3rd International Conference on Tomography of Materials and Structures
Lund, Sweden, 26-30 June 2017, ICTMS2017-212

Analyzing tomography data with Python and the scikit-image library

Emmanuelle Gouillart [*1], Juan Nunez-Iglesias[2], François Boulogne[3], and Stéfan van der Walt[4]

[1]Surface du Verre et Interfaces, UMR 125 CNRS/Saint-Gobain, Aubervilliers, France
[2]Victorian Life Sciences Computation Initiative, University of Melbourne, Australia
[3]Laboratoire de Physique des Solides, Orsay, France
[4]Division of Applied Mathematics, Stellenbosch University, South Africa

Keywords: scikit-image, Python, image processing library

Summary: `scikit-image` is an open-source image processing toolkit for the Python language that is compatible with 2-D and 3-D images. The toolkit exposes a simple programming interfaces and combines a gentle learning curve, versatile image processing capabilities, and the scalable performance required for the analysis of X-ray imaging data. `scikit-image` users also benefit from the rich scientific Python ecosystem.

1. Introduction

The acquisition time of synchrotron tomography images has decreased dramatically over the last decade. However, the time subsequently spent in processing the images has not decreased as much, so that the outcome of a tomography run often takes months to be transformed into scientific results. Often, the sequence of image processing operations is not known beforehand, and trial and error phases are required. Therefore, image processing tools need to offer at the same time flexibility of use, a variety of algorithms, and efficient implementations.

Several software applications and libraries are available to tomography users to process their images. ImageJ is a popular general-purpose tool, thanks to its intuitive menus and graphical tools, and the wealth of available plugins. Alternatively, the use of a programming language gives finer control, better reproducibility, and more complex analysis possibilities, provided classical algorithms can be called from libraries.

`scikit-image` [1, 2] is a general-purpose image processing library for the Python language, and a component of the ecosystem of Python scientific modules. The library is open-source and free of charge. Most of `scikit-image` is compatible with both 2-D and 3-D images, so that it can be used for a large number of imaging modalities. We explain here how `scikit-image` can be used for processing X-ray tomography images.

2. Overwiew of scikit-image

`scikit-image` is a library of the Scientific Python ecosystem. Therefore, users write Python code, in which image processing operations are `scikit-image` functions operating on images. Images are manipulated as numerical arrays of the `NumPy` module, which are a common object to numerous signal processing or visualization librariesi (which also share a similar programming interface and documentation style). Thanks to this rich ecosystem, the package can be used either as main data processing tool, or as a brick in a more complex workflow. A large variety of file formats can be opened as numerical arrays (2D, 3D or nD).

In 2017, the number of active users of `scikit-image` is estimated as 20 000, including tomography users in medical imaging, materials science or geoscience. Information about installation and usage can be found on `http://scikit-image.org`. The package is developed by a diverse team of volunteers, including the authors.

3. Processing tomography images

Image processing capabilities - `scikit-image` offers most classical image processing operations, such as exposure and color adjustment, filtering, segmentation, feature extraction, geometric transformations, and

*e-mail: emmanuelle.gouillart@nsup.org

measurements of region characteristics. In addition to common operations, some advanced algorithms are also implemented, a selection of which is illustrated in Fig. 1. For example, several denoising filters are available to reduce the intensity of noise or artifacts, ranging from general-purpose filters (median, bilateral) to those more suited to specific applications (total variation, non-local means, see Fig. 1 (a)).

Performance - Given the large size of tomography datasets, the execution speed of image processing operations is of critical concern. `scikit-image` relies mostly on calls to `NumPy` operations, of which most are performed in optimized compiled code (C or Fortran). Therefore, the performance of `scikit-image` can be close to the one of libraries written in a compiled language such as C++ or Java.

Although the code of `scikit-image` is written for a single core, parallelization of the computing workflow can be achieved in multiple ways. Several tools such as `joblib` and `dask` are available in order to divide the image into chunks, and to apply the same operation on different chunks, distributed over several cores.

Documentation - Since documentation is a critical point for software usability, several kinds of documentation are available for users. All functions are documented using the same standard as other Scientific Python modules. A graphical gallery of examples showcases graphical examples of common image processing operations.

