J. Vis. Commun. Image R. 25 (2014) 435–444

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

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

Computation of level lines of 4-/8-connectedness [☆]

Yuqing Song

Tianjin University of Technology and Education, 1310 Dagu South Road, Hexi District, Tianjin 300222, China

ARTICLE INFO

Article history:
Received 16 September 2013
Accepted 3 December 2013
Available online 12 December 2013

Keywords:
Level line
Jordan boundary
Level set
Image representation
Level line tree
Square grid
Connectedness
Topdown algorithm

ABSTRACT

We present a topdown algorithm to compute level line trees of 4-/8-connectedness. As a boundary of a level set component, a level line of an image is a Jordan boundary of intensity value on instant interior greater/less than on instant exterior. The interior of a Jordan boundary assumes 4-connectedness and the exterior 8-connectedness, or the inverse. All level lines form a tree structure. The running time of the algorithm is $O(n+t)$, where n is the size of the input image and t is the total length of all level lines. The efficiency of the algorithm is illustrated by experiments.

© 2013 Elsevier Inc. All rights reserved.

1. Introduction

Image representation is of fundamental importance to many image processing and computer vision applications. An image is generally represented in terms of properties of objects contained within the image. Different representation techniques are developed for different purposes, including Polar Harmonic Transforms [1], Non-negative Matrix Factorization [2], Fourier Transforms [3], Wavelet theory [4], Ridgelet Transforms [5], Scale-Space [6], and Level Set Transforms [7]. As compared with other approaches, level set methods yield a complete, contrast invariant representation. In addition level sets make a natural shape hierarchy and can handle cavities, concavities, convolution, and splitting/merging.

Monasse and Guichard [8] argued that for human vision system the boundaries of the level sets are perceptually more important than the level sets themselves. Their algorithm, Fast Level Line Transform (FLLT), computes a tree of level lines, where a shape is a level set component with holes filled, and the boundary of a shape is called a level line. FLLT takes a region growing approach to build the trees of connected components of lower/upper level sets. The two trees are then fused to build a level line tree. An improved version, Fast Level Set Transform (FLST), can be found in [9]. FLST has two modules *flst* and *flst_boundary* to decompose an image into a shape tree and to compute the shape boundaries, respectively. Both FLLT and FLST require an average $O(n \log n)$ time for an image of n points. Caselles et al. recently published an analysis of the link between component tree and tree of shapes, and extended FLLT for the general case of multidimensional images [7]. Level lines offer a powerful framework for many image processing applications, including object disocclusion [10], image denoising [11], and segmentation [12]. We refer to [13] for a detailed review of level line tree and other related tree ordered structures in image processing and computer graphics.

In discrete images we have two notions of connectedness on square grid, 4- and 8-connectedness, according to the number of neighbors of the pixels. It is common to use 4-connectedness for upper level sets and 8-connectedness for lower level sets, or the inverse [8]. Another possibility to consider is a hexagonal lattice. Our previous work [14] reported a topdown algorithm to compute 6-connected level line trees on hexagonal grid. The algorithm begins with image boundary, traversing its private region to find all its child level lines. This process is repeated at each internal node to find its children. The algorithm is optimal, using $O(n+t)$ time, where n is the image size and t is the total length of all level lines.

This paper extends the topdown level line computation algorithm to square grid, and makes the following main contributions:

(1) When we migrate from hexagonal grid to square grid, the main difficulty comes from the fact that the upper and lower level sets use different connectedness. To deal with the asymmetry, we define the *instant interior*/*exterior* of level lines based on the connectedness used by the level sets, and develop new mathematical techniques to analyze the properties of the 4-/8-connected level lines. These techniques can be applied to other discrete image processing tasks as well.

[☆] This paper is partially supported by Natural Science Foundation of China under contracts No. 61070112 and No. 61070116.

E-mail address: yqsong7@hotmail.com

1047-3203/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jvcir.2013.12.007

(2) The algorithm extension is not trivial. For example, a 6-connected child level line is always adjacent to the private region of its parent line (see Proposition 2 in [14]). But the example in Fig. 1 tells a different story on square grid. In the example the line F is isolated from the private region of its parent by its siblings. On hexagonal grid, for a given parent line, the child lines are discovered by searching the private region of the parent; but on square grid, we need to keep track of the visited child lines and locate their adjacent siblings.

The rest of the paper is organized as follows. We prepare with definitions and properties in Section 2, and present the algorithm in Section 3. Section 4 is an experimental report. Section 5 concludes. To ease reading, we defer the proofs of lemmas and propositions to the Annex.

2. Definitions and properties

In this paper we use 4-connectedness for upper level sets and 8-connectedness for lower level sets. The introduced algorithm generates the level lines as the boundaries of the level sets of such connectedness specification. By simply being applied to the negative of the input image, the algorithm can also compute the level lines as the boundaries of 8-connected upper level sets or 4-connected lower level sets.

Given an image f: $\Omega = \{0,1,\ldots,w-1\} \times \{0,1,\ldots,h-1\} \to \{0,1,\ldots,d-1\}$, we call upper level set X_λ of value λ and lower level set X^μ of value μ the subsets of Ω:

$$\chi_\lambda = \{x \in \Omega | f(x) \geq \lambda\}, \chi^\mu = \{x \in \Omega | f(x) \leq \mu\}.$$

The connected components of the upper/lower level sets are nested and can be represented into a tree structure, called the component tree (see Fig. 2). Given a component in the tree, its border is a union of Jordan curves, where one curve is outer boundary and the others are inner boundaries representing the holes of the component. Thanks to Jordan theorem ([15]), a closed Jordan curve divides the plane into two connected components; the bounded one is the interior of the curve and the other one, not bounded, is the exterior.

