Meaningful Disjoint Level Lines Selection
Yongchao Xu, Edwin Carlinet, Thierry Géraud, Laurent Najman

To cite this version:

HAL Id: hal-01082342
https://hal.archives-ouvertes.fr/hal-01082342
Submitted on 13 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
MEANINGFUL DISJOINT LEVEL LINES SELECTION

Yongchao Xu\textsuperscript{1,2}, Edwin Carlinet\textsuperscript{1,2}

\textsuperscript{1} EPITA Research and Development Laboratory (LRDE)
14-16, rue Voltaire
FR-94276 Le Kremlin-Bicêtre, France

\textsuperscript{2} Université Paris-Est,
Laboratoire d’Informatique Gaspard-Monge (LIGM),
A3SI, ESIEE Paris, Cité Descartes, BP 99
FR-93162 Noisy-le-Grand, France

ABSTRACT

Many methods based on the morphological notion of shapes (i.e., connected components of level sets) have been proved to be very efficient in shape recognition and shape analysis. The inclusion relationship of the level lines (boundaries of level sets) forms the tree of shapes, a tree-based image representation with a high potential. Numerous applications using this tree representation have been proposed. In this article, we propose an efficient algorithm that extracts a set of disjoint level lines in the image. These selected level lines yield a simplified image with clean contours, which provides an intuitive idea about the main structure of the tree of shapes. Besides, we obtain a saliency map without transition problems around the contours by weighting level lines with their significance. Experimental results demonstrate the efficiency and usefulness of our method.

Index Terms— Mathematical morphology, Level lines, Tree of shapes, Saliency map, Connected filters.

1. INTRODUCTION

Many authors \cite{1,2,3} claim that significant contours of objects coincide with segments of level lines in images. Each level line is the contour of a level set or a shape which is a connected component with holes filled. The set of shapes equipped with the inclusion relationship allows to represent an image by a tree structure, called tree of shapes \cite{4} or topographic map \cite{11}. Numerous applications using the tree of shapes have been proposed: image filtering (grain filter \cite{5}), our proposed morphological shaping \cite{6}, image simplification \cite{2,7,8}, image segmentation \cite{9,10,11}, scenery images analysis \cite{12,13}, object recognition \cite{14}, and texture indexing \cite{15}. In spite of all these abundant applications, the strong potential of the tree of shapes is still under-exploited.

The number of connected components in the tree of shapes is about as large as the number of pixels in the image. Hence, it is not straightforward to know \textit{a priori} the structure of the tree (as shown in Fig. \textsuperscript{1}). However, this information is fundamental for a deep tree analysis and developing tree-based methods. In this paper, we propose an efficient algorithm that selects a set of disjoint level lines of the tree of shapes from meaningful ones to meaningless ones. These level lines represent the main structure of the original tree, which can now be easily perceived from the contour clean image reconstructed from those level lines. This is the main contribution of this paper. An example of selected level lines is given in Fig. \textsuperscript{1} In (b), too many level lines are preserved that makes it difficult to perceive the main structure of the image. In (c) and (d), the bottom-up and top-down level lines selection do not retrieve correctly the image geometry, whereas the selection by meaningfulness in (e) reveals the main structure of the image.

There exist many similar applications relying on the ex-
traction of level lines from the tree of shapes to achieve image filtering, simplification, and segmentation tasks. Salembier and Wilkinson [16] give a detailed review of the tree-based filtering strategies. The works in [9, 10, 17, 2, 11, 7, 18, 8] are closest to what we propose here. In [9], the author proposes a segmentation algorithm that selects the perceptible level lines matching some criteria, e.g., number of T-junctions, compactness, and contrast. In [10], the authors remove the level lines that enclose a region similar to its parent with a histogram.

2. BACKGROUND

2.1. The Tree of Shapes

Let \( f \) be an image defined on a domain \( \Omega \) and with values on an ordered set \( V \) (typically \( \mathbb{R} \) or \( \mathbb{Z} \)). For any \( \lambda \in V \), the upper level sets \( \lambda^{-} \) and lower level sets \( \lambda^{+} \) of an image \( f \) are respectively defined by \( \lambda^{-}(f) = \{ p \in \Omega \mid f(p) \geq \lambda \} \) and \( \lambda^{+}(f) = \{ p \in \Omega \mid f(p) \leq \lambda \} \). Both upper and lower level sets have a natural inclusion structure: \( \forall \lambda_1 \leq \lambda_2, \lambda^{-}_{\lambda_1} \supseteq \lambda^{-}_{\lambda_2} \) and \( \lambda^{+}_{\lambda_1} \subseteq \lambda^{+}_{\lambda_2} \), which leads to two distinct and dual representations of an image, the Max-tree and the Min-tree [21]. The tree of shapes is a fusion of the Max-tree and Min-tree via the notion of shapes [4]. A shape is defined as a connected component of an upper or lower level set with its holes filled in. Thanks to the inclusion relationship of both kinds of level sets, the set of shapes can be structured into a tree structure, called the tree of shapes. This tree features several interesting properties: it is invariant to contrast changes and forms a self-dual, non-redundant, and complete representation of an image. Furthermore, such a tree inherently embeds a morphological scale-space (the parent of a node/shape is a larger shape) with respect to the theory of multi-scale analysis [22]. An example of these trees is depicted in Fig. 2.

