Serial Femtosecond Crystallography Data Analysis of Photosystem II

by

Natasha Stander

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved September 2019 by the
Graduate Supervisory Committee:

Petra Fromme, Co-Chair
Nadia Zatsepin, Co-Chair
Richard Kirian
Wei Liu

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

Serial femtosecond crystallography (SFX) uses diffraction patterns from crystals delivered in a serial fashion to an X-Ray Free Electron Laser (XFEL) for structure determination. Typically, each diffraction pattern is a snapshot from a different crystal. SFX limits the effect of radiation damage and enables the use of nano/micro crystals for structure determination. However, analysis of SFX data is challenging since each snapshot is processed individually.

Many photosystem II (PSII) dataset have been collected at XFELs, several of which are time-resolved (containing both dark and laser illuminated frames). Comparison of light and dark datasets requires understanding systematic errors that can be introduced during data analysis. This dissertation describes data analysis of PSII datasets with a focus on the effect of parameters on later results. The influence of the subset of data used in the analysis is also examined and several criteria are screened for their utility in creating better subsets of data. Subsets are compared with Bragg data analysis and continuous diffuse scattering data analysis.

A new tool, *DatView* aids in the creation of subsets and visualization of statistics. *DatView* was developed to improve the loading speed to visualize statistics of large SFX datasets and simplify the creation of subsets based on the statistics. It combines the functionality of several existing visualization tools into a single interface, improving the exploratory power of the tool. In addition, it has comparison features that allow a pattern-by-pattern analysis of the effect of processing parameters. *DatView* improves the efficiency of SFX data analysis by reducing loading time and providing novel visualization tools.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

The goal of structural biology is the determination of biomolecular structures in order to understand their function. This leads to diverse applications such as designing drug targets to change the function of medically relevant proteins or reproducing a function in other systems like solar energy conversion.

Crystallography is one of several techniques for obtaining structural information at atomic or near atomic resolutions. In protein crystallography, a protein crystal (three-dimensional repetitive arrangement of the protein) is exposed to an X-ray beam, and the resulting diffraction pattern is recorded on a detector. Multiple diffraction patterns are collected from different orientations in order to reconstruct a three-dimensional model of the protein.

A complete dataset (containing information to the desired resolution from all orientations) can require information from multiple crystals because the X-ray beam induces dose-dependent radiation damage to the crystal. The term serial crystallography can refer to a dataset combining results from multiple crystals, where serial refers to sequential data collection. At the extreme limit of serial crystallography, a single diffraction pattern is collected from each crystal. A good term for this is snapshot crystallography.

Snapshot crystallography falls into two general categories depending on the source of the X-ray beam. Serial femtosecond crystallography (SFX) is done at X-ray free-electron lasers (XFELs) with femtosecond referring to the pulse duration of the beam. Serial millisecond crystallography uses techniques developed for SFX at synchrotrons

where the pulse duration is in the millisecond range. Most snapshot crystallography datasets are SFX datasets, and so the term SFX is often used even in cases were the more general snapshot crystallography terminology is applicable.

SFX has two major advantages compared to traditional crystallography. First, the femtosecond pulse duration is faster than secondary radiation damage processes, which reduces the effects of radiation damage on the data. The diffraction data is collected before the destruction of the crystal. This has been termed diffraction before destruction (Neutze et al. 2000). Second, the intensity and spot size of the XFEL beam and using each crystal only once allows nano/micro crystals to be used. Smaller crystals can be easier to grow than the larger crystals for conventional crystallography where crystals are rotated in an X-ray beam. Smaller crystals also allow time resolved crystallography at shorter time scales because it takes less times for substrates to diffuse throughout the crystal.

However, SFX has disadvantages in data analysis. Additional steps are required to determine which frames contain diffraction patterns and to analyze the snapshots individually. Also, there are additional sources of noise from crystal variations and the XFEL beam. To compensate for the noise, SFX datasets can be very large with hundreds of thousands of patterns. Large datasets have computational challenges in data storage and processing times.

The focus of this thesis is on snapshot crystallography data analysis. Specifically, the primary aims are 1) to determine the effect of processing parameters, including the subset of data used for the analysis, on statistics and the protein model and 2) to improve processing speeds for snapshot crystallography data analysis.

An SFX experiment begins with sample preparation. The datasets in this thesis come from photosystems I and II, and details of the samples and datasets are described

in section 1.1. Diffraction patterns are collected as an X-ray beam interacts with each crystal. Diffraction is reviewed in section 1.2. SFX experiments are performed at XFELs, with datasets in this thesis primarily collected at LCLS. Some datasets were also collected at EXFEL. XFELs are described in section 1.3. SFX differs from traditional crystallography because each crystal must delivered to the XFEL beam in time for the next X-ray pulse. Two main methods of sample delivery are used for datasets in this thesis: jets and fixed-target chips. Sample delivery for both methods is described in section 1.4. Diffraction patterns are collected on detectors. Detectors and detector corrections are described in section 1.5.

After the diffraction patterns are recorded, data analysis begins. An overview of SFX analysis is given in section 1.6 and the steps of data analysis are detailed in the remaining chapters of the dissertation. As a brief overview, the first step in data analysis is a data reduction step. This step takes all the recorded frames (referred to as events) and determines which frames contain diffraction patterns. Frames with potential diffraction patterns are referred to as hits. In this thesis, the tool *Cheetah* (Barty et al. 2014) is used for data reduction, so files containing hits follow *Cheetah* conventions. The second step in data analysis is indexing of the hits to determine the crystal lattice from each image. In this thesis, *CrystFEL* (White et al. 2012, 2016) is used for indexing and merging the results of indexing into a single dataset. SFX data analysis then converges with traditional crystallography data analysis, and the dataset must be phased and refined to generate an electron density map. For a more comprehensive review of crystallography, see (Rupp 2009). The final section in the introduction (section 1.7) gives the motivation for determining the affect of processing parameters, particularly subset selection, in SFX analysis and improving processing speeds

1.1 Samples

At the start of any crystallography experiment is the need for growing a crystal of the target molecule. Most of the datasets used in this work are for photosystem II (PSII), but there are also a few datasets from photosystem I (PSI) and PSI bound to ferredoxin. Both photosystems are large membrane complexes involved in photosynthesis. Photosynthesis, and particularly the production of molecular oxygen, is of interest for solar energy development. A schematic of the proteins involved in photosynthesis is shown in fig. 1 and a good review of photosynthesis is given in (Blankenship 2014). As a quick summary, PSII catalyzes electron transfer from water to plastoquinone, producing molecular oxygen (O_2) and plastoquinol (PQH_2). Plastoquinol docks at the cytochrome b_6f complex where the electrons are transferred to plastocyanin or cytochrome c_6. PSI then transfers the electrons to ferredoxin. PSII is described in section 1.1.1 and PSI is described briefly in section 1.1.2.

1.1.1 Photosystem II

Photosystem II (PSII) is composed of 20 protein subunits with a total molecular weight of 350 kDa per monomer. PSII is a protein-cofactor complex that contains cofactors such as lipids, carotenoids, and chlorophylls where exact cofactor composition depends on the source of PSII. Of particular interest is the Mn_4CaO_5 cluster referred to as the oxygen evolving complex (OEC) which catalyzes the water splitting reaction. The ultimate goal with PSII is a time resolved movie showing the water splitting reaction at the OEC. This requires high resolution structures of the intermediate states.

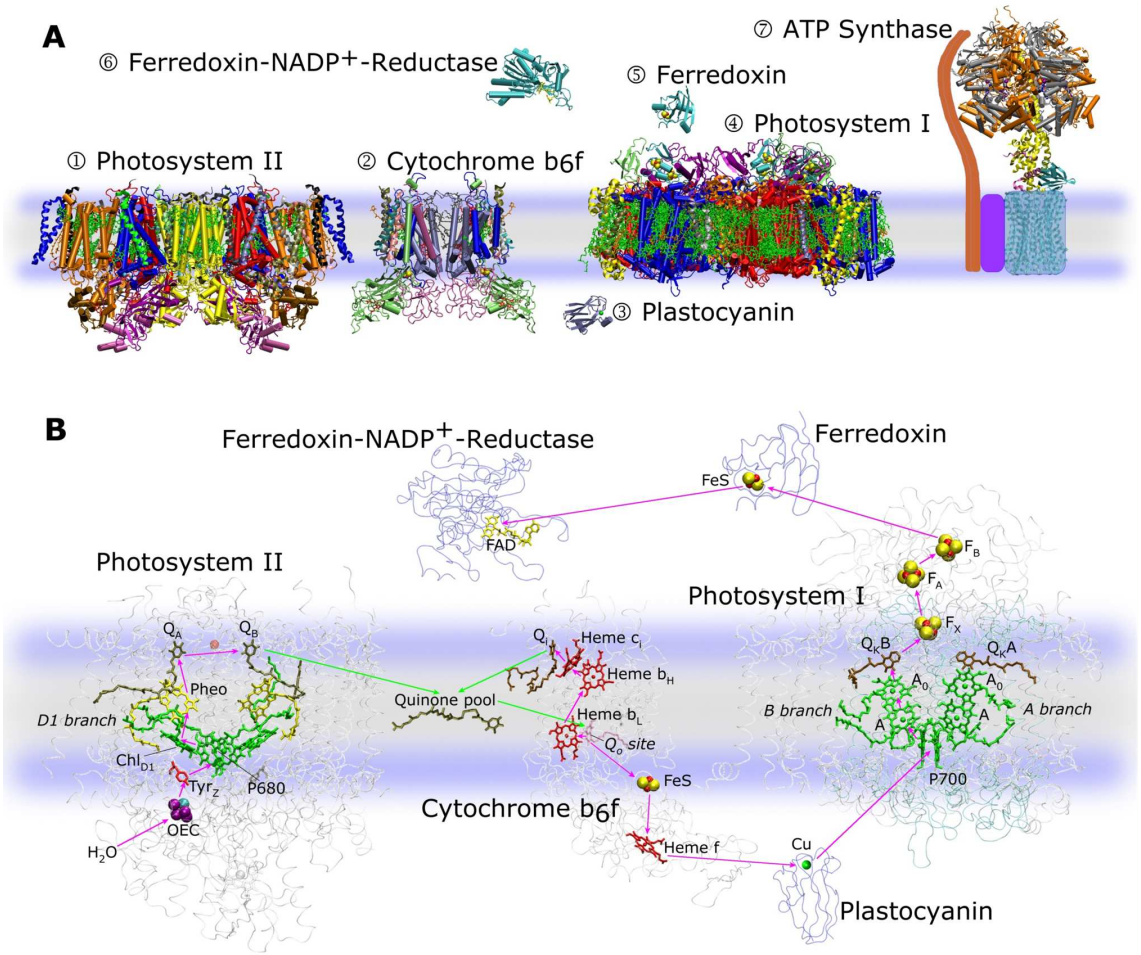

Figure 1. Overview of photosynthesis

Photosystems I and II are reaction centers that convert light energy into chemical energy. Photosystem II catalyzes the reaction $4hv + 2H_2O \rightarrow O_2 + 4H^+ + 4e^-$, increasing the protons in the lumen of the thylakoid membrane (lower part of the figure). The electrons are used to reduce plastoquinone to platoquinol. Plastoquinol transfers the electrons to the cytochrome b_6f complex, which transfers electrons to plastocyanin or cytochrome c_6 while transferring two protons from the stroma to the lumen. The electrons are transferred by photosystem I to ferredoxin. Ferredoxin-NADP$^+$-reductase uses them to reduce NADP$^+$ to NADPH, thus removing protons from the stroma. The reactions form a proton gradient that is used to drive ATP synthase. Figure from (Fromme and Grotjohann 2008).

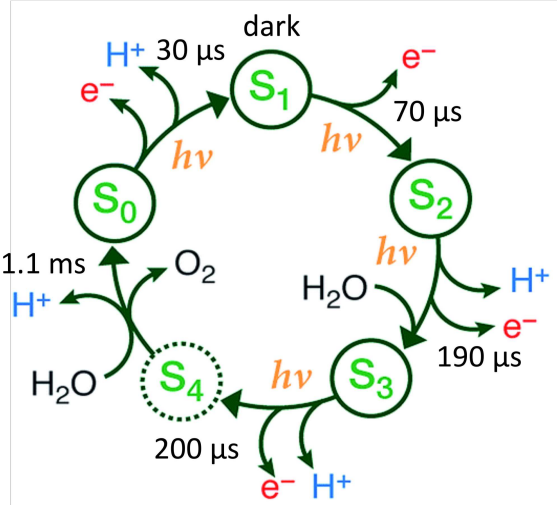

Figure 2. Photosystem II S states

The S states of Photosystem II. Over the course of the cycle, four light flashes catalyze the removal of four electrons and four protons, as well as the generation of O_2. The S1 state is referred to as the dark adapted state. Figure adapted from (Krewald et al. 2015) with time constants added from (Dau and Haumann 2008).

PSII uses light to drive the electron transport across the membrane by a chain of electron carriers: $P680 \rightarrow Pheo \rightarrow Q_A \rightarrow Q_B$. $P680^+$ has the high redox potential of 1.1 V enabling it to extract electrons and protons from the OEC via a Tyrosine (Tyo_z). (See fig. 1b for the cofactors of the electron transport chain). PSII cycles through five different redox states referred to as S states. The cycle consists of four light pulses and the extraction of four electrons and four protons, shown in fig. 2. The four electrons ultimately reduce two molecules of plastoquinone at the acceptor side of PSII.

The S1 state is the dark adapted state, and several structures have been published for it. The earliest synchrotron structure was defined at 3.8 Å (Zouni et al. 2001) and the highest resolution structure at 1.9 Å (Umena et al. 2011). A structure using shorter pulse durations to reduce the effects of radiation damage was solved at 1.95 Å (Suga et al. 2015).

Several time resolved studies have been published for the S3 state. The first time resolved study solved the S1 state to 5 Å and the S3 state to 5.5 Å (Kupitz et al. 2014). However, the resolution of this structure is too low to resolve detailed changes at the OEC. Another study on the S3 state solved the structure at 2.25 Å (Young et al. 2016) and found no evidence of an additional water molecule binding Mn1 in the S3 state. In contrast, (Suga et al. 2017) did find evidence for an additional oxygen near the Mn1 in their S3 state structure at 2.35 Å. Finally, (Kern et al. 2018) found evidence of an oxygen in the S3 state, but at a longer distance than (Suga et al. 2017) in their S3 maps (at 2.5 Å for 150 μs delay from the laser flash to the X-ray, 2.2 Å for 400 μs delay, and 2.07 Å for the combination).

1.1.2 Photosystem I

Photosystem I (PSI) in cyanobacteria is a trimer of heterodimers totaling 1056 kDa, with 12 protein subunits and 127 cofactors, solved to 2.5 Å (Jordan et al. 2001). It catalyzes the transfer of electrons from plastocyanin or cytochrome c_6 to ferredoxin through the chain of electron carriers: $P700 \rightarrow Q_kB \rightarrow F_x \rightarrow F_A \rightarrow F_B$. It was used for the first SFX experiment at LCLS (Chapman et al. 2011). That dataset has been used to develop detwinning algorithms for serial crystallography (Brehm and Diederichs 2014; Liu and Spence 2014). Electron transfer in PSI can be driven by laser illumination. It is also sometimes used as a calibration sample. The only PSI dataset used in this thesis is the EXFEL August 2018 (see chapter 4).

Table 1. PSII frames by dataset and laser

Dataset	Delivery	Laser Delays (μs)	Frames	Indexed Frames	Crystals
2012 Jan	Jet	210 ; 570	64674	46824	50751
		780	2	2	2
		? 210 ; 570	26932	17949	18815
		?780	13030	7475	7976
		?Off	70168	48954	51229
		Off	118904	44155	47074
2012 Jun	Jet	210 ; 570 ; 250	31234	24257	26334
		? 210 ; 570 ; 250	433	359	378
		? Off	1758	1539	1605
		Off	35073	27631	29771
2014 Nov	Jet	500	29763	15593	16438
		Off	30035	15901	16738
2015 Oct	Jet	700 ; 1200	24250	14804	15751
		700 ; 1200 ; 2000	5123	4730	4864
		700 ; 1200; 600	13882	12973	13318
		? 700 ; 1200 ; 600	3303	3110	3229
		Off	50292	39094	40762
2016 Aug	Jet	700 ; 1300 ; 600	27283	25544	26176
		? 700 ; 1300 ; 600	48	46	46
		Off	42746	37924	38956
2016 Nov	Fixed-target	Off	728029	493451	593709
2017 Sep	Fixed-target	530 ; 1500 ; 600	575854	319247	406425
		Off	278490	152467	202008
2018 Mar	Fixed-target	500 ; 1000 ; 600	122246	51783	57920
		Off	78315	25468	26682
2018 Dec	Jet	500 ; 1200 ; 600	48699	16220	17268
		Off	48950	16453	17470

Frames, indexed frames, and crystals (higher than indexed frames in cases of multi-crystal frames) by LCLS dataset and laser scheme. Indexing values from *CrystFEL* 0.7.0 all PSII indexing (see A.1).

1.1.3 Photosystem II Datasets

A complete summary of experiments and PSII datasets is given in the appendix A. PSII datasets have been collected at both LCLS and the European XFEL (EXFEL) (see section 1.3), but the majority of the data was collected at LCLS and only those datasets are referenced in this thesis. Datasets are referenced by location of data collection, month, and year. PSII data for the S1 state (before laser illumination) is referred to as dark, with laser off. PSII data with laser illumination is referred to as light. Most PSII data is dark, from all-dark datasets and from alternating light-dark data collection. Light data depends on the laser time delays with 1 flash targeting the S2 state, two flashes for the S3 state, and three flashes for the transient S4 states. However, different experiments used different laser time delays so not all data for a particular S state is necessarily combinable. A short summary of LCLS PSII datasets sorted by date and laser scheme is given in Table 1.

Identifying which frames were truly light and dark can be complicated. In Table 1, a ? mark indicates suspected problems with laser illumination. This can arise from problems with the laser, uncertainty in laser timing, or jet (see section 1.4.1) speeds that are too fast or too slow. The LCLS September 2017 beam time also has uncertain laser illumination, but in that case the problem is related to fogging and is time dependent. Since the exact time when the laser illumination ceased to be effective is not known, the light data has not been split into groups like the other experiments.

For alternating light-dark patterns collected at LCLS, laser illumination is related to an event code, usually evr41. It's intuitive to assume that when evr41 is 0 (class 1 in *Cheetah* binned output), the laser was off, and when evr41 is 1 (class 3 in *Cheetah*

binned output), the laser was on. However, while the laser is triggered at evr41 = 1, that means the laser applied to the evr41=0 pulse. Identifying which frames are light and dark is important because dark-only runs and the alternating dark frames are usually combined regardless of the laser scheme of the corresponding light frames.

Three of the above datasets have been used for publications. LCLS January 2012 and LCLS June 2012 were used in an initial proof of principle study for time resolved data collection (Kupitz et al. 2014). The S1 and S3 states at 5 Å and 5.5 Å were from LCLS January 2012, and supplementary unit cell analysis included LCLS June 2012. Work on resolution extension with continuous diffuse scattering used LCLS November 2014 (Ayyer et al. 2016; Chapman et al. 2017).

1.1.4 Crystal Variation

In snapshot crystallography, it is important to consider crystal variations because each shot comes from a different crystal. Crystallization conditions for the PSII datasets referred to throughout this thesis follow those published in (Kupitz et al. 2014) with the exception that for time resolved datasets for the transient S4 state an additional quinone was added. However, while the same crystallization conditions are used, crystal quality can vary for different preparations and crystal size may differ from experiment to experiment to accommodate the delivery method.

The easiest crystal change to identify with data analysis is in the unit cell. The unit cell of a crystal defines the smallest translationally repetitive unit and may contain multiple copies of the protein. PSII crystallizes in the $P2_12_12_1$ space group and contains four dimers in each unit cell. Unit cells are defined by three vectors: a,b, and c, and three angles: α, β, and γ. PSII has an orthogonal unit cell, so alpha, beta,

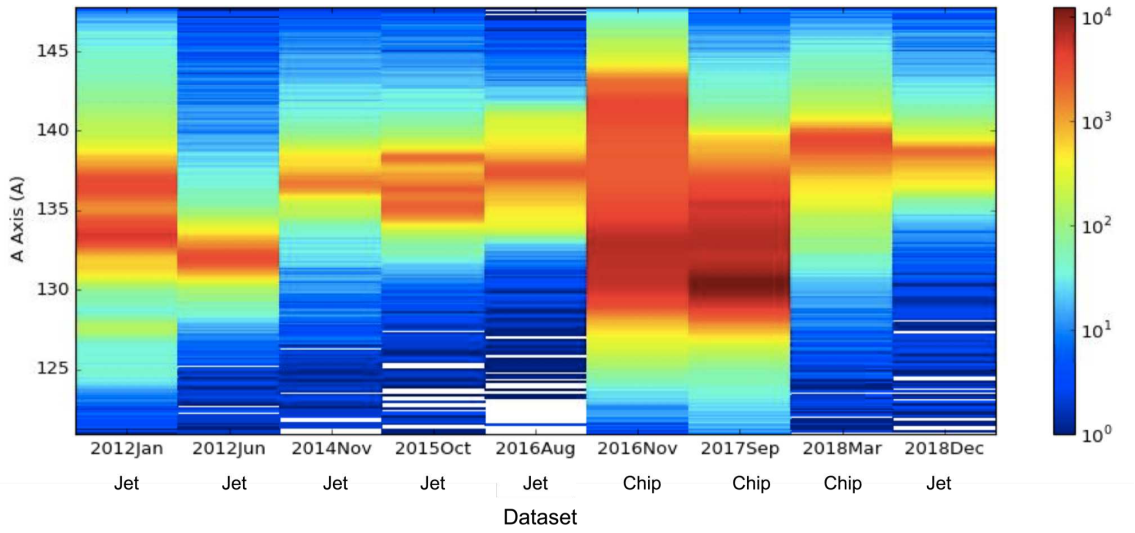

Figure 3. Photosystem II a axis variation

A two dimensional histogram showing the unit cell a axis on the y axis and grouped by LCLS dataset on the x-axis. LCLS November 2016 (2016Nov), LCLS September 2017 (2017Sep) and LCLS March 2018 (2018Mar) were fixed target chip datasets and all other datasets were collected with a jet. See section 1.4 for delivery methods and section 1.4.2 for details of the unit cell variations observed in fixed target chip datasets. Coloring is on a log scale. This data comes from the *CrystFEL* 0.7.0 all PSII indexing (see A.1).

and gamma are all 90°. The a, b and c axis are generally 133-136 Å, 228 Å, and 308 Å but a lot of variation is seen in the datasets used in this thesis (see fig. 3). A second crystallographic parameter is the diffraction resolution limit, described in more detail in chapter 3. A more detailed review of crystal parameters considered when merging is given in section 1.7.1.

Aside from crystal to crystal variations, crystallization itself can impact the determined structure. Crystal contacts can favor non-native conformations of a protein. Of relevance to time resolved PSII datasets, crystallization can also impact the reactivity of proteins. Crystal contacts may limit protein movements, preventing

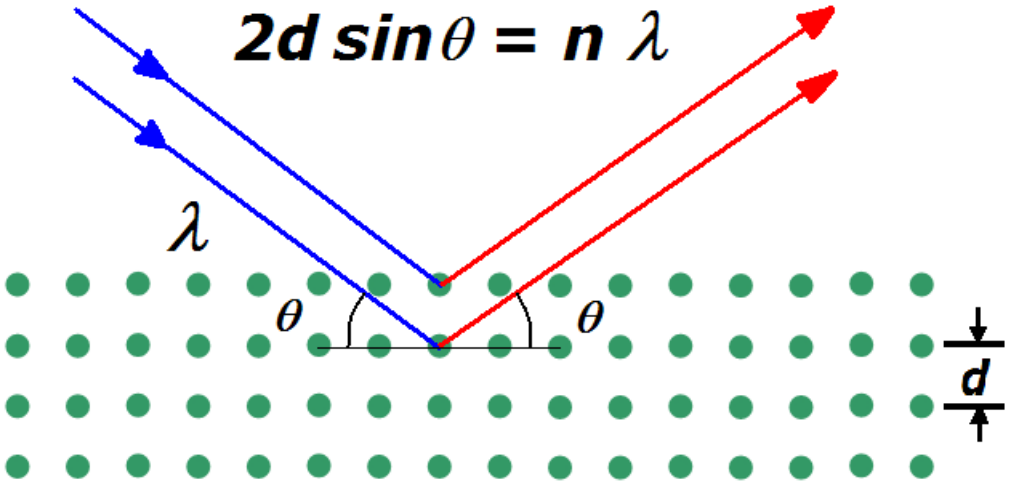

Figure 4. Bragg's law

Bragg's law defines when constructive interference occurs as $2d\sin(\theta) = n\lambda$. The extra distance the lower wave must travel is twice the hypotenuse of the right triangle defined with angle θ and vertical distance d, equal to $2d\sin(\theta)$. In order for constructive interference between waves, that distance must be an integer multiple of the wavelength λ, where n is an integer. Image from (Aboalbiss 2009).

the reaction. Or, the crystal packing may limit diffusion of substrate molecules. For PSII to reach the S4 state, a new quinone molecule must bind after two laser excitations. While these issues don't directly impact data analysis, it is important for interpreting the results. Also, the percentage of excited molecules in the crystal impacts the crystalline order. A sufficient number of molecules must be excited for the changes to be visible in the data.

1.2 Diffraction

A crystallographic diffraction pattern contains peaks at locations determined by Bragg's Law, shown in fig. 4. Since X-rays are waves, constructive and destructive

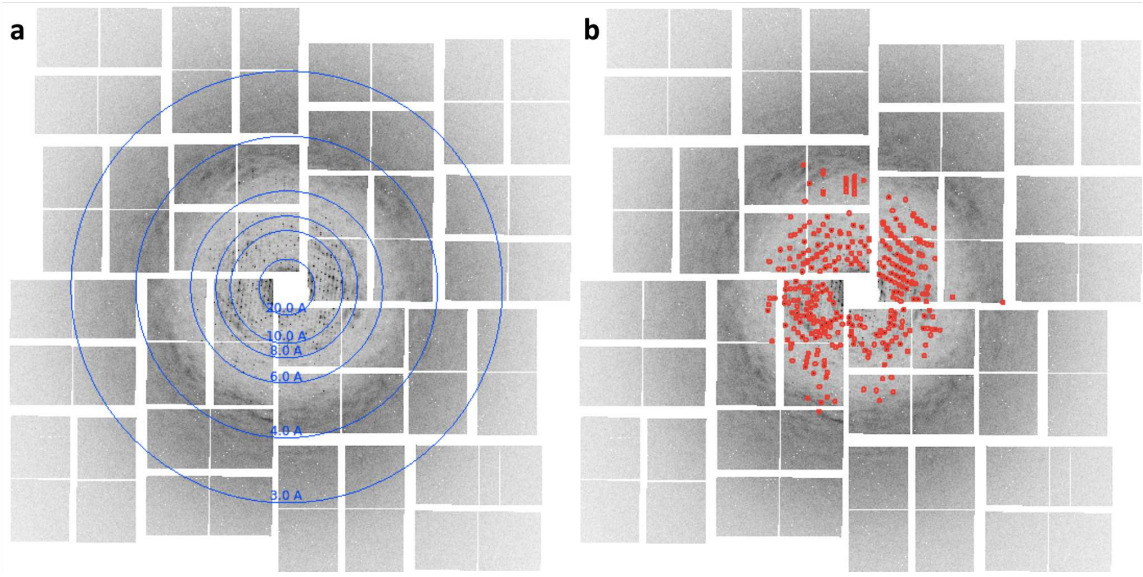

Figure 5. Diffraction pattern

An example diffraction pattern from LCLS October 2015. (a) The image with resolution rings. Bragg peaks extend past the 6 Å. Between the 6 Å and 4 Årings, diffuse scattering is visible. (b) The same image with found Bragg peaks (from *Cheetah* (Barty et al. 2014)) surrounded with red squares.

wave interference occurs as an X-ray beam interacts with a crystal. Bragg's law defines the optimum for constructive interference as $2d\sin(\theta) = n\lambda$ where d is the distance between planes in the lattice, θ is the angle of the wave, λ is the wavelength of the X-ray and n is an integer. Essentially, Bragg's law shows that constructive interference occurs when the extra distance travelled to reach another plane is an integer multiple of the wavelength.

