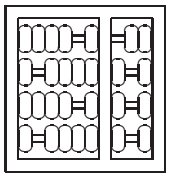


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

**Automatic object detection by tree pruning**

*A.X. Falcão* *P.A.V. Miranda*
*F.P.G. Bergo*

Technical Report - IC-05-19 - Relatório Técnico

September - 2005 - Setembro

# Automatic object detection by tree pruning

Alexandre X. Falcão[1] Paulo A.V. Miranda[1] Felipe P.G. Bergo[1]

[1]Instituto de Computação, Universidade Estadual de Campinas
Av. Albert Einstein, 1251, CEP 13084-970, Campinas, SP, Brasil

**Abstract**

The *Image Foresting Transform* (IFT) has been presented for the design of image processing operators based on connectivity. The IFT reduces image processing problems into a minimum-cost path forest problem in a graph derived from the image. In this paper we propose a new image operator, which solves segmentation by pruning trees of the forest. An IFT is applied to create a minimum-cost path forest whose roots are *seed pixels*, selected inside a desired object. In this forest, the background consists of a few subtrees rooted at pixels (*leaking points*) on the object's boundary. The leaking pixels are identified and their subtrees are eliminated, such that the remaining forest defines the object. Tree pruning reduces image segmentation to the choice of a few pixels in the image, favoring solutions for automatic object detection. We present a user-friendly way of identifying leaking pixels and give solutions for their automatic detection. Since automatic seed selection may be different for each application, we evaluate automatic segmentation with tree pruning in three situations: labeling of multiple objects with similar textures, 3D object definition, and shape-based object detection. The results indicate that tree pruning is a promising approach to investigate automatic image segmentation.

## 1 Introduction

Object detection is a hard and important problem in image analysis, which very often requires user intervention. We have investigated methods for object detection that reduce user intervention to simple selection of a few pixels in the image [1–3], such that automatic solutions become feasible for some applications.

In this paper we propose an image segmentation approach, called *tree pruning*. Tree pruning is based on the *Image Foresting Transform* (IFT)— a general tool for the design, implementation, and evaluation of image processing operators based on connectivity [4]. In the IFT, an image is interpreted as a graph whose nodes are image pixels and whose arcs are defined by an *adjacency relation* between pixels. For 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. That is, each rooted tree consists of pixels more strongly connected to its root than to any other seed. In tree pruning, the seeds must be chosen inside the object and the choice of the path-cost function intends to connect object and background by a few optimum paths, which cross the object's boundary through its "most

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

Figure 1: Watershed transform by markers: (a) Seed selection inside (white line) and outside (black line) the object (darker pepper). (b) First segmentation result. (c) Correction with new seeds. (d) Final segmentation result.

weakly connected" parts (called *leaking pixels*). The topology of the forest is exploited to identify the leaking pixels and eliminate their subtrees, such that the remaining forest defines the object.

In comparison to other optimal region growing approaches that use only internal seeds, there is no stop criterion based on local image properties. In [5, 6], for example, the "strength of connectedness" between a pixel and its "most strongly connected" seed is inversely proportional to the cost of an optimum path between them. In the IFT algorithm, the "cost of a pixel" is the cost of the first optimum path that reaches it. The region growing process can stop when the cost of a pixel is above a certain threshold. The method fails when there are optimum paths reaching internal and external pixels with the same cost. In tree pruning, the criterion to disconnect object and background is global and can only be determined afterward. The optimum paths that reach object pixels are assumed to not pass through the background, except the leaking paths where the costs of the leaking pixels are usually equal to the costs of pixels in the neighborhood outside the object.

In tree pruning, internal seeds compete among themselves and only a few seeds become roots of leaking paths. Other optimal region growing approaches based on seed competition [7–10] can be roughly described in three steps: (i) seed pixels are selected inside and outside the objects (Figure 1a), (ii) each seed defines an *influence zone* which consists of pixels that 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. Due to the absence of boundary information and/or heterogeneity of the background— situations very common in practice— the leaking also occurs in these approaches, as illustrated in Figure 1b using a watershed transform [11, 12]. The correction is usually interactive, where the user selects seeds in both sides along the leaking parts of the boundary (Figures 1c and 1d). Tree pruning differs from these approaches in two aspects: it does not require external seeds (which simplifies seed selection) and exploits the leaking problem to solve segmentation automatically.

Tree pruning has been previously presented in [13]. The present paper provides more details about the method, includes two solutions for automatic identification of leaking points, and presents several validation experiments involving automatic object detection in

three situations: labeling of multiple objects with similar textures, 3D object definition, and shape-based object detection.

Section 2 reviews some concepts of the IFT needed to introduce the tree-pruning approach. The method is described in Section 3 with examples that illustrate its topological properties, advantages, and limitations. The algorithms are presented in Section 4 and Section 5 shows the results for automatic object detection. We state conclusion and discuss future work in Section 6.

## 2 Image Foresting Transform

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

An *adjacency relation* $\mathcal{A}$ is a binary irreflexive relation between pixels $p$ and $q$ of $D_I$. We use $q \in \mathcal{A}(p)$ and $(p, q) \in \mathcal{A}$ to indicate that $q$ is adjacent to $p$. Once the adjacency relation $\mathcal{A}$ has been fixed, the image $\hat{I}$ can be interpreted as a directed graph $(D_I, A)$ whose nodes are the image pixels in $D_I$ and whose arcs are the pixel pairs $(p, q)$ in $\mathcal{A}$. For example, one can take $\mathcal{A}$ to consist of all pairs of pixels $(p, q) \in D_I \times D_I \setminus \{(0, 0)\}$ such that $d(p, q) \leq \rho$, where $d(p, q)$ denotes the Euclidean distance and $\rho$ is a specified constant (i.e., 4-adjacency, when $\rho = 1$, and 8-adjacency, when $\rho = \sqrt{2}$).

A *path* is a sequence $\pi = \langle p_1, p_2, \ldots, p_n \rangle$ of pixels where $(p_i, p_{i+1}) \in \mathcal{A}$, for $1 \leq i \leq n-1$. The path is *trivial* if $n = 1$. Let $org(\pi) = p_1$ and $dst(\pi) = p_n$ be the origin and destination of a path $\pi$. If $\pi$ and $\tau$ are paths such that $dst(\pi) = org(\tau) = p$, we denote by $\pi \cdot \tau$ the concatenation of the two paths, with the two joining instances of $p$ merged into one. In particular, $\pi \cdot \langle p, q \rangle$ is a path resulting from the concatenation of its longest prefix $\pi$ and the last arc $(p, q) \in \mathcal{A}$.

A *predecessor map* is a function $P$ that assigns to each pixel $q \in D_I$ either some other pixel in $D_I$, or a distinctive marker *nil* not in $D_I$ — in which case $q$ is said to be a *root* of the map. A *spanning forest* is a predecessor map which contains no cycles — in other words, one which takes every pixel to *nil* in a finite number of iterations. For any pixel $q \in D_I$, a spanning forest $P$ defines a path $P^*(q)$ recursively as $\langle q \rangle$, if $P(q) = nil$, or $P^*(p) \cdot \langle p, q \rangle$ if $P(q) = p \neq nil$.

A *path-cost function* $f$ assigns to each path $\pi$ a *path cost* $f(\pi)$, in some totally ordered set $V$ of cost values, whose maximum element is denoted by $+\infty$. A path $\pi$ is *optimum* if $f(\pi) \leq f(\tau)$ for any other path $\tau$ with $dst(\tau) = dst(\pi)$, irrespective of its starting point. The IFT establishes some conditions applied to optimum paths, which are satisfied by only *smooth* path-cost functions. That is, for any pixel $q \in D_I$, there must exist an optimum path $\pi$ ending at $q$ which either is trivial, or has the form $\tau \cdot \langle p, q \rangle$ where

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

(C2) $\tau$ is optimum,

(C3) for any optimum path $\tau'$ ending at $p$, $f(\tau' \cdot \langle p, q \rangle) = f(\pi)$.

