Titelblatt

Information Recovery in the Biological Sciences: Protein Structure Determination by Constraint Satisfaction, Simulation and Automated Image Processing

Inauguraldissertation

zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

Bryant Gipson

aus Davis, California, USA

Basel, Switzerland, June 2010

Rückseite des Titelblattes

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät

auf Antrag von

Henning Stahlberg & Andreas Engel

Basel, den April 27 2010

Prof. Dr. Eberhard Parlow
Dekanin/Dekan

creative commons

Namensnennung-Keine kommerzielle Nutzung-Keine Bearbeitung 2.5 Schweiz

---

Sie dürfen:

das Werk vervielfältigen, verbreiten und öffentlich zugänglich machen

Zu den folgenden Bedingungen:

Namensnennung. Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen (wodurch aber nicht der Eindruck entstehen darf, Sie oder die Nutzung des Werkes durch Sie würden entlohnt).

Keine kommerzielle Nutzung. Dieses Werk darf nicht für kommerzielle Zwecke verwendet werden.

Keine Bearbeitung. Dieses Werk darf nicht bearbeitet oder in anderer Weise verändert werden.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter welche dieses Werk fällt, mitteilen. Am Einfachsten ist es, einen Link auf diese Seite einzubinden.
- Jede der vorgenannten Bedingungen kann aufgehoben werden, sofern Sie die Einwilligung des Rechteinhabers dazu erhalten.
- Diese Lizenz lässt die Urheberpersönlichkeitsrechte unberührt.

Quelle: http://creativecommons.org/licenses/by-nc-nd/2.5/ch/ Datum: 3.4.2009

Information Recovery in the Biological Sciences: Protein Structure Determination by Constraint Satisfaction, Simulation and Automated Image Processing

PhD Thesis

Bryant Gipson

For Preeti, Always

Note on Thesis

Sections of the following thesis were included in pre-published and published works, in collaboration with other researchers. On the two papers on which I am not the primary researcher I have listed my contributions to the project.

Chapter 2: Zeng, X., Gipson, B., Zhang, Z. Y., Renault, L. & Stahlberg, H. Automatic lattice determination for two-dimensional crystal images. J. Struct. Biol. 160, 351-359 (2007).

> *2dx_getlat*, *manual lattice refinement by least squares* and relevant mentioned additions to *2dx_image* including the fullscreen browser performed by me.

Chapter 3: Gipson, B., Zeng, X., Zhang, Z. & Stahlberg, H. 2dx—User-friendly image processing for 2D crystals. Journal of Structural Biology 157, 64-72, doi:10.1016/j.jsb.2006.07.020 (2007).

Chapter 4: Gipson, B., Zeng, X. & Stahlberg, H. 2dx_merge: data management and merging for 2D crystal images. Journal of Structural Biology 160, 375-384, doi:10.1016/j.jsb.2007.09.011 (2007).

Chapter 5: Gipson, B., Masiel. D, Browning N. D., Spence, J., Mitsuoka, K., and Stahlberg, H., Automatic Recovery of Missing Amplitudes and Phases in Tilt-Limited Electron Crystallography of 2D Crystals, Nature Methods (Submitted April, 2010).

Chapter 6: Masiel, D. Gipson, B., Morgan, D., Guo, T., Spence, J., Browning, N. D., Metaheuristic Algorithms for the Phase Problem in Ultrafast Diffractive Imaging, (In preparation)

> This work was co-researched by Dan Masiel and I. Initial investigations on the success of genetic algorithms and the effect of pathological supports on iterative reconstruction were performed by me.

Chapter 7, Chapter 8: The projects in these chapters represent unpublished work performed with the labs of Dr. Patrice Khoel and Dr. Subhadip Raychaudhuri, respectively. Unless otherwise indicated, all work presented was entirely my own.

> I have a commonplace-book for facts and another for poetry, but I find it difficult always to preserve the vague distinction which I had in my mind, for the most interesting and beautiful facts are so much the more poetry and that is their success. They are *translated* from earth to heaven. I see that if my facts were sufficiently vital and significant --perhaps transmuted more into the substance of the human mind-- I should need but one book of poetry to contain them all.
> -- Henry David Thoreau

Acknowledgements

It would be reckless to think that I could properly thank all of the innumerable people who have helped along the road to this point. I follow, then, with the briefest sampling of some of the most important people who brought me this far.

Henning Stahlberg, who has become nothing less than an academic father to me. His apparently limitless intuitive knowledge of just about everything he comes across, from microscopes, to Fortran, to Fourier theory is only matched by his ability to easily and understandably convey these subjects to anyone, regardless of background.

From my undergraduate days, David Kornreich, who consistently showed his devotion first to his students, and that analytical and quantum mechanics, group theory, tensors and variational calculus were both “awesome” and could be “intuitively obvious to the most casual observer.” Phyllis Chinn, whose “discovery” based graph theory course remains one of the most influential academic experiences of my life.

Of course a big thanks goes out to all of the members of the Stahlberg/Engel labs both in UC Davis and here in Basel, as well as the guys formerly from the UC Davis Browning Lab: Dan Masiel, Mike Sarahan and David Morgan.

Most of all, though words seem cheap against the incomparable world she's offered me, thanks goes to Preeti Gipson. A wonderful scientist and collaborator who is never shy for offering a discussion or debate (or supportive smile) on virtually any subject --her relentless curiosity ever balanced by her broader sense of perspective. I couldn't have wished for a better Wife and best friend.

Finally, thanks go to my family. My father who helped me see meaning and beauty in every mystery the universe offers. My mother, who helped show me that every mystery, no matter the size, yields to patience. My brother, who never tired of finding out what was beyond "just *one more"* mountain top, valley or outcropping of boulders.

My in-law family in India, for whom I haven't enough words, who effortlessly offered me their support and love, as if I'd simply been away a few years and finally found my way back home.

And Raghunath Prasad (Uncle-Ji) who ever reminds me: (to borrow from Frost) "There are two kinds of teachers: the kind that fill you with so much quail shot that you can't move, and the kind that just gives you a little prod behind and you jump to the skies."

This thesis contains works which benefit from the experience, support and funding from many groups and individuals. From the chapters below the following acknowledgements are offered:

Chapter 2: Automatic Determination of Lattice Parameters

This work was supported by the NSF, Grant No.MCB0447860 and by the NIH, Grant No. U54-GM074929. We thank Richard Henderson for his explanations of the algorithms used in *autoindex*. Development of some of these algorithms was started in the laboratories of Jacques Dubochet in Lausanne, and Andreas Engel in Basel, Switzerland.

Chapter 3: Processing a Single TEM Image

This work was supported by the NSF, grant number MCB-0447860 and by the NIH, Grant No. U54GM074929. We wish to thank Per Bullough, Anchi Cheng, Andreas Engel, Bob Glaeser, Niko Grigorieff, Richard Henderson, Werner Kühlbrandt, Kaoru Mitsuoka, Ansgar Philippsen, Sriram Subramaniam, Vinzenz Unger, Janet Vonck, and Tom Walz, who generously shared their knowledge and experience in image processing, and we thank Bob Glaeser and Rena Hill for comments on the manuscript. Development of some of the underlying scripts was started in the laboratories of Jacques Dubochet in Lausanne, and Andreas Engel in Basel, Switzerland.

Chapter 4: Merging 2D Data into a 3D Dataset

This work was supported by the NSF, Grant No.MCB0447860 and by the NIH, Grant No. U54-GM074929. We thank Remco Wouts, Werner Kühlbrandt, Anchi Cheng, Vinzenz Unger, and Tom Walz for fruitful discussions, and Per Bullough for fruitful discussions and help with symmetry definitions and systematic absences. Development of some of the underlying scripts was started in the laboratories of Jacques Dubochet in Lausanne, and Andreas Engel in Basel, Switzerland.

Chapter 5: Model Free Structure Determination relative to Prior Knowledge

The authors gratefully acknowledge Yoshinori Fujiyoshi of the Department of Biophysics in Kyoto University, Kyoto, Japan for the contribution of the Bacteriorhodopsin dataset. We would also like to thank Tilman Schirmer of the department of Structural Biology in the University of Basel, Basel, Switzerland for his in role in generating comparison atomic models, for analysis of results and for his many helpful suggestions on the paper. Additionally we would like to thank Thomas Strohmer of the Mathematics Department of UC Davis, Davis, California, USA, for discussions on optimization and Fourier representations generally. This work was in part supported by the NSF, grant number MCB-0447860, and by the NIH, grant number U54-GM074929, as well as DOE award DE-FG03-02ER45996.

Figure 16 and initial rigid body fitting performed for CRC comparison were performed by the UCSF Chimera package from the Resource for Biocomputing, Visualization, and Informatics at the University of California, San Francisco (supported

by NIH P41 RR-01081). Figure 18 was generated by COOT (Emsley and Cowtan, 2004).

Chapter 6: Multiple Solutions: Searching Solution Space with Meta-Heuristics

The authors would like to thank George Suarez for his illustration in figure 1. Supported by DOE/NNSA award DEPS5205NA

Chapter 7: Using the GPU for the Simulation of Protein Structural Dynamics

I would like to thank Patrice Koehl for his support, this incredible project and his intelligible explanations of applications of linear algebra to variational calculus. Additionally I would like to thank the entire Koehl lab for their assistance and support.

Chapter 8: Reasonable Simplifications for a Larger Biological Story: Kinetic Monte Carlo Simulations

I would like to thank Philippos Tsourkas for his help with the random number generator and general help in the lab, and Dr. Raychaudhuri as principal coordinator for this project and for his guidance and input into its development.

Chapter 1 Introduction

1. Outline of the Problem

Regardless of the field of study or particular problem, any experimental science always poses the same question: "What object or phenomena generated the data that we see, given what is known?"

In the field of 2D electron crystallography, data is collected from a series of two-dimensional images, formed either as a result of diffraction mode imaging or TEM mode real imaging. The resulting dataset is acquired strictly in the Fourier domain as either coupled Amplitudes and Phases (as in TEM mode) or Amplitudes alone (in diffraction mode). In either case, data is received from the microscope in a series of CCD or scanned negatives of images which generally require a significant amount of pre-processing in order to be useful.

2. From data to structure, automatically

Traditionally, processing of the large volume of data collected from the microscope was the time limiting factor in protein structure determination by electron microscopy. Data must be initially collected from the microscope either on film-

negatives, which in turn must be developed and scanned, or from CCDs of sizes typically no larger than 2096x2096 (though larger models are in operation). In either case, data are finally ready for processing as 8-bit, 16-bit or (in principle) 32-bit grey-scale images.

Regardless of data source, the foundation of all crystallographic methods is the presence of a regular Fourier lattice. Two dimensional cryo-electron microscopy of proteins introduces special challenges as multiple crystals may be present in the same image, producing in some cases several independent lattices. Additionally, scanned negatives typically have a rectangular region marking the film number and other details of image acquisition that must be removed prior to processing.

If the edges of the images are not down-tapered, vertical and horizontal "streaks" will be present in the Fourier transform of the image --arising from the high-resolution discontinuities between the opposite edges of the image. These streaks can overlap with lattice points which fall close to the vertical and horizontal axes and disrupt both the information they contain and the ability to detect them. Lastly, SpotScanning (Downing, 1991) is a commonly used process where-by circular discs are individually scanned in an image. The large-scale regularity of the scanning patter produces a low frequency lattice which can interfere and overlap with any protein crystal lattices.

Chapter 2 introduces a series of methods packaged into 2dx (Gipson, et al., 2007) which simultaneously address these problems, automatically detecting accurate crystal lattice parameters for a majority of images. Further though, this chapter stands as a template for the automation of all subsequent image processing steps on the road to a fully processed dataset.

The broader picture of image processing is one of reproducibility. The lattice parameters, for instance, are only one of hundreds of parameters which must be determined or provided and subsequently stored and accessed in a regular way during image processing. Numerous steps, from correct CTF and tilt-geometry determination to the final stages of symmetrization and optimal image recovery must be performed sequentially and repeatedly for hundreds of images.

The goal in such a project is then to automatically process as significant a portion of the data as possible and to reduce unnecessary, repetitive data entry by the user. Chapter 3 introduces 2dx (Gipson, et al., 2007), the image processing package designed to automatically process individual 2D TEM images. This package focuses on reliability, ease of use and automation to produce finished results necessary for full three-dimensional reconstruction of the protein in question.

Once individual 2D images have been processed, they contribute to a larger project-wide 3-dimensional dataset. Several challenges exist in processing this dataset, besides simply the organization of results and project-wide parameters. In particular, though tilt-geometry, relative amplitude scaling and absolute orientation are in principle known (or obtainable from an individual image) errors, uncertainties and heterogeneous data-types provide for a 3D-dataset with many parameters to be optimized. Chapter 4 introduces *2dx_merge* (Gipson, et al., 2007) the follow-up to the first release of 2dx which had originally processed only individual images. Based on the guiding principles of the earlier release, *2dx_merge* focuses on ease of use and automation. The result is a fully qualified 3D structure determination package capable of turning hundreds of electron micrograph images, nearly completely automatically, into a full 3D structure.

Most of the processing performed in the 2dx package is based on the excellent suite of programs termed collectively as the MRC package (Crowther, et al., 1996). Extensions to this suite and alternative algorithms continue to play an essential role in image processing as computers become faster and as advancements are made in the mathematics of signal processing. In this capacity, an alternative procedure to generate a 3D structure from processed 2D images is presented in Chapter 5. This algorithm, entitled “Projective Constraint Optimization” (PCO), leverages prior known information, such as symmetry and the fact that the protein is bound in a membrane, to extend the normal boundaries of resolution. In particular, traditional methods (Agard, 1983) make no attempt to account for the “missing cone” a vast, un-sampled, region in 3D Fourier space arising from specimen tilt limitations in the microscope. Provided sufficient data, PCO simultaneously refines the dataset, accounting for error, as well as attempting to fill this missing cone.

Though PCO provides a near-optimal 3D reconstruction based on data, depending on initial data quality and amount of prior knowledge, there may be a host of solutions, and more importantly pseudo-solutions, which are more-or-less consistent with the provided dataset. Trying to find a global best-fit for known information and data can be a daunting challenge mathematically, to this end the use of meta-heuristics is addressed in Chapter 6. Specifically, in the case of many pseudo-solutions, so long as a suitably defined error metric can be found, quasi-evolutionary swarm algorithms can be used that search solution space, sharing data as they go. Given sufficient computational power, such algorithms can dramatically reduce the search time for global optimums for a given dataset.

3. The Big picture: after structure determination

Once the structure of a protein has been determined, many questions often remain about its function. Questions about the dynamics of a protein, for instance, are not often readily interpretable from structure alone. To this end an investigation into computationally optimized structural dynamics is described in Chapter 7. Here, in order to find the most likely path a protein might take through "conformation space" between two conformations, a graphics processing unit (GPU) optimized program and set of libraries is written to speed of the calculation of this process 30x. The tools and methods developed here serve as a conceptual template as to how GPU coding was applied to other aspects of the work presented here as well as GPU programming generally.

Chapter 8 takes an apparent step in reverse, presenting a dramatic, yet highly predictive, simplification of a complex biological process. Kinetic Monte Carlo simulations idealize thousands of proteins as interacting agents by a set of simple rules (i.e. react/dissociate), offering highly-accurate insights into the large-scale cooperative behavior of proteins. This work demonstrates that, for many applications, structure, dynamics or even general knowledge of a protein may not be necessary for a meaningful biological story to emerge. Additionally, even in cases where structure and function is known, such simulations can help to answer the biological question in its entirety from structure, to dynamics, to ultimate function.

Chapter 2 Automatic Determination of Lattice Parameters

Originally appeared as: Zeng, X., Gipson, B., Zhang, Z. Y., Renault, L. & Stahlberg, H. Automatic lattice determination for two-dimensional crystal images. J. Struct. Biol. 160, 351-359 (2007).

1. Introduction

Electron crystallography determines the structure of two-dimensional (2D) crystals of membrane proteins or other periodically arranged samples, using cryo-electron microscopy (cryo-EM) data collection and computer image processing (Henderson, et al., 1990; Henderson and Unwin, 1975). The electron microscope can be used in either the imaging or the diffraction mode. In imaging mode, real-space images of the crystalline samples are recorded on the instrument's CCD camera or photographic film. The latter needs to be digitized with a scanner before further processing. Digitized images can then be numerically Fourier transformed, producing complex datasets, which contain amplitudes and phases. Since computational correction of 2D crystal defects in the image can be done by computational ''unbending'' (Crowther, et al., 1996), useful real-space images can also be recorded for crystal samples of limited order. Nevertheless, the resolution of such real-space images is

affected by beam-induced sample charging and drum-head movement, as well as by sample vibration or drift. While phases obtained from Fourier-transformed 2D crystal real-space images are of relatively good quality, the amplitudes are affected by the electron microscope's contrast transfer function, and are therefore less well determined.

Alternatively, the electron microscope can record electron diffraction patterns of the 2D crystal samples, which are preferably recorded onto CCD cameras due to their superior dynamic range. The electron diffraction patterns are then evaluated similarly to X-ray diffraction (XRD) patterns in X-ray crystallography; which yield the intensities of the diffracted rays, and thereby contain the information about the structure's amplitudes. Phase information is not contained in the diffraction pattern, and has to be acquired by different means. Electron diffraction data collection, in contrast, generally does not suffer from sample charging or sample movement during the data collection. Since 2D crystal image unbending cannot be done with a diffraction pattern, in practical terms electron diffraction can only be done with larger, well-ordered 2D crystal samples.

Electron crystallography structure reconstruction of membrane proteins ideally utilizes real-space images to obtain an initial dataset with amplitudes and phases, and then continues completing the dataset with high-resolution amplitudes from electron diffraction patterns alone. The phases for the high-resolution components are then generated or refined by phase extension or molecular replacement, similar to the procedures used in X-ray diffraction structure determination (Grigorieff, et al., 1996).

The atomic models for seven membrane proteins and tubulin have so far been determined by electron crystallography: BR (Henderson, et al., 1990) LHCII (Kuhlbrandt, et al., 1994), AQP1 (Murata, et al., 2000; Ren, et al., 2001), nAChR (Miyazawa, et al., 2003), AQP0 (Gonen, et al., 2005; Gonen, et al., 2004), AQP4 (Hiroaki, et al., 2006), MGST1 (Holm, et al., 2006), and Tubulin (Nogales, et al., 1998). Several other membrane proteins classified as transporters, ion pumps, receptors and membrane bound enzymes have been studied by electron crystallography at lower resolution allowing localization of secondary structure motifs such as transmembrane helices, and are likely to produce atomic models in the near future (Hirai, et al., 2002; Kukulski, et al., 2005; Schenk, et al., 2005; Tate, et al., 2003; Vinothkumar, et al.,

2005). Computer image processing in almost all above-mentioned cases has been performed with the so-called ‘‘MRC programs’’ for image processing (Crowther, et al., 1996). Computer processing of recorded images generally requires the determination of the crystal lattice using spots visible in the Fourier transform of the images. For the processing of electron crystallography images, this determination of the lattice vectors is usually done manually, and represents a time-intensive step, especially if many images are to be processed.

X-ray crystallography diffraction patterns show spots if their reciprocal position overlaps sufficiently with the Ewald sphere. The complex indexing process of XRD is done with robust automated software, such as the program DENZO as a part of the diffraction-image processing suite HKL2000 (Otwinowski and Minor, 1997; Otwinowski and Minor, 2001; Rossmann and van Beek, 1999), MOSFLM (Leslie, 1992), and d*TREK (Pflugrath, 1999). Important representatives of auto indexing algorithms are either based on Fourier analysis (Steller, et al., 1997) or direct indexing of difference vectors (Higashi, 1990; Kabsch, 1988; Kim, 1989). The general principle behind Fourier analysis methods is that the projection of a protein lattice in a chosen direction has a periodic distribution. The periodicity is determined by Fourier analysis. Structural details are encoded in the regular lattice in Fourier space. The basis vectors defining the reciprocal lattice in Fourier space are found by exploring all possible directions. In XRD auto indexing, Fourier-based methods need a few hundred spots to get reliable results, although in some favorable cases as few as 50 can be sufficient (Leslie, 2006). Difference vector methods first sort and estimate the crude base vectors according to their lengths and angle constraints. The selected bases are iteratively refined using estimated positions of observed diffraction spots. Both, Fourier-based and difference vector methods cannot identify single lattices in double-or poly-crystals. In this case, spots of a single lattice have to be selected manually beforehand.

There exist many algorithms for indexing diffraction spots in X-ray crystallography. However, little work is reported for that task in electron crystallography. Unlike X-ray diffraction patterns, electron crystallography gives real space images that have a very low signal to noise ratio, and the Fourier transformations show usually less than 100 visible spots. The common methods of difference vector analysis may not find the accurate basis. (Kabsch, 1993), has proposed a robust solution

that takes into account the moderate accuracy of the automatically determined lattice points and tolerates a small number of artifacts among them. This approach, however, cannot handle multiple crystal lattices.

We present here two new algorithms for determination of the reciprocal lattice of a 2D crystal image. These algorithms are also applicable to poly-crystal images. In addition, we present a refined tool for manual lattice identification in *2dx_image*.

2. The lattice determination algorithm

The first algorithm presented requires and makes use of a-priori knowledge of the lattice dimensions and lattice angle of the crystal sample in real-space, as well as of the sample tilt geometry under which the image was recorded. This algorithm determines the reciprocal lattice in the Fourier transformation (FFT) of the image in two steps: A first program *2dx_peaksearch* compiles a list of peak coordinates from the FFT, and another program *2dx_findlat* uses these peak coordinates to determine one or more lattices. If the unit cell parameters are unknown, a second algorithm is implemented in the program *2dx_getlat*, which guesses a lattice without any a-priori knowledge.

2.1. 2dx_peaksearch

As a first step, 2dx_peaksearch compiles a list of coordinates of peaks in the power spectrum (PS; the squared amplitude component of the calculated FFT) of an edge-tapered 2D crystal image (Figure 1 A). To obtain reliable peak spots, the pixels on the X-and Y-axis and at high resolution outside of a circular mask are replaced by the average grey value, and the resulting masked PS is low-pass and high-pass filtered to flatten potential variations from the contrast transfer function of the microscope, and to reduce the noise (Figure 1 B). The central X-and Y-axis are then again masked to eliminate potential ‘‘cross-wire’’ artifacts. Continuous streaks are then recognized by a pixel-wise neighbor search starting from the origin in the original PS and masked (Figure 1 C).

Since the bright and dark center areas in Figure 1 B are masked with the average, the contrast of the PS is enhanced in Figure 1 C. A set of peak coordinates is then obtained through two peak search processes, each of which finds the specified

numbers of peaks that are a local maxima in a 3 · 3 square, while ignoring other local peaks within a 10 pixel radius or found peaks. The first search process is used to find initial peaks. For each of these identified peak positions, a copy of the PS is shifted so that each peak becomes the new center. These shifted PS images are then averaged, weighted according to the central peak height. The effect of image shift to the distribution of peaks is shown in Figure 1 D. The resulting average PS image usually has a full coverage of low-resolution spots without any systematic absences, and therefore facilitates lattice determination. In addition, this average image has a better signal to noise ratio. This step of shifting-and-averaging PS follows the processing from the MRC program *autoindex* (Crowther, et al., 1996), which identifies the initial peaks using a static threshold and then searches for two independent low-resolution vectors directly in the averaged PS. However, *2dx_peaksearch* subjects this averaged PS image to a second peak search, with the list of peaks coordinates and their amplitudes (heights) written out for further processing by the programs *2dx_findlat* or *2dx_getlat*.

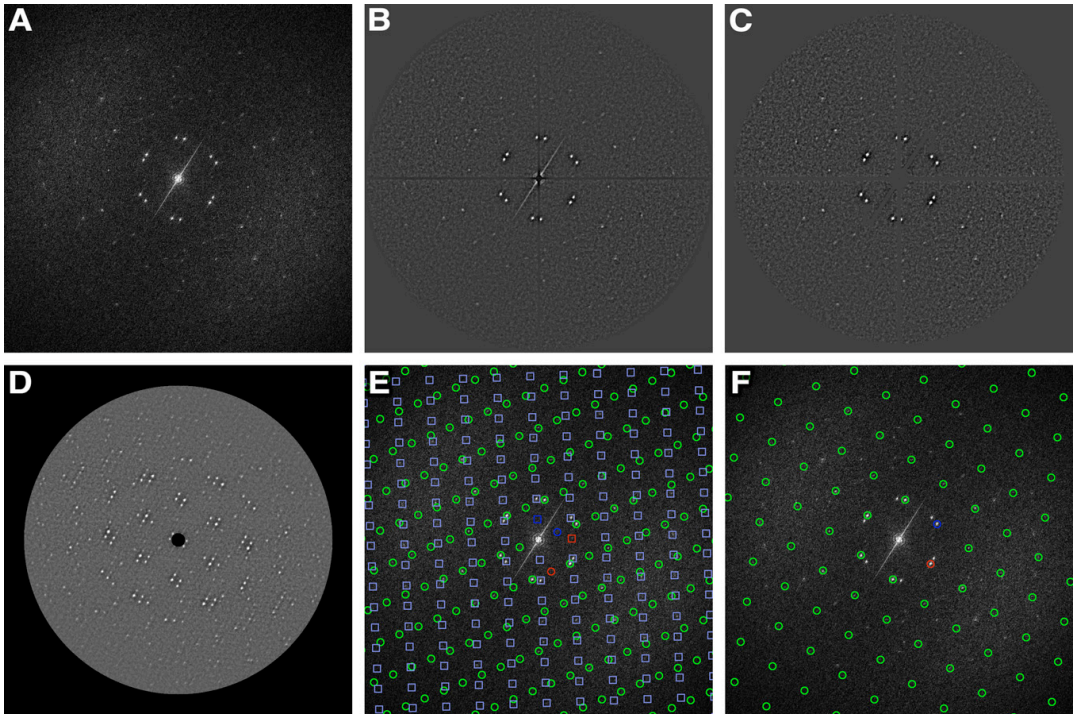

Figure 1: Automatic lattice determination of a crystal. (A) The original power spectrum (PS). (B) The program *2dx_peaksearch* replaces pixels on the X-and Y-axes and at high-resolution outside of a circular mask with the average grey value. This masked PS is then high-and low-pass filtered. (C) Streak artifacts together with X-and Y-axes are again masked with the mean in (C). (D) The origin-shifted and weighted averaged PS. (E) The first (circles) and second (squares) lattices are overlaid over the original PS, as automatically determined by *2dx_findlat*. The first vector of each lattice u = (1,0) is plotted in red, the second one v=(0,1) in dark blue. Note that the brightest diffraction spots in the original PS were correctly recognized by *2dx_findlat* as second order spots with coordinates (1,1), (2,0) and (-1,1), corresponding to a rectangular real-space lattice of a =81A°, b = 136A°, and c =900. (F) The program *2dx_getlat* in this example reported a lattice that covers most but not all lattice nodes, and is a wrongly indexed lattice due to the lack of additional information. This lattice, however, can still be transferred into the correct lattice as shown in (E) with the script "Evaluate Lattice" in *2dx_image* (see text).

2.2. 2dx_findlat

As second step, *2dx_findlat* uses the list of peak coordinates for the search of the best fitting lattice. This is done by calculating a hypothetical test lattice, based on the given real-space unit cell dimensions and included angle, considering the potential distortions due to the sample tilt. The crystal sample parameters are given as a, b for the unit cell dimensions and c for the unit cell angle, and the tilt geometry is defined as in the MRC software by TLTANG and TLTAXIS (Crowther, et al., 1996). The reciprocal lattice vectors $U=(u_1,u_2)$ and $V=(v_1, v_2)$ are then initially set to:

$$\begin{aligned} u_1 &= 0 \\ u_2 &= d/(a*mag*\sin(\gamma)) \\ v_1 &= d*\cos(\pi-\gamma)/(b*mag*\sin(\gamma)) \\ v_2 &= d*\sin(\pi-\gamma)/(b*mag*\sin(\gamma)) \end{aligned} \qquad \text{Eq. 1}$$

where d is the image pixel size and mag is the magnification.

This test lattice (U, V) is then rotated in the sample plane in small angle increments Θ to give (U', V'):

$$U' = \begin{pmatrix} \cos(\Theta) & \sin(\Theta) \\ -\sin(\Theta) & \cos(\Theta) \end{pmatrix} U \qquad \text{Eq. 2}$$

and equivalently for V', and the expected lattice distortion due to the tilt geometry is then applied, to give (U'', V'') for each rotation step:

$$\begin{aligned} U'' = &\begin{pmatrix} \cos(TLTAXIS) & -\sin(TLTAXIS) \\ \sin(TLTAXIS) & \cos(TLTAXIS) \end{pmatrix} \\ &\times \begin{pmatrix} 0 & 0 \\ \cos(TLTAXIS)^{-1} & \cos(TLTAXIS)^{-1} \end{pmatrix} \\ &\times \begin{pmatrix} \cos(TLTAXIS) & \sin(TLTAXIS) \\ -\sin(TLTAXIS) & \cos(TLTAXIS) \end{pmatrix} U' \end{aligned} \qquad \text{Eq. 3}$$

and equivalently for V''. In addition, the magnification mag is varied in a raster search of stepsize λ to accommodate potential inaccuracies in the scale of the lattice or magnification.

Given a set of n peaks $P_i = (x_i, y_i, z_i)$, with peak coordinates (x, y) and peak heights z, a score value F is determined for each rotated, distorted and magnification-varied test-lattice, by summation of the peak values of peaks that lay within a given radius of the listed peak coordinates.

$$F = \sum_{i=1}^{n} g(z_i) \qquad \text{Eq. 4}$$

where $g(z) = \begin{cases} z & ;\text{if spot on lattice} \\ 0 & ;\text{otherwise} \end{cases}$, and I denotes the peak numbers.

A peak with the coordinates (x_i, y_i) is accepted as "*on the lattice*", if

$$|P_1 - x_i| < \delta\sqrt{h^2 + k^2} \quad \text{, and } |P_2 - y_i| < \delta \text{,} \qquad \text{Eq. 5}$$

where $P_1 = u_1 * h + v_1 * k$, and $P_2 = u_2 * h + v_2 * k$, with h, k being the Miller indices of spots in the averaged PS, and δ being a tolerance constant. The condition Eq. 5 allows a larger deviation of spots from the lattice with increasing radius (resolution). This gives high-resolution spots a higher chance of contributing to the lattice vectors than low-resolution spots, in contrast to the method used by (Kabsch, 1993), that is using a constant threshold $|P_1 - x_i| < \delta$.

Any lattice candidate that covers a sufficiently large number n_{min} of low-resolution peaks is further refined. The peak spots in the lattice are first mapped into the lattice coordinates (RM_I, RN_I), using

$$RM_i = \frac{\frac{x_i}{v_1} - \frac{y_i}{v_2}}{\frac{u_1}{v_1} - \frac{u_2}{v_2}}, \quad RN_i = \frac{\frac{x_i}{u_1} - \frac{y_i}{u_2}}{\frac{v_1}{u_1} - \frac{v_2}{u_2}} \qquad \text{Eq. 6}$$

Then the refined lattice vectors are obtained by minimizing the residuals

$$\sum_{i=1}^{n} \Big[(Int(RM_i)u_1 + Int(RN_i)v_1 - x_i)^2 + (Int(RM_i)u_2 + Int(RN_i) \qquad \text{Eq. 7}$$

where Int(x) takes the nearest integer value of a real number x. For the refined lattice, a final scoring value F is then calculated as in Eq. 4.

