BIOINFORMATICS APPLICATIONS NOTE Vol. 26 no. 20 2010, pages 2626–2628
doi:10.1093/bioinformatics/btq481

Structural bioinformatics

Advance access publication August 23, 2010

OpenStructure: a flexible software framework for computational structural biology

Marco Biasini[1,2], Valerio Mariani[1,2], Jürgen Haas[1,2], Stefan Scheuber[1,2], Andreas D. Schenk[3], Torsten Schwede[1,2,*] and Ansgar Philippsen[1]

[1]Biozentrum, Universität Basel, Basel, [2]SIB Swiss Institute of Bioinformatics, Basel, Switzerland and [3]Department of Cell Biology, Harvard Medical School, Boston, MA 02115, USA

Associate Editor: Anna Tramontano

ABSTRACT

Motivation: Developers of new methods in computational structural biology are often hampered in their research by incompatible software tools and non-standardized data formats. To address this problem, we have developed OpenStructure as a modular open source platform to provide a powerful, yet flexible general working environment for structural bioinformatics. OpenStructure consists primarily of a set of libraries written in C++ with a cleanly designed application programmer interface. All functionality can be accessed directly in C++ or in a Python layer, meeting both the requirements for high efficiency and ease of use. Powerful selection queries and the notion of entity views to represent these selections greatly facilitate the development and implementation of algorithms on structural data. The modular integration of computational core methods with powerful visualization tools makes OpenStructure an ideal working and development environment. Several applications, such as the latest versions of IPLT and QMean, have been implemented based on OpenStructure—demonstrating its value for the development of next-generation structural biology algorithms.

Availability: Source code licensed under the GNU lesser general public license and binaries for MacOS X, Linux and Windows are available for download at http://www.openstructure.org.

Contact: torsten.schwede@unibas.ch

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on July 7, 2010; revised on August 6, 2010; accepted on August 14, 2010

1 INTRODUCTION

We introduce OpenStructure, a flexible software framework for computational structural biology, a solid, yet flexible and versatile toolkit for rapid prototyping of new methods as well as their productive implementation. Typically, method development in structural bioinformatics involves combining different independent software tools, and significant effort is devoted to writing code for input/output operations and format conversions between different packages. This culminates when data and algorithms from different domains are to be combined, e.g. protein structures, protein sequence annotation and chemical ligands. Several software tools and frameworks are available today for molecular modeling, e.g. MMTK (Hinsen, 2000), Coot (Emsley *et al.*, 2010) MolIDE (Canutescu and Dunbrack, 2005), Modeller (Eswar *et al.*, 2008), bioinformatics algorithms libraries, e.g. BALL (Kohlbacher and Lenhof, 2000), workflow automation tools, e.g. Biskit (Grunberg *et al.*, 2007) or KNIME (www.knime.org) and visualization e.g. VMD (Humphrey *et al.*, 1996), PyMol (www.pymol.org), DINO (www.dino3d.org), or SwissPdbViewer (Guex *et al.*, 2009).

OpenStructure is a flexible software framework tailored for computational structural biology, which combines a C++ based library of commonly used functionality with a Python layer and powerful visualization tools. While PyMol and VMD also combine a scripting environment with sophisticated visualization tools, they are primarily geared toward visualization and less on providing a clean application programmer interface (API) that is easy to use and allows for rapid development of new algorithms. OpenStructure is also designed to easily accommodate interfaces to already existing software. This allows for rapid visually enhanced prototyping of new functionality, making OpenStructure an ideal environment for the development of next-generation structural biology algorithms. For example, new versions of the QMean tools for model quality assessment (Benkert *et al.*, 2009a, b) are based on OpenStructure, as well as the structural analysis tools in ProteinModelPortal (Arnold *et al.*, 2009). Further, work is on the way to implement the next generation of the SWISS-MODEL pipeline using the OpenStructure framework (Arnold *et al.*, 2006; Bordoli *et al.*, 2009).

