Tie-Zone Watershed, Bottlenecks and Segmentation Robustness Analysis

Romaric Audigier, Roberto de Alencar Lotufo
Dept. of Computer Engineering and Industrial Automation
School of Electrical and Computer Engineering – UNICAMP
C.P. 6101, 13083-852, Campinas (SP), Brazil
audigier@dca.fee.unicamp.br, www.dca.fee.unicamp.br/∼lotufo

Abstract

In a recent paper [1], a new type of watershed (WS) transform was introduced: the tie-zone watershed (TZWS). This region-based watershed transform does not depend on arbitrary implementation and provides a unique (and thereby unbiased) optimal solution. Indeed, many optimal solutions are sometimes possible when segmenting an image by WS. The TZWS assigns each pixel to a catchment basin (CB) if in all solutions it belongs to this CB. Otherwise, the pixel is said to belong to a tie-zone (TZ). An efficient algorithm computing the TZWS and based on the Image Foresting Transform (IFT) was also proposed.

In this article, we define the new concept of "bottlenecks" in the watermerging paradigm. Intuitively, the bottlenecks are the first contact points between at least two different wave fronts. They are pixels in the image where different colored waters meet and tie and from which may begin, therefore, the tie-zones. They represent the origin points or the access of the tie-zones (regions that cannot be labeled without making arbitrary choices). If they are preferentially assigned to one or another colored water according to an arbitrary processing order, as occurs in most of watershed algorithm, an entire region (its influence zone – the "bottle"!) is conquered together. The bottlenecks play therefore an important role in the bias that could be introduced by a WS implementation. It is why we show in this paper that both tie-zones and bottlenecks analysis can be associated with the robustness of a segmentation.

1 Introduction

The watershed (WS) transform is a well-known and powerful segmentation tool for morphological image processing. It was first introduced by Beucher and Lantuéjoul [2] for contour detection and applied in image segmentation by Beucher and Meyer [10]. Nowadays, there are many definitions and algorithms of watershed transforms in literature. Roerdink and Meijster [12] give a comparison of some of them. The algorithm of Vincent and Soille [13] is based on immersion simulation: the image is represented by a topography inundated by water that springs from regional minima. The watershed lines are dams constructed for separating the growing catchment basins (CB) corresponding to minima. The algorithm of Meyer [9] computes the WS transform by solving a shortest path problem with respect to a topographical distance function.

In the numerous WS algorithms, variations may first occur in the input: all regional minima; only imposed minima to avoid oversegmentation (WS from markers [10]); or grayscale markers [8] to specify the depth (handicap) of some imposed minima. Then, the output may be of different types: 'line-algorithms' return separating WS lines that are sometimes valued (as in the topological watershed [3, 11] which conserves the saliency between minima) whereas 'region-algorithms' return labeled regions (the CBs) that form a partition of the image.

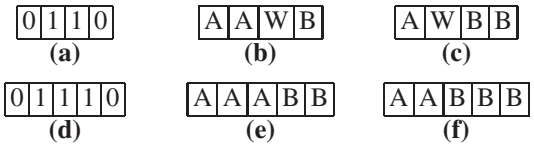


Figure 1. Original images (a)(d) and two possible labeled WS outputs (raster or anti-raster scan) of a line-algorithm (b)(c) or a region-algorithm (e)(f). W represents the WS line.

In lots of (line- or region-) algorithms, the result varies with implementations (scanning order and other arbitrary processing order) or may be inconsistent with the WS definition as observed in [12]. This variation due to implementation can be insignificant in some cases (1 pixel bias for the line/region position, see Fig. 1) but in other cases, it becomes considerable: in some images an entire region is reached passing by a *bottleneck* pixel and consequently

included to the first (or other arbitrary) CB that invades the bottleneck (like in Fig. 2(g)–(j)). Thus, the problem does not occur only on plateaux. Furthermore, it is of theoretical interest having a unique solution for the WS transform. These arguments encouraged the investigation of a WS definition that would result in a unique and consistent solution.

The Image Foresting Transform (IFT), introduced by Falcão, Lotufo and Stolfi [6] and based on Dijkstra algorithm [4], provides a sound framework for the efficient implementation of many image processing operators [5]. For instance, the WS transform is computed as a problem of trees of minimal paths.

Figure 2 shows a *buttonhole* case. It will be commented in both sections 2 and 3. Figure 2(a) presents the input image used to compute the different watersheds, forests and maps of Figures 2(b)–(j). It is important to remark that buttonhole cases "correspond to special pixel configurations which are not so rare in practice" as refered by [11, 13].

This paper is organized as follows. In section 2, an overview of the IFT framework is given to define in this context the Tie-Zone Watershed (TZWS) that results in a unique solution, regardless of implementation. Then, the efficient algorithm introduced in [1] is recalled. The bottlenecks, access of the tie-zones, are defined and characterized in section 3. Finally, as an application of these new concepts, the TZ and bottleneck analysis of real images is done in section 4 and associated to the robustness of the segmentations.