The IFT takes an image $\hat{I}$, a smooth path-cost function $f$ and an adjacency relation $\mathcal{A}$; and returns an *optimum-path forest*— a spanning forest $P$ such that $P^*(q)$ is optimum for every pixel $q \in D_I$. In the forest, there are three important attributes for each pixel: its predecessor in the optimum path, the cost of that path, and the corresponding root (or some label associated with it). In this paper, we are interested in the predecessor map.

## 3 Tree-pruning segmentation

We start from a rationale similar to that of a watershed transform [11, 12]. In the watershed transform, an object's boundary is a watershed line resulting from the following process. The method assumes internal and external seed pixels with one source of water at the location of each seed. It simulates a flooding process over the topographic surface of a gradient-like image and erects barriers (watershed lines) wherever two bodies of water coming from internal and external sources meet.

In tree pruning, we have only internal sources and let the water leak to the background through the lower pixels on the object's boundary. In practice, boundaries do not usually have the same height and these leaking pixels are not so many. That is, the streams of water reaching the background usually pass through a few leaking pixels. The identification of the leaking pixels allows the erection of a barrier at their location, separating the water inside from the water outside the object. This process can be implemented using the IFT as follows.

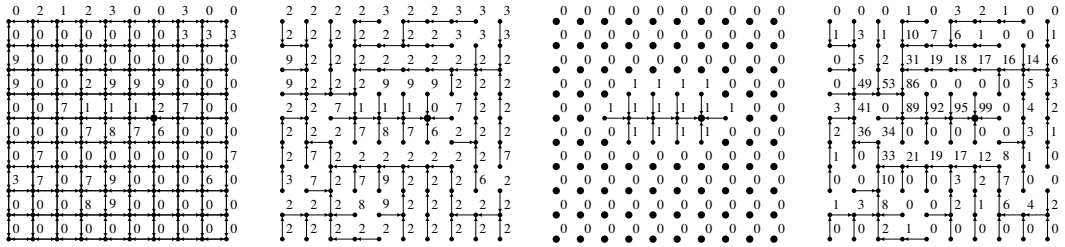
For a given set $S \subset D_I$ of seed pixels inside the object, we are interested in simple connectivity relations to $S$, such as those based on 4- or 8-adjacency (Figure 2a). The cost-path function $f_{\max}$ is a suitable function to simulate the aforementioned flooding process.

$$\begin{aligned} f_{\max}(\langle q \rangle) &= \begin{cases} 0, & \text{if } q \in S, \\ \infty, & \text{otherwise.} \end{cases} \\ f_{\max}(\pi \cdot \langle p, q \rangle) &= \max\{f_{\max}(\pi), \delta(p, q)\}, \end{aligned} \tag{1}$$

where the dissimilarity function $\delta(p, q)$ may be the magnitude of some gradient at arc $(p, q)$ or a gradient intensity at pixel $q$ (e.g., we define $\delta(p, q) = I(q)$ in Figure 2a).

**The dissimilarity (gradient) condition:** An important condition is that $\delta(p, q)$ should be higher across the object's boundary than inside the object and in the neighborhood outside it. Clearly, if this condition is satisfied, the leaking paths will cross the object's boundary through arcs with lower dissimilarity values and the leaking point for an arc $(p, q)$ will be $q$ (e.g., pixel (4,4) in Figure 2b), in the case of $\delta(p, q)$ is a gradient intensity at $q$, or $p$ otherwise. Moreover, the location of the internal seeds does not affect the location of the leaking pixels. By eliminating the subtrees rooted at the leaking pixels, the remaining forest defines the object (Figure 2c). Therefore, the problem becomes the detection of the leaking pixels.

By counting the number of pixels in the subtrees rooted at each pixel of the predecessor map (descendant count), one can observe that (Figure 2d): (i) optimum paths that reach the background have pixels with a higher number of descendants (leaking paths), (ii) the



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

Figure 2: (a) Four-connected graph where the numbers are gradient intensities and the bigger dot denotes a seed inside an object. (b) An optimum path tree rooted at that seed. (c) The object can be defined by eliminating the subtree of the leaking pixel at position (4,4). (d) Descendant count for every pixel in the tree shown in (b).

descendant count decreases considerably (from 86 to 31 and 53) when these leaking paths leave the object, and (iii) the leaking pixels (such as the one with 86 descendants) are the last pixels before this fall. The fall in the descendant count is due to the *first-in-first-out* (FIFO) tie-breaking policy used in the IFT algorithm [4]. 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. After leaking, this condition makes ambiguous the pixels in the neighborhood outside the object, ramifying the leaking paths into several branches (see Figure 2b).

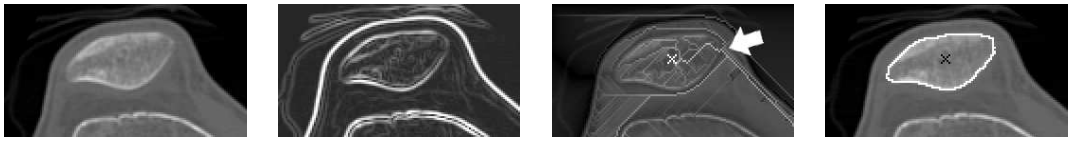
The aforementioned topological properties of the forest can be exploited to detect leaking pixels interactively and automatically.

### 3.1 Interactive identification of leaking points

Figure 3a shows part of a Computerized Tomography (CT) image of a knee, where the patella is the object of interest. The magnitude of the Sobel's gradient is shown in Figure 3b. Leaking paths can be visually identified by displaying the image with a colored overlay, where the color intensity of each pixel is proportional to the descendant count in the forest (Figure 3c). Since the color intensities of the optimum paths that reach the background are higher inside the object, the leaking pixels can be easily identified when these intensities are reduced across the object's boundary. Indeed the background is usually more strongly connected to the object through a few leaking paths, and the object can be completely disconnected from the background by breaking these paths at those leaking pixels (Figure 3d).

The interactive process is very dynamic for 2D images. Figure 4 illustrates a sequence of prunings to get the object of Figure 1. In this process, the user can move the mouse over the image and a program can simulate (with no noticeable delay) the pruning of each subsequent subtree, whose root is the pixel at the current position of the cursor. Once the cursor properly identifies a leaking pixel, the user can click on its position to commit that pruning operation. The user can make as many prunings as necessary to complete the segmentation process (Figures 4b-c).

An interesting effect occurs when there are multiple neighboring objects. The competi-

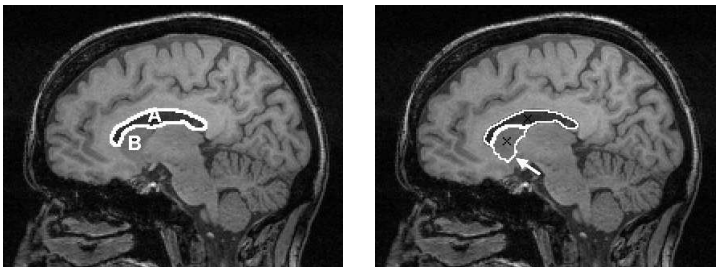


(a) (b) (c) (d)

Figure 3: (a) Patella in a CT image of a knee. (b) Gradient intensity image. (c) Graphic representation of the descendant count for an optimum path tree rooted at an internal seed (cross). A pruning at the pixel indicated by the arrow leads to the result shown in (d).


(a) (b) (c)

Figure 4: (a) Graphic representation of the descendant count for an optimum path tree rooted at a pixel A inside the darker pepper. (b) Result after two prunings at pixels indicated by the arrows. (c) Result after five more prunings as indicated by the arrows.




