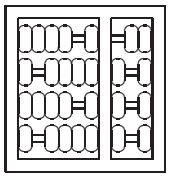


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

# Automatic Image Segmentation by Tree Pruning

*F. P. G. Bergo* *A. X. Falcão*
*P. A. V. Miranda* *L. M. Rocha*

Technical Report - IC-07-23 - Relatório Técnico

July - 2007 - Julho

# Automatic Image Segmentation by Tree Pruning

Felipe P.G. Bergo
LIV – Institute of Computing – Unicamp
CP 6176, 13084-971, Campinas, SP, Brazil

Alexandre X. Falcão
LIV – Institute of Computing – Unicamp
CP 6176, 13084-971, Campinas, SP, Brazil

Paulo A.V. Miranda
LIV – Institute of Computing – Unicamp
CP 6176, 13084-971, Campinas, SP, Brazil

Leonardo M. Rocha
DECOM – FEEC – Unicamp
CP 6101, 13083-970, Campinas, SP, Brazil

**Abstract**

The *Image Foresting Transform* (IFT) is a tool for the design of image processing operators based on connectivity, which reduces image processing problems into a minimum-cost path forest problem in a graph derived from the image. We propose a new image operator, which solves segmentation by pruning trees of the forest. An IFT is applied to create an optimum path forest whose roots are *seed pixels*, selected inside a desired object. In this forest, object and background are connected by optimum paths (*leaking paths*), which cross the object's boundary through its "most weakly connected" parts (*leaking pixels*). These leaking pixels are automatically identified and their subtrees are eliminated, such that the remaining forest defines the object. Tree pruning runs in linear-time, is extensive to multidimensional images, is free of *ad-hoc* parameters, requires only internal seeds, and works with minimal interference from the heterogeneity of the background. These aspects favor solutions that exploit image features and object information for automatic segmentation. We give a formal definition of the obtained objects and conditions to achieve robust segmentation using tree pruning, describe the algorithms and evaluate automatic segmentation by tree-pruning in two applications: 3D MR-image segmentation of the human brain and segmentation of license plates. Given that its most competitive approach is the watershed transform by markers, we include a comparative analysis between them.

## 1 Introduction

We consider the problem of defining the precise spatial extent of a desired object in a given image, namely *image segmentation*. The difficulties in image segmentation stem from the absence of global object information (location, shape, appearance) and the similarities between object and background with respect to local image features (color and texture). These difficulties usually call for user assistance [21, 8, 7, 13, 4, 10], making automatic segmentation viable only in an application-dependent and tailored fashion. Ideally, methods for automatic segmentation should separate the part that is application-dependent from the application-independent part, such that the former can be easily tailored for different applications.

We present a method, called *tree pruning*, which is consistent with the above strategy. Tree pruning uses the *Image Foresting Transform* (IFT)— a tool for the design of image processing operators based on connectivity [12]. The IFT has been applied to compute distance transforms, multiscale skeletonizations, morphological reconstructions, watershed transforms, boundary tracking, fractal dimension, and shape saliences [16, 24, 15, 14, 25, 31, 32]. Tree pruning is the first IFT-based image operator, which exploits a combinatorial property of the forest.

In the IFT framework, an image is interpreted as a graph whose nodes are image pixels and whose arcs are defined by an *adjacency relation* between pixels. For a given set of *seed pixels* and suitable *path-cost function*, the IFT computes a minimum-cost path forest in the graph whose roots are drawn from the seed set. Each tree in the forest consists of pixels more strongly connected to its root than to any other seed. In tree pruning, the seeds are chosen inside the object and the choice of the path-cost function intends to connect object and background by optimum paths (*leaking paths*), which cross the object's boundary through its "most weakly connected" parts (*leaking pixels*). A *combinatorial property* of the forest is exploited to automatically identify the leaking pixels and eliminate their subtrees, such that the remaining forest defines the object.

Tree pruning runs in linear time, is extensive to multidimensional images, is free of *ad-hoc* parameters, requires only internal seeds, and works with minimal interference from the heterogeneity of the background. These aspects favor solutions that exploit image features and object information for automatic segmentation. For example, we can use approaches that locate the object in the image to estimate seeds [35]. Candidate seeds can otherwise be used to obtain a set of possible objects, and the desired one can be chosen based on objective functions [30, 36, 22] or object features and pattern classifiers [20, 9, 23]. One can also exploit some combination between tree pruning and deformable models [21, 8, 7, 5] in order to achieve a better agreement between the geometry of the model and local image features. Even in the context of interactive segmentation, it is highly desirable to make the user's actions simple and minimal. Tree pruning reduces user intervention to a few markers on the image.

In comparison to advanced region-growing approaches that use optimum paths from internal seeds [27, 33], the criterion to disconnect object and background is not based on the costs of the optimum paths but on a combinatorial property of the forest. Optimum paths that reach object pixels are assumed to not pass through the background, instead of having costs strictly lower than the costs of paths that reach the background.

In tree pruning, internal seeds compete among themselves and only a few seeds become roots of leaking paths. Other approaches based on optimal seed competition [24, 3, 34, 29, 28] can be roughly described in three steps: (i) seed pixels are selected inside and outside the objects, (ii) each seed defines an *influence zone* that contains pixels which are more strongly connected to that seed than to any other, and (iii) each object is defined by the union of the influence zones of its internal seeds. The absence of boundary information and/or heterogeneity of the background usually cause invasion (leaking) of object seeds in influence zones of background seeds and vice-versa. These methods are suitable for interactive segmentation, where the user can correct leaking by adding and removing seeds. In the context of the IFT, these corrections can also be done in sublinear time [10, 2] (e.g.,

by differential watershed transforms). Tree pruning exploits the leaking problem and also favors solutions for automatic segmentation, as discussed above. On the other hand, tree pruning and the IFT-watershed transform by markers [24] use the same image graph and path-cost function which makes them competitive approaches, when the external seeds for watershed can be found automatically. We evaluate this aspect in this paper.

Approaches for image segmentation usually exploit image features and some object information to emphasize the discontinuities between object and background. It is desirable, for example, that the external energy in snakes be lower along the object's boundary than inside and outside it [21]; the arc costs in live wire be lower along the object's boundary than within a neighborhood around it [13]; the local affinities in relative fuzzy connectedness [29] be higher inside and outside the object than on its boundary; the gradient values in watershed transform be higher for pixels on the object's boundary than inside and outside it [24, 3, 34]; and the arc weights in graph-cut segmentation be lower across the object's boundary than inside and outside it [4, 30, 22]. Additionally, the energy minimization in [4, 22] using min-cut/max-flow algorithms from source to sink nodes [17] also requires lower arc-weights between source and object pixels, higher arc-weights between sink and object pixels, lower arc-weights between sink and background pixels, and higher arc-weights between source and background pixels. Clearly, the effectiveness of these approaches is affected when the above *robustness conditions* are not fully satisfied, but they can still work under certain hard constraints (usually, involving the user). For example, Boykov and Jolly [4] allow the user to force the arc weights with source and sink by selecting seed pixels inside and outside the object.

Tree pruning can take advantage of the same image features and object information to create a *gradient-like image* (Figure 1a) whose pixel values are higher on the object's boundary than inside it and, at least, in a neighborhood outside it. The *cost of a path* in tree pruning is given by the maximum gradient value along it. Considering all possible paths from the internal seed set to a given pixel, the IFT assigns a path of minimum cost to that pixel. Therefore, the leaking pixels will be those with lower values along the object's boundary and the leaking paths will reach all background pixels around the object with costs equal to their leaking pixel values. By connectivity, the rest of the background will be also conquered by leaking paths that pass through the same leaking pixels. As explained above, the automatic identification of these leaking pixels solves the problem (Figure 1b). Under the same conditions, the watershed transform may fail whenever the heterogeneity of the background results gradient values similar to those of the desired boundary, because the costs of optimum paths from external seeds may saturate before these paths reach the internal ones at the object's boundary (Figure 1c). This will make the location of the external seeds more important to solve the problem in watershed-based approaches (Figure 1d).

Tree pruning was first presented in [11], with two approaches to detect leaking pixels. One is interactive where the user can visually identify leaking pixels and select them with the mouse pointer. The other is automatic, but relies on an *ad-hoc* parameter that is difficult to be adjusted in real applications. In [26], we revisited the method to propose an automatic solution for leaking pixel detection, which is free of *ad-hoc* parameters, and to provide a comparative analysis of tree pruning and watershed transform for automatic segmentation,

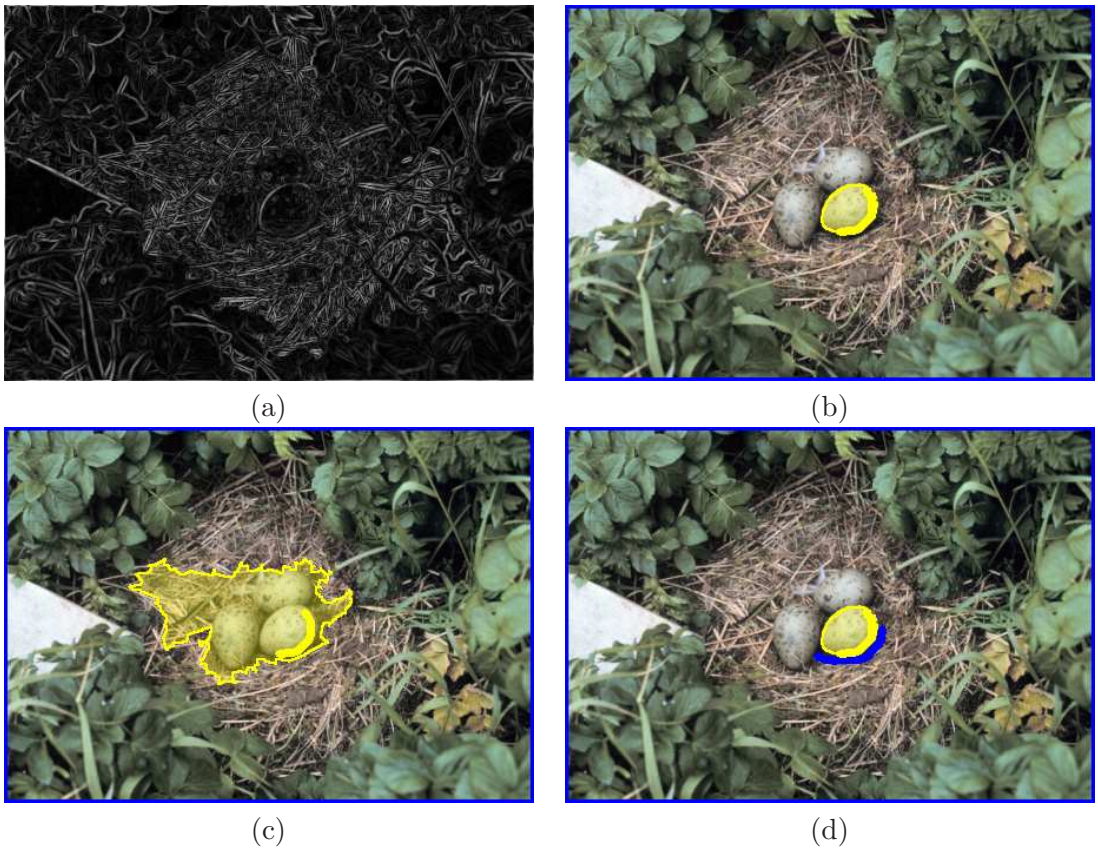


Figure 1: (a) A gradient-like image where the object is an egg. (b) Segmentation result with tree pruning and the yellow marker as internal seeds. (c–d) Segmentation results with watershed under the same conditions and using the image's border and the blue marker as external seeds.

including one experiment for license plate segmentation. In this paper, we provide more details in the presentation of the method (formal definition of the obtained objects, different examples, algorithms, robustness conditions, gradient-like images, and geometrical issues), include other approaches for license plate segmentation, and another application (automatic 3D MR-brain segmentation) where tree pruning and watershed transform are compared with a template-based approach widely used for medical research [18].