The diffraction resolution limit of a crystal is the resolution of the Bragg peaks, and depends on the crystalline order in the crystal. As molecules become less aligned, Bragg's law is no longer met. Detection of Bragg peaks also depends on the crystal size, with large crystals with more molecules giving brighter peaks (Holton and Frankel 2010).

Low resolution Bragg peaks appear near the X-ray beam at low angles and define overall features. As resolution increases, the peaks appear further from the beam, at higher angles, and define more features from secondary structure elements such as alpha helices, to large side chains, and then to locations of individual atoms. Figure 5 shows a PSII diffraction pattern. The X-ray beam goes roughly through the center of the image. Resolution increases further away from the center of the image, and Bragg peaks appear as darker points, highlighted with red boxes in the second part of the figure.

Crystalline disorder such as molecular displacements are a disadvantage for analysis of Bragg peaks, but do result in diffuse scattering. Unfortunately, diffuse scattering is not limited to the molecule of interest, and will also appear for solvent within and around the crystal and air scattering. In fig. 5, the darker isotropic ring between 6 Å and 4 Å comes from solvent. However, anisotropic features come from correlated displacements in the protein, such as the speckles observed in the water ring in fig. 5. Many different models for diffuse scattering have been suggested and tested. Different models were compared in (Peck, Poitevin, and Lane 2018). A more general outlook on diffuse scattering is (Wall, Wolff, and Fraser 2018).

Of particular relevance to PSII, diffuse scattering was used to extend the resolution of the LCLS November 2014 dataset from 4.5 Å to 3.5 Å (Ayyer et al. 2016). This assumes that the diffuse scattering is attributed to rigid body translational movements of PSII, in which case the diffuse scattering is the molecular transform of the asymmetric unit (smallest repeating unit of the crystal) modulated by the lattice. Continuous diffuse scattering analysis of PSII is discussed in detail in chapter 7.

1.3 X-ray Free Electron Lasers

The first X-ray free electron laser (XFEL) was LCLS, located in the USA, opening in 2010 (Emma et al. 2010). SFX was established as a technique using a PSI dataset collected at LCLS (Chapman et al. 2011). Since then, four additional XFELs have opened for user operations. SACLA (Huang and Lindau 2012), located in Japan, opened for users in 2012 . PAL-XFEL (Ko et al. 2017), located in Korea and the European XFEL (EXFEL)(Tschentscher et al. 2017), located in Germany both opened for users in 2017. The SwissFEL (Milne et al. 2017) located in Switzerland opened for users in 2019. Another XFEL is under development in China.

Free electron lasers have two main advantages: (1) tunable wavelength by changing the electron energy and (2) reaching short wavelengths with ultra-relativistic electrons (Andruszkow et al. 2000; Ayvazyan et al. 2002). Initial free electron lasers were limited by optics and seed beams to ultraviolet wavelengths (Milton et al. 2001). LCLS was the first hard X-ray XFEL. XFELs overcome the optics limitation with self-amplified spontaneous emission (SASE). In SASE, electrons pass through a long length of undulators which causes them to emit an electromagnetic wave. The electromagnetic wave then interacts with the electrons, bunching them together into coherent microbunches. This increases the power and enables X-ray wavelengths. Bunch compressor settings control the pulse duration (Ayvazyan et al. 2002).

While SASE has advantages in power, coherence, and tunable wavelength, it is also noisy because it amplifies the most coherent set of electrons in the initial random beam. This means that parameters like the pulse duration and photon energy can vary shot-to-shot. At LCLS, the pulse duration and other properties can be calculated shot-to-shot with information from the XTCAV system (Krejcik et al.

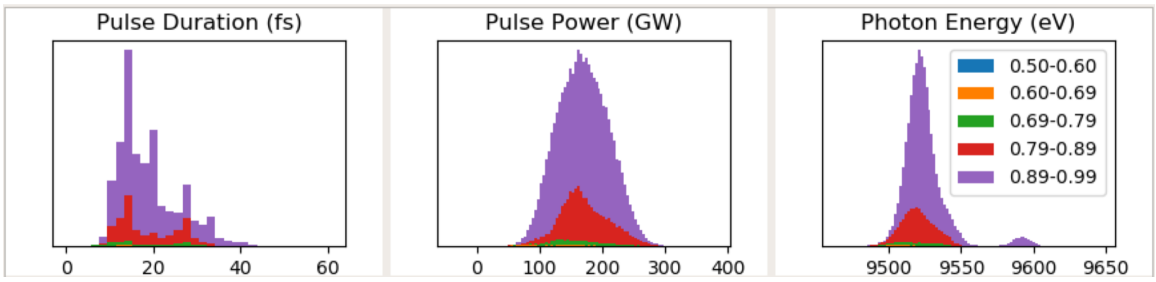

Figure 6. Pulse duration, power, and energy histograms

Parameters of the X-ray beam such as pulse duration, pulse power, and photon energy can vary shot to shot. These histograms were generated with 97,378 frames from LCLS December 2018 with frames targeted at 25 or 18 fs pulse durations. LCLS uses the XTCAV system to determine the pulse duration and pulse power, and these histograms are stacked by XTCAV agreement which is a measure of the certainty of the values by the comparing the agreement between two different algorithms.

2013). The software interface uses two different algorithms to compute statistics, and the agreement between the algorithms is used as a rough estimate of the error. Figure 6 shows the pulse duration and power calculated by the XTCAV and the photon energy stored in the *Cheetah* (Barty et al. 2014) output with coloring by the XTCAV agreement.

Shot-to-shot noise can also be correlated, adding systematic error. This is especially problematic for time resolved studies. For instance, collecting alternating light and dark patterns at 120 hertz at LCLS could be impacted by a systematic difference at 60 hertz in the X-ray beam (Turner et al. 2016). At the EXFEL, pulses are delivered in trains, with each train containing up to 2,700 pulses and 10 trains in a second. Time resolved data may be collected by exciting with a laser at some point during the pulse train, with diffraction patterns before the pulse in an unexcited state and patterns after at different time points for each pulse. Such analysis is difficult if there is a decrease in intensity over the length of the pulse train or other systematic effects. Systematic changes during a pulse train at the EXFEL are currently being studied,

but initial results by Dr. Anton Barty (unpublished) suggest that detector gain stages (see section 1.5.2) change over the duration of the pulse train.

1.4 Delivery

Serial data collection at XFELs is necessary because each crystal is destroyed on contact with the XFEL beam. Therefore, a new crystal must be delivered to the interaction region in time for the next pulse. This is particularly challenging with high repetition rates, like the 27,000 hertz anticipated at the EXFEL (Tschentscher et al. 2017) and 1 Megahertz rates anticipated for LCLS-II. In addition to replenishing the sample, an ideal delivery system does not change the sample, is easy to use, does not waste any sample or any X-ray pulses, and, for data analysis especially, adds no background. For PSII datasets in this thesis, the two methods of delivery are jets and chips. The following two sections give a brief description of the setup of and data analysis challenges associated with jets (section 1.4.1) and chips (section 1.4.2) respectively. The final section (1.4.3) gives a brief comparison of the two methods based on hit rates. For a more complete review of sample delivery, refer to (Grünbein and Nass Kovacs 2019).

1.4.1 Jets

A typical jet experimental setup for a time-resolved PSII experiment is shown in fig. 7. Crystals are suspended in their buffer and focused with a nozzle into a jet. At LCLS, the X-ray beam frequency is 120 hz and a pump laser is used at 60 hz to trigger the reaction. Most jet datasets were collected with a gas dynamic virtual

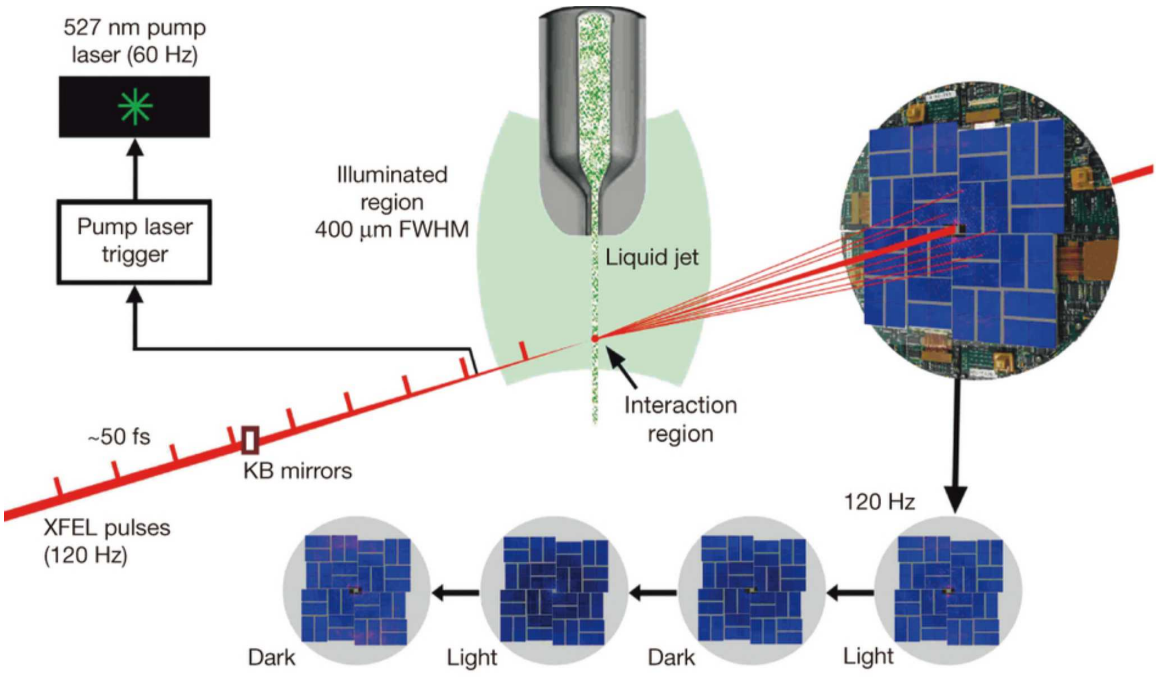

Figure 7. Time-resolved experimental setup with jet

A jet experimental setup used for time resolved studies on photosystem II. The pump laser provides time resolved data by illuminating every other pulse. Figure adapted from (Kupitz et al. 2014).

nozzle (GDVN) (DePonte et al. 2008), but LCLS August 2016 used a double-flow focused liquid injector (DFFN) (Oberthuer et al. 2017). The type of nozzle can impact the background. For example, the DFFN has a sheath of ethanol that adds diffuse scattering in addition to the crystal solvent.

The buffer of the crystal solution also changes the background. Various viscous media have been tried for sample delivery, including LCP (Weierstall et al. 2014), agarose (Conrad et al. 2015), and PEO (Martin-Garcia et al. 2017). LCLS November 2014 dataset used agarose with PSII in the first part of the dataset with around 500 hits. The crystal buffer primarily changes the resolution ring of the background, as shown in fig. 8 for the four proteins studied at APS August 2016. Subtracting

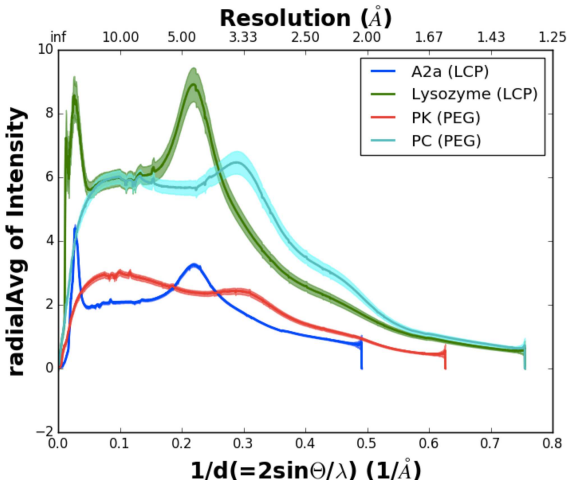

Figure 8. Radial average of intensity for LCP and PEO

An initial version of fig. 8 of (Martin-Garcia et al. 2017), without scaling, of the radial average intensity for four different proteins measured at APS August 2016. Different mediums have background rings at different locations with LCP around 5 Å and PEO (PEG) around 3.33 Å. Different samples were measured at different detector distances, so the lines end at different points.

this background is not trivial. Using corresponding empty frames for background subtraction assumes that the shape of the jet around the crystal is the same as the shape of the jet without a crystal which is generally not true. Usually, some form of radial background subtraction is used, but determining which value to subtract at each radii is still an active area of research, particularly for continuous diffuse scattering studies (Chapman et al. 2017).

Background can also come from other parts of the experimental setup. For example, data collected at the end of LCLS August 2016 had a shadow at the top of the detector, likely from the nozzle. Detecting background artifacts like nozzle shadowing may require radially subtracted images, or a saturating color scheme as used in fig. 9a,b. In such cases the shadowed region usually must be masked out, losing a lot of information. Background can also be increased when the X-ray beam interacts with something

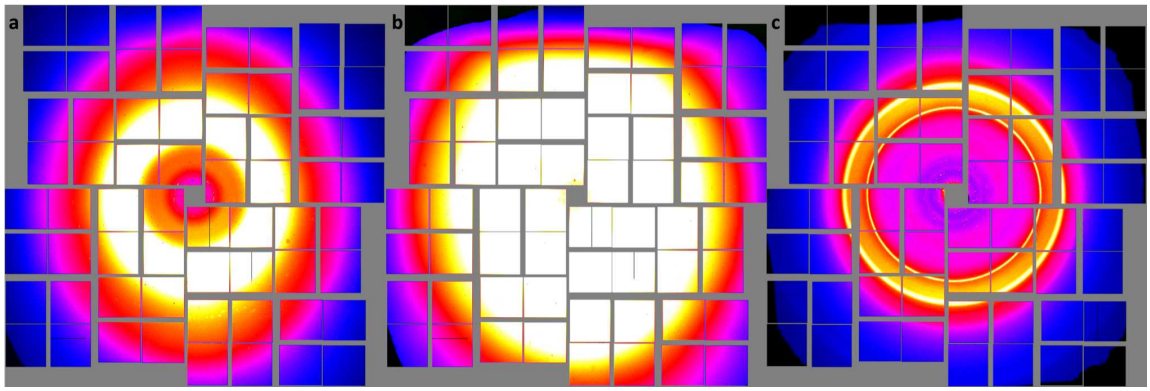

Figure 9. Detector shadows

Powder sums from two different runs from LCLS August 2016. The color scheme is over saturated to show the contrast between (a) a normal powder sum and (b) one with the top shadowed, likely from the nozzle. (c) An odd diffraction pattern from LCLS October 2015 with strong rings that may have resulted from hitting the nozzle.

unexpected, like the strong rings visible in 9c that show up in a single event, possibly from the X-ray hitting the nozzle.

In addition to background, sample delivery can have an effect on the sample that impacts the data. For jets, plotting the unit cell volume over time can reveal problematic trends. For example, the unit cell volume of PSII over time for LCLS October 2015 is shown in fig. 10, with vertical blue lines indicating sample changes. A sudden change in unit cell volume can indicate a change in sample, like is observed near the first sample change (vertical blue line) in the figure, or a change in setup that changed the distance to the detector. More problematic is slow shifts during data collection that can indicate the temperature or humidity of the delivery system is changing the crystals, such as the slow increase in unit cell volume for the first sample, and the slow decrease and then increase in unit cell volume of the third sample (between the second and third vertical lines). From this dataset, the fourth sample (between the third and fourth vertical lines) is the most stable. Another sample

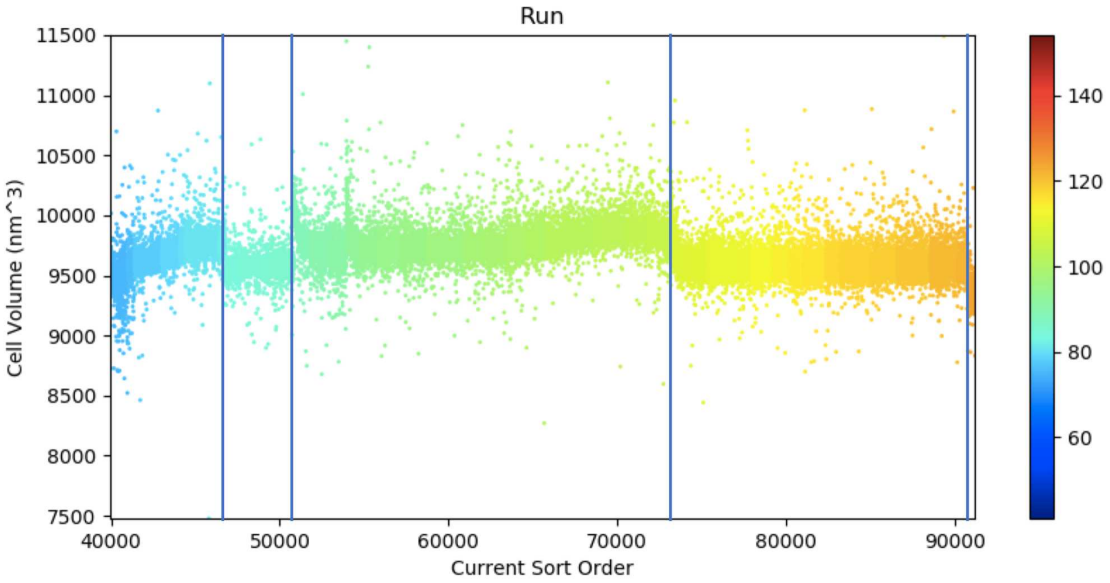

Figure 10. Unit cell volume over time from jet data

A plot of the unit cell volume over time, colored by run number. The blue vertical lines mark the approximate points where the sample was changed. At the beginning of the third section, the drop in unit cell volume may be caused by temperature effects. The fourth section is better because the unit cell volume is relatively constant over time. Data from LCLS October 2015 runs 73-124

problem to be aware of (though not a problem for PSII crystals) is flow aligning of needle-like crystals resulting in preferential orientations, making it difficult to sample the complete space.

1.4.2 Fixed-target Chip

PSII datasets have also been collected on a fixed-target system where crystals are transferred onto a silicon chip (fig. 11a) and the chip is moved with a Roadrunner goniometer (Roedig et al. 2017) to sync the X-rays with the holes in the chip. The Roadrunner goniometer is shown in fig. 11b.

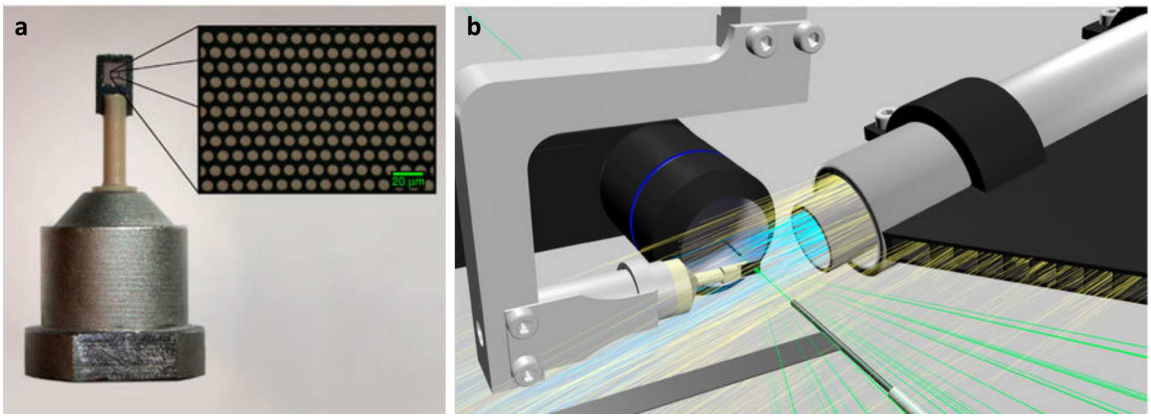

Figure 11. Overview of Roadrunner chip setup

The Roadrunner goniometer moves a silicon chip through the beam, exposing a hole on each pulse. (a) A sample chip (b) The goniometer holding the chip for the X-ray beam (green). Humidified air (blue) focused by helium (yellow) is streamed from the right. Figure adapted from (Roedig et al. 2017).

Data analysis of fixed-target chip data is complicated. The first noticeable problem is unit cell variation. This manifests as an oscillating pattern in plots by time, and is easiest to view with a plot of the unit cell volume by position on a chip (fig. 12b). However, plots such as fig. 12b depend on knowing the row and column on the chip of each diffraction pattern. Since the arrangement of wells in a chip are not necessarily square (for example, they may be hexagonal), the concept of rows and columns is an approximation. Also, in practice, only the chip row information is stored, and chip column information must be extrapolated based on row changes. Row information may also need to be edited by hand when there are multiple runs (periods of continuous data collection) with a single chip because starting a new run may reset the row numbering, or partially reset the row numbering with the first row labeled as 0 and the remainder of the rows correct. The take-away from this is that plots by position on the chip are good for getting a general idea of the trend, but may not be accurate enough for detailed analysis.

In LCLS November 2016, the unit cell variation correlated with changes in the diffraction resolution limit, with small and large unit cells not diffracting as well as the intermediate unit cell (see fig. 12a and 13a). One possible cause of the unit cell variation is a change in chip humidity during data collection for each chip. Figure 12 supports dehydration as influencing the unit cell because the average unit cell decreased over the time of data collection for each chip. A second possibly related variable is the time between blotting the sample on the chip and data collection.

In the second PSII fixed-target chip experiment, LCLS September 2017, more effort was put into tracking humidity in the data collection chamber and the length of time between sample transfer onto the chip and data collection. Interestingly, while unit cell variations were still observed, they were no longer as correlated to the diffraction resolution limit (fig. 13b).

The hypothesis during the beam time was that the unit cell variations were related to humidity and the diffraction resolution limit was more related to the time the chip was left sitting before data collection which could lead to potential dehydration during waiting time. The sample is transferred to the fixed-target chips in a fully humidified blotting chamber, and the chips are covered with a dark sleeve containing buffer in a sponge to prevent drying out during transfer. However, the sleeve is not fully sealed because the chip has to slide into the chamber. Therefore, the crystals could dry out during long waiting times, impacting the diffraction resolution limit. While crystals may be re-humidified during data collection, the diffraction quality may still be lower.

However, plotting the available data doesn't show strong support for this (fig. 14). For the humidity unit cell hypothesis, it's possible the humidity in the sample chamber changed over time (a typical chip for LCLS September 2017 took around

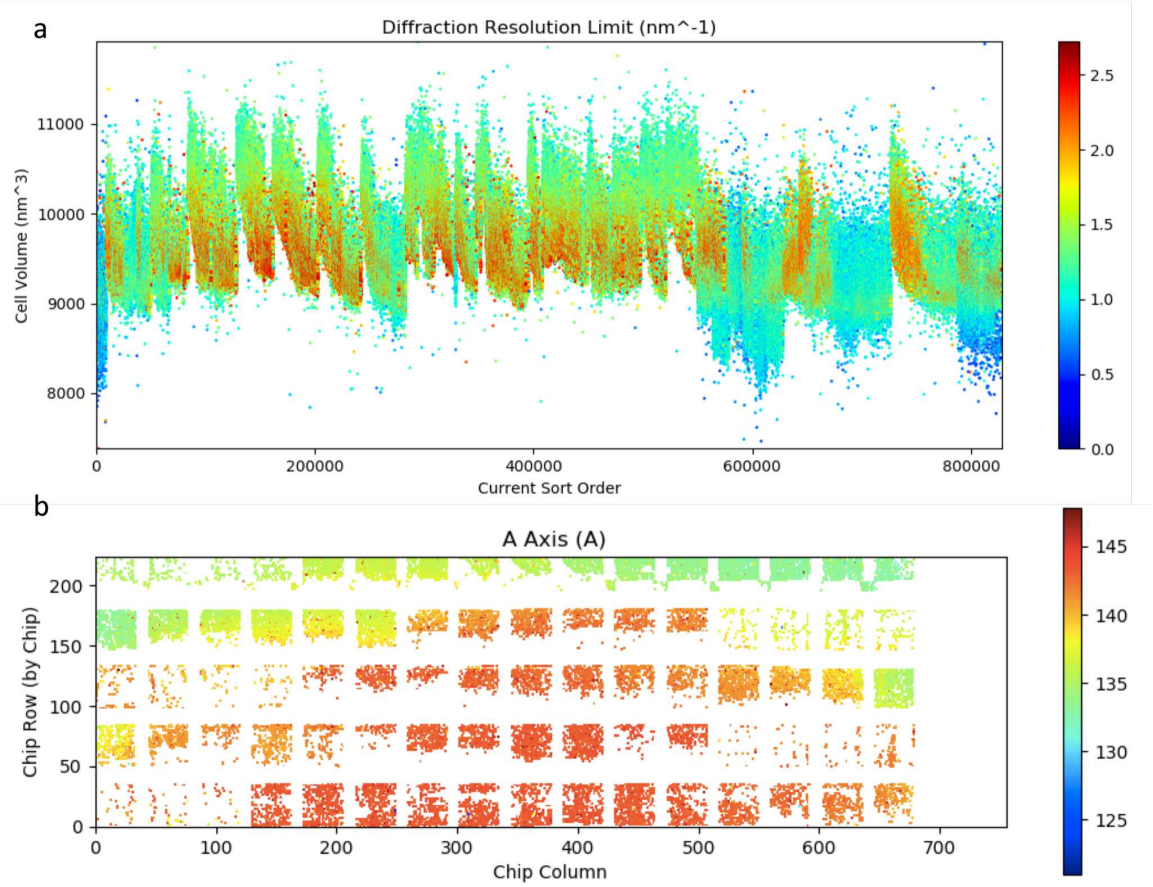

Figure 12. Unit cell changes for chip data

(a) The unit cell changes over time for LCLS November 2016, colored by diffraction resolution limit. Note this covers the entire dataset (33 chips), so the visible jumps reflect a new chip. Oscillations based from a single row on a chip aren't visible with the current scale. (b) A single chip from that dataset showing the change in unit cell by position.

