Writing Reusable Digital Topology Algorithms in a Generic Image Processing Framework

Roland Levillain\textsuperscript{1,2}, Thierry Géraud\textsuperscript{1,2}, and Laurent Najman\textsuperscript{2}

\textsuperscript{1} EPITA Research and Development Laboratory (LRDE)
14-16, rue Voltaire, FR-94276 Le Kremlin-Bicêtre Cedex, France
\textsuperscript{2} Université Paris-Est, Laboratoire d’Informatique Gaspard-Monge, Équipe A3SI, ESIEE Paris, Cité Descartes, BP 99, FR-93162 Noisy-le-Grand Cedex, France
\{roland.levillain,thierry.geraud\}@lrde.epita.fr, l.najman@esiee.fr

Abstract. Digital Topology software should reflect the generality of the underlying mathematics: mapping the latter to the former requires generality. By designing generic solutions, one can effectively reuse digital topology data structures and algorithms. We propose an image processing framework focused on the Generic Programming paradigm in which an algorithm on the paper can be turned into a single code, written once and usable with various input types. This approach enables users to design and implement new methods at a lower cost, try cross-domain experiments and help generalize results.

1 Introduction

Like Mathematical Morphology (MM), Digital Topology (DT) has many applications in image analysis and processing. Both present sound mathematical foundations to handle many types of discrete images. In fact most methods from Mathematical Morphology or Digital Topology are not tied to a specific context (image type, neighborhood, topology): they are most often described in abstract and general terms. Thus they are not limiting their field of application. However, software packages for MM and DT rarely take (enough) advantage of this generality: an algorithm is sometimes reimplemented for each image and/or each value type, or worse, written for a unique input type. Such implementations are not reusable because of their lack of \textit{genericity}. These limitations often come from the implementation framework, which prohibits a generic design of algorithms. A recent and notable exception is the DGtal project, which proposes Digital Geometry (DG) software tools and algorithms built in a generic C++ framework [1]. Thanks to the Generic Programming (GP) paradigm, provided in particular by the C++ language, one can design and implement generic frameworks. This paradigm is especially well-suited to the field of scientific applications where the efficiency, widespread availability and standardization of C++ are real assets. To this end, we have designed a paradigm dedicated to generic and efficient scientific software [2] and applied the idea of generic algorithms to MM in Image
Lamy suggests to implement digital topology in IP libraries [6]. The proposed solution, applied to the ITK library [7,8] "works for any image dimension". In this paper, we present a framework for the generic implementation of DT methods within the Milena library, working for any image type supporting the required notions (value types, geometric and topological properties, etc.). Such a generic framework requires the definition of concepts from the domain (in particular, of an image) to organize data structures and algorithms, as explained in Sect. 2. Given these concepts it is possible to write generic algorithms, like a homotopic thinning operator making use of various definitions of the notion of simple point. We present a generic definition of such an operator in Sect. 3 and show some illustrations in Sect. 4. Section 5 concludes on the extensibility of this work along different axes: existing algorithms, new data structures and efficiency.

2 Genericity in Image Processing

In order to design a generic framework for image processing, we have previously proposed the following definition of an image [3].

**Definition.** An image $I$ is a function from a domain $D$ to a set of values $V$; the elements of $D$ are called the sites of $I$, while the elements of $V$ are its values.

For the sake of generality, we use the term *site* instead of *point*; e.g. a site could represent a triangle of a surface mesh used as the domain of an image. Classical site sets used as image domains encompass hyperrectangles (boxes) on regular $n$-dimensional grids, graphs and complexes (see Sect. 3).

In the GP paradigm, these essential notions (image, site set, site, value) must be translated into interfaces called *concepts* in Milena (*Image*, *Site_Set*, etc.) [9]. These interfaces contain the list of *services* provided by each type belonging to the concept, as well as its *associated types*. For instance, a type satisfying the *Image* concept must provide a *domain()* routine (to retrieve $D$), as well as a *domain_t* type (i.e. the type of $D$) satisfying the *Site_Set* concept. Concepts act as contracts between providers (types satisfying the concept) and users (algorithms expressing requirements on their inputs and outputs through concepts).

For instance, the *breadth_first_thinning* routine from Algorithm 1.3 expects the type $I$ (of the input image) to fulfill the requirements of the *Image* concept. Likewise *nbh* must be a *Neighborhood*; and *is_simple* and *constraint* must be functions taking a value of arbitrary type and returning a Boolean value (*Function_v2b* concept).

3 Generic Implementation of Digital Topology

Let us consider the example of homotopic skeletonization by thinning. Such an operation can be obtained by the removal of *simple points* (or simple sites