Journal of Structural Biology 178 (2012) 177–188

Contents lists available at SciVerse ScienceDirect

Journal of Structural Biology

journal homepage: www.elsevier.com/locate/yjsbi

PyTom: A python-based toolbox for localization of macromolecules in cryo-electron tomograms and subtomogram analysis

Thomas Hrabe, Yuxiang Chen, Stefan Pfeffer, Luis Kuhn Cuellar, Ann-Victoria Mangold, Friedrich Förster *

Max-Planck Institute of Biochemistry, Am Klopferspitz 18, D-82152 Martinsried, Germany

ARTICLE INFO

Article history:
Available online 13 December 2011

Keywords:
Cryo-electron tomography
Subtomogram analysis
Image processing
Software toolbox

ABSTRACT

Cryo-electron tomography (CET) is a three-dimensional imaging technique for structural studies of macromolecules under close-to-native conditions. In-depth analysis of macromolecule populations depicted in tomograms requires identification of subtomograms corresponding to putative particles, averaging of subtomograms to enhance their signal, and classification to capture the structural variations among them. Here, we introduce the open-source platform PyTom that unifies standard tomogram processing steps in a python toolbox. For subtomogram averaging, we implemented an adaptive adjustment of scoring and sampling that clearly improves the resolution of averages compared to static strategies. Furthermore, we present a novel stochastic classification method that yields significantly more accurate classification results than two deterministic approaches in simulations. We demonstrate that the PyTom workflow yields faithful results for alignment and classification of simulated and experimental subtomograms of ribosomes and $GroEL_{14}$/$GroEL_{14}GroES_7$, respectively, as well as for the analysis of ribosomal 60S subunits in yeast cell lysate. PyTom enables parallelized processing of large numbers of tomograms, but also provides a convenient, sustainable environment for algorithmic development.

1. Introduction

Cryo-electron tomography (CET) is a method that enables imaging macromolecular assemblies three-dimensionally in their close-to-native settings (Lucic et al., 2005). The resolution of cryo-electron tomograms, i.e., the maximum spatial frequency where signal is distinguishable from noise, is typically limited to 5–10 nm (Grunewald et al., 2003). Nevertheless, significant information content is present in tomograms beyond this limit, which is buried under the noise in the raw data. The sheer amount of data in tomograms (a tomogram typically contains billions of voxels) requires automated analysis to retrieve this information. The low signal-to-noise ratio (SNR) as well as an incomplete sampling of 3D information in the experiment (missing wedge of data in Fourier space) are challenges for the development of computational algorithms (Frangakis and Forster, 2004; Hrabe and Förster, 2011).

A typical task for interpretation of tomograms is the localization of specific macromolecular complexes based on structural models. Template matching is a widely applied computational method for this purpose (Förster et al., 2010; Frangakis et al., 2002): a structural model of the target molecule (template) is correlated with a given tomogram, and areas of high cross-correlation coefficients indicate potential locations of the target. This approach has been used to identify large macromolecular complexes such as ribosomes and Gro-EL chaperones in their close-to-native settings (Beck et al., 2009; Brandt et al., 2010; Ortiz et al., 2006). Quantitative assessment of sensitivity and specificity of macromolecular detection is challenging, and different approaches have been suggested to estimate the accuracy of localization results (Beck et al., 2009; Ortiz et al., 2006; Xu et al., 2011).

Raw subtomograms of specific complexes extracted from a larger tomogram rarely reveal exciting structural details of the complex due to the modest resolution level. Averaging of subtomograms, each containing an identical type of molecule, yields an average with higher SNR and hence improved resolution (Bartesaghi and Subramaniam, 2009; Förster and Hegerl, 2007). Subtomograms of interest can be located in the tomograms manually or automatically using template matching if an initial estimate of the target is available. Key to maximizing the resolution of the averages is the accurate alignment of subtomograms to a common coordinate system. When the resolution approaches the first zero crossing of the contrast-transfer function (CTF), methods for approximately de-convoluting the micrographs may yield further improvements beyond this resolution limit (Fernandez et al., 2006; Zanetti et al., 2009). Furthermore, classification techniques can be used to sort subtomograms according to their intrinsic variations, which yields an increase in resolution for the different classes compared to the overall average.

Different software tools have been proposed for subtomogram localization (Förster et al., 2010; Frangakis and Forster, 2004;

* Corresponding author. Fax: +49 89 8578 2641.
E-mail address: foerster@biochem.mpg.de (F. Förster).

1047-8477/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jsb.2011.12.003

Rath et al., 2003; Renken et al., 2009; Xu et al., 2011), alignment (Amat et al., 2010; Bartesaghi et al., 2008; Förster et al., 2005; Schmid and Booth, 2008; Walz et al., 1997; Winkler, 2007) and classification (Bartesaghi et al., 2008; Förster et al., 2008; Heumann et al., 2011; Scheres et al., 2009; Stolken et al., 2010; Winkler et al., 2009; Yu et al., 2010; Yu and Frangakis, 2011), but a unified platform that covers the whole workflow from subtomogram localization to averaging and classification is not available to date. Here, we present the open-source toolbox PyTom, which covers a workflow comprising localization of macromolecules based on template matching, correlation-based alignment of subtomograms, and their classification by a novel stochastic method. The package uses the scripting language python as a frontend, which warrants transparency of methods, ease-of-use for non-experts, scripting possibility for image processing experts. Various libraries, mostly implemented in C++, can be accessed through PyTom, which warrants efficient implementation of algorithms and their parallelization. The modular design of PyTom enables using different scores and optimization methods for subtomogram averaging and classification, which are compared in this work.