We give the main definitions and instantiate the IFT for tree pruning and watershed transform in Section 2; describe tree pruning with automatic detection of leaking pixels in Section 3; discuss robustness conditions and geometrical issues in Section 4 and gradient-like images in Section 5; evaluate the method in comparison to other approaches in Section 6; and state the conclusions of this work in Section 7.

## 2 Background

Tree pruning (TP) and watershed (WS) algorithms rely on a *gradient-like image*, being both approaches extensive to multidimensional and multiparametric images. In several situations, the gradient-like image can be simply the magnitude of some gradient operator, such as the Sobel's gradient (Figure 1a). In other situations, it is better to assign an image feature vector to each pixel and compute the gradient-like image as a function of the differences between feature vectors of adjacent pixels (Section 5).

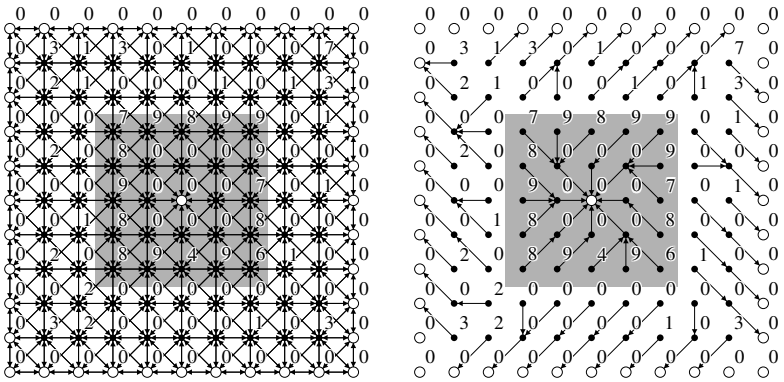
**Gradient condition:** Ideally, a gradient-like image in WS should assign higher values to pixels on the object's boundary than to pixels inside and outside the object. TP relies on similar condition, except that the lower pixel values outside the object are required only within a small neighborhood around the object's boundary (when the gradient condition is not satisfied, the alternatives are discussed in Section 4).

In the following, we present the image foresting transform and its algorithm suitable for TP and WS.

### 2.1 Image Foresting Transform

A gradient-like image $\hat{I}$ is a pair $(D_I, I)$ where $D_I \subset Z^2$ is the image domain and $I(p)$ assigns to each pixel $p \in D_I$ a scalar value.

The gradient-like image is interpreted as a graph $(D_I, A)$ whose nodes are the pixels in $D_I$ and whose arcs are defined by an *adjacency relation* $A$ between pixels [12]. We are interested in 4- or 8-connected relations for 2D region-based image segmentation (Figure 2a).



(a) (b)

Figure 2: (a) An 8-connected image graph for WS, where the numbers indicate the pixel values, the shaded area is the object with one internal seed (white dot), and external seeds are selected on the image's border. (b) The respective optimum-path forest using Eq. 1.

A *path* in the graph is a sequence of adjacent pixels and a *path-cost function* $c$ assigns to each path $\pi$ a *path cost* $c(\pi)$.

**Definition 2.1 (Optimum path)** *A path $\pi$ is* optimum *if $c(\pi) \leq c(\tau)$ for any other path $\tau$ with the same destination of $\pi$.*

For both WS and TP, the cost $c(\pi)$ of a path is defined as the *maximum gradient value* of its pixels, when $\pi$ starts in a set $S$ of seed pixels; and as *infinity cost* otherwise.

$$c(\pi) = \begin{cases} \max_{\forall p \in \pi}\{I(p)\} & \text{if } org(\pi) \in S \\ +\infty & \text{otherwise} \end{cases} \quad (1)$$

where $org(\pi)$ is the origin of path $\pi$. Marker imposition [34, 3, 24] is important in most situations and it is implemented by setting $I(p)$ to 0 for pixels $p \in S$.

We can also assign gradient values to arcs, rather than to pixels, and both approaches will work similarly with the path-cost function given by the maximum arc-weight along the path. In this case, instead of leaking pixels, we would have *leaking arcs* (for more comments, see Section 3). The results may be more accurate due to the better gradient resolution, but for sake of simplicity, we will describe the methods using the pixel resolution.

The IFT assigns one optimum path (Definition 2.1) from $S$ to every pixel $p \in D_I$. These paths form an *optimum-path forest* rooted in $S$ which is stored in a *predecessor map* $P$.

**Definition 2.2 (Optimum path forest)** *A predecessor map $P$ is a function that assigns to each pixel $p \notin S$ its predecessor $P(p)$ in the optimum path from $S$ or a marker nil when $p \in S$. An optimum-path forest is a predecessor map which contains no cycles — in other words, one which takes every pixel to nil in a finite number of iterations.*

WS separates object and background by labeling the trees of the forest in Figure 2b. Therefore, the IFT algorithm presented next computes at same time an optimum-path forest in $P$ and a label map in $L$, being the former useful for TP and the latter applicable for WS.

## 2.2 The IFT algorithm

Let $S = S_o \cup S_b$ be the union of two sets of seed pixels, such that $S_o$ and $S_b$ contain only object and background seeds, respectively. Then, $S_b$ is empty for TP and WS requires $S_b$ not empty. For the purpose of comparison, we are only interested in the case where $S_b$ is the image's border for WS. However, as we will show in Section 4, pixels strategically selected outside the object play an important role in interactive segmentation.

**Algorithm 1** – Image Foresting Transform for WS and TP

| | |
|---|---|
| Input: | Gradient-like image $\hat{I} = (D_I, I)$, adjacency relation $A$, seed sets $S_o$ and $S_b$. |
| Output: | Optimum-path forest $P$ and label map $L$. |
| Auxiliary: | Cost map $C$, priority queue $Q$, and variable *cst*. |

1. *For all $p \in D_I$, set $P(p) \leftarrow nil$ and $C(p) \leftarrow +\infty$.*
2. *For all $p \in S_o$, set $C(p) \leftarrow I(p)$, $L(p) \leftarrow 1$, and insert $p$ in $Q$.*
3. *For all $p \in S_b$, set $C(p) \leftarrow I(p)$, $L(p) \leftarrow 0$, and insert $p$ in $Q$.*
4. *While $Q$ is not empty, do*

```
5.      Remove from Q a pixel p such that C(p) is minimum.
6.      For each q such that (p,q) ∈ A and C(q) > C(p), do
7.          Compute cst ← max{C(p), I(q)}.
8.          If cst < C(q), then
9.              If C(q) ≠ +∞, remove q from Q.
10.             Set P(q) ← p, C(q) ← cst, L(q) ← L(p).
11.             Insert q in Q.
```

Algorithm 1 runs in linear time when $Q$ is implemented as described in [16]. Lines 1–3 initialize maps and insert seeds in $Q$. The main loop computes an optimum path from $S$ to every pixel $p$ in a non-decreasing order of cost (Lines 4–11). At each iteration, a path of minimum cost $C(p)$ is obtained in $P$ when we remove its last pixel $p$ from $Q$ (Line 5). Ties are broken in $Q$ using first-in-first-out (FIFO) policy. That is, when two optimum paths reach an ambiguous pixel $p$ with the same minimum cost, $p$ is assigned to the first path that reached it. The rest of the lines evaluate if the path that reaches an adjacent pixel $q$ through $p$ is cheaper than the current path with terminus $q$ and update $Q$, $C(q)$, $L(q)$ and $P(q)$ accordingly.

The label propagation in $L$ assigns 1 to pixels that belong to the trees rooted inside the object and 0 to pixels of the trees rooted in the background. In WS, it is expected that the object be defined by image components with label 1, which can be directly obtained from $L$ (Figures 1c–d).

Clearly, WS solves segmentation by seed competition for object and background pixels. It also allows simultaneous multiple object segmentation by modifying Algorithm 1 to propagate a distinct label per object.

TP requires to identify leaking pixels in an optimum-path forest $P$ with no external seeds. The object is obtained by pruning all subtrees rooted in the background (Figure 1b). This approach is discussed next.

## 3 Tree-pruning segmentation

Figures 3a and 3b show an image graph very similar to the one of Figure 2a, but with no external seeds, and its corresponding optimum-path forest $P$, respectively. Note that, the IFT algorithm computes optimum paths in a non-decreasing order of costs. Therefore, under the gradient condition for TP, the optimum paths from $S_o$ will reach object pixels before background pixels. Moreover, if a pixel $p$ is the only one with lowest value $I(p)$ on the object's boundary, then all pixels around the object will be reached by leaking paths with minimum cost $I(p)$, which pass through the leaking pixel $p$ (pixel $(6, 8)$ in Figure 3b). By connectivity, the rest of the background will be also conquered by leaking paths that pass through $p$. When the gradient condition is not fully satisfied the method may still works (Figure 4). The same property can be verified when there are multiple leaking pixels, which can be automatically detected for object definition as follows.

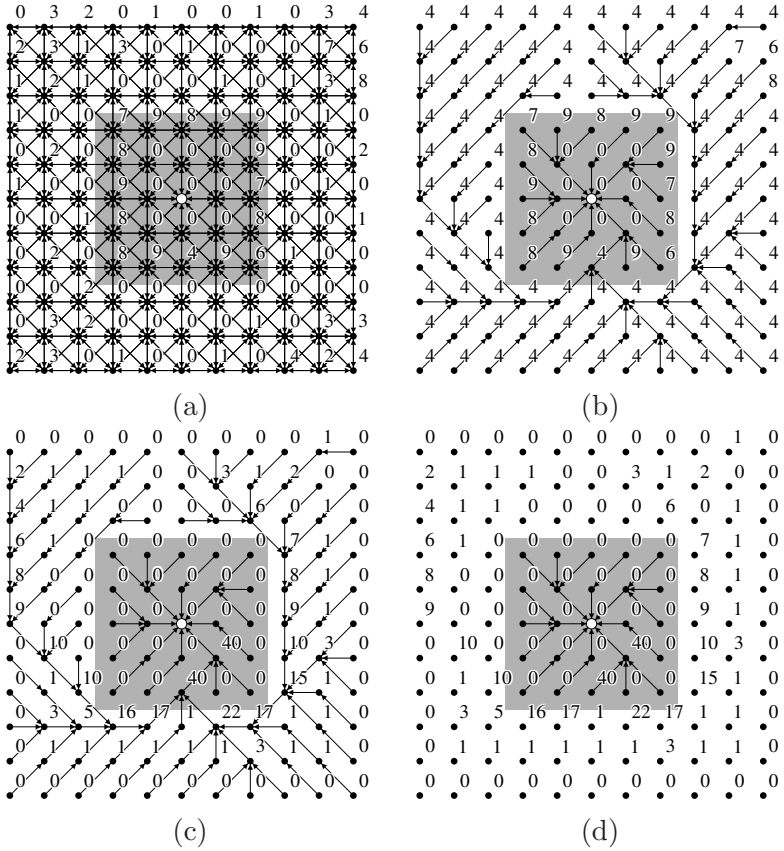



Figure 3: (a) An 8-connected image graph for TP, where the numbers indicate the pixel values and the shaded area is the object with one internal seed (white dot). (b) The respective optimum-path forest using Eq. 1. (c) The numbers indicate the descendant count in $B$ for each pixel (except for the root) of the forest computed in (b). (d) After pruning, the remaining forest defines the object.

