# Tree-Pruning: A New Algorithm and Its Comparative Analysis with the Watershed Transform for Automatic Image Segmentation

Paulo A.V. Miranda
LIV – Institute of Computing – UNICAMP
C.P. 6176, 13084-971, Campinas, SP, Brazil
paulo.miranda@ic.unicamp.br

Felipe P.G. Bergo
LIV – Institute of Computing – UNICAMP
C.P. 6176, 13084-971, Campinas, SP, Brazil
felipe.bergo@ic.unicamp.br

Leonardo M. Rocha
DECOM – FEEC - UNICAMP
C.P. 6101, 13083-970, Campinas, SP, Brazil
leorocha@decom.fee.unicamp.br

Alexandre X. Falcão
LIV – Institute of Computing – UNICAMP
C.P. 6176, 13084-971, Campinas, SP, Brazil
afalcao@ic.unicamp.br

## Abstract

*Image segmentation using tree pruning (TP) and watershed (WS) has been presented in the framework of the image forest transform (IFT)— a method to reduce image processing problems related to connectivity into an optimum-path forest problem in a graph. Given that both algorithms use the IFT with similar parameters, they usually produce similar segmentation results. However, they rely on different properties of the IFT which make TP more robust than WS for automatic segmentation tasks. We propose and demonstrate an important improvement in the TP algorithm, clarify the differences between TP and WS, and provide their comparative analysis from the theoretical and practical points of view. The experiments involve automatic segmentation of license plates in a database with 990 images.*

## 1 Introduction

The *image forest transform* (IFT) has been proposed as a general tool for the design of image processing operators based on connectivity [7]. Image segmentation is one of its most important applications, where it covers both boundary- [8, 4] and region-based paradigms [14, 1, 12, 13]. In the latter, the IFT provides several algorithms for watershed segmentation [10, 11], including the differential IFT [5] which allows to compute a sequence of watershed transforms (segmentation corrections) in sub-linear time.

The IFT interprets an image as a graph whose nodes are the pixels and whose arcs are defined by an *adjacency relation* between pixels. The cost of a path in the graph is given by a suitable *path-cost* function. Given *seeds* inside and outside an object, a watershed transform by IFT [10, 7], assigns a minimum-cost path from the seed set to each pixel such that each seed becomes root of an optimum-path tree composed by its most strongly connected pixels. The resulting optimum-path forest consists of two sub-forests with distinct labels: object and background. In this case, the object is separated from the background by competition among internal and external seeds. A failure ("leaking") occurs when an object seed conquers background pixels or vice-versa.

More recently, tree pruning was proposed as a potential solution for the leaking problem [6]. The main idea is to use only internal seeds and let the leaking occurs at a first moment by computing an IFT with the same adjacency relation and path-cost function of a watershed transform. In this case, the competition among internal seeds make object and background connected by a few optimum paths (*leaking paths*) which cross the object's boundary through its most weakly connected parts (*leaking pixels*). The identification of the leaking pixels and subsequent pruning of their subtrees define the object as the remaining optimum-path forest.

In [6], the authors discuss two approaches for leaking pixel detection. One is interactive where the user can visually identify leaking pixels and select them with the mouse pointer. This visual identification is, unfortunately, a drawback in 3D segmentation. The other is automatic, but it relies on an *ad-hoc* parameter that is difficult to be adjusted in real applications.

In this paper we propose a solution that exploits the topology of the forest for automatic detection of leaking pixels. Our algorithm is free of *ad-hoc* parameters and aims at reducing image segmentation to the selection of a few seeds inside the object. These aspects favor automatic solu-

tions for segmentation when the object's location can be detected by some preprocessing (e.g., by statistical approaches such as the one proposed in [15]).

The watershed transform (WS) may be sensitive to the location of the external seeds in heterogeneous background. On the other hand, if we only consider internal seeds and all pixels in the image's border as external seeds, WS and tree pruning (TP) usually produce similar results. This motivates a comparison between both approaches for automatic segmentation. We also discuss interactive segmentation using these approaches, and evaluate them and the previous TP algorithm [6] for automatic segmentation of license plates, using a database with 990 images.