2. Material and methods

2.1. Software design

PyTom is a modular toolbox for tomogram processing with emphasis on subtomogram analysis. PyTom is divided into four levels (Fig. 1a): (i) the core-level for computationally efficient implementation of basic functions, (ii) the script level that allows easy use of the core-level and combination with other libraries and programming languages, (iii) the application-level that houses classes for specific tasks in CET, (iv) the user level that provides easy access to all algorithms and tools. Together, all levels provide a robust platform for subtomogram processing that should be convenient for a broad audience. Users who are not interested in methodological development profit from easy access through the user level and can process large quantities of data due to efficient implementation and parallelization, while image processing experts can make use of the application, scripting, and core level to implement customized workflows or novel algorithms.

2.1.1. Core level

Components of the core level are written in C/C++ and compiled individually for specific computer platforms. The main component of this layer is the library libtomc, where all numerical operations for subtomogram processing algorithms are implemented. This library is a further development of the core used in (Stolken et al., 2010); the library contains basic functions for 3D image processing, such as real-space transformations, filters and Fourier transforms. In its current version, libtomc supports reading and writing of common binary file types used in CET (EM, CCP4, MRC, Spider). The user furthermore has the choice between linear interpolation (e.g., used in AV3 (Förster et al., 2005)) and cubic interpolation schemes (Lagrange, Spline). Higher order interpolation typically improves interpolation accuracy, but it is computationally more demanding and artifacts may occur close to the Nyquist frequency. We typically over sample subtomograms by a factor ~2 beyond the target resolution when using higher order interpolations. Moreover, libtomc uses standard libraries such as boost (http://www.boost.org) and fftw (http://www.fftw.org), which are highly optimized.

2.1.2. Scripting level

Python (http://www.python.org) is an object-oriented scripting language that is increasingly widespread in the structural biology community. For example, the single-particle analysis (SPA) packages EMAN2 (Tang et al., 2007) and SPARX (Hohn et al., 2007), the X-ray crystallography suite PHENIX (Adams et al., 2002), and the comparative modeling program MODELLER (Sali and Blundell, 1993) use python as a command-line interface. Importantly, python is free-of-charge and open source, which is a considerable advantage over other scripting languages such as MATLAB. The Swig interface (http://www.swig.org) allows accessing libtomc and other C/C++ libraries from python. Thus, python allows 'gluing' different libraries and to access them from a single interface (PyTom). Higher level algorithms are all implemented in python

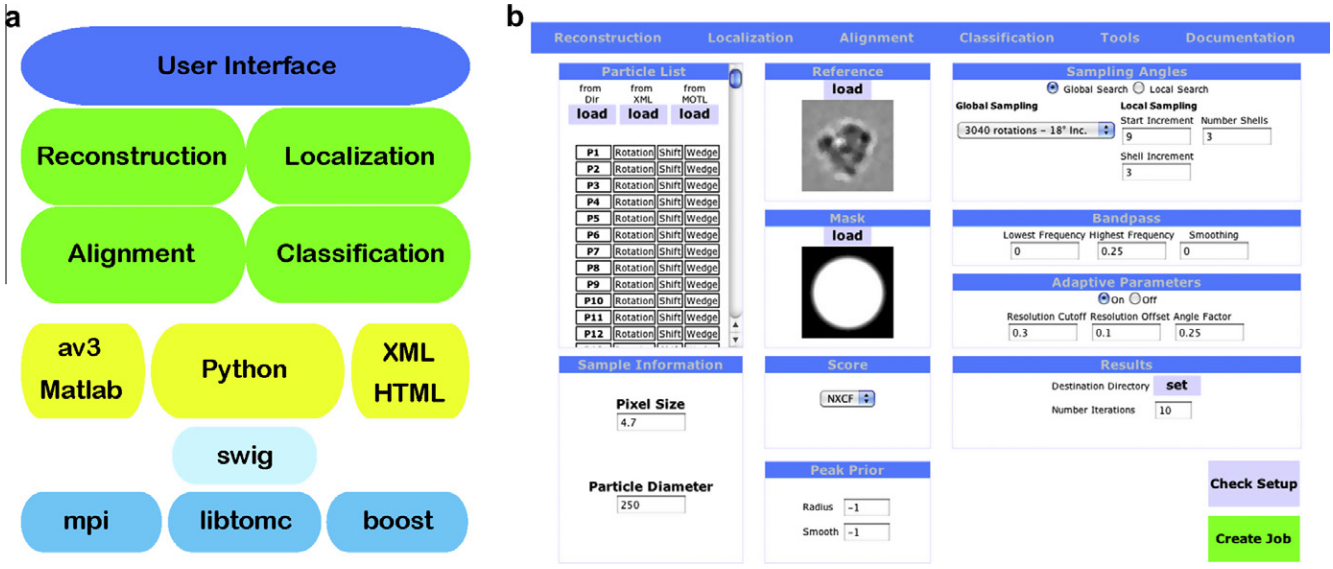

Fig. 1. PyTom software design. (a) PyTom consists of three levels: (i) the "core level" (blue) consists of compiled C/C++ components that are optimized for subtomogram processing (libtomc) and linked to external libraries such as mpi and boost. (ii) The "scripting level" (yellow) is implemented in python. Data storage mainly bases on XML and related languages. Moreover, PyTom provides interfaces to other software such as AV3 and MATLAB. The scripting level accesses core functionalities via Swig. (iii) The "application level" (green) houses all tomogram processing algorithms. (iv) The "user level" provides python scripts for specific tasks and web-interfaces. (b) Screenshot of the web-interface to generate the input for the alignment script.