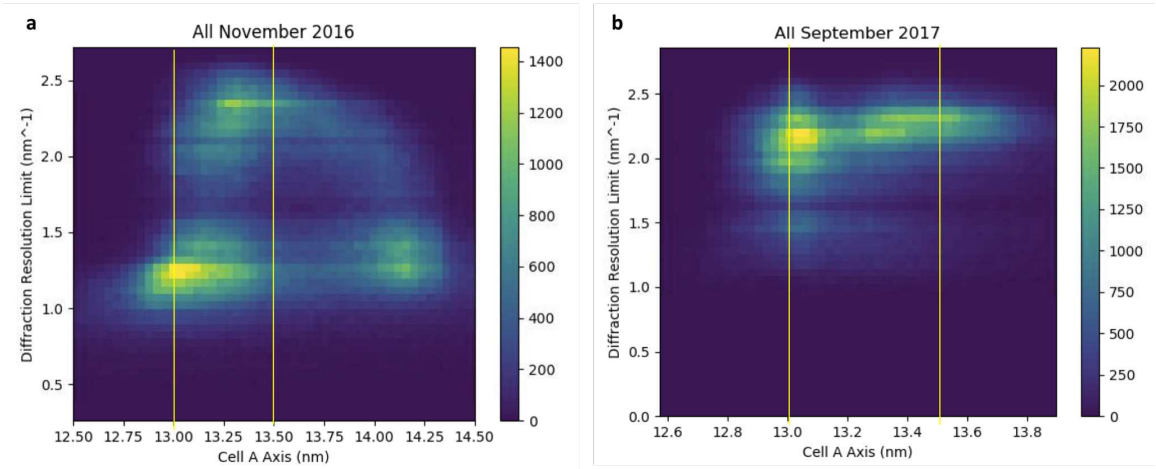

Figure 13. Unit cell and diffraction resolution correlation for chip data

A 2D histogram showing the correlation between the unit cell a axis (which, as the smallest axis, varies the most) and the diffraction resolution limit for (a) LCLS November 2016 and (b) LCLS September 2017 chip data. Note the axes are not identical for the two plots, so yellow vertical line identify x=13.0 nm and x=13.5 nm.

20 minutes for data collection). The available humidity readings reflected the values when the logger for the experiment recorded them in the data spread sheet which is not necessarily always consistent and has limited precision (two digits). It's also possible that humidity while blotting and in transporting matters, and neither of those values was recorded. Or, the unit cell change could depend on other factors, too, such as temperature.

For diffraction resolution limits, it is true that the chips with the highest percentage of better diffracting crystals sat for less than 20 minutes before data collection, and it's possible a more systematic study would show a stronger trend. Another possibility is that diffraction resolution limit is impacted by other factors such as sample or even laser illumination (some chips in LCLS September 2017 had laser illumination, but all chips in LCLS November 2016 were dark).

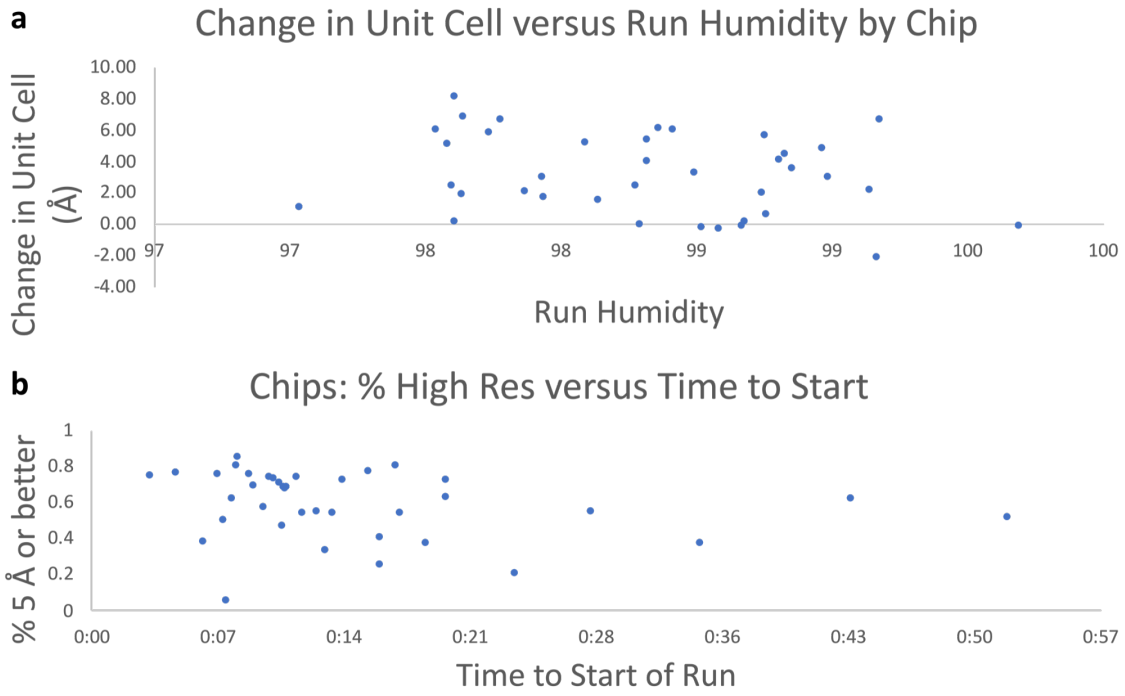

Figure 14. Unit cell change over chip humidity and diffraction resolution change over chip time to data collection

For LCLS September 2017 chips with at least 200 indexed patterns: (a) the average of the cell a axis for the first 100 indexed patterns minus the average of the a axis of the last 100 patterns on the y axis, against the average of the four humidity detectors in the sample chamber on the x axis, for each chip and (b) the percentage of patterns with 5 Å or better diffraction resolution limits against the time between chip blotting and the start of data collection, for each chip.

Another data analysis challenge with fixed-target chip data is preferential orientation. This is less easy to immediately visualize. For LCLS November 2016, the primary diagnostic for preferential orientation was to generate a 3D merge (Yefanov et al. 2014) and look at the number of patterns contributing to the merge at each voxel of the central planes (fig. 15a). A script was developed during the LCLS September 2017 beam time by Yaroslov Gevorkov for plotting the a, b, and c axis in 3D as blue, green, and red dots respectively (fig. 15b,c). To overcome the preferential orientation

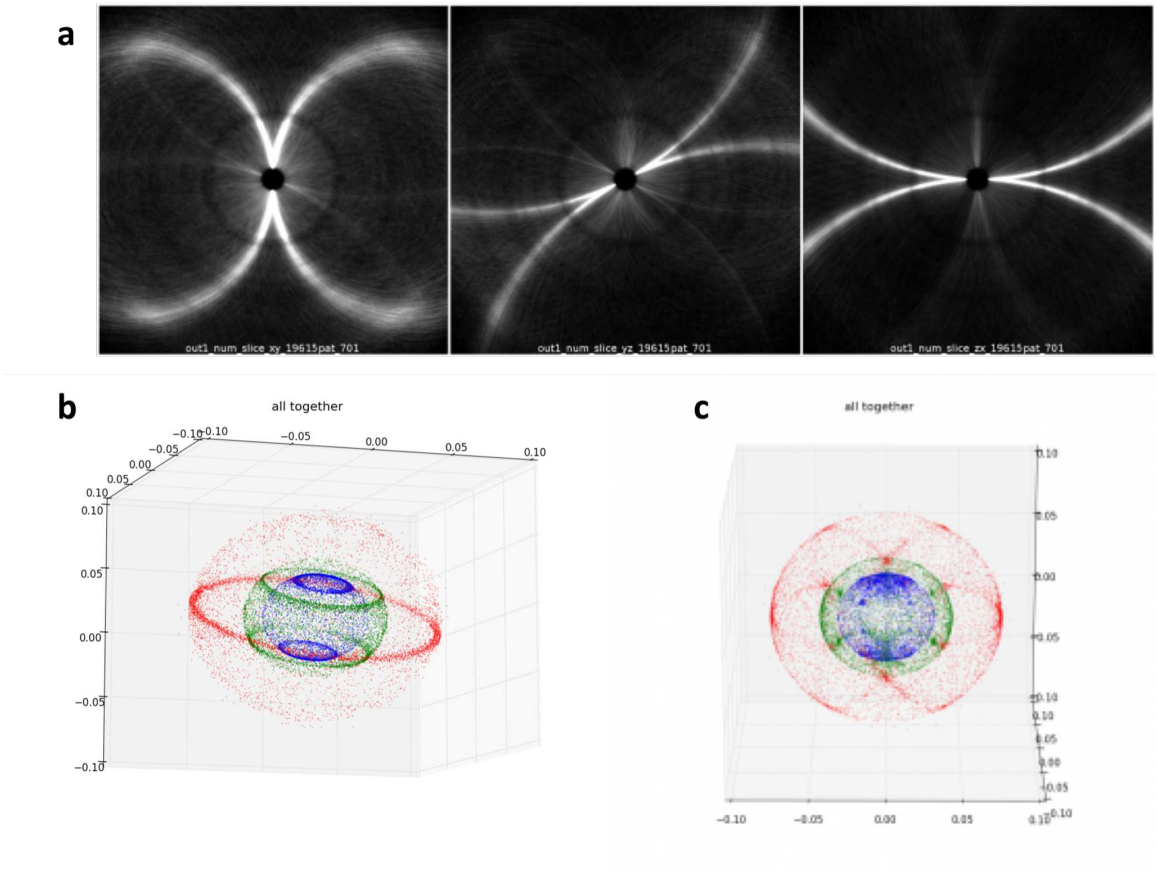

Figure 15. Preferential orientation on chips

(a) Central slices of a 3D merge showing the number of patterns contributing to each point. Strongly defined lines indicate preferential orientation. This is LCLS November 2016 PSII data posted to the electronic log book by Dr. Oleksandr Yefanov. (b,c) A three dimension plot of the a, b, and c unit cell axis in as blue, green, and red dots respectively from LCLS September 2017 data, posted to the electronic log book by Yaroslov Gevorkov. The two plots are from the same chip, but tilted at 32° and 16° respectively.

of the crystals in the chip wells, chips were tilted. Figure 15b,c show data collected from the same chip at two different tilt angles (32° and 16° respectively).

However, tilting the chip can result in other problems. One problem occurs when the X-ray beam hits the silicon chip at the right angle to cause a bright silicon diffraction peak, damaging the detector. Another problem that may be related to

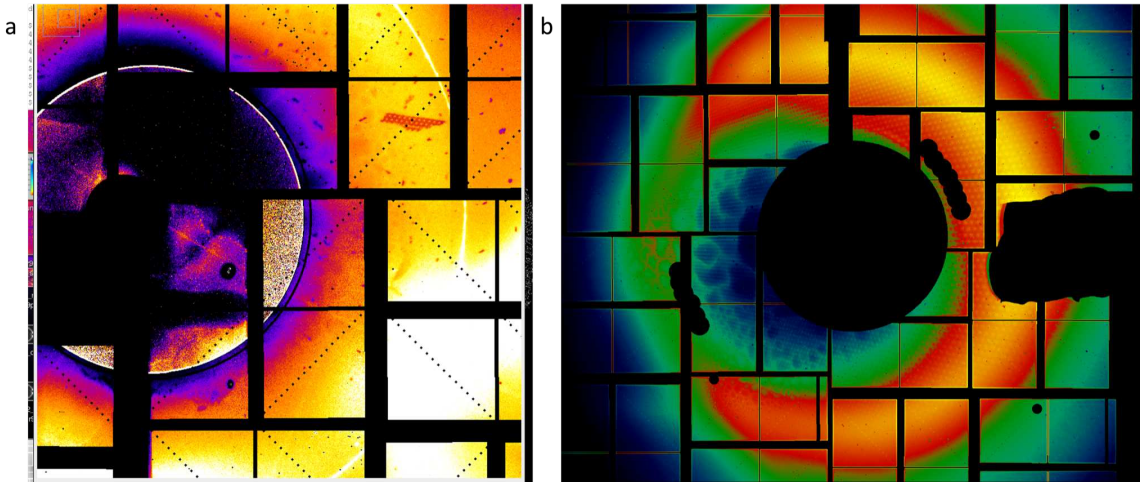

Figure 16. Background on chip datasets

Data collection on chips can have some unique artifacts. (a) A piece of the silicon chip broke off and is visible in the upper right quadrant of the detector. Image from the electronic log book for LCLS September 2017, by Dr. Oleksandr Yefanov. (b) The tilting of the chip shadowed the whole detector resulting in the a visible dot pattern, especially visible in the yellow/orange region. Image from the electronic log book for LCLS March 2018, by Dr. Oleksandr Yefanov.

chip tilting is the shadow of the chip appearing in the background of the frames, as shown in fig. 16b. Background correction for that kind of patterning on the detector has not been developed yet.

Another source of background can be the chip itself. During LCLS September 2017, the chip was damaged during data collection. As background on a detector, a piece of the chip is visible fig. 16a. Debris from the chip also collected on the objective lens of the in-line microscope through which laser illumination is accomplished. So, while chip data were collected with full laser excitation, it is uncertain how many frames are actually illuminated and at what point on the chip the laser has significantly decreased in intensity. As a partial solution during the beam time, data collection began from

the bottom of the chip so that falling debris would not obscure later frames. In LCLS March 2018, a tape drive was used to collect and remove the debris.

1.4.3 Delivery System Comparison by Hit Rate

The hit rate is the number of frames appearing to contain crystal diffraction patterns (hits) divided by the number of recorded events. It measures how many X-ray pulses interacted with a crystal. In terms of sample, the number of crystals that interacted with the beam compared to the total number of crystals is also important. The ideal system would result in every X-ray pulse interacting with a single crystal, and no wasted crystals. However, currently there are many empty frames with no diffraction patterns, and a few frames that contain diffraction from multiple crystals. Also, for jet delivery, sample waste occurs as the sample continues flowing between the X-ray pulses.

In serial data collection, data is collected in runs, where a run is a period of continuous data collection. The division into runs may be based on arbitrary times to keep data for each run to a particular size, or indicate changes in the experimental setup. For instance, changing a sample or moving the detector is done between runs. Hit rates are typically calculated separately for each run, and used for early detection of problems in delivery.

Estimating hit rates is complicated because defining hits is dependent on processing parameters. A loose hit finding criterion will report a large hit rate regardless of the number of useful patterns. Hit rates can also change during the experiment, as shown in fig. 17 and the change in hit rate can be caused by the sample or changes in the delivery system such as icing. This makes it difficult to directly compare delivery

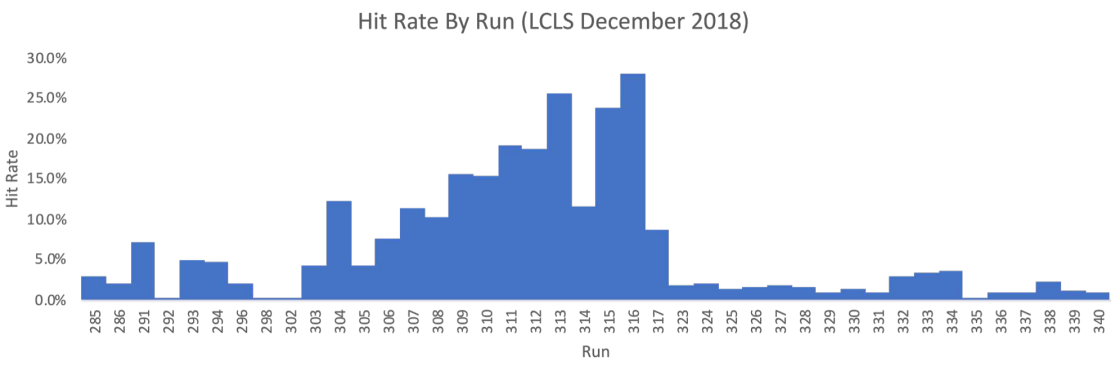

Figure 17. Hit rates by run

Hit rates by PSII run from LCLS December 2018 data (jet delivery). Hit rate can be impacted by the sample, like runs 302-317 having higher hit rates on average than the other runs corresponding to a different sample. Hit rates can also be impacted by delivery issues such as icing of the nozzle where run 291 has a decent hit rate but after icing run 292 has almost no hits.

methods by hit rates. Nevertheless, for LCLS PSII datasets summarized in table 2, datasets collected on a fixed-target chip (Roedig et al. 2017) have better hit rates than datasets collected with a jet. The result is expected because fixed-target chips have much higher crystal density than jets. A high crystal density in a jet leads to clogging. In contrast, sample can be transferred to chips iteratively until the desired density (30-60% coverage of the holes) is reached. However, the blotting process itself may influence the data quality and hit rate since too little crystal suspension may dry out and too much may let all the crystals flow to the bottom of the chip.

In summary, jet and fixed target chip delivery methods both have challenges for data analysis. For jets, the primary difficulty is changes during collection impacting the unit cell and low hit rates. While jets do have background, the background is typically radial and therefore easier to correct. Fixed target chip data collection more strongly suffers from unit cell variations, and for PSII datasets has the additional

Table 2. PSII datasets hit rates

Dataset	Delivery	Events	Hits	Hit / Event	Indexed	Indexed / Hits	Indexed / Events
2018 Dec	Jet	1,639,988	97,700	5%	32,673	33%	2%
2018 Mar	Chip	473,095	200,561	42%	77,251	39%	16%
2017 Sep	Chip	1,541,485	863,112	56%	471,714	55%	31%
2016 Nov	Chip	2,765,655	744,504	27%	493,451	66%	18%
2016 Aug	Jet	7,709,450	74,013	1%	63,514	86%	1%
2015 Oct	Jet	7,279,807	97,177	1%	74,711	77%	1%

The number of events, hits, and indexed frames (*CrystFEL 0.7.0* all PSII indexing, see A.1) by LCLS dataset, with hit rate, indexed rate by hits and indexed rate by events.

problem of preferential orientations. Also, the shadow of the chip in the background is not currently correctable. However, fixed target chip data collection is more efficient in both hit rate and sample usage.

1.5 Detectors

Diffraction patterns are recorded on detectors. The detector used is dependent on the XFEL setup, with diffraction data collected at LCLS recorded on the Cornell-SLAC hybrid Pixel Array Detector (CsPad) (Hart et al. 2012) and diffraction data collected at EXFEL recorded on the Adaptive Gain Integrating Pixel Detector (AGIPD) (Henrich et al. 2011). Both detectors contain multiple ASICs. Each ASIC contains an array of pixels, and at some level, ASICS and/or quadrants are moveable. Therefore, a geometry file is necessary to describe the position of each pixel in real space. Geometry files and optimization are discussed more in sections 3.3.1 and 4.2.3.

Detectors have sources of noise, and raw frames must be detector corrected. Detector corrections can be embedded in the software before the pattern is even

recorded, but at XFELs, detector corrections are typically applied after the data is recorded. Several parallel options exist for detector corrections. *Cheetah* (Barty et al. 2014) is a data reduction program that find hits and records the hits in HDF5/CXI file format. As part of the pipeline, *Cheetah* supports many detector correction options. Also, the XFEL software interface usually has detector correction algorithms. At LCLS, the software interface is called *psana* (Damiani et al. 2016), and at the EXFEL the software interface is called *Karabo* (Fangohr et al. 2018). Both software systems have detector correction support implemented. Datasets used in this thesis use *Cheetah* for detector calibration and background correction since it is already run for data reduction and peak finding.

A complete description of all detector corrections and calibrations is outside the scope of this work. However, for SFX data analysis, there are a few detector related sources of noise or systematic error that are important to be aware of. The next few sections cover maskable errors, dark calibrations, and gain corrections.

1.5.1 Maskable Detector Artifacts

Mask files are used to exclude parts of the detector from further analysis. Masks may be necessary for correcting experimental conditions such as the nozzle shadow described in section 1.4.1 and shown in fig. 9. Masks may also be necessary for detector artifacts. Two examples of detector artifacts are shown in fig. 18a,b. Detector artifacts are typically found by looking at sums of all the frames for a run. These sums are sometimes called powder sums because they are a computational equivalent to a powder diffraction pattern.

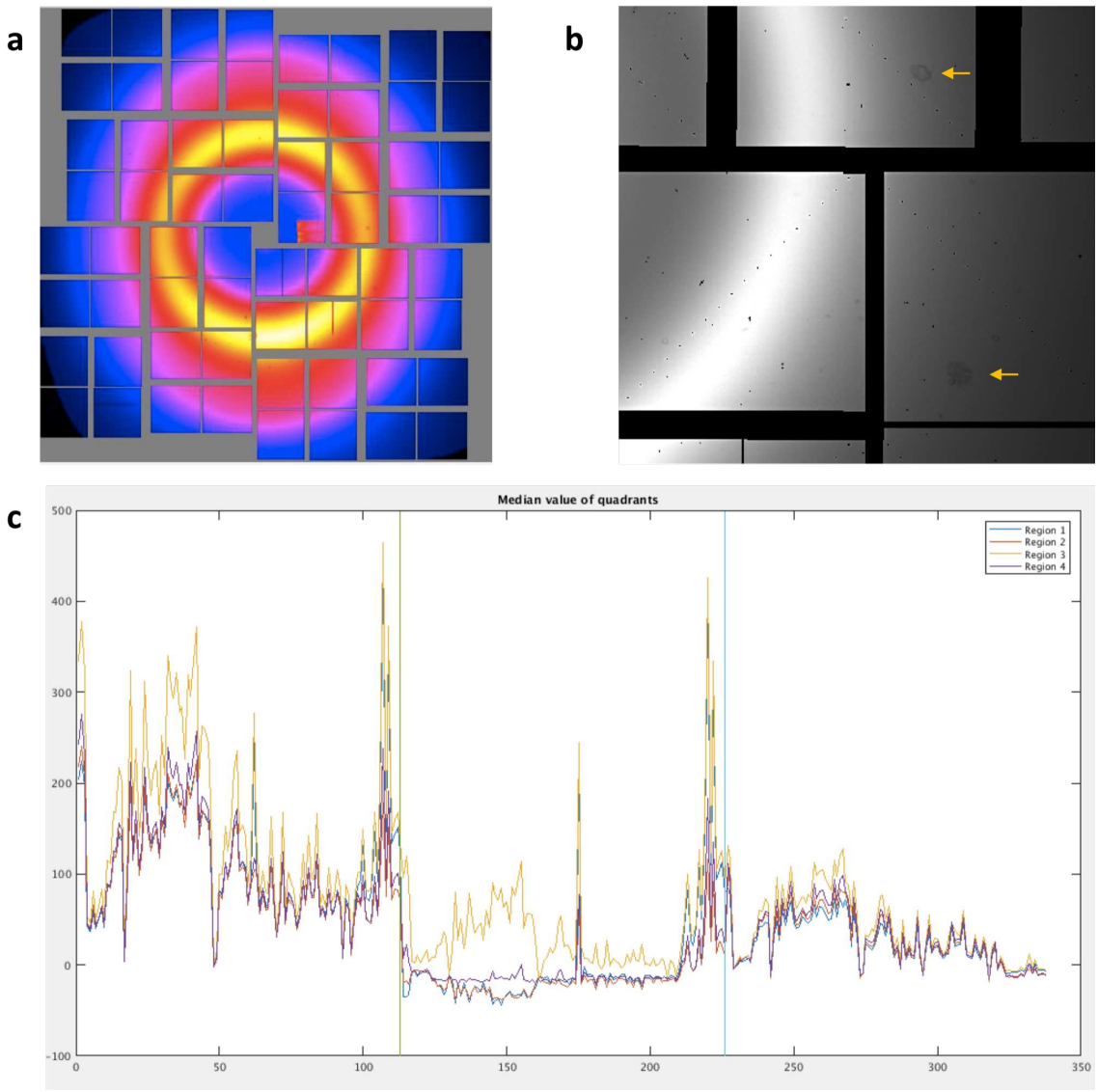

Figure 18. Background on CsPad detector

Some detector level background issues from LCLS Aug 2016. (a) The panel near the center top right gave abnormal values for some runs. The image is the powder for run 140. (b) Speckles appearing in the background (highlighted with yellow arrows), image cropped from the electronic log book post by Dr. Oleksandr Yefanov. (c) Median values of the four different quadrants (yellow, purple, blue, and red lines) for 112 patterns with three different radial subtraction parameters (x-axis). In general, the third quadrant (yellow) is brighter than the other three.

In fig. 18a, the run sum showed that a region near the center of the detector gave abnormally high readout. This type of artifact can also occur for particular pixels, referred to as hot pixels, that always give a high value. The converse is also possible, with cold pixels that always give a low value. All of these should be masked for better data quality. Figure 18b has odder artifacts that, in Dr. Oleksandr Yefanov's words, look like someone sneezed on the detector. A few of the speckles are marked with yellow arrows in the figure.

1.5.2 Detector Dark Calibration

The background signal of the detector without any input is referred to as the dark signal, or pedestal. It is treated as a pixel specific offset to be subtracted from each frame. The background signal can change over time, and particularly if detector settings are changed. So, dark calibrations are taken every so often for correcting the data.

If dark calibrations are done by *Cheetah*, then the data analyst must generate the dark calibration input files for *Cheetah* and update the configuration file to use the correct dark calibration for each run. At LCLS, these steps are accessible from the graphical user interface. Typically, a given dark calibration file is used for all runs following it until a new dark calibration is generated. However, if something changes between dark calibrations, it may be appropriate to use the first dark calibration file after the run.

Inaccurate dark calibrations can leave systematic errors in the data. For example, Figure 18c shows the median signal by quadrant for some patterns from LCLS August 2016 using different background subtraction methods. One quadrant is typically much

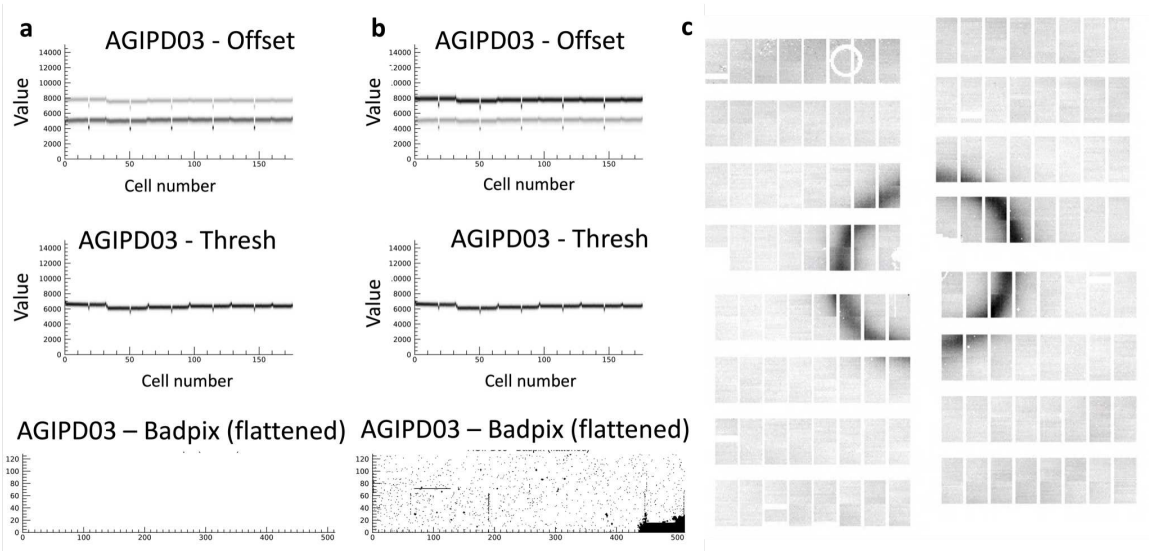

Figure 19. Agipd Detector Corrections

The Agipd detector has three gain modes for each pixel, so detector corrections occur on a pixel by pixel basis. (a) Output from a failed detector correction for a single panel. The lower line on the top graph is too dark, indicating that a dark calibration marked as low or medium gain may have actually been a high gain. (b) The correction for the same panel from the next set of successful dark calibrations. The top line in the first figure is dark because the thresholds for switching between medium and low gain overlap. The final panel at the bottom shows the bad pixel maps which are generated for each new dark calibration. (c) The powder sum of non hits for the first data run after the detector correction in (b). The panel shown in (a) and (b) is the fourth panel down from the top right and the region near the beam stop is the masked region visible in the third graph of (b).

higher than the other quadrants. The exact cause of this is unknown, but it could indicate that the dark calibration correction was not done or that the calibration files weren't correct for this experiment.

