Pattern Recognition Letters 53 (2014) 24–30

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

A new algorithm for number of holes attribute filtering of grey-level images ☆

Juan Climent [a,*], Luiz S. Oliveira [b]

[a] Barcelona Tech (UPC), Jordi Girona, 1, 08034 Barcelona, Spain
[b] Federal University of Parana (UFPR), Rua Cel. Francisco Heráclito dos Santos, 100, Curitiba, Brazil

ARTICLE INFO

Article history:
Received 20 February 2014
Available online 30 October 2014

Keywords:
Attribute filtering
Connected operators
Euler number

ABSTRACT

In this paper we present a new algorithm for filtering a grey-level image using as attribute the number of holes of its connected components. Our approach is based on the max-tree data structure, that makes it possible to implement an attribute filtering of the image with linear computational cost.

To determine the number of holes, we present a set of diverse pixel patterns. These patterns are designed in a way that the number of holes can be computed recursively, this means that the calculations done for the components of the image can be inherited by their parent nodes of the max-tree. Since we do not need to re-calculate the attribute data for all connected components of the image, the computation time devoted to the attribute computation remains linear.

1. Introduction

In mathematical morphology, connected filters [1] are operators that modify an image by only suppressing connected components, and thus, preserving its original contours. Attribute filters [2] are connected operators that keep certain shapes in the image based on a wide range of criteria. Different attributes can be used, like elongation, moment of inertia [3], energy [4], noncompactness [5], circularity [6], etc. The idea behind this approach is to think of attribute filtering as selecting shapes in images based on prior knowledge of the objects of interest. A connected component is preserved if it meets the corresponding attribute criterion, or removed otherwise.

The common attributes used for filtering binary images (like area, perimeter, moments, or Euler number) can be extended to grey-level images by threshold decomposition [7]. A grey-scale image can be decomposed in many binary images (called cross sections or level sets) by thresholding it at each grey-scale level. A cross section at level h is given by the set of all pixels greater or equal than h. A straightforward implementation of this idea, filtering each cross section as an independent binary image, leads to very inefficient algorithms. Nevertheless, attribute filtering of grey-level images can be efficiently implemented using Max-Tree algorithms [8–10].

Here, we propose a new algorithm for filtering a grey-level image using as attribute the number of holes of the image connected components. The number of holes of a connected component is a topological property closely related to the Euler number.

The Euler number is defined as the difference between the number of connected components and the number of holes in a binary image (Equation (1)). It is an important attribute, invariant to several image transformations such as translations, rotations, scale, projections, and even some non-linear deformations. Traditionally, it has been used in a great amount of applications, like signature verification [11], detecting malaria parasites in blood images [12] , reflectance-based object recognition [13], etc.

$$\varepsilon = N - H \tag{1}$$

There exist some algorithms [14–17], and patents [18] for computing the Euler number of a binary image. The most popular algorithm [19] is based on counting certain 2×2 pixel patterns called bit-quads.

The Euler number and the number of holes are equivalent concepts for a single connected component. Since for a single connected component, N value is always 1 in Equation (1), we can conclude that ε and H are practically the same thing. The result of computing the value of the Euler number of a connected component is always 1 minus its number of holes. Therefore, the same algorithm used for computing the Euler number can be used for computing the number of holes of a connected component.

Although there exist several algorithms for computing the number of holes (or Euler number) of the blobs in a binary image, to the best of our knowledge, no algorithm has been proposed up until now for computing the number of holes of the connected components of a grey-level image. Classical features used for grey-level attribute filtering are based on shape or size properties (like moments, volume

☆ This paper has been recommended for acceptance by G. Borgefors.

* Corresponding author. Tel.: +34 934137958.
E-mail address: juan.climent@upc.edu (J. Climent).

http://dx.doi.org/10.1016/j.patrec.2014.10.012
0167-8655/

or perimeter), but in this paper, we describe an algorithm that uses a topological attribute (number of holes) for filtering the image.

The rest of this paper is organized as follows: In Section 2 we review connected filters and the attribute filtering algorithm. Section 3 introduces our algorithm for filtering an image using the number of holes as attribute. We provide some results of the algorithm in Section 4, and make some conclusions in Section 5.

2. Attribute filtering

Connected filters are those that preserve the contours of the original image. They cannot create new contours, or modify the position of the existing ones. Therefore, they have very good contour-preservation properties and are able to low-level filtering and higher-level object recognition. These filters modify the image removing only flat zones [20], which are connected components where the image grey-level is constant.

The hierarchical representation of the connected components by means of max and min-tree, made possible the implementation of efficient algorithms to compute connected filters [10]. The use of these trees allows more sophisticated forms of filtering, like the one based on attributes [8]. Component features, called attributes, are computed on the nodes of the max-tree. The max-tree should contain both the hierarchy of connected components in the image, and the attributes for each component to be used as a filter criterion.

Since max-trees are used in many different applications (motion extraction [21], MSER feature extraction [22], segmentation, 3D visualization [6]), several algorithms have been proposed till now to compute the max-tree. Some of them are dedicated to a particular task, and are not useful for other purposes. In our work, we use the algorithm proposed in [10], that uses the max-tree to achieve, specifically, an efficient sequential implementation of attribute filters. It has a worst-case computational cost $O(kN)$ (being k the number of grey-levels in the image), but which is effectively linear for most natural images. In [23] they provide a full and exhaustive comparison of the state-of-the art of algorithms for computing the max-tree.

The algorithm proposes a separation of the filtering process into two stages: a tree construction stage, and a filtering stage.