(a) (b)

Figure 5: Multiple object segmentation with tree pruning for an MR-image of the brain. (a) Seeds A and B are selected inside the ventricle and the caudate nucleus, respectively. After running the IFT, the competition already solves the segmentation of the ventricle. (b) The caudate nucleus is detected by one pruning as indicated by the arrow.

tion among the objects, as represented by their respective seeds, may solve the segmentation for some of them, such that the identification of leaking pixels will be needed for only the remaining unsolved objects. In this case, the seed competition among distinct objects reduces the number of leaking pixels (i.e., user involvement). Figure 5 shows an example with an MR-image of the brain, where the caudate nucleus and the ventricle are desired objects.

Curiously, the method presents limitations in perfect situations, such as the one illustrated in Figure 6a, because almost all boundary pixels are leaking pixels. A similar problem occurs when the weaker parts of the boundary are perfect gaps (Figure 6b). However, in the presence of noise, the method works for boundaries with gaps (Figure 6c). Fortunately, noise is present in most real situations (Figure 6d), considerably reducing the number of leaking pixels.

Another problem may occur when multiple objects with similar gradient intensities are very close to each other. The abrupt fall in the descendant count may be located outside the object, creating a false leaking pixel, because the absence of space in between the objects to ramify the path at the real leaking point (Figures 6e and 6f).

### 3.2 Automatic detection of leaking points

A first approach for automatic detection of leaking points is to isolate the segments of leaking paths whose pixels have descendant count above a threshold $T$ [13]. Figure 7a illustrates an example where leaking segments (white lines) are correctly isolated with $T = 10\%$ of the number of image pixels. By starting from the end of each segment and following the predecessor pixel up to the corresponding root, we can find the most prominent zero-crossing transition in the second derivative of the descendant count. The white dots in Figure 7a indicate all detected points and the one closest to the root usually occurs at the desired leaking pixel. Since its subtree contains the subtree of the other points, the object can be obtained by pruning the subtrees rooted at all detected pixels (Figure 7b). The method usually works even when there are multiple leaking points.

A drawback in this approach is the threshold $T$, which may be different in applications

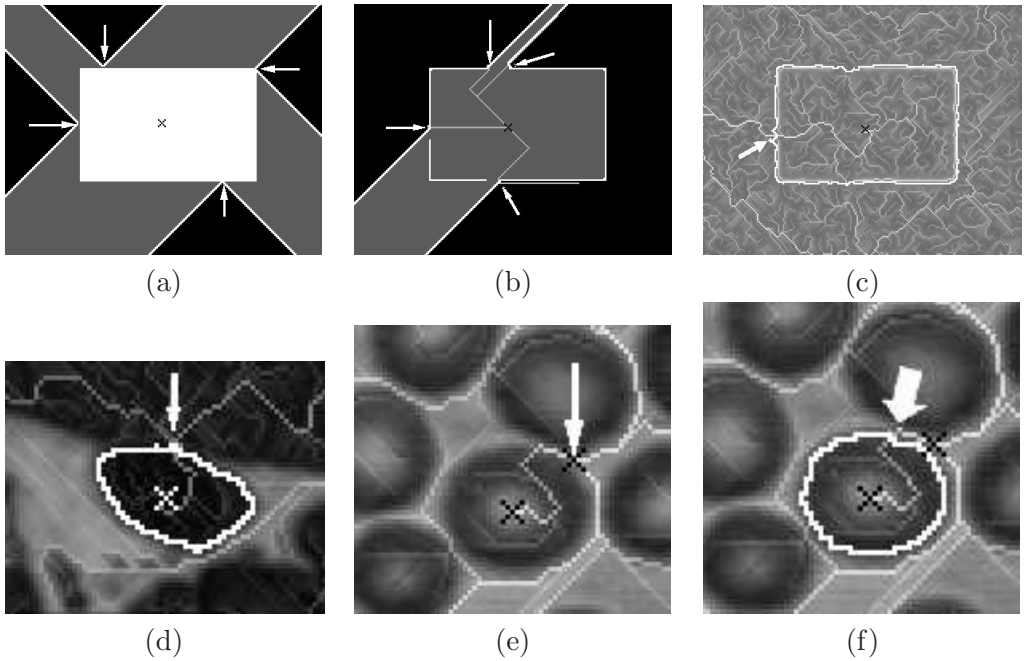


Figure 6: Issues with tree pruning: (a) Perfect object border and (b) perfect gaps after four prunings (arrows). (c) Image (b) with added noise, where the object is obtained with a single pruning (arrow). (d) A real example of (c) using an MR-image of a vein of the wrist. (e) Image of blood cells where a false leaking point (arrow) is detected outside a selected cell. (f) The cell is detected with a pruning at the real leaking point (arrow).

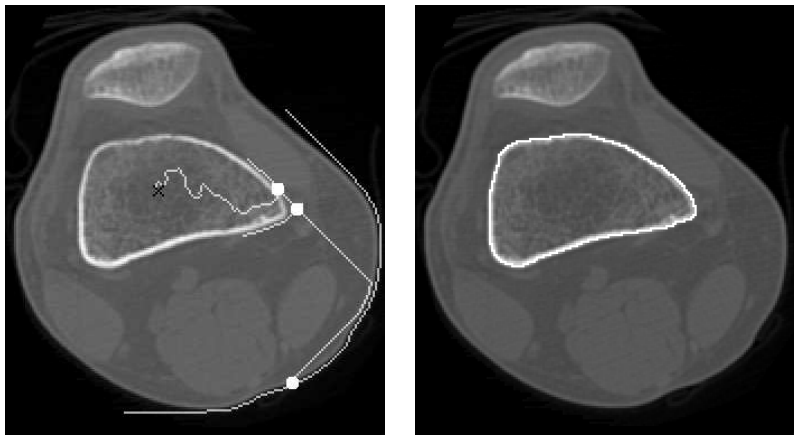
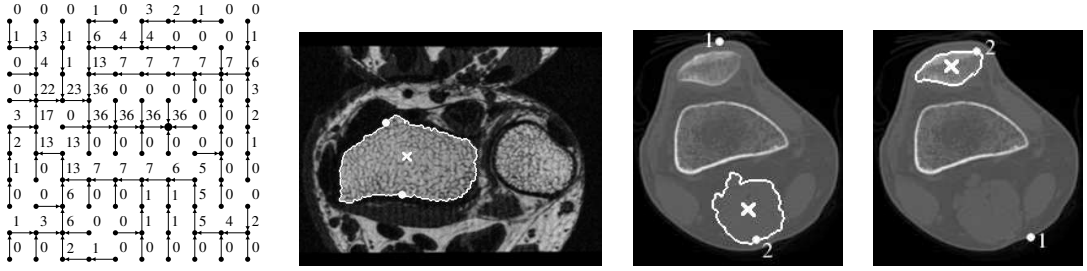


Figure 7: A CT-image of a knee is used to illustrate a first approach for automatic leaking point detection, where the femur is the desired object. (a) Leaking segments are identified by white lines and the most prominent zero-crossing transitions in the second derivative of the descendant count are indicated by dots. (b) Segmentation result after pruning at all detected points.




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

Figure 8: Second approach of automatic leaking point detection: (a) The number of descendants in the image's border for the optimum path tree of Fig. 2b. The descendant count at the leaking point (4,4) is 36, the size of the image's border. (b) An MR-image of a wrist where the larger bone is detected by two automatic prunings (dots) in a single iteration. (c)-(d) CT-image of a knee where a soft tissue and the patella are detected, respectively, in two iterations with one pruning (dot) per iteration.

with objects of too varied sizes. We can circumvent this problem with a second approach, which does not depend on ad-hoc parameters. By computing the number of descendants in the image's border for every pixel, we extract a new information from the topology of the forest— the sum of the new descendant count at the leaking pixels must be the size of the border, because all paths that reach the image's border have to pass through the leaking pixels (Figure 8a). We first tried to exploit this information, but we observed that, by starting from any pixel in the image's border and following its optimum path backwards to the respective root, the first occurrence of the maximum descendant count is usually at a leaking pixel. Therefore, leaking pixels can be automatically detected by repeating this procedure for every pixel of the image's border. Figure 8b illustrates a result of the method for two leaking pixels (white dots). However, in situations such as the one shown in Figure 7, the object boundary is inside another boundary and all paths that reach the image's border pass through a same pixel of the external boundary. In this case, the external boundary can be detected in a first iteration and, by using it in the place of the image's border, we can detect the desired boundary in a second iteration of the method. Figures 8c and 8d illustrate this approach for two other objects, where dots 1 and 2 indicate the corresponding leaking pixels of each iteration.