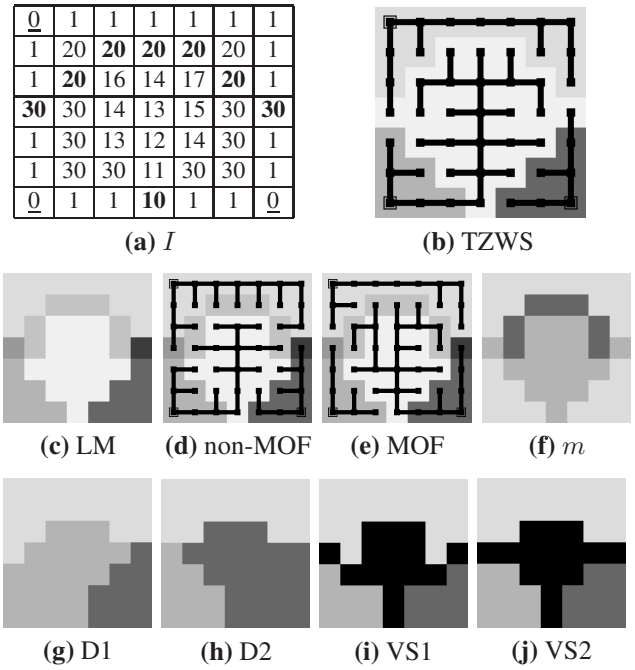


Figure 2. (a): Input grayscale image with 3 (underlined) minima and 8 (bold) bottlenecks. (b): TZWS using 4-adjacency: 3 CBs (gray), TZ (white), forest (black). (c)–(e): Result of the Label Merging algorithm obtained with either a non-MOF (d) or a MOF (e). (f): Map of multiplicity. (g)–(h): Watersheds by Dijkstra-IFT varying scanning-order. (i)–(j): Watersheds (black) by Vincent and Soille's algorithm (raster or anti-raster scan).

2 The Tie-Zone Watershed (TZWS)

2.1 Overview of the Image Foresting Transform (IFT)

Under the IFT framework, an image is seen as a weighted graph $G = (V, A, I)$ where each pixel (or voxel in 3D) is represented by a node or vertex $v \in V$ with intensity $I(v)$ (I is a map from V to $\mathbb{Z}$ for digital image). An arc $\langle u, v\rangle \in A$ exists between vertices u and v when the corresponding pixels are adjacent according to the defined adjacency (usually 4- or 8-adjacency in 2D and 6- or 26-adjacency in 3D). A path from a node u to a node v in a graph (V, A, I) is a sequence $\langle u = v_1, v_2, \ldots, v_n = v\rangle$ of nodes of V such that $\forall i = 1 \ldots n-1$, $\langle v_i, v_{i+1}\rangle \in A$. A path is said simple if all its nodes are different from each other. Let $S \subseteq V$ be a set of particular nodes s_i called seeds or markers.

For a given weighted graph $G = (V, A, I)$ and a set S of seeds, an IFT is a *directed forest* F of G, i.e. a directed acyclic subgraph[1] F of G, such that (i) there exists for each node $v \in V$ a *unique and directed simple path* $\pi(s_i, v)$ in F from a seed node $s_i \in S$ to v and (ii) each such path has a *minimum cost* among all possible paths in G linking v to any seed of S, according a specified path cost function f_C.

Assume that the arcs $\langle u, v\rangle$ are weighted with the gray-level $I[v]$ of the pixel corresponding to v. Assume that the seed nodes correspond to the regional minima of the image (or to imposed minima, i.e. markers). If the path cost function is defined as the 'maximum arc' function f_{max},

$$f_{\max}(\langle v_1, v_2, \ldots, v_n\rangle) = \max\{h(v_1), I(v_2), \ldots, I(v_n)\}$$

where h is a fixed but arbitrary *handicap cost* [8], the IFT computes a region-WS transform where each tree of the forest [2] corresponds to a CB. Note that all vertices (pixels) are covered by this forest. The IFT can result in many optimal forests because many paths of minimum cost are sometimes possible. The set of all optimal forests is denoted by $\mathbf{\Phi}$.

The optimality of the WS by IFT was proved in [7] where a two-component lexicographic cost function f_{LC}

[1] The graph $G' = (V', A')$ is a subgraph of G if $V' \subseteq V$, $A' \subseteq A$ and $A' \subseteq V' \times V'$.

[2] A tree of the forest F is a connected component of F.

was proposed to mimic the flooding process and handle with plateaux too: $f_{LC} = (f_{\max}, f_d)$. The first component, of highest priority, is the max-arc function and represents the flooding process. The second one makes different waters propagate on plateau at a same speed rate:

$$\begin{aligned} f_d(\langle v_1, v_2, \ldots, v_n\rangle) &= \max_{k\in[0,n-1]} \{k, C[v_n] = C[v_{n-k}]\} \\ C[v_n] &= f_{\max}(\langle v_1, v_2, \ldots, v_n\rangle) \end{aligned}$$

This lexicographic path cost, inspired from Meyer's topographical distance strategy [9], is very simple to compute using a priority FIFO queue, avoids a prior lower completion on image with plateaux, and provides partitions that seem to be more equitable (on plateaux) than when only the maximum cost is used.