References

[1] S. Van der Walt, et al. scikit-image: image processing in Python, *PeerJ*, 2 (2014), e453.

[2] E. Gouillart, J. Nunez-Iglesias, and S. van der Walt. Analyzing microtomography data with Python and the scikit-image library. *Advanced Structural and Chemical Imaging* (2017), 18.

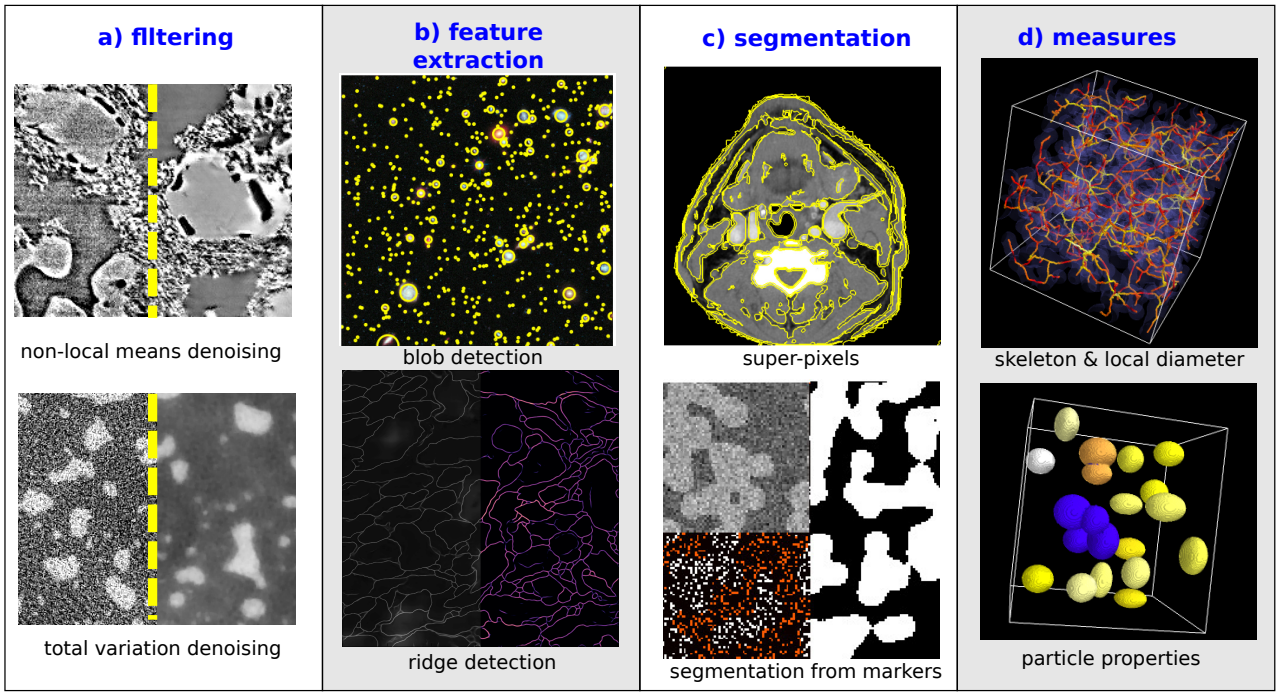

Figure 1: Typical image processing operations with `scikit-image`. a) Filtering - Top: non-local means denoising of an image with a fine-grained texture, acquired by *in situ* synchrotron microtomography during glass melting. Bottom: total-variation denoising of an image with two phases, corresponding to phase separation observed by *in situ* tomography. **b) Feature extraction -** Top: Hubble deep field (NASA), blob detection using the Laplacian of Gaussian method. Bottom: ridge detection using the leading eigenvalue of the Hessian matrix. **c) Segmentation -** Top: super-pixel segmentation of a CT slice of the human head (`https://en.wikipedia.org/wiki/File:Ct-workstation-neck.jpg`), using Felzenszwalb's algorithm. Bottom: random walker segmentation (right) of noisy image (top-left), using histogram-based markers (bottom-left). **d) Measures -** Top: visualization of local diameter (color-coded on the skeleton curve) of an interconnected phase (represented in violet). Bottom: particles color-coded according to their extent.