



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Towards Community Software for Diffraction Microscopy

C. Jacobsen, A. Barty, T. Beetz, H. Chapman, N.
D'Imperio, V. Elser, X. Huang, E. Lima, S. Marchesini,
H. Miao, P. Thibault, D. Shapiro

June 23, 2005

International Workshop on Phase Retrieval and Coherent
Scattering
Porquerolles, France
June 14, 2005 through June 17, 2005

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Towards community software for diffraction microscopy

Chris Jacobsen¹, Anton Barty², Tobias Beetz¹, Henry Chapman², Nicholas D'Imperio³, Veit Elser⁴, Xiaojing Huang¹, Enju Lima¹, Stefano Marchesini², Huijie Miao¹, Pierre Thibault⁴, David Shapiro⁵

¹Department of Physics & Astronomy, Stony Brook University, Stony Brook, NY, USA

²Lawrence Livermore National Laboratory, Livermore, CA, USA

³Computational Science Center, Brookhaven National Laboratory, Upton, NY, USA

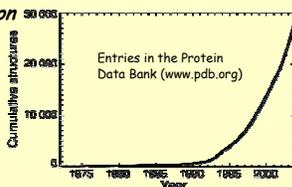
⁴Department of Physics, Cornell University, Ithaca, NY, USA

⁵Center for Biophotonics Science and Technology, University of California at Davis, Sacramento, CA, USA

Examples of community software for "image" reconstruction

Protein crystallography has been revolutionized!

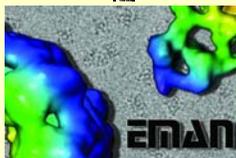
- Improved crystallization methods
 - Improved synchrotron beamlines and MAD/SAD data
 - **Improved, and largely standardized, software**
- It is now possible for cell biologists with little training in crystallography to obtain structures from "easy" crystals



Single-particle methods in electron microscopy

National Center for Macromolecular Imaging (Baylor University) makes EMAN available for data processing: Python scripting with C/C++ code

Collaborative effort between Baylor, Wadsworth/Albany, and LBNL/Donner Lab to develop parallelized software for large data sets. The whole field will benefit.



We should emulate this approach!

Diffraction imaging cannot exist without modern computers

- Iterative algorithms are used to phase the measured diffraction data and produce a real-space image
- Computers are also integral to reliable collection of minimum-dose data throughout a 3D tilt sequence.

Balancing competition and cooperation

- We all want to develop new tricks and get credit for our efforts. This is assuredly good! - competition stimulates creativity.
- However, once an idea is published, why not make it available? Others can cite your paper, and benefit from the results (and you can gain by utilizing a software tool they have developed). "If I have seen further it is by standing on the shoulders of giants" - I. Newton

Example: ALS beamline 9.0.1 apparatus control software

Apparatus is used by three groups

- East coast group: diffraction microscopy of biological specimens
- West coast group 1: diffraction microscopy of materials science specimens
- West coast group 2: diffraction microscopy of laser-aligned viruses

Software approach

- C++ code controls the hardware.
 1. Acts as an internet server (allows for efficient remote operation).
 2. Specific hardware isolated into C++ classes. Switching to different motor drivers or CCD cameras would not be hard.
- IDL code (Research Systems Inc.) in three main modules:
 1. Underlying client layer that handles binary communication with C++ server.
 2. Graphical user interface code for user control, diffraction data display (can run stand-alone, separate from experimental hardware). Has built-in "hooks" for user-added options.
 3. "Script" code for easy specification of a data acquisition sequence.

Example: "merger" and "xewald"

"merger" combines multiple 2D measurements (H. Chapman)

- Scatter-reducing apertures and beamstops are adjusted between individual exposures to obtain clean, non-saturated data
- "merger" is a nice tool for combining these exposures to obtain the full data from one viewing angle.

"xewald" delivers 3D data cube from 2D tilt series (A. Barty)

- Defines a regular 3D grid to hold the data
- Reads in 2D files created by "merger" and interpolates them onto the 3D grid
- Lets you view cuts through the 3D grid, and write it out to a file for iterative reconstruction

These routines are both written in IDL (Research Systems Inc.) so they can run happily on Windows, Mac OS X, Linux...

Areas suggested for future work

Common file formats for reconstruction.

Use big-endian binary format. Provide read-write routines in C, IDL.

- Assembled diffraction intensity: common format for 2D reconstructions from "merger", 3D reconstructions from "xewald". Include provision for specifying the experimental error of each pixel/voxel.
- Support mask: define area of support as binary and possibly graduated bitmapped image
- Iterate amplitude: present result for real or complex object in real space

Routines for evaluating raw data

- Finding the location of the zero-frequency center of a diffraction pattern.
- Characterizing background noise and information content. Displaying the autocorrelation of different diffraction subregions (S. Marchesini)
- Isolating the beamstop, as well as saturated pixels
- Estimating the support from the autocorrelation

C/C++ routines for reconstruction on parallel computers

Message Passing Interface (MPI) enabled code on multiple nodes can be very effective for reconstructing large data sets. See poster by A. Barty. Python scripting of C/C++ routines?

- 2D and 3D Fourier transforms, such as the Apple dist_fft library
- Routines for applying support, Fourier modulus constraints (HIO, difference map, shrink-wrap)
- Routines for file input/output
- Routines for averaging iterates, and evaluating reconstructions
- Refinement of reconstructions: improved estimates of true tilt, scaling, centering of 2D diffraction data?

Data visualization

- Routines for interactive display of 2D and 3D reconstruction
- Routines for generating movies of rotating 3D reconstructions with partial transparency

Suggested mechanisms for participation

- Directory tree of software will have a README file (html or PDF suggested) in each subdirectory, describing the status of each part of code. Authors will then indicate whether code should be used by inquiring about a collaboration, or by citing a paper, or in some other manner.
- Subscribe to the mailing list by sending e-mail stating "subscribe diffmic" to majordomo@xray1.physics.sunysb.edu. Propose new developments on this mailing list.
- Request password for CVS archive cvs_diffmic on xray1.physics.sunysb.edu. Concurrent Version System (CVS) is commonly used to coordinate the work of distributed software developers (e.g., Linux)

Acknowledgements

We gratefully acknowledge support from the following agencies:

- Division of Materials Science, Office of Science, US Department of Energy
- NIGMS, National Institutes of Health
- National Science Foundation