2.2 The Tie-Zone Watershed (TZWS) transform

As we saw in the previous section, many optimal forests and so, many partitions may correspond to an input image-graph. We propose then a new definition of watershed transform in the IFT context which results in a unique partition.

A node is included in a specific catchment basin CB_i when it is linked by a path to a same seed s_i in all the optimal forests, otherwise it is included in the Tie-Zone T:

$$\begin{aligned} CB_i &= \{v \in V, \ \ \forall F \in \Phi, \ \ \exists \pi(s_i, v) \text{ in } F\} \\ T &= V \setminus \bigcup_i CB_i \end{aligned}$$

If a node is in the tie-zone, it means that it could be included in different CBs without affecting the forest optimality. CBs are only the common part of all optimal solutions whereas differing parts are considered TZ. Therefore, the tie-zone existence prevents from making any arbitrary choice between optimal solutions. Consequently, the TZWS solution is defined without ambiguity.

Note that this definition does not produce watershed lines but only regions: catchment basins and tie zone. They form together a *unique* optimal partition of the image. If all pixels are assigned to catchment basins, the tie zone will be empty. This situation can occur when the lexicographic path-cost function unties growing CBs on plateaux. So, the watershed transform possibly does not contain any tie zone.

Unlike in the WS by IFT, each CB corresponds to a tree *or* part of it, while the TZ is composed of many terminal parts of trees as in the example of Fig. 2(b).

2.3 Algorithm

In this section, we present an efficient algorithm that labels the image in order to obtain a TZWS. It is based on Dijkstra's shortest path algorithm [4] and utilizes an ordered queue Q where each bucket has a FIFO policy. Note that the second component C_2 of the lexicographic cost is *not* intrinsically computed by the FIFO policy and must be explicit in the TZWS by IFT in order to prevent 1-pixel bias.

The algorithm input is: the image as a weighted graph $G = (V, A, I)$, the seed node set S with associated labeling function λ and handicap function h. The priority queue Q is initially empty: `DequeueMin` removes from Q and returns the node of minimum cost; `Enqueue`(p, c) inserts node p in Q at priority (cost) c bucket. We denote the neighborhood of a node $p \in V$ by: $N_G(p) = \{q \in V, \langle p, q\rangle \in A\}$. Label map L corresponds to the TZWS result, map P gives each node's predecessor in the tree and maps C_1,C_2 give the lexicographic cost of an optimal path from a seed to each node.

The beginning of the algorithm (lines 1 to 11) is identical with the IFT algorithm in [6]. Lines 12 to 16 are TZWS-specific. In line 12, the second component of lexicographic cost is incremented, as water propagates on plateau. Lines 13 to 16 detect the nodes where paths from (at least) two seeds ($L[p] \neq L[v]$) tie, i.e. have same costs (C_1 and C_2).

Algorithm 1: TZWS by IFT with lexicographic path cost.

1. $\forall p \in V,\ C_2[p] \leftarrow 0$; $done(p) \leftarrow$ FALSE;
2. $\forall p \notin S,\ C_1[p] \leftarrow \infty$; $L[p] \leftarrow$ NIL; $P[p] \leftarrow$ NIL;
3. $\forall p \in S,\ C_1[p] \leftarrow h(p); L[p] \leftarrow \lambda(p);\ P[p] \leftarrow p$; `Enqueue`$(p, h(p))$;
4. **while** `QueueNotEmpty`,
5. $v \leftarrow$ `DequeueMin`; $done(v) \leftarrow$ TRUE;
6. $\forall p \in N_G(v)$ **and** $done(p) =$ FALSE,
7. $c \leftarrow \max\{C_1[v], I[p]\}$;
8. **if** $c < C_1[p]$,
9. **if** p in Q, `Dequeue`(p);
10. $C_1[p] \leftarrow c$; $L[p] \leftarrow L[v]$; $P[p] \leftarrow v$;
11. `Enqueue`$(p, C_1[p])$;
12. **if** $c = C_1[v]$, $C_2[p] \leftarrow C_2[v] + 1$;
13. **else, if** $c = C_1[p]$ **and** $L[p] \neq L[v]$,
14. **if** $c = C_1[v]$,
15. **if** $C_2[p] = C_2[v] + 1$, $L[p] \leftarrow$ TZ;
16. **else** $L[p] \leftarrow$ TZ;

This algorithm is fast and has the same speed performance as the IFT-WS [6]. The solution of TZWS is optimal because it is based on IFT, it keeps therefore the optimality of the shortest-path forest solution as demonstrated in [7, 6]. Besides, the other algorithms generally depend on arbitrary decisions in processing order (which pixel is removed first from a priority bucket of the queue?) that are not in the strict definition of WS and that introduce bias (see the different solutions of Fig. 2(g)–(j)). Observe that the bias problem does not occur only on plateaux and may be unacceptable for some applications (e.g. precise measures on segmented objects).