In the first stage, the tree is built sequentially, structuring the pixels in a way that leads to efficient implementations of the further filtering process. As seen in Fig. 1, each node of the max-tree points to its parent (which has a strictly lower grey-level). The nodes corresponding to the components with the highest intensity are the leaves. The root, having the lowest grey-level, points to itself. This way of linking nodes simplifies the computation of component attributes since every parent inherits the data of its descendants. When dealing with increasing attributes (like area or volume), inheritance is a simple accumulation. The attribute considered in this paper is non-increasing (case of shape attributes), then the inheritance needs a more complex handling, described in next section.

The tree construction uses a flood-filling approach based on hierarchical FIFO queues, where each queue corresponds to a particular grey-level value. These queues are used to define an appropriate order for scanning and processing the pixels. It performs a depth-first sweep of the tree, starting at the lowest grey level of the image (the root), and moving upward in grey scale. When a pixel is processed, the algorithm updates the data of the node currently being flooded. After that, it inspects its neighbours and places them in their respective queues. If a neighbouring pixel is at a higher level, flooding proceeds recursively at this new level. Once a node is completely flooded, the process is completed by determining its parent and attribute fields. The function returns the node attribute information to its parent and the flooding process goes on at the new level.

The max-tree creation relies on a recursive flooding procedure. The algorithm has basically two steps: the first one performs the propagation and the updating of the status of each pixel being processed, and

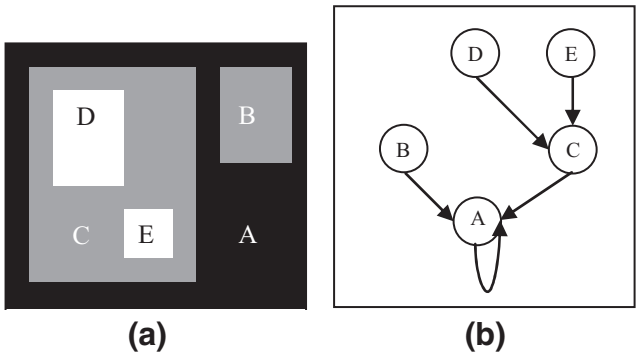

Fig. 1. A synthetic image and its corresponding max-tree. Region grey-level values: A = 0, B = 1, C = 1, D = 2, E = 2.

computes the attribute data of each node, whereas the second step defines the parent/child relationships. The process finishes when all pixels have been processed. The details of this algorithm can be found in [10].

In the filtering stage, it is decided which nodes (connected components) will be suppressed from the tree. The pruning process is governed by a criterion that may involve simple features such as size, contrast, or more complex ones such as texture, motion, or even a criterion of similarity to predefined shapes. Finally the filtered image is obtained from the pruned tree.

Attribute filters use a criterion T to determine which connected components (CC) are preserved. If $T(CC)$ is true, then $Filter(CC) = CC$; otherwise, $Filter(CC) = NULL$. Usually, T is defined by using some attribute of the connected component, and comparing it to a threshold value. The criterion T can be that attribute value is either higher or lower than the threshold. Any node can be removed from the tree. There are four typical strategies for removing the filtered nodes:

- the max rule prunes the branches from the leaves up to the first node that has to be preserved.
- the min rule prunes the branches from the leaves up to the last node that has to be removed.
- the direct rule consists of removing the selected nodes even if this does not create a pruning. The pixels belonging to the connected components that have been removed are merged to the node of their first ancestor that has to be preserved.
- the subtractive rule [24] is the same as the direct rule except that the grey levels of surviving descendants of removed nodes are also lowered, so that the contrast with the local background remains the same.

3. Algorithm for attribute filtering by number of holes

3.1. Computing the number of holes for a binary component

Our algorithm for computing the number of holes of a connected component is originally based on the Euler number. The most popular algorithm for computing the Euler number of an image is the well known algorithm based on locally binary patterns [19], used in the MATLAB image processing toolbox. The local binary patterns consist in the following set of 2×2 binary pixel quads:

$$Q_1 = \left\{ \begin{matrix} 0\,0 \\ 1\,0 \end{matrix}, \begin{matrix} 0\,0 \\ 0\,1 \end{matrix}, \begin{matrix} 0\,1 \\ 0\,0 \end{matrix}, \begin{matrix} 1\,0 \\ 0\,0 \end{matrix} \right\}$$

$$Q_2 = \left\{ \begin{matrix} 1\,1 \\ 0\,1 \end{matrix}, \begin{matrix} 1\,1 \\ 1\,0 \end{matrix}, \begin{matrix} 1\,0 \\ 1\,1 \end{matrix}, \begin{matrix} 0\,1 \\ 1\,1 \end{matrix} \right\}$$

$$Q_3 = \left\{ \begin{matrix} 0\,1 \\ 1\,0 \end{matrix}, \begin{matrix} 1\,0 \\ 0\,1 \end{matrix} \right\}$$

Where Q_1 corresponds to outside corners, Q_2 to inside corners, and Q_3 to eight-connectivity connections. Let C_1, C_2, and C_3 be the number of patterns Q_1, Q_2, and Q_3, respectively present in a binary image I. In [25] they determine that under the definition of four-connectivity, the Euler number can be computed using Equation (2), and under the definition of eight-connectivity, Equation (3) should be used.

$$\varepsilon_4(I) = \frac{C_1 - C_2 + 2C_3}{4} \tag{2}$$

$$\varepsilon_8(I) = \frac{C_1 - C_2 - 2C_3}{4} \tag{3}$$

Instead of using two equations for the different connectivities and a single set of quads, we propose to use a single equation (Equation (4)) and different sets of quads for each connectivity.