It is important to note that the whole segmentation process involves two tightly coupled tasks: *delineation* and *recognition* [8]. Delineation aims at defining 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 and WS are likely to outperform higher level approaches [9, 3, 2, 15] 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.

Section 2 reviews the IFT for region-based image segmentation. The watershed transform and the new version of tree pruning are described in Sections 3 and 4. We provide a comparative analysis between them in Section 5, and present conclusions and future work in Section 6.

## 2 Background

Tree pruning (TP) and watershed (WS) algorithms rely on a preprocessing step which estimates seeds and computes a *gradient-like image*. In this paper we use only the magnitude of the Sobel's gradient for all examples and experiments, but gradient-like images usually require some preprocessing which is application-dependent. Preprocessing is also important for automatic seed estimation, but in interactive segmentation, seeds can be simply selected by the user with the mouse's pointer. TP requires seeds inside the object and WS requires internal and external seeds.

**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 in a small neighborhood around the object's boundary.

Both approaches can be implemented for multidimensional images, and in the case of multi-parametric images (such as colored images), the parameters can be taken into account to create a gradient-like image as described above.

### 2.1 Image Foresting Transform for WS and TP

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.

In the image foresting transform (IFT), a 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 [7]. We are interested in 4- or 8-connected relations for 2D region-based image segmentation (see examples in Figures 1a and 1b).

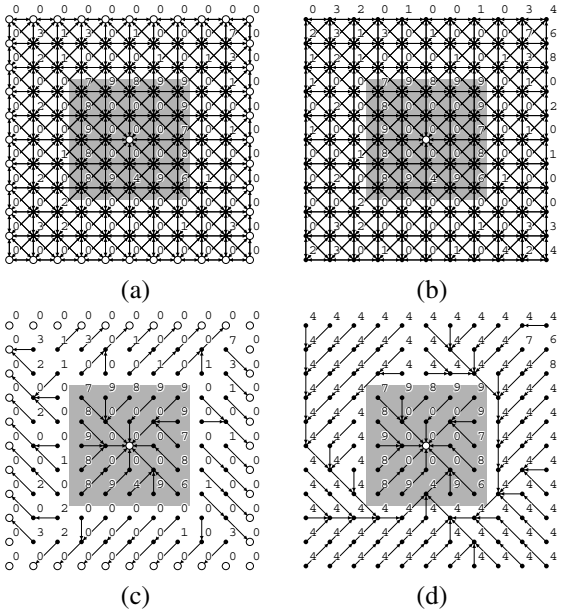



**Figure 1. (a-b) Two 8-connected image graphs where the numbers indicate the pixel values and the shaded area is the object with one internal seed (white dot). External seeds on the image's border are selected only in (a). (c-d) The respective optimum-path forests using Eq. 1.**

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* $c(\pi)$. 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 [14, 1, 10] is important in most situations and it is implemented by setting $I(p)$ to 0 for pixels $p \in S$.

The IFT assigns one optimum path 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* $P$— 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 $p \in S$ (see examples in Figures 1c and 1d).

TP exploits the topology of the forest in Figure 1d while WS separates object and background by labeling the trees of the forest in Figure 1c. 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 for TP and WS

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 discuss in Section 5, 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* $(p, q) \in A$ *and* $C(q) > C(p)$, *do*
7. *Compute* $cst \leftarrow \max\{C(p), I(q)\}$.
8. *If* $cst < C(q)$, *then*
9. *If* $C(q) \neq +\infty$, *remove* $q$ *from* $Q$.
10. *Set* $P(q) \leftarrow p$, $C(q) \leftarrow cst$, $L(q) \leftarrow L(p)$.
11. *Insert* $q$ *in* $Q$.