The TZWS can also be obtained without using an ordered queue by processing the image data in raster-scan and

anti raster-scan order alternatively until stability of the result (algorithm not presented here).

3 Bottlenecks

3.1 Watershed vs. watermerging

As said in section 1, the watershed (WS) transform is compared to the flooding of a topography where dams are built to prevent distinct colored waters from merging (suppose that a color is assigned to each marker/minimum). Now, let us illustrate the watermerging paradigm. For an intuitive comprehension, put the topography (representing the image) up-to-down (see an illustration in Fig.3). Imagine that each marker (former minimum that is now regional maximum) is a source of colored water. When colored waters meet together, no dam is built but the colored waters naturally merge into a water of blended color. Holes are punched in the regional minima (former maxima) for draining the water. Supposing that (abundant) waters propagate along all negative slopes (not only the steepest as occurs in reality), we get colored hills and possibly regions with blended colors. These regions correspond to the (multicolor) tie-zone. The up-to-down transformation is not used in practical implementations but only for an intuitive explanation of the watermerging paradigm. Watermerging is different from the *drop of water principle* or *rainfalling* algorithm [12].

As we defined earlier, the tie-zone is the region where pixels cannot be assigned to a specific label, i.e. there is no unique label that could be assigned to them. We can differenciate the tie-zone in labeled (or colored) tie-zones. A labeled tie-zone belongs necessarily to the tie-zone and assumes a specific tie-zone label: this label contains the information of which labels could be potentially assigned to the labeled tie-zone. In other words, the blended color of a colored tie-zone contains all the pigments of original colored waters that merged together. For example, when label a and label b tie together at a region, this region becomes a labeled tie-zone and will assume a *merged-label* $\{a, b\}$. We say that the original labels a and b are included in the merged-label $\{a, b\}$: $a \in \{a, b\}$; $b \in \{a, b\}$.

The Label Merging (LM) algorithm is a useful variation of the previous algorithm 1. When waters from different minima are merging, it assigns a new blended label to the region invaded by these waters. Substitute in algorithm 1 TZ label (lines 15-16) by `MergeLabels`$(\mathbb{L}[p], \mathbb{L}[v])$ and the simple label map L by a merged-label map $\mathbb{L}$. So, the final labeled image allows a traceability on tie zones: it informs exactly which and how many colored waters of different labels tied together at each node. By analyzing the pigment mix, one can deduce which original pigments are included in. There is no more one TZ label but so many as the distinct label mergings (4 in the example of Fig. 2(c)).

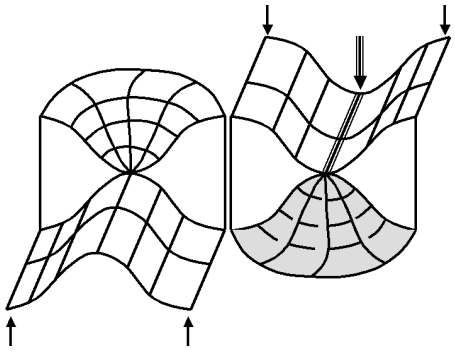


Figure 3. Watermerging paradigm. On the left (right), the arrows point the 2 minima (maxima) of the topography. A triple line (of constant altitude) shows the bottlenecks locus and constitutes with the gray zone the tie-zone where waters merged. Only the front bottleneck in contact with the gray zone (its influence zone) is non-trivial.

3.2 Multiplicity-based Optimal Forests (MOF)

We define the *multiplicity* $m(p)$ of a pixel p as the number of original labels $\lambda_i \in \Lambda$ that could be assigned to it, i.e. the number of labels that tied together. Clearly, a pixel in a catchment basin has always a multiplicity equal to one and a pixel in a tie-zone has necessarily multiplicity greater than one (see Fig.2(f)). Formally, the labeling function λ and the multiplicity m are defined as follows:

$$\begin{array}{rcl} \lambda : S & \rightarrow & \Lambda \\ s & \mapsto & \lambda(s) \end{array}$$

$$\begin{array}{rcl} m : V & \rightarrow & \mathbb{N} \\ p & \mapsto & m(p) = CARD\{\mathbb{L}[p]\} \end{array}$$

And, assuming that Λ is the set of original labels and Λ_I the set of all possible subsets of Λ, the label map $\mathbb{L}$ is now:

$$\begin{array}{rcl} \mathbb{L} : V & \rightarrow & \Lambda_I \\ p & \mapsto & \mathbb{L}[p] = \{\lambda_1, \lambda_2, \cdots, \lambda_{m(p)}\} \end{array}$$

As we saw in sections 2.1 and 2.2, there can be many optimal forests corresponding to an image. In other words, there can be many possible predecessors for some nodes of the image-graph. We distinguish a special type of optimal forests: the Multiplicity-based Optimal Forests (MOF).