Under the definition of 4-C:

$$Q_1^{C4} = \left\{ \begin{matrix} 0 & - \\ 1 & 0 \end{matrix} , \begin{matrix} - & 0 \\ 0 & 1 \end{matrix} , \begin{matrix} 0 & 1 \\ - & 0 \end{matrix} , \begin{matrix} 1 & 0 \\ 0 & - \end{matrix} \right\}$$

$$Q_2^{C4} = \left\{ \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} , \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix} , \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix} , \begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix} \right\}$$

Under the definition of 8-C:

$$Q_1^{C8} = \left\{ \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix} , \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix} , \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix} , \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix} \right\}$$

$$Q_2^{C8} = \left\{ \begin{matrix} 1 & - \\ 0 & 1 \end{matrix} , \begin{matrix} - & 1 \\ 1 & 0 \end{matrix} , \begin{matrix} 1 & 0 \\ - & 1 \end{matrix} , \begin{matrix} 0 & 1 \\ 1 & - \end{matrix} \right\}$$

(Where the pixel value '–' does not care)

$$\varepsilon(I) = \frac{C_1 - C_2}{4} \tag{4}$$

Using this approach, we get rid of Q_3 masks, and only count for outside and inside corners. Q_3 is not necessary if using a different set of masks for different connectivities.

Proof. Q_3 only takes into account patterns of pixels that are diagonally connected. Let us assume that there is a pixel pattern in the image like one of those quads of Q_3. Then, counting Q_1, Q_2, and Q_3 patterns, $C_1 = 0$, $C_2 = 0$, and $C_3 = 1$. Therefore, ε_4 would increase in two units, and ε_8 would decrease in two units, using Equations (2), and (3) respectively.

Using Q_1^{C4} and Q_2^{C4} patterns, $C_1 = 2$, and $C_2 = 0$. Using Q_1^{C8} and Q_2^{C8} patterns, $C_1 = 0$, and $C_2 = 2$. Therefore ε would also increase in two units for 4-C, and decrease in two units for 8-C using Equation (4). □

Attribute filtering is a connected operator, thus, it filters connected components. Since the Euler number is defined as the difference between the number of blobs and the number of holes in the image, it makes no sense to compute it for a single connected component, which always consists of a unique blob. Therefore, the real interesting attribute for filtering the connected components is just their number of holes, not their Euler number. Equation (5) is a trivial derivation of Equations (1) and (4) for computing the number of holes of a binary connected component of an image.

$$\text{Number of holes } (CC) = 1 - \frac{C_1 - C_2}{4} \tag{5}$$

3.2. Extension to grey-level images

First, we need some definitions to extent to grey-level images, the idea given in previous section.

The h-level set of a grey-level image I is defined as:

$$V_h = \{x \in I \,|\, f(x) \geq h\} \tag{6}$$

A dome D_i^h of a grey-level image is a connected region in which $f(x) \geq h$ for all $x \in D_i^h$, and all neighbors of D_i^h have a grey level smaller than h. D_i^h is defined as the ith connected component of the level set V_h.

An extension of attribute filters to grey scale can be made by using thresholded images V_h. The grey-scale image may be considered as a stack of level sets, each consisting of progressively bigger domes as h decreases.

As h decreases, new flat-zones are added to existing domes, and consequently, new inside and outside corners may appear and should be taken into account. Let p be an already non-processed pixel from connected component i, with grey-level h. p will add a new outside corner to D_i^h if, and only if, all its neighbours have a grey value strictly lower than h. On the other hand, it will add a new inside corner if, and only if, one of its neighbours has a grey value strictly lower than h, and the rest of them have a grey value higher or equal than h.

We present next, the sets of masks needed to compute this new corners for both four and eight connectivities. Q_2 masks must consider all possible cases where a pixel p has an inside corner as a neighbour. These cases include patterns where the neighbour pixels of the corner, are equal than pixel p but also those that are strictly greater than pixel p. The sets of masks are complete because they must consider all possible patterns, and they are mutually exclusive in a way that the same corner cannot be counted twice. It can be proved by simple enumeration of all possible patterns.

$$Q_1^{C4} = \left\{ \begin{matrix} L & - \\ p & L \end{matrix} , \begin{matrix} - & L \\ L & p \end{matrix} , \begin{matrix} L & p \\ - & L \end{matrix} , \begin{matrix} p & L \\ L & - \end{matrix} \right\}$$

$$Q_2^{C4} = \left\{ \begin{matrix} G & \mathrm{p} \\ L & G \end{matrix} , \begin{matrix} \mathrm{p} & G \\ G & L \end{matrix} , \begin{matrix} G & L \\ \mathrm{p} & G \end{matrix} , \begin{matrix} L & G \\ G & \mathrm{p} \end{matrix} , \begin{matrix} \mathrm{p} & g \\ L & G \end{matrix} , \begin{matrix} g & \mathrm{p} \\ G & L \end{matrix} , \begin{matrix} G & L \\ g & \mathrm{p} \end{matrix} , \begin{matrix} L & G \\ \mathrm{p} & g \end{matrix} , \begin{matrix} g & g \\ L & \mathrm{p} \end{matrix} , \begin{matrix} g & g \\ \mathrm{p} & L \end{matrix} , \begin{matrix} \mathrm{p} & L \\ g & g \end{matrix} , \begin{matrix} L & \mathrm{p} \\ g & g \end{matrix} \right\}$$

being:

- G grey-level higher than pixel p
- L grey-level lower than pixel p
- g grey-level higher or equal than pixel p
- l grey-level lower or equal than pixel p
- - do not care

$$Q_1^{C8} = \left\{ \begin{matrix} L & L \\ p & L \end{matrix} , \begin{matrix} L & L \\ L & p \end{matrix} , \begin{matrix} L & p \\ L & L \end{matrix} , \begin{matrix} p & L \\ L & L \end{matrix} \right\}$$

$$Q_2^{C8} = \left\{ \begin{matrix} \mathrm{p} & - \\ L & G \end{matrix} , \begin{matrix} - & \mathrm{p} \\ G & L \end{matrix} , \begin{matrix} G & L \\ - & \mathrm{p} \end{matrix} , \begin{matrix} L & G \\ \mathrm{p} & - \end{matrix} , \begin{matrix} g & - \\ L & \mathrm{p} \end{matrix} , \begin{matrix} - & g \\ \mathrm{p} & L \end{matrix} , \begin{matrix} \mathrm{p} & L \\ - & g \end{matrix} , \begin{matrix} L & \mathrm{p} \\ g & - \end{matrix} \right\}$$

The columns of masks are ordered according to the corner pointing to NE, NW, SW, and SE directions. For every corner direction, Q_2 rows are ordered according to a combination of G-g patterns that guarantees that all possible pixel patterns are considered, and the same corner is never counted more than once. Fig. 3(a) shows two examples of different Q_2 patterns. Notice that if masks in Q_2^{C4} were not ordered according these G-g patterns (i.e. we only had used g), then the inside corner of region B (the lower left one) would be counted three times.