2 IMPLEMENTATION

In OpenStructure, molecular or chemical entities, such as macromolecules, sequences, alignments or electron density maps, are represented as objects, offering a comprehensive set of functions for data manipulation and information querying. Typically, users interact with a high-level Python interface, while 'power users' with high computational requirements access the API at the level of C++.

Functionality in OpenStructure is grouped into modules. Each of these modules consists of a computational core as a shared library of C++ code and a set of Python modules built on top of the exported API. Parts of the computational core and the graphical user interface of the Image Processing Library and Toolkit IPLT (Philippsen *et al.*, 2007) have been incorporated into OpenStructure to offer versatile handling of image data with support for various algorithms in one, two and three dimensions. A graphics module for real-time rendering of molecules, density maps and molecular surfaces provides functionalities for data visualization.

*To whom correspondence should be addressed.

Processing and visualization of molecular entities often requires filtering by certain selection criteria. These selections are implemented as so-called EntityViews, containing subsets of atoms, residues, chains and bonds of the respective EntityHandle chosen using selection statements (queries). The EntityView class shares a common interface with the EntityHandle class it points to, and hence they can be used interchangeably. This handle/view concept pertains to the full structural hierarchy, i.e. residue views will only contain the atoms that were part of the selection, etc. The query language supports sophisticated selection criteria (for example, distance-based selection, Boolean operators, selections based on user-defined properties, and so on).

In order to infer connectivity and topology when reading molecular coordinate files, we make use of the chemical components dictionary which is part of the official PDB distribution (Berman *et al.*, 2003). Thus, detailed information is available on any of the chemical components, allowing the framework to ensure correct connectivity and topology during the load process and issue appropriate warnings. The connectivity step is extensible and its behavior can be adapted by overloading functions. Additionally, a heuristic method is available as a fallback for loading unknown residues or to handle non-standard residue and atom names.

3 APPLICATION EXAMPLE

Most users will interact with OpenStructure using Python. The code fragment in Supplementary Table S1 illustrates the expressiveness of the OpenStructure API in combining data from different domains. In this example, we compare the sequence conservation of residues in contact with a ligand with the rest of the protein, quantifying the visually derived hypothesis that the binding-site residues of the SH2 domain are more conserved than the rest. This is achieved by mapping of a conservation score derived from a multiple sequence alignment of various SH2 domains ('sh2.aln') onto a representative structure (PDB: 3IMJ) (DeLorbe *et al.*, 2009) and identifying residues in direct contact with the ligand. Figure 1 shows the results displayed in the DNG ('DINO/DeepView Next Generation') graphical user interface, using the conservation score to color a molecular surface representation.

The OpenStructure distribution contains several scripting examples to introduce new users to the functionalities and usage style of the tool kit, such as scripts to animate molecular dynamics trajectories, calculate electron density maps from atomistic structures or rank short peptide fragments according to their correlation with electron density. Exhaustive documentation and tutorials are provided on the web site. Mailing lists for OpenStructure users and developers provide a forum to ask questions, report problems or suggest new developments.

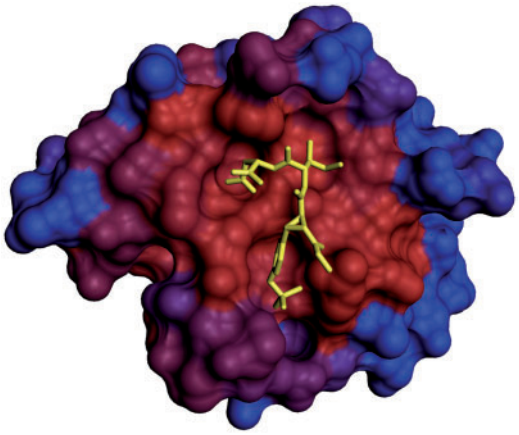

Fig. 1. Molecular surface representation of a SH2 domain (PDB:3IMJ) colored by conservation of the positions in a multiple sequence alignment. The color scale ranges from red for conserved residues to blue for residues with high variability. The ligand peptide is shown as yellow stick representation. The image was rendered in OpenStructure, the molecular surface was calculated using MSMS (Sanner *et al.*, 1996). See Supplementary Table S1 for details on calculation of sequence conservation scores.