In the set of possible optimal forests, the MOFs are those where each node points to the predecessor (father) of highest multiplicity among all possible predecessors. Note that there can exist many MOFs associated to a same image-graph. Figure 2 shows examples of a non-MOF (d) and two MOFs (b) and (e).

3.3 Bottleneck characterization

The bottlenecks are particular pixels that play an important role in the comprehension of the image and its partition in catchment basins or in tie-zone. Indeed, they are the points from which tie-zones appear.

Intuitively, they represent a unique and thin access ('bottleneck') for the water to enter and invade another region ('bottle'). More generally, in our context, they are the first contacts between two (or more) wave fronts of different colored merged-labels (and whose merging will result in another merged-label). Once the wave fronts have merged, the resulting wave front can, in some cases, invade a region (the 'bottle' of the bottleneck). Clearly, a bottleneck is never in a CB as it is a merging point.

In the watermerging paradigm (LM), *bottlenecks* are defined as nodes (pixels) whose merged-label is different from the predecessor's merged-label, when considering any particular MOF of the image-graph. For example, the bottlenecks of the image of Fig. 2(a) (in bold) can be detected from either MOF in (b) or (e) using the definition:

$$\mathbb{L}[p] \neq \mathbb{L}[P[p]] \Leftrightarrow p \text{ is bottleneck} \quad (1)$$

Let us consider now the region whose access is permited by the bottleneck: the influence zone. If all possible MOFs are considered, the *extended influence zone* $IZ_e(b)$ of a bottleneck b is the set of all possible descendants (direct and indirect children) of the bottleneck and itself. The number of nodes of a bottleneck's influence zone is called extended weight w_e of the bottleneck. $w_e(b) = CARD\{IZ_e(b)\}$. When considering only the descendant nodes with same merged-label as the bottleneck's one, we simply refer to the *influence zone* $IZ(b)$ of bottleneck b. And the number of nodes within is defined as the *weight* w of the bottleneck:

$$w(b) = CARD\{IZ(b)\}.$$

A bottleneck is *trivial* if and only if its weight is one. When the bottleneck has weight strictly greater than one, it is said *non-trivial* (see Fig. 3).

An adapted version of the LM algorithm allows to identify the bottlenecks on-the-fly during the construction of the forest. This Bottleneck Identification algorithm processes the label merging like in the LM algorithm and also updates the predecessor map according to the multiplicity criterion even if it has no influence on the cost and label maps. Thus, a particular MOF is obtained. When a node is removed from the queue (see algorithm 1), its definitive label is compared to the label of its predecessor to decide whether or not it is a bottleneck according to relation (1).

4 On the robustness of a segmentation

Now the bottleneck (BN) concept has been defined and related to the tie-zone (TZ) concept, we will show that the analysis of both TZs and BNs can be associated with the robustness of a segmentation. In this paper we call segmentation's *robustness* the impartiality or stability associated to the result of a segmentation process utilizing the same inputs and method. In our case, if different WS algorithms and implementations are used for segmenting a given image with given parameters (like adjacency and seeds), result's variations will affect the robustness (reliability).

Observe that a robust segmentation is not necessarily a 'good' segmentation in a semantic point of view but it is constant in relation to arbitrary choices of implementation. Moreover, one can be sure that a non-robust segmentation is only a possible but unreliable segmentation among numerous other ones. In other words, a non-robust segmentation is biased by the implemented algorithm as the image, seeds and segmentation process should not define thin WS but a large TZ.

4.1 Robustness based on tie-zone analysis

First, the area of the TZ can be a robustness measure. Larger the TZ is, less robust the segmentation is. For example, Fig.4(a) shows WS lines obtained from a non-filtered airplane image gradient by the Vincent and Soille's algorithm. Clearly, anyone can observe an oversegmentation as the *airplane* and the buildings are the expected objects of interest. This result is not only 'bad' in a semantic point of view but also very partial. The large and numerous tie-zones of the TZWS in Fig.4(d) point out the high uncertainty that affects any thin solution. In Fig.4(b) and (c), a filtering was applied on gradient's minima : basins (minima) with height smaller than $h = 10$ and $h = 35$ were respectively removed. The segmentation is much better in both semantic and robustness points of view (Fig. 4(e)(f)). It is necessary to normalize the TZ area with the image size to have an objective and size-independent robustness measure. The first (non)robustness measure is consequently:

$$R_1 = \frac{CARD\{T\}}{CARD\{V\}}.$$

We can also observe that in many cases as in Fig.4(f), the TZ is reduced to a thin WS line contouring the object of interest. As the output of the segmentation can be separating lines, we may consider that the TZ area of this line