At the EXFEL, the AGIPD detector has three gain modes for each pixel, so dark calibrations are more complex. A dark calibration must be generated for each gain mode. The *Cheetah* GUI does not currently support generation of dark calibrations at the EXFEL, and so a mixture of python and IDL scripts must be run from the

command line. Logging of information at the EXFEL is still new, so information about which runs correspond to dark calibrations for particular gain modes can be lost. When bad or mislabeled dark runs are used, the scripts to generate dark calibrations can fail. Figure 19 shows output from a dark calibration that failed to generate, output from the next successful dark calibration, and a frame from the following data run. Dark calibration at the EXFEL also autogenerates masks for hot pixels and hot regions, as shown in the bottom section of fig. 19b.

1.5.3 Detector Gain and Saturation

Detector gain relates the number of photons to the digital value recorded. It can be thought of as the amount the recorded value will change for a photon. So, in high gain, the change in recorded value from a single photon is large, and in low gain the change from a single photon is small. In high gain, values are more precise but the maximum number of photons that can be accurately recorded is lower.

Saturated pixels are pixels where the maximum recorded value is reached and so the true value is not known. During a beam time, checking for saturated pixels is important to ensure the data will be as useful as possible. This is typically done by plotting the intensity of found Bragg peaks (recorded by *Cheetah*) against the radius of the peak with the script *peakogram*. Low resolution peaks are typically brighter than higher resolution peaks, so the brightest pixels are typically found at smaller radii. Results for LCLS October 2015 and LCLS August 2016 are shown in fig. 20a,b respectively. In addition to diagnosing saturated pixels, the plots can also be used to get a rough idea of the overall intensity. However, comparing results is

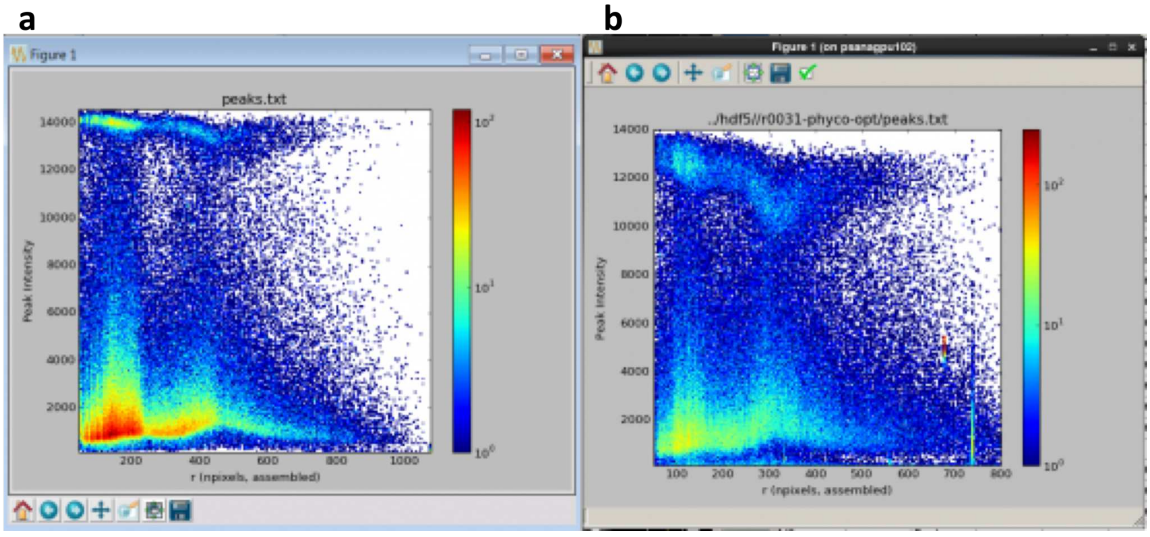

Figure 20. Peakogram and attenuator

Peakogram results for (a) LCLS October 2015 and (b) LCLS August 2016, posted together by Dr. Oleksandr Yefanov in the electronic log book for LCLS August 2016. The two experiments had different wavelengths and different detector distances, so the two plots are not really comparable.

not straightforward. Since LCLS October 2015 and LCLS August 2016 used different wavelengths and different detector distances, they are not directly comparable.

One way to keep high gain mode for high resolution pixels without saturating or damaging the detector at low resolutions is collecting data with attenuation. The X-ray beam can be attenuated to have lower flux. Or, a physical attenuator can shield the low resolution part of the detector, as done in all the PSII LCLS experiments collected on a fixed target chip. Example patterns with a foil at low resolution are shown in fig. 21 without and with correction. Using a physical attenuator requires masking the edge of the attenuator at each detector distance used, and computationally multiplying the value of shielded pixels by an offset dependent on the material of the physical attenuator. The multiplied values per pixel are stored in a file also referred to as a gain file. This correction can be applied by *Cheetah*. However, *Cheetah* applies

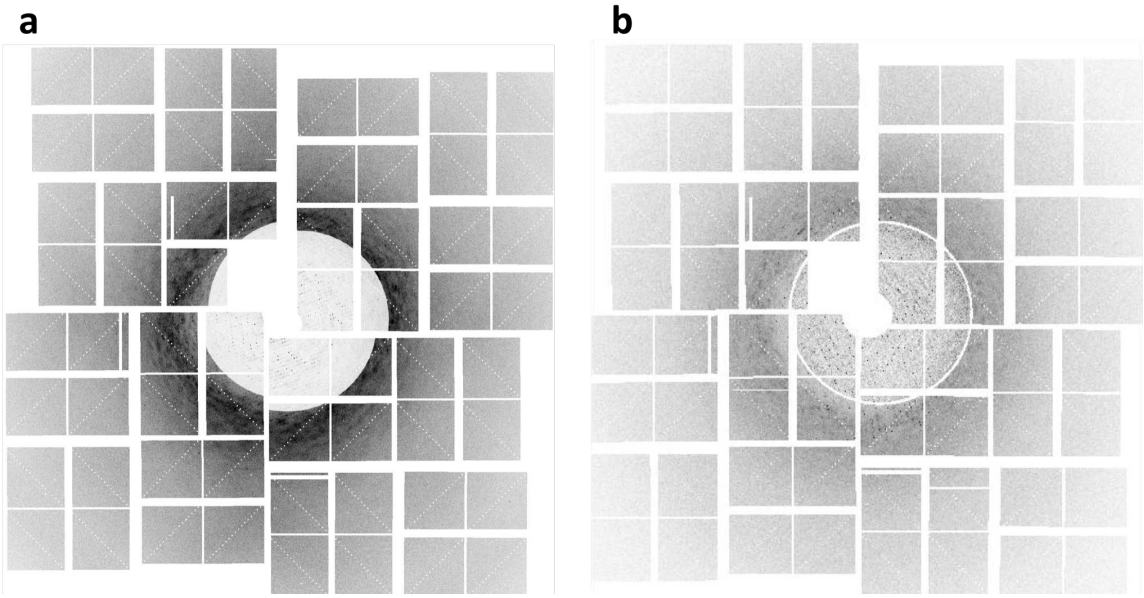

Figure 21. Physical attenuators

Images from LCLS September 2017. The images are not the same frame because only gain corrected data was saved so the image for (a) is from the electronic log book and cannot be mapped to the saved gain-corrected images. (a) An image before gain correction was applied. (b) An image with gain correction applied by *Cheetah*. The white circle is the mask for the edge of the foil.

detector corrections including gain correction before determining if the frame is a hit, so gain correction can inflate the hit rate by amplifying noise. For that reason, gain correction was not applied by *Cheetah* for LCLS March 2018 data.

1.6 Analysis

In conventional crystallography, much of the data processing is automated with many steps combined into one command. Since SFX is still a relatively new field and requires additional steps, most of the steps are still separate commands that are each manually optimized. Multiple software suites have been developed for SFX and the

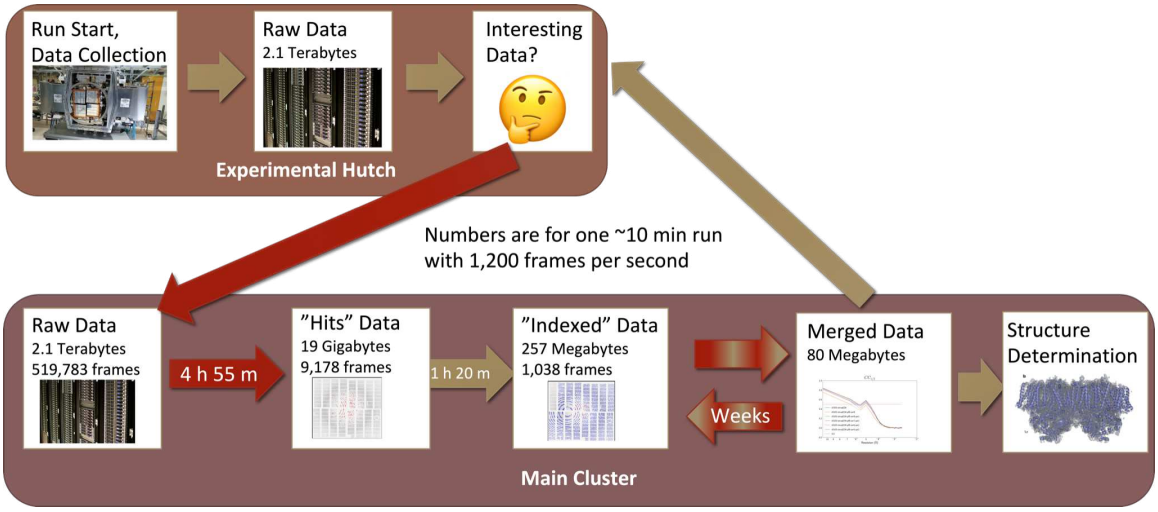

Figure 22. Overview of data analysis pipeline (EXFEL)

An overview of data analysis at the EXFEL with data size and times for run 103 (PSI data) from EXFEL August 2018 with red arrows indicating time consuming steps. EXFEL has two computing clusters, and analysis takes place primarily on the main cluster. Data must be marked as interesting to be copied from the experimental cluster to the main cluster for further processing. Hit finding reduces the data to just hits by finding images containing Bragg peaks. Indexing determines the orientation of the lattice and integrates reflections. An iterative process of parameter optimization may then take weeks, with final merged results being used for structure determination with conventional methods. Analysis results are used to give feedback during the experiment. The image of the AGIPD detector under Data collection is from (Mancuso), the server image is from (Victorgrigas 2012), the hit and indexed images from that run, the plot under merged data from merging statistics for different parameters for that run, and the PSII electron density adapted from (Ayyer et al. 2016).

division of steps among the different programs depends on the software being used. The most common SFX analysis toolkits are *CrystFEL* (White et al. 2012, 2016) and *cctbx.xfel* (Hattne et al. 2014). Analysis in this thesis uses *CrystFEL* and other software tools developed by the Center for Free Electron Lasers (CFEL) at Deutsches Elektronen-Synchrotron (DESY).

The steps of data analysis can also depend on the facility where data was collected. For instance, at the EXFEL, data is not copied to the offline cluster for analysis until someone marks the data as "interesting." Since this is not usually until a run is completed, there is an additional delay for data to become available compared to LCLS where data is copied as it is collected. Also, as mentioned in section 1.5, whether a detector correction step is necessary also depends on the facility with XFELs often requiring detector correction. In contrast, synchrotrons often correct frames before they are saved.

An overview of data analysis steps is shown in fig. 22 with the EXFEL setup that requires someone to mark the data as interesting before it is copied. The first data analysis step is usually a data reduction step to save only frames containing diffraction patterns. This reduces the data to be saved long term or being copied and also provides a smaller dataset for processing therefore reducing the processing time for later steps. Determining whether a frame contains diffraction is typically done by searching for Bragg peaks. The found peaks are stored in the output, so the peak-finding step is often paired with data-reduction. However, peak finding can also be run during indexing. Data reduction and peak finding together are referred to as hit finding in this thesis, and are described in chapter 2.

Indexing uses found Bragg peaks to determine the orientation of the crystal lattice, identify the unit cell, and locate reflections. Reflections, as used in this thesis, refers to the predicted locations of Bragg peaks. Reflections are labelled with the Miller index (HKL) of the lattice point which is used to determine the location of the point in 3D. The 3D space of reflections is referred to as reciprocal space, and h, k, and l are the axes in reciprocal space corresponding to the unit cell axes a, b and c in real space. In *CrystFEL*, indexing usually includes an integration step. Integration sums up the

values at each reflection and stores that final intensity in the output. The output of *CrystFEL* indexing and integration is a stream file. Indexing is described in chapter 3.

Merging combines the results from multiple crystals into one dataset. The *CrystFEL* programs for merging, *process_hkl* and *partialator* use the integrated reflection values from the stream file output to create an hkl file, which is essentially a table with a single intensity value for each Miller index. Merging can also be done in 3D, resulting in a 3D volume instead of values only for Miller indices. This can be done with the program *merge3d* (previously named *intor*) (Yefanov et al. 2014), which uses the orientation in the stream file to map each pixel to the appropriate voxel in 3D. Merging to a hkl file is discussed in chapter 5. Three-dimensional merging is discussed in the context of continuous diffuse scattering analysis in chapter 7.

After merging, SFX data analysis converges with traditional X-ray structure data analysis. A Fourier transform (Bracewell and Bracewell 1986) is often used to transform the values in reciprocal space to an electron density map in real space. However, a Fourier transform requires both the amplitude and phase in reciprocal space, and all that is known from diffraction is intensity values that are proportional to the amplitude. Therefore, the next step is phasing to determine the phase of each Miller index.

Many methods exist for determining phases. For datasets in this thesis, molecular replacement is used. Since structures of the photosystems are already available, phases from those structures are used as initial phases that are refined by the experimental intensity values. Refinement is the process of optimizing phases and locations of atoms to fit the experimental intensity constraints and real space constraints such as the sequence of amino acids in the protein. Phasing and refinement are discussed in chapter 6.

As mentioned in section 1.2, continuous diffuse scattering has been used to extend the resolution of PSII data and the structure from 4.5 Å to 3.5 Å (Ayyer et al. 2016). Resolution extension uses the Bragg model determined with the steps described above as a loose constraint for iterative phasing of a 3D merge to output a 3D volume of intensities and phases. This volume is converted to Miller indices and then refined with conventional refinement programs. Chapter 7 describes the steps in the process in more detail.

1.7 Motivation

This section introduces background for three main areas addressed in this thesis. Section 1.7.1 describes combining datasets from multiple crystals and the snapshot crystallography equivalent of selecting subsets of the data for further processing. Section 1.7.2 describes the affect of processing parameters on output statistics. Finally, section 1.7.3 describes some computational challenges with SFX data addressed in this thesis.

1.7.1 Subset Selection

The use of multiple crystals in a dataset is not unique to snapshot crystallography. In fact, experimental phasing methods such as multiple isomorphous replacement require data from multiple crystals. However, the question of which crystals can be merged together has received renewed interest, beginning with single-wavelength anomalous dispersion (SAD) and multiple-wavelength anomalous dispersion (MAD) datasets. Anomalous signals from heavy atoms can be used for phasing, but are

relatively weak. Initial work compared results from single datasets with cumulative combinations of datasets. The results showed that the multi-crystal datasets had better signal than single datasets (Liu, Zhang, and Hendrickson 2011).

Further work developed statistics for determining which datasets were combinable. The three statistics used in initial studies for cluster analysis were the unit cell, the relative anomalous correlation coefficient (RACC) of the single dataset to the whole dataset, and the diffraction dissimilarity between pairs. Datasets were created by combining crystals and by combining successive wedges of data. Multiple crystal datasets were shown to be effective and performance was better combining wedges than crystals, showing the benefits of reducing radiation damage (Liu et al. 2012; Liu, Liu, and Hendrickson 2013). Interestingly, a later paper emphasized the importance of the diffraction dissimilarity for determining combinable crystals since the RACC and unit cell deviations were not sufficient to identify a dataset whose inclusion decreased the overall quality (Liu et al. 2014). Other papers have also found that including all available data does not always improve the quality (Diederichs and Karplus 2013). However, multiple crystal datasets surpassed asymptotic limits of single crystal datasets in Map CC values and suggested that including multiple crystals overcomes systematic errors in single crystal datasets (Liu et al. 2014).

The software *BLEND* clusters crystal datasets based on unit cells (Foadi et al. 2013) and unit cell analysis was also used to detect to different clusters of unit cells in (Zeldin et al. 2015). In contrast to clustering, search algorithms such as the genetic search algorithm have also been adapted for grouping crystal datasets (Zander et al. 2016). Another work used an iterative approach beginning with unit cell deviations and followed by outlier rejection (Guo et al. 2018).

However, an initial study with an SFX dataset did not find improvements by limiting analysis to a subset of the data. The initial goal of the paper was to compare self-seeded to SASE XFEL beams, but found no differences. Further analysis showed no improvements when clustering on unit cell, rejecting runs based on cross correlation, or removing outliers with a low CC to the merged data (Barends et al. 2015). Yet, (Assmann, Brehm, and Diederichs 2016) suggest that outlier rejection should improve SFX dataset quality and (Diederichs 2017) found evidence for systematic errors in a PSI SFX dataset based on clustering analysis. Also, the continuous diffuse scattering resolution work on PSII used only a subset of the strongest patterns to improve the signal to noise (Ayyer et al. 2016).

Several sections of this thesis examine the effect of the subset of SFX data used for analysis on further statistics. Section 7.4 shows the effect of binning by several indexing statistics on continuous diffuse scattering analysis. Section 4.3.2 introduces subsets based on unit cell and diffraction resolution limit to address the oscillating unit cells found in fixed-target chip PSII datasets. The subsets are also compared in sections 5.1.2, and 6.3. Section 4.3.3 compares some of the unit cell and diffraction resolution limit subsets with subsets based on clustering analysis, both relative (Liu et al. 2012) and pairwise (Diederichs 2017).

1.7.2 Parameter Optimization

There are a lot of options in SFX data analysis, both in choice of software packages and in parameters at each step. For PSII datasets, the effect of software was examined in (Wang et al. 2017). Work in this thesis presents many intermediate screens showing the effect of processing parameters at each step. Section 3.3 shows the affect of indexing

parameters on merging statistics. Two indexing conditions are further processed and compared throughout later sections (see 4.3.1, 5.1.1, and 6.2).

Most SFX analyses have used a single set of parameters for the entire dataset. However, the software package *IOTA* optimizes parameters for each pattern (Lyubimov, Uervirojnangkoorn, Zeldin, Brewster, et al. 2016) for data processed with *cctbx.xfel* (Hattne et al. 2014). In this thesis, chapter 4 introduces a new tool called *DatView* that gives similar functionality to *CrystFEL* users. It's use for indexing optimization is described in section 4.2.

1.7.3 Processing Speeds

Processing speeds for the early steps in SFX analysis are particularly important to give fast feedback during an experiment. Section 2.3 describes an improvement for *Cheetah* (Barty et al. 2014) to improve processing speeds. The other improvement to processing speeds in this thesis is the software *DatView* (see chapter 4). *DatView* improves the time needed to visualize datasets and create subsets of data for further processing. It is particularly useful for large datasets, such as the fixed-target chip PSII experiments that are too large to process as a whole with some programs. *DatView* can load all statistics from the LCLS PSII datasets (over 2.5 million patterns) in less than a minute and is used to generate the comparison figures across datasets such as fig. 3. Chapter 4 gives detailed time comparisons with other tools and demonstrates the unique capabilities available with *DatView*.

Chapter 2

HIT FINDING

Hit-finding, as defined earlier for this dissertation (see section 1.6) is the process of data reduction by locating Bragg peaks. This chapter will focus on *Cheetah* (Barty et al. 2014), which performs three main tasks: 1) apply detector corrections (see section 1.5) 2) locate Bragg peaks and 3) save hits into a CXI/HDF5 output format.

There are other tools covering all or part of the tasks *Cheetah* performs. For example, *cctbx.xfel* has hit finding tools (Hattne et al. 2014), and beamline tools like *Psocake* are also available (Shin, Kim, and Yoon 2018). Another tool by CFEL, *OnDA* (Mariani et al. 2016) is particularly relevant. *OnDA*, standing for online data analysis, is focused on providing fast feedback during the experiment. It provides hit rate (see section 1.4.3) estimates during data collection by skipping more time consuming steps such as saving the output.

With *OnDA* providing fast experimental feedback, the main purpose of running *Cheetah* during an experiment is saving the hits in CXI/HDF5 output format for further analysis such as indexing (see chapter 3). Since indexing results are also used during an experiment to detect trends such as changing unit cells (see section 1.4), *Cheetah*'s speed also matters.

The first section in this chapter describes peak finding (2.1). The next two sections cover extensions to *Cheetah*. Section 2.2 describes an extension to allow *Cheetah* to read the CBF file format (Bernstein and Hammersley 2006) commonly used at synchrotrons. Section 2.3 describes a parallelization script to improve *Cheetah* processing speeds.

2.1 Peak Finding

The aim of peak finding is to locate Bragg peaks in an image. A Bragg peak is expected to be a particular size from one to a few pixels depending on crystal order, unit cell size, and detector distance. It is also expected to be brighter than surrounding pixels. A frame that actually contains diffraction from a crystal is expected to contain multiple Bragg peaks.

2.1.1 Peak Finding Parameters in Cheetah

Because the number and size of Bragg peaks is dependent on the sample and the detector distance, peak finding is optimized at each experiment. Different peak finding parameters may also be used for different samples or runs within an experiment. Hit finding parameters for *Cheetah* are given in an ini file to the command line. The portion of the ini file relevant to peak finding is:

```
hitfinder=1
hitfinderDetectorID=0
hitfinderAlgorithm=8
hitfinderADC=50
hitfinderMinSNR=5
hitfinderNpeaks=20
hitfinderNpeaksMax=5000
hitfinderMinPixCount=1
hitfinderMaxPixCount=20
hitfinderLocalBgRadius=3
#hitfinderMinPeakSeparation=0
hitfinderMinRes=0
hitfinderMaxRes=1200
#peakmask=../../calib/mask/mask.h5
```

The first line tells *Cheetah* to run hit finding (1 for on, 0 for off), and the second line specifies which detector is giving the crystal diffraction pattern. *Cheetah* can be run saving output from multiple detectors which is used in LCLS August 2016 to save an XES spectrum detector in addition to readout from the CsPad. The third line determines the algorithm. The 8th algorithm, referred to as peakfinder8, is used for all analysis in this dissertation, and is now also available at the indexing step (White 2019). Other algorithms are documented online (Kirian and Zatsepin 2015).

ADC is a threshold value, so the pixel value must be at least the hitfinderADC to be considered. This parameter is not relevant for most datasets in this dissertation, since the CsPad readouts are typically much higher than 50. However, when *Cheetah* was adapted for use at synchrotrons (section 2.2), the Pilatus detector (Henrich et al. 2009) used in APS August 2016 had much lower readouts, and the usual value of 50 meant many hits were not found. Good values for the hitfinderADC can be determined by looking at the pixel values of peaks on raw frames of data.

The most frequently modified parameter in experiments referenced in this dissertation is hitfinderMinSNR. The MinSNR is the minimum signal to noise ratio of the peak compared to the background pixels (with the radius of pixels considered in the background controlled by hitfinderLocalBgRadius), and is more useful than the hitfinderADC threshold because it works in areas of high background such as peaks found in solvent rings.

The Npeaks and NpeaksMax control how many peaks must be on a pattern for it to be considered a hit. Too many peaks indicates that peak finding parameters are too loose and noise is being found. Npeaks depends strongly on sample. Samples with high molecular weight like the photosystems with large unit cells will have more peaks

than systems with smaller unit cells. Also, well-diffracting samples like lysozyme will have more peaks than poorer diffracting samples.

The minimum and maximum pixel counts control the range of Bragg peak size. In many of the photosystem datasets referenced, minPix has to be 1 because the detector distance is small to achieve higher resolutions. This is problematic because hot pixels are usually a single very bright pixel that therefore fits the criteria for a Bragg peak. Masking becomes particularly important. A mask that is applied just for the peak detection step is available as the last parameter. The maximum number of pixels helps prevent ice rings or streaks being identified as peaks.

The hitfinder min and max resolution control the annulus searched for Bragg peaks. A minimum resolution can be important to avoid noise close to the beam stop. A maximum resolution can cut off noise from ice rings or other high resolution noise. The # symbol comments out a line so it is not used. In the example, neither the minPeakSeparation nor peakmask are used.

2.1.2 Parameter Optimization

The parameters in the ini file above were the initial parameters used with the sample Proteinase K (PK) at APS August 2016 (Martin-Garcia et al. 2017) which was recorded on a Pilatus detector (Henrich et al. 2009). (For comparison, PSII for LCLS December 2018 had an ADC=150, MinSNR=8,and nPeaks=15, and other parameters were the same). In run 124 of APS August 2016, only a single hit was identified (event 383, see fig. 23b). With 1092 events in the run, that gives a hit rate of less than 0.1%. With the software run and a result output, the next question is whether the result is "good" or "correct."

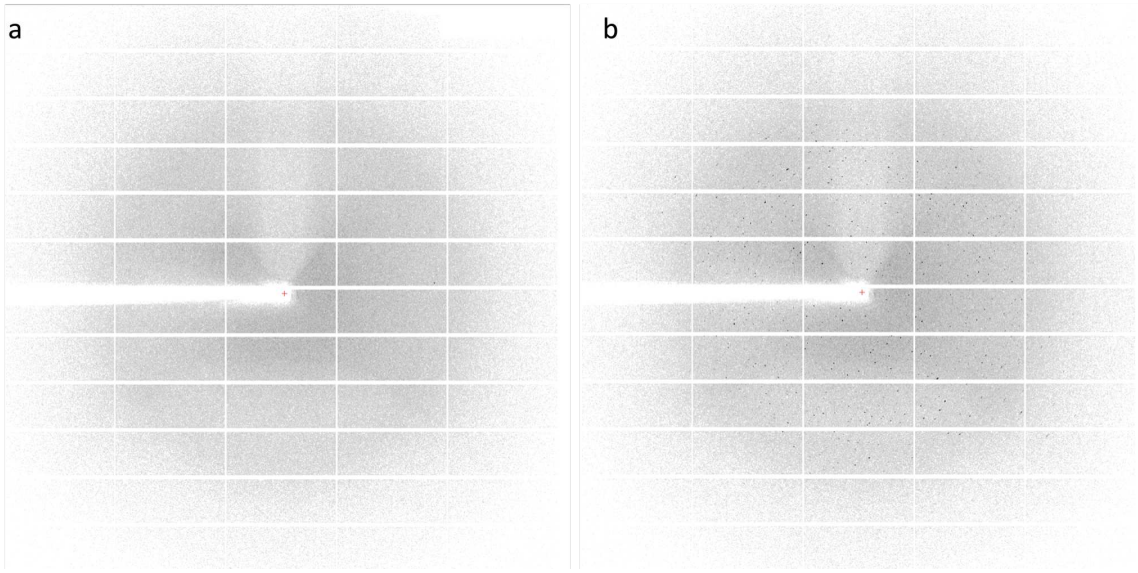

Figure 23. Frames from APS

Two frames from APS August 2016 run 124 (sample PK). Data was collected at the GM/CA 23-ID-D beamline on a Pilatus detector. (a) event 382, a non-hit (b) event 383, the only hit identified with initial hit finding parameters.