## 4 Algorithms

Algorithm 1 outputs an optimum-path forest suitable for tree pruning. It is an instance of the IFT algorithm with FIFO tie-breaking policy [4] for path-cost function $f_{\max}$ (Equation 1). For simplicity, we will consider $\hat{I} = (D_I, I)$ as a gradient intensity image and $\delta(p,q) = I(q)$, $\forall (p,q) \in \mathcal{A}$, in Equation 1. The adjacency relation $\mathcal{A}$ must be a connected relation (e.g., 4/8-adjacency in 2D and 6/27-adjacency in 3D).

**Algorithm 1** – Optimum-Path Forest Computation

Input: Image $\hat{I} = (D_I, I)$, Adjacency relation $\mathcal{A}$, seed set $S$.
Output: Optimum-path forest $P$.
Auxiliary: Cost map $C$ and priority queue $Q$.

1. *For all* $p \in D_I$, *set* $P(p) \leftarrow nil$, $C(p) \leftarrow +\infty$.
2. *For all* $p \in S$, *set* $C(p) \leftarrow 0$ *and insert* $p$ *in* $Q$.
3. *While* $Q$ *is not empty, do*
4. *Remove from* $Q$ *a pixel* $p$ *such that* $C(p)$ *is minimum.*
5. *For each* $q$ *such that* $(p, q) \in \mathcal{A}$ *and* $C(q) > C(p)$, *do*
6. *Compute* $C' \leftarrow f_{\max}(\pi \cdot \langle p, q \rangle) = \max\{C(p), I(q)\}$.
7. *If* $C' < C(q)$, *then*
8. *If* $C(q) \neq +\infty$, *remove* $q$ *from* $Q$.
9. *Set* $P(q) \leftarrow p$, $C(q) \leftarrow C'$, *and insert* $q$ *in* $Q$.

Algorithm 1 can run in linear time, if $Q$ is implemented as suggested in [4], and we could also eliminate line 8 for this particular case of $f_{\max}$ [14], but line 8 is necessary in the general case that uses $\delta(p, q)$.

Given an optimum path forest $P$, the descendant count $D$ for every pixel can be efficiently computed with a reverse breadth-first traversal of $P$ as follows.

**Algorithm 2** – Descendant Count Computation

Input: Optimum-path forest $P$.
Output: Descendant count map $D$.
Auxiliary: FIFO queue $F$, Stack $G$.

1. *For all* $p \in D_I$, *do*
2. *Set* $D(p) \leftarrow 0$.
3. *If* $P(p) = nil$ *then insert* $p$ *in* $F$.
4. *While* $F$ *is not empty, do*
5. *Remove* $p$ *from* $F$, *push* $p$ *in* $G$.
6. *For every* $q$ *such that* $(p, q) \in \mathcal{A}$ *and* $P(q) = p$, *insert* $q$ *in* $F$.
7. *While* $G$ *is not empty, do*
8. *Pop* $p$ *from* $G$.
9. *If* $P(p) \neq nil$ *then set* $D(P(p)) \leftarrow D(P(p)) + D(p) + 1$.

Note that, $P$ is defined in $D_I$ and $\mathcal{A}$ must be the same relation used in Algorithm 1. Lines 1–3 insert the roots of $P$ in $F$, lines 4–6 perform the breadth-first traversal of $P$, storing the visited pixels in $G$, and lines 7–9 perform the reverse breadth-first traversal of $P$, computing the descendant count in $D$. Since every node adds one to the descendant count of its predecessor, before the predecessor is visited, the result is correct for every node.

The next algorithm uses the second derivative of the descendant count $D$ to find the most prominent zero-crossing transition for each leaking segment of $P$ identified by a threshold $T$.

**Algorithm 3** – MAXIMUM ZERO-CROSSING DETECTION

INPUT: Optimum-path forest $P$, Descendant count map $D$, and Threshold $T$.
OUTPUT: Set $L$ of detected points.
AUXILIARY: FIFO queue $F$, set $G$ of pixels, second derivative map $D''$.

1. *Set* $G \leftarrow \emptyset$.
2. *Compute the second derivative of* $D$ *and store it in* $D''$.
3. *For all pixels* $p \in D_I$, *if* $P(p) = nil$ *then insert* $p$ *in* $F$.
4. *While* $F$ *is not empty, do*
5. *Remove* $p$ *from* $F$.
6. *For each pixel* $q$ *such that* $(p, q) \in \mathcal{A}$ *and* $P(q) = p$, *do*
7. *If* $D(q) \geq T$ *Then insert* $q$ *in* $F$.
8. *Else, Then* $G \leftarrow G \cup \{p\}$.
9. *For each* $p \in G$, *do*
10. *Set* $q \leftarrow p$, *set* $d_{max} \leftarrow -\infty$, *set* $r \leftarrow nil$.
11. *While* $P(q) \neq nil$, *do*
12. *If* $sgn(D''(q)) \neq sgn(D''(P(q)))$, *Then*
13. *Set* $d \leftarrow D''(P(q)) - D''(q)$.
14. *If* $d > d_{max}$ *Then set* $d_{max} \leftarrow d$ *and* $r \leftarrow P(q)$.
15. *Set* $q \leftarrow P(q)$.
16. *If* $r \neq nil$ *Then* $L \leftarrow L \cup \{r\}$.

Line 3 inserts the roots of $P$ in $F$. Lines 4–8 transverse $P$ in breadth-first order and determine the end pixels $p$ of each leaking segment, identified by pixels $q$ with $D(q) \geq T$. The end pixels are placed in $G$. Lines 9–16 traverse the leaking segments from their end pixels to their roots, selecting the most prominent zero-crossing transition $r$ in $D''$ and inserting $r$ in $L$.

The algorithms for the second approach are presented next. First we need to compute for every pixel its number of descendants in the image's border. Algorithm 4 is a straightforward way to obtain this information.

**Algorithm 4** – DESCENDANTS-IN-THE-BORDER COUNT COMPUTATION

INPUT: Optimum-path forest $P$.
OUTPUT: Descendants-in-the-border count map $D$.

1. *For all* $p \in D_I$, *set* $D(p) \leftarrow 0$.
2. *For each pixel* $p \in D_I$ *such that* $p$ *belongs to the image's border, do*
3. *Set* $q \leftarrow p$.
4. *While* $P(q) \neq nil$, *do*
5. *Set* $D(P(q)) \leftarrow D(P(q)) + 1$, *and set* $q \leftarrow P(q)$.

This algorithm makes a backward traversal of each optimum path that reaches the image's border and increments the descendant count along the path between the border pixel and its root.

The descendants-in-the-border count map $D$ can be used to detect leaking pixels as follows.

**Algorithm 5** – MAXIMAL COUNT DETECTION

INPUT: Optimum-path forest $P$ and descendants-in-the-border count map $D$.
OUTPUT: Set $L$ of leaking points.

1. *For each pixel* $p \in D_I$ *such that* $p$ *belongs to the image's border, do*
2. *Set* $q \leftarrow p$, *set* $d_{max} \leftarrow -\infty$.
3. *While* $P(q) \neq nil$, *do*
4. *If* $D(q) > d_{max}$ *Then set* $d_{max} \leftarrow D(q)$ *and set* $r \leftarrow q$.
5. *Set* $q \leftarrow P(q)$.
6. *Set* $L \leftarrow L \cup \{r\}$.

For each pixel $p$ in the image's border, a leaking pixel $r$ is localized at the first maximum value $d_{\max}$ of $D$ in the way back from $p$ to its root in $P$ (see Figure 8a). Considering that a same leaking pixel may be detected in more than one path and the sum of their descendant count is the size of the border, Algorithm 5 can stop earlier when the later observation is detected.