From all the rotated, tilt-distorted, magnification-varied, and then refined lattices, that pair of lattice vectors with the highest scoring value is selected as the final lattice. By excluding the peaks that overlap with this identified lattice from the peak set, the algorithm is iteratively re-applied to the remaining peaks to find other potential lattices of multi-layered crystals or poly crystals (Figure 1 E).

2.3. 2dx_getlat

If the unit cell parameters of the crystal sample and/or the tilt geometry are unknown, an alternative algorithm is implemented in the program *2dx_getlat*, which is used to identify candidate lattice vectors for the refinement and scoring evaluation.

2dx_getlat requires only the peak list of the averaged PS generated by *2dx_peaksearch*. Using this, a set of difference vectors between certain low-resolution peak positions is generated, from which the most likely pair of lattice-generating vectors is found. This pair is then used as the basis for an iterative refinement process (usually requiring a maximum of 2 or 3 steps) assigning miller indices to found peaks which are then used in a least squares refinement of the basis vectors for the next round of refinement.

The strongest peak in the average PS image will necessarily fall on the strongest of any present lattices, and will very likely be a lattice-generating basis vector itself. As such, *2dx_getlat* compiles a list of every possible pair of points drawn from a list of all peaks occurring closer to the origin than the farthest of the k strongest peaks. Each such pair is then used to form basis vectors that generate separate candidate lattices, which are then individually compared against the full peak list. The parameter k can be changed to decrease the total calculation time and is usually set to 4 or 8 as the ideal basis vectors are almost always contained within the set of peak-vectors which are shorter than the longest of the 8 brightest peaks.

Lattice fitness for each candidate lattice, defined by vectors $\vec{u}$ and $\vec{v}$, is determined by first transforming each peak from the full peak list into a generalized *Miller space* via multiplication with the 2x2 matrix $[\vec{u},\vec{v}]$ (with $\vec{u}$ and $\vec{v}$ the lattice generating column vectors). The computed Euclidean distances from integer values for each transformed peak are then summed, with each term in the summation multiplied by the strength of the peak in question. For lattices where the included angle Θ between the vectors $\vec{u}$ and $\vec{v}$ is smaller than a certain limit ($\Theta \leq \Theta_{Max}$), a penalty weighting-factor of $e^{(\Theta_{Max}-\Theta)}$ is multiplied to each term. A value of $\Theta_{Max} = 10°$ was found suitable. This factor is applied to prevent trivial solution lattices, which have tightly spaced nodes and achieve artificially high apparent fitness. This problematic fitting occurs if the resulting lattice nodes approach continuity such that all peaks in the PS inevitably fall within reasonable distances of a node.

The two vectors, which generate the lattice with the lowest error are then transformed into the shortest lattice defining right-handed lattice that lies in the right half of the FFT: First, by inverting vectors with negative x-values, then by iteratively

either subtracting the shorter of the two vectors from the other if the included angle is smaller than 90º, or by adding them if the included angle is greater than 90º, until convergence to the shortest solution is reached. These resulting vectors are then ordered canonically, with the first vector defined as the one being closer to the negative Y-axis.

Using this basis, all peaks in the average PS are then transformed into the generalized Miller space, which is defined by the inverse of the 2 x 2 matrix of the newly found lattice vectors $[\vec{u},\vec{v}]$. These transformed peaks will then be assigned a given Miller index if they fall within ε of the integer values associated with this index. An ε of 0.0707 corresponds to 10% of the maximally possible error of $\frac{1}{\sqrt{2}}$, and is usually sufficient to exclude peaks from artifacts or other lattices. Finally, a peak assigned to a given index is discarded if another peak is found to lie closer to the index in question. Using the generated Miller index/peak position pairs, a least squares fit is then performed to refine the lattice $[\vec{u},\vec{v}]$. This process is then iteratively repeated until the method converges to a final lattice, which usually is reached within 2 or 3 iterations. As this method requires nothing beyond the peak list itself, it is highly sensitive to errors or absences found in this list.

2.4. Manual lattice indexing

To assist the user in manual indexing of the reciprocal lattice, we have implemented a lattice refinement function into the full-screen browser of *2dx_image* (Figure 2), with

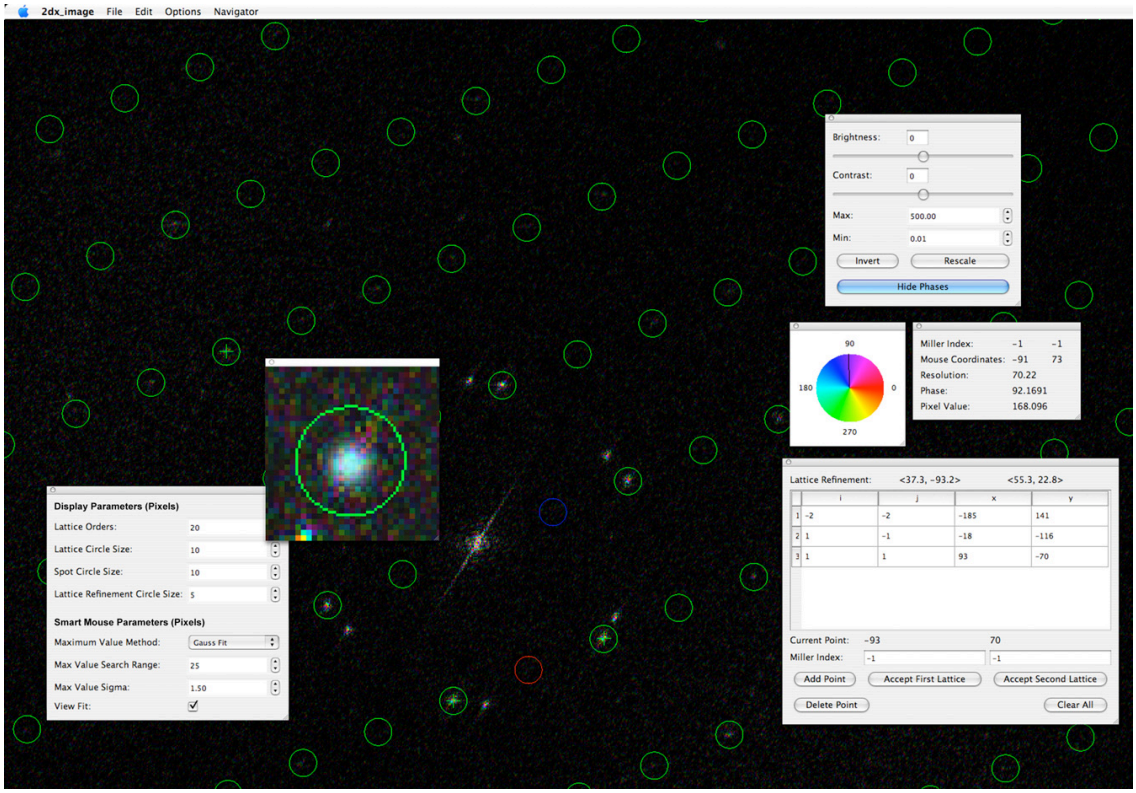

Figure 2: The manual lattice determination function of the full-screen browser in *2dx_image*. The calculated FFT of an image is displayed, here optionally with color-coded phase information (top right panel). The currently valid lattice is indicated by circles, with spot (1,0) in red, and (0,1) in blue. The information of the current mouse pointer location is displayed in the panel on the right. The user can manually identify individual peak positions by mouse-click and then assign Miller indices to the peak, while the most likely Miller indices based on the current lattice estimate are automatically pre-entered as default values (bottom right panel). Double-clicking close to a peak in the FFT will activate the smart-mouse function, which either selects the highest peak within a given radial distance, or will perform a Gauss profile peak search over the pixels within the given radius (here 25 pixels) and select the best fitting location as click-location. Parameters for the smart mouse function can be adjusted in the panel bottom left. The Gauss peak fit at the automatically re-centered location is displayed in the zoomed window (center left panel).

design and function largely inspired by the functions and development work found in the MRC program *Ximdisp.exe* (Smith, 1999). *2dx_image* allows displaying of phase information in the calculated Fourier transformation of an image as color code. In cases of very well ordered 2D crystal images, true diffraction peaks can be recognized by coherent phase information of neighboring pixels, while noise peaks have generally more random phases (Amos, et al., 1982). The manual lattice indexing function further supports the user by pre-entering the most likely Miller coordinates of the chosen peak location, so that in most cases the user can confirm that Miller index by simply hitting the ‘‘Enter’’ key on the keyboard. A *smart mouse* function in the full-screen browser is

activated upon double- clicking into the Fourier transformation. This function will then either correct the click-location to the strongest pixel within a given radius, or will perform a peak-search within that given radius with a Gaussian profile of a given half-width. In addition, *2dx_image* offers a set of lattice arithmetic functions in the ''Evaluate Lattice'' script, which allow the user to swap the primary and secondary reciprocal lattices, scale, skew, or rotate the reciprocal lattice, as well as invert the handedness of the current reciprocal lattice. This script also allows calculating the corresponding real-space lattice, and can give feedback on the agreement of the chosen lattice with the determined peak-locations in the averaged PS.

3. Results

The performance of these algorithms has been tested on a variety of images from non-tilted and tilted 2D crystals of various lattice dimensions and signal-to-noise levels.

Table 1 shows the results of applying the first algorithm (*2dx_findlat*) to different electron micrographs of 2D membrane protein crystals. For the first peak search in the original PS, 40 peaks were selected to generate the average shifted PS image and 140 peaks were selected from the latter to determine the lattice vectors. *2dx_findlat* found lattices in all test images using a stepsize of $\Theta = 0.1°$

Data set	Number of crystals	Tilt geometry [TLTANG, TLTAXIS][a]	Real-space lattice(a,b,γ)	Lattice Error [%]
AmtB	2[b]	0°, 0°	81Å, 136Å, 90°	1.51277 1.39484
BR	1	0°, 0°	62Å, 62Å, 120°	1.30567
AQP2	1	0°, 0°	98Å, 98Å, 90°	1.18784
AQP2	2[b]	33.85°, 63.04°	98Å, 98Å, 90°	0.75341 0.86334
AQP2	1	45.36°, 60.73°	98Å, 98Å, 90°	2.45832

Table 1: Performance of the automatic lattice determination by *2dx_findlat* for different images.
[a] Angle definition as in the MRC software (Crowther, et al., 1996).
[b] These lattices were overlaid and correctly identified in the same image.

for the rotational angle increment of the test lattice, a magnification variation of $\lambda = mag * 0.006$, a minimal number of required lattice peaks of n_{min} = 8, and a tolerance value of $\delta = 3$ pixels in reciprocal space (px^{-1}) for the spot acceptance.

The normalized root-mean-square deviation ($RMSD_N$) of the locations of peaks that fit the final lattice is calculated and normalized by the unit cell length

$$RMSD_N = \frac{2}{n}\sum_{i=1}^{n}\frac{\|L\cdot\vec{m}_i - \vec{x}_i\|}{\max\left(\|\vec{u}+\vec{v}\|,\|\vec{u}-\vec{v}\|\right)} \qquad \text{Eq. 8}$$

where $L=[\vec{u}\ \vec{v}]$, the matrix whose columns are formed by the reciprocal lattice vectors, and $\vec{m}_i=[h_i\ k_i]^T$ are the miller indices of spot *i* and $\vec{x}_i=[x_i\ y_i]^T$ represent the positions of the peaks in reciprocal space. In addition to this criterion, all the lattices were visually verified.

The combination of the streak-removal routine in *2dx_peaksearch* and the algorithm used in *2dx_findlat* was found to be insensitive to artifact peaks or streaks in the PS. The degree of tolerance of artifact peaks is related to the parameter δ, which should be chosen according to the confidence in the peak spot locations, i.e. the sharpness of the peaks and the signal-to-noise ratio in the PS. A small δ should be chosen in case of a PS with strong and sharp peaks, which then will give a high accuracy of the determined lattice vectors. A larger δ is recommended if the peaks in the PS are broad and noisy, to allow the algorithm to still find approximate lattice bases. Experiments were carried out with a dataset of peak spots of an AQP2 2D crystal with the vector length of 119 px^{-1}. The peak coordinates were deliberately distorted by Gaussian distributed offsets. *2dx_findlat* failed to find the correct lattice from peak locations that had been disturbed with a position deviation of σ =10 px^{-1}, when using a δ = 3.0 px^{-1}, but could still find the correct lattice with a δ = 6.0 px^{-1} (Table 2)(Table 3).

Peak deviations σ [px^{-1}]	Lattice tolerance δ [px^{-1}]	*U*	*V*	Lattice error [%]
2.0	3.0	18.895, -89.272	110.294, -4.216	2.78121
5.0	3.0	18.651, -89.157	109.903, -4.143	2.92306
8.0	3.0	18.782, -88.957	110.359, -4.229	3.24457
10.0	3.0	19.939, -98.402	116.967, -12.795	Wrong lattice
10.0	6.0	19.323, -90.239	110.596, -6.014	4.13556

Table 2: Lattice vector identification of an AQP2 image, where the peak positions were displaced by random amounts with a Gaussian distribution with standard deviation σ. The image was from a sample tilted at TLTANG = 45.360, with TLTAXIS = 60.730, with a = b =98A°, c =900. The tolerance value of d = 3.0 px^{-1} was sufficient to identify the correct lattice for smaller peak distortions, but a larger tolerance of d = 6.0 px^{-1} was required for strongly distorted peak coordinates. At larger peak deviations r, the resulting lattice unavoidably has a larger RMSDN. Nevertheless, the correct lattice was found as visually verified, except in the one mentioned case.

Protein	Tilt Angle [°]	Lattice Error [%]		Number of Peaks (used/allowed)		Node Density [nodes/peak]	
		2dx_findlat	2dx_getlat	2dx_findlat	2dx_getlat	2dx_findlat	2dx_getlat
AQP2	0.0	1.56784	1.64021	128/140	128/140	2.85953	2.85883
AmtB	0.0	1.51277	1.45631	74/300	74/300	1.35132	1.35226
AmtB*	0.0	1.39484	Wrong Lattice	50/140	28/140	1.99996	1.13391
BR	0.0	1.30567	0.95906	101/140	95/140	0.78599	0.86090
OmpF	0.0	1.95961	1.97492	69/140	68/140	0.76422	0.80033
Synthetic	0.0	0.76159	0.76123	140/140	140/140	1.12429	1.12407
V0	20.0	0.54396	0.54398	88/140	88/140	1.46933	1.46932

Table 3: Comparison of the performance of *2dx_findlat* and *2dx_getlat*. The algorithms were applied to 7 images from 5 different samples and different tilt angles. The Lattice Error is calculated as $RMSD_N$ of the distance of peak locations from the chosen lattice. Only peaks closer than a given threshold (see text) are included in this calculation. The Number of Peaks indicates how many peaks fulfilled the selection criteria (used), and how many peaks from the averaged PS were given to the algorithms (allowed). Node Density indicates the calculated number of lattice nodes for a given area relative to the number of peaks that fall within this area. (Ideally this number should be close to one and is a measure of whether the found lattice is too big or small by integer amounts. This measure also describes the number of layers present in the image for multi-layer crystals.) The computation times for *2dx_findlat* are about three orders of magnitude longer than for *2dx_getlat*, while the precision of both algorithms is comparable and usually better than a human operator can perform. However, in one image (AmtB*), *2dx_getlat* failed to recognize the correct lattice, and chose a lattice twice as large instead, see Figure 1. This still produced a lattice where 34 out of 140 spots could be used to report an acceptable $RMSD_N$, since this lattice had some 34 spots that were precisely fitting to this lattice. Nevertheless, the peak indexing was wrong, while *2dx_findlat* identified the correct lattice.

Tests were also done to investigate the tolerance to the errors in the tilt geometry and the unit cell length, using an image of a tilted AQP2 2D crystal with $a = b$ =98Å, γ=90º, and a tilt angle of 45º. The lattice could be correctly identified with tilt angle and axis variations of ±80, and with the unit cell length varying between 93 and 106A˚ (data not shown).

The difference-vector based algorithm implemented in *2dx_getlat* does not require a-priori knowledge of an expected lattice or tilt geometry. Since this second algorithm does not perform an exhaustive search, but rather guesses the lattice from direct calculations, this algorithm was found to be 100–1000 times faster in computational costs, and was still able in most cases to correctly identify the lattice. Only cases of PS with significant absences of lattice nodes caused *2dx_getlat* to fail to report the correctly indexed lattice. One such case is shown in Figure 1 F.

4. Discussions

The peaks from the averaged PS allow much better identification of the lattice than the peaks from the original PS. Our algorithm as implemented

in *2dx_peaksearch* follows the developments for the average PS that were also implemented in the MRC program *autoindex* before (Crowther, et al., 1996). In addition, *2dx_peaksearch* also removes streaks, which can arise from image edge effects, or, as in the case for Figure 1, from the edges of a negatively stained 2D crystal itself. The resulting averaged PS usually shows a pattern without absences of a much-increased signal-to-noise level, from which peak coordinates are evaluated for further processing.

The results presented here show that the informed exhaustive search implemented in *2dx_findlat* can accurately identify one or more lattices in images of 2D crystals, including those of tilted and polycrystalline crystals. The ability of distinguishing multi-layer crystals arises from the fact that *2dx_findlat* makes use of a-priori known information about the 2D crystal lattice, such as the unit cell dimensions and the tilt geometry. The required dimensions of the real unit cell can be obtained either manually from one easier-to-index non-tilted image, or by using the difference vector based algorithm implemented in *2dx_getlat*, which does not require any a-priori knowledge.

Identifying the lattices in a large number of 2D crystal images can be done first by indexing the lattice of one good non-tilted image with *2dx_getlat*, or manually. In most cases, *2dx_getlat* will find the correct lattice. However, in case of tricky lattices with systematic absences, *2dx_getlat* might report a lattice that assigns wrong indices to the correctly identified lattice nodes (as shown for example in Figure 1 F). In this case, the graphical user interface (GUI) of the *2dx_image* software (Gipson, et al., 2007) offers a script called ‘‘Evaluate Lattice’’, which allows modification of the identified lattice. This script allows the user to scale the lattice by doubling or halving one or both lattice vectors, skewing the lattice by replacing one vector with the sum or difference of both vectors, rotating the lattice clock or anti-clock wise (for square or hexagonal lattices), as well as changing the handedness of the lattice. This script can be used to transform the automatically determined lattice into the visually chosen one. This script also compares the current lattice with the list of closest peak positions to report the precision of the current lattice in the form of $RMSD_N$. It also reports the corresponding real-space lattice dimension and included angle, which can then be entered into the *2dx_image* database into the default configuration file *2dx_image.cfg*. This then

creates the a-priori knowledge that is required for using the exhaustive search algorithm in *2dx_findlat*, which should from then on automatically identify the correct lattice in images of highly tilted samples or those with tricky lattices. For difficult lattices in highly tilted images, *2dx_findlat* was usually capable of identifying the lattice more reliably than a manual user.

With the real-space lattice dimensions known, the script ‘‘Evaluate Lattice’’ also calculates the tilt geometry that would correspond to the given lattice when compared to the real-space lattice dimensions. This is done with a slightly adapted version of the program EMTILT, which is part of the MRC software (Crowther, et al., 1996; Shaw and Hills, 1981; Valpuesta, et al., 1994).

When large tolerance boundaries for the magnification and lattice tolerance are used, *2dx_findlat* might report the correct lattice, but have assigned the basis vectors incorrectly. For example, the reported lattice could have a wrong handedness assignment (i.e. *u* and *v* are exchanged). In the case of a tilted sample, this could then result in EMTILT reporting a wrong tilt geometry. Comparison of the EMTILT-determined tilt geometry with the defocus-gradient based tilt geometry allows identifying the correct lattice indexation. In practice, the user can use the script ‘‘Evaluate Lattice’’ to quickly cycle through different lattice indexations until the resulting tilt geometry agrees with the defocus-gradient determined geometry, as displayed in the *Status* panel in the *2dx_image* GUI.

The precision of the determined lattices as determined by *2dx_laterror* and measured in $RMSD_N$ for us was always higher for lattices determined by either of the automatic algorithms than for lattices that we indexed manually.

5. Conclusions

A set of three programs *2dx_peaksearch*, *2dx_findlat* and *2dx_getlat* was created, which allow the automatic determination of a 2D crystal lattice from a real-space image. *2dx_peaksearch* determines a list of Fourier peak coordinates, which are used by *2dx_findlat* to determine one or more lattices, using a-priori knowledge of the real-space crystal unit cell dimensions and the sample tilt geometry. If these are unknown, the program *2dx_getlat* can be used to rapidly determine the most likely lattice, and thereby obtain a guess for the unit cell dimensions. Alternatively, the user

can manually identify a lattice, while being supported by a set of functions in the full-screen browser of *2dx_image*. These programs are available as part of the *2dx* software package for the user-friendly image processing of 2D crystal images (Gipson, et al., 2007), and are available at http://2dx.org .

Chapter 3 Processing a Single TEM Image

Originally appeared as: Gipson, B., Zeng, X., Zhang, Z. & Stahlberg, H. 2dx—User-friendly image processing for 2D crystals. Journal of Structural Biology 157, 64-72, doi:10.1016/j.jsb.2006.07.020 (2007).

1. Introduction

Structural biology of membrane proteins is of central importance for health, disease, and the development of new drugs. Membrane proteins represent the majority of today's drug targets in pharmaceutical research. Nevertheless, the PDB database contains only a few hundred membrane protein structures, only a third of which can be considered unique conformations. Compared with the wealth of knowledge on the structure and function of soluble proteins, the low number of determined membrane protein structures stands in stark contrast to their biological importance.

Membrane protein structure determination faces several technical hurdles. Difficulties in over-expression, non-destructive detergent solubilization and gentle purification limit the amount of membrane protein sample available for structural

studies. Structure determination by X-ray diffraction (XRD)[1] of three-dimensional (3D) crystals, nuclear magnetic resonance (NMR), and cryo-electron microscopy (cryo-EM) of two-dimensional (2D) crystals has revealed an amazing array of structural concepts and mechanisms that nature employs to solve the challenging tasks that membrane proteins perform. Recent highlights include the 1.35A˚ structure by XRD of the ammonium channel AmtB (Khademi, et al., 2004), the structure of the water channel Aqp0 from cryo-EM at 1.9A˚ and XRD at 2.2A˚ resolution (Gonen, et al., 2005; Gonen, et al., 2005; Harries, et al., 2004) and the structure of Mistic (Roosild, et al., 2005) by NMR (Wuthrich, 1998), to name a few.

Electron crystallography uses cryo-electron microscopy to study the structure of membrane proteins that are reconstituted into phospholipid bilayers and laterally crystallized into 2D membrane protein crystals. Atomic models for seven membrane proteins and tubulin have been determined by electron crystallography: BR(Henderson, et al., 1990) LHCII (Kuhlbrandt, et al., 1994), AQP1 (Murata, et al., 2000; Ren, et al., 2001), nAChR (Miyazawa, et al., 2003), AQP0 (Gonen, et al., 2005; Gonen, et al., 2004), AQP4 (Hiroaki, et al., 2006), and MGST1 (Holm, et al., 2006), and Tubulin (Nogales, et al., 1998). In addition, several low-resolution structures of transporters, ion pumps, receptors and membrane bound enzymes, that reveal secondary structural motifs such as transmembrane helices are likely to produce atomic models in the near future (Hirai, et al., 2002; Kukulski, et al., 2005; Schenk, et al., 2005; Tate, et al., 2003; Vinothkumar, et al., 2005; Aller and Unger, 2006).

The crystallization of membrane proteins in a 2D array within the lipid bilayer represents a valuable alternative route for structure determination. Electron Crystallography has matured into a methodology that allows the determination of membrane protein structures at a resolution of 3Å or better (Grigorieff, et al., 1996; Gonen, et al., 2005; Mitsuoka, et al., 1999). 2D membrane crystals offer the possibility of assessing membrane-inserted protein conformations. Existing 2D crystals can be incubated with ligands or other protein binding partners, or they can be exposed to different buffer conditions, and the structure of the complex or altered conformation can

[1] **Abbreviations used: 2D, two-dimensional; 3D, three-dimensional; XRD, X-ray diffraction; NMR, nuclear magnetic resonance; cryo-EM, cryo-electron microscopy; MRC, Medical Research Council.**

then be studied by electron diffraction. However, electron crystallography remains a labor-intensive method: beam-induced charging and/or drumhead-type movement of tilted samples in the electron microscope still affect the success rate for recording high-resolution images—despite recent advances though the use of the SpotScanning method (Downing, 1991) and/or the sandwich sample preparation method (Gyobu, et al., 2004). During the screening of crystallization conditions, high-resolution data collection or computer image processing, the lack of automation also requires time-intensive operator interaction.

Computer image processing of electron crystallography data in almost all the aforementioned cases has, to date, been performed by the ‘‘MRC programs’’ for image processing (Crowther, et al., 1996). These ‘‘MRC programs’’ are a compilation of individual programs, most written in Fortran-77, that were designed to process images of two-dimensional crystals as well as electron diffraction patterns (Henderson, et al., 1990; Kuhlbrandt, et al., 1994; Murata, et al., 2000; Unwin and Henderson, 1975). While this software collection offers a vast repertoire of tools for the processing of 2D crystal images, learning how to employ these programs is time-intensive, and the their usage involves a high amount of direct user interaction.

The MRC programs and bsoft programs (Heymann, 2001) are a collection of stand-alone programs written in Fortran-77 or C/C++. These programs need to be executed either manually, one-by-one in a terminal window, or from a shell script. The later has the advantage of facilitated usage, along with high flexibility and adaptability, but maintaining such scripts can be labor intensive. The execution speeds of computational tasks in scripts are slow, and readability of the scripts and interpretation of results in the form of log-files can be difficult.

A number of other software packages exist for the processing of 2D crystal images. SPECTRA from the ICE package facilitates the usage of the MRC software (Schmid, et al., 1993; Hardt, et al., 1996). Wilko Keegstra at the University of Groningen, The Netherlands, is currently developing the Groningen Image Processing Package (GRIP) that can also interface with the MRC software (unpublished). The Image Processing Library and Toolkit (IPLT) is a new ground-up image processing development for electron crystallography (Philippsen, et al., 2003).

We present a new software system, 2dx that is designed for the electron crystallography community. The purpose of this software system is to facilitate and streamline the processing of electron crystallography data, by providing a user-friendly interface, user-guidance throughout data processing, and a high degree of automation. In the current implementation, 2dx utilizes programs from the MRC software, as well as additional stand-alone programs written specifically for interaction with the 2dx environment as well as providing additional functions and features. 2dx is highly dynamic and can easily be used in conjunction with other image processing packages, including IPLT (Philippsen, et al., 2003), bsoft (Heymann, 2001), and/or Spider (Frank, et al., 1996). 2dx is developed under the Gnu Public License (GPL), and is freely available as open source. 2dx is available at http://2dx.org and runs natively on Mac OSX and Linux/X11 (Linux, IRIX and other Unix variants).

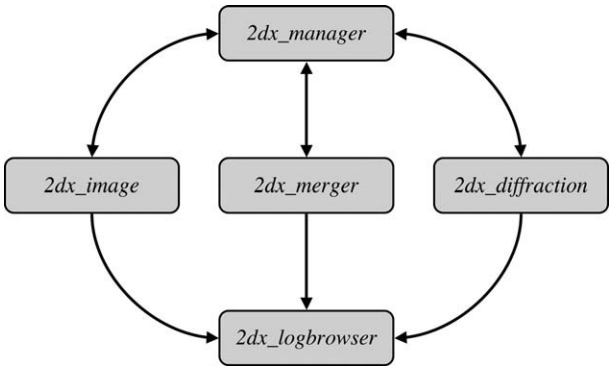

Figure 3: The five programs of the 2dx package. *2dx_manager* coordinates the project, and launches the *2dx_image* and *2dx_diffraction* programs for the processing of images and diffraction patterns. Data will be merged by *2dx_merger*. *2dx_logbrowser* assists in the evaluation of the log-files.

2. Software design

2dx is a collection of five programs, *2dx_manager*, *2dx_image*, *2dx_diffraction*, *2dx_merger* and *2dx_logbrowser* (Figure 3). *2dx_manager* assists in the management of an image-processing project, which typically amounts to 3D structure determination of one membrane protein. *2dx_manager* maintains control over the existing data (images or diffraction pattern), their parameters (e.g., resolution, sample tilt geometry) and results. *2dx_manager* also launches other programs such as *2dx_image* and *2dx_diffraction* as interactive instances, or submits them to a distributed computing

cluster. *2dx_merger* manages 2D and 3D merging of the data. The *2dx_diffraction* program will perform the computer evaluation of electron diffraction patterns where *2dx_image* performs the processing of one image of a 2D crystal. *2dx_logbrowser* assists in analyzing the log-files that result from processing. The *2dx_merger* and *2dx_diffraction* programs are currently under development, while *2dx_manager* at present assists only in the initialization of a project directory structure (Figure 4). Here we introduce the programs *2dx_image* and *2dx_logbrowser*.

2dx_image and *2dx_logbrowser* are written in C++, and are based on the Qt Open Source Edition for cross-platform software development (Trolltech, http://www.trolltech.com), and FFTW(Frigo and Johnson, 2005; http:// www.fftw.org). 2dx as well as FFTW are available under the GNU GPL, and Qt is available open source, free of charge for non-commercial software.

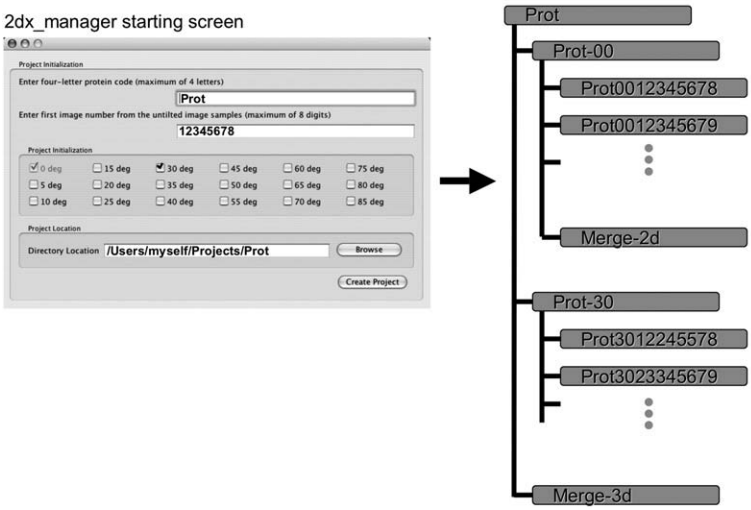

Figure 4: *2dx_manager* in its current state assists in the generation of a default directory structure for a protein project, which here is called "Prot". Images should reside in their own dedicated directory (e.g.,Prot0012345678), which are grouped according to their nominal tilt angles (here: non-tilted in "Prot-00", and 30-deg tilted in "Prot-30"). Merging directories for the 2D merging of the non-tilted images, and for the 3D merging of the entire project are also provided.