The hit rate itself can be a sign whether the hit finding parameters are working correctly. A high hit rate like 100% is almost certainly a sign that the hit finding parameters are too loose. A low hit rate like the 0.1% here is, unfortunately, ambiguous because it could be that there is a problem with the sample or delivery or the problem could be with the hit finding parameters.

Indexing rates can also be checked. A really high indexing rate can indicate that only the strongest patterns are being found so weaker parameters should be tried to pick up more patterns. Conversely, a low indexing rate indicates that most of the hits were very weak or not really hits in the first place. However, indexing rates depend on many factors in addition to hit finding, so they are not ideal for hit finding feedback. In this case, event 383 could not be indexed with the peaks found by *Cheetah* and the indexing rate of 0% would incorrectly imply that there were no hits in the run.

Sometimes, powder sums can be used to diagnose hit finding. If strong rings appear in the sum of non-hit patterns, it can be a sign that many hits are being missed. However, with low hit rates, a small percentage of missed hits is unlikely to result in strong rings in a powder sum, and rings in a powder sum can also be caused by other artifacts like the strong rings in fig. 9c.

In the end, initial peak finding parameters often require visually looking at many events to determine by eye what the expected hit rate should be by counting the number of events that look like real diffraction patterns. For synchrotron frames such as used in fig. 23, *Adxv* is a useful tool (Arvai 2012). For *Cheetah* output, *cxiview* (packaged with *Cheetah*) is a useful viewer.

In this case, looking through run 124 revealed other frames that looked like diffraction patterns such as events 586 and 1082 shown in fig. 24. With a set of patterns that look like hits, tools like *onda_parameter_tweaker* can be used to see the affect of different hit finding parameters. However, the peak finding algorithm is not necessarily identical to *Cheetah*'s, so parameters may still need editing for use with *Cheetah*.

For this run, several hit finding parameters were tested. The main change necessary was to lower the ADC as the Pilatus detector has much lower values than the CsPad. Figure 25 shows seven hit finding conditions ordered loosest to most strict along the vertical axis. The horizontal axis lists frames with red shading indicating the indexing timed out on that pattern and green indicating the pattern was successfully indexed. Based on the table, the third row of parameters with ADC 13, SNR 5, Max-Res 1200, and a peak mask blocking a hot pixel were used for all other PK runs.

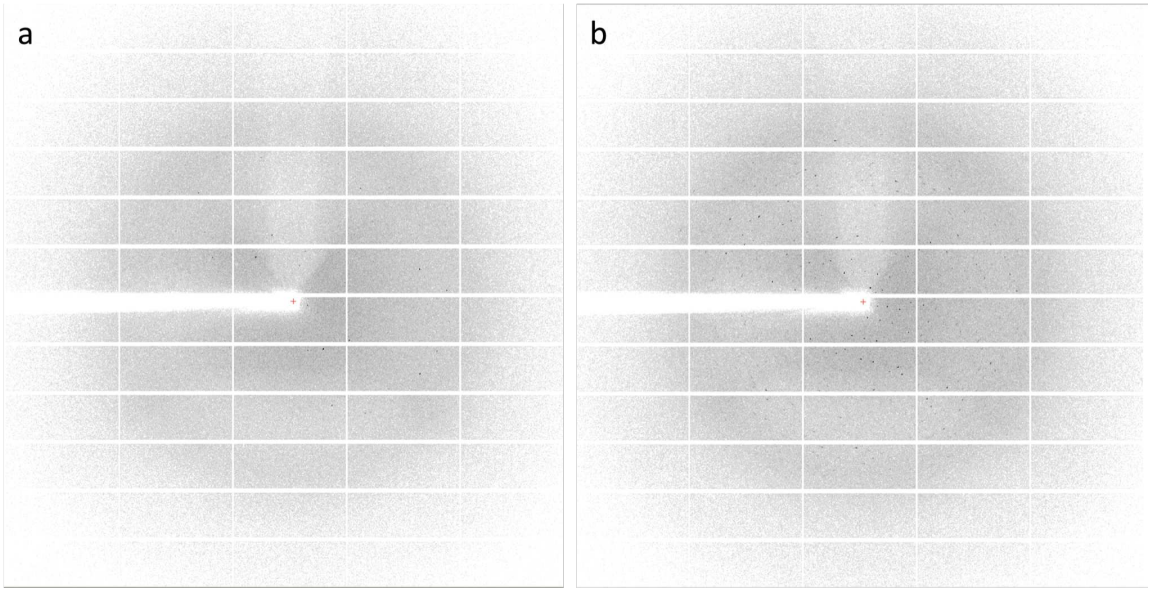

Figure 24. Frames from APS

Two frames from APS August 2016 run 124 (sample PK) that both appear to be hits but were not detected with initial hit finding parameters. (a) event 586 and (b) event 1082.

ADC	SNR	Max-Res	Mask	File ID																												
13	5	1200		181	208	272	275	288	289	306	383	432	456	470	482	528	534	544	546	547	550	551	552	563	581	598	609	611	620	638	665	672
13	5	650		181	208	272		288	289		383			470		528		544	546	547	550		552			598	609	611	620	638	665	672
13	5	1200	Yes	181	208	272	275	288	289	306	383	432	456	470		528	534	544	546	547	550		552			598	609	611	620	638	665	672
15	3	1200		181	208			288			383								546	547	550								620	638		672
15	5	1200		181	208			288			383								546	547	550								620	638		672
20	5	1200						288			383								546										620			672
50	5	1200									383																					
ADC	SNR	Max-Res	Mask	File ID																												
13	5	1200		675	706	707	710	716	718	742	747	748	765	818	828	842	898	904	905	906	908	922	926	927	933	945	957	966	1049	1077	1081	1082
13	5	650		675	706	707	710	716		742	747	748	765		828	842	898	904	905	906	908	922	926		933	945	957	966			1081	1082
13	5	1200	Yes	675	706	707	710	716	718	742	747	748	765		828	842	898	904	905	906	908	922	926		933	945	957	966	1049	1077	1081	1082
15	3	1200		675	706	707	710									842	898		905	906						945	957				1081	1082
15	5	1200		675	706	707	710									842	898		905							945	957				1081	1082
20	5	1200				707													905								957				1081	1082
50	5	1200																														

Figure 25. Peak finding parameters affect on hits

The y axis shows 7 hit finding conditions (arranged by number of hits with the most at the top), repeated twice to reduce image width. The first four columns give the values for the hitfinderADC, hitfinderMinSNR, hitfinderMaxRes, and peakmask with a yes indicating that a peakmask was used and empty space indicating no peakmask was used. The x axis shows file numbers for files identified as hits for that hit finding condition. Green shading indicates the hit was indexable. Red shading indicates that indexing was terminated for that hit because the process was taking too long. This data is from APS August 2016, run 124 (Sample is Proteinase K).

From fig. 25, it is evident that for some hits such as 288, 707, and 383 the success of indexing depends on the peak finding parameters, since the hit is identified multiple times but not always indexable. Peak finding influence on indexing solutions is the motivation for the software *IOTA* (Lyubimov, Uervirojnangkoorn, Zeldin, Brewster, et al. 2016) that optimizes peak finding parameters per pattern. The affect of peak finding on indexing is revisited in section 4.2.2. Figure 54 from that section shows the result of indexing for different peak finding parameters for particular frames.

2.2 Cheetah CBF

As mentioned in the chapter introduction, *Cheetah*'s three main functions are 1) applying detector corrections 2) determining hits through peak finding, and 3) saving the hits into HDF5/CXI format. The first task was described in section 1.5 and the second described above (section 2.1). The third task of saving output in HDF5/CXI format is important for several reasons. First, as a data reduction software, creating files containing just the hits limits the amount of data copied from the beam line to the home institution. Smaller input files to later steps also reduce computation time.

However, another important aspect is having a standard format (HDF5/CXI) that can be read by many programs. Raw files saved at different beam lines have different formats. For instance, LCLS files are saved in XTC format. Reading XTC files away from LCLS can require installing *psana* (Damiani et al. 2016) on other machines which is not trivial. The EXFEL and SACLA store output in HDF5/CXI format initially, and APS stores files in the CBF file format.

For APS August 2016, it was necessary to extend *Cheetah* to read CBF files. In part, this was to enable peak finding, but the more important reason is to convert the

smaller set of hits to HDF5/CXI to allow later programs like *indexamajig* to use them. At the time, the other alternative was to convert all CBF files to HDF5 and then use the built-in *zaef* peak finding algorithm of *indexamajig*. However, in version 0.7.0 or higher of *CrystFEL*, *indexamajig* can directly read CBF files and run *peakfinder8* (White 2019).

Using *indexamajig* with CBF files is preferable to *cheetah-cbf* for several reasons. First, *cheetah-cbf* with the computational resources available at APS was unable to keep up with the rate of data collection so all "raw" CBF files had to be copied. Running *cheetah-cbf* at a home institution just duplicates hits into another format. Second, *cheetah-cbf* has the overhead of applying detector corrections which aren't necessary for the detectors used at synchrotrons. Third, *Cheetah* itself is not particularly stable as a software. Stability in software means that software updates rarely cause errors and that newer versions of the software work with older input files (backwards-compatibility). However, *Cheetah* is often edited during beam times to solve particular problems, sometimes breaking existing functionality. Also, *Cheetah*'s ability to read beam line output files makes it dependent on beam line software that is also unstable. In short, this means that running the current version of *Cheetah* on an old dataset may not necessarily work, and since *cheetah-cbf* is not frequently used, the current version is unlikely to work without modifications to the code.

2.3 Cheetah Parallel

Another extension to *Cheetah* improved the processing time. In order to improve the processing time for *Cheetah*, it's first necessary to identify the most time-consuming step, referred to as the bottleneck. The speed of software can be related to the hardware

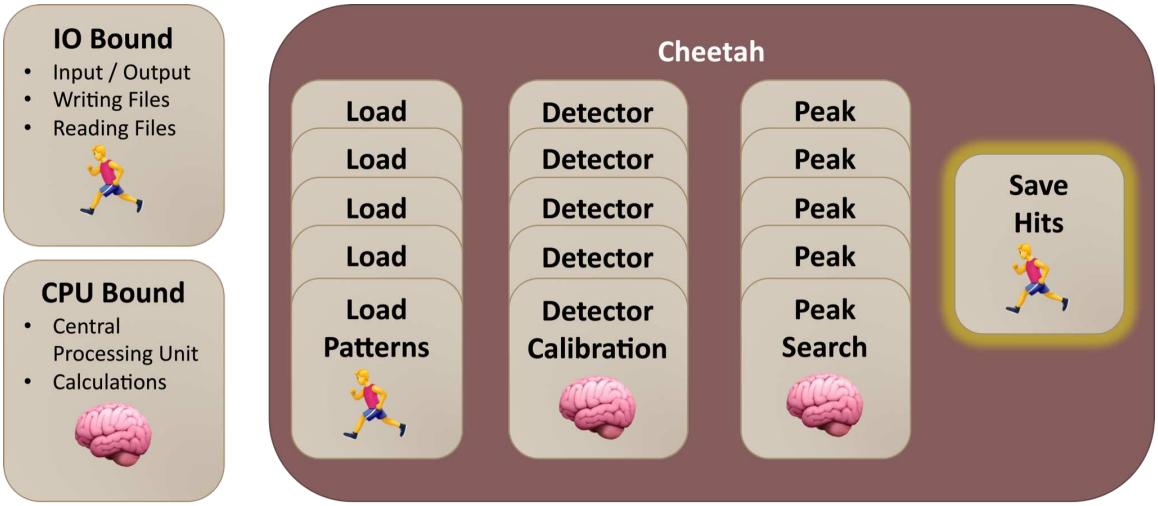

Figure 26. Cheetah program steps

The bottleneck in programs is usually IO bound, meaning limited by the read/write speeds to input/output files or CPU bound, limited by the number of calculations. *Cheetah* is a threaded program, so each thread can load a pattern, apply corrections, and search for peaks. However, there is only a single output file for storing hits, so the program's speed is limited by the write speed of the found hits.

it relies on the most. IO (input/output) bound processes do a lot of file reading and writing and so the speed is related to the hard drive. CPU (central processing unit) bound processes do a lot of calculations and are limited by the number of threads and cores.

2.3.1 Cheetah Structure

An overview of basic steps in *Cheetah* is shown in fig. 26. *Cheetah* is a threaded process, with each thread able to load a pattern (IO), apply detector corrections (CPU), and locate Bragg peaks (CPU). With *Cheetah* already a threaded process, and 80 cores per node available at the EXFEL (meaning at least 80 threads), the

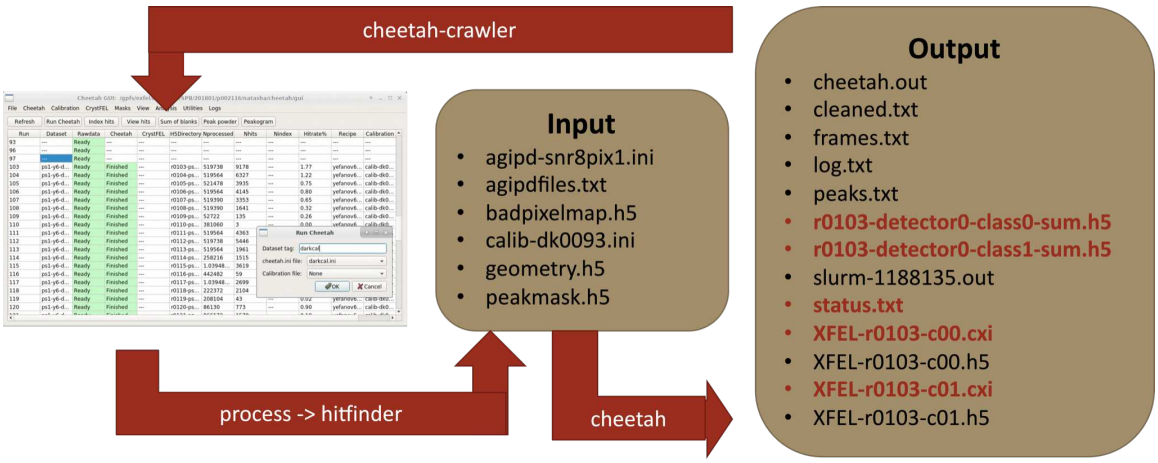

Figure 27. Cheetah GUI workflow

The *Cheetah* GUI generates a folder for each unique run-tag identifier, populates it with the appropriate input files, and launches cheetah. As *Cheetah* runs, it updates output files in the directory. Files in red font are used in the GUI. Status.txt is used to update the table values. The CXI files store the hits and are used to display hits through the GUI. The detector sum H5 files are viewable through the powder sum menu.

CPU is unlikely to be the problem. Rather, the problem is likely IO limited, both by the single output file that all 80 threads must use and by the large amount of data that a single instance of *Cheetah* is processing. For run 103 of EXFEL August 2018, 2.1 terabytes of raw data (519,783 frames collected in approximately 7 minutes) were recorded which took 4 hours and 55 minutes for *Cheetah* to process.

A straightforward fix to IO bound problems like the one above is to launch multiple instances of *Cheetah*, which breaks down the terabytes of input data into smaller chunks and increases the number of writing threads. However, the *Cheetah* GUI is tied to output grouped by run, and even if it wasn't, statistics are typically calculated and presented by run as well. So, a modification to run multiple instances of *Cheetah* for a single run must sync the output together to work seamlessly with the GUI.

The initial setup relating the *Cheetah* GUI to output is shown in fig. 27. From the GUI, the user selects runs to be processed, the name for the output folder, and the ini files for *Cheetah* to use. This information is passed through a series of scripts that creates an output directory, copies all needed input files into it, and then submits a *Cheetah* job to the computational cluster queuing system. Another script called *cheetah-crawler* is managed by the GUI. Every minute it scans all the directories and updates the information shown in the main *Cheetah* GUI table. Specifically, the *cheetah-crawler* uses status.txt to update the table with the current number of processed events and found hits and the GUI uses the CXI files to display the frames (from the "View Hits" button) and the H5 detector sums for the display of the "Sum of blanks" and "Peak powder" buttons.

2.3.2 Cheetah Parallel Workflow

To launch multiple instances of *Cheetah* for a single run therefore requires at minimum a script that will provide appropriate status.txt, CXI, and powder sum files in the directory searched by *cheetah-crawler*. The setup with the parallelization script is shown in fig. 28. *Cheetah* parallel replaces the call to the batch queue with a call to a new script *runparallel.py*. This script creates subfolders for each process of *Cheetah*. It was initially developed at LCLS where it creates a subfolder for each XTC file stream. (It became apparent that the first XTC file in a stream contains information that later files do not so *Cheetah* cannot be run for each XTC file individually.) The script was adapted for the EXFEL where it can launch an instance of *Cheetah* for every raw H5 file.

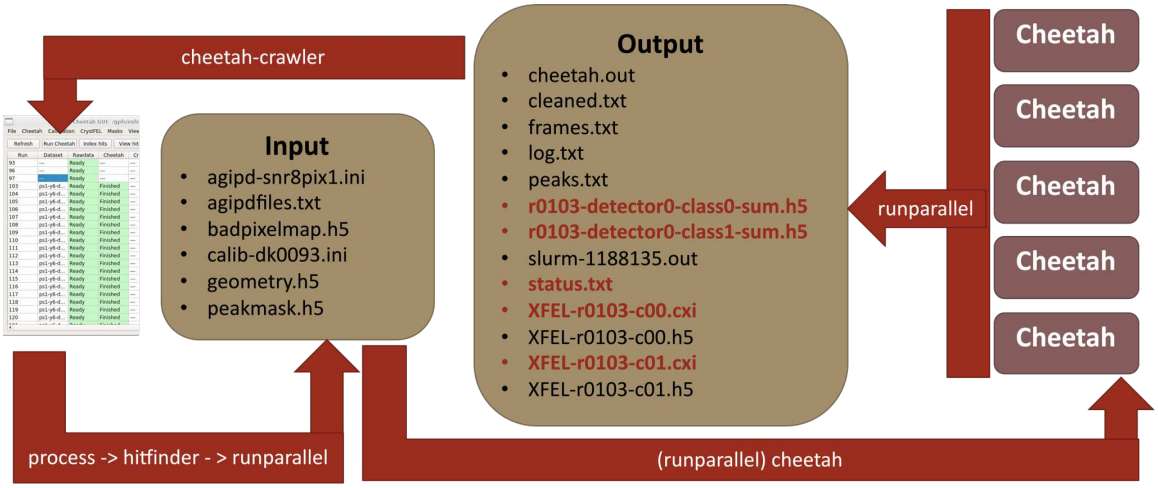

Figure 28. Cheetah GUI Parallelized Workflow

The script *runparallel* creates subdirectories and launches multiple *Cheetah* processes, one per folder. It then monitors those processes to sync the results into the output directory so the GUI updates normally.

The script then switches into a crawler mode, similar to *cheetah-crawler* where every one minute it syncs the output from the subdirectories into the top directory for the *cheetah-crawler* to find. An overview of the output in the output folders is shown in fig. 29. The CXI files with frames are the easiest to sync. A symbolic link to each file is created with a unique name based on the input H5 file at the EXFEL (which naming scheme follows S00000 S00001 and so on). It turns out that the GUI is already able to handle multiple output files for a single run because the output format for events has changed several times.

2.3.3 Cheetah Output Files

Initially *Cheetah* stored each hit in its own HDF5 file. In LCLS October 2015, default *Cheetah* output changed to record a single CXI file for each class (hits and non hits by laser scheme for the PSII experiments in this dissertation) in a run. Then,

when CXI files became too large, a limit on the number of events in a CXI file was created so data in LCLS November 2016 has multiple files per class identified with c00, c01, and so on in the name. To complete the current story of updates, at LCLS September 2017, CXI files were split into a CXI file and H5 file of the same name where the CXI file contained the full frames and the frequently used metadata and the H5 file contained the majority of metadata but no raw frames. That optimization was supposed to improve speeds for programs that didn't need the full file, but in terms of stability the H5 files were initially one event short of the CXI files so metadata for one event per each file is missing in LCLS September 2017. Aside from explaining why the *Cheetah* GUI automatically handles multiple output files, this history is also relevant for the development of new softwares such as *DatView* presented in chapter 4. Enabling *DatView* to read metadata from any of the possible locations above (and avoid crashing when it asked for metadata only in the H5 file for missing events in LCLS September 2017) and to display frames from all output formats required understanding the many possible *Cheetah* output formats.

Returning to the syncing of *Cheetah* output files, another important file is status.txt. The contents of a typical status.txt file are:

```
# Cheetah status
Update time: Mon Aug 27 08:46:25 2018
Elapsed time: 4hr 55min 15sec
Status: Finished
Frames processed: 519738
Number of hits: 13942
```

Updating the file is a matter of parsing the status.txt files in all subdirectories, summing the frames processed and number of hits, updating the times with the times of the *runparallel* script, and setting the status to "Finished" or "Not Finished" as

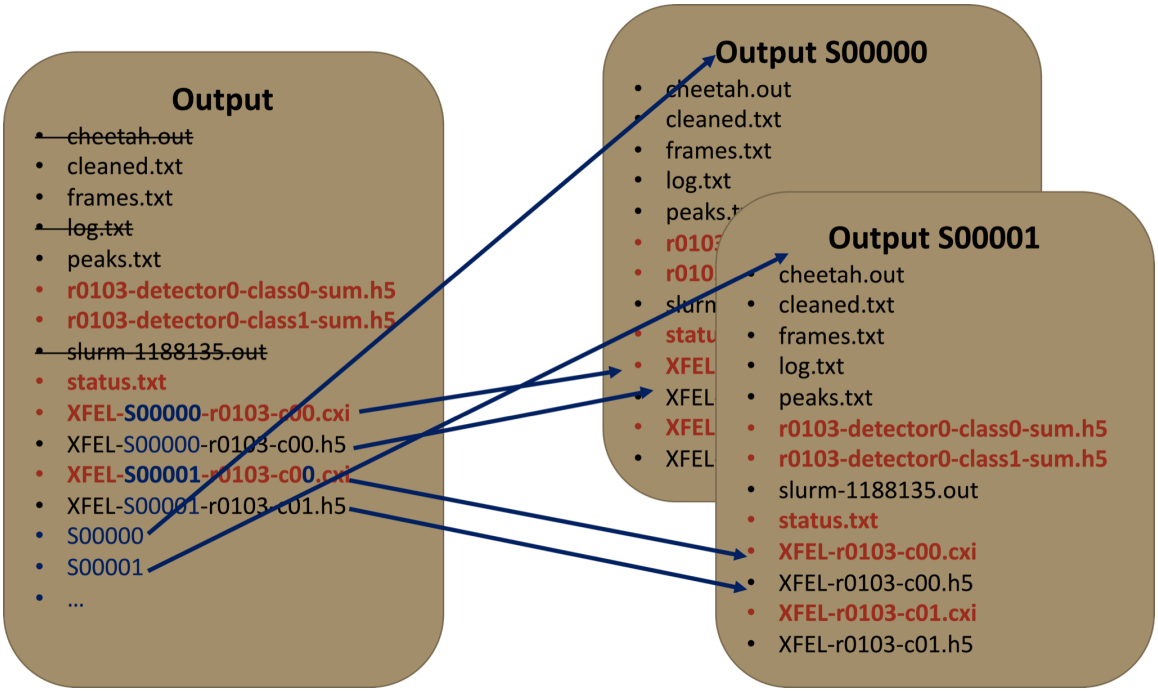

Figure 29. Cheetah output synchronization

An overview of the new output structure with parallelized *Cheetah*.

appropriate. While this updates the GUI, it unfortunately doesn't work with most versions of scripts like *hits* that gives the same information presented in the GUI on the command line. This is because those scripts use frames.txt or log.txt. Frames.txt, cleaned.txt, and peaks.txt are all tables that in general can be concatenated (skipping the header) from the files in the subdirectories with one caveat. Some of them reference output file names, which are identical in the different subdirectories. So, *runparallel* first substitutes the file names with the symbolic linked names and then concatenates those files. Log.txt is not currently synced because 1) the *hits* script will work correctly in slow mode with *frames.txt*, 2) updates to the hits script that use *status.txt* have been developed, and 3) the log.txt file does not have an easily reproducible format. Similarly, neither cheetah.out nor batch output such as slurm-*.out or bjobs.out are

synced into the top directory as they are typically a log of what a particular *Cheetah* instance wrote to the command line.

The only output files that have not been described yet are the detector sums. The detector sum files are the least robust (most likely to break) part of *runparallel* because the script needs to be modified if the number of output classes or the types of sums being saved are changed. A detailed description of the combination of powder sums is given in appendix C.

2.3.4 Using Cheetah Parallel

There are several things users of parallel *Cheetah* must know. First, the *hitfinder* script for *runparallel* chooses whether to use parallelization or not based on the .ini file name. If the ini file name contains "fast", then parallel mode is used. This design was chosen because some *Cheetah* processes launched from the GUI should not be parallelized such as dark calibrations. Also, very small runs will run faster in a single instance of *Cheetah* because there is overhead with creating multiple processes and wait time for the processes to be launched on the queue.

Second, all 'raw' files should be finished copying before *Cheetah* parallel is launched. At LCLS, *Cheetah* can be launched while data is still being collected because of the way files are read. However, *runparallel* gathers the list of files to use once at the beginning of the script and never again, so all files to be processed must be present.

Third, to avoid a lot of duplication, *runparallel* creates symbolic links to CXI output files. This means that scripts that naively search the *Cheetah* output directory for any CXI files are going to process each file twice: once for the original file and once for the symbolic link. The fix for this is to set the max-depth parameter of the

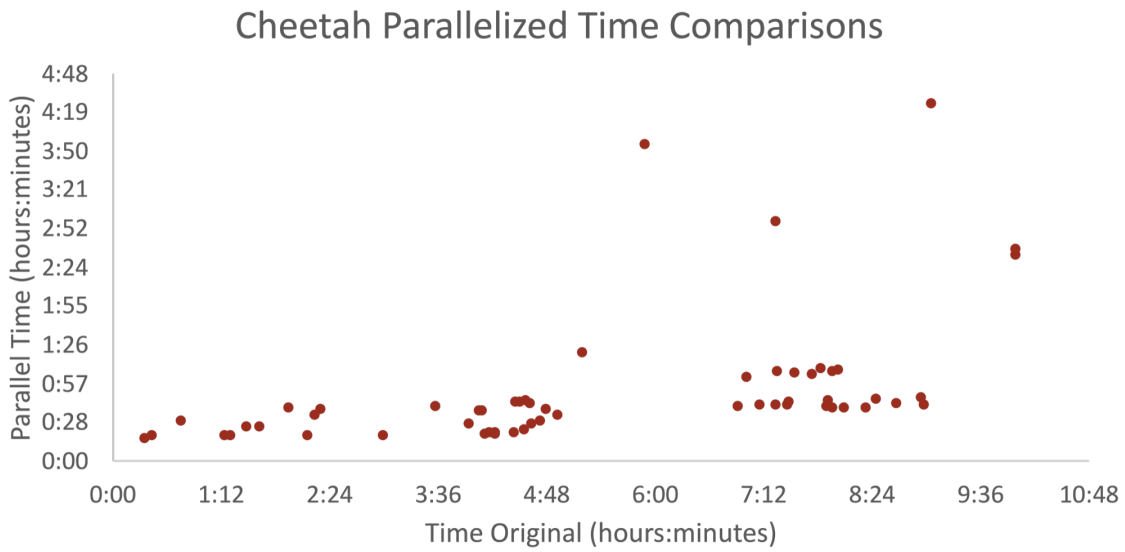

Figure 30. Time improvements for parallelized Cheetah

Processing times for runs from EXFEL August 2018 that were processed normally (x axis) and in parallelized mode (y axis). The outliers that take longer than an hour under parallel mode can come from bad input files that never finish processing and from some *Cheetah* processes starting later than others depending on job allocation for the computational cluster.

find command to only search the top level directories. Also, analysts should be aware when copying the data that copy programs may require additional flags to copy the symbolic links.

