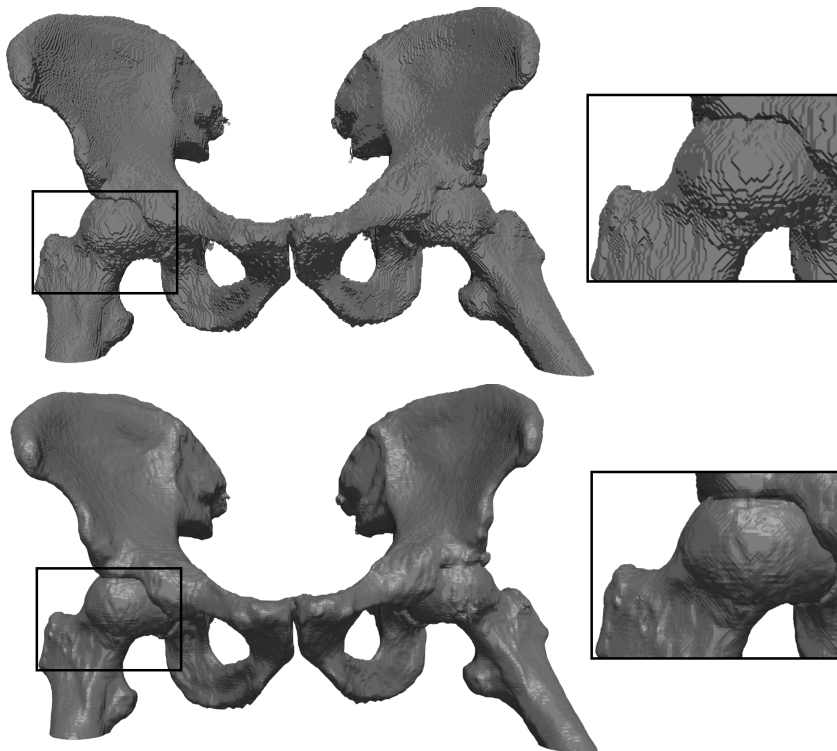
## 4 Applications

In this section, we review the methods presented in [9] and [10]. Both these methods improve the segmentations obtained by the IFT by performing operations locally around the segmentation boundary. Thus, the methods benefit from the on-the-fly approach presented here.

### 4.1 Sub-pixel segmentation with the IFT

The original IFT produces *crisp* segmentations, i.e., each image element is assigned the label of exactly one seed-point. However, due to the finite resolution of digital images, an image element may be partially covered by more than one (continuous) object. By allowing mixed labels, it is possible to obtain segmentations with sub-pixel precision. Numerous studies have confirmed that *pixel coverage segmentation* [14] outperforms crisp segmentation for subsequent measuring of object properties such as length/surface area and area/volume, see, e.g., [13, 12].

In [9], we presented a method, called the *sub-pixel IFT*, for approximating pixel coverage segmentation within the IFT framework, by computing mixed labels at the segmentation boundaries. Experiments, reported in [9], indicate that the sub-pixel IFT is less sensitive to small variations in seed-point placement than the crisp IFT. A segmentation result obtained with the sub-pixel IFT is shown in Figure 2.

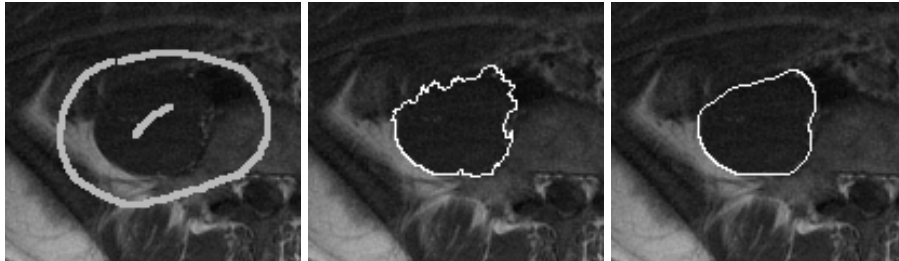


**Fig. 2.** Hip bones of a human, segmented from a CT volume image. The segmentation was obtained using the IFT, with seed-points selected interactively by a human user. A polygonal surface was extracted from the segmented volume using the Marching Cubes algorithm [8], which takes sub-pixel information into account when available. (Top) Segmentation obtained by the IFT. (Bottom) Segmentation obtained by the sub-pixel IFT proposed in [9].

