Population library

Vincent Tariel

Abstract

The Population library is dedicated to the quantitative and qualitative analysis of images coming from 2D/3D microscopy with filtering, segmentation, geometrical/physical characterization, visualization and modeling. The licenses are GPL/CeCill license for non-commercial purpose and commercial license otherwise. Written by Vincent Tariel at the beginning of his PhD in 2005 until now, the version 1 was in C inspired by Pink library, the version 2 was in C++ in low-level generic programming influenced by thinking in C++ by Bruce Eckel and the current version is in C++ in high-level generic programming influenced by Modern C++ Design: Generic Programming and Design Patterns Applied by Andrei Alexandrescu. Its integration in the Caméléon language allows the democratization of its utilization such that users can create a data-flow as a Lego game with a calibration in real time.

1 Innovation

Population is a high-level generic programming library written in C++, based on new programming theories given a large collection of efficient and versatile algorithms to deal with the image complexity.

1.1 Around function concept

The aim of this theory is to increase the programming productivity for the implementation of algorithms working on functions with a direct link.
Concept/model programming: in the traditional way of programming, we implement an algorithm for a specific data. For an algorithm working with various data, we iterate this implementation for each one. Nowadays, a new way of programming, more mathematical, concept/model programming, emerges where the thinking can be summarized in this assertion: "whatever the data-types with some properties, I can process/define that". In the first part of the assertion "whatever the data-types having these properties", we extract the underlying requirements of a data collection by removing any dependence on specific data with which it might originally have been connected. In the second part of this assertion, "I can process/define that", we implement some generic procedures/classes working whatever the data having the required properties. Without sacrificing efficiency in C++, this programming makes code writing much easier and faster for people familiar with mathematics, but it is not user-friendly. The code is extremely compressed: each statement expresses a great deal of information.
Function concept: the idea of the function, f, is that the input quantity completely determines the output quantity. The input quantity belongs to a space E and the output to the

space F:

$$f : \mathcal{D} \subset E \to F \tag{1}$$

$$x \mapsto y = f(x) \tag{2}$$

In computer, the evaluation of the output element from an input element can be done:

1. with a symbolic relation for instance, for a function equal to $x \mapsto x^2$, the output element is evaluated by the multiplication of the input element by itself,
2. with a direct relation for instance, for a function representing by a table, the output element is the value in the cell located by the input element.

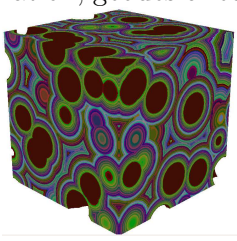
A function with a direct relation as table, image, graph, video, sound signal, plays on important role in a wide range of applications. The function concept is an abstract data representing this collection of data. It is a bridge between mathematical properties on the set E and F and the equivalent programming requirements. For instance, for the erosion algorithm, in mathematics, the space E is a topological and finite space and the space F is a complete lattices. In programming, the concept E has two associated types, neighborhood iterator and total iterator and the concept F has one valid expression, min. Thanks to this concept, I implemented a wide range of generic procedures working on any models of the function concept. By adding your own models, you can directly call theirs.

1.2 Around seeded region growing

History: Introduced by Adams and Bishop in 1994, many fields in computer science, stereovision, mathematical morphology, use algorithm which principle is Region Growing. This method consists in initializing each region with a seed and then iterate pixel by pixel growth controlling by a metric until convergence. In the above figure, the region growing is the watershed transformation controlled by the gradient magnitude of the original image:

Formalism: The implementation of each algorithm can be daunting since many data have to interact. My contribution is the introduction of a conceptual framework with a modern architecture design for the localization and the organization of the region growing. This architecture allows the implementation in few lines of code of optimized algorithms for the function concept. The aim is to let spend more time focusing on the specific logic of the algorithm rather than the implementation/optimization.

Algorithms: Thanks to this work, I implemented a wide range of classical region growing algorithms: Voronoi tessellation, cluster to label, regional minima, distance function, watershed transformation, geodesic reconstruction and Adam's algorithm and new ones: quasi-euclidean



distance , adaptative meshing for phase-field (see figure 1) and simulated annealing reconstruction.