The central philosophies guiding the development of the 2dx software have been ease of use and independence from particular algorithmic implementations and/or platforms. To this end we have developed the software to be intuitive and automatic. That is, users do not need advanced knowledge about the technical details of the image processing in order to process a 2D crystal image in a straightforward way. Ideally,

once a few essential parameters, such as the image file name and other parameters concerning the protein, are known and submitted, the software is capable of processing an image from start to finish with no further need for user interaction. Unfortunately, such automated designs easily lead to a trade-off between ease-of-use and processing precision. 2dx is therefore designed with a high degree of flexibility and customizability, rooted in ground-up platform independence.

Excellent image processing packages, such as MRC, IPLT, and bsoft contain numerous efficient, rigorous routines, each with their own benefits. We have kept the 2dx front-end GUI implementation independent from the software backend, relying on low-level algorithmic templates (reminiscent of c-shell style scripts), which organize processing procedures around modular programs. A processing routine is then subject only to the confines of the modules on which it depends, each of which can be easily replaced as needed. Further, since procedural level changes amount only to modification of template files, large structural changes in workflow become little more than script editing.

The defining features of a template file include a variables section, describing parameters necessary for the execution of the script; a script section, describing the actual program flow; and a series of simple semaphore, which allow communication with the GUI front end (Figure 5).

Parameters found in the variables section of a template are drawn from a configuration file containing all variables necessary to execute the script. Variables appearing in this configuration file are distinguished by unique identifiers and defined by a human readable data structure, which describes every aspect of the variable's appearance in the GUI. This structure allows control over how the user will

```
#!/bin/csh -e
##############################################################################
# Title: Determine Spacegroup  (simplified version)                          #
##############################################################################
#
# SORTORDER: 45
#
#=============================================================================
# SECTION: Allspace parameters
#=============================================================================
#
# LABEL: Spacegroups to test
# LEGEND: Choose the spacegroup groups that should be considered.
# EXAMPLE: test_spacegroups_val= "ALL"
# HELP: http://2dx.org/documentation/variable/allspace
# TYPE: Drop_Down_Menu "ALL;HEXA;SQUA;RECT;OPLI;TWO;THRE;SIX;FOUR"
set test_spacegroups_val = "ALL"
#
# GLOBAL: RESMAX
#
#$end_local_vars
#
set RESMAX = ""
set realang = ""
set realcell = ""
#
#$end_vars
#
\rm -f 2dx_allspace.results
#
echo "<<@progress: 10>>"
#
2dx_allspacea.exe << eot
${test_spacegroups_val}
T T F 4000                          ! SEARCH,REFINE,TILT,NCYC
0. 0. 0. 0.                         ! ORIGH,ORIGK,TILTH,TILTK
3.0,121                             ! STEPSIZE, PHASE SEARCH ARRAY SIZE
${realcell} ${realang} 200.0 ${RESMAX} 2.0, 200.0
F 0 5                               ! ILIST,ROT180,IQMAX
"SPCGRP.txt"
eot
#
echo "<<@progress: 80>>"
#
set SPCGRP = `cat SPCGRP.txt | cut -d\  -f1`
set PHAORI = `cat SPCGRP.txt | cut -d\  -f2-`
#
echo "::Spacegroup ${SPCGRP} is best."
echo "set SYM = ${SPCGRP}" >> 2dx_allspace.results
echo "set phaorip1 = ${PHAORI}" >> 2dx_allspace.results
echo "# IMAGE: outputimage.mrc" >> 2dx_allspace.results
#
echo "<<@progress: 100>>"
```

Figure 5: An example for the script template used by *2dx_image*. This c-shell template contains code words that control the widget generation in the graphical user interface (GUI) of *2dx_image*. "# Title:" and "# SORTORDER:" allow the definition of the title and order under which the script will appear in the GUI. "# SECTION:" signals the beginning of a new parameter section in the central panel of the GUI. The following 6 lines define one parameter entry for that pane: LABEL is the title of the parameter, LEGEND is the short explanation in the pop-up window associated with that parameter. EXAMPLE allows suggesting the syntax for a correct entry. HELP defines the web page, where online help can be found. TYPE instructs the GUI to construct the widget for this parameter in a specific way (here as Drop_Down_Menu). Finally, "set test_spacegroups_val =" defines the default value for that parameter. The following section with the code words "# GLOBAL:" requests other globally known parameters that should appear in the GUI (here only RESMAX). This section is terminated with the flag "#$end_local_vars". The following section requests parameters,

which the GUI will enter when translating this script template into the actual executable script. In this example, "realang" and "realcell" will not be editable in the GUI for this script, because they are not declared as "# GLOBAL:". However, these values will available for this script. This section terminates with "#$end_vars". The remainder of the script template is a normal c-shell script. The output of the command echo ">@progress: 10»" will cause the GUI to advance the progress bar, setting it here to 10% of the execution progress. Logfile output starting with "::" will be displayed by the GUI also under only the lowest verbosity settings. ":" defines moderate verbosity output, and lines without leading colons appear only under highest verbosity settings. Output into the file 2dx_allspace.results (<filename>.results) in the form of, for example, "echo "set SYM = ${SPCGRP}" »2dx_allspace.results" would return a new value for the parameter SYM to the GUI, which would store it in the *2dx_image*.cfg database. The results file can also be used to flag image files that should appear in the list of images for inspection. "echo "# IMAGE: outputimage.mrc" »2dx_allspace.results" in this example instructs the GUI to include this image file in the list of viewable images. The final command "echo ">@progress: 100»" advances the progress bar to 100%.

interact with the variable through the front end, in addition to providing basic information about the variable itself. The variable's 'LEGEND' value, for instance, contains a brief line of text describing the meaning of the parameter, whereas the 'HELP' value contains an html link, which points to a more detailed discussion of the variable on the 2dx.org web server. Each help description page on the 2dx.org server features a discussion thread in the form of an online blog, so that users can discuss their experiences or questions regarding the 2dx.org documentation (Renault, et al., 2006).

Since the content of the configuration file defines the appearance of the *2dx_image* GUI (and is designed with readability in mind) adding, deleting and reorganizing processing parameters and their layout can be easily achieved. Even large structural changes in the layout and appearance of the GUI can be done by editing this configuration file.

The executable portion of any template generally corresponds to a c-shell script in flow and syntax. Since neither a parameter section nor use of semaphore is required for any template, the user is free to incorporate any existing c-shell script they wish into 2dx with a minimum of alteration.

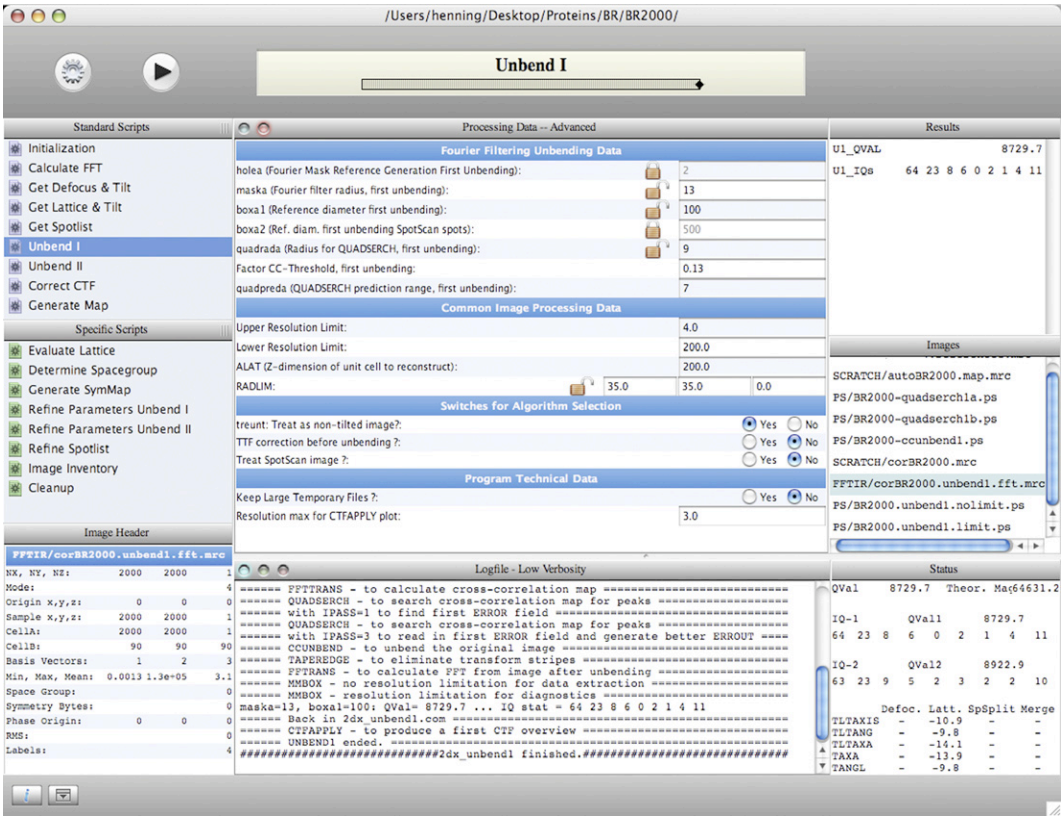

Figure 6: The *2dx_image* graphical user interface. For a description see text. The bottom left pane displays either the image file header information, as in this example, or the image thumbnail preview.

3. Graphical user interface and workflow

In its current state, the *2dx_manager* assists in the generation of a directory structure for a 2D crystal project (Figure 4). A four-letter project code and the image number of the first non-tilted image are requested, together with a selection of sample tilt ranges that the user intends to use for data collection. The *2dx_manager* then initializes the directory structure as reproduced in Figure 4, to be used in the following conventions: Each 2D crystal image should be processed in its own directory, where the image file, its parameter files and output files are stored. Image directories are grouped according to their nominal tilt angle, starting with one directory for all images of non-tilted samples. Residing in the image directories of non-tilted samples is a merge-directory that can be used to generate a 2D merge dataset. Other tilt-angle sessions are organized in their respective directory structures, and the entire project is merged into a 3D dataset in the highest-level merge directory.

The purpose of *2dx_image* is the processing of one 2D crystal image, which resides in its own dedicated directory. *2dx_image* maintains a simple image database in the form of a structured text file (*2dx_image*.cfg), where all parameters relevant to the processing of that image are stored. Certain project-wide ‘‘global’’ parameters in this text file, such as the crystal symmetry or the real-space unit cell dimensions of the protein crystal, are synchronized at run time of the *2dx_image* program with a project-wide default configuration file.

The *2dx_image* main graphical user interface is reproduced in Figure 6. The top section houses buttons to ‘‘Save’’ the image parameter file, and to ‘‘Execute’’ one or several selected script(s). This section also displays the currently running script, and its execution status. The central pane, entitled ‘‘Processing Data’’, displays the image and processing parameters—all of which can be edited, saved, and optionally locked to protect against accidental changes by the user or from changes made by executed programs. Two user-levels can be chosen, to allow access to only the most significant parameters (‘‘simple’’), or to the full-parameter set (‘‘advanced’’).

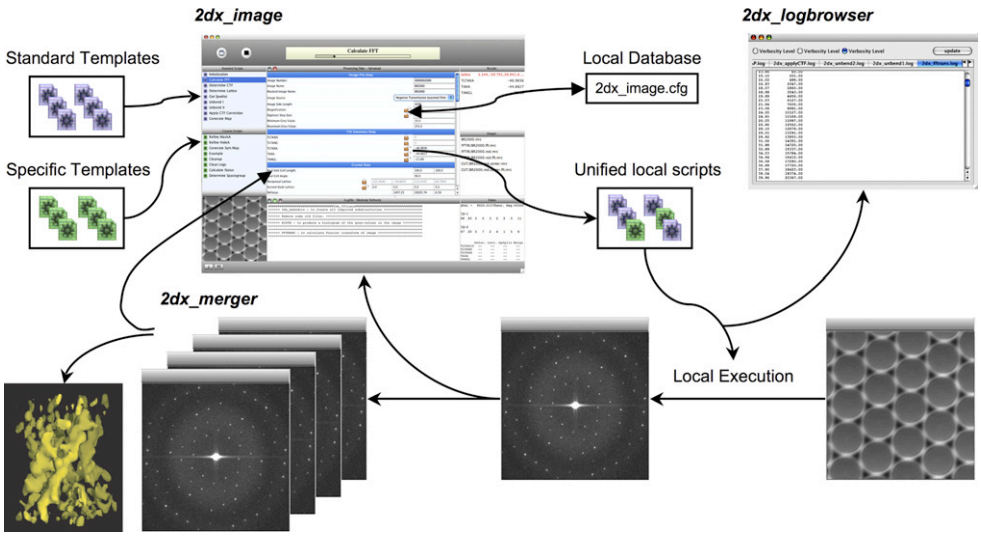

Figure 7: The *2dx_image* workflow. *2dx_image* maintains a local database (*2dx_image*.cfg), with which script templates are translated into local executable scripts. These scripts can be launched, with their output and processing results then channeled back into *2dx_image*.

The top left pane entitled ‘‘Standard Scripts’’ lists a set of scripts that are usually sequentially executed when one image is to be processed. This can be done by selecting one script at a time (mouse-click on the script), and executing it via the

‘‘Execute’’ button. Alternatively, any subset of these scripts can be selected and automatically executed sequentially. Upon execution, *2dx_image* loads the template, complements it with the template-requested data from the database, creates an executable c-shell command file in the local directory, and launches that command file as an independent child process (Figure 7). The progress of the executing job is monitored and graphically displayed in the top-banner of the *2dx_image* program, which also allows the user to halt the running task. Output of the running job is displayed in the lower central pane of the *2dx_image* GUI, entitled ‘‘Logfile’’. Display of the job output has one of three verbosity levels, with the user being able to switch between levels by selecting one of three buttons on the top banner of that pane. Double-clicking this top banner launches the *2dx_logbrowser*, which allows the user to browse all available logfiles, each with the choice of three verbosity levels.

Running jobs can signal to the *2dx_image* GUI the names of important image-files, by including the label ‘‘# IMAGE:’’ in the log-output. Image files on the hard-drive that are flagged in this way are listed in the panel on the center right of the *2dx_image* GUI, entitled ‘‘Images’’. For example, the log-file entry ‘‘# IMAGE: SCRATCH/ corBR2000.mrc’’ would add that MRC-format file to the list of image files in that pane. Currently, both MRC-format and PostScript format files are viewable. Their format is recognized by the ending of the file name. Selecting one of these image files in the *2dx_image* GUI launches the creation of a thumbnail preview of that image, which is displayed in the lower left pane of the GUI. Alternatively, the user can switch from the thumbnail view to a header-view, by selecting the button at the lower left end of the GUI, entitled ‘‘i’’. Double-clicking an image name or the thumbnail preview launches a full-screen image browser for that image (Figure 8). This browser displays images and Fourier transformations, and also allows the user to manually adjust or edit, and save the reciprocal lattice in a Fourier transform, the Fourier spotlist, and the defocus.

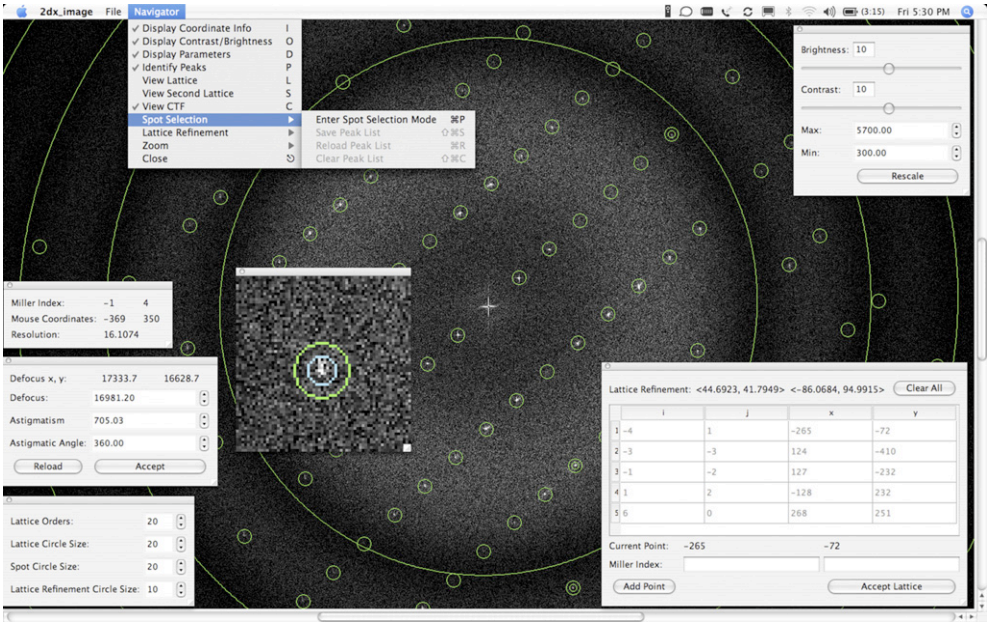

Figure 8: The *2dx_image* full-screen browser, here displaying a Fourier transformation of an image. A pull-down menu allows activating various panels. The Coordinate Info panel displays the current mouse coordinates, the corresponding resolution, and the Miller indices of the closest lattice spot. The Contrast/Brightness panel allows adjusting the display parameters. The Parameters panel allows defining the dimensions of the different symbols. The spots in the current spotlist are displayed when the "Peaks" option is selected. The current Lattice and Second Lattice can be displayed, as well as the Thon rings of the contrast transfer function (CTF), which is defined by the given defocus and astigmatism values. A Spot Selection mode and a Lattice Refinement mode allow manually editing or refining the spotlist and lattice vectors. The entire display can be zoomed up or down. Following the excellent development in the MRC program *Ximdisp*.exe, a mouse-activated local zoom window can be produced with the mouse. Determined values for the spotlist, the lattice vectors or the defocus values are automatically transferred back into the *2dx_image* GUI.

During the execution of jobs, determined or refined parameters can be returned to the 2dx_image database, by writing them into a results file. A script ''2dx_all-space.com'' for example can output a determined space group and phase origins by creating a file named ''2dx_allspace.results'', which should contain entries of the form ''set SYM = p3'' (Figure 5). *2dx_image* will then interpret the results file and update the database accordingly. Selection of the ''lock'' icon located next to the entered values in the central pane will prevent a running script from updating specific parameters in the database. This would, for example, be useful if a user spent time and energy to manually determine the reciprocal lattice of a difficult Fourier transformation, and did not want the automatic lattice determination routine to overwrite the manually fine-tuned lattice vectors.

The bottom right pane of the *2dx_image* GUI displays the processing ‘‘Status’’ for the current image. This pane summarizes the most important parameters of the current image-processing job, which are maintained in a file named ‘‘*2dx_image*.status’’. These parameters include the quality value of the entire processing (QVal, see below), the refined theoretical magnification (for comparison with the nominal magnification, to indicate possible errors in pixel size, magnification or lattice vector dimensions), the statistic of IQ-values as defined by R. Henderson et al (Henderson, et al., 1990; Henderson and Unwin, 1975), as well as the five parameters describing the tilt geometry of the sample and the crystal, as determined by four different methods.

The data in this ‘‘Status’’ pane informs the user about the status of the processing of this image, and indicates possible errors in the processing. Discrepancies in the tilt geometry between values determined from defocus variations across the image, distortions of the reciprocal lattice, spot-splitting due to the tilted transfer function (Henderson, et al., 1990) and those refined during merging can be identified here.

Most of the fields, labels and names in the *2dx_image* GUI have a context-sensitive right-mouse-click activated help function. Right-mouse-clicking a variable name in the parameter pane, for example, produces a window with a short explanation of that variable, its purpose and the units for the value, as well as an Internet link to the documentation in the corresponding web page on the 2dx.org server.

4. Scripting conventions

The entire *2dx_image* construction is kept as user-adjustable and flexible as possible. The *2dx_image* database named ‘‘*2dx_image*.cfg’’, for example, is kept in a self-explanatory, editable text format. A user can easily add or delete variables, define their format (e.g., float, integer, pull-down menu, Boolean switch, etc.), and define the corresponding help information and web-page link. The standard and custom scripts can be modified, extended or replaced by other scripts that might launch other user-defined software. The format for reporting data to the *2dx_image* database (*.results) and for updating the status window is self-explanatory and easy to implement into existing software/scripts (see also Figure 5).

5. Implemented algorithms

In the current state of *2dx_image* we have provided a collection of standard scripts for the processing of 2D crystal images, as we use them in our laboratory—most of which are based on the MRC programs. We also added functions for automatic lattice determination (Zeng, et al., 2007), spot-list determination, and crystal masking, as well as for the determination of the tilt geometry (using ctffind2; (Grigorieff, 1998)). The need for the determination of the optimal reference patch location is eliminated by choosing a one pixel diameter Fourier mask in the first unbending round (unbend1): The resulting reference map is of low quality, but shows no deviation over the entire map. The reference patch can therefore be chosen in the center of that map. Further unbending rounds (unbend2) with wider Fourier masks will then retrieve the structure's underlying signal, while keeping the reference location in the center of the image.

Additional scripts are available in the lower left pane in the *2dx_image* GUI, entitled ''Specific Scripts''. For example, the script ''Determine Spacegroup'' allows the determination of the symmetry space group and/or phase origin for a given symmetry (using allspace; (Valpuesta, et al., 1994)).

6. The quality value QVal

The scripts calculate a single, one-dimensional, value QVal that attempts to describe the quality of the entire image processing. While the IQ-values that were introduced by R. Henderson (Henderson and Unwin, 1975) are defined as a function of the intensity ratio between a specific reciprocal spot and its local background, the QVal addresses the entire image-processing phase. QVal is calculated by an empirical formula that combines different performance measures and indicators into a single number. The library function qval(nIQ,rscamax) resides centrally in the file .../2dx/mrc/lib/2dx_func.for, and allows user fine-tuning of the formula used in calculating the QVAL, when a different function is desired. We currently employ the formula

$$QVAL = R * \{IQ1 * 17.5 + IQ2 * 12 + IQ3 * 8 + IQ4 * 5.2 + IQ5 * 3.3 + IQ6 * 2 + IQ7\} / 500.0$$

where IQ1 to IQ7 denotes the number of IQ-values of the categories 1 to 7, and R denotes the height of the central pixel of the averaged Fourier peak profiles, as

calculated, for example, by the MRC program mmbox. Division by the calibration factor 500.0 is done to allow easier display. This empirically derived formula corresponds to a more-than-linear weighting of the calculated diffraction power.

The QVal can also be reduced in the form of ‘‘penalty’’ points, if for example discrepancies between tilt geometries determined by different methods are encountered, or when other image processing parameters or results appear ‘‘suspicious’’.

The QVal can then be used by the user or by the *2dx_manager* program to judge the reliability or usability of an image for inclusion in the merging process. The QVal can also be used by the *2dx_image* program to automatically refine the image-processing task. The ‘‘Specific Scripts’’ ‘‘Refine Parameters Unbend I’’ and ‘‘Refine Parameters Unbend II’’ optimize the QVal during a systematic variation of parameters, and automatically determine the parameter combination that results in the highest QVal. Sensitive parameters like the Fourier mask radius (e.g., maska) or the diameter of the reference patch for cross-correlation in the unbending procedure (e.g., boxa1) can be systematically tested, and the optimized parameter setting can then automatically be saved and used in future processing.

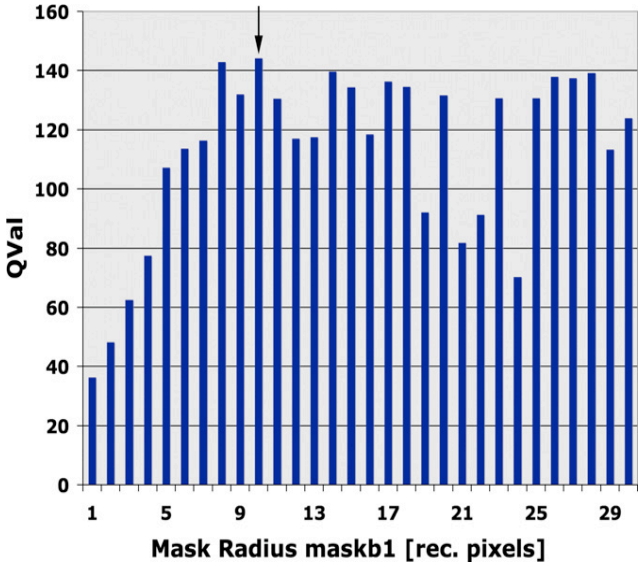

Figure 9: The QVal-based parameter refinement. A search for the best parameter for maskb1 resulted, in this example, in an optimal QVal value with maskb1 = 10.

The parameter refinement scripts are computationally intensive, and can be applied to one representative image. The identified optimal parameters can then be saved as future default parameters for the processing of other images in the same tilt-angle group. Figure 9 displays the result of a refinement of maskb1, for which the QVal was calculated for values between 1 and 30. Attention should be paid to exclude unrealistically small reference sizes, which can produce high QVals due to noise

correlation. This ‘‘overfitting’’ of the unbending procedure would produce good IQ statistics and a high QVal, but does not improve the resolution of the image processing (see also (Grigorieff, 2000)). The single QVal-based refinement strategy can easily be adapted by the user to refine parameters for other scripts and/or programs, such as those that use bsoft, SPIDER, IPLT or other MRC programs (Heymann, 2001; Philippsen, et al., 2003; Frank, et al., 1996; Crowther, et al., 1970).

7. Conclusions

2dx is a user-friendly software system for electron crystallography. In its current state the components *2dx_image* and *2dx_logbrowser* allow the processing of 2D crystal images. Future development for electron diffraction pattern evaluation and 3D merging is under way. 2dx is currently employed to run the ‘‘MRC programs’’ (Crowther, et al., 1996), but can be used in conjunction with other systems. While the focus of 2dx lies on user-friendliness, user-guidance, transparency, processing efficiency, and automation, we have implemented routines for the automatic determination of the crystal lattice, determination of the tilt geometry, spot-list creation, and crystal masking. Most of these implementations are based on the excellent developments of others (Henderson, et al., 1990; Henderson and Unwin, 1975; Heymann, 2001; Philippsen, et al., 2003; Grigorieff, 1999) and our aim has been to merge these into a user-friendly and efficient software system. Contributions in the form of additional user scripts or suggestions for additional functions are most welcome.

Chapter 4 Merging 2D Data into a 3D Dataset

Orginally appeared as: Gipson, B., Zeng, X. & Stahlberg, H. 2dx_merge: data management and merging for 2D crystal images. Journal of Structural Biology 160, 375-384, doi:10.1016/j.jsb.2007.09.011 (2007).

1. Introduction

Electron crystallography can determine the structure of membrane proteins that are reconstituted into a lipid membrane and two-dimensionally (2D)[2] crystallized in the membrane. These crystals are imaged by transmission electron microscopy, and the 3D structure of the membrane proteins can be reconstructed by computer image processing (Renault, et al., 2006). This method has led to the determination of several atomic resolution structures of membrane proteins, ranging from the work of (Henderson and Unwin, 1975), to the 1.9Å 3D structure of Aqp0 by (Gonen, et al., 2005).

[2] **Abbreviations used: 2D, two-dimensional; 3D, three-dimensional; XRD, X-ray diffraction; cryo-EM, Cryo-electron microscopy; MRC, Medical Research Council.**

Detergent-solubilized and purified membrane proteins can be crystallized as 2D sheets by dialysis or other methods (Jap, et al., 1992; Kuhlbrandt, 1992; Levy, et al., 2001; Remigy, et al., 2003; Signorell, et al., 2007) and high-resolution cryo-electron microscopy (cryo-EM) images are recorded from non-tilted and tilted preparations of the planar membrane protein crystals (Gyobu, et al., 2004; Fujiyoshi, et al., 1991; Schmidt-krey, 2007). Images are recorded on CCD cameras, or on photographic film with subsequent digitization. The resulting 2D image datasets are usually of large pixel dimensions and may contain high-resolution information covered by very high levels of noise.

Computer image processing can be applied to extract the structural information from the recorded images. Henderson at the Medical Research Council in UK, and coworkers have developed a powerful suite of programs for this purpose, which is available as the so-called MRC software (Crowther, et al., 1996). The MRC programs are a collection of programs, mostly written in Fortran-77, which perform various image processing and data merging operations. The basic algorithm for the processing of one image consists of determining the 2D crystal distortions that might be present in one image, and correcting these by computer image ‘‘crystal unbending’’. The distortion corrected image is then Fourier transformed, and for every crystal lattice diffraction spot, values for amplitudes and phases are extracted, together with data that allow one to judge the reliability of the measured values. These data are then corrected for the effect of the electron microscope’s transfer function and are used for the synthesis of a final 2D reconstruction map, which shows the projection map of one or more unit cells of the 2D crystal structure.

If data from several different images are available, these can also be merged into one common dataset, which in the case of several non-tilted images will then give a more reliable, or better resolved, projection map of the crystal structure. If data from differently tilted 2D crystal samples are available, these can be merged into a 3D dataset, which eventually can lead to a full 3D reconstruction of the membrane protein structure.

Structure determination of membrane proteins by electron crystallography usually employs large amounts of data. Depending on the crystal quality and the perfor-

mance of the sample preparation and data collection, several thousands of images usually need to be recorded, from which a sub-set of micrographs are selected by optical laser diffraction (Aebi, et al., 1973). Photographic films that show optical diffraction are then digitized. If a typical micrograph (e.g. Kodak SO-163) is digitized at 7 lm pixel size as 16 bit gray-value pixels, then an image file of 13,500 · 10,700 pixels or 289Mbyte in size can be produced. If image processing of several hundreds or thousands of such images is to be performed, a computer system capable of archiving, processing and managing Terabytes of image data is required. Computer image processing of 2D crystal images has mostly been done with the MRC programs. An interesting new development for 2D crystal image processing is the IPLT software package (Philippsen, et al., 2003).

We have previously published our software system 2dx, which assists the user in the processing of 2D crystal images (Gipson, et al., 2007). The central component of this software system is *2dx_image*, which allows the fully automated processing of one 2D crystal image. *2dx_image* is based on the MRC software, and partners the slightly adapted Fortran programs of the MRC package with a graphical user interface (GUI) that guides the user through the different processing steps, assists in the choice of processing parameters, and provides graphical and commented feed-back on the processing results. In addition to the MRC programs, *2dx_image* contains several functions and independent programs to assist the user, streamline the processing, and allow automation—as in the case of the automatic lattice determination for 2D crystal images (Zeng, et al., 2007). *2dx_image* also contains a module for Maximum Likelihood processing of data from one 2D crystal image, and for a cross-correlation based single particle processing of the 2D crystal unit cell images (Zeng, et al., 2007).

We present here the program *2dx_merge*, which assists the user in the management of a 2D crystal image processing project, and additionally performs the merging of data from multiple images. *2dx_merge* is part of the 2dx software, and interacts with the other components of the package. 2dx is available at http://2dx.org and runs natively on Mac OSX and various Linux versions.

2. Software design

2dx_merge was designed following the same philosophy as used for *2dx_image*. Both allow the user to edit scripts, change and save parameters, and view images and log file output at different verbosity levels. *2dx_merge* manages the processing of a series of images, each individually processed with *2dx_image*, and can merge the results. *2dx_merge* combines in one program the ‘‘merger’’ and the ‘‘manager’’ functions that were announced in (Gipson, et al., 2007).