For binary images, this new set of masks is equivalent to the one presented in Section 3.1.

Proof. In a binary image, pixel p belongs to a connected pixel from foreground (a '1' using Section 3.1 notation). A grey value strictly lower than p (a 'L' using new notation) is necessarily a '0'. Thus, Q_1 masks sets from Sections 3.1 and 3.2 are exactly the same.

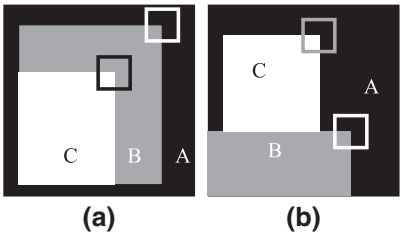

Fig. 2. In D_1^B corner of region C disappears in image (a), but remains in image (b).

A 'G' value will never show up in a binary image, since there are no grey values greater than '1'. Then, only the masks from last row of Q_2 set are possible. A grey value 'g' (higher or equal than p) is necessarily a '1' in a binary image. Thus, Q_2 masks sets from Sections 3.1 and 3.2 are exactly the same. □

3.3. Computing number of holes of all grey-level components

Pixel quad counting provides a very simple mean of determining the Euler number of a binary image. The computational complexity of the algorithm, for a binary image, is linear with the number of pixels in the image. When extending the algorithm to grey-level images, the computation time devoted to the analysis of the max-tree (criterion assessment and decision) is a function of the criterion complexity. This amount of time remains linear if the attribute can be computed recursively, that is, if it is possible to take into account the evaluation done for the child nodes while computing the attribute of the current node. For example, computing the area of a node is as simple as to count the number of pixels of the current node grey-level and add the area of its child nodes.

However, this extension to attribute filtering of grey-level images is not always so easy, specially when the attribute used is not increasing. The number of holes is not an increasing attribute, since the number of holes of a child node can be either higher or lower than the number of holes of a parent node. This limitation makes that the local binary pattern approach cannot be applied straightforward to the different level sets of a grey-level image. Inheritance of child nodes attributes cannot be done by just simple addition. When a node inherits the attributes from its children, the amount of local binary patterns resulting from the fusion strongly depends on both connected components topologies. A simple example in Fig. 2 shows different results of attribute inheritance depending on image topology.

In the original Max-Tree algorithm in [10], they use the term 'merge' to refer to the procedure of attribute inheritance from child nodes. For coherence with the original algorithm, we will use the same term. That is, D_i^h is a dome result of the merging of h-level flat zone i with its child nodes $D_j^{h_j}$, being $h_j > h; \forall j$.

In the first example (Fig. 2(a)), both flat zones B and C present a North-East corner pattern (first binary quad of Q_1), and only one should remain in D_1^B. In the second example (Fig. 2(b)), B and C present the same corner pattern again, but both corners remain in D_1^B. Thus, in the image shown of Fig. 2(b), the number of North-East corners of D_1^B could be computed by simple addition using the number of NE corners of D_1^C, but this approach would give wrong results in the image of Fig. 2(a).

We need to extend the set of patterns in order to solve the merging problem shown in Fig. 2. For this extension we need two new sets of patterns for every corner direction. The new sets of patterns, Q_3 and Q_4, take into account pixels whose neighbours are corners that will disappear when the respective components merge (the component where the pixel p belongs and the one where the corner does). The intuitive idea of this extension patterns is to subtract to the attribute value, those corners that vanish when different grey-level sets merge. For example, Fig. 2(a) shows two different corners pointing to NE. The

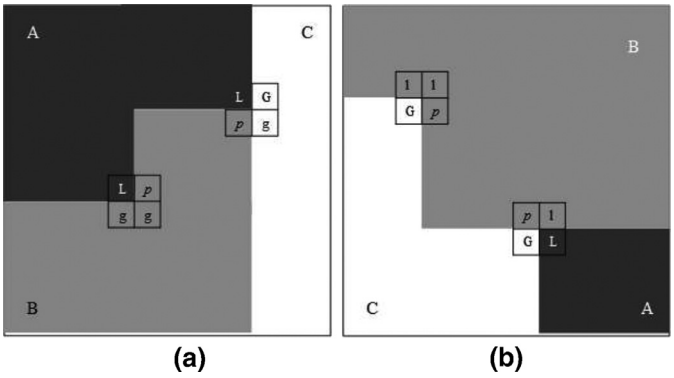

Fig. 3. (a) Example of presence of Q_2 patterns. (b) Example of presence of Q_3 patterns.

corner in component C should be considered a corner in D_1^C, but it will disappear in D_1^B (when connected component C merges with its parent B). For this reason, we should decrease the score to the attribute data of the corresponding B node (that is, $Q_3 = -1$). When region B inherits the attributes of region C, the score of node B (-1) should neutralize the score of node C $(+1)$ due to the presence of the NE corner shown in Fig. 2(a).