However, using *runparallel* improves the speed of experimental feedback. Figure 30 shows the processing times for *Cheetah* in normal (x axis) and parallel (y axis) modes for runs processed both ways at EXFEL August 2018. The exact times of *runparallel* will depend on the computational resources available. If the job queue is busy, then parallelizing *Cheetah* adds to the number of requested nodes and may be slower. In general, though, *runparallel* is expected to be very useful as a speed improvement during and after beam times.

Chapter 3

INDEXING

The main output of indexing is a list of reflections by Miller index (HKL) with intensity and sigma values. The h, k, and l axes of the Miller index correspond to the a, b, and c axes in real space. In the process of assigning parts of the image to a Miller index, indexing also determines the orientation of the unit cell and the unit cell parameters: the a, b, and c axes and α, β, and γ angles. In this dissertation, the term peaks is used for peaks found with peak finding (see section 2.1) and reflections refer to peaks predicted by indexing.

Most indexing algorithms were developed for traditional crystallography (rotation series of patterns). *CrystFEL*'s *indexamajig* (White et al. 2012) is actually a wrapper for several traditional indexing algorithms, running the algorithm on snapshots and converting the results into a *CrystFEL* stream file format. The common indexing algorithms used with *CrystFEL* in this dissertation are *MOSFLM* (Leslie 2006), *Dirax* (Duisenberg 1992), and *XDS* (Kabsch 2010). *CrystFEL* also has some internal indexing algorithms the most common of which are *asdf* and the recently developed *xgandalf*. Other programs independent of *CrystFEL* also exist for analyzing snapshot patterns such as *cctbx.xfel* (Hattne et al. 2014), *nXDS* (Kabsch 2014), and *SPIND* (Li et al. 2019).

This chapter uses *CrystFEL's indexamajig* as *CrystFEL* is used for all data processing presented in the dissertation. As with most tools, the first question facing users is whether the resulting output is reliable. The first section (3.1) therefore covers *indexamajig* output. A more complete review of visualization tools for *indexamajig*

output is given in chapter 4. The second section (3.2) continues the theme of assessing output quality by describing some common merging (see chapter 5) statistics used to compare indexing results. The final section (3.3) describes indexing optimization for LCLS June 2012 PSII data based on merging statistics, also describing input parameters to *indexamajig* as they are used.

3.1 Indexing Output

The output of *indexamajig* is a *CrystFEL* stream file (.stream). The stream file is a plain text document. This means the full contents can be viewed with command line tools like *less* and the document can be searched with command line tools like *grep*. The document begins by specifying the file format, the version of *CrystFEL*, and the exact command to generate the file. Copies of the geometry (see section 3.3.1) and provided unit cell information are included next. The remainder of the document is split up into chunks (one chunk per frame) and crystals (one crystal per crystal lattice). *Indexamajig* has the option to find multiple crystal lattices on a single frame so a single chunk may have 0, 1 or several crystals.

The output included in the stream file depends on the steps of *indexamajig* used. The steps in *indexamajig* are 1) peak finding if previous peak finding results are not used, 2) indexing to locate reflections and 3) integration to determine the intensity at each reflection. The list of peaks from peak finding or previous results is included in each chunk (per file). Indexing gives statistics such as the unit cell (section 3.1.1), diffraction resolution limit (section 3.1.2), and profile radius (section 3.1.3) for each crystal lattice. Integration gives the intensities at each reflection and overall statistics such as the number of implausibly negative reflections (section 3.1.4).

3.1.1 Unit Cell

The unit cell information of a crystal includes the six values for the a, b and c axes and α, β, and γ angles, the lattice type, the centering, and the orientation matrix. In the stream file, the information looks like:

```
Cell parameters 14.12365 22.96616 30.95583 nm, 92.22021\
91.64719 89.50883 deg
astar = +0.0003493 +0.0521086 +0.0479801 nm^-1
bstar = -0.0095544 -0.0290028 +0.0310876 nm^-1
cstar = +0.0312478 -0.0050616 +0.0066274 nm^-1
lattice_type = orthorhombic
centering = P
unique_axis = *
```

Several tools exist that parse this information from the stream file and display it. The most common is *CrystFEL's cell_explorer* (fig. 47) which displays histograms of the six cell parameters stacked by centering. *Cell_explorer* is described in more detail in section 4.1. A script for the orientation matrix was introduced with fig. 15 in section 1.4.2. Figure 31 is generated from *DatView* (see chapter 4). *DatView* calculates the unit cell volume with the formula from (Jeffrey 2006):

$$V = a * b * c * \sqrt{1 - \cos(\alpha)^2 - \cos(\beta)^2 - \cos(\gamma)^2 + 2 * \cos(\alpha) * \cos(\beta) * \cos(\gamma)} \quad (3.1)$$

Consistent unit cells with a single peak are a sign that indexing worked correctly. A double peak (like LCLS January 2012), multiple peaks (like LCLS October 2015) or broad peaks (like LCLS November 2016 and LCLS September 2017) can be signs that indexing parameters (usually the geometry file) are not correct. However, since fig. 31 is displaying the summary of an experiment, the observed peaks are related to displaying multiple samples at the same time and the unit cell oscillations discussed in section 1.4.2.

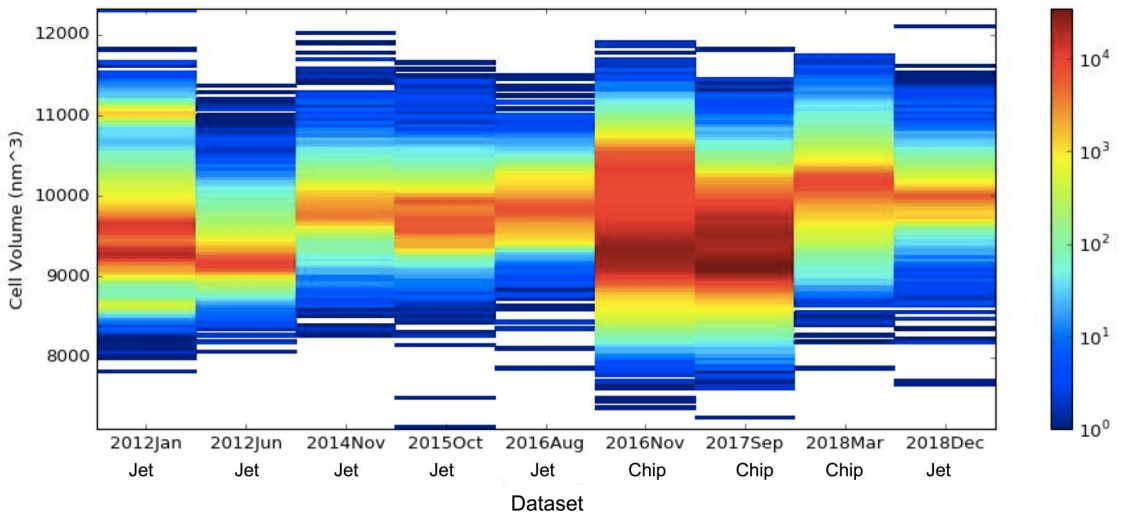

Figure 31. Photosystem II cell volume

The unit cell volume by LCLS experiment for PSII datasets (*CrystFEL* 0.7.0 all PSII indexing, see A.1). This figure displays all PSII data from each experiment so multiple peaks can come from different runs or laser schemes, and the broad distributions from unit cell oscillations described in section 1.4.2.

3.1.2 Resolution Limit

The diffraction resolution limit calculated per pattern by *CrystFEL* is stored in the stream file as:

diffraction_resolution_limit = 0.79 nm^−1 or 12.73 A

It depends on the peaks found, so resolution cutoffs during peak finding can impact the diffraction resolution limit. The diffraction resolution limit is not an indication of indexing quality, but is used during an experiment as an indication of sample quality. A comparison of diffraction resolution limits by LCLS PSII experiment is given in fig. 32.

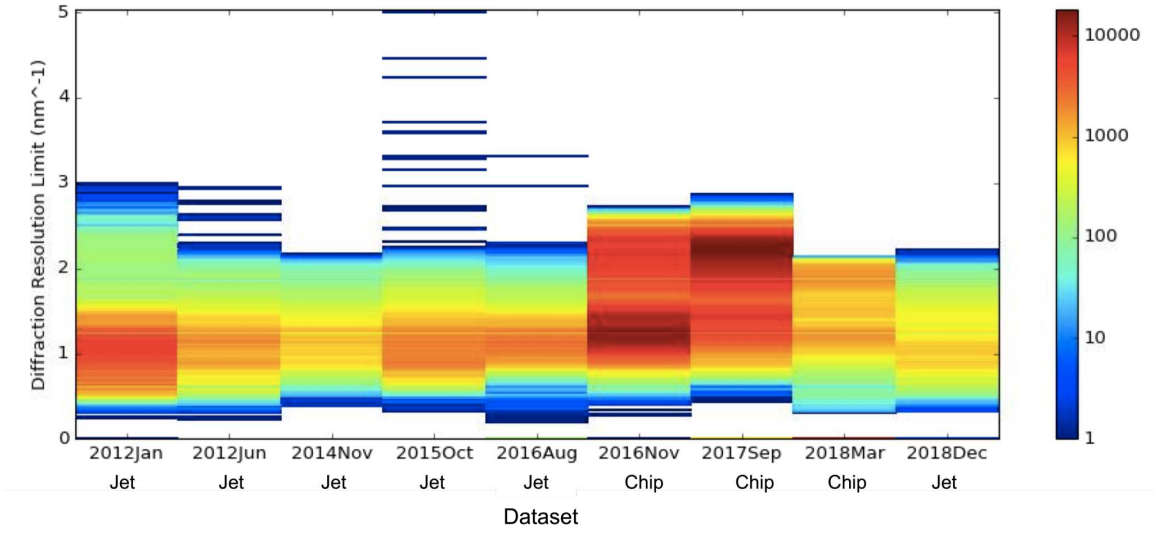

Figure 32. Photosystem II diffraction resolution limit

The diffraction resolution limit (nm^{-1})by LCLS experiment for PSII datasets (*CrystFEL* 0.7.0 all PSII indexing, see A.1). The diffraction resolution limit depends on peak finding, so the few high resolution patterns from LCLS October 2015 are from noise at high resolutions, not a sudden change in diffraction quality of the PSII crystals.

Units for the diffraction resolution limit can be nm^{-1} or Å. For most users, Å is more intuitive. However, nm^{-1} is almost always used in software. First, no resolution limit has a value of 0 nm^{-1} which corresponds to infinity in Å. Infinite values are harder to deal with computationally than 0. Second, a plot with nm^{-1} as the axis is a better representation of the spacing between resolution shells than one using Å. In this dissertation, nm^{-1} is generally used. A few helpful conversions are 1 nm^{-1} is 10 Å, 2 nm^{-1} is 5 Å, 2.5 nm^{-1} is 4 Å, and 3.33 nm^{-1} is 3 Å. The PSII datasets used in this dissertation are often around 2 nm^{-1} (5 Å), with the exception of the fixed target chip datasets LCLS November 2016 and LCLS September 2017.

3.1.3 Profile Radius

The profile radius appears in the stream file as:

profile_radius = 0.02000 nm^−1

As an output parameter, it is a measure of indexing accuracy. Like the diffraction resolution limit, it depends on the peaks found. It can be read as the distance between peak center and corresponding reflection center for which 99% of peak-reflection distances on a pattern are less than or equal to. It can also be set to a particular value as part of input to *indexamajig* in which case the value will always be the same and is therefore not a useful indication of per-pattern indexing accuracy. It's influence as an input parameter is discussed in section 3.3.4. The profile radius for LCLS PSII datasets is shown in fig. 33 (it was not fixed as an input parameter for the data in the figure).

3.1.4 Implausibly Negative Reflections

There are several statistics related to reflections in the stream file:

num_reflections = 15549
num_saturated_reflections = 0
num_implausible_reflections = 39

The number of saturated reflections is related to saturated pixels discussed in section 1.5.3. Saturated reflections have not been an important factor in the analysis of PSII data. However, the number of implausibly negative reflections has been used in this dissertation for pattern selection (see section 7.4). In part, the parameter is noticeable because non-zero values are written as warnings in the command line while

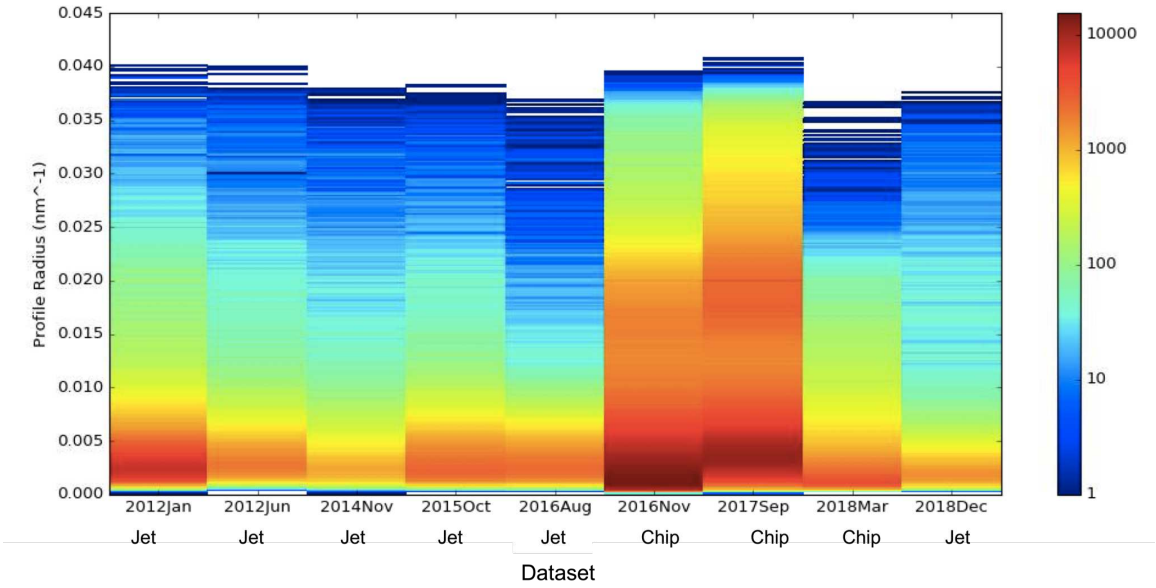

Figure 33. Photosystem II Profile Radius

The profile radius by LCLS experiment for PSII datasets (*CrystFEL* 0.7.0 all PSII indexing, see A.1). Coloring is on a log scale.

indexamajig is running. It depends on integration parameters. A comparison across LCLS PSII datasets is shown in fig. 34.

3.2 Merging Statistics

Merging reduces a stream file to an hkl file which contains a single, merged intensity value for each Miller index. With merged data, the overall completeness and signal to noise ratio (SNR) of the dataset can be calculated. Merges are also typically run on half datasets, with statistics such as R_{split}, $CC_{1/2}$ and CC^* comparing the two halves. Half datasets are usually created by merging the odd and even numbered patterns in the file separately, and the merging statistics are described in detail below. Calculation of statistics in *CrystFEL* is done with *check_hkl* for the full merge and

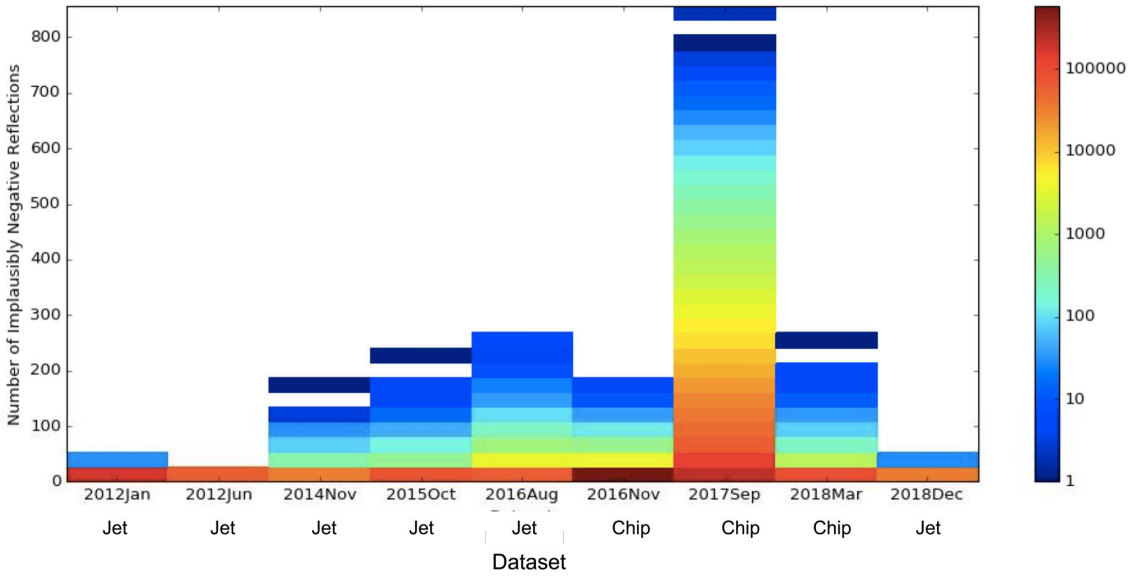

Figure 34. Photosystem II implausibly negative reflections

The number of implausibly negative reflections by LCLS experiment for PSII datasets (*CrystFEL* 0.7.0 all PSII indexing, see A.1). Coloring is on a log scale. Note that the purpose of the *CrystFEL* 0.7.0 all PSII indexing was to compare the different datasets with the same *CrystFEL* version and parameters. The large number of implausibly negative reflections in LCLS September 2017 comes because the indexing parameters are not optimized for that dataset.

compare_hkl for half merges. Output is computed in resolution shells and an overall value is printed to the command line.

The *CrystFEL* script *stream2stats* is a wrapper for merging and generating the completeness, SNR, R_{split}, $CC_{1/2}$ and CC^*. It plots several statistics on a single plot, as shown in fig. 35. The completeness of the data (blue line in the figure) is important for the reliability of the other statistics. Statistics from incomplete resolution shells are not as reliable. R factors are reviewed and the statistics $CC_{1/2}$ and CC^* introduced in (Karplus and Diederichs 2012), with a follow up comparing statistics in (Karplus and Diederichs 2015). The R factor for merging is defined (Karplus and Diederichs

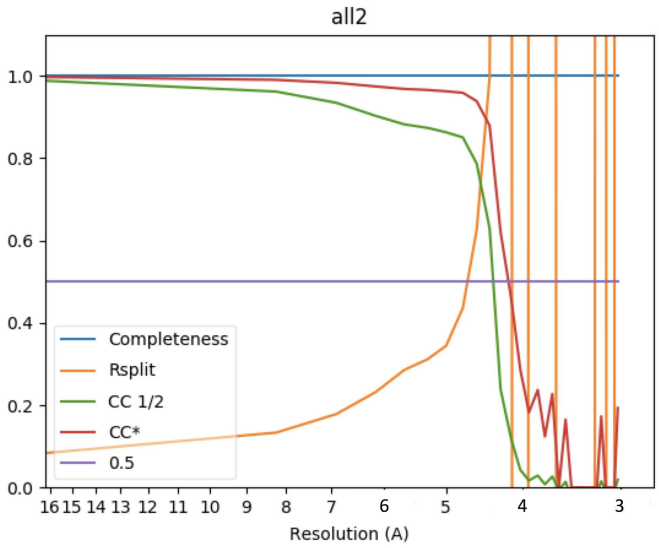

Figure 35. Stream2Stats plot

The script *stream2stats* creates merges, calculates several statistics, and generates the plot shown in the figure. The R_{split} (orange line) has almost vertical lines at higher resolutions because shells without sufficient data are 0, making the line jump between high values and 0. One measure of the resolution is the point where CC^* is 0.5, so for this data the resolution cutoff would be a little above 4 Å. This plot was generated from June 2012 light data initial with-cell indexing (indexing condition 0 in table 7.

2012) as:

$$R_{merge} = \frac{\sum_{hkl} \sum_{i=1}^{n} |I_i(hkl) - \overline{I}(hkl)|}{\sum_{hkl} \sum_{i=1}^{n} I_i(hkl)} \tag{3.2}$$

Variations on the R factor arose with different correction factors. R_{split} in snapshot crystallography is defined as (White et al. 2013):

$$R_{split} = \frac{1}{2^{1/2}} \frac{\sum |I_{even} - I_{odd}|}{\frac{1}{2} \sum (I_{even} + I_{odd})} \tag{3.3}$$

Lower values of R are desirable, but R values in general are not recommended as a basis for resolution cutoffs (Karplus and Diederichs 2015). Rather, Pearson's correlation coefficient between half datasets $CC_{1/2}$ is recommended, defined as:

$$CC_{1/2} = \frac{\sigma_\tau^2}{\sigma_\tau^2 + \sigma_\epsilon^2} = \frac{< I^2 > - < I >^2}{< I^2 > - < I >^2 + \sigma_\epsilon^2} \tag{3.4}$$

with σ_ϵ the mean error in a half dataset. CC^* is an estimate of the CC with the "true" values based on $CC_{1/2}$ defined as (Karplus and Diederichs 2012):

$$CC^* = \sqrt{\frac{2CC_{1/2}}{1 + CC_{1/2}}} \tag{3.5}$$

Exact resolution cutoffs are still debated, but $CC_{1/2}$ values around 0.14 from statistics (corresponding to $CC^* = 0.5$) are generally accepted values. The *stream2stats* script outputs a horizontal line at 0.5 for convenience, so a resolution estimate from fig. 35 would be about 4 Å.

3.3 Indexing Optimization

Before describing indexing optimization of the PSII data from LCLS June 2012, it's important to consider which data is being used. In an ideal case, hit finding is optimized during the experiment and a single set of hits copied to the home institution for indexing optimization. Unfortunately, some experiments, including LCLS June 2012, have multiple hit finding results at the home institution. While the hit finding parameters are stored with the results, analysts must still decide which data to analyze.

A shortened summary of hits by hit finding tag for runs with alternating light and dark data is shown in table 3. There are other hit finding tags for smaller subsets of runs, or hit finding tags without light and dark list files. Even for the common tags in the table, some runs are missing light and/or dark list files. Also, while data in this table are consistent, it's not uncommon to find that the numbers of actual files don't match the numbers in output files like status.txt, or that the total number of events in a run is different for different hit finding tags. While all data can be indexed, different hit finding tags for the same run(s) can't be directly merged together because

some frames will be duplicated (because they were identified as hits under multiple conditions).

For LCLS June 2012, bolded values in table 3 indicate which light and dark files were used for the analysis, with the choice usually dependent on which tag had the larger number of hits. Hit finding tag 'a' was used for runs 33-35, 43, 48, 49, 86, and 89-91. Hit finding tag 'b' was used for 82-83, 85, and 87. Hit finding tag 'cht' was used for run 88, with light.txt reconstructed by subtracting dark.txt from cleaned.txt (with the assumption that the empty light.txt was not intentional). Initial indexing and optimizations were done on the light data.

The initial indexing command was:

```
indexamajig −i split.lst00 −o split.stream00 −j 8\
−g [...]/jun12_140506_yef2.geom\
−−peaks=hdf5 −−int−radius=3,4,6\
−−indexing=mosflm−nocell−nolatt −−integration=rings−sat
```

Where -i and -o are for the input and output, -j for the threads, -g for a geometry file (described in section 3.3.1), –peaks for the source of peaks (in this case from hit finding as stored in the hdf5 input files), –int-radius for the integration radius (described in section 3.3.3), –indexing for the indexing algorithms to use (in this case *MOSFLM* without prior cell or lattice information) and –integration for the integration method (described in section 3.3.2).

A second iteration of indexing used prior unit cell information (–pdb to *indexamajig*) with the unit cell file:

```
CrystFEL unit cell file version 1.0

lattice_type = orthorhombic
centering = P
```

Table 3. Hits by hit finding tag

Tag	F	33	34	35	43	48	49	82	83	85	86	87	88	89	90	91	92
	L	**224**	**1003**	**644**	**1857**	**49**	**214**	0	1326	2737	**98**	6459		**100**	**3555**	**3594**	
	D	**268**	**967**	**708**	**1789**	**39**	**211**	1098	1420	2419	**99**	6429		**93**	**3538**	**3592**	
a	U	0	0	1	1	0	0	0	0	508	0	5		0	0	0	456
	K	493	1971	1354	3648	89	426	2057	2747	5672	198	12894	17718	194	7094	7187	457
	F	492	1970	1353	3647	88	425	2056	2744	5668	197	12893	17717	193	7093	7186	456
	L	213	940	612	1710	42	186	**887**	**1851**	**2763**	76	**8863**					
	D	257	913	666	1651	36	198	**1004**	**1925**	**2865**	87	**8936**					
b	U	0	0	1	1	0	0	0	0	1	0	8					
	C	471	1854	1280	3363	79	385	1892	3777	5630	164	17808	15491	167	6105	6183	
	F	470	1853	1279	3362	78	384	1891	3776	5629	163	17807	15490	166	6104	6182	
	L	213	933	0	1699	42	186	653	1842	2747		8812	**6056**		3043		0
	D	256	906	250	1642	36	196	731	1921	2848		8878	**0**		3031		0
cht	U	0	0	0	1	0	0	0	0	1		8	2453		0		402
	C	470	1840	251	3343	79	383	1385	3764	5597	164	17699	14669	168	6075	2811	403
	F	469	1839	256	3342	78	382	1384	3763	5596	163	17698	14668	167	6074	2848	402

Number of hits listed in each file (F column, L for light.txt, D for dark.txt, U for unk.txt and C for cleaned.txt), and total number of files (F row of F column) found in directory for alternating light/dark runs (columns) from June 2012. These are only the three hit finding tags that appeared for most runs. Hit finding tags are arbitrary labels given by the initial analyst. Note that cleaned.txt has a header line and so is expected to be 1 larger than the actual number of files. Light, dark, and unk (unknown) list files by laser scheme. Bolded values indicate files used in analysis. For run 88, dark was the cleaned list minus the light and unknown list.