2.1. File structure conventions

Upon launch, *2dx_merge* requests the user to navigate to a project root directory, which, for example, could have the location ‘‘~/MYPROT/’’. If not yet existing, *2dx_merge* will create a sub-directory ‘‘merge’’ in the project root directory, where it installs a local configuration data file called ‘‘*2dx_merge*.cfg’’. *2dx_merge* will then open a graphical user interface (GUI) window that shows several panels to display data, or to select and execute specific functions or routines (Figure 10). 2D crystal images should each reside within one directory; these directories should be placed within tilt-angle group directories, which should themselves reside within the ‘‘~/MYPROT/’’ top-level project directory. If, for example, the user has a set of non-tilted images and sets of images that were recorded at 30° and at 45° nominal sample tilt-angles (rounded to nearest 5°), then three sub-directories should be created called ‘‘~/MYPROT/MYPR-00’’, ‘‘~/MYPROT/MYPR-30’’, and ‘‘~/MYPROT/MYPR-45’’. These should contain the image directories, which each should contain one 2D crystal image, as also described in (Gipson, et al., 2007). The purpose of the tilt-angle grouped directory structure is to facilitate the definition of tilt-angle specific default settings, as each tilt-angle directory has its own copy of the default parameter configuration file 2dx_master.cfg. A user could for example have as default settings for the 30° tilted images a resolution of 5Å, and as default setting for the 45° tilted images a resolution of 7Å. This directory structure is automatically generated by *2dx_merge* during import and directory selection. Nevertheless, *2dx_merge* will also accept a different directory organization, as long as each image directory contains a copy of the *2dx_image*.cfg file, and each image-group directory contains a copy of the 2dx_master.cfg file.

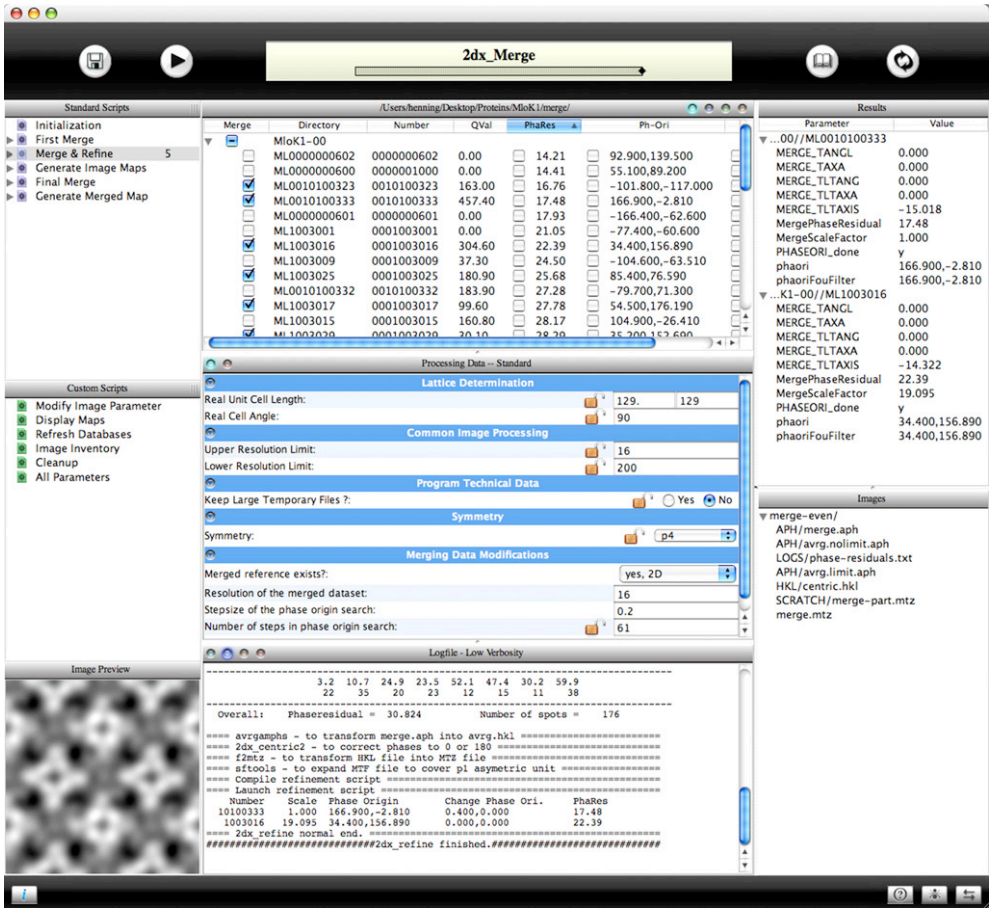

Figure 10: The graphical user interface of the *2dx_merge* program.

2.2. Adding images to an image processing project

2dx_merge assists in adding newly recorded images into the list of available images for processing. This is done via a pull-down menu option ‘‘Import Images ...’’, which prompts the user to locate either an individual image or a directory containing several image files. Image files can be in MRC or TIFF file format, and their name should follow the following convention: the first four characters of the name identify the protein, the next two digits describe the tilt-angle category, the next six digits describe the micrograph number, and the last two digits are reserved for potential copies of an image, in case this image should be processed several times. This might, for example be useful if one image contains two overlaying crystal lattices. A 30°-tilt image of micrograph 123456 could then for example be called ‘‘MYPR3012345600.tif’’. *2dx_merge* will use the information contained in the image file name to recognize the tilt-angle and micrograph number, create the corresponding

image sub-directory, move the image file into that directory, and prepare it for image processing by running the initialization script of *2dx_image* on it.

2.3. Graphical user interface

The central GUI of *2dx_merge* (Figure 10) is structured similarly to the GUI of *2dx_image*: The main part of the window is split into three columns, each of which is split into several vertically arranged sub-panels. In the default setting, the top section of the central column displays the list of all image-containing sub-directories that belong to the current project, with each image directory displayed as a row in the central table, grouped by tilt-range category. The data for this table are drawn from the ‘‘*2dx_image*.cfg’’ data files that reside in each image sub-directory. This central table displays for each image several selectable values, like QVal, phase residual, resolution, etc., the specific selection and arrangement of which can be modified by the user and saved as default setting for future usage. For instance, if ‘‘Quality Value’’ is chosen as a sorting data column via single click of the header of the column, the images are organized according to their QVal values. The QVal number for an image is a one-dimensional value that quantifies the quality of the processed image. The QVal for an image is calculated by an empirical formula in *2dx_image*, as described in (Gipson, et al., 2007). It can be used to recognize the ‘‘good’’ images for merging. It would then be a simple matter to select the first N images above a certain QVal to form the basis of a merging step. Selection sets of images may also be saved for quick re-application of previous merge results, or to run parallel reconstructions.

Below the image directory pane is the ‘‘Processing Data’’ pane, which displays the processing parameters that are relevant to the current script. The pane below this displays the ‘‘Logfile’’ of the currently selected script, once processing has been performed and a log file has been produced.

The left column in the GUI displays, in the top pane, the ‘‘Standard Scripts’’ that the user can select by mouse click, and execute by pressing the ‘‘play’’ button in the top bar of the GUI. This pane shows the scripts that are to be executed sequentially during the normal merging procedure. The pane below the ‘‘Standard Scripts’’ shows a set of ‘‘Custom Scripts’’, which the user might utilize for specific tasks. The last pane

in the left column displays a view of the currently selected image's header information, or an image preview if an image is selected (see below).

If an executing script produces results in form of defined variables or identified image files, these will be displayed in the right column of the GUI: The top pane on the right side of the GUI displays variable parameters, grouped by their relevant merging or image directories. If for example the phase origin and the beam tilt are to be refined for several images, then the ''Results'' pane will display these newly refined values grouped under their respective parent directory sub-sections. Finally, the lower right pane in the GUI displays the list of image files or other files that a script might have produced and reported for display.

The top of the GUI features a header bar, which contains the progress bar, indicating the name of the currently running script, and the progress of the execution of this script. On the left side of the progress bar are buttons to ''Save'' the current set of parameters, and the ''play'' button to run the currently selected script. On the right side of the progress bar are buttons to activate the ''Manual'' function of each script, or to ''Reload'' the image value table in the top central pane.

The footer bar of the GUI has on the left edge a button that toggles the ''Image Preview'' pane between the file header display mode, and the image preview mode. On the right edge is a button allowing the user to toggle between normal execution mode and ''dry run'' mode, in which a script can be executed, but the results are not written back into the local or image sub-directory datafiles. A bug report button, immediately left of the ''Dry Run'' button, allows users to submit bugs/issues/feature requests relating to 2dx. It points to the bug report tool at: http://2dx.org/documentation/bug_report. Immediately to the left of the Bug Report button is a ''Help Button'' which points the user to general documentation of 2dx at: http://2dx.org/documentation/2dx-software.

3. Communication between GUI and scripts

The program architecture of *2dx_image* and *2dx_merge* are nearly identical. As with *2dx_image*, scripts are implemented as cshell-style templates. Upon clicking the ''run'' button in the GUI, the empty variable declarations in the top part of the script templates are completed with data from the database, and an executable cshell script is

created and placed in the ‘‘proc’’ sub-directory in the local image directory. The GUI then executes that script as an independent sub-process, and interprets the output of the running script. For example, the *2dx_merge* script ‘‘2dx_refine. script’’ produces a logfile ‘‘LOGS/2dx_refine.log’’ and a results file ‘‘LOGS/2dx_refine.results’’. The content of the logfile is displayed by the GUI of *2dx_merge* in the lower central ‘‘Logfile’’ pane, and the content of the results file is used to populate the ‘‘Results’’ and ‘‘Images’’ panes in the GUI.

The interpretation of the results files is different for *2dx_image* and *2dx_merge*: *2dx_image* allows single parameter values to be updated to a local parameter set. In *2dx_merge*, a script can update the parameters of the merging directory, but also of any image directory. A directive ‘‘set (Statements, et al.) = (Client)’’ will update the merging parameters, while encapsulating a set of such directives between the flags <IMAGEDIR=‘‘{directory name}’’> and </IMAGEDIR> will set variables in the relevant image sub-directory to the appropriate value.

Scripts can indicate produced image files with an entry in the results file as ‘‘# Image:’’. The names of these files are then displayed in the ‘‘Images’’ pane, again grouped by their relevant sub-directories. Files in this pane can be opened by double-clicking on the file name.

Scripts can now assign ‘‘Nicknames’’ to files as well, which can be any plain text description of that file (Figure 11). For example, an entry in the results file as:

#IMAGE: “PROT0012345600.ref.fft.mrc” <Fourier Transform of Reference>

would prompt the GUIs of *2dx_image* and/or *2dx_merge* to list this file as ‘‘Fourier Transform of Reference’’ in the results pane, and showing as mouse-over help text its real file name ‘‘PROT0012345600.fft.mrc’’.

Files can also be categorized as normal or important files:

#IMAGE-IMPORTANT: “PROT0012345600.fft.mrc” <Fourier Transform>

would prompt the GUI to display this file highlighted in red. The ‘‘Images’’ file list can be filtered such that only these ‘‘important’’ images are shown in the results view.

Similarly to the parameters, all image/results files appear in the results output relative to the directory they are contained in.

While processing errors in an individual image are often easily repaired, unwanted changes to a collection of images (e.g. the coarse refinement of phase origins for a large number of images) can be labor-intensive to recover from. To aid in efficient user exploration of parameter space or new processing configurations, we have added a ‘‘Dry Run’’ option allowing the results of processing to be displayed with none of the results committed to any directories/ parameters—the goal of such steps being to promote user experimentation in an environment where it is safe to do so.

3.1. Translator infrastructure for viewing results

We have added a ‘‘translator’’ architecture, to facilitate the viewing of results files. By default, files with extensions

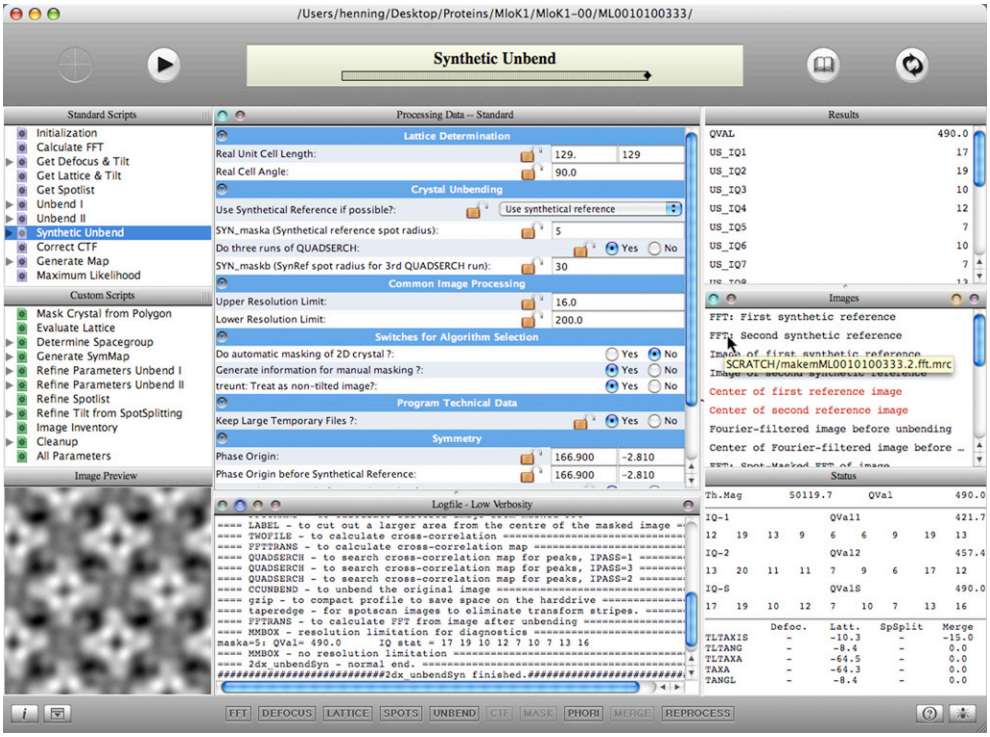

Figure 11: The graphical user interface of the *2dx_image* program. A “Synthetic Unbend” script is now included, which uses the merged dataset to generate synthetic references. The resulting IQ statistics from the “Synthetic Unbend” run are listed as third entry in the “Status” pane bottom right, labeled IQ-S. Images are displayed as nicknames, with the real file names as “mouse-over” text.

.aph, .hkl, .mrc, .mtz, .ps, and .txt, are viewable in platform-specific standard viewers (navigator, text editing software, etc.). Additionally, users may define custom

translators in the plugins/translators directory. These csh scripts take the form of ‘‘[ext].tr’’ where [ext] defines the file extension the translator should be applied to.

In most respects these translators are identical to the ‘‘script’’ files that are executed as processing commands. The main difference is that they inherit a list of configured editors as defined by config/2dx.cfg as ${program_name}_2dx_app. For instance, the translator file for MTZ files (a binary CCP4 format for reflection data) is called mtz.tr. We provide it as a cshell script that uses the CCP4 program ‘‘mtzdump’’ to transform the binary MTZ file into a temporary text file ‘‘outfile’’, which is then displayed with the standard platform (and implementation) dependent script editor, by calling ‘‘$scriptEditor_2dx_app ${outfile}’’ as the last command in mtz.tr.

The user can add a new translator by creating a new script for the extension one wishes to operate on, and adding it to the plugin/translators directory. All translators found there will automatically over-ride *2dx_merge*’s default behavior. This could be used, for instance if one wished to open .aph files in a different browser than .txt files.

3.2. Visualization

Included with *2dx_merge* is the standard navigator available for *2dx_image* allowing users to view Fourier and real space images. Since the initial release of the 2dx processing platform several new improvements have been made to this viewer.

One such addition is that Fourier space images are viewable as complex numbers showing the amplitude and phase. As also possible in the MRC program *Ximdisp*, Fourier pixels are displayed via the HSV (Hue, Saturation, Value) table, where Saturation is set to 1, Value displays the amplitude, and Hue (based on a periodic color wheel) shows the phase (Figure 12) useful to inspect synthetically generated references, and can be helpful for the manual selection of lattice spots (Zeng, et al., 2007).

The display of reconstructed projection maps indicates their dimensions and coordinates in phase origin space, i.e. x, y coordinates are mapped relative to the unit cell covering the range of -180° to 180°. This facilitates the manual adjustment of phase origin, via single click, to aide in choosing symmetry centers for the merging process.

2dx_merge also features an image ‘‘photo album’’, which displays images in a readily viewable grid, which could be useful for example to quickly assess the validity of the phase origins of the images (Figure 13).

3.3. Inheritance

To facilitate efficient automated processing, a hierarchy of configuration files exists relative to the directory structure. The project root directory contains a master configuration file, called 2dx_master.cfg.

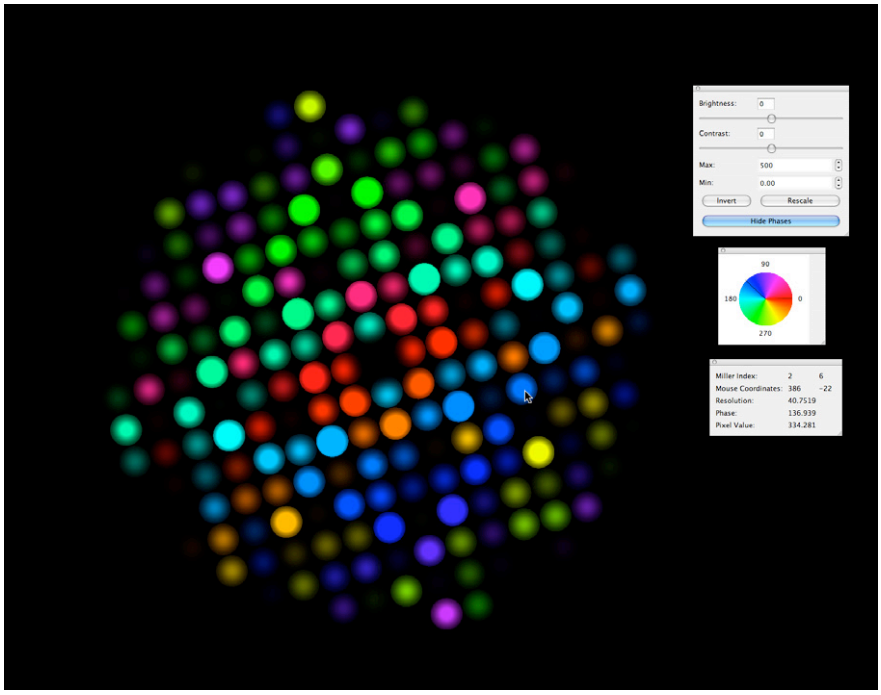

Figure 12: The full-screen browser of *2dx_image* can display Fourier transform data with color-coded phase information. This example shows a synthetic reference generated in the “Synthetic Unbend” script by the MRC program MAKETRAN.

This file defines the parameters that are valid for the entire project, like crystal space group, and real space crystal lattice dimensions. The merge directory contains a merge parameter set, *2dx_merge*.cfg. Each tilt-range directory contains another 2dx_master.cfg parameter file, which is used to provide default values for a number of parameters that should be used upon initialization of a new image directory in that tilt-range (e.g. default defocus settings or magnification). These copies of 2dx_master.cfg could also be symbolic links to the upper-level 2dx_master.cfg file. Finally, each image directory contains a local image parameter set, *2dx_image*.cfg, governing local image parameters.

The 2dx software system applies two types of inheritance, called Initialization and Sync. Sync inheritance keeps selected parameters constantly in line with the configuration file in the immediately superior directory, while Initialization inheritance parameters are copied only on the first image directory initialization. This can be useful, if, for example, all images of 30° tilted samples were recorded on a 300 kV instrument at 1000 nm defocus, and digitized at5 lm pixels size, while all non-tilted images were recorded at 200 kV and 500 nm defocus, and digitized at 7 µm pixel size.

The behavior for Initialization and Sync of each variable is defined in the configuration files themselves. It allows permanently synchronizing certain parameters (e.g. space group), or preparing the default configuration file for a newly imported image such that the fully automatic processing in *2dx_image* has the highest chance of success.

4. Workflow

4.1. Implemented algorithms

As with *2dx_image*, we have provided a collection of standard and specific scripts designed for the merging of 2D crystal images, which are mostly based on the MRC programs. These scripts are in editable text form, and can easily be replaced or modified by the user.

2dx_merge fulfills two functions, the management of a processing project, and the merging of the data. The latter is done by processing several images, using their Fourier-filtered maps as local references. *2dx_merge* then creates an initial merged dataset, which can be

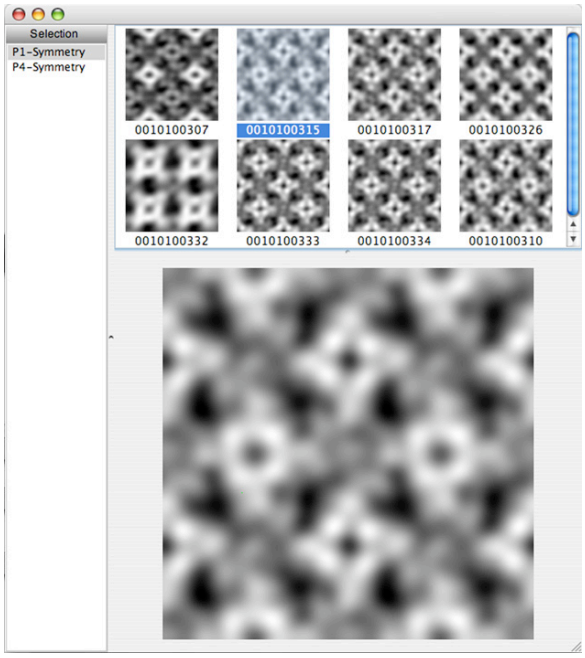

Figure 13: Image album showing reconstructed projection maps from individual 2D crystal images of MloK1 (Chiu, et al., 2007). In this example, all images are from non-tilted crystals of the same space group, and should show a similar projection map. Comparison of these maps allows identification of images of a different crystal form (e.g. the differently stained crystal in image 0010100307, first in first row), or wrongly assigned lattices (e.g. image 0010100332, first image in the second row) or wrongly assigned phase origins (e.g. image 001010333, second in second row).

used to create a synthetic reference for refined unbending of all images. This iterative refinement process should continue, while new data are added to the process.

As implemented in the MRC programs, each reflection from each image is accompanied by a figure-of-merit (FOM) value, which reflects the reliability of an individual amplitude and phase measurement. During merging of a large number of images, amplitude and phase values are weighted according to these FOM values. This has the effect that up to a certain extent the user can merge high-resolution and low-resolution images together, without running the risk that the lower-resolution images destroy the high-resolution data of other images. Nevertheless, the user can include or exclude data via the selection criteria.

Available image sub-directories are displayed in the central pane of *2dx_merge*. Double-clicking on one row in that pane will launch *2dx_image* in that image directory, where the user can then process one image either manually or automatically with the *2dx_image* program, using the scripts Unbend-I and Unbend-II. A switch in the database decides if these scripts operate with static CTF correction, or make use of the ‘‘tt’’ set of MRC programs for correction of the tilted transfer function (TTF), as depicted in Figure 14, top panels. After having processed (or re-processed) a sufficient number of images, the user can return to the *2dx_merge* program. Clicking on the top right ‘‘re-scan’’ button will refresh the central image directory pane, and update the displayed values in that table. This set of images may then be merged into either a 2D merged projection map dataset, or a 3D volume dataset. In the currently released version, *2dx_merge* supports the merging into a 2D merged dataset. Extension of the scripts to include 3D volume merging functionality is ongoing. A first preliminary 3D merging function is included in the released software.

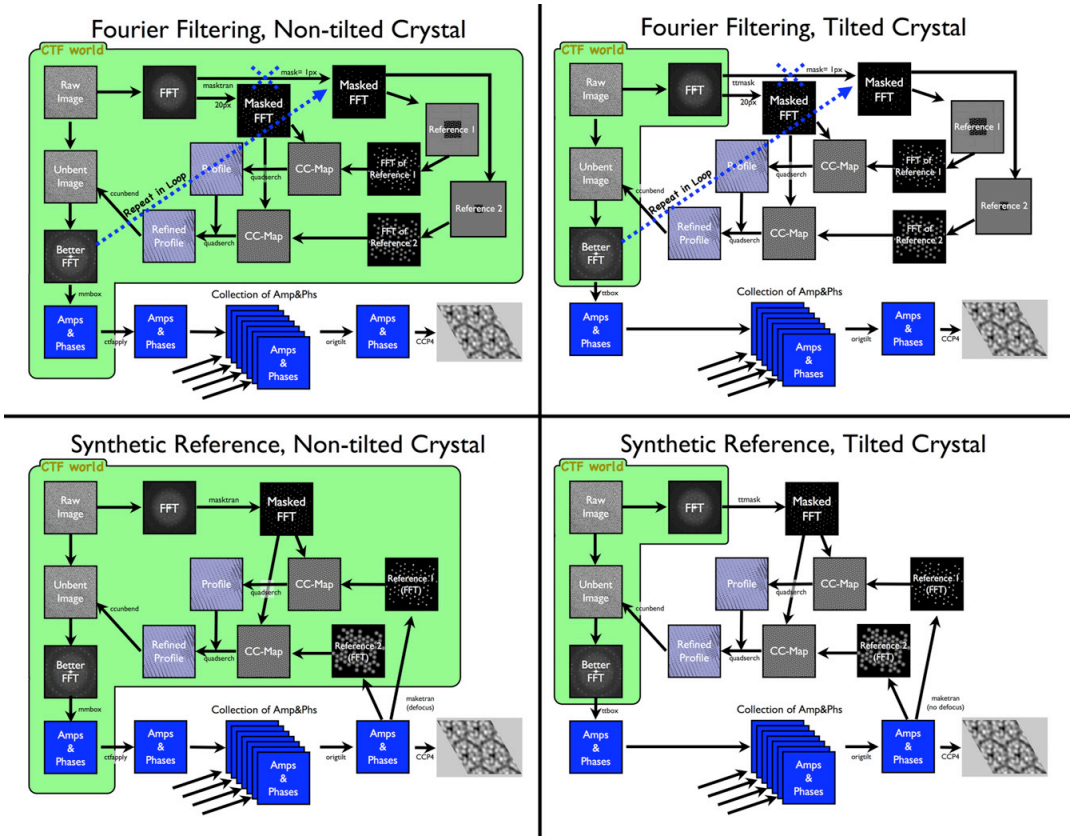

Figure 14: Four algorithms implemented in the 2dx software suite, displayed for images of non-tilted (left) or tilted (right) samples, and using the Fourier-filtered image itself (top) or a synthetically generated map (bottom) as reference for the unbending. The processing steps that deal with CTF-modulated data are located in the fields labeled by "CTF world".

The user can choose a sub-set of images to merge by selecting their image directories by mouse-click on the button in front of each image sub-directory row in the central pane (Figure 10). The currently selected group of images can then be merged, by sequentially following the ‘‘Standard Scripts’’ in the top left pane of the GUI:

First Merge. This script merges the currently selected images into a merged file, which will be created in the merging directory, and will be called ‘‘/MYPROT/merge/ merge.mtz’’. This ‘‘First Merge’’ merges the available images together using their current phase origins.

Merge & Refine. This script allows the merging of selected images into one merge.mtz merged dataset (in repetition of the previous script), and subsequently uses this merged dataset as reference to refine the parameters of the individual images. This script can be executed a pre-set number of times, using the iteration counter that appears to the right of the script name in the ‘‘Standard Scripts’’ pane. By default, this

counter is set to 5, prompting *2dx_merge* to iteratively merge and refine the merging parameters 5 times, in each round using the latest merged dataset as reference for the refinement of phase origin and other parameters. Double-clicking the iteration counter allows the user to change this number.

Generate Image Maps. This script re-creates the projection maps for all selected images, using the refined phase origins and defocus values. These maps allow manual verification of the correctly defined phase origins, by manually browsing over the list of images in the ‘‘Images’’ pane and visually inspecting their image previews. This can be useful, for example, to verify all determined phase origins or handedness assignments.

Final Merge. This script is again a repetition of the merging script (as ‘‘First Merge’’), and is used to calculate a final merged dataset to be used as reference for re-processing of all images (see below).

Generate Merged Map. This script finally allows the calculation of a resulting map of the merged dataset. This script concludes the merging phase of the chosen set of images.

Merging is based on a slightly adapted version of the MRC program origtiltk.for, here called 2dx_origtiltk.for. We created a simple Fortran program *2dx_merge_compileA* that receives from *2dx_merge* a list of the directories that contain image data to be merged. *2dx_merge_compileA* will then extract the required parameters from the *2dx_image*.cfg datafiles in the relevant image directories, and create as output a cshell script that contains a call to 2dx_origtiltk.for, providing that program will all required parameters. That cshell is then launched and performs the merging of the data. A similar program *2dx_merge_compileB* prepares the cshell script for the refinement of variables, again based on 2dx_origtiltk.for. The GUI of *2dx_merge* will then evaluate the results file of that running cshell script and update the relevant parameters (e.g. better phase origins) into the image datafiles *2dx_image.cfg*, or into the merging datafile *2dx_merge.cfg*. A third Fortran program *2dx_merge_compileM* is responsible for the compilation of the cshell script that re-creates all image maps.

These simple Fortran ‘‘helper programs’’ allow the usage of the MRC program origtiltk.for without the need to modify large fractions of that program. Future versions

of the MRC program origtiltk.for can therefore be easier updated into 2dx, and the functionality of iteratively re-merging of a growing number of images is maintained. *2dx_merge* provides in a local file *2dx_merge*_dirfile.dat a list of the selected directories, which the scripts can use for any required purpose. The ‘‘Cleanup’’ Custom-Script, for example, uses the list of provided directories to delete all temporary files in the specified image directories.

The remainder of the merging process is done by the MRC programs avrgamphs.for, 2dx_centric2.for (based on centric.for, but now including systematic absences), and the CCP4 programs f2mtz and sftools. The scripts for the calculation of 3D lattice line values and 3D volume reconstruction are in preparation.

4.2. Guided merging

2dx_merge follows a similar approach in user-guidance as *2dx_image*: The user should sequentially run the ‘‘Standard Scripts’’ in the top left pane, to merge a dataset. However, while the processing of one 2D crystal image in *2dx_image* can often successfully be done in the fully automated mode, we recommend that the merging of image data should still be done interactively, executing each step individually.

For instance, the alignment phase of merging remains a subtle process requiring user verification of proper alignment. Generally this can be approached by limiting the upper resolution and using an appropriate phase origin search step size and total number of steps (MergeStep-Size = 6.0° and IBOXPHS = 61), followed by running 5 rounds of ‘‘Merge and Refine’’. This should roughly align all images up to the specified resolution. This can be followed by further refinement rounds, while gradually increasing the merge resolution and reducing the phase origin search range. Eventually, an alignment with 61 steps of 0.1° step size should align the dataset with high accuracy. Since the difficulty of the search of the phase origin (image alignment) or the determination of the beam-tilt values for each image varies from one project to the other, this step remains a largely user-guided process.

4.3. Re-unbending of images

The merged dataset in /MYPROT/merge/merge.mtz is then available for the creation of synthetic references for a re-unbending of all individual images, using the

MRC program MAKETRAN (Kunji, et al., 2000). While the normal *2dx_image* Fourier-filtering as implemented in the Unbend-I and Unbend-II scripts uses repeated refinement of the unbending crystal distortion PROFILE to iteratively apply the improved Fourier-filtered reference, the process during synthetic unbending differs: Since the synthetically determined reference is assumed to be a ‘‘perfect’’ reference (or at least better than the information from one image alone could provide), iterative refinement of the reference is not done.