We present next the sets of patterns Q_3 and Q_4 for both four and eight connectivities (columns of patterns sets are sorted in a NE, NW, SW, SE order). Let C_3, and C_4 count for the number of patterns Q_3, and Q_4 respectively. Let p be an already non-processed pixel from a h-level connected component. C_3 will increase if, and only if, one of p neighbours was an outside corner and all the rest of its neighbours have a grey value lower or equal than h. C_4 will increase if, and only if, all of p neighbours have a grey value strictly higher than h (that is, p was an inside corner). Fig. 3(b) shows two examples of presence of Q_3 patterns.

$$Q_3^{C4} = \left\{ \begin{matrix} p & - \\ G & L \end{matrix}, \begin{matrix} - & p \\ L & G \end{matrix}, \begin{matrix} L & G \\ - & p \end{matrix}, \begin{matrix} G & L \\ p & - \end{matrix} \right\}$$

$$\left\{ \begin{matrix} l & - \\ G & p \end{matrix}, \begin{matrix} - & l \\ p & G \end{matrix}, \begin{matrix} p & G \\ - & l \end{matrix}, \begin{matrix} G & p \\ l & - \end{matrix} \right\}$$

$$Q_4^{C4} = \left\{ \begin{matrix} G & G \\ p & G \end{matrix}, \begin{matrix} G & G \\ G & p \end{matrix}, \begin{matrix} G & p \\ G & G \end{matrix}, \begin{matrix} p & G \\ G & G \end{matrix} \right\}$$

$$Q_3^{C8} = \left\{ \begin{matrix} L & p \\ G & L \end{matrix}, \begin{matrix} p & L \\ L & G \end{matrix}, \begin{matrix} L & G \\ p & L \end{matrix}, \begin{matrix} G & L \\ L & p \end{matrix} \right\}$$

$$\left\{ \begin{matrix} p & l \\ G & L \end{matrix}, \begin{matrix} l & p \\ L & G \end{matrix}, \begin{matrix} L & G \\ l & p \end{matrix}, \begin{matrix} G & L \\ p & l \end{matrix} \right\}$$

$$\left\{ \begin{matrix} l & l \\ G & p \end{matrix}, \begin{matrix} l & l \\ p & G \end{matrix}, \begin{matrix} p & G \\ l & l \end{matrix}, \begin{matrix} G & p \\ l & l \end{matrix} \right\}$$

$$Q_4^{C8} = \left\{ \begin{matrix} G & - \\ p & G \end{matrix}, \begin{matrix} - & G \\ G & p \end{matrix}, \begin{matrix} G & p \\ - & G \end{matrix}, \begin{matrix} p & G \\ G & - \end{matrix} \right\}$$

The functions for updating the attribute information of tree nodes (domes) are detailed in Algorithms 1–3. Algorithm 3 is the one that computes attribute information of a parent node, taking into account the information already computed for its child nodes. The number of inside and outside corners is updated by adding the number of child corners, but subtracting the number of corners that disappears in the merging procedure.

The filtering process, which is executed after flooding, visits each node starting from the root. For every node, a criterion is evaluated depending on its attribute value (the criterion is usually a specified attribute threshold). The function used to get the attribute value (number of holes) of a connected component is given in Algorithm 4, based on Equation (5).

Algorithm 1: NewNode(p)

// Creates a new connected component for pixel p

input : pixel being processed
output: creates node data

begin
$C_1 \longleftarrow CountPatternsQ_1(p)$
$C_2 \longleftarrow CountPatternsQ_2(p)$
$C_3 \longleftarrow CountPatternsQ_3(p)$
$C_4 \longleftarrow CountPatternsQ_4(p)$
end

Algorithm 2: AddToNode(p)

// Adds p to an existing connected component

input : pixel being processed
output: updates node data

begin
$C_1 \longleftarrow C_1 + CountPatternsQ_1(p)$
$C_2 \longleftarrow C_2 + CountPatternsQ_2(p)$
$C_3 \longleftarrow C_3 + CountPatternsQ_3(p)$
$C_4 \longleftarrow C_4 + CountPatternsQ_4(p)$
end

Algorithm 3: MergeNode(node,child)

// Inherits attribute information from child and updates attribute information of parent component

input : parent connected component, child connected component
output: updates parent node data

begin
$C_1 \longleftarrow C_1 + Child\_C_1 - C_3^-$
$C_2 \longleftarrow C_2 + Child\_C_2 - C_4^-$
end

Algorithm 4: GetAttributeValue(p)

// Gives the number of holes of connected component

input : pixel identifying a connected component
output: the number of holes of the connected component

begin
return $1 - \frac{C_1 - C_2}{4}$
end

4. Some tests

First of all, we test the proposed algorithm for computing the number of holes by means of counting corners, using the synthetic image shown in Figs. 4 and 5. Dome D_1^C has one hole considering eight-connectivity, and has no holes considering four-connectivity. Dome D_1^B has no holes in any case.

Fig. 4(a) shows the Q_1^{C8} and Q_2^{C8} patterns detected in flat zone C. For visual counting purpose, pixels p of Q_1^{C8} are marked with a slash, while pixels p of Q_2^{C8} are marked with a cross. There is no presence of patterns Q_3^{C8} or Q_4^{C8} in C. There can be found six Q_1^{C8} patterns ($C_1^{C8} = 6$) and six Q_2^{C8} patterns ($C_2^{C8} = 6$), thus, using Equation (5) the number of holes of dome D_1^C is 1. Fig. 4(b) shows the Q_1^{C4} and Q_2^{C4} patterns detected in flat zone C. For the case of four-connectivity we get $C_1^{C4} = 8$, and $C_2^{C4} = 4$, therefore, D_1^C has no holes considering four-connectivity.