**Variant of maximum gradient (dissimilarity):** As discussed in Section 3.1 and illustrated in Figures 6e-f, objects close to each other may cause false leaking pixels in the background. A possible solution assumes that, if the dissimilarity condition is satisfied (Section 3), then the right location of a leaking point is at the maximum gradient (dissimilarity) value in the segment between the detected pixel and its root. We call this *variant of maximum gradient (dissimilarity)*. It can be implemented for both approaches by substituting $L \leftarrow L \cup \{r\}$ for $L \leftarrow L \cup \{\text{FINDMAXGRAD}(\hat{I}, P, r)\}$, in line 16 of Algorithm 3 and line 6 of Algorithm 5. FINDMAXGRAD is given below as Algorithm 6. This variant does not affect the result when the leaking pixels are already detected on the object's border.

**Algorithm 6** – FINDMAXGRAD

INPUT: Gradient image $\hat{I}$, Optimum-path forest $P$, starting pixel $r$.
OUTPUT: Maximum gradient pixel $r_{max}$.

1. *Set* $q \leftarrow r$, *set* $I_{max} \leftarrow I(q)$.
2. *While* $P(q) \neq nil$, *do*
3. *If* $I(q) > I_{max}$ *Then*
4. *Set* $r \leftarrow q$, *set* $I_{max} \leftarrow I(q)$.
5. *Set* $q \leftarrow P(q)$.
6. *Set* $r_{max} \leftarrow r$.

## 5 Validation

We have evaluated the effectiveness of tree pruning for automatic object detection in three situations: (i) labeling of multiple objects with similar textures, (ii) 3D object definition, and (iii) shape-based object detection.

In (i), similar textures facilitate automatic seed selection and gradient estimation, but nearby objects may cause false leaking pixels outside them (Section 3.1) and touching objects have to be separated from each other. We have chosen a particular application for this situation, where fragments of archaeological pieces must be detected to resemble the original pieces [15].

In (ii), the idea is to verify the straightforward 3D extension of the method. We have chosen 3D brain segmentation from Magnetic Resonance (MR) images to evaluate this situation. In this case, the relative size of the brain is exploited for automatic seed selection and the most difficult task is gradient estimation.

More generally, by changing seed location in the image, one can create various candidate objects with tree pruning and exploit some global information to select the right one. Particularly, we strongly believe on this approach to combine local and global features for object detection. In (iii), we have chosen the detection of license plates as application due to their rectangular shape. Note that, we do not intend to propose the best solution for each of these applications. Our aim is to show that some preliminary and simple solutions already lead to good results.

Consider the following implementations of tree pruning (Section 4):

V1. Tree pruning with Algorithms 1, 2, and 3.

V2. Tree pruning with Algorithms 1, 4, and 5.

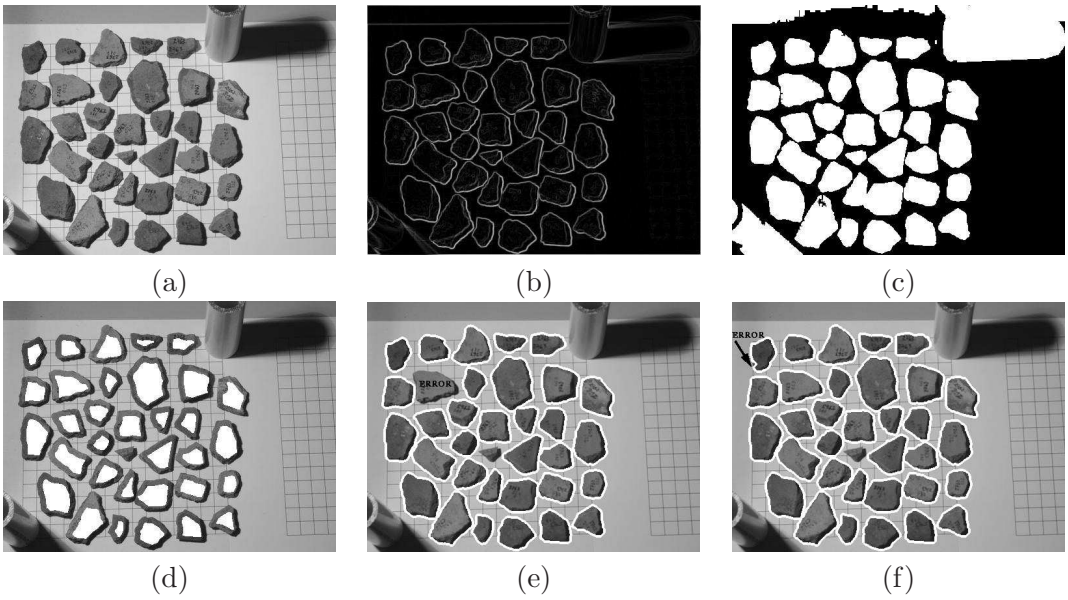
In both cases, we have included the variant of maximum gradient (Algorithm 6).

Some 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 object pixels.

## 5.1 Labeling of fragments

Figure 9a shows an image where each fragment is one object of interest. The strategy is to find one connected seed set for each fragment, label each set with a distinct number, and execute tree pruning for each set, separately, in order to propagate their labels. In the case of touching fragments, but only for those which have one seed set each, tree pruning outputs their union in both executions. This condition is detected and the fragments are separated afterward by executing a watershed transform restricted to their union [14]. Note that, if we try to segment the fragments using watershed transform and the skeleton by influence zones [16] of the seeds as background marker, the result will not be correct for touching fragments.

The grid pattern in the background of Figure 9a can be eliminated by morphological closing with a disk of radius 1. We compute the Sobel's gradient of the filtered image and use its magnitude at every pixel $q$ as dissimilarity function $\delta(p, q)$ in Equation 1 (Figure 9b).




(a) (b) (c)

(d) (e) (f)

Figure 9: Labeling of archaeological fragments: (a) Original image. (b) Gradient image. (c) Binary image after thresholding. (d) Original image with internal seeds. (e-f) Segmentation results with V1 and V2, respectively. One seeded fragment was lost with V1 and one seeded fragment was incorrectly detected with V2.

### 5.1.1 Seed selection

The image shown in Figure 9c is obtained by thresholding the filtered image for values above its mean intensity and by closing the holes inside the fragments that appear due to the thresholding. This binary image presents undesired components connected to the image's border and fragments that touch each other. A morphological erosion with a disk of radius 10 is applied to separate the touching fragments, and then we eliminate the remaining components that touch the image's border. In order to avoid multiple seed sets inside a same fragment, we finally apply a morphological dilation with a disk of radius $\sqrt{2}$ (Figure 9d).

### 5.1.2 Experiments

We used V1 (Figure 9e), with $T = 10\%$ of the image pixels, and V2 (Figure 9f) to segment 6 images with a total of 211 fragments. We counted any visible failure on an object boundary as a segmentation error. Since we are using an ad-hoc procedure for seed selection, this process may miss seeds inside fragments connected to the image's border and small fragments, and may not result in one seed set for each touching fragment. In this sense, a failure in the seed selection process should not be considered a failure of V1 or V2.

Among the 211 fragments, we correctly found seed sets for 201 fragments (95.25%). Among the 201 fragments with seed sets, V1 missed 5 fragments (2.49%), but correctly detected 196 (97.51%), and V2 detected all 201 fragments (100%), but failed the boundary of 7 (3.48%).