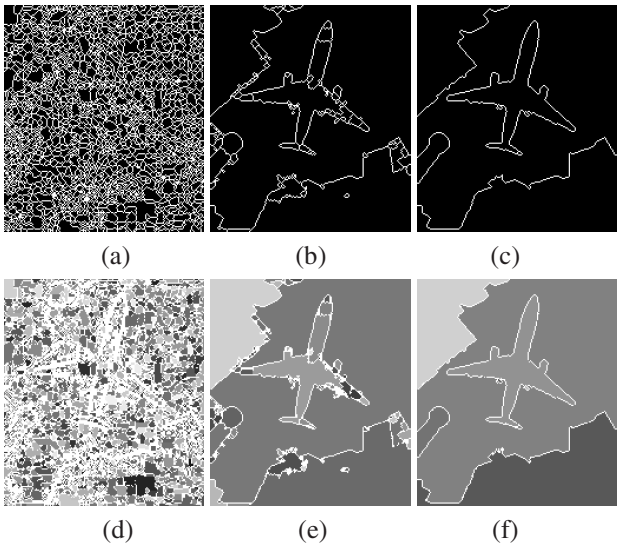


Figure 4. *Airplane* **image: Watershed lines (by Vincent-Soille's algorithm) of the gradient (a) and the filtered gradient (b)(c) (with** $h = 10; 35$ **resp.). (d)–(f) Respective TZ in white.**

does not represent a lack of robustness. Furthermore, when a lot of objects of interest are expected (cells separation for example), the previous measure R_1 will bias the reality. We cannot consider the lines contouring the objects as a real handicap for robustness. Thus, we apply an erosion ϵ on the TZ to eliminate the contouring lines and, thereby, not to have a measure depending on the number of objects. The *residue* R of this erosion represents now the thickness of the TZ without depending on the number of objects. Therefore, a second objective measure of (non)robustness is the normalized area of the residue:

$$R_2 = \frac{CARD\{\epsilon(T)\}}{CARD\{V\}} = \frac{CARD\{R\}}{CARD\{V\}}$$

For an application, one may consider that R_2 must be less than 1% to ensure a robust segmentation. After processing the TZWS for an image and its filtered versions, the variation of R_2 is computed and segmentations with the required robustness are automatically selected. Figure 5 shows this variation for 8 images and the 1%-threshold. Observe that the artificial *weavetile* segmentation is very robust for $h \geq 0$ whereas natural *mirage* gradient image becomes robust for $h \geq 6$ and *leaf-grass* for $h \geq 37$ (Fig. 6). This is due to the grass background and natural light and shade effects. Clearly, when the image is too much filtered, the number of objects decreases and can be less than the expected number (no object at worst!): the image is undersegmented. Thus, there is a compromise between having a robust segmentation and a semantically good segmentation. If this compromise is impossible, another segmentation strategy (filtering, marker choice, etc.) must be applied. Semantic criteria together with R_2 can help the user to choose a reasonable filtering.

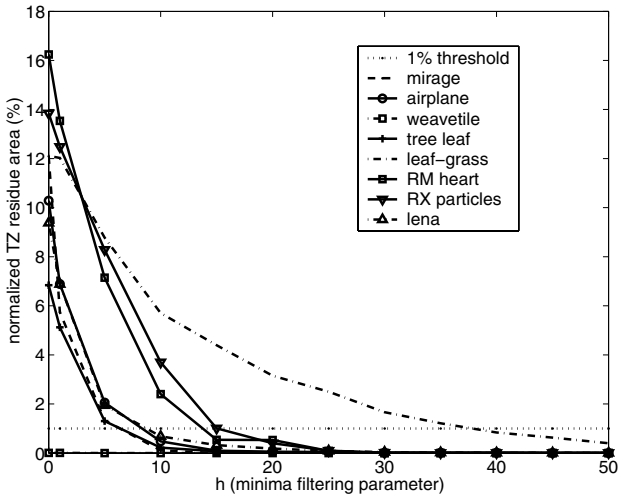


Figure 5. R_2 **in function of** h **for** *weavetile*, *tree-leaf*, *mirage*, *airplane*, *lena*, **MR** *heart*, **XR** *particles* **and** *leaf-grass* **images.** $R_2 < 1\%$ **for** $h \geq 0, 3, 3, 8, 8, 14, 15, 37$ **respectively.**