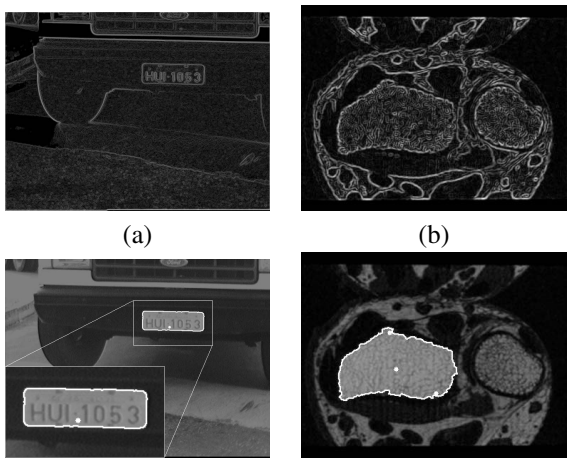
Algorithm 1 runs in linear time if $Q$ is implemented as described in [4]. Lines 1–3 initialize maps and insert seeds in $Q$. The main loop computes an optimum path from $S$ to every pixel $p$ in a non-decreasing order of cost (Lines 4–11). At each iteration, a path of minimum cost $C(p)$ is obtained 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 $L$. TP requires to identify leaking pixels in $P$ to prune all subtrees rooted in the background. We discuss these approaches next.

## 3 The watershed transform

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.

Considering the path-cost function $c$ and the gradient condition for WS (Section 2), the external seeds will reach background pixels through optimum paths whose costs are strictly lower than the costs of optimum paths rooted at internal seeds. The same is true for object pixels with respect to the internal seeds, because any path which crosses the object's boundary will have higher costs. Pixels on the object's boundary, however, may be conquered by internal or external seeds depending on the tie-breaking policy in $Q$ (i.e., we may also give preference to object seeds).




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

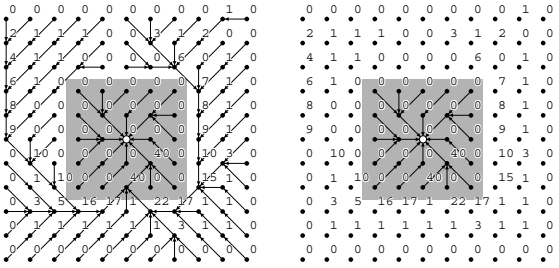
**Figure 2. (a-b) Two gradient images showing a license plate and an MR-wrist bone as interest objects. (c-d) The respective results of WS for one internal seed (white dot) and external seeds on the image's border.**

WS will work whenever the optimum paths rooted at ob-

ject and background meet each other at the object's boundary. This may happen even when the gradient condition is not fully satisfied (Figure 2). When the object contains internal boundaries (holes, components, basins), WS may require multiple internal seeds and the location of these seeds may be important (e.g., at least one internal seed should be outside the letters of the plate in Figure 2c).

## 4 Tree-pruning segmentation

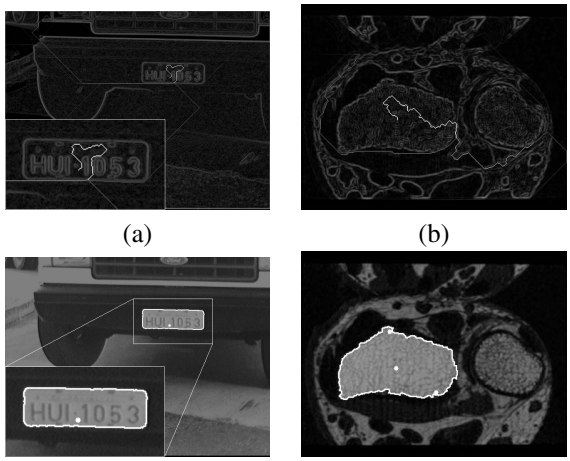
The IFT computes optimum paths in a non-decreasing order of cost. Therefore, in the gradient condition for TP, the optimum paths from $S_o$ will reach object pixels before background pixels. Moreover, if a pixel $p$ is the only one with lowest value $I(p)$ on the object's boundary, then all pixels around 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 1d). By connectivity, the rest of the background will be also conquered by leaking paths that pass through $p$. The same property can be verified when there are multiple leaking pixels, which can be automatically detected as follows.