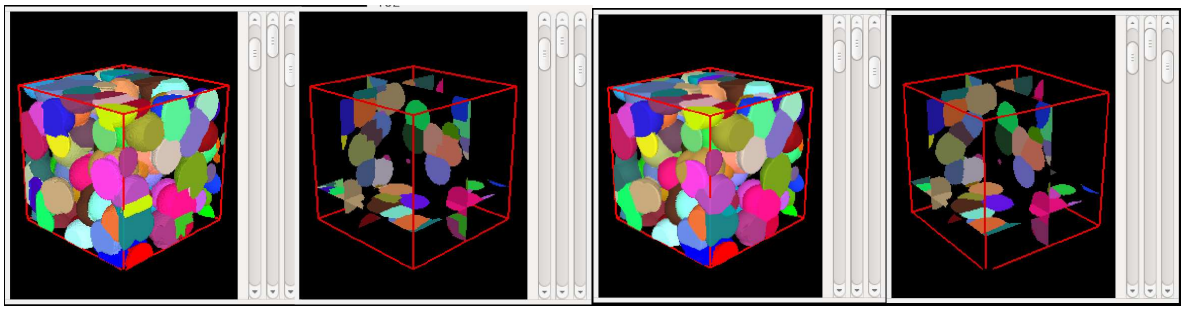
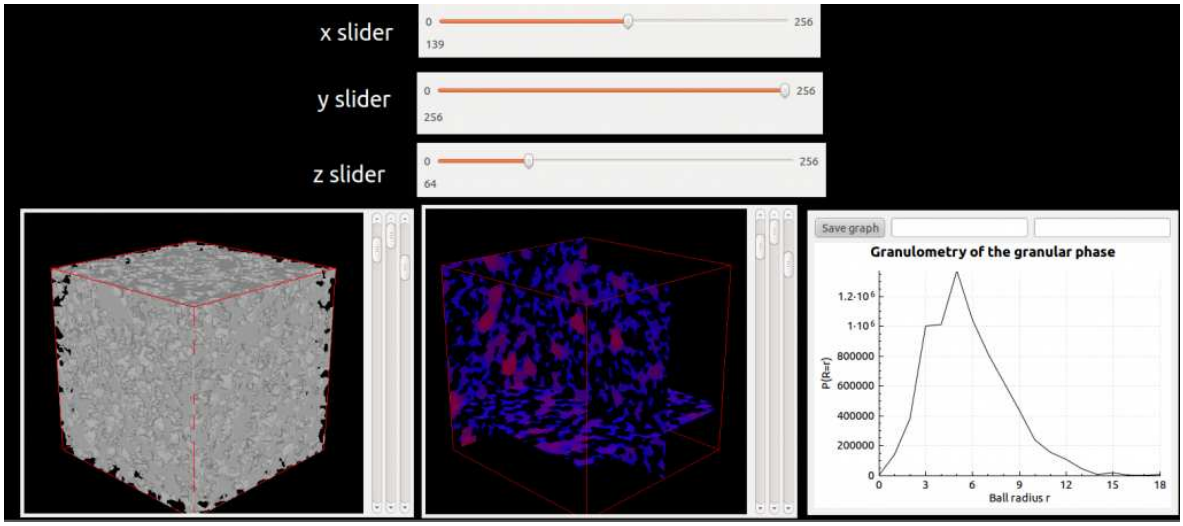


Figure 1: left figure: decomposition in elementary components with L. Vincent's procedure, right figure: regularization of the previous decomposition with Allen-Cahn equation for a minimal surface criteria

2 Some applications

This library is already used by external users. For instance, Jean-François Bruchon says: "I am a PhD student in Geotechnics and Imagery & Materials groups in the Laboratoire Navier at Ecole des Ponts ParisTech. I'm working on Hydro-Mechanical couplings in granular materials with a X-ray computed tomography approach. I am using Caméléon and Population in my research work to analyze images obtained with the X-Ray CT. Why ? Because Population owns a lot of tools to filter, segment and characterize any images. My images are three-dimensional and quite big (around 3GB). Tools of Population work perfectly and quickly on 3D images compared with other softwares. An other advantage is the manipulation and the combination of tools which become very easy with the macro-programming. Now with Caméléon I can build easily many tests to adjust the parameters of operators without time consuming. Caméléon and Population are open source and the support is responsive. I advise these softwares for all these reasons."
A screenshot of his work:



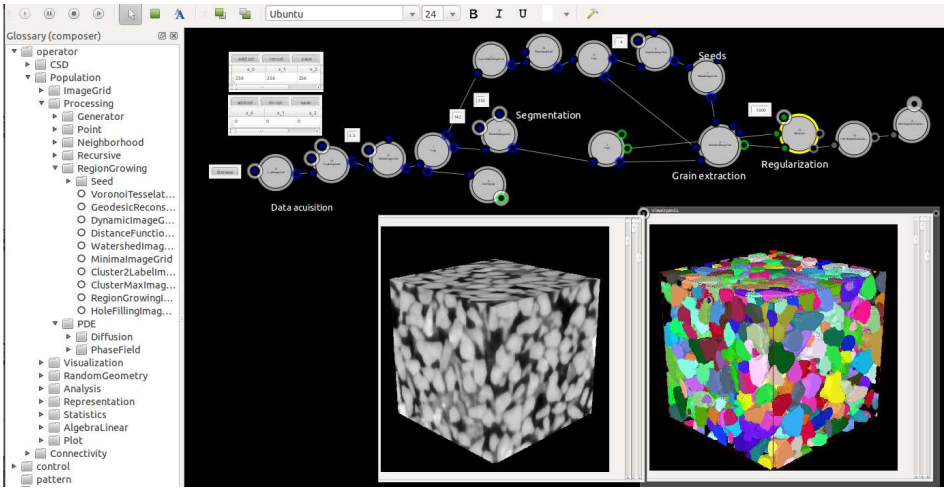


Figure 2: data-flow prototype

3 Productivity in complexity

Since programming problems become more and more complex, our vision is that development should be processed at two scales: micro- and macro-programming. At micro-level, the relevant paradigm is how things happen. We choose the C++ language, a popular imperative language. At macro-level, the relevant paradigm is how things connect. With O. Cugnon de Sevricourt, we develop our own work-flow language called Caméléon following a user-centered paradigm with these key components:

- editor: you can create a work-flow process with elementary with a low technical skill in an ergonomic and intutive interface. The aim is to let you spend more time focusing on the specific logic of your core business rather than the implementation,
- virtual machine: you can drive the work-flow execution for a construction and a calibration in real time,
- exporter: you can share your work-flow in term of deliverable application,
- development kit: technical experts can integrate new elementary bricks in the Caméléon language to create your own business language.

The integration of the Population library in this language is the Population software (see figure 2). Specialists and users of image processing can prototype an image processing workflow for their specific problems, for instance, a geologist analyzing rocks by X-ray tomography. We aim to create a community of end-users and to integrate new libraies by peer production to address the large spectrum of image processing.