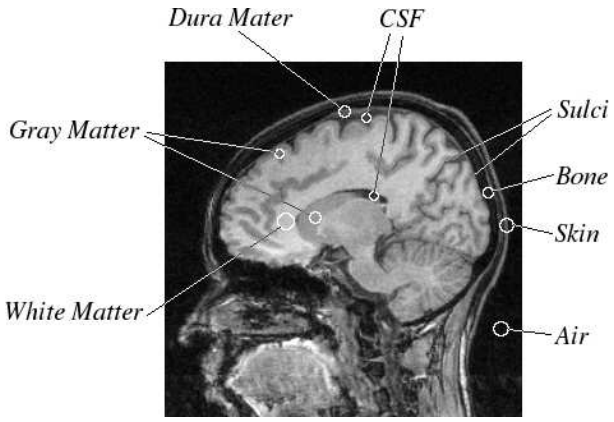



Figure 10: Tissues in MR-T1 images of the brain. CSF is the cerebrospinal fluid (liquor).

## 5.2 3D MR-brain segmentation

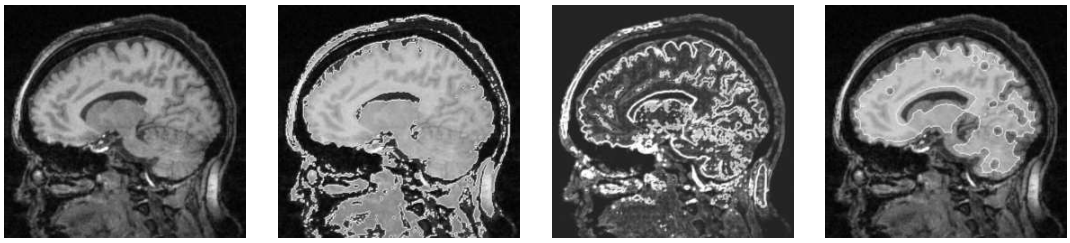
Segmentation of the human brain from Magnetic Resonance (MR) images is an open problem, which has been addressed in several different ways, each one with its pros and cons[17]. Figure 10 shows an MR-T1 slice image of the brain, where some structures are indicated. We wish to separate the gray matter (GM) and white matter (WM), as a single object, from the rest of the image. We have evaluated V1 and V2 using phantom MR-T1 images (which are available at the BrainWeb site[1] [18] together with their ground truth) and real MR-T1 images from control subjects. We have also compared the results of V1 (with $T = 2\%$ of the number of voxels) and V2 to those obtained by the template-based algorithm provided in SPM2[2] [19]— a software widely used in neuroscience research for brain segmentation and analysis [20, 21].

### 5.2.1 Gradient estimation

In the case of phantom images, a 3D extension of the Sobel's gradient is enough to provide good segmentation results. On the other hand, real images present other artifacts which require the enhancement of the border between GM and CSF (Figure 10) in order to satisfy the gradient condition (Section 3).

MR-T1 images of the brain contains two large clusters: the first, with air, bone and CSF, is represented by darker voxels, and the second, represented by brighter voxels, consists of GM and WM together with other structures (e.g., dura matter, spine, skin, and muscles). We observed that the Otsu's optimal threshold [22] can separate these two clusters, such that the GM/CSF border becomes part of the border between these clusters (Figures 11a and 11b). In order to enhance the GM/CSF border, we multiply the original image $\hat{I} =$

[1]URL: http://www.bic.mni.mcgill.ca/brainweb/
[2]URL: http://www.fil.ion.ucl.ac.uk/spm/software/spm2/



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

Figure 11: (a) Sample slice of MR-T1 image of the brain. (b) Binary image resulting from Otsu's thresholding overlaid on image (a). (c) Gradient image. (d) Binary image of seed voxels overlaid on image (a).

$(D_I, I)$ by a weight $w(p)$, computed for each voxel $p \in D_I$ as follows:

$$w(p) = \begin{cases} 0 & I(p) \leq m_1 \\ 2\left(\frac{I(p)-m_1}{m_2-m_1}\right)^2 & m_1 < I(p) \leq \tau \\ 1-2\left(\frac{I(p)-m_2}{m_2-m_1}\right)^2 & \tau < I(p) \leq m_2 \\ 2 & I(p) > m_2 \end{cases} \quad (2)$$

where $\tau$ is the Otsu's threshold, $m_1$ is the mean intensity in $I$ considering only values below $\tau$, and $m_2$ is the mean intensity in $I$ considering only values above $\tau$. Finally, we compute a 3D morphological gradient with 6-connected adjacency (Figure 11c).

### 5.2.2 Seed selection

Note that, GM and WM can not be obtained by simple morphological operations on Figure 11b. On the other hand, a morphological erosion 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 11d).

### 5.2.3 Experiments

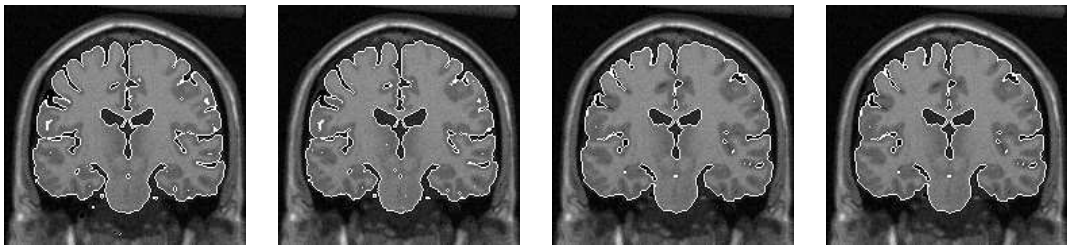
On the first set of experiments, we generated 8 MR-T1 phantoms with varying noise and inhomogeneity (INU) settings (Table 1). We segmented these phantoms with V1, V2, and SPM2 . Since tree pruning detects the external boundary of the brain, the resulting objects still contain small CSF regions (sulci and ventricles). To properly classify them as background, we take the intersection between the pruned object and the voxels with intensities above $\tau$ in V1 and V2.

Table 1 shows the segmentation errors for each method as compared to the ground truth. Note that, the methods produce similar results being SPM2 slightly more accurate (see also Figures 12 and 13). On the other hand, the same does not hold for real images.

On the second set of experiments, we selected 12 MR-T1 images of control subjects with no known anomalies, acquired with a 2T Elscint scanner and voxel size of $0.98 \times 0.98 \times 1.00$

| Parameters | | Tree Pruning (V1) | | Tree Pruning (V2) | | SPM2 | |
|---|---|---|---|---|---|---|---|
| Noise (%) | INU (%) | FP (%) | FN (%) | FP (%) | FN (%) | FP (%) | FN (%) |
| 3 | 20 | 3.63 | 1.81 | 3.57 | 1.82 | 1.63 | 1.46 |
| 5 | 20 | 3.63 | 1.79 | 3.52 | 1.80 | 1.60 | 1.43 |
| 7 | 20 | 3.22 | 2.13 | 3.10 | 2.18 | 2.28 | 1.66 |
| 9 | 20 | 3.06 | 2.97 | 2.99 | 3.03 | 2.83 | 2.30 |
| 3 | 40 | 3.78 | 1.63 | 3.61 | 1.86 | 0.90 | 1.64 |
| 5 | 40 | 3.41 | 2.07 | 3.34 | 2.14 | 1.61 | 1.40 |
| 7 | 40 | 3.29 | 2.10 | 3.01 | 2.33 | 2.31 | 1.57 |
| 9 | 40 | 2.76 | 3.10 | 2.71 | 3.11 | 2.78 | 2.29 |

Table 1: Brain phantom segmentation errors with tree pruning and SPM2. False positives (FP) and False negatives (NP) are shown as a percentage of the volume of GM and WM obtained from the ground truth mask.



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

Figure 12: Sample slice from the brain phantom segmentation experiment: (a) Phantom ground truth and the segmentations obtained with (b) SPM2, and tree pruning with (c) V1 and (d) V2.