(a) (b)

**Figure 3. (a) The numbers indicate the descendant count in $B$ for each pixel (except for the root) of the forest computed in Figure 1d. (b) After pruning, the remaining forest defines the object.**

Let $B \subset D_I$ be the image's border. All paths that reach $B$ must pass through some leaking pixel (Figure 1d). We compute the number of descendants that every node (except the roots) of the forest has in $B$ to obtain a *descendant map* $D$ (Figure 3a). This map is different from the one presented in [6]. By following backwards any optimum path in $P$ which has terminal node in $B$, the first occurrence of the *maximum descendant count* is at a leaking pixel (e.g., pixel $(6, 8)$ in Figure 3a). 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. This ramification drastically reduces the descendant count for pixels of the subtrees rooted at the leaking pixels [6]. We can then repeat the procedure for all nodes in $B$ to detect all leaking pixels, and define the object by removing their subtrees from the forest (Figure 3b).



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

**Figure 4. (a-b) The same gradient images of Figures 2a and 2b, but overlaid by the descendant map $D$ (brighter leaking paths). (c-d) The respective results of TP for the same internal seeds (white dots) shown in Figures 2c and 2d.**

Algorithm 2 computes the descendant map $D$. For each node in $B$, the algorithm traverses its optimum path backwards to its root, incrementing the descendant count along the path.

**Algorithm 2** – DESCENDANT MAP COMPUTATION

INPUT: Optimum-path forest $P$ and set $B$.
OUTPUT: Descendant map $D$.

1. *For all* $p \in D_I$, *set* $D(p) \leftarrow 0$.
2. *For each pixel* $p \in B$, *do*
3. | *Set* $q \leftarrow p$.
4. | *While* $P(q) \neq nil$, *do*
5. | └ *Set* $D(P(q)) \leftarrow D(P(q)) + 1$, *and* $q \leftarrow P(q)$.

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 maximal value in $D$ is a leaking pixel. The set $L_k$ of leaking pixels is computed by Algorithm 3.

**Algorithm 3** – 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* $p \in B$, *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* $d_{max} \leftarrow D(q)$, $r \leftarrow q$.
5. | | *Set* $q \leftarrow P(q)$.
6. | *Set* $L_k \leftarrow L_k \cup \{r\}$.

Similar results can be obtained with TP for the examples of Figure 2 (see Figure 4). Note that TP requires two main conditions: (i) the optimum paths to object pixels must not pass through the background and (ii) the optimum paths that reach the set $B$ must pass through all leaking pixels. The gradient condition (Section 2) plays an important role, but the method may work even when it is not satisfied. The comments about seed location and multiple seeds for WS are equally applied to TP.

## 5 Comparative Analysis

Consider 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 topological information (descendant map $D$) from the forest. Therefore, the role of the image's border is very different in these approaches.

The methods will produce similar results (Figures 2 and 4) whenever optimum paths from $S_b$ and $S_o$ meet each other at the leaking pixels. However, the heterogeneity of the background (Figure 5a) may create tie-zones (plateaus of cost) for WS whose costs are greater than or equal to the leaking pixel values (tie-zones appear as white pixels in Figure 5b). Depending on the tie-breaking policy, the segmentation line could be anywhere on these plateaus (Figure 5c). Since tree pruning does not depend on the costs of the optimum paths outside the object, but only on the topology of the forest, it is much less susceptible to the heterogeneity of the background (Figure 5d).