## 4.2 The relaxed IFT

Numerous studies have shown that the IFT, and similar methods based on minimal cost paths, are capable of producing high quality segmentations in a wide range of contexts. However, in images with weak or missing boundaries the IFT tends to produce irregular segmentation boundaries. An explanation for this is that the IFT propagates information from the seed-points only along minimum cost paths. Since two adjacent image elements may receive their information from different seed-points, regularity of the segmentation boundary is not enforced.

In [10], we address this weakness of the IFT by proposing the *relaxed IFT* (RIFT). This modified version of the IFT features an additional parameter that controls the smoothness of the segmentation boundary, thereby making the results more predictable in the presence of noise and weak edges. The method works by applying an iterated relaxation procedure to the segmentation labels,



**Fig. 3.** Segmentation of a muscle in a slice from an MR volume image. (Left) Original image with seed-points representing muscle and background. (Middle) Segmentation result obtained by the IFT. (Right) Segmentation result obtained after 30 iterations of the relaxation procedure proposed in [10].

in a narrow band around the segmentation boundary. The effect of the relaxation procedure is illustrated in Figure 3. In [10], the RIFT was used to refine manual segmentations of a thoracolumbar muscle in MR images. The manual segmentations, segmentations obtained with the IFT, and segmentations obtained with the RIFT, were graded by 12 observers. The segmentations obtained by the RIFT were preferred over the segmentations obtained by the IFT without relaxation. Additionally, the segmentations obtained by the RIFT were found to be qualitatively comparable to the manual segmentations, while intra-user variations were reduced by more than 50%.

## 5 Conclusion

We have shown that the boundary of the segmentations obtained by the IFT may be computed on-the-fly, as a by-product of the DIFT algorithm. This allows the sub-pixel IFT and the RIFT to be implemented efficiently in conjunction with the DIFT.

As mentioned in Section 2.2, the minimum cost path between two image elements may not be unique. Therefore, a strategy is needed for assigning labels in ambiguous cases. In previous literature on the IFT, such a strategy is usually referred to as a *tie-breaking policy* [5]. In Algorithm 1, a *first-in-first-out* (FIFO) policy is assumed. However, extensions to other tie-breaking policies (e.g., the mean tie-breaking policy proposed in [9]) should be straightforward.

## Acknowledgments

Ingela Nyström and Ewert Bengtsson at the Centre for Image Analysis, Uppsala University, are acknowledged for scientific support.

## References

1. R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.
2. R. Audigier, R. Lotufo, and A. Falcão. On integrating iterative segmentation by watershed with tridimensional visualization of MRIs. In *Proceedings of the Computer Graphics and Image Processing, XVII Brazilian Symposium*, pages 130–137. IEEE Computer Society, 2004.
3. Y. Boykov and G. Funka-Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.
4. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
5. A. X. Falcão and F. P. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Transactions on Medical Imaging*, 23(9):1100–1108, 2004.
6. A. X. Falcão, J. Stolfi, and R. A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.
7. L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.
8. W. E. Lorensen. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
9. F. Malmberg, J. Lindblad, and I. Nyström. Sub-pixel segmentation with the image foresting transform. In P. Wiederhold and R. P. Barneva, editors, *Proceedings of the 13th International Workshop on Combinatorial Image Analysis (IWCIAP)*, volume 5852 of *LNCS*, pages 201–211. Springer, 2009.
10. F. Malmberg, I. Nyström, A. Mehnert, C. Engstrom, and E. Bengtsson. Relaxed image foresting transforms for interactive volume image segmentation. In B. M. Dawant and D. R. Haynor, editors, *Proceedings of SPIE Medical Imaging*, volume 7623. SPIE, 2010.
11. A. J. H. Mehnert and P. T. Jackway. An improved seeded region growing algorithm. *Pattern Recognition Letters*, 18:1065–1071, 1997.
12. N. Sladoje and J. Lindblad. Estimation of moments of digitized objects with fuzzy borders. In F. Roli and S. Vitulano, editors, *Proceedings of the 13th International Conference on Image Analysis and Processing (ICIAP)*, volume 3617 of *LNCS*, pages 188–195. Springer-Verlag, 2005.
13. N. Sladoje and J. Lindblad. High-precision boundary length estimation by utilizing gray-level information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):357–363, 2009.
14. N. Sladoje and J. Lindblad. Pixel coverage segmentation for improved feature estimation. In P. Foggia et al., editors, *Proceedings of the 15th International Conference on Image Analysis and Processing (ICIAP)*, volume 5716 of *LNCS*, pages 929–938. Springer-Verlag, 2009.