### 3.1 Object definition

Let $\mathcal{R}$ be the set of the roots in the optimum-path forest (Definition 2.2). By removing the roots of the forest, we get a forest of subtrees. Each tree of this new forest may be classified as being an *object tree* or as a *leaking tree*. An object tree is a maximal subtree of $\mathcal{R}$, which is contained within the object. A leaking tree is a maximal subtree of $\mathcal{R}$, which contains leaking paths. Figure 3b shows several object trees and a single leaking tree.

Let $B \subset D_I$ be the image's border or any other closed set around the object's boundary. We compute the number of descendants that every node of the new forest has in $B$ to obtain a *descendant map* $D$ (Figure 3c). This combinatorial property of the forest allows to identify the leaking pixels. The map $D$ is different from the one presented in [11], which computes all descendants in the forest.

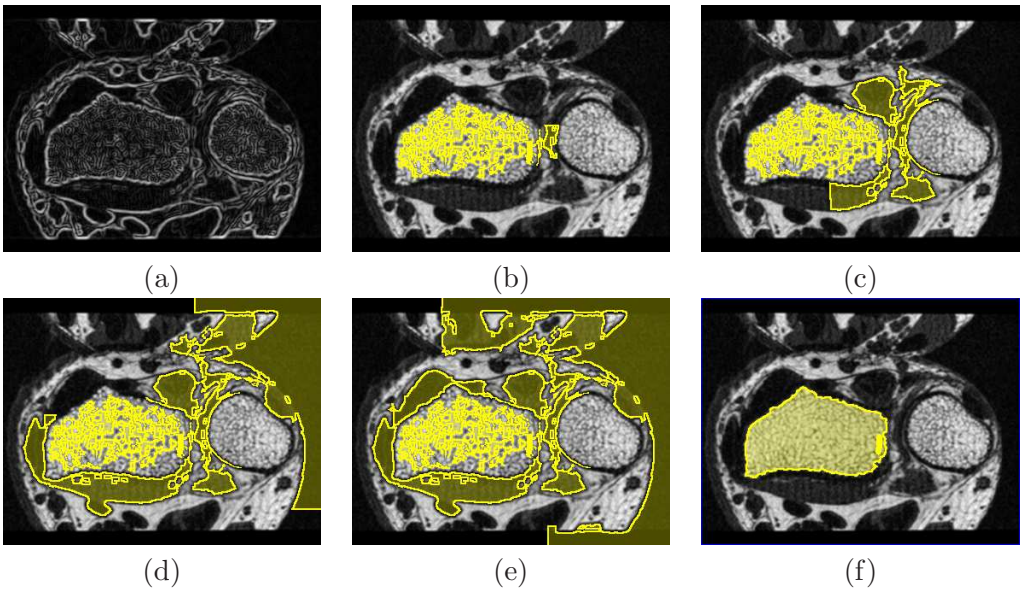



Figure 4: (a) A gradient-like image where the object is a bone of the wrist. (b-e) The region growing of the IFT from internal seeds. The leaking occurs before filling the entire object, but these leaking paths surround the object, avoiding further connections between object and background. (f) The object is obtained by automatic leaking pixel detection.

**Definition 3.1 (Descendant map)** *A descendant map is a function $D$ which assigns to each pixel $p \in D_I \backslash \mathcal{R}$ the number of descendants of $p$ in $B$.*

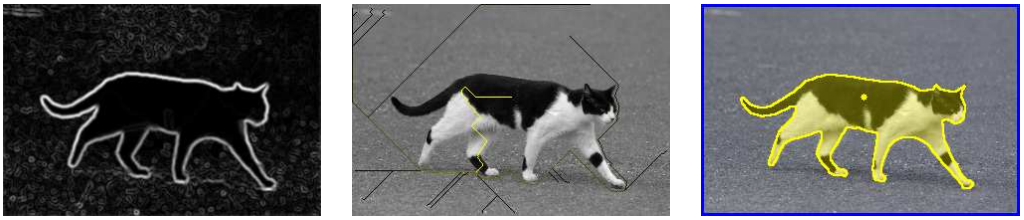
Each leaking tree is supposed to cross the object's boundary through a single pixel, named leaking pixel, and all paths that reach $B$ must pass through some leaking pixel (Figure 3b). The leaking pixels can be detected from the predecessor and descendant maps.

**Definition 3.2 (Leaking pixel)** *A leaking pixel is defined as the* last one *with the highest descendant value along an optimum path that reaches $B$.*

That is, by following backwards any optimum path in $P$ which has terminal node in $B$, the leaking pixel is the *first one* with the highest descendant value in the way back to the root of its tree (e.g., pixel $(6, 8)$ in Figure 3c). We can repeat this procedure for all nodes in $B$ to detect all leaking pixels automatically.

This property occurs at the object's boundary thanks to the FIFO policy of $Q$. After leaking, the gradient condition makes ambiguous the pixels in the neighborhood outside the object, ramifying the leaking path into several branches (Figure 5). This ramification drastically reduces the descendant count for pixels of the subtrees rooted at the leaking pixels. The object is finally obtained by removing the subtrees of the leaking pixels from the original forest (Figures 3d, 4f and 5c).

**Definition 3.3 (Object by tree pruning)** *Let $\mathcal{P}$ be the set of pixels that belong to the subtrees of leaking pixels, $\mathcal{T}_l$ be the set of pixels that belong to the leaking trees, and $\mathcal{T}_o$ be the set of pixels that belong to the object trees. The object is defined as $\mathcal{R} \cup \mathcal{T}_o \cup \{\mathcal{T}_l \backslash \mathcal{P}\}$.*



(a) (b) (c)

Figure 5: (a) A gradient-like image. (b) The original image overlaid by the descendant map. The leaking path ramifies into several branches on the object's boundary, provoking a decreasing in $D$ (the lines become darker) and avoiding further connections between object and background. (c) The resulting segmentation with TP.

## 3.2 Algorithms

Algorithm 2 computes the descendant map $D$ in linear time. It visits all pixels of the forest in reverse breadth-first order, accumulating the number of descendants from the leaf pixels to the root pixels. Lines 1–3 insert the forest roots in a FIFO queue $Q$ and initialize the descendant count map $D$. Lines 4–8 traverse the optimum-path forest in breadth-first order, inserting every visited pixel in the stack $S$. Lines 9–13 use the stack $S$ to visit the forest in reverse breadth-first order, calculating and propagating the descendant count up to the roots of the forest. The resulting map $D$ counts, for each pixel, the number of descendant pixels in the optimum-path forest that belong to the set $B$. Algorithm 2 visits each pixel exactly three times, and its running time is $\Theta(|D_I|)$.

**Algorithm 2** – Descendant Map Computation (Linear Time)

Input: Optimum-path forest $P$, adjacency relation $A$, set $B$.
Output: Descendant map $D$.
Auxiliary: Queue $Q$, Stack $S$.

1. *For all $p \in D_I$, do*
2. ⎸ *Set $D(p) \leftarrow 0$.*
3. ⎸ *If $P(p) = nil$, insert $p$ in $Q$.*
4. *While $Q$ is not empty, do*
5. ⎸ *Remove a pixel $p$ from $Q$.*
6. ⎸ *Insert $p$ in $S$.*
7. ⎸ *For each pixel $q$ such that $(p, q) \in A$, do*
8. ⎸ ⎸ *If $P(q) = p$, insert $q$ in $Q$.*
9. *While $S$ is not empty, do*
10. ⎸ *Remove a pixel $p$ from $S$.*

```
11.    If P(p) ≠ nil, then
12.        Set D(P(p)) ← D(P(p)) + D(p).
13.        If p ∈ B, set D(P(p)) ← D(P(p)) + 1.
```

Algorithm 3 computes the same descendant map $D$ with a different approach: For each pixel in $B$, the algorithm follows backwards its optimum-path up to the root, updating the descendants-in-the-border count. In the worst case – an unlikely situation where all pixels of $B$ belong to a same optimum-path that contains all pixels of the image – Algorithm 3 will visit each pixel $|B|$ times, leading to $O(|D_I|\,|B|)$ performance. For 2D images, $|B| \propto \sqrt{|D_I|}$, and its running time becomes $O(|D_I|^{\frac{3}{2}})$. For 3D images, $|B| \propto \sqrt[3]{|D_I|^2}$, and its running time becomes $O(|D_I|^{\frac{5}{3}})$.

**Algorithm 3** – Descendant Map Computation (Alternative)

Input: Optimum-path forest $P$ and set $B$.
Output: Descendant map $D$.

```
1.  For all p ∈ D_I, set D(p) ← 0.
2.  For each pixel p ∈ B, do
3.      Set q ← p.
4.      While P(q) ≠ nil, do
5.          Set D(P(q)) ← D(P(q)) + 1, and q ← P(q).
```

In real applications, Algorithm 3 visits much less than $|D_I|$ pixels, since a large number of pixels belong to optimum paths that do not reach the set $B$ and thus are never visited. While Algorithm 2 guarantees linear performance, Algorithm 3 leads to reduced running times in practical applications. Experimental comparison of these algorithms in real 3D segmentation applications showed that Algorithm 3 is 1.8 times faster than Algorithm 2. The implementation of Algorithm 3 is also shorter and simpler than the implementation of Algorithm 2.

The descendant map $D$ is used to detect the leaking pixel associated to each node in $B$. For every backwards path from $B$ to its root, the first pixel with the highest value in $D$ along the backwards path is a leaking pixel. The set $L_k$ of leaking pixels is computed by Algorithm 4.

**Algorithm 4** – Leaking Pixel Detection

Input: Optimum-path forest $P$, descendant map $D$, and set $B$.
Output: Set $L_k$ of leaking pixels.