When automatic segmentation fails, the immediate alternative is interactive segmentation. The user can add internal and external seeds in WS to correct results by running subsequent IFTs in a differential way [5]. A similar procedure is possible in TP, but instead of external seeds, the user adds external pixels to the set $B$ and Algorithms 2 and 3 are re-executed to identify missed leaking pixels. A failure in leaking pixel detection is possible because seed competition among leaking paths may also prevent some path that crosses a leaking pixel to reach the set $B$. This leaking pixel can be detected by adding any pixel of the leaking region to $B$, which is simpler and more efficient to do in the 3D case than the interactive procedure described in [6]. By doing that, we are essentially extracting more information from the topology of the forest to correct segmentation. Note

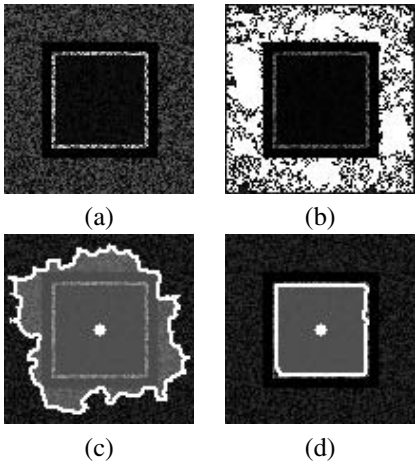



**Figure 5. (a) A synthetic gradient-like image, where the heterogeneity of the background is given by a random external noise. (b) Watershed tie-zones in white. (c-d) Respective segmentation results by WS and TP for a same internal seed (white dot).**

that, we will need to re-execute the IFT in TP only when we add more internal seeds.

In theory, one may think that TP will require less user involvement in interactive segmentation than WS. Figure 6, for example, shows that TP requires less number of interactions than WS to correct segmentation. TP also allows a variant of the Algorithm 3 to correct leaking pixel detection, whenever the leaking pixel is detected outside the object. This is possible in the case of true or false external boundaries as shown in Figure 7. Figure 7b shows the result of TP for an internal seed (white dot). Similar result is obtained with WS for a different reason. A false external boundary was detected in TP because there was a single leaking path passing through one leaking pixel of each boundary: the object's boundary and the external false boundary. In WS, the external boundary was detected because it fails the WS gradient condition. This problem with nested boundaries can be corrected in TP but not in WS. In TP, we can iteratively search backwards along the optimum path from the detected pixel to its root for the next pixel whose gradient value is maximum (Figure 7c). The only alternative in WS is to re-execute the method by taking the false external boundary as external seed set. In this case, however, it will fail because the first detected boundary and the object's boundary have a part in common (Figure 7d).

In practice, it is difficult to say which approach is better for interactive segmentation when the automatic leaking pixel detection fails. There are also examples where seed selection in WS seems to be more robust than seed and $B$-pixel selection in TP. Figure 8 illustrates an example with touching fragments where the left one is the object of inter-

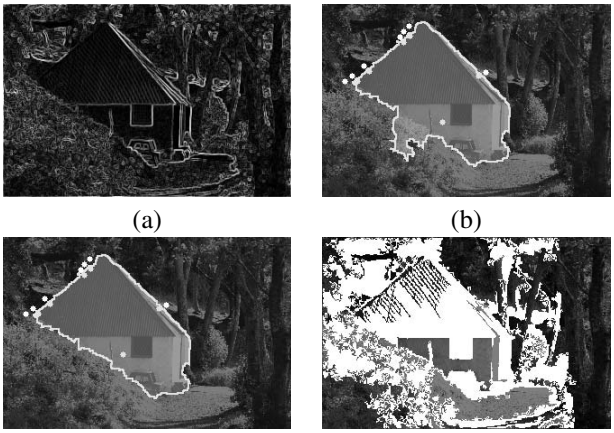


**Figure 6. (a) Gradient image where the house is the desired object. (b-c) Segmentation results with WS and TP with a same seed and $B$-pixel selection, respectively. (d) Watershed tie-zones in white.**