Each boundary of a level set component is called a *level line*. A level line l_1 is called "included" by another level line l_2 if the interior of l_1 is included in the interior of l_2. We retrieve the boundaries (both outer boundaries and holes) of upper level set components and organize them into an inclusion tree, called the *positive level line tree* (Fig. 3(a)). The boundaries of the lower level set components are represented by the *negative level line tree* (Fig. 3(b)). The positive and negative trees of an image differ at level lines meeting the image boundary; inside level lines appear the same in both trees. For the example in Fig. 3, the level lines x and y meet the image boundary; u and v are inside level lines which do not meet the image boundary. The positive and negative trees share the common level lines u and v, and differ at x and y. FLST [9] fuses the two trees together such that each level line meeting the image boundary has an area equal to or less than half the image. Our strategy is to keep the bi-tree structure and let the ambiguity solved by high level semantics [14].

In this paper we use 4-connectedness for upper level sets and 8-connectedness for lower level sets. With this connectedness specification, the outer boundary of an upper level set component is a *48-Jordan boundary*[1] of intensity value on *instant interior* greater than on *instant exterior*, and an inner boundary of an upper level set component is an *84-Jordan boundary* of intensity value on instant interior less than on instant exterior. See Fig. 4. Boundaries of the two types are called *positive* and *negative* level lines, respectively. Jordan boundary and related concepts are introduced in Section 2.1; level lines and their properties are elaborated in Section 2.2.

2.1. Jordan boundary

In this paper we always let $\varpi = 4$ or 8, and $\underline{\varpi} = 12 - \varpi$. In the digital plane Z^2, the complement of a pixel set A is denoted as A^c. We use the following terms and notations in our discussion.

1. Regarding connectedness:
   1.1 Two pixels (x_1,y_1) and (x_2,y_2) are called 4-*adjacent* or 4-*neighbors* if $|x_1 - x_2| + |y_1 - y_2| = 1$. They are called 8-*adjacent* or 8-*neighbors* if $|x_1 - x_2| + |y_1 - y_2| = 1$ or 2, i.e., they are 4-neighbors ($|x_1 - x_2| + |y_1 - y_2| = 1$) or diagonal neighbors ($|x_1 - x_2| + |y_1 - y_2| = 2$).
   1.2 An edgel is the common edge shared by two 4-adjacent pixels. Each edgel is represented as (p,q), where p and q are 4-neighbors and they are called the *immediate interior pixel* (*IIP*) and the *immediate exterior pixel* (*IEP*), respectively. Each pixel p has 4 edgels around it in a counter clockwise order: (p,q_0), (p,q_1), (p,q_2), (p,q_3). See Fig. 5. Edgels (p,q) and (q,p) are *inverse* to each other. Inverse edgels have inverse directions. The *inverse* of an edgel $e = (p,q)$ is denoted as e^{-1}, i.e., $(p,q)^{-1} = (q,p)$.
   1.3 A ϖ-*path* is a sequence of pixels $\langle p_1,p_2,\ldots,p_k\rangle$ in the plane such that every two consecutive pixels, p_i and p_{i+1} (for $0 < i < k$), in the sequence are ϖ-adjacent.
   1.4 A set of pixels is called ϖ-*connected* if for every two pixels in the set, there is a ϖ-path in the set connecting them.
   1.5 A ϖ-*component* of a pixel set is a ϖ-connected component of the set.
2. Regarding regions:
   2.1. A *region* is a finite and non-empty set of pixels in the plane.
   2.2. A ϖ-*region* is a ϖ-connected region.
   2.3. An sϖ-*region* is a simply-connected ϖ-region, i.e., it is a ϖ-region such that its complement is $\underline{\varpi}$-connected.
   2.4. A *hole* of a ϖ-region X is a bounded $\underline{\varpi}$-component of X^c. The set of holes is denoted as $Ho_\varpi(X)$. See Fig. 6(a).
   2.5. The *surrounding set* of a ϖ-region X, denoted as $Su_\varpi(X)$, is the unbounded $\underline{\varpi}$-component of X^c. See Fig. 6(b).
   2.6. The ϖ-*extension* of a ϖ-region X, denoted as $Ex_\varpi(X)$, is the extension of X by filling its holes, i.e., $Ex_\varpi(X)$ is the union of X and its holes. See Fig. 6(c). It is easy to verify that $Su_\varpi(X) = (Ex_\varpi(X))^c$ and that for an sϖ-region X, $Ex_\varpi(X) = X$.

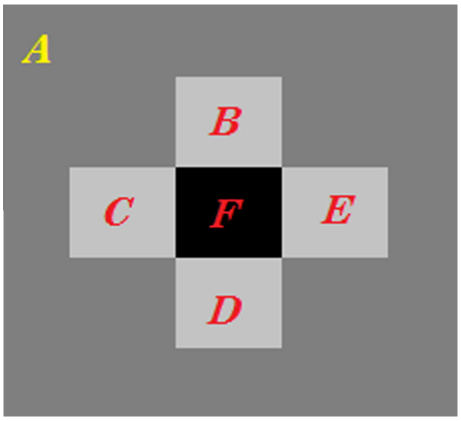

Fig. 1. An example of 4-/8-connected level lines. In the example the parent line A has 5 child lines: B, C, D, E, and F, where the child line F is surrounded by its four siblings.

[1] As to be defined in Section 2.1, a $\varpi\underline{\varpi}$-*Jordan boundary* is a boundary where the interior is ϖ-connected and the exterior is $\underline{\varpi}$-connected.