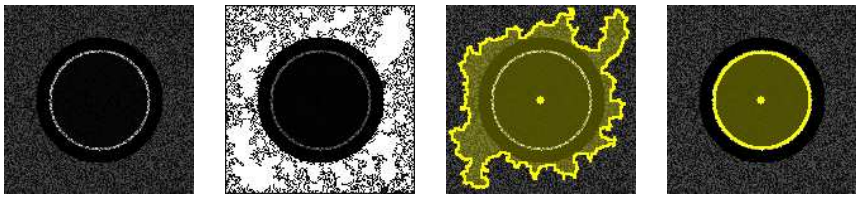
```
1.  For each pixel p ∈ B, do
2.      Set q ← p, set d_max ← −∞.
3.      While P(q) ≠ nil, do
4.          If D(q) > d_max Then set d_max ← D(q), r ← q.
5.          Set q ← P(q).
6.      Set L_k ← L_k ∪ {r}.
```

In the pixel resolution, the subtrees of the leaking pixels belong to the background. If we adopt gradient values per arc and the maximum arc-weight path-cost function, the leaking arcs will cross the object's boundary such that the last pixel with the highest descendant value along the optimum path will be at the external node of the arc. In this case, the removal of the leaking arc corresponds to eliminate the tree rooted at this external node.

## 4 Robustness conditions and geometrical issues

This section discusses important aspects to understand the differences between WS and TP, their advantages and limitations. In all examples, we use the same gradient-like image and internal seeds $S_o$ for watershed transform (WS) and tree pruning (TP). Additionally, WS uses the image's border as external seeds $S_b$ and TP uses the image's border $B$ to extract the descendant map $D$. Therefore, the role of the image's border is very different in these approaches. When the segmentation fails, we may correct it by adding seeds to $S_o$, to $S_b$ in the case of WS, and pixels to $B$ in the case of TP, and re-executing the algorithms.

The gradient conditions of WS and TP (Section 2) play an important role in the effectiveness of these methods. When they are satisfied, the optimum paths from $S_o$ and $S_b$ in WS will meet at the object's boundary independent of seed location. Indeed, WS will work with only two seeds, one external and one internal. However, high gradient values inside or outside (Figure 6a) the object in WS may create ambiguous regions (plateaus of cost or tie zones) whose pixels are only reached by optimum paths with costs greater than or equal to the leaking pixel values (Figure 6b, tie zones appear as white pixels). Depending on the tie-breaking policy, the watershed line could be anywhere on these plateaus (Figure 6c). Since TP does not depend on the costs of the optimum paths outside the object, but only on the way the leaking paths conquer the background pixels around the object, it is much less susceptible to the heterogeneity of the background (Figure 6d). However, high gradient values inside the object may also cause problems for TP.



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

Figure 6: (a) A synthetic gradient-like image, where the heterogeneity of the background is given by a random external noise. (b) Watershed tie zones in white. (c-d) Respective segmentation results by WS and TP for a same internal seed (yellow dot).

TP requires three important conditions:

(i) Each leaking tree must contain a single pixel on the object's boundary,

(ii) The optimum paths to object pixels must not pass through the background, and

(iii) The optimum paths that reach the external set $B$ must pass through all leaking pixels.

Condition (i) is obvious from Algorithm 4 that detects leaking pixels. If a leaking tree has multiple pixels on the object's boundary, the last pixel with the highest descendant value along the leaking path will be inside the object, failing segmentation. This situation may happen when the gradient condition is not satisfied inside the object (Figures7a–b). Condition (ii) may also fail due to the same reason. The problem can be usually solved with multiple internal seeds (Figures 7c–d).

The same example does not satisfy the gradient condition for WS. Figures 7e–f show that, besides the image's border, additional external seeds are needed in $S_b$ to correct segmentation with WS.

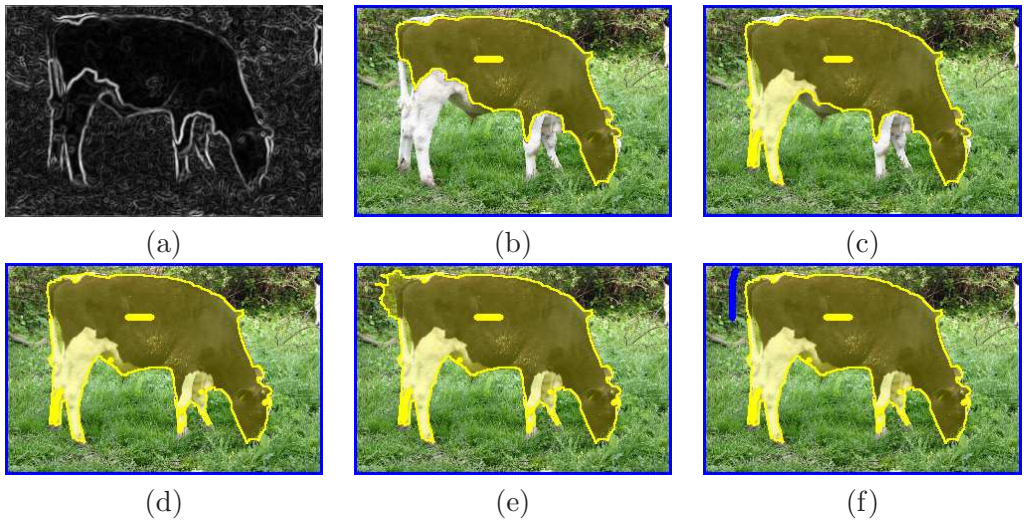
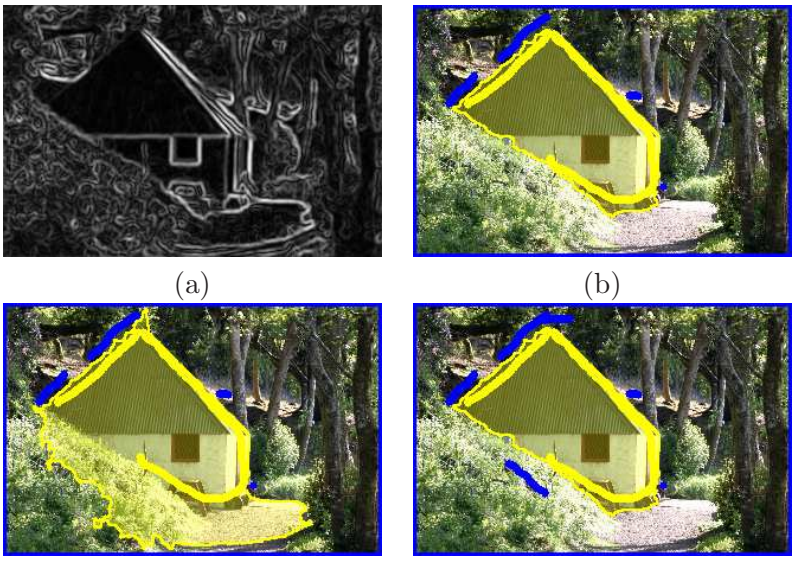



Figure 7: (a) A gradient-like image. (b–d) Results of segmentation with TP and incremental seed sets (yellow markers). (e–f) WS requires the image's border and additional external seeds (blue markers) to work.

When the gradient condition is not satisfied, TP may also require additional pixels in set $B$ (Figures 8a–b). Even in this situation, TP may require less intervention than WS (Figures 8c–d), but it is difficult to affirm that TP is a better approach for interactive segmentation than WS. Another example shows that seed location in TP may be more important than in WS when we need more pixels in $B$ (Figure 9). Apart from their differences, they should produce similar results.

Condition (iii) implies that all leaking pixels must have descendants in $B$. This is usually the case when we have a few leaking pixels, but there are two extreme situations which lead to several leaking pixels: perfect boundaries and perfect gaps. Condition (iii) will fail in these situations whenever the competition among leaking paths prevents paths that cross some leaking pixels to reach $B$. We present some alternatives to these situations.

Figures 10a and 10b show a perfect boundary and its segmentation with internal seeds. A simple solution is to order the pixels by their gradient values such that pixels with the



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

Figure 8: (a) A gradient-like image. (b) Resulting segmentation with TP, internal seeds (yellow markers) and additional pixels in $B$ (blue markers). (c-d) Resulting segmentation with WS and additional external seeds, besides the image's border (blue markers).

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

Figure 9: (a) Result of segmentation with TP. WS obtains similar result with the image's border as external marker. (b) WS segments the object with one additional external seed (blue dot). (c) TP fails with the same seed selection. (d) TP works when we change the location of the internal seed (yellow dot).

same intensities are randomly ordered (Figures 10c and 10d). In the case of perfect gaps (Figure 10e), we can add internal seeds in $S_o$ and external pixels in $B$ around the gaps to solve segmentation (Figure 10f). The same applies to WS. Another solution is to run an edge detection method, compute the Euclidean distance transform to the edges, and create an edge distance map as the complement of the pixel distances to the edges. By adding the edge distance values to the original gradient value, we obtain a new gradient image where

the number of leaking pixels is drastically reduced (Figure 10g), solving the problem with perfect gaps (Figure 10h). Fortunately, images usually have noise that considerably reduces the number of leaking pixels. Therefore we never needed to use this solution in practice.

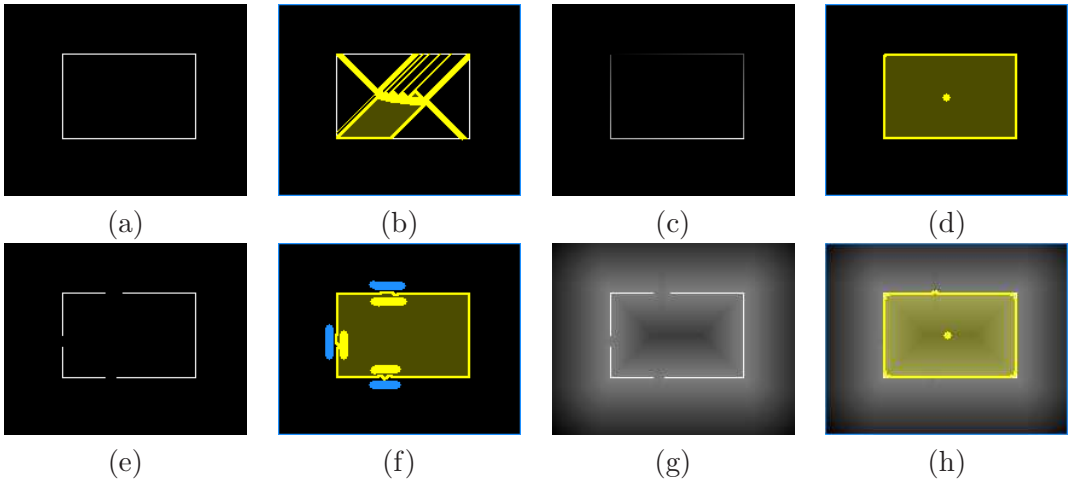
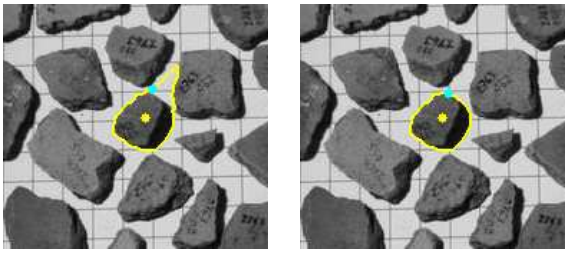



Figure 10: (a) Synthetic gradient with perfect boundary. (b) TP fails because it missed several leaking pixels. (c) A randomly ordered gradient. (d) TP works on (c) with a single seed. (e) Synthetic gradient with perfect gaps. (f) A hard solution using markers around the gaps. (g) A gradient image with edge distance map. (h) TP works on (g) with a single seed.