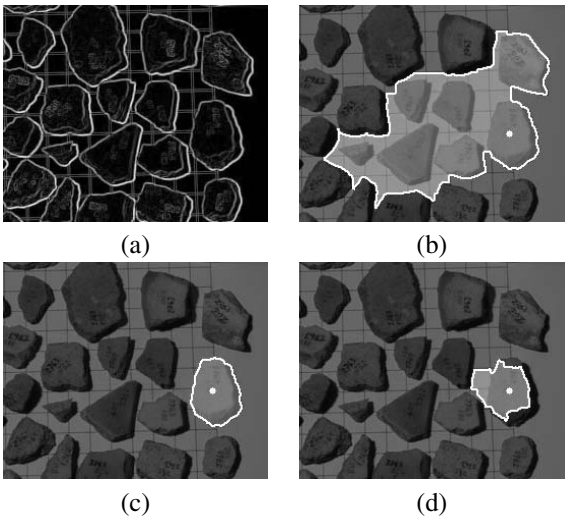


**Figure 7. (a) Gradient image of archaeological fragments. (b) A false external boundary detected by TP. Similar result is obtained in WS. (c) The result can be corrected in TP without $B$-pixel selection, (d) but the same is not possible with WS.**

est. Both methods can separate the fragments with internal and external pixel selection close to the touching part (Figures 8a and 8b). However, TP fails when we move the internal seed to the center of one fragment (Figure 8c) while WS works for any positions of the internal and external seeds (Figure 8d).

Therefore we decided to compare WS and TP in the context of automatic segmentation, where TP with automatic

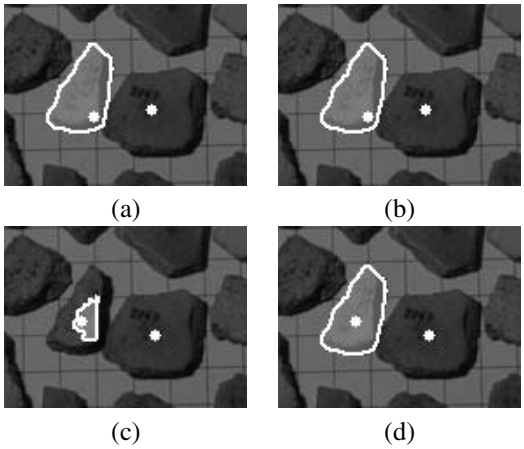


**Figure 8. Touching fragments where the left one is the desired object. (a-b) Segmentation results with TP and WS, respectively. (c) TP fails when we move the internal seed to the center. (d) WS works for any positions of the internal and external seeds.**

detection of leaking pixels is also compared with the previous version. For the experiments, we have chosen the segmentation of license plates. They essentially evaluate the sensitivity of the methods with respect to the heterogeneity of the background in a real situation where the gradient condition is not fully satisfied for WS and TP.

### 5.1 Experiments with license plates

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 9a). We have chosen this application because the ground truth for the plates is easy, due to its known shape. The magnitude of the Sobel's gradient is used (Figure 9b) 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. Therefore, a natural strategy is to run the methods for each seed set, 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.

**Seed selection.** Seed selection aims at isolating some pixels inside the plate. We have defined a common procedure for all approaches, which is based on some 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 9c). 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, fol-

lowed by a dilation with a disk of radius 7.0 (Figure 9d), and consider the components in Figure 9c which do not belong to Figure 9d. 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 9e). This last dilation was used to reduce the number of seed sets (connected components).

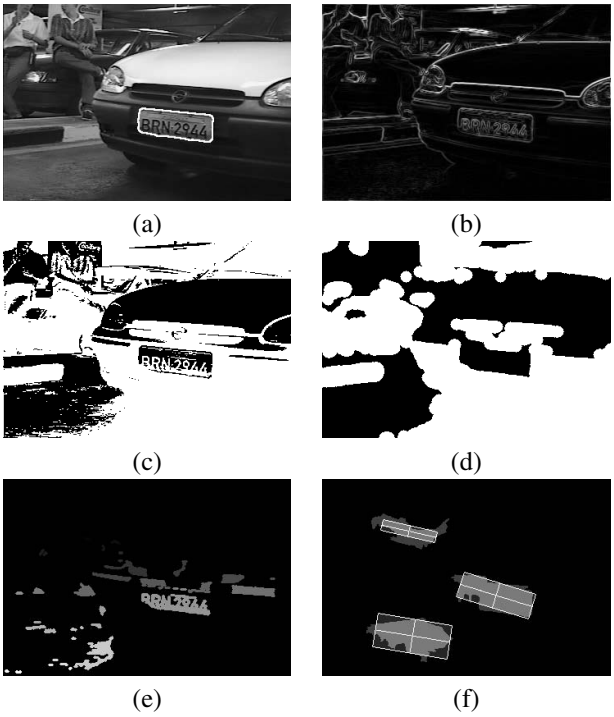



**Figure 9. (a) Original image with the result of TP. (b) Gradient image. (c) Image (a) after thresholding. (d) Image (c) after erosion and dilation. (e) Image of labeled seeds. (f) The best rectangles for three false candidate objects.**