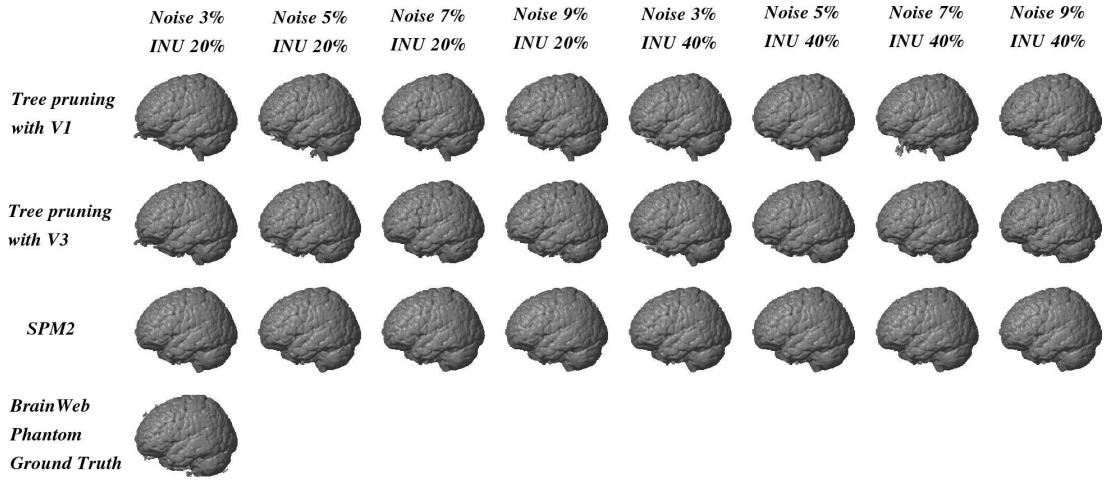



Figure 13: Renditions of the brain phantom segmentations for all 8 instances of the BrainWeb phantom. The ground truth is the same for all datasets.

$mm^3$. The images were linearly interpolated to voxel size $0.80 \times 0.80 \times 0.80$ $mm^3$. The segmentation results for 6 out of the 12 images are shown in Figure 14. Perhaps due to the quality of our images, SPM2 detected structures out of the brain in all 12 cases while V1 and V2 provided similar good results for all images. In both cases, phantoms and real images, we set as object voxels in SPM2 those with probability higher than 50% to be GM or WM. Some structures out of the brain can be eliminated with higher thresholds, but SPM2 misses some GM voxels in these cases. On the other hand, SPM2 eliminates the spine and a little more of sulci than V1 and V2.

Tree pruning not only provided better segmentation results than SPM2, but was also much faster. The entire segmentation task (marker extraction, gradient computation, optimum-path forest computation, leaking point detection and tree pruning) took 48 seconds for a $312 \times 312 \times 192$ MR-T1 volume, while SPM2 took 7 minutes, on the same Athlon64 3200+ workstation.

## 5.3 Detection of license plates

We wish to find the precise location and spatial extent of license plates using tree pruning in images, such as the one shown in Figure 15a. Our database consists of 990 images of $352 \times 240$ pixels. The magnitude of the Sobel's gradient is used as dissimilarity function (Figure 15b) and 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 as well. Therefore, a natural strategy is to run tree pruning for each set of seeds, 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 slightly deformed rectangles due to the relative positions between car and camera.

In this application, we have observed that V2 is consistently better than V1 for any fixed threshold $T$. Therefore, we decided to run the experiments with V2 only.

### 5.3.1 Seed selection

Seed selection aims at isolating some pixels of the plate's number. Since they are dark pixels, Figure 15c shows a binary image of those pixels in Figure 15a with value below 30% of its maximum intensity. If none plate is detected at this threshold, the algorithm reduces the threshold to 15% of the maximum intensity and repeats the segmentation process. In order to isolate seed pixels inside the plate, we apply a morphological erosion with a disk of radius 5.0, followed by a morphological dilation with a disk of radius 7.0 (Figure 15d), and consider the components in Figure 15c which do not belong to Figure 15d. 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 15e). This last dilation was used to reduce the number of seed sets (connected components).

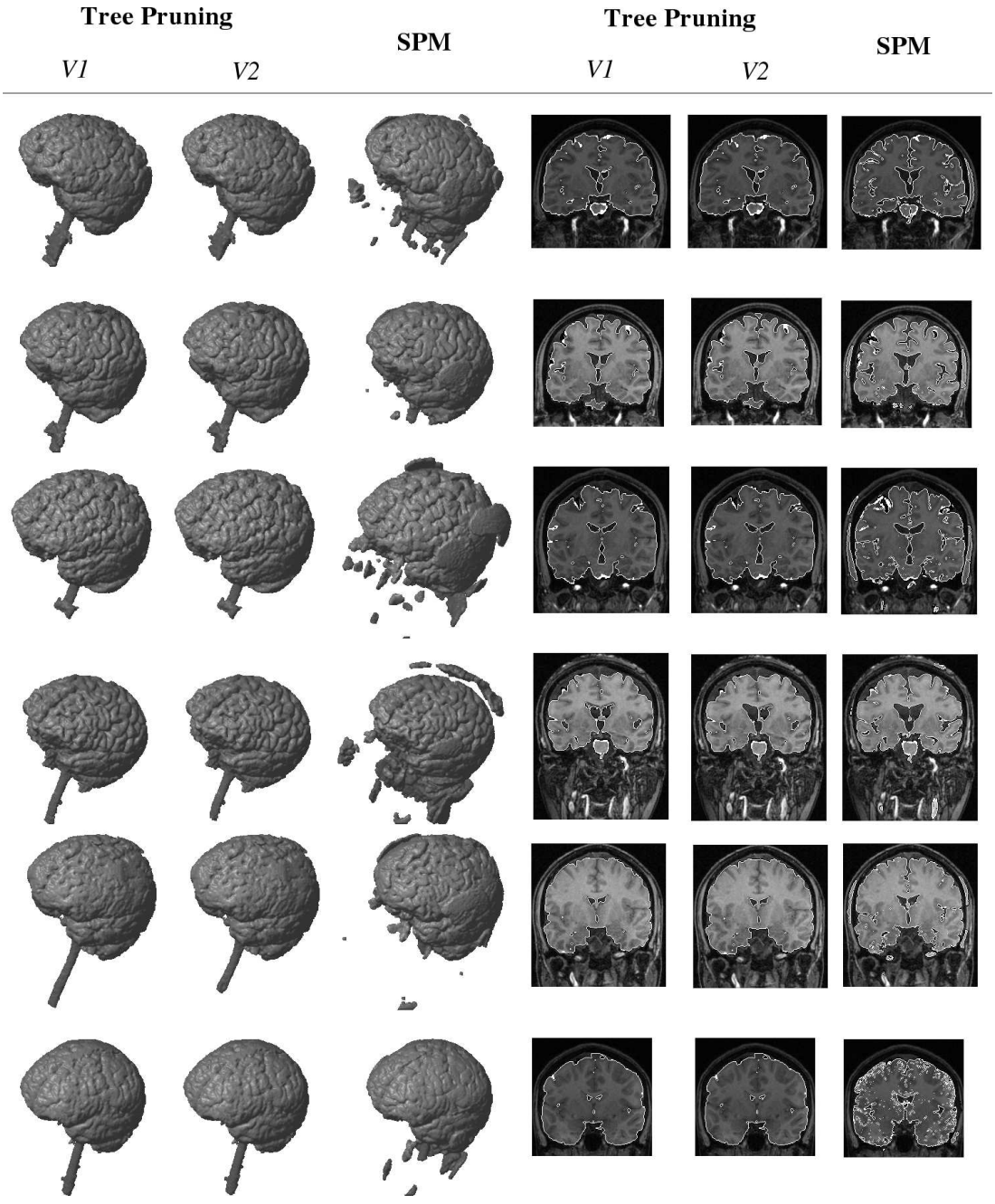



Figure 14: Brain segmentation results with tree pruning and SPM2 for 6 of the 12 control subjects, shown as 3D renditions and sample coronal slices.