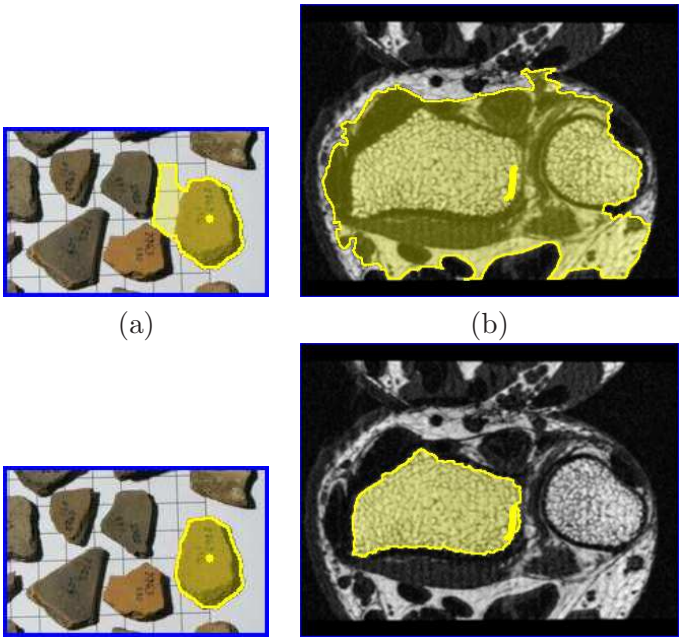
**The variant of maximum gradient:** Another situation may occur when multiple objects with similar gradient values are very close to each other. The absence of space in between the objects to ramify the path at a boundary's pixel creates a false pruning pixel outside the object (Figure 11a). If the gradient condition is satisfied, we may assume that the correct location of a leaking pixel is always at the maximum gradient value in the path segment between the detected pixel and its root. We call this *variant of maximum gradient*. Note that it does not affect the location of real leaking pixels on the object's boundary, but solves the problem as illustrated in Figure 11b. This variant has been used in all examples and experiments in this paper.

Condition (iii) may also fail in the case of nested boundaries. External (true or false) closed boundaries may prevent leaking paths of internal boundaries to reach set $B$. The problem can be solved with an alternative pixel in $B$, which should be selected between the boundaries. In [26], we proposed an alternative solution which iteratively searches the desired boundary by matching candidates with a template. This solution goes backwards along the optimum path from the detected leaking pixel to its root, looking for a next pixel for pruning, whose gradient value is maximum in the remaining segment. A third option is to improve gradient computation in order to avoid such situations (Figure 12).



(a) (b)

Figure 11: Image with one object fragment marked for detection. (a) The detected pixel is outside the fragment due to its proximity to the other fragments. (b) The correct leaking pixel is automatically detected using the variant of maximum gradient.



(a) (b)

(c) (d)

Figure 12: Results of TP for a fragment and a bone of the wrist using (a-b) the magnitude of the Sobel's gradient and (c-d) the improved gradient presented in Section 5.

## 5 Gradient-like images

It should be clear that under the gradient condition for WS, TP and WS provide similar results, except for some pixels on the object's boundary because WS divides the boundary between influence zones of internal and external seeds. In order to create suitable gradient-like images for both methods, we should be able to exploit local image features which distinguish object and background.

Let $\vec{f}(p) = (f_1,(p), f_2(p), \ldots, f_n(p))$ be a vector at pixel $p$ such that the values $f_i(p)$, $i = 1, 2, \ldots, n$, are the brightness values of $p$ and of its 8-neighbors, for instance. The differences $d_j(p, q_j)$, $j = 1, 2, \ldots, 8$, measure the brightness variations around arcs $(p, q_j)$ between $p$ and each of its 8-neighbors.

$$d_j(p, q_j) \quad = \quad \frac{1}{n}\sum_{i=1}^{n} f_i(q_j) - f_i(p). \tag{2}$$

The vector $\vec{d}_j(p, q_j) = d_j(p, q_j)\frac{q_j - p}{|q_j - p|}$ represents the brightness variation in the direction of $(p, q_j)$. The gradient vector $\vec{G}(p)$ at $p$ is obtained by the sum of its projections on these directions.

$$\vec{G}(p) \quad = \quad \sum_{j=1}^{8} \vec{d}_j(p, q). \tag{3}$$

The magnitude of $\vec{G}$ is used as gradient-like image $\hat{I}$. In the case of colored images (the originals of Figures 7, 8, 9, and 12), we obtain one gradient-like image for each channel; red $\hat{I}_r$, green $\hat{I}_g$, and blue $\hat{I}_b$; and compute the final gradient value $I(p) = \max\{I_r(p), I_g(p), I_b(p)\}$ for all pixels $p \in D_I$.

## 6 Evaluation

This section evaluates TP for automatic image segmentation in two applications: (i) license plate segmentation and (ii) 3D MR-image segmentation of the human brain. The experiments also include WS and other approaches that have been specifically designed for these applications.

In (i), we use the magnitude of the Sobel's operator as gradient-like image. This choice illustrates our observation that WS is more dependent of the heterogeneity in the background than TP. The experiments also evaluate situations where candidate objects can be created by TP, and the desired one can be chosen by some object recognition technique. Although we use a simple template matching, the experiments show that this strategy deserves further investigation.

In (ii), we design a 3D gradient-like image based on the method described in Section 5, which satisfies the gradient condition of WS for 3D MR-brain image segmentation. This choice illustrates our observation that, in such a condition, WS and TP produce similar results.

## 6.1 License Plate Segmentation

The experiments used 990 images ($352 \times 240$ pixels) from a database of license plates. We wish to find the precise location and spatial extent of the plates (Figure 13a). We have chosen this application because the ground truth for the plates is easy, due to its known shape. Seed selection is a difficult task, because any attempt to estimate seeds inside a plate is likely to find seeds in other parts of the image. Besides, the gradient condition of TP is not satisfied inside the plates (Figure 13b), which requires that some pixels in $S_o$ be selected outside the plate numbers.

Therefore, a natural strategy is to run the methods for candidate seed sets, score the candidate objects, and choose the one with the best score. The score of an object can be obtained based on shape features, since the plates are deformed rectangles.

### 6.1.1 Seed selection

We have implemented three different methods for seed selection.

1. Seed selection based on thresholding and morphological operations.

   This approach exploits our knowledge about the problem and relies on parameters that are learned by inspection.

   Pixels of the plate's number can be usually enhanced by thresholding the image at 30% of its maximum intensity (Figure 13c). This threshold is reduced to 15% when no plate is detected. To get seeds inside the plate, we apply an erosion with a disk of radius 5.0, followed by a dilation with a disk of radius 7.0 (Figure 13d), and consider the components in Figure 13c which do not belong to Figure 13d. This choice of parameters is justified by the fact that 93% of the plates appear with similar sizes around 2936 pixels. The larger dilation radius was chosen to avoid seeds over the border of the plate. We also eliminate components that touch the image's border and components with less than 6 pixels. Seeds in the top region of the image (30% of the height) are rejected since there is no license plate there. The resulting components are finally dilated by a disk of radius 1 and labeled with a distinct number (Figure 13e). This last dilation was used to reduce the number of seed sets (connected components).

   In spite of these *ad-hoc* parameters, this seed selection approach estimates some seed set inside 100% of the plates in our database.

2. Exhaustive seed selection

   This method aims to minimize the *adhoc* parameters for seed selection. We define a rectangular seed set $S_o$ with $70 \times 10$ pixels and translate its center of $dx = 5$ and $dy = 5$ pixels from left to right and from top to bottom in the image to create about 3000 candidate sets. The values of these parameters are justified in the description of the score process below.

3. Seed selection based on the method proposed by Zheng et al [37] for plate location.

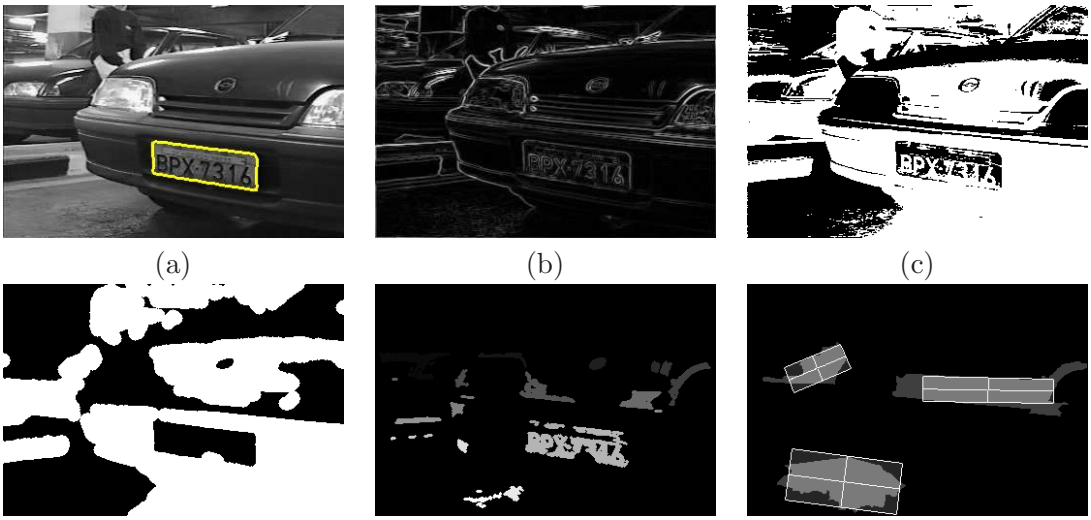


Figure 13: (a) Original image with the result of TP. (b) The magnitude of the Sobel's gradient. (c) Image (a) after thresholding. (d) Image (c) after erosion and dilation. (e) Image of labeled seeds. (f) The best rectangles for three false candidate objects.

The original image (Figure 14a) is first enhanced, then vertical edges are extracted using the Sobel's operator (Figure 14b). An edge density map is computed using a rectangular window (Figure 14c). The center of the plate is expected to be the highest value in this map.

Zheng et al [37] have shown 99.7% of correct segmentations in their database, among the candidates with the three highest values in the density map (i.e., the plate comes with two false positives). Their method was compared to four other approaches, which presented 88.83% of correct segmentations (the minimum was 82%) under the same conditions.

In our database, their method presented poor segmentation results (Figure 14a), because our plates appear with much more deformation than those of their database. However, their method locates 96.97% of the plates in our database among the best three candidates (regional maxima of the density map) and 98.58% among the best seven. This result motivated us to develop a method for seed estimation based on the Zheng's approach.