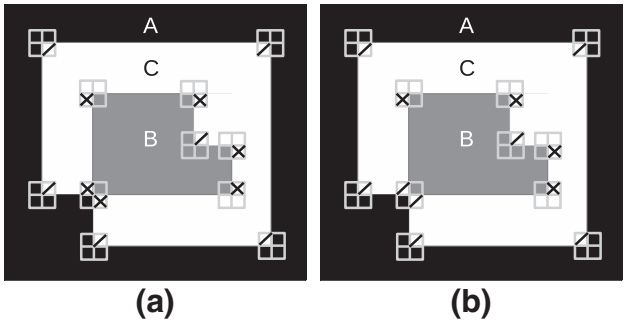

Fig. 4. (a) Quads found in C using 8-C. (b) Quads found in C using 4-C.

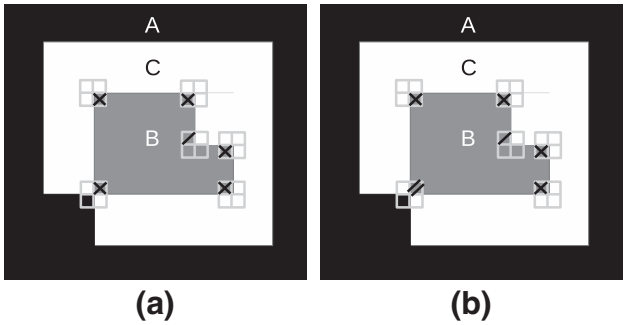

Fig. 5. (a) Quads found in B using 8-C. (b) Quads found in B using 4-C.

Fig. 5(a) shows the Q_3^{C8} and Q_4^{C8} patterns detected in flat zone B. Pixels p of Q_3^{C8} are marked with a slash, while pixels p of Q_4^{C8} are marked with a cross. There is no presence of patterns Q_1^{C8} or Q_2^{C8} in B. We count $C_3^{C8} = 1$, $C_4^{C8} = 5$. Using the merging procedure described, we obtain $C_1^{C8}(B) = C_1^{C8}(C) - C_3^{C8}(B) = 5$, $C_2^{C8}(B) = C_2^{C8}(C) - C_4^{C8}(B) = 1$. Thus, using Equation (5) the number of holes of dome D_1^B is 0.

Fig. 5(b) shows the Q_3^{C4} and Q_4^{C4} patterns detected in flat zone B. There is a pixel with a double slash because two different quads in Q_3^{C4} match the pixel pattern. The same pixel presents also a Q_2^{C4} quad. There is no presence of Q_1^{C4} patterns in B. We count $C_2^{C4} = 1$, $C_3^{C4} = 3$, $C_4^{C4} = 4$. Therefore, we get $C_1^{C4}(B) = C_1^{C4}(C) - C_3^{C4}(B) = 5$, $C_2^{C4}(B) = 1 + C_2^{C4}(C) - C_4^{C4}(B) = 1$. Notice that, regardless of which connectivity is being considered, D_1^B has five outside corners and one inside corner. The number of holes of dome D_1^B, using Equation (5), is 0.

We show next an example of the results of filtering real images using the number of holes as attribute. Once the max-tree has been created, the filtering strategy consists in simplifying the tree and reconstructing a new image from the simplified one. The tree nodes might be preserved or removed depending on the number of holes of their respective connected components. The objective in all cases is to enhance some image structures containing the desired number of holes, while reducing or suppressing other components with an inappropriate number of holes.

In our experiments we have chosen a direct rule strategy for removing the filtered nodes, that is, surviving connected components remain with the same grey level than in the original image. This choice has not a determining effect on the final result. We use direct filtering just for presentation purposes because it gives higher contrasted output images.

In this test we try to detect car license plates in outdoor scenes using the proposed attribute filtering. The criterion used for filtering the image is to retain connected components having exactly seven or eight holes. Small holes are also filtered using an area closing to avoid

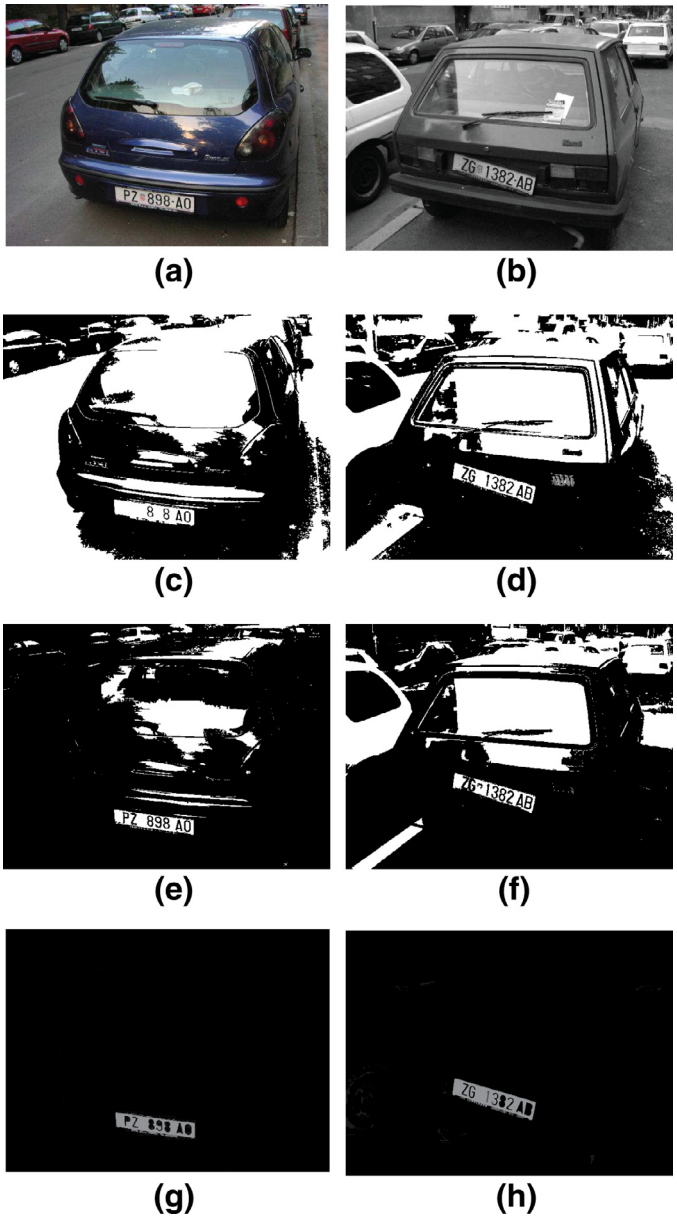

Fig. 6. (a) and (b) Original images, (c) and (d) binary images using Otsu's thresholding, (e) and (f) binary images using Pun's thresholding, (g) and (h) results using number of holes attribute filtering.