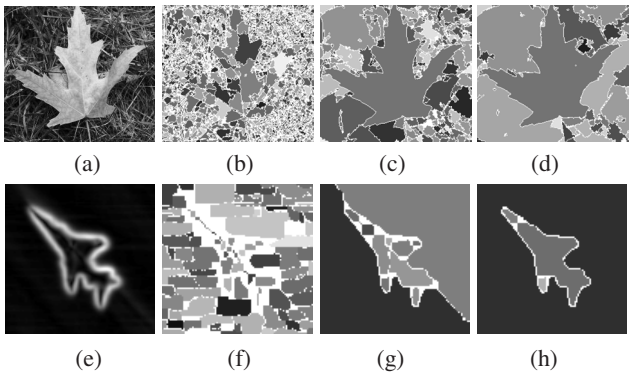
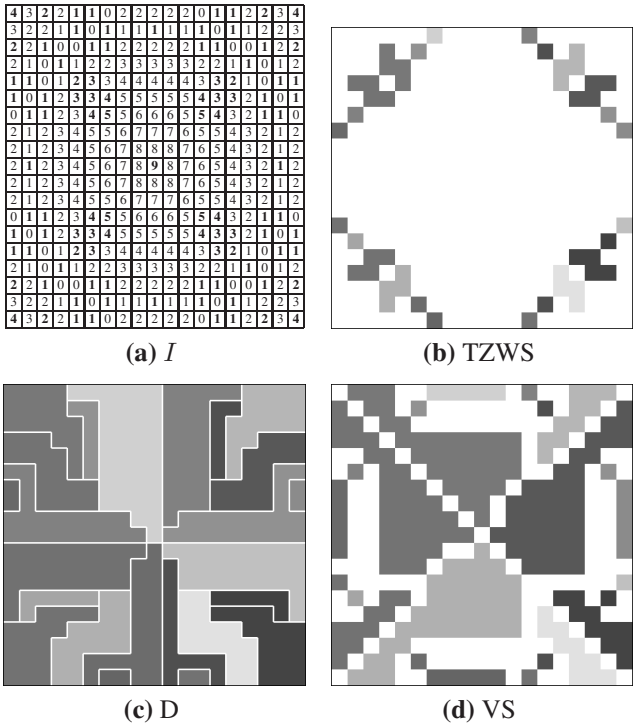


Figure 6. (a) Original *leaf-grass* **image. (b)–(d) TZWS (white) after filtering the gradient minima with** $h = 10; 35; 50$ **(2936, 411, 123 min.) resp. (e) Input** *mirage* **gradient image. (f)–(h) TZWS with** $h = 0; 5; 10$ **(116, 13, 4 min.) resp.**

The pseudo-cone of Fig. 7 is an interesting case of very large TZ. Any thin WS solution is totally unreliable and inconsistent because many lines have to converge to a same central pixel by passing through same pixels when closer to the top. This phenomenon is due to the discretization of the space and the non-null thickness of the lines. Observe that only the TZWS (b) has a 4-axis-symmetry like the input image (a). The other algorithms (c)(d) are biased by the processing order.



(a) I **(b)** TZWS **(c)** D **(d)** VS

Figure 7. Pseudo-cone. (a) Input image with 24 minima at 0-level (4-adjacency) and 101 bottlenecks (in bold). (b) TZWS (white). (c)–(d) WS by Dijkstra's and Vincent-Soille's algorithms (raster-scan order).

4.2 Robustness based on bottlenecks analysis

In section 3, we saw that bottlenecks are the roots of the patches that constitute the TZ. They are thereby the origin of the segmentation's non-robustness. The bottlenecks analysis allows to know how the sources of partiality are distributed in the segmented image and how large their influence zones are. Figure 8 shows for example a detail of the non-filtered *mirage* gradient image segmented by Label Merging (LM) algorithm. CBs (solid gray) meet and bottlenecks (BN) appear (white contour). Some of them have a large influence zone (IZ). Others have a little one or are trivial.

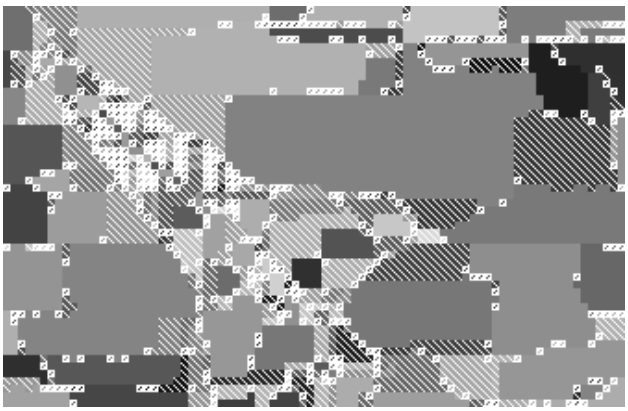


Figure 8. LM algorithm on *mirage* (detail): bottlenecks (white contour) and IZs (shaded).

We compute therefore the histogram H of the BNs according to their weight w. It informs which types of BNs are most present in the segmented image. For example, for a same R_2 or TZ area, there can be a lot of trivial bottlenecks in the entire image which proves that the potential bias is not concentrated in a particular region; or maybe, there can be few BNs with important weight. It depends on the essence of the image. This BNs distribution by weight can be a useful information for the choice of filtering strategy when the user attempts to increase the segmentation's robustness. More useful is to know what contribution each type of BNs has. To this end, the weighted histogram H_w is computed. For each weight w corresponds the sum of the weights (IZ areas) of all BN with weight w. So we have the contribution of each BN-type to the total TZ area. Look for example at the cumulated weighted histogram H_w^c in Fig. 9(a) for the non-filtered *mirage* case. We can see that the TZ area is greater than 2500 and there exists BN with $w \approx 120$ or 80 and nearly 500 trivial BNs. To have more compact informations, the quartiles are calculated (Fig. 9(b)). For the *mirage* image segmentation without filtering, the non-robustness is due to BNs with weight $w \leq 2$ (for 25%), BNs with $2 < w \leq 9$ (for other 25%), BNs with $9 < w \leq 32$ (for 25% too) and BNs with $32 < w \leq 123$ (for the last 25%). Interestingly, for the *leaf-grass* image, the filtering modifies this distribution of contribution in an unexpected way. It creates BNs with larger IZs. Note that the total TZ area increases with this filtering too. Be careful that it does not appear in the quartiles diagrams and that the ranges of BN weights contribute *equally* to the TZ area.