We first select the best seven candidates for plate location using the Zheng's approach. Given that these candidates may not be far from the border of the plate, we consider a rectangular seed set around them with $60 \times 10$ pixels and all possible translations from $dx = -5$ to $dx = 5$ pixels and $dy = -5$ to $dy = 5$ pixels to create candidate sets.

### 6.1.2 Object score

We have implemented two template-based methods for object scoring.

(a) (b) (c)

Figure 14: (a) Original image with the result of the method proposed by Zheng. (b) Vertical edges after filtering. (c) The edge density map.

1. Score by matching shapes with the best-fit rectangle

   This approach exploits our knowledge about the problem to estimate some parameters for scoring.

   During the plate's election we reject candidate objects with less than 1200 pixels or more than 8,200 pixels, since the size of the plates varies from 1,559 up to 7,753 pixels. To compute the score for the remaining candidates, we first find the best-fit rectangle. To guarantee rotation invariance, we determine the object's major semi-axis by principal component analysis. By starting at its geometric center, we go along the major semi-axis in both orientations, alternately and at same time, until we reach the background in some orientation. The same process is repeated to the orthogonal semi-axis. These boundary points give us all necessary information to trace a rectangular model (Figure 13f).

   Considering each fitted rectangle as a pseudo "ground truth", we compute the error $ER$ as the sum of *false negatives* and *false positives*, normalized by the area of the rectangle. Then, we compute a score $SC$ for the corresponding object based on $ER$, and the aspect ratio $R$ of the rectangle, as follows.

$$\begin{aligned} SC &= (1.0 - \min\{1.0, ER\}) \cdot W \\ W &= \begin{cases} 1.0 & \text{if } 2.4 \leq R \leq 3.0, \\ \exp\left(-(2.4 - R)^2/3.0\right) & \text{if } R < 2.4, \\ \exp\left(-(3.0 - R)^2/3.0\right) & \text{if } R > 3.0. \end{cases} \end{aligned} \tag{4}$$

   where the interval $[2.4, 3.0]$ was chosen in between the minimum and maximum expected aspect ratios of the plates.

2. Score by matching shapes using shape descriptors

   This approach aims to eliminate the parameters of the previous scoring process. We have selected two shape descriptors for our experiments: Beam Angle Statistics (BAS) [1] and segment saliences (SS) [31].

   A descriptor is defined here as a pair, *feature vector* and *distance metric*. The feature vector subsumes the object properties and the distance function measures the dissimilarity between two objects with respect to their properties. Each object can

| | Error | | Correct |
|---|---|---|---|
| | Scoring | Method | |
| TP | 28 (2.8%) | 39 ( 3.9%) | 923 (93.23%) |
| WS | 43 (4.3%) | 65 ( 6.6%) | 882 (89.09%) |
| PTP(T=1%) | 66 (6.7%) | 149 (15.0%) | 775 (78.28%) |
| PTP(T=2%) | 78 (7.9%) | 68 ( 6.9%) | 844 (85.25%) |
| PTP(T=5%) | 79 (8.0%) | 74 ( 7.5%) | 837 (84.54%) |

Table 1: Number of errors due to object scoring and each method, and the number of correctly segmented plates.

be interpreted as a 'point' in the underlying metric space, where similar objects form groups of points.

For a given shape descriptor, the basic idea is to represent each plate and candidate object by their corresponding feature vectors. The templates are feature vectors randomly extracted from 25 plates in the database (2.5%). We compute the similarity between the feature vector of a candidate object and the feature vector of each template to select the maximum similarity value. This process is repeated for each candidate object and the one with maximum similarity value is classified as plate.

The size $70 \times 10$ pixels of the rectangular seed set was chosen to guarantee seeds outside the plate's number and to avoid small rectangular objects, because these shape descriptors are scale invariants. The displacements $dx = 5$ and $dy = 5$ pixels were chosen to accelerate the entire process on a 2.8GHz PC: about 8 hours for BAS and 6 hours for SS to classify about 30,000,000 objects (750,000,000 matchings) extracted from the database (about 3,000 objects per image).

#### 6.1.3 Results

We have divided the experiments in three parts. The first part aimed to compare the effectiveness of TP with respect to WS and its previous version [11], named PTP in Table 1, where $T$ is an *ad-hoc* parameter. We used seed selection based on thresholding and morphological operations and object score by matching with the best-fit rectangle. Table 1 summarizes the results.

TP correctly segmented 923 (93.23%) license plates out of 990. The mean time for the entire segmentation process (seed estimation, TP, and scoring) was 1s per image, running on a AMD AthlonXP 2400+ (2GHz). TP alone took about 80ms per image. Among the 67 missed plates, 28 (2.8% of the database) were due to the scoring process and 39 (3.9% of the database) were due to errors of the TP method (i.e., the actual accuracy of TP was 96.06%).

Note that TP was more accurate than PTP for any fixed value $T$, because it is difficult to adjust $T$ in this application. Table 1 also shows that WS was more sensitive than TP with respect to the heterogeneity of the background.

The second part aimed to evaluate TP with a minimum of *adhoc* parameters for seed estimation and object scoring. We used the exhaustive seed search and the object scoring

| | Samples | Correct |
|---|---|---|
| BAS | 25 | 894 (90.30%) |
| SS | 25 | 888 (89.70%) |

Table 2: The number of correctly segmented plates using exhaustive seed search, TP, and shape descriptors for object scoring.

by matching shape descriptors. The accuracies shown in Table 2 were slightly lower than the highest one of Table 1 and higher than the others. Given that the templates represent only 2.5% of the database and each plate is selected among 3,000 objects per image, this can be considered an excellent result. Indeed, the approach of combining exhaustive TP with object recognition methods deserves further investigation, which will be feasible in a near future with the hardware of the IFT.

The third part aimed to investigate a possible combination of the Zheng-based seed estimation, TP and the object score by matching with the best-fit rectangle. The number of correctly segmented plates in this approach was 92.22% (slightly lower than the 93.23% obtained with seed selection by thresholding and morphological operations). However, we consider the Zheng-based seed selection better, because it is less sensitive to the color of the plates.

Finally, Figure 15 illustrates some cases where the Zheng's method fails in locating the plate as the best candidate, in spite of the correct location may be within the best seven candidates. Figures 16a-c illustrate why the Zheng's segmentation approach does not work for our database, and Figures 16d-o show some segmentation results using TP with seed selection by thresholding and morphological operations, and score based on the best-fit rectangle.

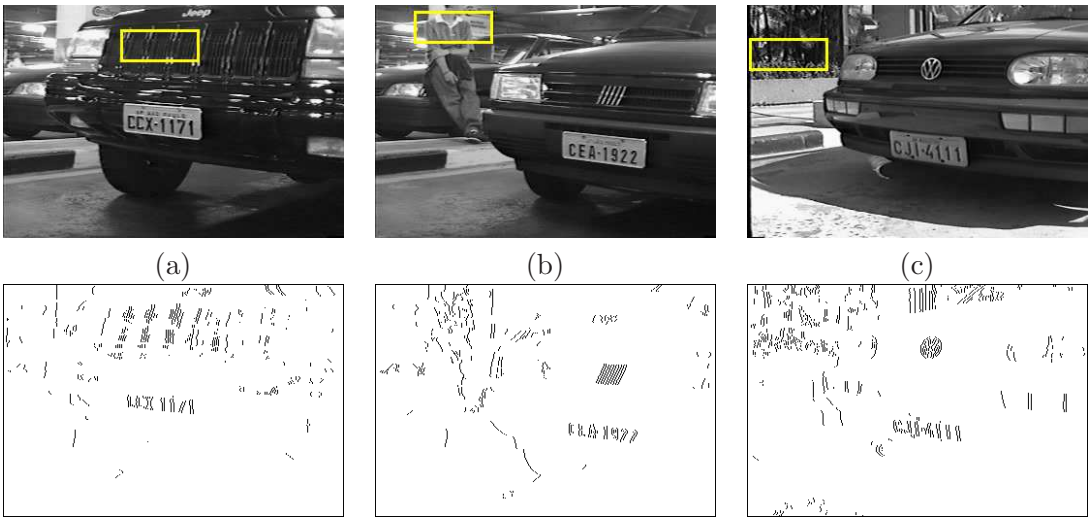



Figure 15: (a-c) Wrong plate detections by the Zheng's method. (d-f) The corresponding vertical edges after filtering.


Figure 16: (a-c) Very poor plate segmentations by the Zheng's method. (d-o) License plate segmentations by TP.

## 6.2 3D MR-brain segmentation

Segmentation of the human brain from Magnetic Resonance (MR) images is an open problem, which has been addressed in several different ways, each one with its pros and cons[19].

Figure 17 shows an MR-T1 slice image of the brain, where some structures are indicated. We wish to separate the gray matter (GM) and white matter (WM), as a single object, from the rest of the image. We have evaluated tree pruning using phantom MR-T1 images (which are available at the BrainWeb site[1] [6] together with their ground truth) and real MR-T1 images from control subjects.

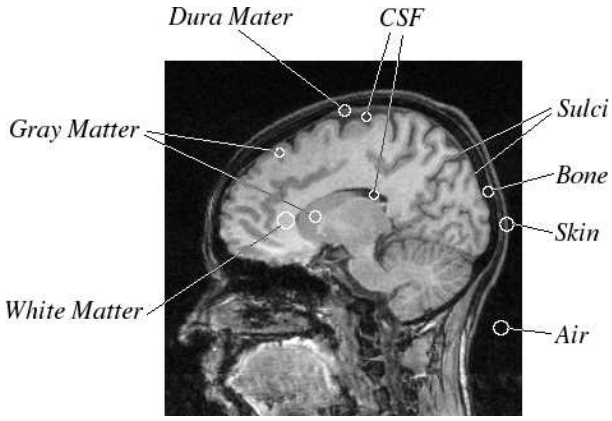



Figure 17: Tissues in MR-T1 images of the brain. CSF is the cerebrospinal fluid (liquor).

The experiments compare the results with some ground truth. We define FN as the *percentage of false negatives*— the number of pixels that belong to the object, but have been classified as background, divided by the number of object pixels— and FP as the *percentage of false positives*— the number of pixels classified as object, but that belong to the background, divided by the number of background pixels. The error is computed as the number of misclassified pixels divided by the total number of pixels in the image.

### 6.2.1 Seed selection and gradient estimation

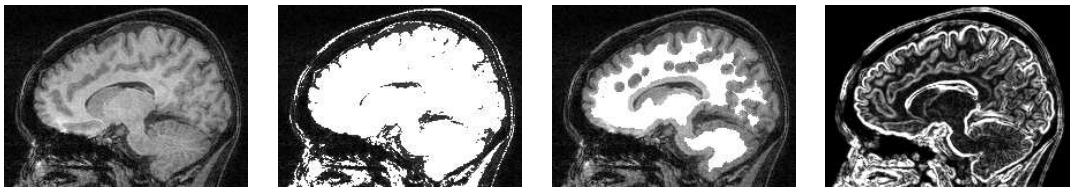
Figure 18a shows an MR-T1 slice image of the brain. Note that, GM and WM can not be obtained by automatic thresholding on Figure 18a and simple morphological operations on Figure 18b. On the other hand, a morphological erosion on Figure 18b with a sphere of radius 5 can assuredly separate GM and WM from other structures, and the largest connected component resulting from erosion can be used as a seed set inside the brain (Figure 18c).