**Object score.** During the plate's election we rejected candidate objects with less than 1200 pixels or more than 8200 pixels, since the size of the plates varies from 1559 up to 7753 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. 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 9f shows three of these rectangles for false candidates.

| | 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.**

Considering each fitted rectangle as a pseudo "ground truth", we define FN as the *percentage of false negatives*— the number of pixels that belong to the ground truth, but have been classified as background, divided by the number of ground-truth 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 ground-truth pixels. Then, we compute a score $SC$ for the corresponding object based on $FP$, $FN$, and the aspect ratio $R$ of the rectangle, as follows.

$$SC = (1.0 - \min\{1.0, FP + FN\}) \cdot W \quad (2)$$

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

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

**Results.** Table 1 summarizes the experimental results for each method, where TP denotes the new tree pruning and PTP is its previous version, implemented as described in [6] where $T$ is an *ad-hoc* parameter. TP correctly detected 923 (93.23%) license plates out of 990. In spite of the use of *ad-hoc* procedures for seed estimation, it was possible to estimate some seed set inside the plates in 100% of the cases. Among the 67 plates not detected by TP, 28 (2.8% of the database) errors were due to the score process and 39 errors (3.9% of the database) were due to the TP method. So, we can still improve this result by using a better score process, which takes into account, for example, texture information.

We can note that TP was more accurate than PTP for any fixed value $T$, because it is difficult to adjust $T$ for a given application. Table 1 also shows that WS was more sensitive to the heterogeneity of the background than TP.

A recent work [16] has achieved $99.7\%$ of accuracy in a database where the plates appear as perfect rectangles. We have tested their method in our database, but the accuracy was only $5.5\%$ using our matching criterion. On the other hand, their method could locate $98\%$ of the plates in our database among the best three candidates. This reduction in the number of candidates makes attractive a combination of their approach for seed estimation and tree pruning for segmentation.

## 6 Conclusions and future work

We presented an important improvement in tree-pruning segmentation by introducing a method for automatic leaking pixel detection, which is free of *ad-hoc* parameters. Our comparative analysis has shown that our approach is more robust and can provide automatic image segmentation of license plates with higher accuracy rate than the previous approach [6] and the watershed transform [10]. We have also provided a different solution for interactive segmentation by tree pruning, which is more suitable to the 3D case than the previous one [6].

Although *ad-hoc* procedures were used for seed estimation and object recognition, the accurate results encourage future works that combine tree pruning with statistical approaches [15, 3, 2].

## Acknowledgments

The authors thank Roberto Lotufo for the database, and CNPq (Proc. 302427/04-0), FAPESP (Proc. 05/59808-0 and Proc. 03/13424-1), and CAPES for the financial support.

## References

[1] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Intl. Journal of Computer Vision*, 1:321–331, 1987.

[10] R. Lotufo and A. 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.

[11] R. Lotufo, A. Falcão, and F. Zampirolli. IFT-watershed from gray scale marker. In *XV Brazillian Symposium on Computer Graphics and Image Processing*, pages 146–152, Oct 2002.

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

[13] L.-G. C. Shao-Yi Chien, Yu-Wen Huang. Predictive watershed: a fast watershed algorithm for video segmentation. *IEEE Trans. on Circuits and Systems for Video Technology*, 13:453–461, 2003.

[14] 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.

[15] 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.

[16] D. Zheng, Y. Zhao, and J. Wang. An efficient method of license plate location. *Pattern Recognition Letters*, 26(15):2431–2438, Nov 2005.