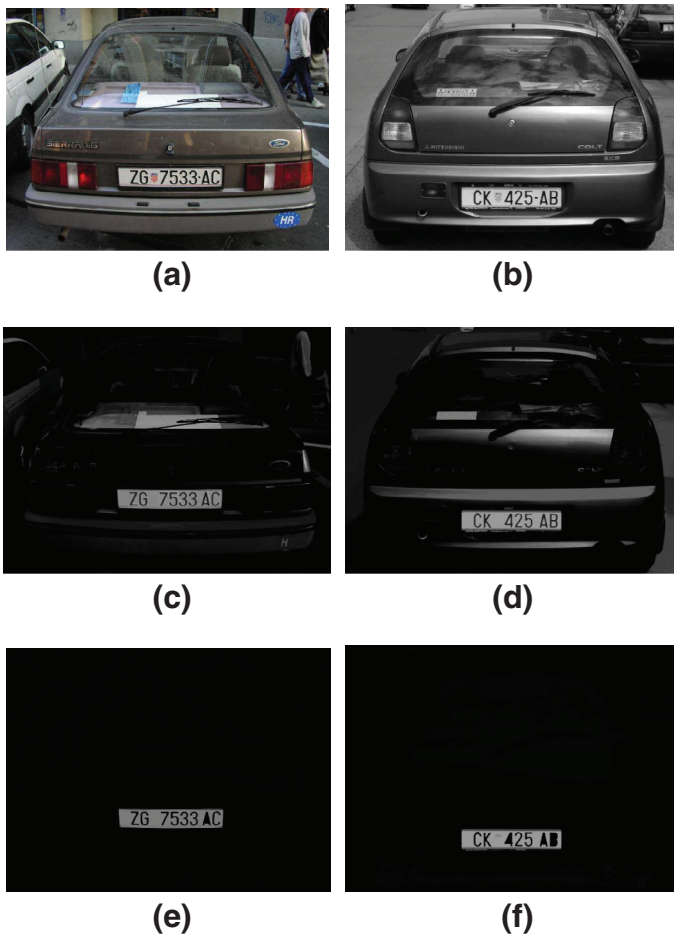

Fig. 7. (a) and (b) Original images, (c) and (d) results using shape attribute filtering, (e) and (f) results using number of holes attribute filtering.

the presence of small holes due to noise effect or irrelevant regions. We have not used any information about the hypothetical position or orientation of the license plates in the image.

Since it does not exist another algorithm for filtering a grey-level image by its number of holes, we compared our results with two other different approaches. The first one is simply to threshold the image, and filter the binary blobs according to their number of holes. The second approach is to use a grey-level attribute filtering using other shape attributes and compare the results with the filtering using the number of holes attribute.

In the first test we select those blobs in the binary image whose number of holes is equal to seven or eight. We have chosen the threshold level using two different classical algorithms, the first one based on clustering [26], the second one based on entropy maximization [27]. Fig. 6(c) and (d) shows the result of binarizing the car images using Otsu's threshold. The result shown in Fig. 6(d) is good, and the blob corresponding to the plate would be selected because it has eight holes. However, some numbers are lost in Fig. 6(c) due to the binarization process, thus, the plate would be removed because it has only four holes. If we use Pun's threshold instead, the result is good for the first car image (Fig. 6(e)), but now some holes are lost in Fig. 6(f) because they end up connected to the background. The conclusion of this experiment, as could be expected, is that it does not exist a threshold that suits for all images, and the result is too dependent on the thresholding process. Fig. 6(g) and (h) shows the result of filtering the grey-level images using the number of holes as attribute.

In a second test, we filter the grey-level images using shape attributes or the number of holes attribute. Concretely, Fig. 7(c) and (d) is the filtered image using elongation and compactness as attribute respectively. Shapes with an elongation lower than 2 have been removed, and the surviving shapes with a low compactness have been also removed because they present a complex shape. The result is that some unwanted objects (but compact and elongated shapes), like the paper on the back slide, the sticker in the rear screen, or some reflections are not removed. Using the number of holes attribute, only the plates remain (Fig. 7(e) and (f)). The conclusion of this experiment is that the number of holes can be a discriminative attribute, and give valuable information to the filtering process.

The objective of this paper is not to present a method for license plate detection, but to show the results of our algorithm in a real application. For an exhaustive evaluation of a method for license plate detection, we should definitely use additional shape attributes, but this is out of the scope of this work. The objectives of this experiment are, first, to show that the number of holes of connected components can be a useful attribute, second, to show that the proposed algorithm

filters correctly an image based on the number of holes, and third, to test the computing time of the algorithm. The algorithms were implemented in C language on a PC workstation (Core Duo at 3.0 GHz, 2GB RAM). We have used 515 (640×480) grey-level images for testing (available at: www.zemris.fer.hr/projects/LicensePlates/english/baza_slika.zip). Processing time of the attribute filtering based on number of holes is 0.38 seconds in average.

5. Conclusions

We have proposed a new algorithm for filtering the connected components of an image according to their number of holes. This topological property gives important information about the components of an image, and used together with other attributes can enhance the robustness of a filtering process.