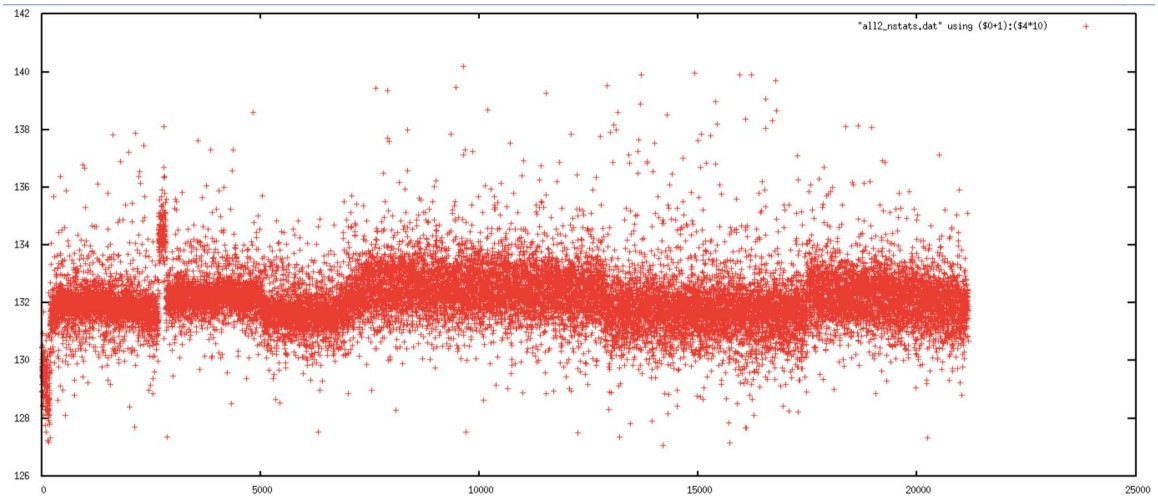

Figure 36. Cell a volume by sorted order

The unit cell *a* axis plotted by sorted file number, showing two samples that had distinct unit cell *a* axis values compared to the majority of the data. Those regions correspond to run 33 (low values) and runs 48 and 49 (high values).

a = 133.25 A
b = 226.26 A
c = 307.09 A
al = 90.00 deg
be = 90.00 deg
ga = 90.00 deg

A a plot of the unit cell *a* axis is shown in fig. 36. Based on this plot, runs 33, 48, 49, and 89 were excluded from further analysis because of their markedly different unit cell values.

3.3.1 Geometry Files

A geometry file converts the way data is stored into the way pixels are arranged in real space. Data is not usually stored as it appears in real space for several reasons. First, real space has empty places between panels that would increase the size of

the file if they were stored. Second, those empty places are not necessarily integer multiples of pixel size and so storing the empty spaces is less accurate. Finally, the accuracy of real space pixel locations can be improved and so it is better to keep the values in a separate file that can be easily updated.

Figure 37a shows how the data is stored in the file, with all panels adjacent to each other. The axes of data storage are referred to as fs and ss for fast-scan and slow-scan directions. In the fast-scan direction, adjacent pixels are stored next to each other in the file. In the slow-scan direction, adjacent pixels are offset by the length of the fast-scan direction. Opening the same file in *hdfsee* with geometry information displays the panels at their locations in real space (fig. 37b). The geometry file can specify a mask and bad regions in addition to whatever mask may have been applied to the raw data during hit finding. Geometry files also give other experiment information such as the camera length, photon energy, and pixel size.

For early SFX experiments, geometry optimization was done by manually moving panels or panel groups with *hdfsee* to align virtual powder rings. Several programs and algorithms have since been developed for geometry optimization (Yefanov et al. 2015; Ginn and Stuart 2017). This dissertation uses *geoptimiser* (Yefanov et al. 2015). In general, geometry optimization moves the panels to minimize the distance between peaks and their corresponding reflections. *Geoptimiser* outputs a colored image showing average displacement by pixel of the detector before and after optimization. From fig. 38a,b it is evident that the geometry file with the data was already optimized. In contrast, the output from initial geometry optimization of LCLS January 2012 is shown in fig. 38c,d. Geometry optimization, since it depends on indexing results, can be iterative. Indexing is rerun with the new geometry file and the process is repeated while improvements are seen.

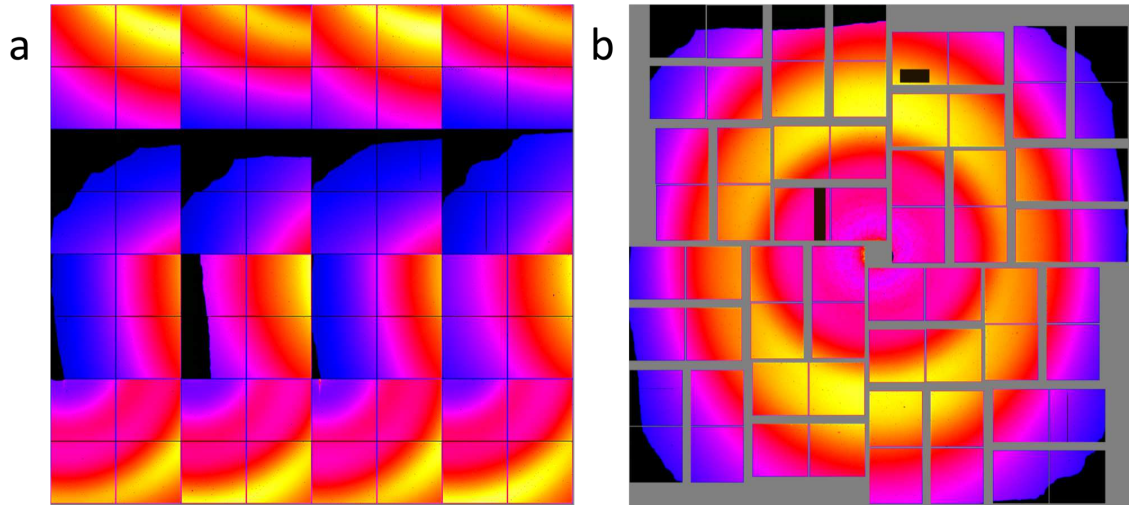

Figure 37. Geometry layout

A geometry file describes how to the array of data as stored (a) corresponds to the arrangement of points in real space (b). It can also include a mask or define bad regions to exclude areas from analysis. The images are the powder sums from run 80 of LCLS October 2015 opened in *hdfsee* without (a) and with (b) a geometry file.

Programs like *geoptimiser* are good for finding a local minimum from a reasonable starting geometry file. However, initial geometry files may still need to be manually edited before attempting *geoptimiser*. One relatively simple check that can have a big impact is the predicted detector shifts from *CrystFEL's indexamajig*. In the stream file, they appear for each crystal lattice as:

$$\text{predict\_refine/det\_shift x} = -0.058 \text{ y} = 0.088 \text{ mm}$$

They can be plotted with the *CrystFEL* script *detector-shift*, which can also accept a geometry file input and update the geometry with the mean shifts. This check is particularly important at the EXFEL because changing the detector distance can shift the beam center by a significant amount (0.3 mm for the data in chapter 4). Geometry optimization is revisited in section 4.2.3 and the *detector-shift* script is described as part of section 4.1 and shown in fig. 49 in that section.

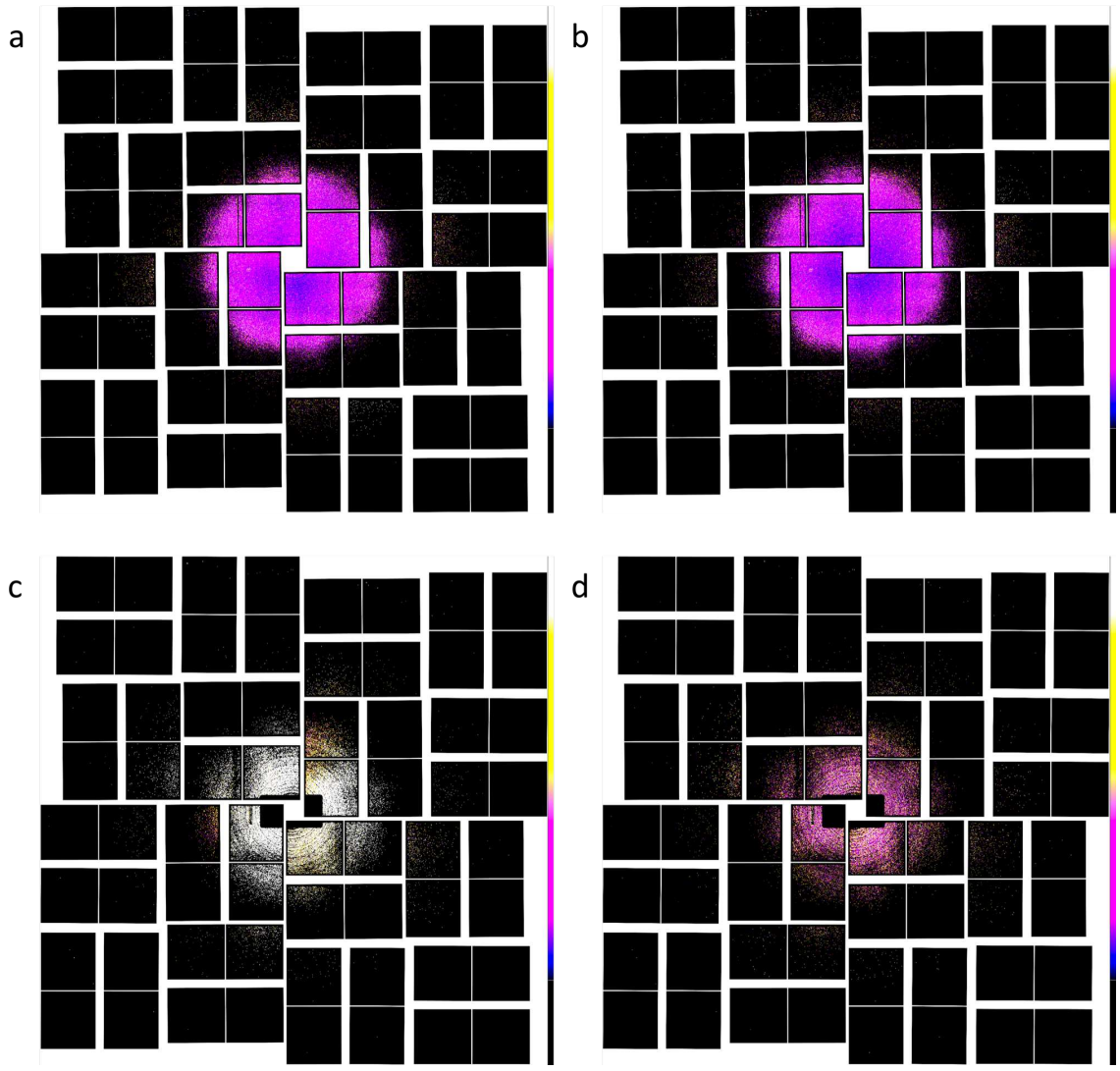

Figure 38. Geometry optimization

Geoptimiser output for LCLS June 2012 PSII light data before (a) and after (b) optimization showing the existing geometry was already good. For contrast the before (c) and after (d) output for LCLS January 2012 PSII light data showing improvement after optimization. The plots color each pixel by average displacement between observed and predicted peaks with white having the highest displacement and black the lowest, so darker images are better.

3.3.2 Integration Method

With a known set of hits and a good geometry file, the next step is optimizing the parameters used with *indexamajig*. Running *indexamajig* many times on the whole dataset for parameter screening would take significant computational time. With the assumptions that the quality of the dataset depends on the best patterns and that optimal conditions for a subset will also be good for the entire dataset, the 2,000 light PSII patterns with the highest diffraction resolution limit were selected and are used throughout the remainder of this chapter. Another simplifying assumption made for these optimizations is that indexing parameters are independent.

The *indexamajig* parameter "–integration-method" has two main methods: rings and prof2d. Rings uses three values to determine the radius of the peak, and the inner and outer radius of the background region. Prof2d uses a 2D profile fitting. On top of the method for integration, several modifiers can be applied. The 'sat' already used tells *indexamajig* to include values from saturated pixels. The 'cen' option is for centering the integration on the actual peak locations. The 'grad' option fits the background with a gradient.

Figure 39 shows merging statistics by integration method. With only 2,000 patterns, the merging statistics are not reliable at higher resolutions and are only used to compare the influence of indexing parameters. The integration methods were 'rings-sat', 'prof2d', 'prof2d-sat', 'rings', 'rings-cen', and 'prof2d-cen'. $CC_{1/2}$, CC^*, R_{split}, and SNR are all shown in the figure for completeness. For most future figures, only $CC_{1/2}$ is shown because it's easier to see the different lines with $CC_{1/2}$ than CC^* and CC's are recommended (Karplus and Diederichs 2012, 2015). The figure shows two versions of every plot as a left and right column. Merging statistics depend on

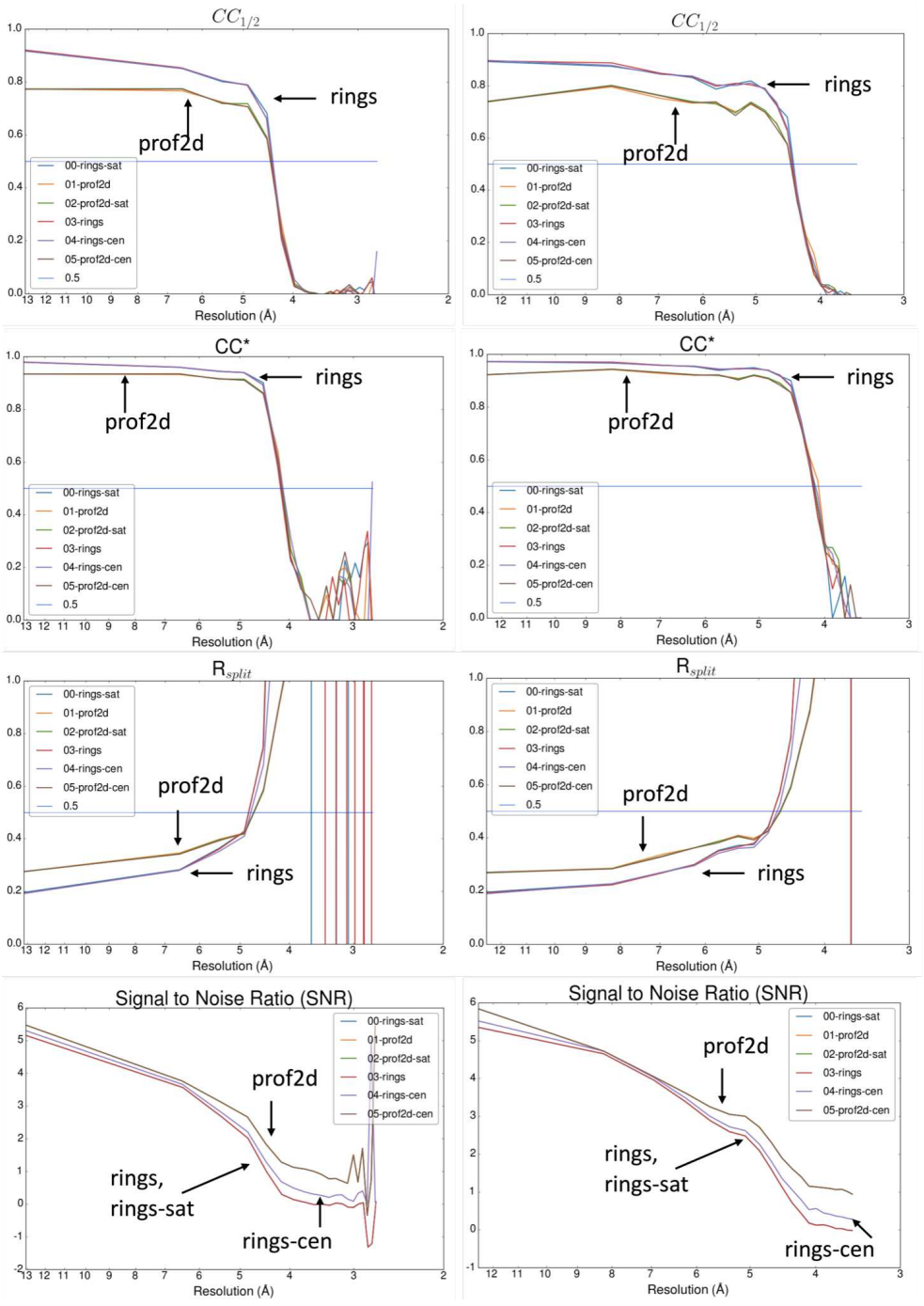

Figure 39. Merging statistics for integration method

Merging statistics for top 2000 light patterns (by diffraction resolution limit) with different integration methods. The left column shows the initial statistics with no range limitations. The right column shows the same statistics limited from 20 Å to 3.5 Å. Plots have labels showing the line groups corresponding to integration with rings (00, 03, and 04 corresponding to blue, red, and purple) or prof2d (01, 02, and 05 corresponding to orange, green, and brown).

the number and limits of resolution shells over which they are calculated. The left column has no range limit and the default number of resolution shells (10). The right column uses 20 resolution shells from 20 Å to 3.5 Å. Because decisions are made based on the plots, it's important to consider which range of data was used to calculate the statistics. From the CC plots, methods using 'rings' have better values than methods using 'prof2d'.

Table 4. Statistics by integration method

	$CC_{1/2}$	CC*	R-Split	SNR	Res. Limit nm^{-1}	Pro. Rad. nm^{-1}	Imp. Neg. Ref.
00-rings-sat	0.9380866	0.9838975	0.4476	0.798339	1.7573	0.0053	0.242
01-prof2d	0.862064	0.962249	0.4251	**1.553173**	1.7573	0.0053	**0**
02-prof2d-sat	0.8620281	0.9622382	0.4234	**1.553173**	1.7573	0.0053	**0**
03-rings	**0.9400564**	**0.9844299**	0.4462	0.79865	1.7573	0.0053	0.241
04-rings-cen	0.9335477	0.9826657	0.4367	1.181929	1.7573	0.0053	0.2565
05-prof2d-cen	0.8617836	0.9621649	**0.4229**	**1.553173**	1.7573	0.0053	**0**

Statistics for different integration methods for the top 2000 light patterns (by diffraction resolution limit) with no limit on merging shells. The resolution limit (Res. Limit), profile radius (Pro. Rad.), and number of implausibly negative reflections (Imp. Neg. Refl.) are averages (sum of the values divided by the number of images). Bolded values are the best in that column.

Overall statistics by integration method are given in table 4 which uses the default range and shells (no limits and 10 shells). In contrast, in the overall summary table (table 8), the resolution range is 20 Å to 3.5 Å with the same input files. Merging statistics ($CC_{1/2}, CC^*, R_{split}$ and SNR) have the full number of digits output by the programs. With the different 'ring' methods having similar values, the integration methods was left at its original value of 'rings-sat' for the next section.

3.3.3 Integration Radius

The integration radii are given on the command line as "–int-radius=3,4,5" with 3 for the radius of the peak, 4 for the inner radius of the background, and 5 for the outer radius of the background. Because only single digit values for the radii are used in this dissertation, the integration radius is often abbreviated by removing the commas, giving 345 for the example.

Determining the integration radii is not "try a bunch of values to see what works" parameter. Appropriate values to explore can be determined by manually examining actual patterns. For *hdfsee*, a rings parameter can draw rings around peak locations. The radius of the ring can be set in the GUI, and should match the radius used by *indexamajig*. So, to determine the correct integration radii, different ring radii are shown on patterns. Peak radii should not overlap, and for ideal data background rings would not overlap either. In viewers such as *cxiview* (Barty et al. 2014) and *DatView* (see chapter 4), the image can be easily zoomed with the scroll wheel and pixels counted. The ring radius drawn in *DatView* is editable through the configuration file. Figure 40 shows rings at different radii for a single panel for two different patterns from *hdfsee*.

Based on fig. 40, 2 and 3 are used as values for the peak radius. Since at a ring radius of 5, values rings start to overlap, values of 3 and 4 are used for the inner radius of the background. In principle, the background annulus can overlap another background annulus, so values of 4, 5, and 6 are used for the outer radius of the background. Indexing was run for integration radii 345, 245, 246, and 234 in addition to the initial indexing value of 346.

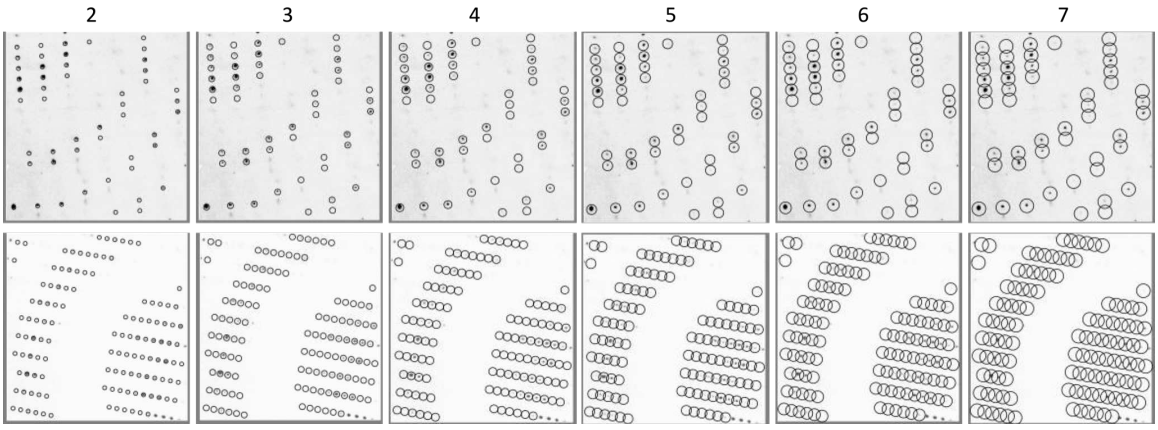

Figure 40. Ring radii for two patterns

For two patterns from June 2012 (rows in figure) a single panel with the rings at radii 2,3,4,5,6 and 7 (columns in figure). The integration radius consists of three values defining the peak and the inner and outer radii of the background. Ideally, rings from neighboring peaks should not overlap because that is a sign that the peaks are too close together to integrate correctly.

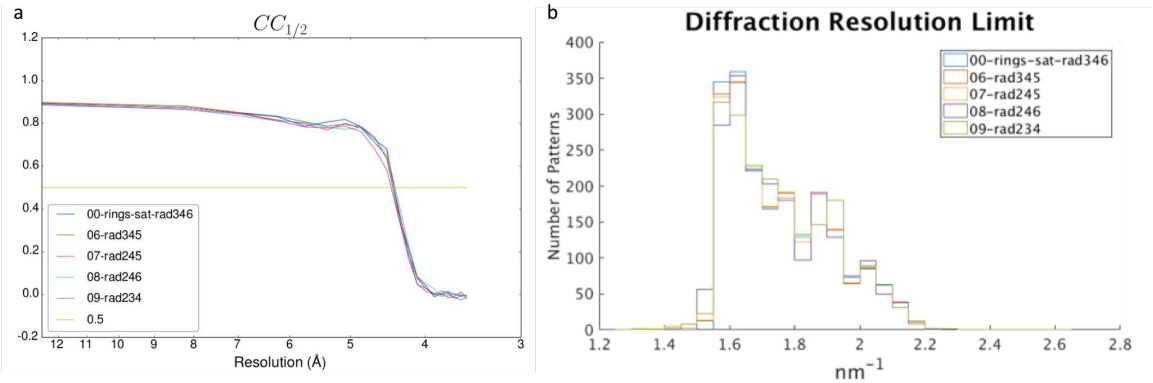

Figure 41. Merging statistics by integration ring radius

For June 2012 top 2000 light patterns (by diffraction resolution limit): (a) $CC_{1/2}$ plot with bins from 20 Å to 3.5 Å for different integration radii where the three numbers are the inner, middle, and outer radii. (b) A histogram of the diffraction resolution limit by integration radii.

Figure 41a shows $CC_{1/2}$ by resolution shell. The lines cross each other which makes it difficult to determine a "best" parameter. Overall values of CC are used instead, and are shown in table 6. Based on overall CC values, an integration radius of 245 is used as the starting condition for future indexing optimization, referred to as indexing condition 7.

Table 5. Statistics by integration radii

	$CC_{1/2}$	CC*	R-Split	SNR	Res. Limit nm^{-1}	Pro. Rad. nm^{-1}	Imp. Neg. Ref.	Ind. Pat.
00-346	0.9380866	0.9838975	0.4476	0.798339	1.7573	0.0053	0.242	**2000**
06-345	0.9375229	0.983745	0.4803	**0.94908**	1.7559	0.0053	0.345	1997
07-245	**0.9446041**	**0.9856536**	0.3899	0.946621	1.7557	0.0053	0.1585	1994
08-246	0.942414	0.9850652	**0.3775**	0.848753	**1.7574**	0.0053	0.1762	1992
09-234	0.9415449	0.9848312	0.4005	0.813462	1.7551	0.0053	**0.1435**	1993

Statistics for different integration radii (the labels are Indexing ID - integration radius) for the top 2000 light patterns (by diffraction resolution limit) with no limit on merging shells. The resolution limit (Res. Limit), profile radius (Pro. Rad.), and number of implausibly negative reflections (Imp. Neg. Refl.) are averages. Ind. Pat. Is the count of indexed patterns (out of 2,000). Bolded values are the best in that column.

Interestingly, the average diffraction resolution limit changed for the different integration radii. A histogram of diffraction resolution limits is shown in fig. 41b. This suggests the the integration radii parameter has an affect on indexing as well as integration. The affect of the integration radii is revisited in section 4.2.1.

3.3.4 Profile Radius

The profile radius as an output parameter is described in section 3.1.3. As an input parameter to *indexamajig*, it is given as "–fix-profile-radius=0.02e9" with a value in m^{-1}. Because the profile radius is recorded in nm^{-1} in the output, and m^{-1} is

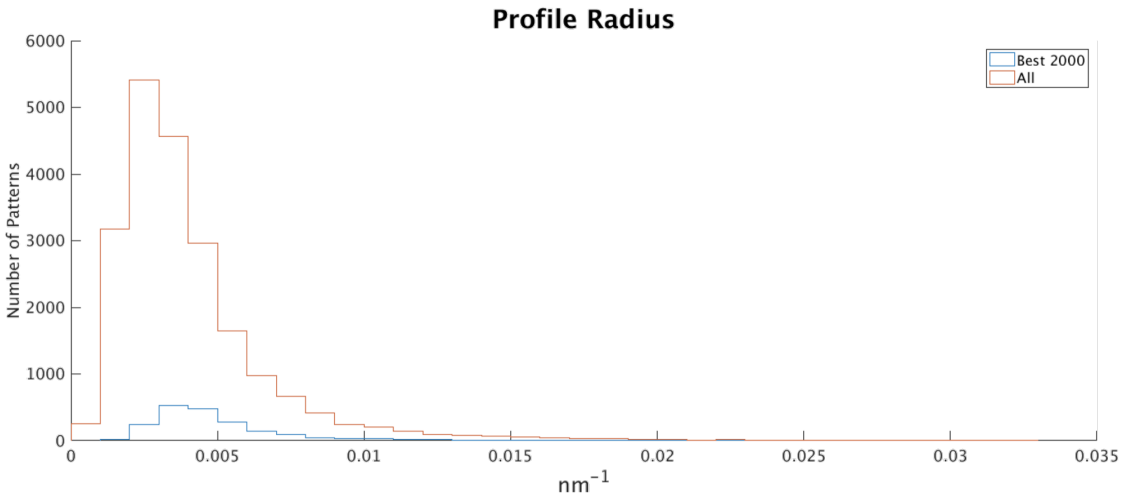

Figure 42. Profile radii distribution

The distribution of the profile radii for June 2012. All from the cell indexing before removal of the different unit cells, and the best 2000 is the top 2000 light patterns by diffraction resolution limit.

only used for the indexing command, values in the text are always in nm^{-1} rather than m^{-1}. The code initiates the profile radius with an arbitrary value of 0.02 (nm^{-1}) before updating it to the automatically determined value. For other possible values to try, the distribution of profile radii, shown in fig. 42, was examined for both the complete dataset and the 2000 "best" patterns used for indexing optimization.