2.2. Tree computation and representation

Though many tree of shapes computation algorithms exist [4, 23, 19], we rely on the one proposed in [20] that ensures a worst-case quasi-linear complexity. Even if this paper does not aim at explaining this algorithm since it has already been fully described by its authors, we still need to remind that it does not work directly on the domain \( \Omega \) but on a simplicial version of the 2D discrete grid: the Khalimsky grid. We note \( K_\Omega \), the domain \( \Omega \) immersed on this grid. In Fig. 3, the original points of the image are the 2-faces, the boundaries are materialized with 0-faces and 1-faces. Géraud at al. [20]'s algorithm ensures that shapes are open connected sets (e.g., the purple shape in Fig. 3) and that levels lines (shape's borders) are composed of 0-faces and 1-faces only (e.g., the dark curve in Fig. 3). Given a connected open set \( A \), we note \( \bar{A} \) the closure of \( A \) and \( \partial A \) the frontier of \( A \), i.e., the level line associated to the shape \( A \).

Using the tree representation from [20], a node is represented by a single pixel (2-face) called the canonical element. The tree is encoded through an image \( parent : K_\Omega \rightarrow K_\Omega \) that states the parent-child relationship between nodes. In parent, each non-canonical element point is attached to the canonical element representing the node it belongs to. In the following, we denote by \( getCanonical : K_\Omega \rightarrow K_\Omega \), the routine that returns the canonical element of each point in the image. We also rely on a sorted vector of pixels \( S \) such that processing \( S \) in the direct order is a top-down traversal of the tree and the reverse order is a bottom-up traversal.

Fig. 2: An example of the tree-based image representations relying on threshold decomposition.

Fig. 3: An example of Khalimsky's grid by materializing the points in a 2D image with 0-faces (blue disks), 1-faces (green strips), and 2-faces (red squares).
3. DISJOINT LEVEL LINES SELECTION

In natural images, there are about as many level lines as pixels, which make the tree of shapes difficult to be visualized for further analysis. In fact, most shapes only differ from a few pixels when climbing the tree of shapes, thus many level lines overlap. In this section, we propose an efficient algorithm that allows to select a set of disjoint level lines whatever the selection ordering is.

3.1. Disjoint nodes computation

As shortly reviewed in Section 2.2, a shape \( A \) is an open connected set in \( K_O \) and the corresponding level line is the border \( \partial A \) composed of 0 and 1-faces only. In other words, for any shape \( A \), we aim at computing the set of shapes \( B \) such that for each \( B \in B \), either \( A \subset B \wedge \partial A \cap \partial B \neq \emptyset \) or \( B \subset A \wedge \partial A \cap \partial B \neq \emptyset \). Actually, these relations are dual. We can focus on computing only the first one: the set of shapes in the parenthesis sharing a border with \( \partial A \). Let \( SES(A) \) denotes the Smallest Enclosing Shape of \( A \). Then, \( SES(A) \) is the smallest open set that totally includes the border of \( A \). Since \( SES(A) \) is open, its border does not belong to \( SES(A) \) and thus, \( \partial SES(A) \cap \partial A = \emptyset \). In other words, the level line \( \partial SES(A) \) is the first (w.r.t the inclusion relationship) disjoint level line of \( \partial A \). On the contrary, for each \( B \in \{ x \in SES(A) \} \) (where \( x \rightarrow y \) is the path from \( x \) to \( y \) in the tree), the borders of \( A \) and \( B \) intersect.