The gradient-like image is computed as described in Section 5, but using a three-dimensional 6-neighborhood instead of the two-dimensional 8-neighborhood (Figure 18d).

### 6.2.2 Experiments with phantoms

On the first set of experiments, we generated 8 MR-T1 phantoms with varying noise and inhomogeneity (INU) settings (Figure 19), and automatically segmented them with tree

[1]URL: http://www.bic.mni.mcgill.ca/brainweb/



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

Figure 18: (a) Sample slice of MR-T1 image of the brain. (b) Binary slice resulting from Otsu's thresholding overlaid on image (a). (c) Binary slice of seed voxels overlaid on image (a). (d) Slice of the gradient image.

pruning (TP) and watershed (WS), using the same internal seeds, gradient-like image, and the image's border as background seeds for WS and set $B$ for TP. The results were compared with the available ground truth.

Since TP and WS detect the external boundary of the brain, the resulting objects still contain small CSF regions (sulci and ventricles). To properly classify them as background, we take the intersection between the pruned object and the voxels with intensities above the Otsu's threshold.

### 6.2.3 Experiments with real MR-T1 images

On the second set of experiments, we selected 20 MR-T1 images of control subjects with no known anomalies, acquired with a 2T Elscint scanner and voxel size of $0.98 \times 0.98 \times 1.00$ $mm^3$. All volumes were interactively segmented with differential watersheds [10] in order to provide a basis for comparison (a ground truth). The images were automatically segmented by 3 methods: TP, WS (with the image's border as external seeds), and SPM2 [18]— a widely used template-based approach for medical research.

### 6.2.4 Results

Table 3 shows the segmentation errors of TP and WS for the phantoms with respect to the available ground truth. Figure 20 shows 3D renditions of the obtained segmentations.

Table 4 shows the segmentation errors for the 3 methods on real images, using the interactive segmentation as the basis for comparison. Figure 21 shows renditions of the segmented brains.

We can observe the high degree of agreement among TP, WS and the ground truths in both experiments. Note that SPM2 provided segmentations with noticeable artifacts, such as disconnected components outside the brain (Figure 21). This illustrates how difficult is to segment our images.

TP and WS not only provided better segmentation results than SPM2 on the real images, but also were much faster. The entire segmentation task with TP (marker extraction, gradient computation, optimum-path forest computation, leaking point detection and tree

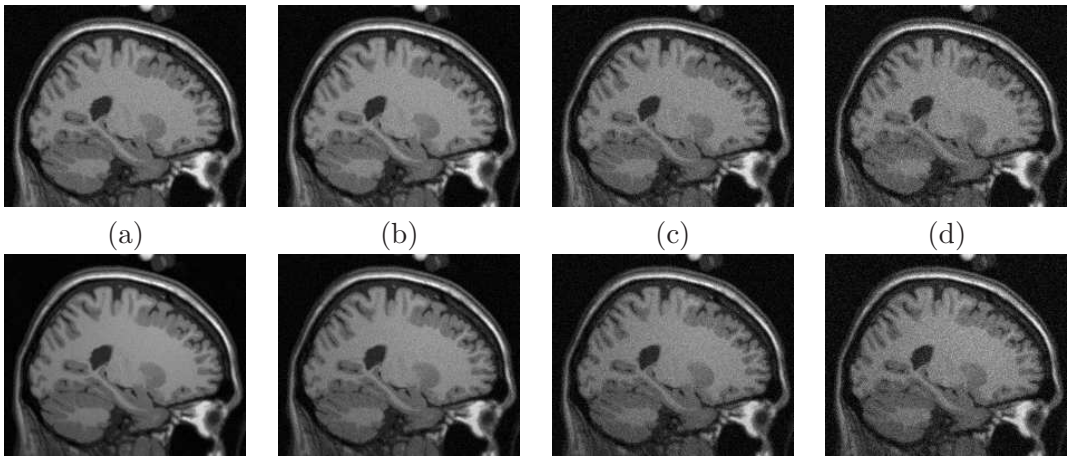


Figure 19: Sample slices of the phantoms with different degrees of noise (N) and inhomogeneity (INU): (a) $N = 3\%$ and $INU = 20\%$, (b) $N = 5\%$ and $INU = 20\%$, (c) $N = 7\%$ and $INU = 20\%$, (d) $N = 9\%$ and $INU = 20\%$, (e) $N = 3\%$ and $INU = 40\%$, (f) $N = 5\%$ and $INU = 40\%$, (g) $N = 7\%$ and $INU = 40\%$, and (h) $N = 9\%$ and $INU = 40\%$.

| | **Tree Pruning** | **Watershed** |
|---|---|---|
| Noise / INU | Error (FN,FP) | Error (FN,FP) |
| 3% / 20% | 1.17% (3.20%,0.59%) | 1.04% (2.53%,0.61%) |
| 5% / 20% | 1.14% (3.14%,0.57%) | 1.00% (2.55%,0.56%) |
| 7% / 20% | 1.19% (3.65%,0.49%) | 1.06% (2.96%,0.52%) |
| 9% / 20% | 1.36% (4.48%,0.46%) | 1.23% (3.85%,0.48%) |
| 3% / 40% | 1.20% (3.16%,0.63%) | 1.05% (2.45%,0.65%) |
| 5% / 40% | 1.17% (3.44%,0.53%) | 1.07% (2.73%,0.59%) |
| 7% / 40% | 1.28% (4.06%,0.49%) | 1.15% (3.38%,0.52%) |
| 9% / 40% | 1.51% (5.23%,0.44%) | 1.39% (4.66%,0.45%) |
| Average | 1.25% (3.93%,0.51%) | 1.12% (3.14%,0.55%) |

Table 3: Brain phantom segmentation errors with tree pruning and watershed.

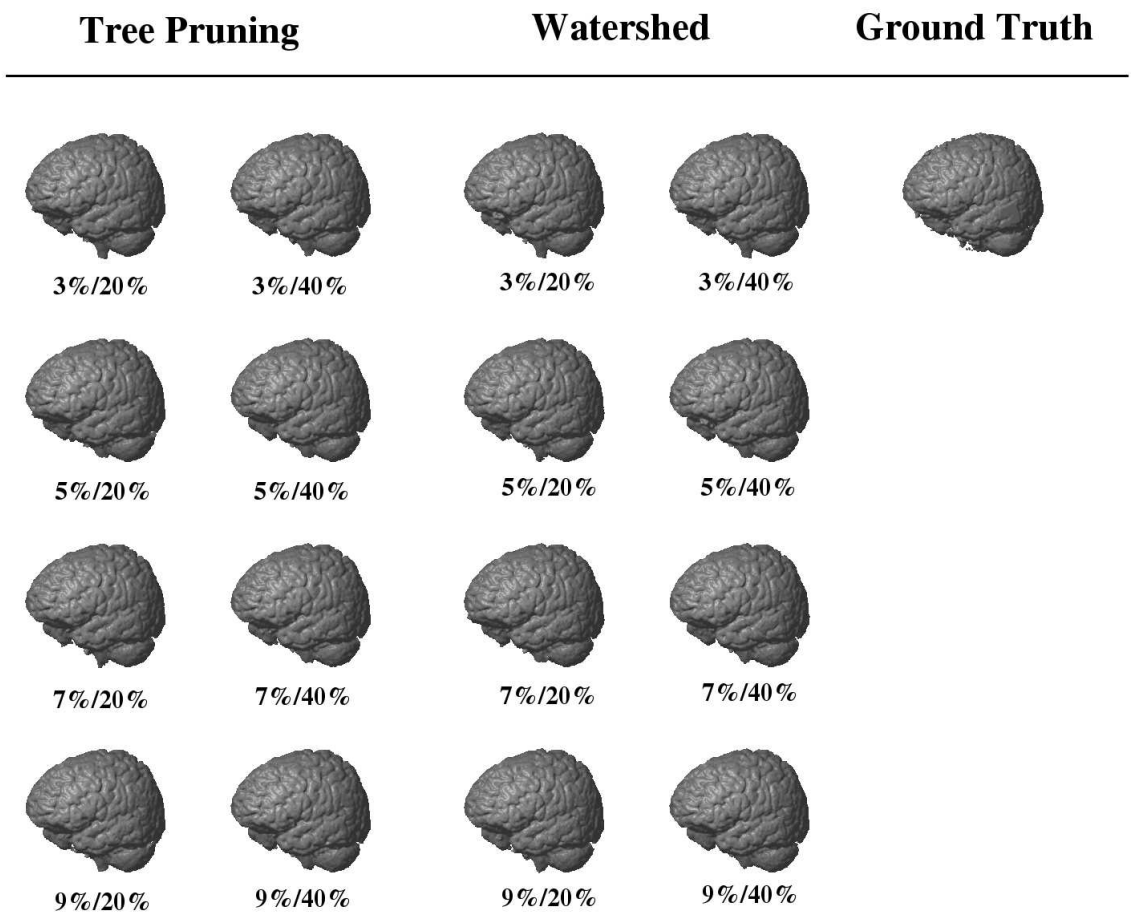



Figure 20: Renditions of the 8 brain phantom segmentations by tree pruning (first and second columns), watershed (third and fourth columns) and of the provided ground truth (fifth column).

| Subject | **Tree Pruning** Error (FN,FP) | **Watershed** Error (FN,FP) | **SPM2** Error (FN,FP) |
|---|---|---|---|
| 01 | 0.90% (2.88%, 0.41%) | 0.76% (2.51%, 0.33%) | 1.45% (4.87%, 0.60%) |
| 02 | 1.17% (1.62%, 1.05%) | 1.10% (1.42%, 1.02%) | 2.02% (5.85%, 1.03%) |
| 03 | 1.10% (2.33%, 0.77%) | 1.02% (1.80%, 0.81%) | 1.86% (4.31%, 1.22%) |
| 04 | 1.33% (3.46%, 0.76%) | 0.87% (3.04%, 0.29%) | 1.78% (6.39%, 0.56%) |
| 05 | 1.36% (2.34%, 1.05%) | 1.41% (2.06%, 1.21%) | 2.90% (10.21%, 0.64%) |
| 06 | 1.25% (2.39%, 0.88%) | 1.08% (2.05%, 0.77%) | 2.18% (4.05%, 1.57%) |
| 07 | 1.00% (1.55%, 0.87%) | 1.03% (1.27%, 0.97%) | 1.50% (3.89%, 0.95%) |
| 08 | 1.17% (3.36%, 0.59%) | 0.98% (2.87%, 0.49%) | 1.88% (5.37%, 0.97%) |
| 09 | 1.28% (2.70%, 0.89%) | 1.28% (2.33%, 1.00%) | 1.74% (5.21%, 0.81%) |
| 10 | 1.57% (1.80%, 1.52%) | 1.51% (1.56%, 1.50%) | 1.81% (5.70%, 0.99%) |
| 11 | 0.98% (1.82%, 0.77%) | 0.97% (1.56%, 0.82%) | 1.88% (5.09%, 1.06%) |
| 12 | 1.65% (2.22%, 1.51%) | 1.65% (1.82%, 1.60%) | 2.08% (5.20%, 1.30%) |
| 13 | 1.21% (1.98%, 1.01%) | 1.15% (1.56%, 1.05%) | 1.81% (4.62%, 1.07%) |
| 14 | 1.40% (3.89%, 0.74%) | 1.32% (3.43%, 0.76%) | 1.97% (6.00%, 0.91%) |
| 15 | 0.91% (2.07%, 0.58%) | 0.78% (1.78%, 0.50%) | 1.58% (4.99%, 0.62%) |
| 16 | 1.39% (2.45%, 1.05%) | 1.80% (2.20%, 1.67%) | 2.16% (5.20%, 1.17%) |
| 17 | 0.98% (2.94%, 0.50%) | 0.93% (2.61%, 0.52%) | 1.64% (5.01%, 0.83%) |
| 18 | 0.95% (2.81%, 0.39%) | 0.85% (2.61%, 0.31%) | 2.29% (7.92%, 0.59%) |
| 19 | 1.43% (1.41%, 1.44%) | 1.29% (1.29%, 1.28%) | 2.09% (4.17%, 1.54%) |
| 20 | 2.12% (1.85%, 2.20%) | 1.89% (1.70%, 1.94%) | 2.46% (3.31%, 2.23%) |
| Average | 1.26% (2.39%, 0.95%) | 1.18% (2.07%, 0.94%) | 1.95% (5.37%, 1.03%) |