The algorithm for computing the number of holes of each connected component was originally based on the use of the well known pixel quads patterns, but we propose a new set of patterns with the aim of extending the algorithm to grey-level images. The proposed new set of patterns ensures that the attribute data of each component should not be re-computed for every cross section of the image. The component attributes are updated by simple additions and subtractions each time that a new pixel is processed.

The filtering algorithm is based on a max-tree, which has a linear computational cost. Together with the fact that attribute data of every node can be computed recursively using the previous evaluation of their child nodes, the resulting implementation is suitable for real-time operation.

Acknowledgments

The authors would like to thank the courtesy of Prof. Slobodan Ribarić, from the University of Zagreb, who loaned us the car images used in the experiments.

This project was financially supported by the National Council for the Improvement of Higher Education (CAPES-Brazil) and the DPI2013-42458-P project (CICYT-Spain).

References

[1] P. Salembier, M. Wilkinson, Connected operators: A review of region-based morphological image processing techniques. IEEE Signal Proc. Mag. 6 (2009) 136–157.
[2] H.J.A.M. Heijmans, Connected morphological operators for binary images. Comput. Vis. Image Underst 73 (1) (1999) 99–120.
[3] M.H.F. Wilkinson, M.A. Westenberg, Shape preserving filament enhancement filtering, in: W.J. Niessen (Ed.), Proceedings of the Medical Image Computing and Computer-Assisted Intervention (MICCAI 01), 2001, pp. 770–777.
[4] Y. Xu, T. Géraud, L. Najman, Morphological filtering in shape spaces: Applications using tree-based image representations, CoRR. abs/1204.4758 (2012).
[5] G.K. Ouzounis, M.H.F. Wilkinson, Hyperconnected attribute filters based on k-flat zones. IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 224–239.
[6] M.A. Westenberg, J.B.T.M. Roerdink, M.H.F. Wilkinson, Volumetric attribute filtering and interactive visualization using the max-tree representation. IEEE Trans. Image Proc. 16 (12) (2007) 2943–2952.
[7] P. Maragos, R.D. Ziff, Threshold superposition in morphological image analysis systems. IEEE Trans. Pattern Anal. Mach. Intell. 12 (5) (1990) 498–504.
[8] R. Jones, Connected filtering and segmentation using component trees. Comput. Vis. Image Underst. 75 (3) (1999) 215–228.
[9] U. Braga-Neto, J.K. Goutsias, Grayscale level connectivity: Theory and applications. IEEE Trans. Image Proc. 13 (12) (2004) 1567–1580.
[10] P. Salembier, A. Oliveras, L. Garrido, Antiextensive connected operators for image and sequence processing. IEEE Trans. Image Proc. 7 (4) (1998) 555–570.
[11] M. Vatsa, R. Singh, P. Mitra, A. Noore, Signature verification using static and dynamic features, in: N. Pal, N. Kasabov, R. Mudi, S. Pal, S. Parui (Eds.), Neural Information Processing, Lecture Notes in Computer Science, vol. 3316, Springer Berlin Heidelberg, 2004, pp. 350–355.
[12] S. Kumarasamy, S. Ong, K. Tan, Robust contour reconstruction of red blood cells and parasites in the automated identification of the stages of malarial infection. Mach. Vision Appl. 22 (3) (2011) 461–469.
[13] S.K. Nayar, R.M. Bolle, Reflectance based object recognition, Int. J. Comput. Vision 17 (3) (1996) 219–240.
[14] M.H. Chen, A fast algorithm to calculate the euler number for binary images. Pattern Recogn. Lett. 8 (5) (1988) 195–297.
[15] J.L. Díaz-De-León, J.H. Sossa-Azuela, On the computation of the euler number of a binary object. Pattern Recogn. 29 (3) (1996) 471–476.
[16] S. Di Zenzo, L. Cinque, S. Levialdi, Run-based algorithms for binary image analysis and processing, IEEE Trans. Pattern Anal. Mach. Intell. 18 (1) (1996) 83–89.
[17] L. He, Y. Chao, K. Suzuki, An algorithm for connected-component labeling, hole labeling and euler number computing, J. Comput. Sci. Technol. 28 (3) (2013) 468–478.
[18] T. Acharya, B. Bhattacharya, A. Bishnu, M. Kundu, C. Murthy, Computing the euler number of a binary image, 2006 (URL: http://www.google.com/patents/US7027649). US Patent 7,027,649.
[19] W.K. Pratt, Digital Image Processing: PIKS Inside, third ed., John Wiley & Sons, Inc., New York, NY, USA, 2001.
[20] P. Salembier, J. Serra, Flat zones filtering, connected operators, and filters by reconstruction. Trans. Imag. Proc. 4 (8) (1995) 1153–1160.
[21] L. Garrido, P. Salembier, D. Garcia, Extensive operators in partition lattices for image sequence analysis, in: Signal Processing, vol. 66 (2). Elsevier North-Holland, Inc. 1998, pp. 157–180.
[22] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions. Image Vis. Comput. 22 (10) (2004) 761–767.
[23] E. Carlinet, T. Géraud, A comparison of many max-tree computation algorithms, in: Proc. of the Intl. Symp. on Mathematical Morphology (ISMM), LNCS, vol. 7883, Springer, 2013, pp. 73–85.
[24] E.R. Urbach, J.B.T.M. Roerdink, M.H.F. Wilkinson, Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images. IEEE Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 272–285.
[25] S.B. Gray, Local properties of binary images in two dimensions. IEEE Trans. Comput. 20 (5) (1971) 551–561.
[26] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Syst. Man Cyber. 9 (1) (1979) 62–66.
[27] T. Pun, A new method for grey-level picture thresholding using the entropy of the histogram. Signal Proc. 2 (3) (1980) 223–237.