Instead, the script ‘‘Synthetic Unbend’’ in *2dx_image* (Figure 11) will use the merged dataset to generate two synthetic reference images that will be used for the determination of the unbending profile of an individual image: the first synthetic reference should have dominant low-resolution features, which can be created by using a temperature factor of 0 and a smaller Fourier mask size. The second synthetic reference should then have a stronger presence of high-resolution features, which can be created by using a negative temperature factor and a larger Fourier mask size. (The larger Fourier mask corresponds to the choice of a smaller reference box size in the Fourier filtering procedure.) The script ‘‘Synthetic Unbend’’ will use the first reference to determine a first unbending profile. This profile will then be refined, using a restricted search radius from the identified PROFILE node locations, with the second ‘‘sharpened’’ synthetic reference. These algorithms for non-tilted and tilted images are depicted in Figure 14, lower panels.

4.4. Iterative re-processing

Iterative processing within the 2dx framework can now be done on the level of interplay between unbending, Maximum Likelihood processing, and merging: A first unbending run using a Fourier-filtered reference can be used to determine an initial QUADSERCH unbending profile in *2dx_image*. This profile contains the X/Y coordinates of the identified crystal unit cells, and is now also written out in SPIDER format, if the user wishes to continue with single particle processing in the SPIDER software. This profile can be used for the initial particle stack selection for the Maximum Likelihood (ML) processing within *2dx_image*, the result of which is then translated back into amplitudes and phases (APH-file). The APH-file of either the Fourier-filtered unbending or the post-processing by the ML program is then, together

with the results from other images, used by the *2dx_merge* program to generate a merged dataset.

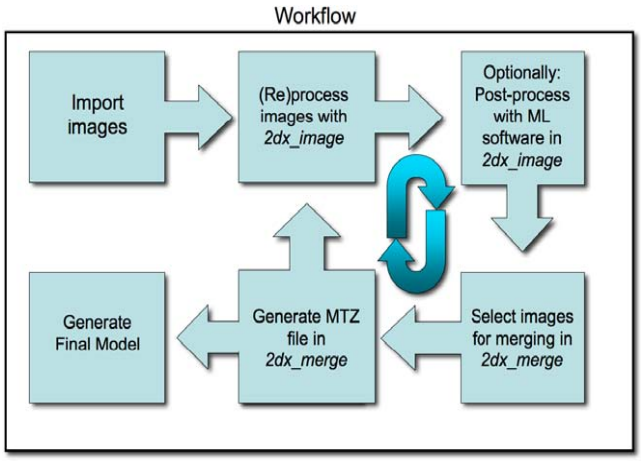

Figure 15: Generic workflow in 2dx, from images to reconstruction.

This merged dataset can then be used as reference for a refinement of the unbending process in *2dx_image* with the new standard-script ‘‘Synthetic Unbending’’. This will generally result in a refined QUADSERCH profile, which in turn will allow a slightly better ML processing, finally resulting in a better-merged dataset, and so on. This processing can be applied iteratively, but bears the risk of locking in to noise correlation, if not handled carefully, similar to the behavior of single particle processing of noisy images (Grigorieff, 2000). This iterative application of the different algorithms, scripts and programs, while gradually including more images as they become available, can be performed until convergence is reached (Figure 15).

4.5. Interaction with *2dx_image*

2dx_merge and *2dx_image* closely interact with each other. In *2dx_merge*, double-clicking on any of the image directory lines in the central pane opens an instance of *2dx_image*, allowing the user to process or modify variables for that image.

2dx_merge can also launch *2dx_image* as a GUI-free background process. For this, it makes use of a new feature of *2dx_image*, which can now be called from the command line with a script name (including wildcard matching) as parameter. *2dx_image* will then execute that script silently. For instance executing the

2dx_fftrans.script through *2dx_image* on a protein directory /MYPROT/MYPR-00/MYPROT001 from a *2dx_merge* script could be done with:

```
$app_2dx_image /MYPROT/MYPR-00/MYPROT001
"2dx_fftrans"
```

The first argument to *2dx_image* is the directory to be operated on. The second argument is a list of scripts (allowing wildcards), which are to be executed. Standard scripts are defined by their name (i.e. 2dx_fftrans), custom scripts must be preceded by a ''+''. Standard scripts listed in this manner will be executed according to their sort order while custom scripts will be executed in the order in which they are listed. Custom scripts are always executed after all standard scripts are completed.

The command:

```
$app_2dx_image /MYPROT/MYPR-00/MYPROT001 "*,
+2dx_evaluateLattice"
```

then would execute all standard scripts according to their sort order and finally execute 2dx_evaluateLattice (a custom script). This feature prepares *2dx_image* for remote execution on a computer cluster.

5. Conclusions

2dx_merge facilitates the management of an electron crystallography project, and enables the merging of the collected and processed data. At the outset of each execution of *2dx_merge* an initialization script is called to ensure proper setup of all folders, hierarchies and configuration files. An initial merge step is used to create a first merged dataset. The ''Merge & Refine'' step then iteratively aligns images to the reference. After alignment, the reconstruction maps are re-generated by the ''Generate Image Maps'' script, and can be used to visually verify alignment. The last two scripts: ''Final Merge'' and ''Generate Merged Map'' generate the reconstruction, applying symmetry and resolution limits, from the aligned data. The merged dataset can then be used for the creation of a synthetic reference, to re-unbend all images in the *2dx_image* program, and to provide the Maximum Likelihood processing in *2dx_image* with better particle starting locations. Results of this re-processing of all images can be iteratively used for re-merging.

2dx is designed as a project oriented ‘‘meta-processor’’, which organizes and executes script templates. The current package with included scripts allows users to process and merge electron crystallography data. Both, the individual processing steps and the over-all merging workflow are entirely customizable and should satisfy most merging needs. Default scripts follow the standard MRC merging path, but users are free to edit/modify/implement custom designs. 3D merging is in preparation. 2dx is available under the GNU Public License at http://2dx.org.

Chapter 5 Model Free Structure Determination relative to Prior Knowledge

Originally appeared as: Gipson, B., Masiel. D, Browning N. D., Spence, J., Mitsuoka, K., and Stahlberg, H., Automatic Recovery of Missing Amplitudes and Phases in Tilt-Limited Electron Crystallography of 2D Crystals, Nature Methods (Submitted April, 2010).

1. Introduction

Electron crystallography of two-dimensional membrane protein crystals records electron microscopy images of the thin sheet-like crystals, which are oriented in the microscope horizontally, or at a certain tilt angle of typically up to 60º. The Fourier transformation of such images then contains information about the amplitudes and phases of the diffraction orders, while these are modulated by the instrument contrast transfer function. Alternatively, electron diffraction patterns can be recorded of the crystals, which then only yield information about the amplitudes of the structure factors. Image processing conventionally tries to evaluate and merge the amplitude and phase information from several recorded images and diffraction patterns, yielding a dataset in the Fourier domain that is void of measurements in a so-called missing cone region

around the Z-axis, due to the limited tilting possibilities of the samples in the microscope.

The central problem facing electron crystallography of two-dimensional membrane protein crystals can also be formulated as the following: "What protein structure best generates the data observed, given what is known?" The recorded data populate a continuous 3-dimensional Fourier domain as independent amplitudes (from electron diffraction patterns), independent phases (from real-space images, when amplitudes are taken from diffraction patterns), or fully coupled amplitude and phase terms (from real-space images, when diffraction patterns are not available). The terms are irregularly sampled in the direction normal to the membrane and, in Transmission Electron Microscopy (TEM), suffer from changes in the amplitude profile due to uncertainties in the determination of the microscope's contrast transfer function. Uncorrelated noise also introduces variation of certainties in data quality.

Iterative enforcement of known constraints allows the above question to be answered in full, using the measured data as well as additional prior knowledge. We present here both simulations and an experimental application of projection onto non-convex sets using a pseudo-inverse mapping between regularized real space and non-uniform Fourier space. We show that this mapping, combined with the application of Lions-Mercier (Lions and Mercier, 1979) style projections, permits the native use of uncoupled amplitudes and phases, in addition to allowing a complete recovery of the missing cone.

The algorithm is completely general and, without modification, should be applicable to any constrained tomographic problem including, for example, 2D electron crystallography, single particle analysis, thick-specimen tomography and computed wave tomography.

2. Incomplete Data

There are an infinite number of unconstrained real-space solutions that generate an incomplete 3D Fourier dataset – a case that is heavily exaggerated in the case of uncertain, noisy data. Constraints applied to either space however, can reduce the size or dimension of the space of solutions dramatically. Introducing this known information

about a protein under study in an explicit, non-iterative, algebraic manner can be difficult, particularly in the case where constraints are only loosely known.

Projection onto convex sets is a robust and rigorous theory of optimization, which is applied using a straightforward alternating iterative application of known convex constraints to a dataset. Closely related is the study of projections onto non-convex sets, which governs constraint sets with ambiguous projections. An example of such a non-convex set would be that defined by the points on the circumference of a circle. Here the point at the center of the circle for example has an undefined projection, as all points in the set are equidistant from it. In spite of this, extensions to known convex projection algorithms (in particular Lions-Mercier (Lions and Mercier, 1979)) have proven weak convergence properties (subject to satisfiability) in theory (Levi and Stark, 1984; Bauschke, et al., 2002), numerical simulation (Fienup, 1987), and experiment (Spence, 2007).

The language of projection operators is natural and well suited to most constraints. In the case of electron crystallography, constraints exist strictly in the real or Fourier domain and naturally translate into projection operators. For example, due to the planar nature of a 2D crystal the real space volume of a membrane bound structure is strictly contained within a 2D slab of finite thickness. This has been enforced as a constraint specifying that the real-space volume is non-zero only within this known region. The corresponding "oversampling" along the reciprocal-space rods normal to the slab was found to greatly reduce the number of high-resolution cryo-em images needed at the experimentally difficult high tilt angles, to phase a data set (Spence, et al., 2003). The associated projection is unique for all structures (and therefore convex) as it simply sets to zero all known empty regions. In the Fourier domain, the constraint of known amplitudes or phases (namely the collected data) at particular points has an equally simple projection operator: the enforcement of the appropriate phase or amplitude at the point in question. While changing an existing phase to a known phase represents a unique, convex constraint, the case of changing a zero amplitude to a non-zero amplitude presents an ambiguous projection –namely that it is undefined which phase should be assigned to the new amplitude– and as a result the amplitude constraint is non-convex.

3. Truncated singular value decomposition

Alternating projections between sets requires an invertible mapping between projection spaces. In the case of simulated data sets the values are present at a regular sampling and the Discrete Fourier Transform (DFT) suffices as a one-to-one mapping between the real and Fourier domains. In the case of experimental data, values occur out of register at an arbitrary sampling and require either regularization or an operator, which takes this non-uniform sampling into account. Traditional methods (Crowther, et al., 1996; Agard, 1983) have used SINC interpolation as an initial regularizing step before using the DFT as the mapping. Using a non-uniform transform from regularly sampled real-space to irregularly sampled Fourier space, however, provides a method to find the best unbiased real linear estimator for a given Fourier data-set.

The relationship between a discretely sampled real-space 2D-crystal structure and an irregular finite sampling of the continuous Fourier domain is a linear one,

$$\Im_{r,s} x_r = \hat{x}_s \qquad \text{Eq. 9}$$

where $\Im_{r,s}$ is the discrete to s-sampled Fourier transform, x_r is an unknown regularly spaced real space solution, and $\hat{x}_s$ is the Fourier domain data sampled at s. In most experimental cases, depending on the number and distribution of the data samples, this represents a severely ill conditioned matrix. A standard tactic (Hansen, 1987) in ill-conditioned inverse problems is to use a pseudo-inverse matrix calculated by truncated singular value decomposition, which provides a stabilized least squares solution to the problem.

$$x_r = \Im^{+}_{r,s} \hat{x}_s \qquad \text{Eq. 10}$$

Applying constraints in the Fourier or Real domain may then be performed directly on the data, with numerical accuracy dependent only on the quality of the data and the number of assumed discrete real samples.

As zero terms in the real domain (i.e. those values outside of the 2D-crystal slab) do not contribute to the Fourier sum, they may be removed, resulting in a $m \times n$ transform with an implicit support constraint. In cases where this is the only real-space constraint, the above may be re-written as

$$\Im_{r,s}\Im_{r,s}^{+}\hat{x}_s = \Gamma\hat{x}_s = \hat{x}_s \qquad \text{Eq. 11}$$

or that all Fourier domain solutions lie in the Eigenvector space of Γ, describing all finite slabs that generate the observed data.

Both the sensitivity of Γ to noise and the accuracy of solutions depend heavily on the choice of the regularization parameter ν in the SVD truncation. High condition value matrices are extremely sensitive to noise, while low condition numbers yield a less accurate reconstruction. The reliance on ν can be lessened by calculating an estimated error for Γ:

$$W = diag^{+}\left(abs(\Gamma)\cdot\varepsilon J\right) \qquad \text{Eq. 12}$$

Here, J is the N dimensional vector of all ones, ε is the estimated error level, $diag^{+}(v)$ indicates the diagonal matrix formed from vector v, where the non-zero elements have been inverted and $abs(A)$ is the element-wise absolute value. This weighting function can additionally be normalized, combined with other known confidence values or otherwise altered to fit the purposes of a given dataset. While this error function could in principle be used as a least squares weighting function,

$$\Gamma_W = \Im_{r,s}\left(W\Im_{r,s}\right)^{+}W \qquad \text{Eq. 13}$$

it will later be shown in Eq. 19 that this is not optimal for our purposes.

In the most general case of electron crystallography of 2D crystals, Fourier amplitudes (A) and phases ($e^{i\theta}$) are collected as uncoupled values

$$\begin{pmatrix} A \\ * \end{pmatrix}, \begin{pmatrix} * \\ e^{i\theta} \end{pmatrix} \qquad \text{Eq. 14}$$

which form the basis for the Fourier constraints. A projection operator $P_{\Im}$ (Eq. 16) enforcing the constraints in Eq. 14 can be applied to an arbitrary image in the form of Eq. 15

$$v = \left[\begin{pmatrix} \alpha \\ \Lambda \end{pmatrix} \circ \begin{pmatrix} e^{i\phi} \\ e^{it} \end{pmatrix}\right] \qquad \text{Eq. 15}$$

$$P_{\Im}(v) = \left[\begin{pmatrix} \boldsymbol{A} \\ \Lambda \end{pmatrix} \circ \begin{pmatrix} e^{i\phi} \\ e^{i\boldsymbol{\theta}} \end{pmatrix} \right] \qquad \text{Eq. 16}$$

where α and e^{it} are discarded and replaced by known amplitudes and phases. Here, $\circ$ represents the Hadamard (element-wise) product of two vectors or matrices. Using Eq. 11 and Eq. 16, iterative application of these Fourier constraints can be performed.

$$\Gamma P_{\Im}\left(v_n\right) = v_{n+1} \qquad \text{Eq. 17}$$

4. Iterations in fractional time steps

Due to the non-convex nature of the system, however, strictly enforcing these constraints as in Eq. 17 can lead to pseudo-solutions and exponentially slow convergence (stagnation) (Bauschke, et al., 2002). These problems can be alleviated using fractional time-step parameters as used in Lions-Mercier (Lions and Mercier, 1979), Fienup's Hybrid Input Output (Fienup, 1987), and the Douglas-Rachford Algorithm (Douglas Jr and Rachford Jr, 1956).

$$\left(1-\beta\right)\cdot v_n + \beta \cdot \Gamma P_{\Im}\left(v_n\right) = v_{n+1} \quad | \quad 0 \le \beta \le 1 \qquad \text{Eq. 18}$$

While such fractional parameters have historically been globally enforced as in (Lions and Mercier, 1979), we have used the weights of Eq. 12 in a point-wise fashion independently operating, subject to known confidence levels, on each value.

$$(1-W)\cdot v_n + W \cdot \Gamma P_{\Im}(v_n) = v_{n+1} \qquad \text{Eq. 19}$$

This has the simultaneous benefit of introducing a per-point fractional time step for convergence purposes, as well as allowing values that are more certain to guide the convergence process as a whole.

Combining the concepts of Eq. 9 and Eq. 19, we then form a completely general optimization routine using both real and Fourier constraints. The data in real space (x) and Fourier space (v) are iteratively obtained as

$$x_n = \Im^{+}_{r,s}\left(v_n\right) \qquad \text{Eq. 20}$$

$$\Im_{r,s}\left((1-\beta)\cdot x_n + \beta\cdot P_{\Re}(x_n)\right) = v_{n+1} \qquad \text{Eq. 21}$$

$$(1-W)\cdot v_{n+1} + W\cdot P_{\Im}\left(v_{n+1}\right) = v_{n+2} \qquad \text{Eq. 22}$$

Here the final $P_{\Re}$ and $P_{\Im}$ represent all real and Fourier space constraints, respectively, that are appropriate for the system. For example, $P_{\Re}$ might include (but by no means be limited to) real-valuedness, non-negativity and additional support information, as well as more vague concepts such as appropriately defined regional continuity or non-diffuse (i.e. sharp) boundaries between zero and non-zero regions. Equally, $P_{\Im}$ might include symmetry and spectral profile (i.e. from a small-angle X-Ray scattering experiment), as well as optical constraints such as that from a known point-spread function. The presence of a global β in Eq. 21 is necessary here, as point-wise confidence about the real-domain is typically unknown --though obviously Eq. 21 could be modified according to Eq. 22, if such were available.

5. Shrinkwrap optimization

Shrinkwrap (Marchesini, et al., 2003), a final added optimization that has recently gained increased theoretical support (Jin, et al., 2010), involves setting constant (or zero) all real-space values found below the assumed noise threshold as an iterative extension to the known support.

$$x_n(i) = \begin{cases} x_n(i) & ; x_n(i) > \varepsilon \\ 0 & ; x_n(i) \le \varepsilon \end{cases} \qquad \text{Eq. 23}$$

Knowledge that the 2D-crystal exists as a slab of finite thickness provides an initial estimate for the support of the object; empty regions found within this slab are thus iteratively refined and can be used in the above procedure by appropriately modifying the real space projection operator $P_{\Re}$.

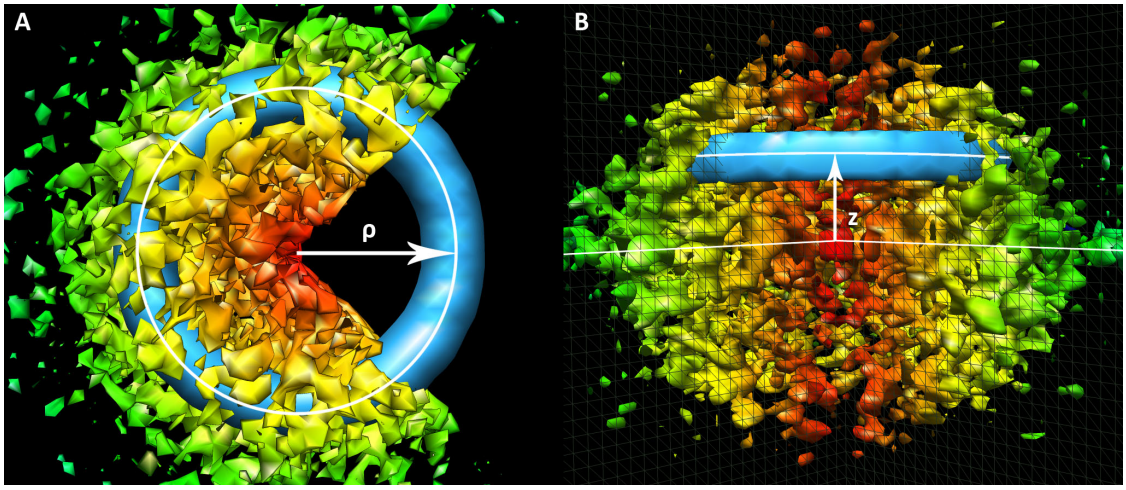

Figure 16: The Cylindrical Ring Correlation measures pixels along a ring of radius ρ and height *z* in Fourier space. (A) top-view along *z*, (B) side view. These pixels can be correlated with their symmetric counterparts or a known external reference dataset, resulting in a plot of correlation as a function of ρ and z (shown in Figure 17).

6. Cylindrical Ring Correlation (CRC) as convergence measure

In addition to using standard resolution measures for analysis, such as Fourier Shell correlation (Van Heel, 1987), we developed a cylindrical ring correlation measure (CRC), which calculates the Fourier Ring Correlation (Figure 16) for each horizontal 2D slab of 3D Fourier space. This CRC measure produces a 2D-map that describes resolution in ρ, z space (i.e. the ring correlation value as a function of radius ρ and height z of the ring of constant diameter in Fourier space). Using this measure it is possible to measure point-wise ring-correlation for all points inside and outside of the missing cone, and observe the convergence of the algorithm in real time when a reference map is available.

7. Application to underdetermined simulated experimental data

We applied Eq. 20, Eq. 21 and Eq. 22 with Eq. 23 to a series of test cases based on computed datasets of actual 3D models of biological proteins. The atomic coordinates used in this study were the maps with PDB entry codes 1AT9, 1BRR and 1C3W, the latter two both contained ordered lipids that fit well with observed densities.

To assess the improvement in amplitudes, a combination of phaser (McCoy, et al., 2007) and refmac5 (Murshudov, et al., 1999) from the CCP4 processing suite (Collaborative Computational Project, 1994) were used to generate Table 1 and comparison atomic models. This amounted to Molecular Replacement of PCO refined

amplitudes followed by a refmac5 rigid-body fit. Table 1 in the manuscript was generated from the outputs of the final round of refmac5. The last line of Table 1 resulted from the real-space correlation factor comparison by sfcheck (Vaguine, et al., 1999) of the refmac5 output pdb and refmac5 output density.

The CRC comparisons were performed on the full PCO refined phases and amplitudes and were compared to a one-time rigid-body fit of 1BRR to the density by Chimera (Pettersen, et al., 2004). No other external tools were used in this comparison. Beyond the algorithm defined in this paper, no refinement, CCP4 or otherwise, was applied to the data itself at any time.

The atomic maps mentioned above were used to generate a series of 2.5Å models of bacteriorhodopsin with simulated tilt-limitations between 30º and 60º. Essentially perfect convergence (>99% Fourier Shell Correlation at Nyquist) was observed for all cases tested and was achieved in fewer than 200 rounds, including those in the presence of a SNR of ~10%. These results were found to be essentially in complete in agreement with those of (Spence, et al., 2003).

The CRC for simulations showed that the convergence process is being spatially localized in Fourier space, growing from known constraint regions slowly outward into unknown regions, such as the missing cone, in Fourier space. This highlights the importance of the choice of the starting point, x_0 , for undetermined Fourier pixel values at the beginning of the iterative processing. Setting x_0 equal to the minimum energy solution to Eq. 9 (or to zero, a closely related solution path) consistently out-performed, in terms of time to convergence, guesses with random initial amplitudes and phases for the entire Fourier domain and also outperformed hybrid-guesses, where only unknown values in the Fourier domain were set to random values.

Random starting points will typically partially satisfy constraints in both the real and Fourier domain. These semi-consistent solutions produce densities in the Fourier domain that persist through iterations, and can guide the refinement of neighboring densities into wrong pseudo-solutions. This appears to result in slowly growing "solution fronts" both from pseudo-solution densities and constraint densities, which eventually collide and slowly attempt to reconcile. Starting with the unknown densities set to zero instead insures that a single, self-consistent, solution front is allowed to

propagate through "resistance free" zero-valued solution space. This was found to be a rapid process. In the absence of noise however, all starting positions converged to essentially complete correlation, differing only by required iterations –typically ~30 for a zero starting point to more than 1000 rounds for a random guess.

8. Application to an electron crystallography dataset of Bacteriorhodopsin

Continuing the tradition of related signal recovery methods such as (Agard and Stroud, 1982), the routine was also applied to an experimental data-set of Bacteriorhodopsin (Kimura, et al., 1997). This dataset was used in 1997 to generate a 3D reconstruction at 3.5Å resolution and contains both electron diffraction intensities and approximate phases derived from real-domain transmission electron microscope projection images. Data were available from -60º to +60º sample tilt angle, leaving an empty, vertically aligned “missing cone” of 60º angular diameter. These data were the basis of our refinement processing with the projective constraint optimization.

Due to memory limitations, the algorithm was approached in two stages. The first applied the described algorithm strictly in one dimension along the Z-axis in order to produce a regularly sampled 3D-volume. The algorithm was again applied, this time using the three-dimensional DFT as an invertible mapping, on the resulting volume to produce the final map. An extension, where a full 3D Fourier SVD decomposition is calculated, is planned for future versions of the software.

Eq. 20 shows that a finite number of pixels in the support of the membrane slab are being solved for. From these coefficients any arbitrary Fourier sampling can be achieved, and the effect of the above 2-step process can be minimized simply by choosing a fine-enough target sampling.

A sharp boundary constraint was performed by first applying a band-pass filter between 10-2.5Å to the current round of processing. A binary threshold was applied (0 to all values bellow a density of 0.14 and 1 above). The resulting map was then intersected with the constant 2D-slab support to produce a new sharp-edge preserving, Shrinkwrap generated, support that was used as usual in the remainder of the algorithm. The band-pass filter acts as an edge detection filter within the range of resolutions of biological interest. This results in generally contiguous structures with sharp, well-defined boundaries.

The above-described algorithm was applied to this raw dataset, running 24 rounds with conventional constraints (without Shrinkwrap), followed by 122 rounds of iterations including the Shrinkwrap constraint and with the inclusion of a sharp-boundary constraint, until convergence was reached.

Applied real constraints included real-valuedness, the known Z-height of the membrane slab, and Shrinkwrap paired with a sharp-edge constraint resulting from a threshholded band-pass edge detection on the Shrinkwrap support. Fourier constraints included 97157 electron-diffraction-derived known amplitudes with estimated error levels, and 25263 phases derived from Transmission Electron Microscopy images also with estimated error levels. P3 symmetry was additionally enforced in the Fourier domain. Full data details are available in Kimura et al. 1997 (Kimura, et al., 1997).

The algorithm was run for 146 rounds before convergence, with most large-scale changes observed during the first 24 rounds. The convergence parameter β for real space from Eq. 21 was set at 0.5. The pointwise weights W from Eq. 19 were based on traditional (Crowther, et al., 1996) crystallographic SNR error estimates, but were normalized to the range of 0 to 1. Near round 24 a one time multiplicative factor of 1.5 was applied to all purely refined Fourier values (i.e. $1.5*(1-W)$). While this application did not apparently affect the CRC or real-space result, it did boost the amplitudes in purely refined regions closer to uniformity.

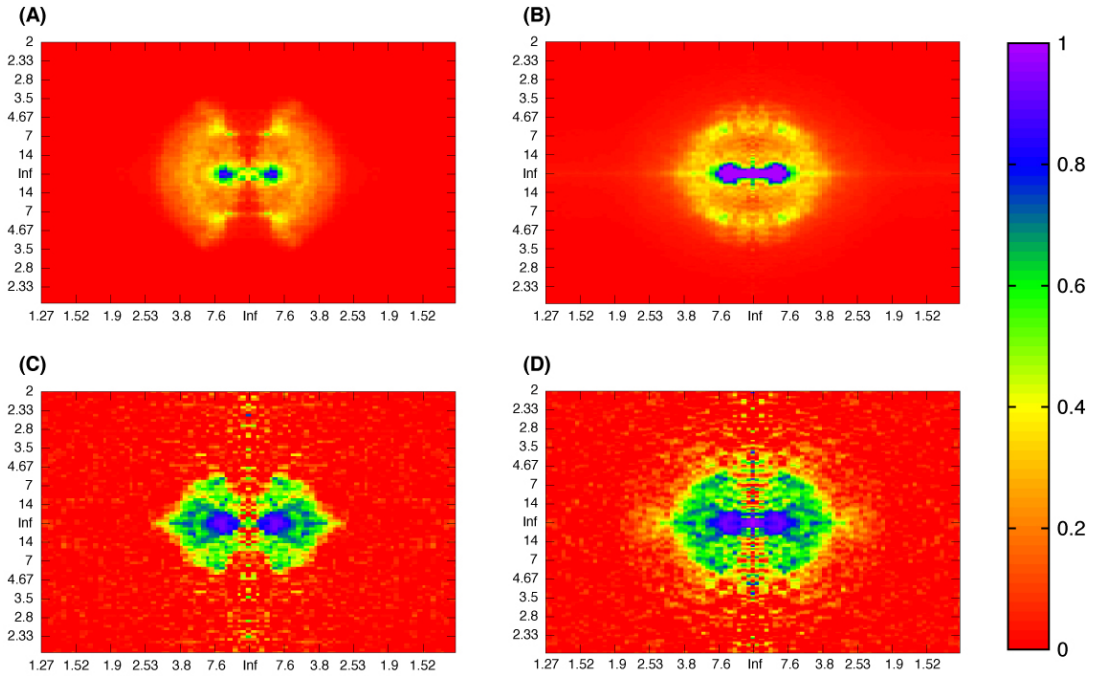

Figure 17: Normalized Intensity profiles (top) and Cylindrical Ring Correlation (CRC) plots (bottom) calculated relative to an atomic bacteriorhodopsin crystallographic model (1BRR), without (left) and with (right) refinement by the here described refinement algorithm. Plots are given as a function of ring of radius ρ (horizontal axis, in Å), and height *z* (in the vertical axis, in Å) in Fourier space.

Processing was performed on a 2.4 Ghz Intel Core 2 Duo Macbook Pro with 4 GB of RAM and a GeForce 9600M GT, and on an 8-core 2.60 Ghz RHEL 5.0 Linux desktop computer with a Nvidia Tesla 1060.

The raw measured amplitudes and phases from the images were directly used as the input of this algorithm; a lattice line fitting procedure as commonly performed in the MRC image processing of 2D crystal data (Agard, 1983) was not done. Maximum structure correlation (Vaguine, et al., 1999) of better than 0.8, of the obtained refined map relative to the atomic model 1BRR (Essen, et al., 1998), was achieved when filtered to 2.5Å. A more complete picture of angle dependent resolution is available from the full CRC (Figure 17), which shows correlation out to beyond the data resolution limit. Comparison with the CRC in the region of the missing cone also shows significant correlation and a generally isotropic distribution of resolution, with the exception of characteristic semi-regular absences in correlation in the vertical direction. While these absences could be attributed to the low number of pixels at radii close to the center of the cylinder (leading to less meaningful correlation factors) or the

tightness of the 2D slab mask, it may also be a result of ”striation” errors typical (Spence, 2002) of iterated projection solutions.

	PCO			*MRC*		*PCO - No Cone*	
Resolution limit	4.00	2.98	2.50	4.00	2.98	4.00	2.98
Number of used reflections	6669	16226	27442	2837	5084	2870	5182
Percentage observed	100.00	100	99.99	80.27	59.91	81.22	61.08
Percentage of free reflections	5.538	5.222	5.088	3.961	4.741	3.981	4.760
Overall R factor	**0.361**	**0.4455**	**0.496**	**0.336**	**0.367**	**0.326**	**0.388**
Free R factor	0.353	0.424	0.468	0.319	0.339	0.306	0.393
Overall weighted R factor	0.331	0.4067	0.434	0.300	0.330	0.326	0.388
Free weighted R factor	0.320	0.384	0.407	0.365	0.370	0.306	0.393
Overall correlation coefficient	0.796	0.8353	0.865	0.739	0.717	0.817	0.833
Free correlation coefficient	0.805	0.8525	0.878	0.728	0.711	0.818	0.824
Structure correlation	**0.601**	**0.731**	**0.806**	**0.669**	**0.613**	**0.671**	**0.741**

Table 4: refmacs5 generated table describing comparison of atomic structure (1BRR) with the Projective Constraint Optimization refinement data, unrefined MRC processed data that contain a missing cone region, and refinement result data with Fourier pixels from within the formerly missing cone region excluded. sfcheck was used to produce the real-space structure correlation values (last line).