Table 4: Real brain segmentation errors with tree pruning, watershed and SPM2.

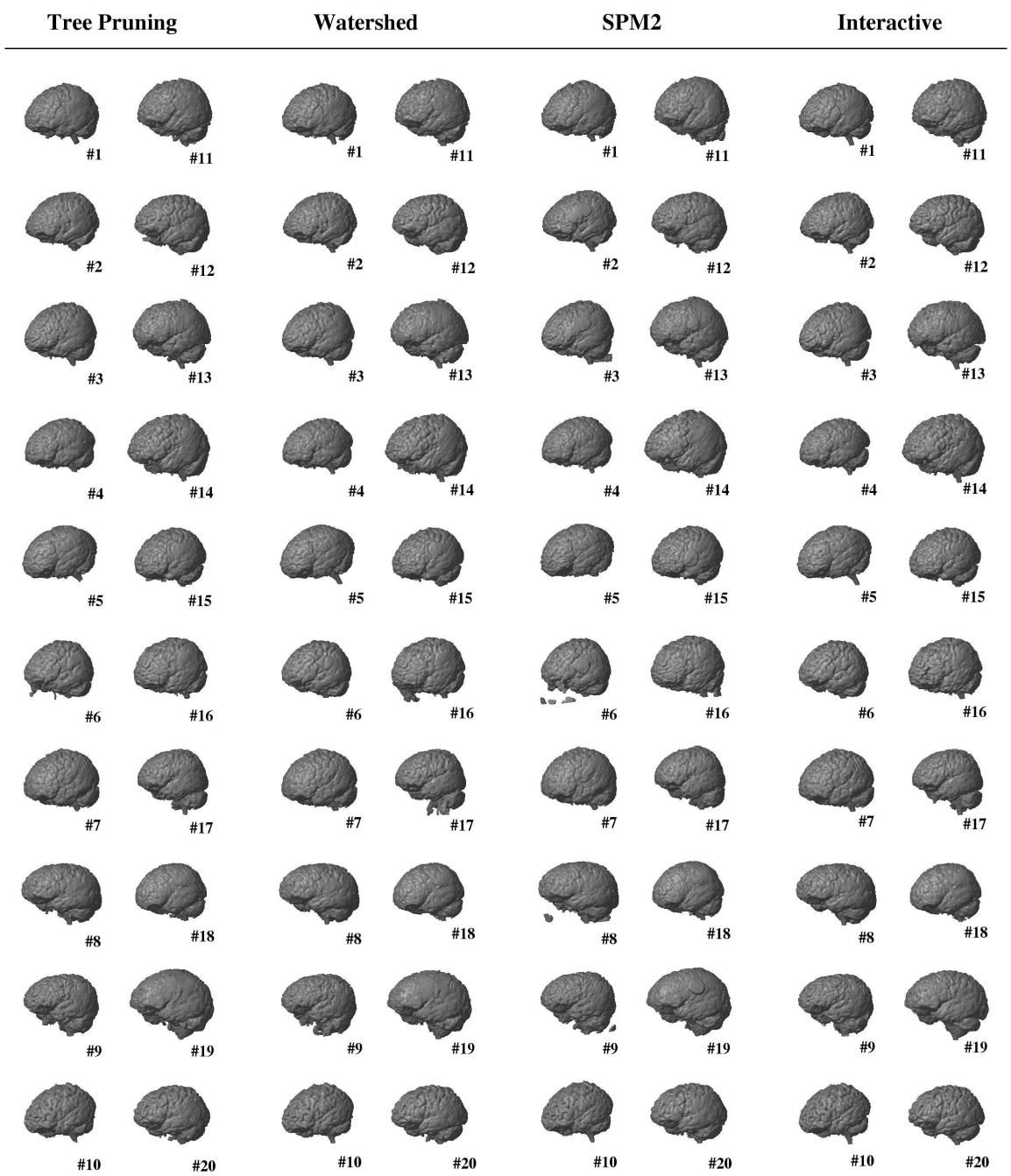



Figure 21: Brain segmentation results with tree pruning, watershed and SPM2 for 20 control subjects, shown as 3D renditions.

pruning), for example, took 25 seconds for a $181 \times 217 \times 181$ MR-T1 volume, while SPM2 took 5 minutes, on the same Athlon64 3200+ workstation.

The background is homogeneous in most part of these MR-brain images and there are no strong barriers between the brain's border and the image's border. This favored gradient-like images that satisfy the gradient condition for WS. As we expected, such a situation makes TP and WS to provide similar results. However, WS performed slightly better than TP in both experiments. Their segmentation differences are mostly voxels of a thin layer over the brain's cortex, indicating off-by-one errors in the location of the object's boundary. In the case of real images, the slight advantages of WS in performance are better explained due to the fact that the ground truth was created using the interactive watershed on the same 3D gradient-like image [10].

## 7 Conclusion

We have presented a considerably extended version of our previous work on tree-pruning segmentation [11, 26], which adds a formal definition of the obtained objects; several new examples; algorithms; a discussion about robustness conditions and geometrical issues; an improved gradient computation with respect to classic approaches, such as the Sobel's operator; and more evaluation experiments that compare tree pruning (TP) with watershed (WS) and other approaches for automatic license plate image segmentation and 3D MR-image segmentation of the human brain.

We have shown that TP can provide highly accurate results in both applications, is less sensitive than WS with respect to the heterogeneity of the background, and that both approaches provide similar results under the gradient condition for WS.

If we consider a more general definition for image segmentation, as two tightly coupled tasks; *delineation* and *recognition* [13]; we may conclude that possible combinations between TP and higher level approaches [8, 7, 35, 9, 23, 20] can lead to more effective methods for automatic segmentation.

Delineation aims to define the precise spatial extent of an object in the image while its approximate location (e.g., seed estimation) is a recognition task. Recognition also involves other cognitive tasks, such as to verify the correctness of a segmentation result or to identify a desired object among candidate ones. TP is likely to outperform higher level approaches in delineation, but the other way around may be verified in recognition. In this sense, these methods should complement each other for more effective image segmentation. Our future work goes in this direction.

## Acknowledgments

The authors thank Roberto Lotufo (FEEC-UNICAMP) for the images of license plates and Laurent Najman (ESIEE) for his comments on this paper. This work was supported by CNPq (Proc. 302427/04-0), CAPES and FAPESP (Proc. 03/09793-1 and Proc. 03/13424-1).

## References

[1] N. Arica and F. T. Y. Vural. BAS: A Perceptual Shape Descriptor based on the Beam Angle Statistics. *Pattern Recognition Letters*, 24(9-10):1627–1639, Jun 2003.

[2] R. Audigier, R.A. Lotufo, and A.X. Falcão. 3D visualization to assist iterative object definition from medical images. *Computerized Medical Imaging and Graphics*, 30(4):217–230, 2006.

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

[4] Y. Y. Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 105–112, 2001.

[5] L.D. Cohen. On active contour models and ballons. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2):211–218, 1991.

[6] D.L. Collins, A.P. Zijdenbos, V. Kollokian, J.G. Sled, N.J. Kabani, C.J. Holmes, and A.C. Evans. Design and construction of a realistic digital brain phantom. *IEEE Trans. on Medical Imaging*, 17(3):463–468, June 1998.

[7] T. Cootes, G. Edwards, and C.J.Taylor. Active appearance models. In *European Conference on Computer Vision (ECCV)*, volume 2, pages 484–498, 1998.

[8] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models – their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[9] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* Wiley-Interscience, 2000.

[10] A. X. Falcão and F. P. G. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Trans. on Medical Imaging*, 23(9):1100–1108, 2004.

[11] A. X. Falcão, F. P. G. Bergo, and P. A. V. Miranda. Image segmentation by tree pruning. In *XVII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 65–71. IEEE, Oct 2004.

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

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

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

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

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

[17] L. Ford and D. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[18] R.S.J. Frackowiak, K.J. Friston, C. Frith, R. Dolan, C.J. Price, S. Zeki, J. Ashburner, and W.D. Penny. *Human Brain Function*. Academic Press, 2nd edition, 2003.

[19] V. Grau, A. U. J. Mewes, M. Alcaniz, R. Kikinis, and S. K. Warfield. Improved watershed transform for medical image segmentation using prior information. *IEEE Trans. on Medical Imaging*, 23(4):447–458, Apr 2004.

[20] S. Haykin. *Neural Networks: A comprehensive foundation*. Prentice-Hall, 1998.

[21] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1987.

[22] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):147–159, Feb 2004.

[23] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

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

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

[26] P. A. V. Miranda, F. P. G. Bergo, L. M. Rocha, and A. X. Falcão. Tree-pruning: A new algorithm and its comparative analysis with the watershed transform for automatic image segmentation. In *XIX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 37–44. IEEE, Oct 2006.

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

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

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

[30] Jiambo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.

[31] R. S. Torres, A. X. Falcão, and L. F. Costa. A graph-based approach for multiscale shape analysis. *Pattern Recognition*, 37(6):1163–1174, 2004.

[32] R.S. Torres and A.X. Falcão. Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3–13, Jan 2007.

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

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

[35] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Intl Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511–I–518, 2001.

[36] Song Wang and Jeffrey Mark Siskind. Image segmentation with minimum mean cut. In *Intl. Conf. on Computer Vision (ICCV)*, volume 1, pages 517–525, Jul 2001.

[37] D. Zheng, Y. Zhao, and J. Wang. An efficient method of license plate location. *Pattern Recognition Letters*, 2005. Article in press, available at www.sciencedirect.com.