ACKNOWLEDGEMENT

We would like to thank Andras Aszodi for inspiring discussion during the conception phase of the project, and Pascal Benkert and Tobias Schmidt for critical feedback. OpenStructure uses Eigen (http://eigen.tuxfamily.org), FFTW (http://www.fftw.org), Boost (http://www.boost.org), Qt4 (http://qt.nokia.com) and PyQt4 (http://www.riverbankcomputing.co.uk).

Funding: Development of OpenStructure was funded by the SIB Swiss Institute of Bioinformatics and the Biozentrum University of Basel. Implementation of the structure comparison for the Nature PSI SBKB Protein Model Portal based on OpenStructure was supported by the National Institutes of Health as a sub-grant with Rutgers University, under Prime Agreement Award Number: (3U54GM074958-05S2).

Conflict of Interest: none declared.

REFERENCES

Arnold,K. *et al.* (2006) The SWISS-MODEL workspace: a web-based environment for protein structure homology modeling. *Bioinformatics*, **22**, 195–201.

Arnold,K. *et al.* (2009) The protein model portal. *J. Struct. Funct. Genomics*, **10**, 1–8.

Benkert,P. *et al.* (2009a) QMEAN server for protein model quality estimation. *Nucleic Acids Res.*, **37**, W510–W514.

Benkert,P. *et al.* (2009b) Global and local model quality estimation at CASP8 using the scoring functions QMEAN and QMEANclust. *Proteins*, **77** (Suppl. 9), 173–180.

Berman,H. *et al.* (2003) Announcing the worldwide Protein Data Bank. *Nat. Struct. Biol.*, **10**, 980.

Bordoli,L. *et al.* (2009) Protein structure homology modeling using SWISS-MODEL workspace. *Nat. Protocol*, **4**, 1–13.

Canutescu,A.A. and Dunbrack,R.L. Jr (2005) MollDE: a homology modeling framework you can click with. *Bioinformatics*, **21**, 2914–2916.

DeLorbe,J.E. *et al.* (2009) Thermodynamic and structural effects of conformational constraints in protein-ligand interactions. Entropic paradoxy associated with ligand preorganization. *J. Am. Chem. Soc.*, **131**, 16758–16770.

Emsley,P. *et al.* (2010) Features and development of Coot. *Acta Crystallogr. D. Biol. Crystallogr.*, **66**, 486–501.

Eswar,N. *et al.* (2008) Protein structure modeling with MODELLER. *Methods Mol. Biol.*, **426**, 145–159.

Grunberg,R. *et al.* (2007) Biskit–a software platform for structural bioinformatics. *Bioinformatics*, **23**, 769–770.

Guex,N. *et al.* (2009) Automated comparative protein structure modeling with SWISS-MODEL and Swiss-PdbViewer: a historical perspective. *Electrophoresis*, **30** (Suppl. 1), S162–S173.

Hinsen,K. (2000) The molecular modeling toolkit: a new approach to molecular simulations. *J. Comput. Chem.*, **21**, 79–85.

Humphrey,W. *et al.* (1996) VMD: visual molecular dynamics. *J. Mol. Graph*, **14**, 33–38, 27–38.

Kohlbacher,O. and Lenhof,H.P. (2000) BALL–rapid software prototyping in computational molecular biology. Biochemicals Algorithms Library. *Bioinformatics*, **16**, 815–824.

Philippsen,A. *et al.* (2007) Collaborative EM image processing with the IPLT image processing library and toolbox. *J. Struct. Biol.*, **157**, 28–37.

Sanner,M.F. *et al.* (1996) Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, **38**, 305–320.