SHAPE-TREE SEMILATTICES

Variations and Implementation Schemes

Renato Keshet

Hewlett-Packard Labs—Israel
Technion City, Haifa 32000, Israel
renato.keshet@hp.com

Abstract The shape-tree semilattice is a new framework for quasi-self-dual morphological processing, where eroded images have all shapes shrunk in a contrast-invariant way. This approach was recently introduced, and is further investigated here. Apart of reviewing their original definition, different algorithms for computing the shape-tree morphological operators are presented.

Keywords: Complete inf-semilattices, self-dual operators, tree of shapes, fillhole.

1. Introduction

We have very recently introduced a new quasi-self-dual morphological approach for image processing [1]. The resulting flat morphological erosion, for instance, causes all shapes in an image to shrink, regardless their contrast (i.e., regardless to whether the are bright or dark). Motivation and connection to other works are described in [1].

For discrete binary images, the approach yields a set of morphological operators on the so-called *adjacency complete lattice*, which is defined by means of the adjacency tree representation (see Section 2 below). The scheme is generalized to discrete grayscale images by means of the *Tree of Shapes* (ToS), a recently-introduced grayscale image representation [7–10], which can be regarded as a grayscale generalization of the adjacency tree (see Section 3). In the grayscale case, however, the underlying space structure is not that of a complete lattice anymore, but of a complete inf-semilattice.

While the main motivation in the original article was on laying out the theoretical basis of the scheme, and investigating its mathematical soundness, the current article focuses on implementations. Different algorithms for implementing the desired basic operators are investigated; they slightly differ in general, but produce identical results for typical structuring elements.

C. Ronse et al. (eds.), Mathematical Morphology: 40 Years On, 139–148.

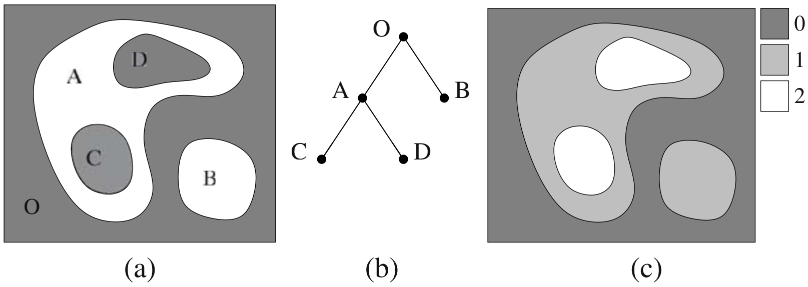

Figure 1. (a) A binary image. Each letter corresponds to a connected component. O is the background component, A and B correspond to the two white connected components, and C and D correspond to the two dark connected components inside A. (b) The adjacency tree of (a). (c) Its unfolding transform.

2. Binary Approach: Adjacency Lattice

The basic idea for our binary approach is to generate a quasi-self-dual complete lattice by using the data in the *adjacency tree*. In [2, page 89, Figure III.10], Serra describes the adjacency tree (which he called *homotopy tree*) as follows. If X is a bounded input binary image in an Euclidean space E, then the root of the adjacency tree is the infinite connected component of X^c. The first level nodes of the tree are those connected components of X that are adjacent to the root. The second level of nodes are the connected components of X^c that are adjacent to the first level of nodes, and so on. See an example in Fig. 1. The adjacency tree was thoroughly studied by Heijmans in [3].

Next, we review a characterization of the adjacency tree, which we presented in [1], and the subsequent derivation of the adjacency lattice. Let E be the 2D grid $\mathbb{Z}^2$, and R be a bounded region in E. Consider the set $\mathcal{P}(R)$ of all subsets of R (i.e., binary images within R). Define the *fillhole* $\phi(X)$ of a bounded binary image $X \in \mathcal{P}(R)$ as the complement of the morphological reconstruction (according to a given connectivity; e.g., either of the well-known 4- or 8-connectivities) of $E - X$ from the marker $E - R$. The fillhole operator is extended to discrete grayscale images by applying it to each level set:

$$\phi(f) = \theta^{-1}\{\phi[\theta_n(f)]\}, \tag{1}$$

where $\theta_n(f) \triangleq \{x \in E | f(x) \geq n\}$ is the level set of f of height n, and $\theta^{-1}\{T_n\} \triangleq \sup\{n \in \mathbb{N} | x \in T_n\}$ consolidates level sets back to a function. The grayscale fillhole operator ϕ is essentially the same as the FILL$(\cdot)$ operation, described in [4, page 208].

If a set or a function is invariant to the fillhole operator, then it is said to be *filled*.