Computing \( SES : K_O \rightarrow K_O \) for each shape is a standard incremental attribute computation. This is given in Algorithm 1. For each 2-face \( x \) in \( K_O \), we retain the lowest node which contains \( x \) (i.e., the closure of \( x \)). This is straightforward if the depth of each point in the tree is given. When processing each 0 and 1-face \( e \) connecting to \( x \), we have two cases: either \( e \) is an internal edge or an external edge of the shape containing \( x \). In the former case, we have \( depth(e) \geq \text{depth}(x) \), thus \( e \) is ignored. In the latter case, we have \( depth(e) < \text{depth}(x) \). The node that contains \( e \) is a potential smallest enclosing shape. Since \( A = \cup \{ x \mid x \in A \} \), we propagate the SES from the bottom to the top of the tree incrementally, and eventually we get the SES for each node.

Algorithm 1: Computation of SES.

\begin{verbatim}
1 COMPUTE_SES(parent, S, depth)
2 foreach x in K_O do SES(x) ← getCanonical(x)
3 foreach 2-face x in reverse order of S do
4     foreach 0 and 1-face e in x do
5         if depth(e) < depth(SES(x)) then
6             q ← parent(x)
7             if depth(SES(x)) < depth(SES(q)) then
8                 SES(q) ← SES(x)
9         return SES
\end{verbatim}

3.2. Final disjoint nodes selection

Based on the Smallest Enclosing Shapes, we are able to select a set of nodes in an arbitrary order such that any pair of nodes have disjoint borders. In this paper, we consider three selection strategies: top-down, bottom-up and another one based on a meaningfulness criterion. Let \( \text{Order} \) denotes the sequence of shapes sorted by one of the previous strategy. \( \text{Order} \) is actually an array of canonical elements. The final disjoint nodes selection is given in Algorithm 2, which proceeds as follows. Every shape has a three-state status: \( \text{Null} \), the shape has not yet been processed, it is a candidate shape; \( \text{Active} \), the shape has been selected; \( \text{Unactive} \), the shape has been dismissed. At first, every shape has the \( \text{Null} \) status and we process the candidate shapes in the order given by \( \text{Order} \). Given a candidate shape \( A \), we have to check that the border does not intersect with an already \( \text{Active} \) shape, that is: \( A \) has not been marked \( \text{Unactive} \) by a sub-shape and that any shape in \( A \hookrightarrow SES(A) \) is not \( \text{Active} \). If both conditions are fulfilled, \( A \) is selected, marked \( \text{Active} \) and we dismiss every shape in \( A \hookrightarrow SES(A) \) since they intersect \( A \)'s border.

The smallest enclosing shapes computation described in Algorithm 1 has a linear complexity. For the disjoint level lines selection shown in Algorithm 2, the bottom-up selection has also a linear complexity, whereas the other two strategies have a worst case \( O(N^2) \) complexity, where \( N \) is the number of shapes. We have implemented the proposed method using our C++ image processing library [24]. Processing a 640 × 960 pixels image takes about 3s on a regular PC station, where the tree computation takes half of the time.

4. EXPERIMENTAL RESULTS

We have experienced three different orders for disjoint level lines selection: top-down (from the root to the leaves), bottom-up (from the leaves to the root), and by meaningfulness, e.g., the average of the gradient’s magnitude along the level lines. An example of such disjoint level lines selec-
Fig. 4: An illustration of the disjoint level lines selection with different orders. From top to down: original image, bottom-up, top-down, average of gradient’s magnitude ordering. Left: grayscale image; Right: corresponding randomly colorized image.

Fig. 5: An illustration of the disjoint level lines selection applied to an image of bee board.

g. Input image f.
(b) Simplified image f’.
c. Random color of f.
(d) Random color of f’.
(e) Segmented objects.
(f) Saliency map.

Fig. 6: An illustration of the disjoint level lines selection applied to an image of license plate.

(a) Input image f.
(b) Simplified image f’.
c. Saliency map m.
(d) Thresholding of m.

5. CONCLUSION AND PERSPECTIVES

In this paper, we have presented an efficient algorithm that selects disjoint level lines from a tree of shapes. The selection process is based on the incremental computation of the smallest enclosing shapes. The method extracts a set of disjoint level lines from meaningful ones to meaningless ones, that briefly reflect the main structure of the tree of shapes. Not only does it allow to analyse the tree structure, but it can also be a start for more advanced tree-based processing methods. Indeed, we have shown that it provides a relevant simplification of the images where the meaningful objects are separated one from each other and also provides a saliency map without transition problem around image contours. However, in natural images, the meaningful level lines are not all strictly separated, so in the future, we would like to relax this condition and tolerate a small superposition between the selected level lines. As a second future work, we aim at applying level lines shortening that yields an image curvature microscope before extracting disjoint level lines. Another major perspective is to extend this method to 3D images and the tree of shapes for color images [26, 27].
References