Real-space correlation values (Table 4) show significant improvement for the here described refinement algorithm, resulting in a final structure that, when limited to 2.5Å resolution shows an 80.6% structure correlation coefficient with the atomic 1BRR model. Additionally, comparison of the refined result strictly within the experimentally sampled region of the unrefined map shows unambiguous improvement over the unrefined map out to 4.0Å. Interestingly, however, the calculated R-Factors appear to increase commensurately with the amount of purely refined (i.e. unobserved) data. This effect may have been the result of over-fitting of atomic comparison models by refmac5 (Murshudov, et al., 1999).

Alternatively, this phenomenon could also be caused by the absence of B-factor consideration during the refinement process; this would lead to an inherently diffuse dataset that would mildly conflict with the sharp-boundary constraint, itself tending to up-weight high-resolution amplitudes. Iterative application of real-space boundaries

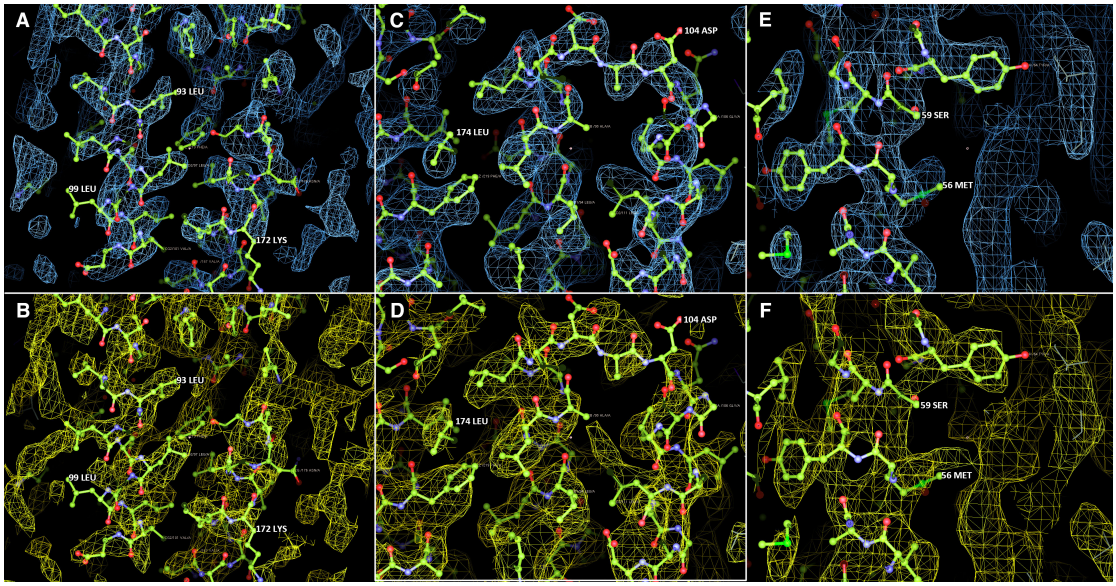

Figure 18: Close-up views of selected map positions of the bacteriorhodopsin dataset without (bottom) and with (top) refinement. H backbone (A), (B) is well defined, and the loop at 103 ALA (C), (D) is connected and well resolved relative to the rest of the structure. All prominent side chains (E), (F) are accounted for.

equates to a series of convolutions on Fourier space. As a result, amplitudes generated purely by refinement must be energetically stabilized by constrained neighbors. Without additional energetic Fourier constraints or consideration[3], refined amplitudes would be down-weighted relative to their distance to constraint densities and thus yield higher R-factors. Phases, however, would remain free of this requirement and would remain largely accurate regardless of distance, leading to the observed higher correlation factors. Enforcing constant radial averages for the amplitudes, or applying a known low-angle scattering spectral profile, or otherwise accounting for B-factors during the refinement would likely alleviate this problem.

[3] **The one-time scaling of all purely refined (i.e. not experimentally derived) values by a multiplicative factor (e.g. 1.5) after several rounds of iteration was found to improve refined amplitudes by the end of processing, though appeared to have little effect on the CRC or real space result.**

Visual inspection of the map (Figure 18) shows that the refinement procedure produces a reconstruction with a well-resolved helical pitch and side chain densities that were not as clearly seen after application of traditional lattice line interpolation methods[4]. In particular, horizontal features such as extracellular loops, most dependent on vertical resolution, are in our refined reconstruction clearly visible at a single map-wide iso-surface threshold.

9. Discussion

Noise typically leads to an increased density of local minima and stagnation of the iterative refinement procedure (Spence, 2007). Eq. 10 offers a solution to this problem as data is collected in the continuously oversampled Fourier domain, causing neighboring values (relative to real-space support) to correlate. In this way global noise contribution can be controlled by the addition of more data. The amount of data necessary and the chances for achieving a unique solution were seen to be most sensitive to the ability to estimate error. When error was not considered, the algorithm effectively proceeded as assuming an error of zero, which typically corresponded to the enforcement of conflicting constraints, which, as can also easily be shown through simulations, produced radically diverging densities within a few tens of rounds. Even only approximate estimations of error, however, applied as in Eq. 19, were found to stabilize convergence.

Over-fitting was found to be a problem in cases where Fourier reconstruction was attempted well beyond the limits of sampled resolution (i.e. 1Å). As both phases and amplitudes are unbounded in the refinement regime, spurious high frequency Fourier terms begin to dominate the reconstruction, likely counter-balancing known Fourier constraints in an attempt to satisfy both Fourier-domain noise and the real-space support constraint. This appeared in real space as either high-valued, apparently randomly scattered, point densities or 1-pixel wide slabs touching the boundaries of the allowed support regions. When a real-space sharp region-boundary constraint was

[4] **All unrefined results (lattice line fit, or MRC) refer to only the MRC processed dataset used prior to CNS/CCP4 refinement in Kimura et al. 1997. No non-MRC refinement has been performed on the comparisson dataset.**

applied, however, all above-mentioned over-fitting effects were almost completely absent.

10. Conclusion

Iterative application of known projective constraints presents a dynamic and completely general framework for the refinement, which automatically reduces solution search space sizes for a wide range of problems. The procedure defined here has shown that even in cases where irregular sampling and variable levels of noise are present, complete structure recovery is experimentally possible. The presented refined bacteriorhodopsin data set clearly shows the power of this algorithm for general-purpose electron crystallography of 2D crystals.

Additionally, this result demonstrates experimentally that in such constrained or correlated settings as above, an increase in the amount of data (regardless of sampling) can be directly leveraged into an increase in resolution in a repeatable manner. As recent results (Raines, et al., 2010) demonstrate, given sufficiently oversampled data, almost any tilt range suffices as the basis for full three-dimensional reconstruction. Such a reduction in tilt-range requirements could have far-reaching implications, both for data-collection strategies in electron microscopy, and for pushing the current resolution limits of biologically relevant structural results.

This software is released as open source under the GPL at 2dx.org. It currently exists as both a stand-alone C++ program and individual Octave (Eaton, 2002) modules. Additionally, all software has been natively optimized to take advantage of a CUDA ready GPU, if present. The stand-alone software will be incorporated into all future releases of our electron crystallography processing software, 2dx (Gipson, et al., 2007), as an optional alternative to traditional MRC lattice line fitting, and will be available at http://2dx.org.

Chapter 6 Multiple Solutions: Searching Solution Space with Meta-Heuristics

Will appear as: Masiel, D. Gipson, B., Morgan, D., Guo, T., Spence, J., Browning, N. D., Metaheuristic Algorithms for the Phase Problem in Ultrafast Diffractive Imaging, (In preparation)

1. Introduction

Some of the most exciting developments in the field of coherent diffractive imaging (CDI) in recent years have come from the use of ultrafast x-ray pulses as a source of illumination. Ultrafast electron pulses with similar wavelengths and temporal profiles can be produced with scattering cross sections that are typically much higher. Sources intended for use in ultrafast electron diffraction and dynamic transmission electron microscopy (DTEM) are quite viable for CDI and in many cases with few changes required in the experimental setup. There is a multitude of already established techniques and facilities for producing ultrafast electron pulses, and while they may not be perfectly suited for atomic resolution structural determination, they represent an underutilized and highly appropriate resource for studying material dynamics using CDI.

Many ultrafast x-ray CDI experiments to date have focused on using short pulse duration as a means of delivering doses well above the damage threshold before radiation damage can occur (Gaffney and Chapman, 2007). To achieve atomic resolution imaging in non-periodic objects, diffraction intensities must be measured at extremely high exposures without the benefit of Bragg amplification. In these regimes, radiation damage becomes a significant problem. Soft X-ray lasers currently operating can produce about 1E12 photons in 10 fs. It is expected that damage will be minimized if the pulse terminates before significant nuclear or valence electron motion occurs. Since most diffraction is due to valence electrons, this requires a pulse duration shorter than the Auger recombination time of a few femtoseconds. While this may not be readily achievable using ultrafast electron pulses, their high scattering cross sections in most materials make them a promising candidate for imaging transient states, albeit at lower spatial resolutions, without crossing the damage threshold.

Recent results from DTEM have clearly demonstrated the efficacy of electron pulses for studying material dynamics (Kim, et al., 2008). In order to directly image structural changes taking place in metallic and semiconducting materials in the DTEM it is desirable to overcome spatial and temporal limitations induced by Coulombic effects in an electron pulse as it travels through the microscope column (Armstrong, et al., 2007). These effects become particularly detrimental to image resolution at crossover points present when the electron lenses are set to imaging mode. One approach to solving this problem is to collect data in diffraction mode (where the crossovers are much less compact) and reconstruct images using CDI.

The high brightness, ultrafast electron sources that are currently being developed also show great promise for use in dynamic CDI experiments (Claessens, et al., 2005; van Oudheusden, et al., 2007). However, the columbic effects mentioned above, degrade both temporal and spatial coherence of an electron pulse as it travels from the sample to the detector. This complicates the phase retrieval problem for an electron pulse compared to that of an x-ray pulse of similar duration. Currently, computational methods for inverting diffraction patterns in CDI experiments rely on iterative phase retrieval algorithms. While these algorithms have led to some striking successes, the development of more robust and reliable methods will likely prove crucial for the extension of CDI to ultrafast electron diffraction.

Typically, these techniques rely on two well known and widely used algorithms: the error reduction (ER) algorithm first described by Gerchberg and Saxton and the hybrid input output algorithm (Fienup, 1982). While these algorithms are extremely powerful at exploiting local minima, their tendency to stagnate in said minima can give rise to ambiguity in determining convergence and uniqueness. These problems have been discussed at length in the literature and several suggestions for overcoming them have been proposed (Fienup and Wackerman, 1986; Chen, et al., 2007). Most of these suggestions rely on using multiple sets of random starting phases, referred to from here on out as seeds. These population based approaches are closely related to many algorithms that are common in the field of global optimization. This relationship suggests that tapping into the vast amount of knowledge that has been produced on the subject of global optimization algorithms could yield new insight into heuristics that can more reliably provide convergent and unique solutions to phase retrieval problems.

While several thorough and well written reviews exist in the literature on the subject of iterative phase retrieval, it is worthwhile to reexamine the basic algorithms in the context of global optimization (Spence, 2007; Marchesini, 2007; Miao, et al., 2008; Bauschke, et al., 2002). For a given recorded diffraction intensity, **I**, the initial seed is given by

$$G_1(u) = \left|I(u)^{1/2}\right| \exp[i\theta_1(u)] = \left|F(u)\right| \exp[i\theta_1(u)] \qquad \text{Eq. 24}$$

where θ_1, is an array of random numbers between 0 and 2π that serves as an initial estimate of the phases, and F(u) is the modulus. A typical iterative phase retrieval algorithm (in this case HIO) then proceeds as follows, where β is constant parameter typically chosen to be between 0.5 and 1 and k is the iteration number:

1. $g'_k(x) = \Im^{-1} G'_k(u)$
2. $g_{k+1}(x) = \begin{cases} g'_k(x) & x \in S \\ g_k(x) - \beta g'_k(x) & x \notin S \end{cases}$ Eq. 25
3. $G_{k+1}(u) = \left|G_{k+1}(u)\right| \exp[i\theta_{k+1}(u)] = \Im g_{k+1}$
4. $G'_{k+1}(u) = \left|F(u)\right| \exp[i\theta_{k+1}(u)]$

The support region, S, is either known a priori or can be determined by a method known as shrink wrapping (Marchesini, et al., 2003). In the case of ER step 2 is simply replaced with:

$$g_{k+1}(x) = \begin{cases} g'_k(x) & x \in S \\ 0 & x \notin S \end{cases} \quad \text{Eq. 26}$$

In the context of global optimization, and more pertinently MAs, heuristics such as ER and HIO that operate on only one member of a population are referred to as local searchers. Many of the global optimization techniques that could be incorporated into an MA for phase retrieval require an objective function that is to be either minimized or maximized. Most of the iterative phase retrieval algorithms discussed in the literature utilize error metrics only for monitoring convergence; however, these metrics are well suited for use as objective functions to be minimized. The objective function used in this work is given by

$$E_{Fk} = \frac{\sqrt{\sum_u \left[|G_k(u)| - |F(u)| \right]^2}}{\sum_u |F(u)|^2} \quad \text{Eq. 27}$$

and is commonly referred to as the Fourier space error metric in the CDI literature. For a given algorithm this error metric is typically a good indicator of image quality. However, it is important to note, that when comparing two algorithms error metric alone is not sufficient to determine which solution will produce a higher quality image. That is to say, a solution produced by HIO with a given error value will not necessarily yield a higher quality image than a solution generated by SOPR with the same error value (Bauschke, et al., 2003).

There are several examples in the literature of algorithms that incorporate global optimization concepts for use with iterative phase retrieval. The earliest example of this was discussed by Fienup in 1986 and described as a “Voting Method” for eliminating stagnation in minima that are characterized by erroneous sinusoidal striations in the reconstructed image (Fienup and Wackerman, 1986). This technique essentially employs a recombination operator, akin to those used in Genetic Algorithms (GAs) in the Fourier domain. It starts with 3 random seeds, allows them to proceed to stagnation using HIO and combines them to form a new solution that will ideally proceed via HIO without further stagnation.

A more recent example is the Guided HIO (GHIO) method (Chen, et al., 2007). This algorithm is directly derived from a technique known as Guided Simulated Annealing and can be accurately classified as a MA. GHIO, as described, takes 16 seeds and performs 1000 rounds of HIO to form the first generation of solutions. An error metric similar to that shown in Eq. 27 is then used to rank the solutions. The solution with the lowest error is taken as the "template", $\rho^{0,template}$. The remaining solutions are then aligned to $\rho^{0,template}$ and the next generation is created by taking the square root of the product of $\rho^{0,template}$ and the mth solution from the current generation

$$\rho^{g+1,m}(x) = \sqrt{\rho^{g,template}(x) \times \rho^{g,m}(x)} \qquad \text{Eq. 28}$$

This process is continued until a satisfactorily low error is achieved.

Memetic Algorithms is a term originally coined to describe algorithms that combine GAs with local search heuristics. The term has subsequently been expanded to include algorithms that incorporate any population based global optimization technique with local search methods (Moscato and Cotta, 2003). A cursory glance at the MA literature reveals that Guided HIO can be accurately classified as a very basic MA that incorporates a recombination operator interacting on a fixed schedule with two local search heuristics, in this case HIO and a variant of the ER algorithm. According to the MA taxonomy proposed by Krasnogor et al., GHIO is an MA of index D=2 (Krasnogor and Smith, 2005). Considering GHIO in the context of MAs opens several avenues for exploring new metaheuristic approaches to the phase retrieval problem. GHIO is only scratching the surface of this vast field and its success over simple HIO at seeking global optima in phase retrieval search spaces indicates that these approaches deserve a more thorough examination.

2. Methods

Swarm Optimized Phase Retrieval (SOPR) represents another attempt to utilize MA approaches for solving the phase problem. The fundamental difference is that instead of employing a recombination operator akin to those used in GAs, SOPR uses a population based approach known as Particle Swarm Optimization (PSO) as its global search component (Kennedy and Eberhart, 1995). PSO takes its inspiration from the behavior of birds flocking around a food source. Each member of the population, or

particle, uses its own personal best solution and the best solution from the entire population, the global best, to update its location in the search space. This task is performed through a velocity update given by

$$V_{n+1} = c_1 V_n + c_2 r_1 \circ (P_B - P_n) + c_3 r_2 \circ (G_B - P_n) \quad \text{Eq. 29}$$

Where P_n is the particle's current solution, P_B is the lowest error solution obtained by the particle, G_B is the global best solution, the r terms are random numbers between 0 and 1 (regenerated for each iteration) and the c terms are constant parameters. For all of the work described here $c_1 = 0.5$, $c_2 = 2$ and $c_3 = 2$. These values are commonly cited as good starting parameters in the PSO literature. The velocity vector is simply added to the current solution to update its position. This approach has been shown to outperform GAs in several classes of problems (Shi and Eberhart, 1999). Figure 19 shows an illustration of the objective function of a hypothetical 2D search space. Its worth noting that the actual direction a given velocity update will move a particle is dependant on both the constants parameters that have been selected as well as the random r values generated at that iteration.

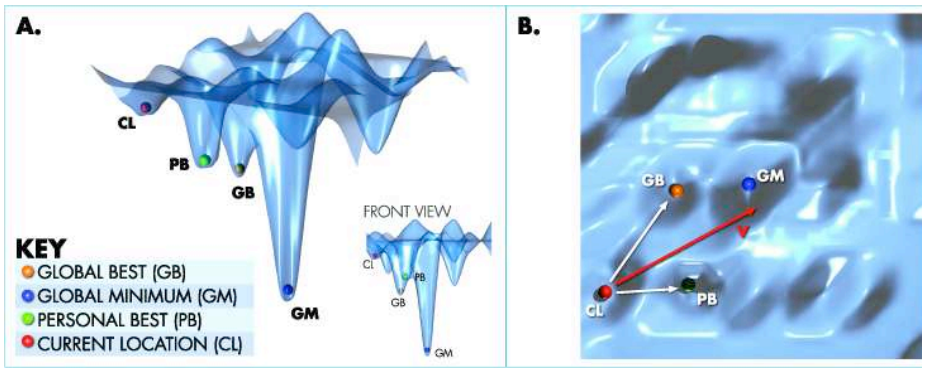

Figure 19: An illustration depicting the objective function of a hypothetical 2D search space with a single particle shown in red. Part B shows the plan view of the space along with the velocity update vector and the relevant particle positions that contribute to that vector. Illustration by George Suarez.

The starting seeds are initialized by performing 100 iterations of HIO on each particle. These particles are then run through the main iterative cycle of the algorithm which consists of alternating rounds of local search using HIO and ER and global search using the velocity update from PSO. Some particles will arrive in local minima containing twinned, or conjugate, images. While this isn't typically a problem when

using conventional HIO approaches, the velocity update step in SOPR can introduce severe artifacts if a given particle's current position is closer to the conjugate of the global best position. To avoid this, the SOPR variants employ the reduced area support constraint method, as described by Fienup, following the PSO velocity update for the first 10 iterations (Fienup and Wackerman, 1986).

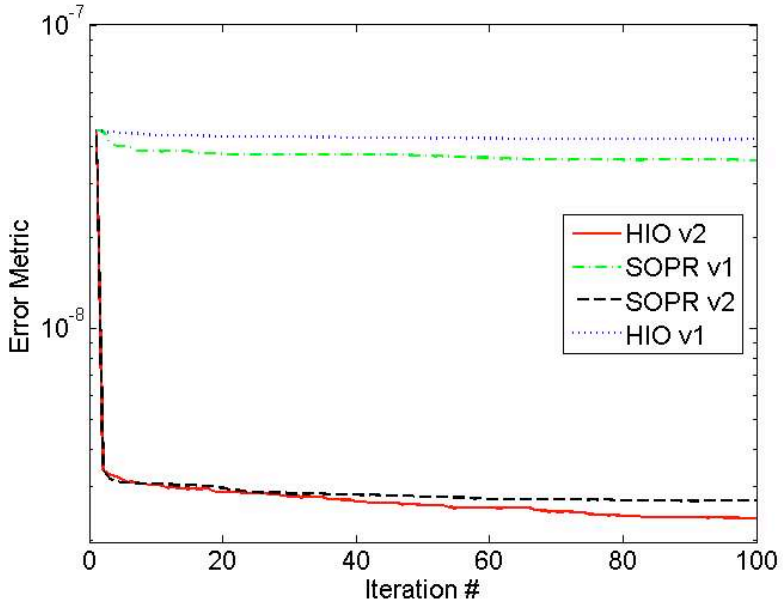

Figure 20: Error Plots for the HIO and SOPR comparison.

3. Results

SOPR was compared to both GHIO and HIO directly In addition, common local search heuristics involving HIO and ER were compared with and without PSO velocity updates included. The comparison between SOPR and HIO/ER was made using an image of a butterfly as the test data. This image was chosen because of its circular support which is a notoriously difficult problem for iterative phase retrieval techniques. Figure 20 shows a plot of the error metric, given by Eq. 27, for the global best (GB) solution for each algorithm averaged over 5 runs each with different starting seeds. Figure 22 shows the global best solutions produced by each algorithm, the original image for qualitative comparison, as well as block diagrams describing the differences in each variant.

The comparison between GHIO and HIO is made by taking the local search functions from GHIO, as described in the literature, and simply replacing the GHIO recombination operator, given in Eq. 28, with the PSO velocity update given in Eq. 29 (Chen, et al., 2007). Additionally, variants of each algorithm, in which the GB selection is performed after both local search steps, are compared. In these cases, an image of a crater on the moon is used as the test data. Figure 23 shows a plot of the error metrics for these algorithms. For these comparisons the ER variant described by Miao et al. is used and is indicated as ER* (Chen, et al., 2008). Figure 24 shows the GB solution produced by each algorithm, the original image, as well as block diagrams describing each. The image obtained from GHIO v1 after the ER* and recombination steps is included as well to show that the image quality does not significantly improve. Figure 25 shows the error metric for each of these algorithms tracked after every local search iteration (HIO or ER*) over the final 5000 iterations.

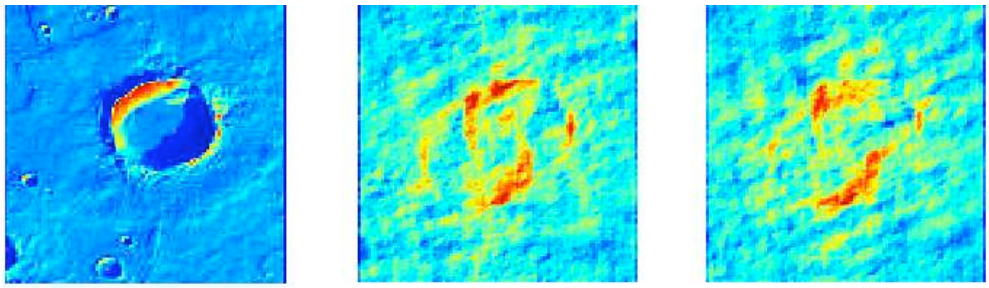

Figure 21: Results of SOPR and HIO reconstructions with simulated partial coherence.

It is anticipated that limited coherence in ultrafast electron pulses may adversely impact the quality of CDI reconstruction from UED data. The effects of partial coherence on image reconstructions have been discussed extensively in the literature and it has been shown that they can be mitigated by introducing phase curvature in the illuminating field (Williams, et al., 2007). While this may be experimentally achievable for CDI experiments in DTEMs, it would be advantageous to computationally mitigate this problem for UED systems where producing curved phase illumination is not typically feasible. To this end HIO and SOPR were compared against the crater data set with simulated partial coherence.

The effect of partial coherence on the simulated diffraction intensities was implemented using the method described by (Vartanyants, et al., 2001). Figure 21 shows the results of typical run with a coherence length equivalent to 1.5 times the length of the object in both the x and y directions. While the SOPR case performs marginally better than HIO at resolving the main feature of the image, both results show severe doubling and completely fail to resolve low intensity features. These results imply that the experimental challenges associated with partial coherence are not likely to be mitigated by the introduction of metaheuristic approaches to phase retrieval.

4. Discussion

There are two very important points worth noting when comparing algorithms in this manner that are often overlooked in phase retrieval literature. First, no heuristic will perform ideally for all instances of a problem. That is to say, any two optimization algorithms are equivalent when compared over all possible problems (Wolpert, et al., 1997). This is typically referred to as the "No Free Lunch Theorem," and it implies that the only way a given heuristic can outperform another is if it is specialized for the problem at hand (Ho and Pepyne, 2002). Second, as was discussed above, a lower error metric value will only imply a higher quality image for that specific algorithm. This means that qualitative assessment of image quality is crucial for comparing the results of two algorithms at a given error value.

These points do not rule out the possibility of comparing two algorithms; they merely imply that careful consideration must be taken when making the claim that a given phase retrieval algorithm outperforms another. While no such claim is made here, from the results shown above the efficacy of SOPR is clear. When compared to HIO alone, adding a PSO velocity update step clearly improves the rate of convergence.

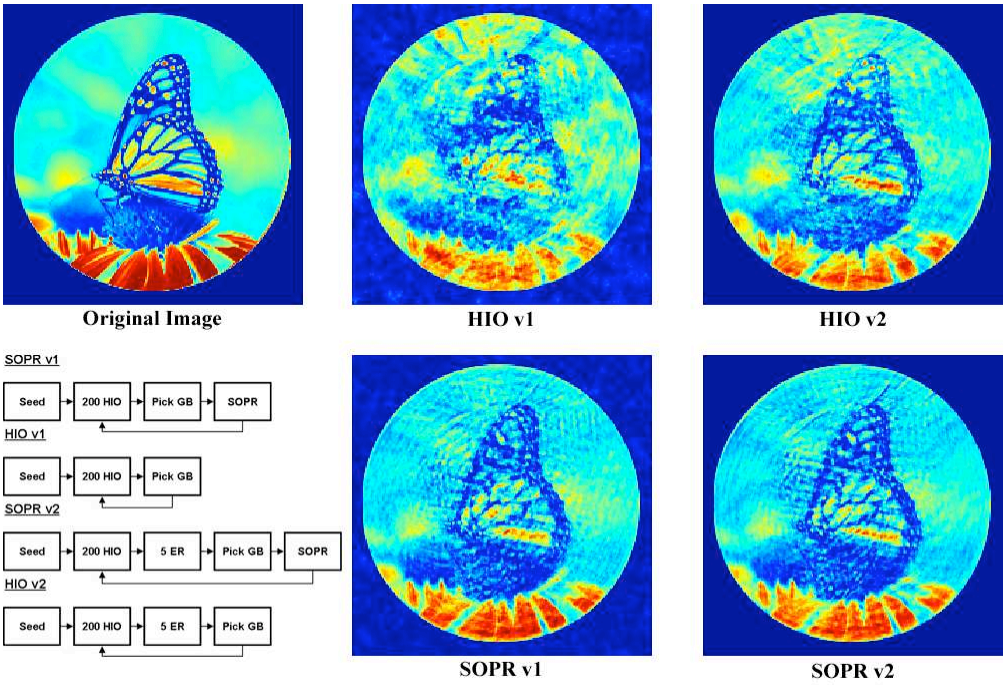

Figure 22: Solutions produced by each variant along with the original image. All images are colored based on intensity.

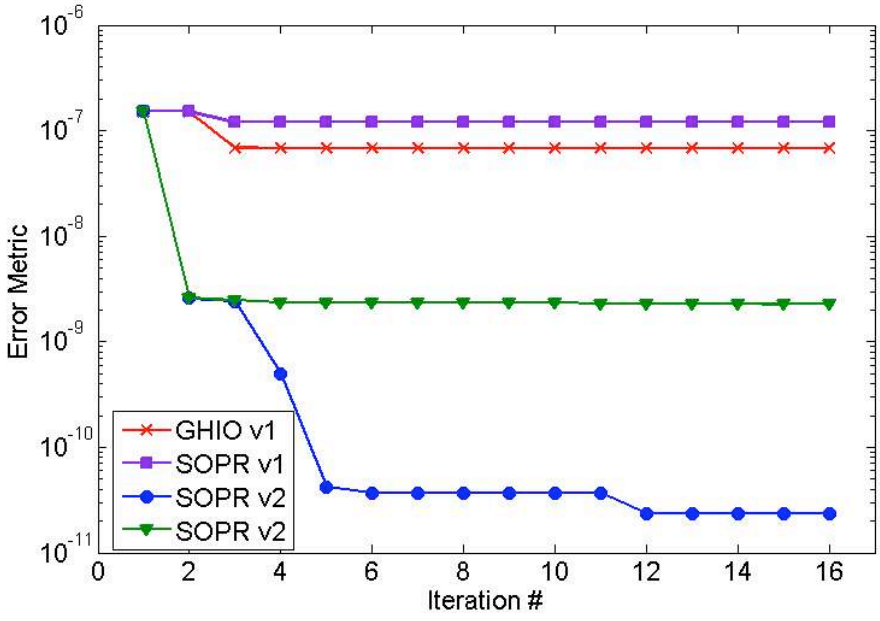

Figure 23: Error Plots for the GHIO and SOPR comparison. Error Plots for the GHIO and SOPR comparison.

However, HIO is seldom used by itself for iterative phase retrieval. Typically several rounds of the ER algorithm are performed at specified intervals. The second variants of the algorithms compare such a heuristic, and again, SOPR out performs HIO, although much less dramatically. It is also worth noting that the SOPR variants appear to converge quicker following the cessation of the reduced area support constraint method at iteration 10.

Comparing the GHIO and SOPR variants, it is quite obvious that slight adjustments in the heuristic procedures lead to vastly different results. In these cases simply switching the position of the rankfinding step not only produces better results, but reverses which global update is more effective. Qualitative comparison of the images shows that all four variants produce high quality solutions free of the common stagnation issues associated with non-population based heuristics. Careful observation of the low intensity regions of the image (particularly the shadow of the large crater) shows that the SOPR v2 variant is indeed the closest solution as indicated by the error metrics. Moreover, erroneous vertical striations are visible in the low contrast regions of the GHIO solutions that appear in neither the original image nor the SOPR v2 solution.

5. Conclusions

Given these results it is clear that SOPR is a viable method for phase retrieval from diffraction intensities. However, it cannot be stressed enough that these comparisons (and all others in the literature for that matter) are only useful for the problems at hand. The important conclusion is that global optimization techniques, and more specifically MAs, show great promise as tools for the development of more robust and reliable metaheuristics for phase retrieval.

SOPR and GHIO are clearly effective metaheuristic approaches to solving the phase problem. They are both simple MAs that, in some cases, out perform the more traditional algorithms for phase retrieval such as HIO and ER. The clearest implication that can be garnered from the comparisons made here is that MA approaches to phase retrieval deserve far more attention than they have been given to date. The development of more advanced MAs that incorporate the multitude of highly successful local search heuristics for phase retrieval; “learning” from different problem instances; and more

advanced scheduling routines may very well lead to a general purpose phase retrieval metaheuristic.