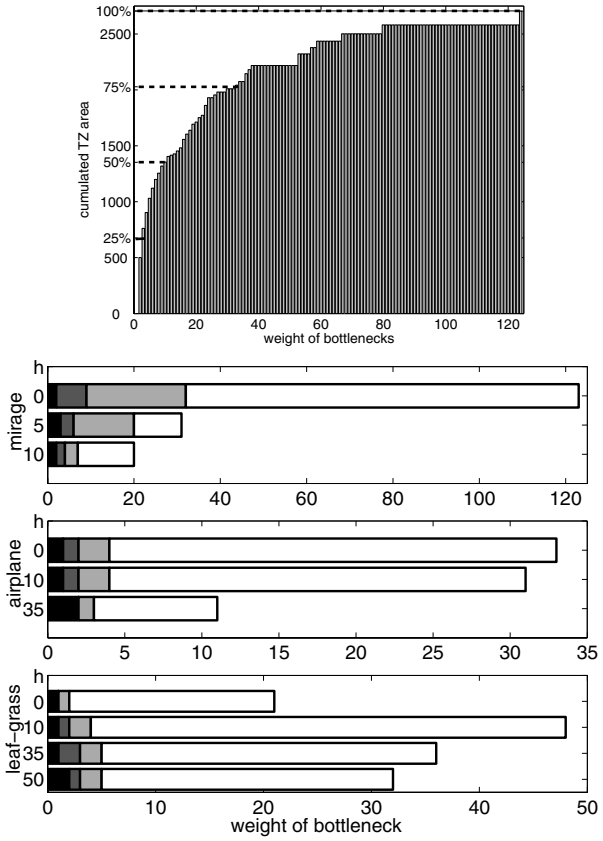


Figure 9. (a) Cumulated weighted histogram. (b) Quartiles for 3 segmented images varying h: w-ranges of the BNs that contribute for each quarter of the total TZ area.

In conclusion, the bottlenecks analysis gives information on how a segmentation can be biased whereas the TZ analysis does on how much it can be.

5 Conclusions and future work

In this work, we recalled the tie-zone watershed (TZWS) transform and, for understanding the TZ's essence, defined a new concept based on the watermerging paradigm: the bottleneck (BN). Each BN is responsible for a part of the TZ and, therefore, part of the possible bias of a WS transform. Theoretical examples and real images were shown to demonstrate that some WS segmentations have no sense when the TZ is too large: they are not representative. Furthermore, the robustness of a segmentation can be characterized quantitatively by the (eroded) TZ's area and qualitatively by the BN's distribution and contribution.

In future work, we will show the central role of BNs in a thinning procedure of the TZWS for segmentation purpose. We also intend to investigate the relation of the bottlenecks with pass-value and saliency concepts used by the topological watershed.

Acknowledgments

This work is supported by CAPES.

References

[1] R. Audigier, R. Lotufo, and M. Couprie. The tie-zone watershed: Definition, algorithm and applications. In *IEEE Proceedings of ICIP'05*, Genova, Italy, Sept. 2005. In press.

[2] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, 1979.

[3] M. Couprie and G. Bertrand. Topological grayscale watershed transformation. In *SPIE Vision Geometry VI Proceedings*, volume 3168, pages 136–146, 1997.

[4] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959.

[5] A. Falcão, B. da Cunha, and R. Lotufo. Design of connected operators using the image foresting transform. *SPIE on Medical Imaging*, 4322:468–479, Feb. 17-23 2001.

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

[7] R. Lotufo and A. Falcão. The ordered queue and the optimality of the watershed approaches. In *5th International Symposium on Mathematical Morphology*, pages 341–350, Palo Alto (CA), USA, June 2000. Kluwer Academic.

[8] R. Lotufo, A. Falcão, and F. Zampirolli. IFT-watershed from gray-scale marker. In *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing*, pages 146–152, Fortaleza (CE), Brazil, October 2002. IEEE Computer Society.

[9] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113–125, 1994.

[10] F. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Communication and Image Processing*, 1(1):21–46, 1990.

[11] L. Najman and M. Couprie. Watershed algorithms and contrast preservation. In *Discrete geometry for computer imagery*, volume 2886 of *Lecture Notes in Computer Science*, pages 62–71. Springer, 2003.

[12] J. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41(1-2):187–228, January 2000.

[13] 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):583–598, 1991.