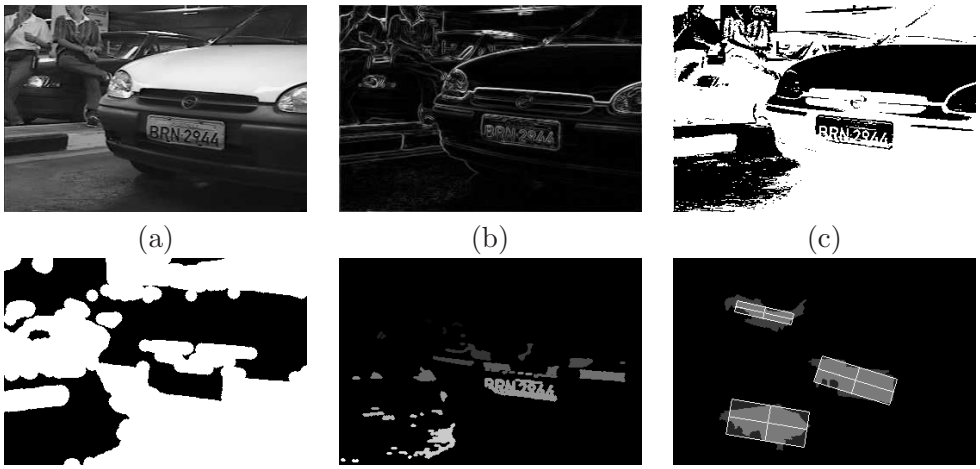


Figure 15: (a) Original image. (b) Gradient image. (c) Image (a) after thresholding. (d) Image (c) after erosion and dilation. (e) Image of labeled seeds. (f) The best rectangles for three false candidate objects.

### 5.3.2 Object score

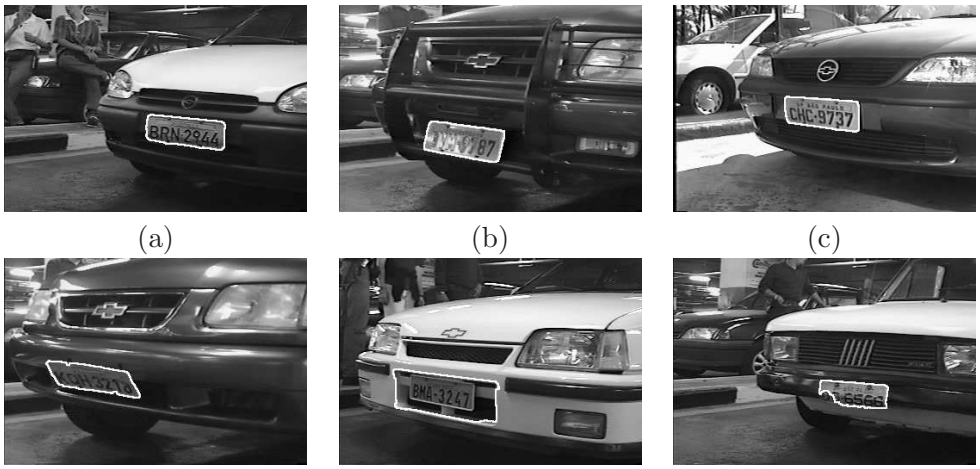
Tree pruning with V2 is applied to each seed set with a distinct label and the resulting object receives a score proportional to its similarity to a rectangular shape. At the end, the object with the best score is selected as a license plate. This process is restricted to an interest region of $200 \times 100$ pixels around each seed set. We also reject objects with less than 1200 pixels or more than 8200 pixels, since the size of the plates varies from 1559 up to 7753 pixels.

First, we find the best fit between each candidate object and a rectangle. To guarantee rotation invariance, we determine the object's major semi-axis by principal component analysis [23]. Then 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 15f shows three of these rectangles for false candidates.

Considering each rectangle as a pseudo "ground truth", we compute a score $SC$ for the corresponding object based on false positives $FP$, false negatives $FN$, and the aspect ratio $R$ of the rectangle, as follows.

$$\begin{aligned} SC &= (1.0 - \min\{1.0, FP + FN\}) * W \\ W &= \begin{cases} 1.0 & \text{if } 2.4 \leq R \leq 3.0, \\ \exp\left(-(2.4 - R)^2/3.0\right) & \text{if } R < 2.4, \\ \exp\left(-(3.0 - R)^2/3.0\right) & \text{if } R > 3.0. \end{cases} \end{aligned} \tag{3}$$

where the interval [2.4.3.0] was chosen in between the minimum and maximum expected aspect ratios of the plates.




(a) (b) (c)

(d) (e) (f)

Figure 16: (a)-(d) Correct detections with scores 0.937, 0.913, 0.902, and 0.767, respectively. (e)-(f) Wrong detections with scores 0.917 and 0.801, respectively.

### 5.3.3 Experiments

The method correctly detected 931 (94.04%) license plates out of 990. Some examples of correct and wrong segmentations are shown in Figure 16. Among the 59 plates not detected, the method failed the object score process for 28 plates, and tree pruning failed the remaining 31 plates (3.13% of the database).

According to a recent publication [24], where the authors use a database with 784 images of $384 \times 288$ pixels, our results seems to be good. The images in this database have slightly higher resolution and the license plates seem to appear as perfect rectangles. Zheng et al. [24] have evaluated four distinct approaches and found an average rate of 88.83% of correct detections, where their method obtained 99.7% of correct detections and the minimum rate was 82.0%.

## 6 Conclusion

We described a new IFT-based image operator, called tree pruning, for object detection. It is the first time we exploit the topology of the optimum-path forest in an IFT-based operator. Tree pruning essentially reduces object detection to the choice of internal seeds and leaking pixels on the object's boundary. We presented interactive and automatic solutions for tree pruning with several examples, discussed its pros and cons, and evaluated its performance in three situations involving automatic object detection: labeling of multiple objects (fragments of archaeological pieces), 3D MR-brain segmentation, and detection of license plates. In all cases, we obtained good segmentation results. We showed that tree pruning can provide better results than SPM2 [19]— a popular template-based algorithm used in neuroscience research— for MR-brain image segmentation, and that its results for

license plate detection are above the average of methods used in a recent work [24]. Considering these are challenging applications, we may conclude that tree pruning is a promising approach for automatic object detection.

We are currently investigating other approaches for the detection of leaking pixels and looking for new applications. We also wish to further improve our method of license plate detection and compare tree pruning with SPM2 using a larger number of images.

## Acknowledgments

The authors thank Roberto Lotufo (FEEC-UNICAMP) for the images of license plates, Helena Leitao (IC-UFF-Brazil) for the images of fragments, and Fernando Cendes (FCM-UNICAMP) for the MR-brain images. This work was supported by CNPq (Proc. 302427/04-0), CAPES and FAPESP (Proc. 03/09793-1 and Proc. 03/13424-1).

## References

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

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

[10] G.T. Herman and B.M. Carvalho. Multiseeded segmentation using fuzzy connectedness. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(5):460–474, 2001.

[11] 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.

[12] 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.

[13] A.X. Falcão, F.P.G. Bergo, and P.A.V. Miranda. Image segmentation by tree pruning. In *Proc. of the XVII Brazillian Symposium on Computer Graphics and Image Processing*, pages 65–71. IEEE, Oct 2004.

[14] 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.

[15] H.C.G. Leitao and J. Stolfi. A multiscale method for the reassembly of two-dimensional fragmented objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(9):1239–1251, September 2002.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] J. Ashburner and K.J. Friston. Voxel-based morphometry – the methods. *NeuroImage*, 11:805–821, 2000.

[21] M. Maddah, K.H. Zou, W.M. Wells, R. Kikinis, and S.K. Warfield. Automatic optimization of segmentation algorithms through simultaneous truth and performance level estimation (STAPLE). In *MICCAI (LNCS 3216)*, pages 274–282. Springer-Verlag, Oct 2004.

[22] N. Otsu. A threshold selection method from gray level histograms. *IEEE Trans. Systems, Man and Cybernetics*, 9:62–66, March 1979.

[23] E. L. Hall. *Computer Image Processing and Recognition*. New York, NY: Academic Press, 1979.

[24] 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.