The scope of phase retrieval problem spaces derived from experimental data is vast and grows larger with each CDI technique that is developed. However, the prospect of producing high time and spatial resolution images of dynamic material processes through ultrafast electron diffraction is enticing enough that the search for a nearly universal approach to tackling the problem is warranted.

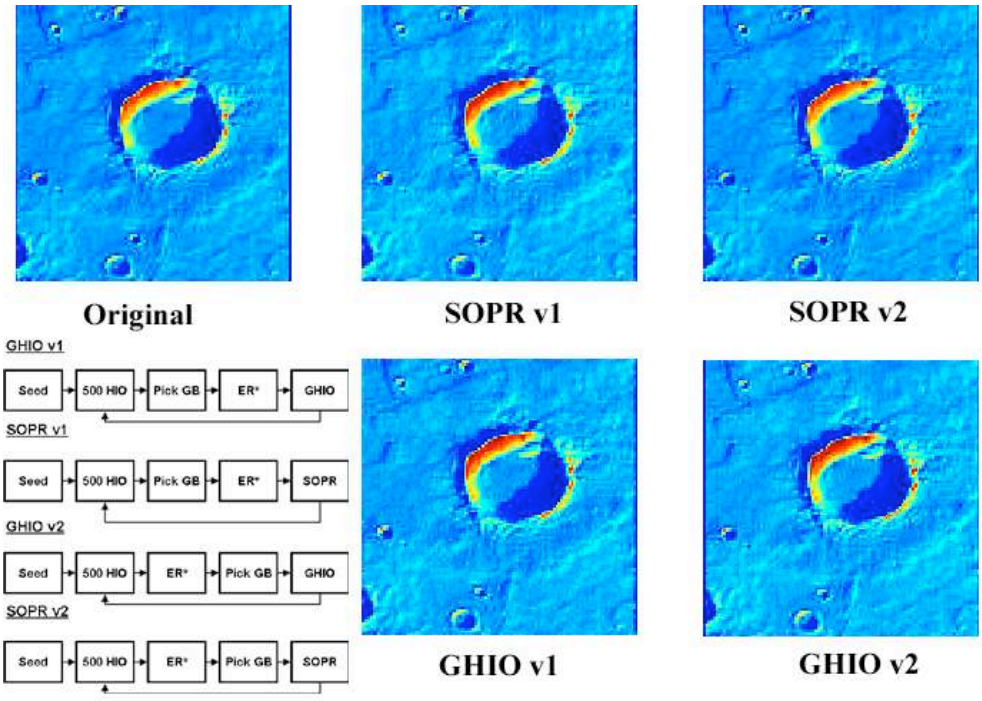

Figure 24: Solutions produced by each variant along with the original image. All images are colored based on intensity.

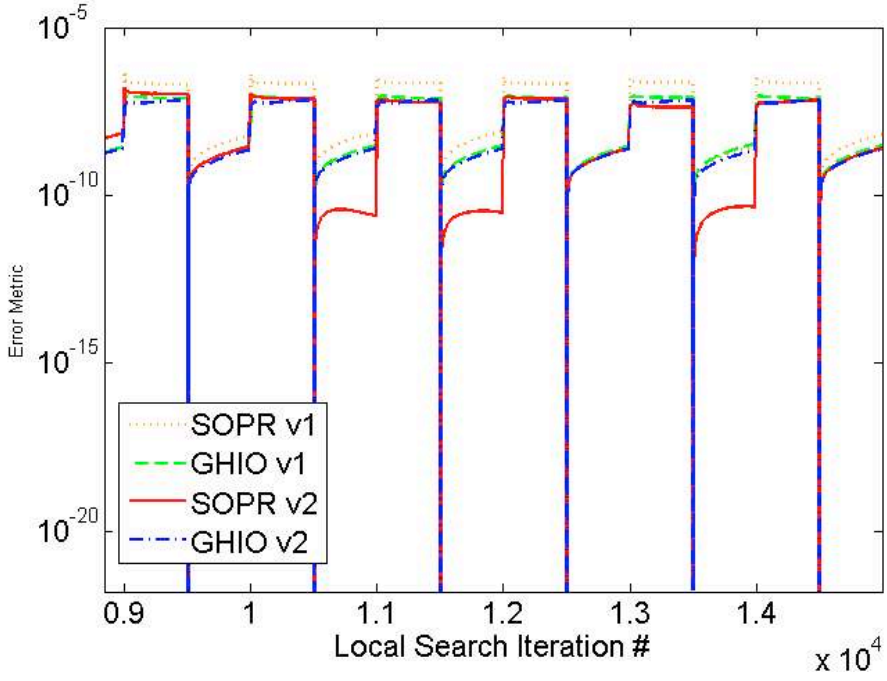

Figure 25: Plot of the error metric for the GHIO and SOPR variants measured after each of the final 5000 local search steps.

Chapter 7 Using the GPU for the Simulation of Protein Structural Dynamics

1. Introduction

While complete ab-initio structure determination of proteins by molecular dynamics still remains a challenging task, determining the changes a protein makes while undergoing conformational shifts is a well constrained optimization problem which can be easily linearized. That is, picking the optimum folding pattern from the near limitless set of possible arrangements present in conformational space is a far harder problem than moving along the path of minimum action between two known states.

The linearization of this problem proceeds as follows. First the protein's two states are mapped to vectors in a 3N dimensional space, representing the spatial coordinates of each atom with a total number of N atoms. Next, quadratic potentials are determined and catalogued in two 3Nx3N "connectivity matrices" Q, P defining internal energies and, essentially, bonds. Finally a constant "Pulling Force" is

determined for each atom yielding a constant force vector F_c. The conformational trajectory is then simply that which minimizes the Onsanger-Machlup action:

$$S = \frac{1}{2}\int (\dot{X} - F(X))^2 dt \qquad \text{Eq. 30}$$

This minimization can be alternately represented as the minimization of a "Generalized Energy" in the form of:

$$U_\ell = \frac{1}{2}(X_0 - X_A)^T Q(X_0 - X_A) + F_c(X_0 - X_A) \qquad \text{Eq. 31}$$

Minimizing this energy is then simple linear algebra followed by differentiation:

$$\frac{dU_\ell}{dz} = Qz + F_c = 0 \rightarrow z = -Q^+ F_c \qquad \text{Eq. 32}$$

With $z=(X_0-X_A)$ and Q^+ the pseudo-inverse, formed by inverting the non-zero entries of it's associated diagonalized eigenvalue matrix:

$$Q = V^T \mathrm{E} V \rightarrow Q^+ = V^T \mathrm{E}^+ V \qquad \text{Eq. 33}$$

Thus finding the optimal path becomes a diagonalization, inversion problem.

Generally, then, this problem decomposes into a series of one of the following algebraic operations:

$$\begin{aligned} &z = \alpha \cdot op(A) \cdot x + \beta \cdot y \quad \text{(Here op(A) is either } A^T \text{ or A)} \\ &z = \alpha \cdot x + y \\ &D = \alpha \cdot op(A) \cdot op(B) + \beta \cdot C \\ &op(A) * x = b \rightarrow x \end{aligned} \qquad \text{Eq. 34}$$

Fortunately such operations are exactly those modern graphics cards typically use in 3D calculations such as for surface normal calculation or for determining lighting coefficients. cudaBlas (http://www.nvidia.com/object/cuda_home.html) is an NVIDIA based BLAS (http://www.netlib.org/blas/) wrapper allowing access to most of the level 1 and level 2 based BLAS functions in an accelerated environment. Additional commands initializing the GPU for use and routines for pulling and pushing data to and from the card are also included.

An existing software package written by J. Franklin entitled "LNM" already makes extensive use of BLAS calls to solve exactly the above problem. Using this fact

we wrote a complete gsl_blas and gsl_linalg wrapper in the form of cuda_blas and cuda_linalg along with all other appropriate definitions such as cuda_vector and cuda_matrix to mirror their gsl counter-parts.

Modifying this program was then a process of replacing all gsl_* calls and definitions with associated cuda_* commands and definitions. Additionally most previously un-optimized routines involving matrices and vector calculations were migrated to this context. (Many vector-vector sums were formerly performed explicitly, for instance.)

```
cuda_vector *cuda_vector_alloc(size_t size);
cuda_vector *cuda_vector_calloc(size_t size);
cuda_vector *cuda_vector_new(gsl_vector *g, bool sync=true);
void cuda_vector_free(cuda_vector *v);
void cuda_vector_push(cuda_vector *v);
void cuda_vector_pull(cuda_vector *v);
void cuda_vector_set(cuda_vector *v, size_t index, float value); //Desyncs data
void cuda_vector_set_zero(cuda_vector *v, bool push = true);  //Desyncs data when push false
float cuda_vector_get(cuda_vector *v, size_t index);
void cuda_vector_memcpy(cuda_vector *dest, cuda_vector *src);
void cuda_vector_sync(cuda_vector *v);

// Matrix Operations
cuda_matrix *cuda_matrix_alloc(size_t size1, size_t size2);
cuda_matrix *cuda_matrix_calloc(size_t size1, size_t size2);
cuda_matrix *cuda_matrix_new(gsl_matrix *gm, bool sync=true);
void cuda_matrix_free(cuda_matrix *m);
void cuda_matrix_push(cuda_matrix *m);
void cuda_matrix_pull(cuda_matrix *m);
void cuda_matrix_set(cuda_matrix *m, size_t index, size_t jndex, float value); //Desyncs data
void cuda_matrix_set_zero(cuda_matrix *m, bool push = true);  //Desyncs data when push false
float cuda_matrix_get(cuda_matrix *m, size_t index, size_t jndex);
void cuda_matrix_swap_rows(cuda_matrix *m, size_t i, size_t j);
void cuda_matrix_sync(cuda_matrix *m);

// Universal Operations
void cuda_pull(cuda_matrix *m);
void cuda_pull(cuda_vector *v);

void cuda_sync(cuda_matrix *m);
void cuda_sync(cuda_vector *v);

// Permuation Operations
void cuda_permute_vector(const gsl_permutation *p, cuda_vector *v);

/* CUDA BLAS Wrapper */

void cudaError(char *error);
void checkStatus(cublasStatus status);

float cuda_blas_dot(cuda_vector *x, cuda_vector *y);
int cuda_blas_sgemv(cudaTranspose_t TransA,
                    float alpha,
                    cuda_matrix *A,
                    cuda_vector *X,
                    double beta,
                    cuda_vector *Y);

void cuda_blas_sgemm(cudaTranspose_t TransA, cudaTranspose_t TransB,
                     float alpha,
                     cuda_matrix *A,
                     cuda_matrix *B,
                     float beta,
                     cuda_matrix *C);

void cuda_blas_saxpy(float alpha,
                     cuda_vector *x,
                     cuda_vector *y);


/* CUDA LINALG Wrapper */
/* Fixme LU_decomp requires optimization */
void cuda_linalg_LU_decomp (cuda_matrix * A, gsl_permutation * p, int *signum); // Desyncs data
void cuda_linalg_LU_solve (cuda_matrix * LU, const gsl_permutation * p,cuda_vector * b, cuda_vector * x
);
```

Figure 26: The list of functions available largely mirrors existing GSL calls with the addition that behind the scenes memory transfers are performed automatically and calculations are GPU based.

While sending data to the GPU is essentially no slower than writing to RAM, reading from the graphics card is quite slow by comparison. For this reason it is necessary that data transfers me minimized especially with regard to "get" requests from the GPU. To this end, an automatic syncing method was established to pull or push data a minimum number of times without need for user interaction. More importantly a user is free to perform calculations without excessively worrying about the physical location of the data.

2. Methods and Materials

The entire library was written in object free c compatible with NVIDIA's nvcc compiler and was created as a separate library to allow for mixed c and c++ code. External libraries included GSL (http://www.gnu.org/software/gsl/) and cudaBlas.

Simulations were run on a Dell Amd 64 computer running Fedora Core Linux 7.0. The graphics card used for calculations was an NVIDIA GeForce 8800 GT with 512 Mb of onboard RAM.

Datasets included in the package were tested, though specifically the 637 atom "bacillus" protein (no other name provided) moving between the closed and open states.

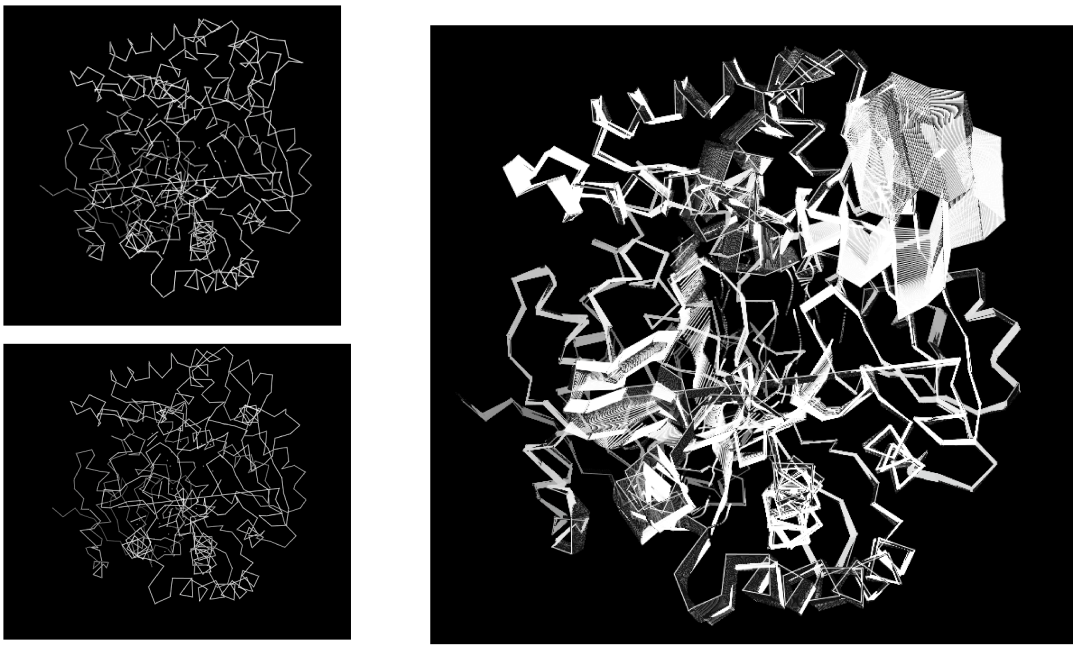

Figure 27: (Left Top) Initial state of protein. (Left Bottom) Final state. (Right) Trajectory through conformational space minimizing action, showing changes mainly in the variable regions of the conformations.

3. Results

For extremely small test systems consisting of 2 atoms yielding 6x6 matrices, runs were completed within 98ms for the GSL simulation vs 202ms for the cuda simulation. The results of the larger test system, comprised of 637 atoms and 1908x1908 matrices, can be seen in Figure 27. Native CPU GSL computations required 25m25s to complete, cuda GPU enhanced simulations were complete within 1m10s, yielding a speed increase of 21.8x compared to the un-accelerated case.

4. Discussion

The speed increase derived from GPU acceleration was achieved at relatively low cost and replaced essentially all relevant calculations in this project.

4.1 Future Directions

Algorithmic optimization would be necessary to improve the speed of calculation for LNM. Currently a version minimizing the matrix equation $\|Qz + F_c\|$ (as opposed to matrix inversion) has resulted in a 15x speed increase via CPU calculations. It is proposed that a combination of this with cudaBlas would yield a 400x speed increase.

Currently all calculations are performed in single precision. In cases of matrix multiplication with large systems, errors easily accumulate, resulting quickly in significant truncation problems. Future versions of cudaBlas are due to address this issue with the addition of 64 bit processing.

A change of coordinates from Cartesian to Dihedral angles would drastically reduce the size and sparsity of the Q and P connectivity matrices. This will require a reformulation of the Onsanger-Machlup action, however. Ideally this would reduce the matrix sizes overall increasing calculation speed significantly.

5. Significance

In any simulation time is a precious commodity, though in few is it more so than with molecular dynamics calculations. Many simulations can take days, weeks or even months to complete. A time reduction of 2x would reduce a 12 day calculation to 6. The reported 21.8x time reduction would reduce a day long calculation to essentially an

hour or a month long calculation to a few days. While a predicted 40x time reduction should be possible for larger matrices, the importance of even a 22x reduction (on a single computer) cannot be overstated. Such an improvement would, in principle, allow a 1ns simulation to be extended to 20ns, or the practical maximum size of a simulated protein to be increased to biologically relevant sizes. (600 atoms is extremely small by most standards).

Chapter 8 Reasonable Simplifications for a Larger Biological Story: Kinetic Monte Carlo Simulations

1. Introduction

The mechanics of protein association and protein-protein interaction provide a nearly infinite array of continuously varying parameters. Describing even comparatively simple systems, such as protein dissociation, quickly leads to a computationally unfeasible problem when dealing with systems of even a few proteins.

Instead, chemists and biologists have long tackled this problem by describing systems using idealized ensemble descriptions with experimentally determined stochastic parameters such as rates of reaction. While these systems, described by "master equations", provide quantitatively accurate results they do so for mainly for equilibrium states, illustrating average, global behaviors and offer no information about dynamic or transient processes. These models are usually decreasingly accurate, even non-applicable, for non-ideal cases such as partial mixing, systems with hysteresis, non-contiguous boundary conditions, etc.

For such non-ideal cases, or for cases where differential master-equations must still be solved numerically, an alternative description of reactive systems has been proposed in the form of Kinetic Monte Carlo (KMC) simulations(Gillespie, 1976). KMC abstracts a reactive system into idealized species which are free to interact according to a rule-set appropriate to the system, with probabilistic parameters determined from experiments, usually in the form of rates of diffusion, Ka, etc. Species consist of single atoms which interact among themselves and other species in a strictly binary mode, that is "interacting" and "not interacting."

We applied KMC to a common ligand-receptor problem in biology. That is, the case where two species A and B have a probability to react and form C, which has a probability of dissociating. i.e.

$$A + B \leftrightarrow C \qquad \text{Eq. 35}$$

This can be shown to lead to the following differential equation relating the three species:

$$\frac{dC}{dt} = K_{on}AB - K_{off}C \qquad \text{Eq. 36}$$

Where K_{on} and K_{off} are association and dissociation constants which can be determined experimentally. This relation immediately leads to the result that, at equilibrium, where $\frac{dC}{dt} = 0$:

$$Ka = \frac{Kon}{Koff} = \frac{C}{AB} \qquad \text{Eq. 37}$$

Which is to say that for an experimentally determined K_a the relative equilibrium population of the species involved can be determined.

This system can be simulated in the KMC formalism as follows. A rectangular array is defined in either two or three dimensions and a number of cells are populated with "particles" in initial proportions. This typically is represented as an array of integers with possible values 0,1,2,3 (with 1,2,3 each representing a member of species

A,B,C and 0 representing empty space). Choosing initial conditions representative of the system of interest is important and for a well mixed reaction starting with an equal measure of A and B the array can be randomly populated using A=B, C=0 (Figure 28).

The second part of KMC is the reaction step, outlined in the following:

- Randomly choose a particle.
- Choose whether to react or diffuse based on P_{on}, the probability of association, or P_{off}, the probability of disassociation, depending on whether the choice was A, B or C.
- If diffusing, randomly choose a direction and move to new location.
    - If the new location is not valid, do nothing.
- If reacting and particle is of A or B, randomly choose a neighboring partner to react with
    - If no neighboring reactable partner, do nothing
    - If reactable partner, remove partner and replace particle with member from C.
- If reacting and particle is of C, randomly choose a direction to dissociate
    - If no available direction, do nothing
    - If available direction, randomly choose from A or B and replace particle with choice, then replace empty neighbor in direction chosen with the opposite of the pair (i.e. A,B or B,A).

A single time step Δt, here termed a round, is defined as N reaction steps (with N the current number of particles.) Allowing this system to run until $\Delta C \approx 0$ should then yield an equilibrium state Figure 29 as defined in the master equation for the given parameters.

Up to this point this method yields a numerical approximation to the master equation. However, once a run has been performed using KMC a wealth of data is available including transient behavior and, unlike ensemble approximations, a variety of initial conditions can be tested in subsequent runs.

Organizing and understanding dynamic three dimensional data then requires a viewing system in order to observe time varying or local behaviors. To this end we

developed a viewer which allows users to view system evolution in real time, leaving one free to change parameters according to experimental conditions at any time.

Fitting the parameters derived from experiments into the model requires translation from reaction constants to equilibrium probabilities. In order to do this a series of test runs must be performed for various P_{on}, which will result in a series of K_a for given P_{on} which can be used to determine a P_{on} for any experimentally determined K_a. P_{off} can be determined directly from the relation:

$$K_{off} \cdot \Delta t = P_{off} \qquad \text{Eq. 38}$$

with Δt defined as before, as dissociation is a purely internal mechanism and requires no pair localization (excepting the requirement of enough free space to dissociate into).

2. Methods and Materials

The project was written entirely in C++ and was split up into several components requiring varying levels of external libraries in order to keep all elements as independent and generally applicable as possible.

Thus, the object defining the KMC system including 2/3 dimensional array as well as all "utility" functions including a random component were separated into their own, independent library which used standard C++ calls and STL only. This was to keep this library as platform and library independent as possible.

The random library was written to over-ride the standard c library random function which is notoriously non-random and was written in its entirety by Philippos Tsourkas, a Postdoc in the Raychaudhuri lab.

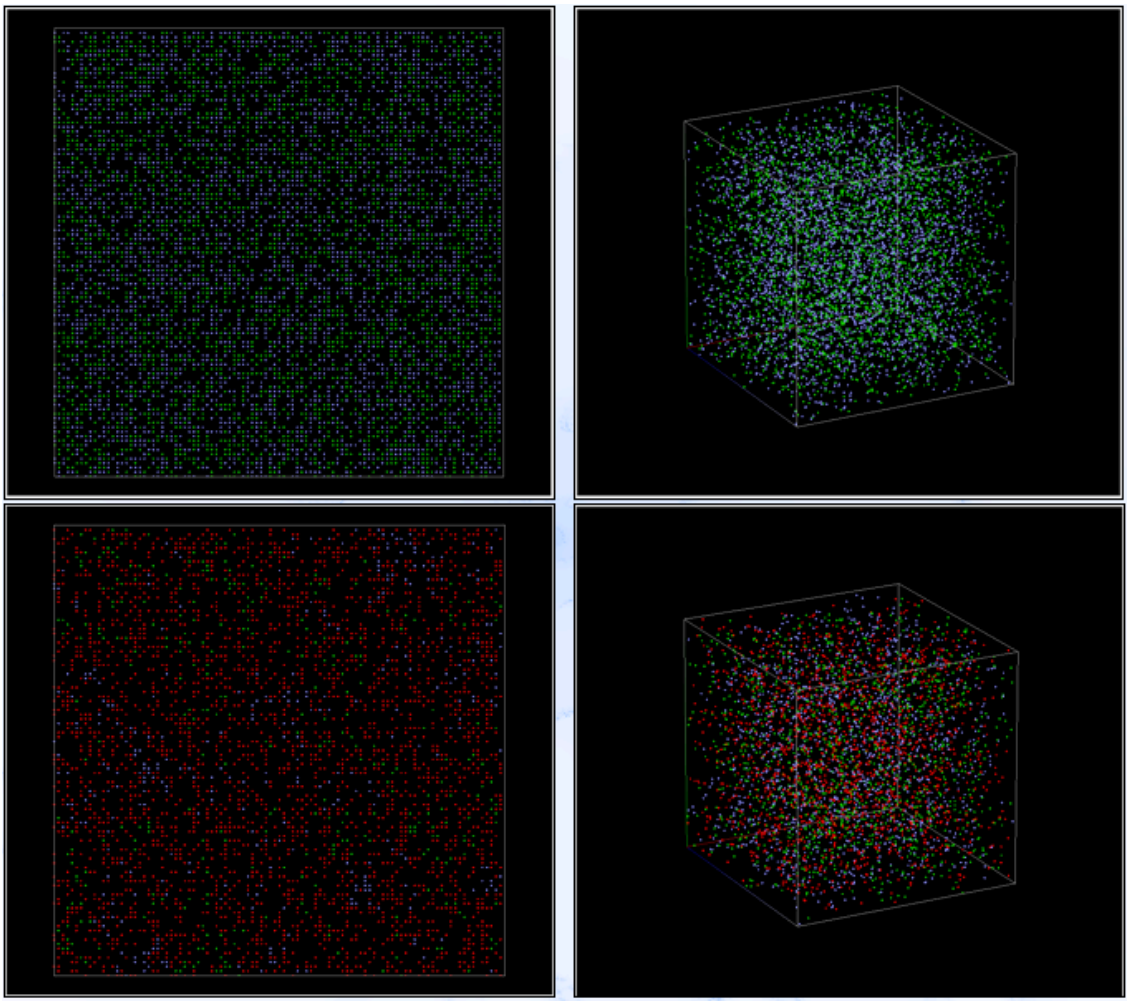

Figure 28: (Left Column) Initial (top) and equilibrium (bottom) states for a two dimensional system. (Right Column) Three dimensional system in initial (top) and equilibrium (bottom) states.

The KMC library could then be called either from a command line wrapper program which ran a simulation and output the data to a file or from the visualizer (Figure 28). This visualization tool was written using Qt-4.3 and took advantage of OpenGL in addition to standard Qt widget sets. The visualizer allows the user to change parameters during an experimental run as well as advance through reaction rounds and view the results in real time.

All runs providing data were performed on a 2.4 Ghz Duo-Core MacBook Pro with 4 GB of RAM. Some visualization was performed on an older Dell computer running Ubuntu Linux.

3. Results

A series of runs were performed with 2000, 3000, 5000 particles on arrays of 100x100x100 and 100x100x1proceeding for 2500 rounds before termination. The experimentally determined parameters were: Time to equilibrium T_{eq} = 60.0s Ka = 1.0, Koff=1x10^-2. The P_{on} was chosen as 0.5. Additionally a run was performed on a 200x200x200 system with 10,000 particles which ran for 10,000 rounds before terminating.

Convergence was achieved for the following data accordingly:

Particle Number	Time to Convergence
2000	1729
3000	1129
5000	1972
10000	10000+

The system with 10,000 particles did not converge and was terminated after 10,000 steps.

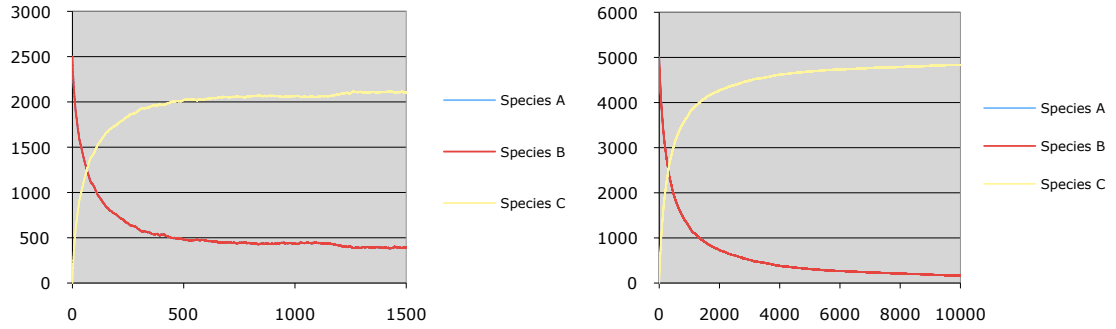

Figure 29: Time evolution of systems for 5000 particles in a 100x100x100 volume (left) and 10,000 particles in a 200x200x200 volume (right). Systems asymptotically approach their equilibrium distributions which still has not been achieved for the system of 10,000 particles even after 10,000 rounds.

4. Discussion

While the equilibrium and overall exponential nature of the development of the system conformed to the predictions of the master equation, real-time observation of the evolving system yielded interesting results. Though the system begins heterogeneously mixed, particles from A and B quickly react, depleting the region of A, B pairs. This results in clusters of unpaired A or B particles relatively well separated from one

another (Figure 30). Though these clusters have no internally attractive forces, they are an emergent natural result of slight inhomogeneities in the original distribution. These clusters can lead to sudden spikes in the population of products when groups of reactants "collide" and quickly pair.

Additionally, a non-time-local phenomenon of dissociation followed by rebinding some short time period later was observed for members of C. That dissociation forces products to be proximal immediately implies that K_{off} is skewed in a non-linear manner by P_{on}. This result, while perhaps not surprising, shows details about interactions not assumed by ensemble methods.

While it might be expected that a 100x100x100 cube might be small enough that edge effects would skew results, boundaries were found to play little role in that they only mildly affected diffusion and did not obviously affect dissociation for systems which were sufficiently sparse. Equilibrium was usually achieved within 1500 steps, likely owing to the relatively small reaction space. Interestingly, increasing the reaction space to 200x200x200, even with the addition of 10,000 particles caused the system to fail to converge after 10,000 steps (though characteristic asymptotic decay was still observed,) showing the system's non-linear response to increased volume, compared with its semi-linear response to increased particle number.

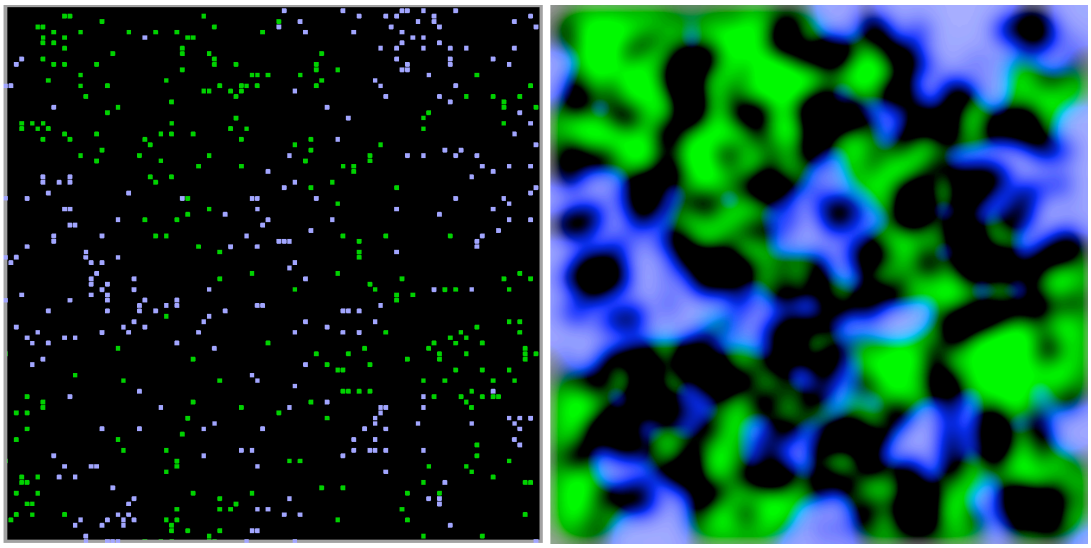

Figure 30: Large scale population distribution heterogeneities in a system nearing equilibrium. (Left) Particle positions of both species. (Right) Local density map of populations.

Though the KMC model and visualization tool were successfully implemented and run, a number of reasonable improvements could be proposed.

Automatic determination of a K_a vs. P_{on} curve for a given parameter set would greatly simplify the process of curve fitting as it is necessary for each concentration/relative proportion change, which is altered both by the enclosing volume and the amount of A, B particles present.

