RUMINATIONS ON TARJAN'S UNION-FIND ALGORITHM AND CONNECTED OPERATORS

Thierry Géraud

EPITA Research and Development Laboratory (LRDE)
14-16 rue Voltaire, F-94276 Le Kremlin-Bicêtre, France
Phone: +33 1 53 14 59 47, Fax: +33 1 53 14 59 22
thierry.geraud@lrde.epita.fr

Abstract This papers presents a comprehensive and general form of the Tarjan's union-find algorithm dedicated to connected operators. An interesting feature of this form is to introduce the notion of separated domains. The properties of this form and its flexibility are discussed and highlighted with examples. In particular, we give clues to handle correctly the constraint of domain-disjointness preservation and, as a consequence, we show how we can rely on "union-find" to obtain algorithms for self-dual filters approaches and levelings with a marker function.

Keywords: Union-find algorithm, reconstructions, algebraic openings and closings, domain-disjointness preservation, self-dual filters, levelings.

Introduction

Connected operators have the important property of simplifying images while preserving contours. Several sub-classes of these operators have been formalized having stronger properties [8] and numerous applications have been derived from them, e.g., scale-space creation and feature analysis [17], video compression [14], or segmentation [10]. The behavior of connected operators is to merge most of the flat zones of an input image, thus delivering a partition which is much coarser than the input one. In that context, a relevant approach to implement such operators is to compute from an input image the resulting partition. The Tarjan's Union-Find Algorithm, *union-find* for short, computes a forest of disjoint sets while representing a set by a tree [16]. A connected component of points or a flat zone is thus encoded into a tree; a point becomes a node and a partition is a forest. union-find has been used to implement some connected operators; among them, connected component labeling [2], a watershed transform [6], algebraic closing and opening [19], and component tree

C. Ronse et al. (eds.), Mathematical Morphology: 40 Years On, 105–116.

computation [4, 11]. A tremendous advantage of union-find lies in its *simplicity*. However, the descriptions of morphological operators relying on this algorithm are usually spoiled by the presence of too many implementation details.

This paper intends to provide the image processing community with a simple and general form of union-find, which is highly adaptable to the large class of connected operators. We show that the description of a given operator with union-find is actually straightforward, comprehensive, and takes very few code. We also present how union-find can be used for the connected operators θ which verify a domain disjointness preservation property. Consequently we show that union-find is a simple way to get algorithms for folding induced self-dual filters [5], the inf-semilattice approach to self-dual morphology [3], and levelings defined on two functions [10].

In order to keep implementation details away from algorithmic considerations, we do not address any single optimization issue. Moreover, we do not enter into a comparison between union-find-based algorithms and other approaches; for those subjects, the reader can refer to [13, 7]. We claimed in [1] that our generic C++ image processing library, Olena [12], has been designed so that algorithms can easily be translated into programs while remaining very readable. To sustain this claim, programs given in this paper rely on our library and, thanks to it, they efficiently run on various image structures (signals, 2D and 3D images, graphs) whatever their data types (Boolean, different integer encodings, floating values, etc.)

In the present document we start from the simplest operator expressed with union-find, namely a connected component labeling, in order to bring to the fore the properties of union-find-based algorithms (Section 1). We stress on the notions of domains and of disjointness-preservation and we present a general formulation of union-find (Section 2). In the second part of this document we give a commented catalogue of connected operators with the help of that formulation (Section 3). Last we conclude (Section 4).

1. Practicing on Connected Component Labeling

In union-find, a connected component is described as a tree. At any time of the algorithm computation, each existing component has a canonical element, a root point at the top of the tree. A link between a couple of nodes within a tree is expressed by a parent relationship. A convenient way to handle the notion of "parent" is to consider that parent is an image whose pixel values are points—given a point x, parent[x] is a point—and that a root point is its own parent. Finding the root point of a component recursively goes, starting from a point of this component, from parent to parent until reaching the root point.