For optimization of the process, sparse matrix methods are needed as size requirements even for nearly empty, reasonably sized volumes can quickly become prohibitive. For a cube with a side length of 100 to grow to 200 requires an additional 26 Mb of storage, while growth to 700 requires 1 Gb extra RAM. This could easily be reduced at the expense of processing time, given that even 10^6 particles would require ~10 Mb of storage space, which would be independent of bounding volume.

Casting KMC as a Markov model would greatly help with determination, simplification and possibly elimination of run-time parameters. Further, it would give KMC a stronger theoretical connection to standard ensemble master equations.

The visualization tool could benefit from a number of improvements in data readout, such as displaying a growing population history graph or saving each round of data and allowing the user to progress to any stage in the systems development via a movie-control style slider bar.

5. Significance

While reacting inorganic reagents, in vitro, in well mixed, precisely balanced amounts, can lead to highly predictive models about reaction rates and equilibrium behavior in the lab, these conditions are essentially non-biological. For non-heterogeneous reactions involving reaction pathways, noncontiguous boundaries (i.e. cytoskeletal-transmembrane protein complexes,) electrical gradients, etc, such models are decreasingly useful. Kinetic Monte Carlo methods function to both numerically solve associated ensemble master equations in addition to being applicable in such cases where master equations are unfeasible and non-uniform conditions are imposed.

KMC is already being applied to study, for example, membrane diffusion (Nicolau, et al., 2007), and center bound exiton migration in dendrimers (Raychaudhuri, et al., 2002).

Concluding Thoughts and A Look to the Future

Historically, data processing was an intensive procedure, requiring painstaking care for known or determined variables. Moreover, since the parameters that guided image processing were typically defined repeatedly over several static script files, variation of parameters was difficult and led to a chance of error or inconsistency. More than simply a reduction in time, automation offers users the freedom to vary parameters in a uniform manner and observe the changes in output. This ability to "play" with the data hopefully will offer greater theoretical insight into the system at hand or, at a minimum, provide the user useful feedback on the sensitivity of results to inputs. Insights such as these allow optimizations in data-collection strategies as well as efficient time use in parameter determination. Lastly, exhaustive parameter space searches, such as that for finding optimal unbending parameters as described in Chapter 3 provide project-specific results for which no other obvious optimization (theoretical or otherwise) is currently known.

One of the primary goals of automation in the above projects has been to make data-acquisition, or biochemical work on proteins the rate-limiting step in protein structure determination. Besides simply improving the speed of the process, making

data processing independent of interaction allows a user to be free to take more data or run more experiments while the data is busy being processed on a remote server.

It was shown in Chapter 2 - Chapter 4 that data processing and initial reconstruction could be largely automated. It was shown in Chapter 5 and Chapter 6 that an increase in data, in a virtually limitless manner, leads to an increase in at least vertical (if not total) resolution. Finally, it was shown in Chapter 7 that significant speed increases can be quickly implemented through coding for the GPU. With these in mind, the following workflow is offered:

1. Collect data at the microscope from crystalline regions via intelligent auto-detection and automated imaging.
2. Refine data-processing automation to the point that only one-time initial user input is required to produce a 3D volume from input data.
3. Have 2dx_merge eternally watch an input folder into which raw data is dropped.
4. Update, using speed increases leveraged from the GPU, the 3D results based on the input of the new image.
5. If waiting for more data, use heuristic or exhaustive searches on parameter space to produce a globally optimum solution from globally optimized 2D data processing.

Such a workflow is both theoretically and computationally feasible with currently available hardware and software tools. The easiest of these to implement immediately would be the second and third steps, fully automated processing of datasets. These two advances alone would offer considerable benefit to the field of crystallography. In such cases, teams of crystallographers could focus on optimizing crystallization conditions leaving either a centralized server or decentralized cluster free to process dozens of simultaneous projects in the background.

This vision is already, in part, being realized for single particle projects in the form of Leginon (Potter, et al., 1999). Here a combination of automated data acquisition is combined with automatic particle picking and structure generation. In the case of electron crystallography, the problem is less challenging, as crystalline regions are more easily characterized than single particles. Further, the presence of the lattice eases

computational requirements, allowing efficient rotational orientation and automatic scaling as more-or-less direct results of lattice determination. As such, a similar system to characterize appropriate regions for data-acquisition and automate collection could potentially provide thousands of useful images in a few days work. Automatically processing them would then be a relatively simple matter of advancing the work presented in Chapter 3 and Chapter 4, focusing on development for areas that still require modest or large amounts of user interaction. Once processed, application of the advanced refinement algorithms presented here on a vastly oversampled dataset would, in principle, lead to dramatic resolution improvements.

The work presented in Chapter 5 and Chapter 6 are merely the beginning in terms of the possibilities offered by recent work on minimization. The least squares minimization described in Chapter 5, for instance, could be replaced by a far more complete L1 sparsity minimization. In this case, rather than trying to minimize the "energy" of the system, this type of optimization minimizes "non-zeroness", finding the fewest number of non-zero values needed to produce a given data set. This would have the benefit of applying a kind of "Shrinkwrap" at the minimization level, which would be completely compatible with the remainder of the iterative application. Such an advancement would likely decrease processing time, the number of pseudo-solutions and, improve the accuracy of the reconstruction.

Finally, in many cases, computers found in a lab sit idle for a majority of their workday. Even when a project is being processed, a majority of their time is spent waiting for more data or interacting with the user. Applying the increases in computing power, brought both by advanced CPU architectures hosting multiple cores and the remarkable developments in GPUs, to optimizing data parameters could bring a host of advantages. In particular, investigations of optimal tilt-orientation, perfect unbending parameters, and CTF determination could then be performed in the background, updating the dataset as appropriate. Even computationally costly alternatives such as maximum-likelihood could be running in parallel with current reconstruction methods, in the background, comparing results and integrating with the full dataset if appropriate. Such optimizations would quickly yield predictive power and would be relatively simple to implement on a large scale.

Future developments aside, combining the already existing tools presented here, along with a method for rapid, high-volume, image collection, would alone have wide ranging implications for the field of 2D crystallography and membrane protein structure determination generally.

Bibliography

1 Chiu, P. *et al.* The Structure of the Prokaryotic Cyclic Nucleotide-Modulated Potassium Channel MloK1 at 16 Å Resolution. *Structure* **15**, 1053-1064 (2007).

2 Emsley, P. & Cowtan, K. Coot: model-building tools for molecular graphics. *Acta Crystallogr D Biol Crystallogr* **60**, 2126-2132, doi:S0907444904019158 [pii], 10.1107/S0907444904019158 (2004).

3 Downing, K. H. Spot-scan imaging in transmission electron microscopy. *Science* **251**, 53-59 (1991).

4 Gipson, B., Zeng, X., Zhang, Z. & Stahlberg, H. 2dx—User-friendly image processing for 2D crystals. *Journal of Structural Biology* **157**, 64-72, doi:10.1016/j.jsb.2006.07.020 (2007).

5 Gipson, B., Zeng, X. & Stahlberg, H. 2dx_merge: data management and merging for 2D crystal images. *Journal of Structural Biology* **160**, 375-384, doi:10.1016/j.jsb.2007.09.011 (2007).

6 Crowther, R., Henderson, R. & Smith, J. MRC image processing programs. *Journal of Structural Biology* **116**, 9-16 (1996).

[7] Agard, D. A least-squares method for determining structure factors in three-dimensional tilted-view reconstructions. *Journal of Molecular Biology* **167**, 849 (1983).

[8] Henderson, R. *et al.* Model for the structure of bacteriorhodopsin based on high-resolution electron cryo-microscopy. *J. Mol. Biol.* **213**, 899-929 (1990).

[9] Henderson, R. & Unwin, P. N. Three-dimensional model of purple membrane obtained by electron microscopy. *Nature* **257**, 28-32 (1975).

[10] Grigorieff, N., Ceska, T. A., Downing, K. H., Baldwin, J. M. & Henderson, R. Electron-crystallographic refinement of the structure of bacteriorhodopsin. *J. Mol. Biol.* **259**, 393-421 (1996).

[11] Kuhlbrandt, W., Wang, D. N. & Fujiyoshi, Y. Atomic model of plant light-harvesting complex by electron crystallography. *Nature* **367**, 614-621, doi:10.1038/367614a0 (1994).

[12] Murata, K. *et al.* Structural determinants of water permeation through aquaporin-1. *Nature* **407**, 599-605 (2000).

[13] Ren, G., Reddy, V. S., Cheng, A., Melnyk, P. & Mitra, A. K. Visualization of a water-selective pore by electron crystallography in vitreous ice. *Proc Natl Acad Sci U S A* **98**, 1398-1403, doi:10.1073/pnas.041489198, 041489198 [pii] (2001).

[14] Miyazawa, A., Fujiyoshi, Y. & Unwin, N. Structure and gating mechanism of the acetylcholine receptor pore. *Nature* **424**, 949-955 (2003).

[15] Gonen, T. *et al.* Lipid-protein interactions in double-layered twodimensional AQP0 crystals. *Nature* **438**, 633-638 (2005).

[16] Gonen, T., Sliz, P., Kistler, J., Cheng, Y. & Walz, T. Aquaporin-0 membrane junctions reveal the structure of a closed water pore. *Nature* **429**, 193-197 (2004).

[17] Hiroaki, Y. *et al.* Implications of the aquaporin-4 structure on array formation and cell adhesion. *J. Mol. Biol.* **355**, 628-639 (2006).

[18] Holm, P. J. *et al.* Structural basis for detoxification and oxidative stress protection in membranes. *J. Mol. Biol.* **360**, 934-945 (2006).

[19] Nogales, E., Wolf, S. G. & Downing, K. H. Structure of the alpha beta tubulin dimer by electron crystallography. *Nature* **391**, 199-203 (1998).

[20] Otwinowski, Z. & Minor, W. Processing of X-ray diffraction data collected in oscillation mode. *Methods in enzymology*, 307-325 (1997).

[21] Otwinowski, Z. M. & Minor, W. DENZO and SCALEPACK. *In: Rossmann, M.G., Arnold, E* **pp**, 226-235 (2001).

[22] Rossmann, M. G. & van Beek, C. G. Data processing. *Acta Crystallogr D Biol Crystallogr* **55**, 1631-1640, doi:BA0021 [pii] (1999).

[23] Leslie, A. G. W. Recent changes to the MOSFLM package for processing film and image plate data. *Joint CCP4 + ESF-EAMCB News-letter Prot. Cryst.* **26**, 22-33 (1992).

[24] Pflugrath, J. W. The finer things in X-ray diffraction data collection. *Acta Cryst. D Biol. Crystallogr.* **55**, 1718-1725 (1999).

[25] Steller, I., Bolotovsky, R. & Rossmann, M. G. An algorithm for automatic indexing of oscillation images using Fourier analysis. *J. Appl. Cryst.* **30**, 1036-1040 (1997).

[26] Higashi, T. Auto-indexing of oscillation images. *J. Appl. Cryst.* **23**, 253-257 (1990).

[27] Kabsch, W. Automatic processing of rotation diffraction patterns. *J. Appl. Cryst.* **21**, 67-71 (1988).

[28] Kim, S. Auto-indexing oscillation photographs. *J. Appl. Cryst.* **22**, 53-60 (1989).

[29] Leslie, A. G. The integration of macromolecular diffraction data. *Acta Cryst. D Biol. Crystallogr.* **62**, 48-57 (2006).

[30] Kabsch, W. Automatic processing of rotation diffraction data from crystals of initially unknown symmetry and cell constants. *J. Appl. Cryst.* **26**, 795-800 (1993).

[31] Smith, J. M. Ximdisp-A visualization tool to aid structure determination from electron microscope images. *J. Struct. Biol.* **125**, 223-228 (1999).

[32] Amos, L. A., Henderson, R. & Unwin, P. N. Three-dimensional structure determination by electron microscopy of two-dimensional crystals. *Prog Biophys Mol Biol* **39**, 183-231 (1982).

[33] Shaw, P. J. & Hills, G. J. Tilted specimen in the electron microscope: A simple specimen holder and the calculation of tilt angles for crystalline specimens. *Micron* **12**, 279-282 (1981).

[34] Valpuesta, J. M., Carrascosa, J. L. & Henderson, R. Analysis of electron microscope images and electron diffraction patterns of thin crystals of phi 29 connectors in ice. *J. Mol. Biol.* **240**, 281-287 (1994).

[35] Khademi, S. *et al.* Mechanism of ammonia transport by Amt/MEP/Rh: structure of AmtB at 1.35 A. *Science* **305**, 1587-1594, doi:10.1126/science.1101952, 305/5690/1587 [pii] (2004).

[36] Gonen, T. *et al.* Lipid-protein interactions in double-layered two-dimensional AQP0 crystals. *Nature* **438**, 633-638, doi:nature04321 [pii], 10.1038/nature04321 (2005).

[37] Harries, W. E., Akhavan, D., Miercke, L. J., Khademi, S. & Stroud, R. M. The channel architecture of aquaporin 0 at a 2.2-A resolution. *Proc Natl Acad Sci U S A* **101**, 14045-14050, doi:10.1073/pnas.0405274101, 0405274101 [pii] (2004).

[38] Roosild, T. P. *et al.* NMR structure of Mistic, a membrane-integrating protein for membrane protein expression. *Science* **307**, 1317-1321, doi:307/5713/1317 [pii], 10.1126/science.1106392 (2005).

[39] Wuthrich, K. The second decade--into the third millenium. *Nat Struct Biol* **5 Suppl**, 492-495, doi:10.1038/728 (1998).

[40] Gonen, T., Sliz, P., Kistler, J., Cheng, Y. & Walz, T. Aquaporin-0 membrane junctions reveal the structure of a closed water pore. *Nature* **429**, 193-197, doi:10.1038/nature02503, nature02503 [pii] (2004).

[41] Mitsuoka, K. *et al.* The structure of bacteriorhodopsin at 3.0 A resolution based on electron crystallography: implication of the charge distribution. *Journal of molecular biology* **286**, 861-882 (1999).

[42] Gyobu, N. *et al.* Improved specimen preparation for cryo-electron microscopy using a symmetric carbon sandwich technique. *J. Struct. Biol.* **146**, 325-333 (2004).

[43] Unwin, P. N. & Henderson, R. Molecular structure determination by electron microscopy of unstained crystalline specimens. *J Mol Biol* **94**, 425-440 (1975).

[44] Heymann, J. B. Bsoft: image and molecular processing in electron microscopy. *J. Struct. Biol.* **133**, 156-169 (2001).

[45] Schmid, M. F., Dargahi, R. & Tam, M. W. SPECTRA: a system for processing electron images of crystals. *Ultramicroscopy* **48**, 251-264 (1993).

[46] Hardt, S., Wang, B. & Schmid, M. F. A brief description of I.C.E.: the integrated crystallographic environment. *J. Struct. Biol.* **116**, 68-70 (1996).

[47] Philippsen, A., Schenk, A. D., Stahlberg, H. & Engel, A. Iplt-image processing library and toolkit for the electron microscopy community. *J. Struct. Biol.* **144**, 4-12 (2003).

[48] Frank, J. *et al.* SPIDER and WEB: processing and visualization of images in 3D electron microscopy and related fields. *J. Struct. Biol.* **116**, 190-199 (1996).

[49] Renault, L. *et al.* Milestones in electron crystallography. *J Comput Aided Mol Des* **20**, 519-527, doi:10.1007/s10822-006-9075-x (2006).

[50] Zeng, X., Gipson, B., Zhang, Z. Y., Renault, L. & Stahlberg, H. Automatic lattice determination for two-dimensional crystal images. *J. Struct. Biol.* **160**, 351-359 (2007).

[51] Grigorieff, N. Three-dimensional structure of bovine NADH:ubi˚ in ice. J. Mol. Biol. 277, quinone oxidoreductase (complex I) at 22. *A* **1033**, -1046 (1998).

[52] Grigorieff, N. Resolution measurement in structures derived from single particles. *Acta Cryst. D Biol. Crystallogr. 56 (Pt* **10**, 1270-1277 (2000).

[53] Crowther, R. A., Amos, L. A., Finch, J. T., De Rosier, D. J. & Klug, A. Three dimensional reconstructions of spherical viruses by fourier synthesis from electron micrographs. *Nature* **226**, 421-425 (1970).

[54] Grigorieff, N. Structure of the respiratory NADH:ubiquinone oxidoreductase (complex I). *Curr. Opin. Struct. Biol.* **9**, 476-483 (1999).

[55] Jap, B. K. *et al.* 2D crystallization: from art to science. *Ultramicroscopy* **46**, 45-84 (1992).

[56] Kuhlbrandt, W. Two-dimensional crystallization of membrane proteins. *Q Rev Biophys* **25**, 1-49 (1992).

[57] Levy, D., Chami, M. & Rigaud, J. L. Two-dimensional crystallization of membrane proteins: the lipid layer strategy. *FEBS Lett.* **504**, 187-193 (2001).

[58] Remigy, H. W. *et al.* Membrane protein reconstitution and crystallization by controlled dilution. *FEBS Lett.* **555**, 160-169 (2003).

[59] Signorell, G. A., Kaufmann, T. C., Kukulski, W., Engel, A. & Remigy, H. W. Controlled 2D crystallization of membrane proteins using methyl-beta-cyclodextrin. *J. Struct. Biol.* **157**, 321-328 (2007).

[60] Fujiyoshi, Y. *et al.* Development of a superfluid helium stage for high-resolution electron microscopy. *Ultramicroscopy* **38**, 241-251 (1991).

[61] Schmidt-krey, I. Electron crystallography of membrane proteins: two-dimensional crystallization and screening by electron microscopy. *Methods (San Diego, Calif*, 417-426 (2007).

[62] Aebi, U., Smith, P. R., Dubochet, J., Henry, C. & Kellenberger, E. A study of the structure of the T-layer of Bacillus brevis. *J. Supramol. Struct.* **1**, 498-522 (1973).

[63] Zeng, X., Gipson, B., Zheng, Z. Y., Renault, L. & Stahlberg, H. Automatic lattice determination for two-dimensional crystal images. *Journal of Structural Biology* **160**, 353-361, doi:10.1016/j.jsb.2007.08.008 (2007).

[64] Zeng, X., Stahlberg, H. & Grigorieff, N. A maximum-likelihood approach to two-dimensional crystals. *Journal of structural biology* **160**, 362–374 (2007).

[65] Statements, A. *et al.* AWK REFERENCE.

[66] Client, V. G. Least Squares, Fourier Analysis, and Related Approximation Norms. 1-38 (2004).

[67] Kunji, E. R., von Gronau, S., Oesterhelt, D. & Henderson, R. The three-dimensional structure of halorhodopsin to 5 A by electron crystallography: A new unbending procedure for two-dimensional crystals by using a global reference structure. *Proc Natl Acad Sci U S A* **97**, 4637-4642, doi:10.1073/pnas.080064697, 080064697 [pii] (2000).

[68] Lions, P. & Mercier, B. Splitting Algorithms for the Sum of Two Nonlinear Operators. *SIAM J. Numer. Anal.* **16**, 964-979 (1979).

69 Levi, A. & Stark, H. Image restoration by the method of generalized projections with application to restoration from magnitude. *Journal of the Optical Society of America A* **1**, 932 (1984).

70 Bauschke, H., Combettes, P. & Luke, D. On the structure of some phase retrieval algorithms. *IEEE International Conference on Image Processing,(Rochester, NY* (2002).

71 Fienup, J. Reconstruction of a complex-valued object from the modulus of its Fourier transform using a support constraint. *Journal of the Optical Society of America A* **4**, 118-123 (1987).

72 Spence, J. Diffractive (Lensless) Imaging. *Science of Microscopy* **2**, 1196-1227 (2007).

73 Spence, J., Weierstall, U., Fricke, T., Glaeser, R. & Downing, K. Three-dimensional diffractive imaging for crystalline monolayers with one-dimensional compact support. *Journal of Structural Biology* **144**, 209-218 (2003).

74 Hansen, P. C. The truncated SVD as a method for regularization. *BIT* **27**, 534-553 (1987).

75 Bauschke, F. R. P. O., H., C., P & Luke, D. Phase retrieval, error reduction algorithm, and Fienup variants: a view from convex optimization. Journal of the Optical Society of. *America A* **19**, 1334-1345 (2002).

76 Douglas Jr, J. & Rachford Jr, H. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society* (1956).

77 Marchesini, S. *et al.* X-ray image reconstruction from a diffraction pattern alone. *arXiv* **physics.optics**, doi:10.1103/PhysRevB.68.140101 (2003).

78 Jin, Y., Kim, Y.-H. & Rao, B. D. Support Recovery of Sparse Signals. *arXiv* **cs.IT** (2010).

79 Van Heel, M. Angular reconstitution: a posteriori assignment of projection directions for 3D reconstruction. *Ultramicroscopy* **21**, 111-123 (1987).

80 McCoy, A. J. *et al.* Phaser crystallographic software. *J Appl Crystallogr* **40**, 658-674, doi:10.1107/S0021889807021206 (2007).

81 Murshudov, G., Vagin, A. & Lebedev, A. Efficient anisotropic refinement of macromolecular structures using FFT. *Acta Crystallogr. D, Biol. Crystallogr.* (1999).

82 Collaborative Computational Project, N. The CCP4 suite: programs for protein crystallography. *Acta Crystallogr D Biol Crystallogr* **50**, 760-763, doi:10.1107/S0907444994003112 (1994).

83 Vaguine, A. A., Richelle, J. & Wodak, S. J. SFCHECK: a unified set of procedures for evaluating the quality of macromolecular structure-factor data and their agreement with the atomic model. *Acta Crystallogr D Biol Crystallogr* **55**, 191-205, doi:10.1107/S0907444998006684, S0907444998006684 [pii] (1999).

84 Pettersen, E. F. *et al.* UCSF Chimera--a visualization system for exploratory research and analysis. *J Comput Chem* **25**, 1605-1612, doi:10.1002/jcc.20084 (2004).

85 Agard, D. & Stroud, R. Linking regions between helices in bacteriorhodopsin revealed. *Biophysical Journal* **37**, 589-602 (1982).

86 Kimura, Y. *et al.* Surface of bacteriorhodopsin revealed by high-resolution electron crystallography. *Nature* **389**, 206-211, doi:10.1038/38323 (1997).

87 Essen, L., Siegert, R., Lehmann, W. D. & Oesterhelt, D. Lipid patches in membrane protein oligomers: crystal structure of the bacteriorhodopsin-lipid complex. *Proc Natl Acad Sci U S A* **95**, 11673-11678 (1998).

88 Spence, J. Phase recovery and lensless imaging by iterative methods in optical, X-ray and electron diffraction. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **360**, 875-895 (2002).

89 Raines, K. S. *et al.* Three-dimensional structure determination from a single view. *Nature* **463**, 214-217, doi:10.1038/nature08705 (2010).

90 Eaton, J. W. *GNU Octave Manual*. (Network Theory Limited, 2002).

91 Gaffney, K. J. & Chapman, H. N. Imaging Atomic Structure and Dynamics with Ultrafast X-ray Scattering. *Science* **316**, 1444-1448, doi:10.1126/science.1135923 (2007).

92 Kim, J. *et al.* Imaging of transient structures using nanosecond in situ TEM. *Science* **321**, 1472 (2008).

93 Armstrong, M. *et al.* Practical considerations for high spatial and temporal resolution dynamic transmission electron microscopy. *Ultramicroscopy* **107**, 356-367 (2007).

94 Claessens, B. *et al.* Ultracold Electron Source. *Phys Rev Lett* **95**, 164801 (2005).

95 van Oudheusden, T. *et al.* Electron source concept for single-shot sub-100 fs electron diffraction in the 100 keV range. *Journal of Applied Physics* **102**, 093501 (2007).

96 Fienup, J. Phase retrieval algorithms: a comparison. *Appl Opt* **21**, 2758-2769 (1982).

97 Fienup, J. & Wackerman, C. Phase-retrieval stagnation problems and solutions. Optical Society of. *America, Journal, A: Optics and Image Science* **3**, 1897-1907 (1986).

98 Chen, C., Miao, J., Wang, C. & Lee, T. Application of optimization technique to noncrystalline x-ray diffraction microscopy: Guided hybrid input-output method. *Physical Review B* **76**, 64113 (2007).

99 Marchesini, S. Invited Article: A unified evaluation of iterative projection algorithms for phase retrieval. *Rev Sci In* **strum78**, 011301 (2007).

100 Miao, J., Ishikawa, T., Shen, Q. & Earnest, T. Extending X-Ray Crystallography to Allow the Imaging of. *Noncrystalline Materials, Cells, and Single Protein Complexes. Annu Rev Phys Chem* **59**, 387-410 (2008).

101 Bauschke, H. H., Combettes, P. L. & Luke, D. R. Phase retrieval, error reduction algorithm, and Fienup variants: A view from convex optimization. (2002).

102 Bauschke, H., Combettes, P. & Luke, D. Hybrid projection-reflection method for phase retrieval. Journal of the Optical Society of. *America A* **20**, 1025-1034 (2003).

103 Moscato, P. & Cotta, C. A Gentle Introduction to Memetic Algorithms. *In Handbook of Metaheuristics* **pp**, 105-144 (2003).

104 Krasnogor, N. & Smith, J. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *Evolutionary Computation, IEEE Transactions* **on9**, 474-488 (2005).

105 Kennedy, J. & Eberhart, R. Particle swarm optimization. *Neural Networks* (1995).

106 Shi, Y. & Eberhart, R. Empirical study of particle swarm optimization. Proceedings of the. *1999 Congress on Evolutionary Computation* **3**, 1948-1950 (1999).

107 Chen, D., Jakana, J., Liu, X., Schmid, M. & Chiu, W. Achievable resolution from images of biological specimens acquired from a 4k× 4k CCD camera in a 300-kV electron cryomicroscope. *Journal of Structural Biology* (2008).

108 Williams, G., Quiney, H., Peele, A. & Nugent, K. Coherent diffractive imaging and partial coherence. *Phys Rev B* **75**, 104102 (2007).

109 Vartanyants, F. R. P. O., I & Robinson, I. Partial coherence effects on the imaging of small crystals using coherent X-ray diffraction. *J Phys: Condens Matter* **13**, 10593-10611 (2001).

110 Wolpert, D., Macready, W., Center, I., Jose, S. & C. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions* **on1**, 67-82 (1997).

111 Ho, Y. & Pepyne, D. Simple Explanation of the No-Free-Lunch Theorem and Its Implications. *Journal of Optimization Theory and Applications* **115**, 549-570 (2002).

112 Gillespie, D. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics* **22**, 403-434 (1976).

113 Nicolau, D. V., Jr., Hancock, J. F. & Burrage, K. Sources of anomalous diffusion on cell membranes: a Monte Carlo study. *Biophys J* **92**, 1975-1987, doi:S0006-3495(07)71004-4 [pii], 10.1529/biophysj.105.076869 (2007).

114 Raychaudhuri, S., Shapir, Y. & Mukamel, S. Disorder and funneling effects on exciton migration in treelike dendrimers. *Phys Rev E Stat Nonlin Soft Matter Phys* **65**, 021803 (2002).

115 Potter, C. S. *et al.* Leginon: a system for fully automated acquisition of 1000 electron micrographs a day. *Ultramicroscopy* **77**, 153-161 (1999).

Bryant Gipson - Curriculum Vitae June 17, 2010

Address:
WRO-1058 Mattenstrasse 26
University Basel, CH-4058 Basel, Switzerland

Personal Details:
Data & Place of Birth: December 28th, 1979, San Jose, California, USA
Marital Status: Married
Children: None
Nationality: USA

Current Position:
PhD Research Scholar, C-CINA, University of Basel, Switzerland

Degree Qualifications:
PhD, 07 June 2010 (Graduation Date), University of Basel
BA/BA Math/Physics, May 2005, Humboldt State University

Publications:

- **B. Gipson**, D. Masiel, N. D. Browning, J. Spence, K. Mitsuoka and H. Stahlberg, *Automatic Recovery of Missing Amplitudes and Phases in Tilt-Limited Electron Crystallography of 2D Crystals*, Submitted, Nature Methods, (April, 2010)
- D. Masiel, **B. Gipson**, D. G. Morgan, T. Guo, J. Spence, N. D. Browning, *Metaheuristic Algorithms for the Phase Problem in Ultrafast Diffractive Imaging,* (In Preparation)
- J. P. Buban, Q. M. Ramasse, **B. Gipson**, N. Browning, H. Stahlberg, *High-resolution low-dose scanning transmission electron microscopy*, Journal of Electron Microscopy, (DOI:10.1093/jmicro/dfp052), Nov, 2009, Epub ahead of print
- **B. Gipson**, X-Y Zeng, Z-Y Zhang, H. Stahlberg. *2dx - Automated 3D structure reconstruction from 2D crystal data*. Microscopy and Microanalysis (2008), 14:1290 (DOI:10.1017/S1431927608081919), Aug. 2008
- D. Masiel, **B. Gipson**, D. G. Morgan, J. Spence, N. D. Browning, *Particle Swarm Optimization of Iterative Phase Retrieval Algorithms for Ultrafast Coherent Diffractive Imaging*, Microscopy and Microanalysis (2008), 14:504-505 (DOI:10.1017/S1431927608085607), Aug. 2008
- **B. Gipson**, X-Y Zeng, H. Stahlberg, *2dx_merge: Data management and merging for 2D crystal images*, Journal of Structural Biology, Volume 160, Issue 3, Electron Crystallography of Membrane Proteins, Pages 375-384, ISSN, December 2007
- X-Y Zeng, **B. Gipson**, Z-Y Zheng, L. Renault, H. Stahlberg, *Automatic lattice determination for two-dimensional crystal images*, Journal of Structural Biology, Volume 160, Issue 3, Electron Crystallography of Membrane Proteins, Pages 353-361, ISSN 1047-8477, (DOI: 10.1016/j.jsb.2007.08.008), December 2007
- **B. Gipson**, X-Y Zeng, H. Stahlberg, R. Wouts, *2dx – User Friendly Image Processing (and Merging) for 2D Crystals.* Microscopy and Microanalysis, (13 (Suppl. 03), pp 160-161 DOI:10.1017/S1431927607080804), September 2007
- PL Chiu, M. D. Pagel, J. Evans, HT Chou, X-Y Zeng, **B. Gipson**, H. Stahlberg, Crina M. Nimigean, *The Structure of the Prokaryotic Cyclic Nucleotide Modulated Potassium Channel MloK1 at 16 A Resolution*, Structure, Volume 15, Issue 9, Pages 1053-1064, ISSN 0969-2126, (DOI:10.1016/j.str.2007.06.020), September 2007
- **B. Gipson**, X-Y Zeng, Z-Y Zhang, H. Stahlberg, *2dx--User-friendly image processing for 2D crystals*, Journal of Structural Biology, Volume 157, Issue 1, Software tools for macromolecular microscopy, Pages 64-72, ISSN 1047-8477, (DOI:10.1016/j.jsb.2006.07.020), January 2007
- L. Renault, HT Chou, PL Chiu, R.M. Hill, X-Y Zeng, **B. Gipson**, Z-Y Zhang, A. Cheng, V. Unger, H. Stahlberg, *Milestones in electron crystallography*, J Comput Aided Mol Des. 2006 Jul-Aug;20(7-8):519-27. Epub, Nov 2006

Awards:
Distinguished Scholar Award (DSA) from the Microbeam Analysis Society (MAS), 2008