For the 2000 patterns used for indexing optimization, the mean profile radius is 0.0053 and the median is 0.0044. For the full dataset, the mean is 0.0041 and the median is 0.0033. Those four values, along with 0.02 found in the code, were used as input for a fixed profile radius. $CC_{1/2}$ by resolution shell for the different conditions is shown in fig. 43a, and the histogram of implausibly negative reflections shown in fig. 43b.

Overall statistics for the profile radius screen are shown in table 6. The impact of resolution shells considered for statistics is important for the profile radius in

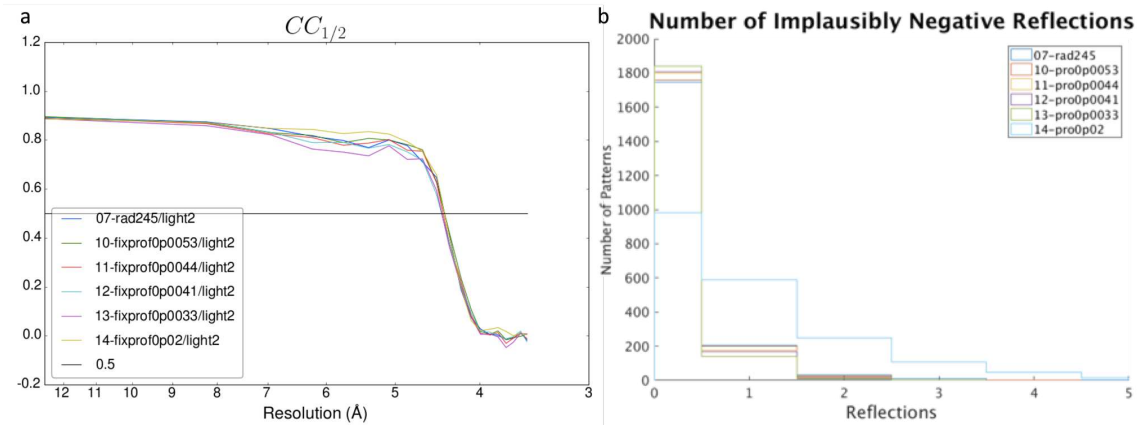

Figure 43. Merging statistics by profile radii

For June 2012 top 2000 light patterns (by diffraction resolution limit): (a) $CC_{1/2}$ plot with bins from 20 Å to 3.5 Å for different fixed profile radii values (0p0053 means 0.0053 and so on) (b) A histogram of the number of implausibly negative reflections by fixed profile radii values.

particular. By the data with no range limit shown in the table, indexing condition 12 with a profile radius of 0.0041 is best. However, when considering only the range from 20 Å to 3.5 Å (see table 8), indexing condition 10 with overall $CC_{1/2} = 0.9381196$ is better than indexing condition 12 with overall $CC_{1/2} = 0.937887$.

3.3.5 Summary

After the profile radius screen, the integration method was revisited because the 'grad' option was not initially tested. Therefore, indexing conditions 15 and 16 complete the set of integration methods with the original parameters. The integration method 'rings-cen-grad' with both modifiers was tested with the current optimal parameters of integration radii at 246 and fixed profile radius of 0.0053. Table 7 summarizes the indexing conditions screened for the top 2,000 patterns.

Table 6. Statistics by fixed profile radius

	$CC_{1/2}$	CC*	R-Split	SNR	Res. Limit nm^{-1}	Pro. Rad. nm^{-1}	Imp. Neg. Ref.
07-none	0.9446041	0.9856536	0.3899	0.946621	1.7557	0.0053	0.1585
10-0.0053	0.9447934	0.9857044	0.3799	0.915977	1.7557	0.0053	0.1364
11-0.0044	0.9445342	0.9856348	0.3607	0.9113	1.7557	0.0044	0.1103
12-0.0041	**0.946239**	**0.9860918**	0.3512	0.923247	**1.7558**	0.0041	0.1034
13-0.0033	0.9441191	0.9855234	**0.3368**	**0.988881**	1.7557	0.0033	**0.0843**
14-0.02	0.9258046	0.9805473	0.6375	0.757119	1.7557	0.02	0.8616

Statistics for different fixed profile radii for the top 2000 light patterns (by diffraction resolution limit) with no limit on merging shells. The resolution limit (Res. Limit), profile radius (Pro. Rad.), and number of implausibly negative reflections (Imp. Neg. Refl.) are averages. Bolded values are the best in that column. Indexing condition 12 indexed 1,993 patterns out of 2,000 and the other conditions all indexed 1,994 patterns out of 2,000.

Table 8 shows overall statistics calculated with 20 resolution shells from 20 Å to 3.5 Å for the 18 indexing conditions. It is sorted by $CC_{1/2}$. An interesting aspect of the table is that while many of the CC values are close together, there is a wider range in other statistics such as R_{split} and SNR. While CC is shown to be useful for resolution determination (Karplus and Diederichs 2015), it may not necessarily identify the best indexing parameters for the data. In particular, is it better to consider changes in the third and fourth digits of $CC_{1/2}$ or the high SNR and low R_{split} of indexing condition 17?

$CC_{1/2}$, R_{split}, and SNR are plotted by resolution shell for the top four indexing conditions in table 8. As in the table, indexing condition 17 has lower $CC_{1/2}$ values but good R_{split} and SNR values. Because the "best" indexing condition isn't clear with different statistics having different results, both indexing conditions 0 and 17 were used for further analysis. Merging results for the two conditions are in section

Table 7. Summary of indexing parameter screen conditions

ID	Integration Method	Integration Radius	Fixed Profile Radius
0	rings-sat	346	
1	prof2d	346	
2	prof2d-sat	346	
3	rings	346	
4	rings-cen	346	
5	prof2d-cen	346	
6	rings-sat	345	
7	rings-sat	245	
8	rings-sat	246	
9	rings-sat	234	
10	rings-sat	245	0.0053
11	rings-sat	245	0.0044
12	rings-sat	245	0.0041
13	rings-sat	245	0.0033
14	rings-sat	245	0.02
15	rings-grad	346	
16	prof2d-grad	346	
17	rings-grad-cen	245	0.0053

Summary of conditions in the indexing parameter screen.

5.1.1, and phasing and refinement results throughout chapter 6. Comparisons of the 18 conditions using *DatView* is described in section 4.3.1.

This chapter has described indexing and given results for indexing optimization based on merging statistics. The analysis assumes independent indexing parameters, that optimization of a subset will also be optimal for the whole dataset, and that patterns with the highest diffraction resolution limit are the "best" patterns. Also, the analysis presented in this chapter uses CC over other metrics. However, as discussed in section 3.3.4, merging statistics such as CC depend on the resolution range and shells used. Furthermore, as a merging statistic, it is also dependent on merging parameters. Therefore, using the CC assumes that merging parameters and indexing parameters are independent.

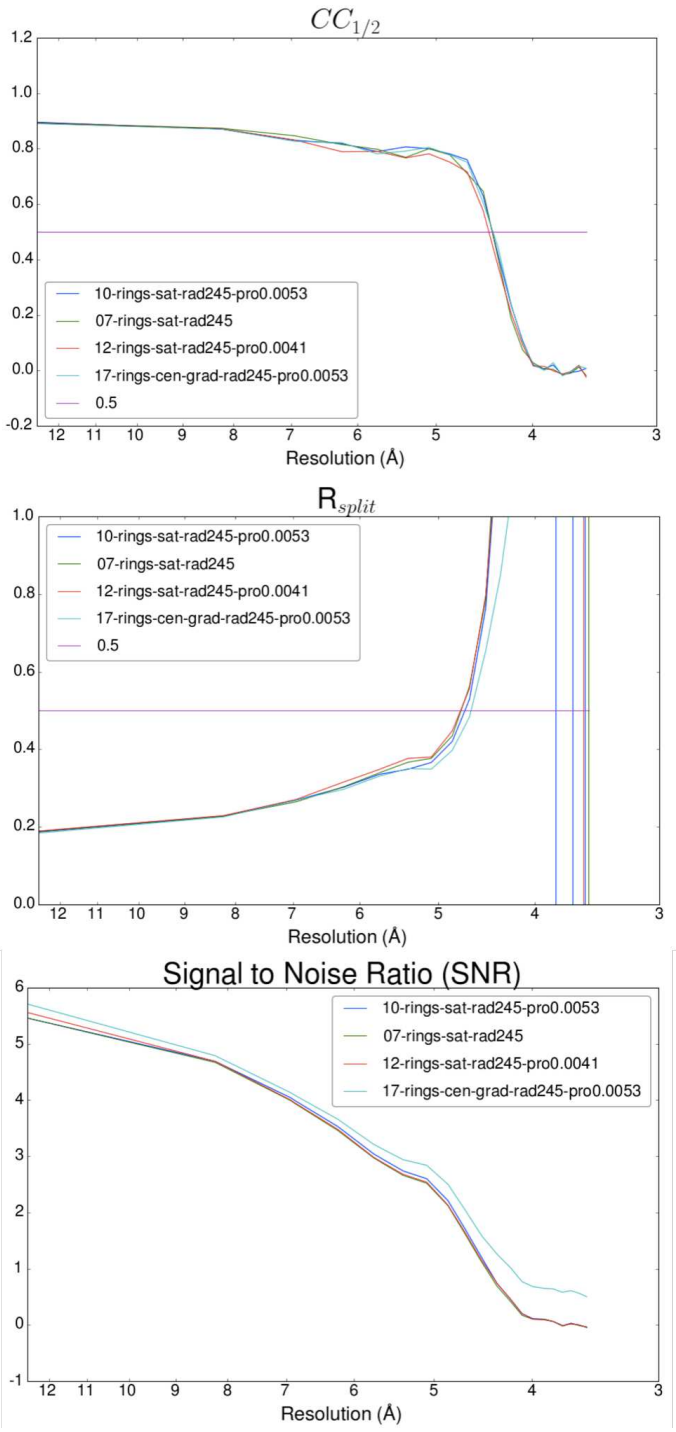

Figure 44. Merging statistics for best indexing conditions

For June 2012 top 2000 light patterns (by diffraction resolution limit), merging statistics for the top four indexing methods (by $CC_{1/2}$), see table 7, with bins from 20 Å to 3.5 Å.

Table 8. Summary of indexing parameter screen statistics

ID	CC-Half	CC*	R-Split	SNR	Res. Limit nm^{-1}	Pro. Rad. nm^{-1}	Imp. Neg. Refl.	Indexed Patterns
10	**0.9381196**	**0.9839065**	29.77	1.66433	1.7557	0.0053	0.1364	1994
07	0.9380858	0.9838973	30.38	1.626627	1.7557	0.0053	0.1585	1994
12	0.937887	0.9838435	**29.7**	1.64593	1.7558	0.0041	0.1034	1993
17	0.9368891	0.9835733	29.11	2.055195	1.7557	0.0053	0.4774	1994
11	0.9367664	0.98354	29.91	1.652837	1.7557	0.0044	0.1103	1994
15	0.9358233	0.9832842	32.1	1.66852	1.7573	0.0053	0.1955	2000
09	0.9352167	0.9831195	30.83	1.612365	1.7551	0.0053	0.1435	1993
08	0.9351567	0.9831032	30.16	1.63709	1.7574	0.0053	0.1762	1992
04	0.9350547	0.9830755	32.34	1.851304	1.7573	0.0053	0.2565	2000
03	0.9349884	0.9830575	33.24	1.612598	1.7573	0.0053	0.241	2000
06	0.934942	0.9830449	34.16	1.566075	1.7559	0.0053	0.345	1997
13	0.9348641	0.9830237	30.2	1.619128	1.7557	0.0033	0.0843	1994
00	0.9327012	0.9824352	33.29	1.612617	1.7573	0.0053	0.242	2000
14	0.9309766	0.9819647	34.28	1.54388	1.7557	0.02	0.8616	1994
02	0.8740061	0.9657989	34.56	**2.151346**	1.7573	0.0053	0	2000
05	0.8730673	0.965522	34.57	**2.151346**	1.7573	0.0053	0	2000
01	0.872626	0.9653917	34.83	**2.151346**	1.7573	0.0053	0	2000
16	0.862391	0.9623469	35.49	2.131416	1.7573	0.0053	0	2000

Statistics for all indexing conditions for the top 2000 light patterns (by diffraction resolution limit) with 20 Å to 3.5 Å merging shells. The resolution limit (Res. Limit), profile radius (Pro. Rad.), and number of implausibly negative reflections (Imp. Neg. Refl.) are averages. Bolded values are the best in that column, and the table is sorted by $CC_{1/2}$. Indexing conditions are summarized in table 7.

Chapter 4

DATVIEW

This chapter introduces a new tool called *DatView* for data visualization and subset export. With the development of fixed target chip delivery systems, the hit rate at LCLS experiments increased. Higher hit rates resulted in larger datasets. A comparison of the total number of hits and indexed patterns by experiment is shown in fig. 45, with fixed target chips having much larger datasets. The major motivation for *DatView* was LCLS November 2016, which had around 350 GB of stream files to cover the 744,504 hits.

One problem with the large size of the stream files is that a 350 GB stream file is too large to process in many programs with the computational resources available at the time. The other problem is due to the oscillating unit cells (see section 1.4.2), it was worth splitting the data into subsets to ensure that the various unit cells observed

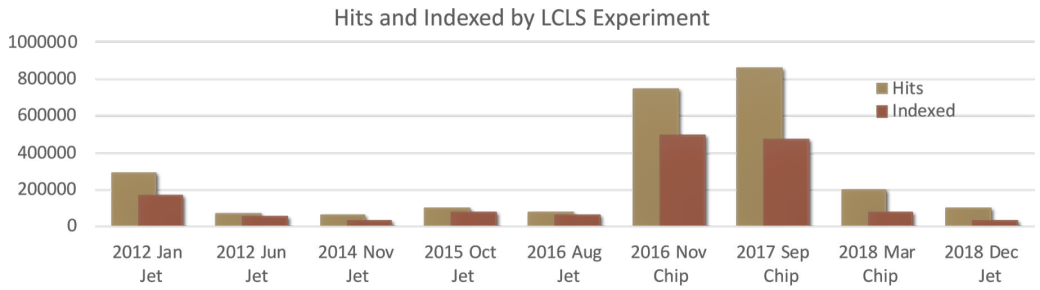

Figure 45. Hits and indexed counts by experiment

A histogram of the number of hits and number of indexed by LCLS PSII experiment. LCLS November 2016, September 2017, and March 2018 were fixed-target chip experiments, and the remainder were jet experiments. Indexing numbers come from *CrystFEL* 0.7.0 all PSII indexing (see A.1).

were combinable with each other. While some basic scripts for extracting a subset from a stream file exist, none of them run particularly fast on 350 GB of stream files, and getting the overall statistics to determine where to split the dataset is also not trivial.

DatView was developed to address the two issues of quickly getting statistics from a large dataset and exporting a subset of the patterns. *DatView* reduces the speed of loading statistics from large datasets by separating parsing of the stream file from display of statistics. Parsing of the stream file is accomplished with a script, *datgen.py*, which converts a stream file into a tab separated table format (dat table). Each row in the table corresponds to a frame or crystal. A frame that has multiple crystal lattices will have a row for each crystal lattice in the table. Columns in the table are scalar values found in the stream file. Additional columns can be added from external sources such as *NumPy* arrays or the image file. Parsing the stream file only needs to be done once.

DatView has a graphical user interface (*datview.py*) that loads the smaller dat table. The graphical user interface provides five major plot types: histograms (fig. 46a), 2D histograms (fig. 46c), scatter plots (fig. 46d), pixel plots, and aggregated plots. Pixel plots are for visualization of fixed-target chip data. Figure 12b in the introduction is an example of a pixel plot where the color of each pixel was determined by the unit cell volume of the pattern at that point. Aggregated plots calculate statistics over bins to produce line plots. Figure 55 is an example of an aggregated plot where the x axis is patterns, the y axis is the profile radius. The dataset is split into bins based on the patterns (x axis) and the y axis is the average profile radius of the bin.

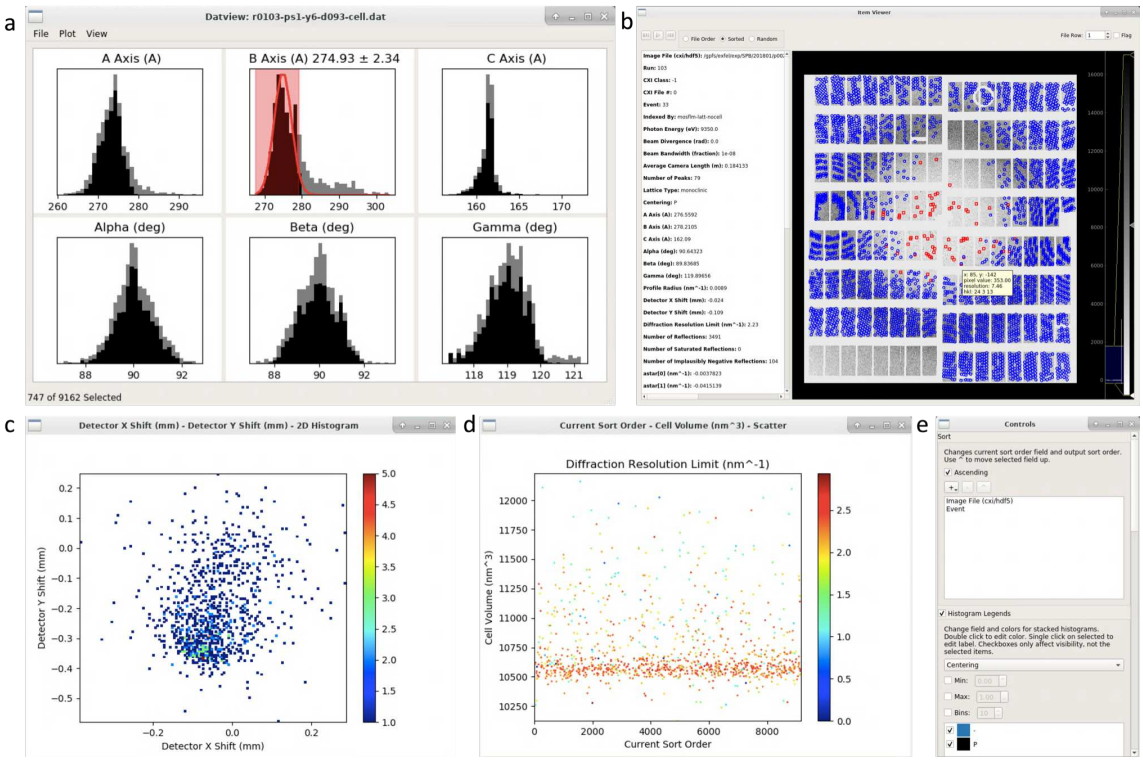

Figure 46. DatView overview

DatView improves loading time by creating a smaller table for the interface that provides the functionality of the other graphical programs. This is the data for the initial indexing of run 103 from EXFEL August 2018 (PSI data). (a) The main window of DatView, which is similar to *cell_ explorer*, see fig. 47. The plot menu gives access to many additional plot types. Histograms are added to the main window in additional rows, and other plots appear in their own windows. (b) The item viewer, accessible through the View menu, displays the frame similar to *cxiview*, see fig. 48. It also shows all related statistics for the image and has a flag option to allow selecting frame by frame. Pixel specific features are visible in the tool-tip (hover text). (c) A 2D histogram mimicking the output of *detector-shift*, see fig. 49. (d) Additional plots and parameters are also available, such as this plot over time for the unit cell volume, colored by diffraction resolution limit. (e) A portion of the control panel showing the sort order and histogram legend options. Sort order is used for (d) to approximate time, and the histogram legend uses centering to mimic *cell_ explorer* but can be configured for any parameter and color scheme.

The plots in *DatView* allow both quick visualization of statistics and selection of subsets of the data. From any plot, the user can select a region with shift+click+drag, and the current global selection will update to include only patterns meeting all selection criteria. The current selection is exportable, addressing the second major aim of *DatView*. Export is available in stream file format (if the dat table was generated from a stream file), list file format, and dat table format from the file menu. Plots have options of displaying the full dataset, the current selection, or both with the full dataset semi-transparent and the current selection in full color. Therefore, changing the selection on one plot will update all other plots, making it easy to visualize trends.

Plots in *DatView* are interactive. All plots support panning with click+drag and zooming with the scroll wheel. Additional options are plot specific. For example, the binning for histograms can be changed with +/- keys as well as from the context (right-click) menu. Tool tips (hover text) provide information about the area below the cursor. *DatView* does not restrict possible values of labels for any of the plots, and so it is possible for plot labels to be too long and overlap. To accommodate this, the tool tip includes the full labels for the point.

DatView also incorporates an item viewer so analysts can view frames in the current selection. The viewer also displays all available statistics for each item. Rows in the input table are referred to as items rather than frames/crystals/patterns because *DatView* is not limited to SFX data analysis and an item may be an unindexed frame or a single crystal lattice from a frame that has multiple crystal lattices.

Finally, *DatView* provides a comparison mode to visualize changes between input tables. The script *datcompare.py* takes multiple input tables and outputs a single *NumPy* file (.npz) file containing all items in the input tables and additional information about which items were equivalent. For SFX, a comparison table would contain all

the patterns from all the input tables (corresponding to different indexing parameters) and information identifying identical frames between the input tables. Filters can then select the input table (indexing parameters) for each frame that minimized or maximized a parameter (such as the profile radius or diffraction resolution limit, see fig. 60 for an example).

DatView has a tutorial available at https://zatsepinlab.atlassian.net/wiki/spaces/DAT/pages/827785219/Tutorial+2019. More technical documentation is provided in a manual at https://zatsepinlab.atlassian.net/wiki/spaces/DAT/pages/566263974/Manual. The code is available at https://github.com/nstander/DatView. A manuscript on *DatView* is currently under review (Stander, Fromme, and Zatsepin, Accepted).

The next section in the chapter (section 4.1) compares *DatView* with other *CrystFEL* visualization tools. Section 4.2 gives an example of indexing optimization using *DatView*. Both sections 4.1 and 4.2 use a PSI dataset collected at EXFEL in August 2018. The final section, 4.3 returns to PSII datasets, reviewing LCLS June 2012 used in chapter 3 and introducing subset analysis for LCLS November 2016.

4.1 Data Visualization Tools

DatView was influenced by many existing tools, combining functionality into a single interface with export capabilities. Without *DatView*, a typical analysis might begin with determining the indexing rate. Various scripts exist, and the one commonly used in this dissertation is *howmany_indexed* which essentially uses *grep* to count the number of chunks (frames) and crystals and output the result.

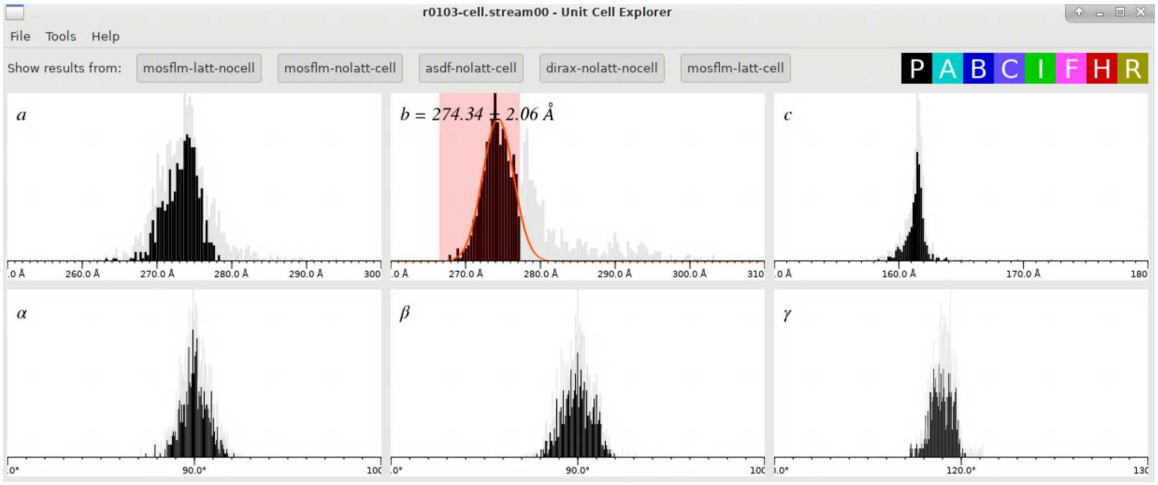

Figure 47. Cell explorer

The output of *cell_explorer* for initial indexing of run 103 from EXFEL August 2018 (PSI data).

For example data, PSI data from run 103 from EXFEL August 2018 is used. Run 103 was collected with 120 pulses per pulse train and 10 pulse trains per second, resulting in 1,200 frames per second. It is used for the numbers in fig. 22, contained 519,783 raw frames (2.1 terabytes) and 9,178 hits (19 GB). Running *howmany_indexed* on initial indexing results gave "1038 / 9162 (11.32 %)." Note that the number of chunks in a stream file may be less than the number of frames in a run because frames that take too long to process (for example, with too many peaks) are skipped and therefore not included in the output file.

An indexing rate of 11.32 % is low, suggesting something is wrong with either hit finding or indexing. The next step in analysis would be to look more closely at the results. *Cell_explorer* is a *CrystFEL* tool that reads a stream file and plots histograms of the six unit cell parameters colored by centering. Output for the sample stream file is shown in fig. 47.

Cell_ explorer allows selecting regions of a histogram with shift+click+drag that are shown in shaded red. All other histograms update with the current selection in full color and the complete dataset semi-transparent. The keyboard shortcut ctrl+f fits a gaussian to selected regions and shows the mean and standard deviation as text. The number currently selected out of the total is written to the command line.

The unit cell distributions in fig. 47 look reasonable. An ideal distribution would look gaussian and these distributions are a little skewed, but are not uncommon for PSI. Since indexing was done with prior unit cell information, there aren't any outliers and all the coloring is black for primitive (P) centering. So, for this data, *cell_ explorer* does not identify any obvious problems with indexing parameters.

There are two main frame viewers for *CrystFEL* data. *Hdfsee* is an early program designed for single event image files that has been updated to handle multi-event files. There are wrapper scripts to show peak or reflection results. *Cxiview* was developed for multi-event image files with *Cheetah* and can accept a stream file as input. It then has options to show peaks and/or reflections, and has a more flexible interface for panning, zooming, changing the color map and playing through images. Opening the stream file in *cxiview* for run 103 and navigating to the first indexed frame gives fig. 48.

The frame shown in fig. 48 looks odd because there are no reflections (green boxes) in the center of the image. This result can be related to the integration radii parameter for *indexamajig* and is the first hint at what may be wrong with the initial indexing. Optimization of the integration radii for this run is described in section 4.2.1.

Continuing with visualization tools for the data, another useful script is *detector-shift* which plots the predicted detector shifts from a stream file and can update a

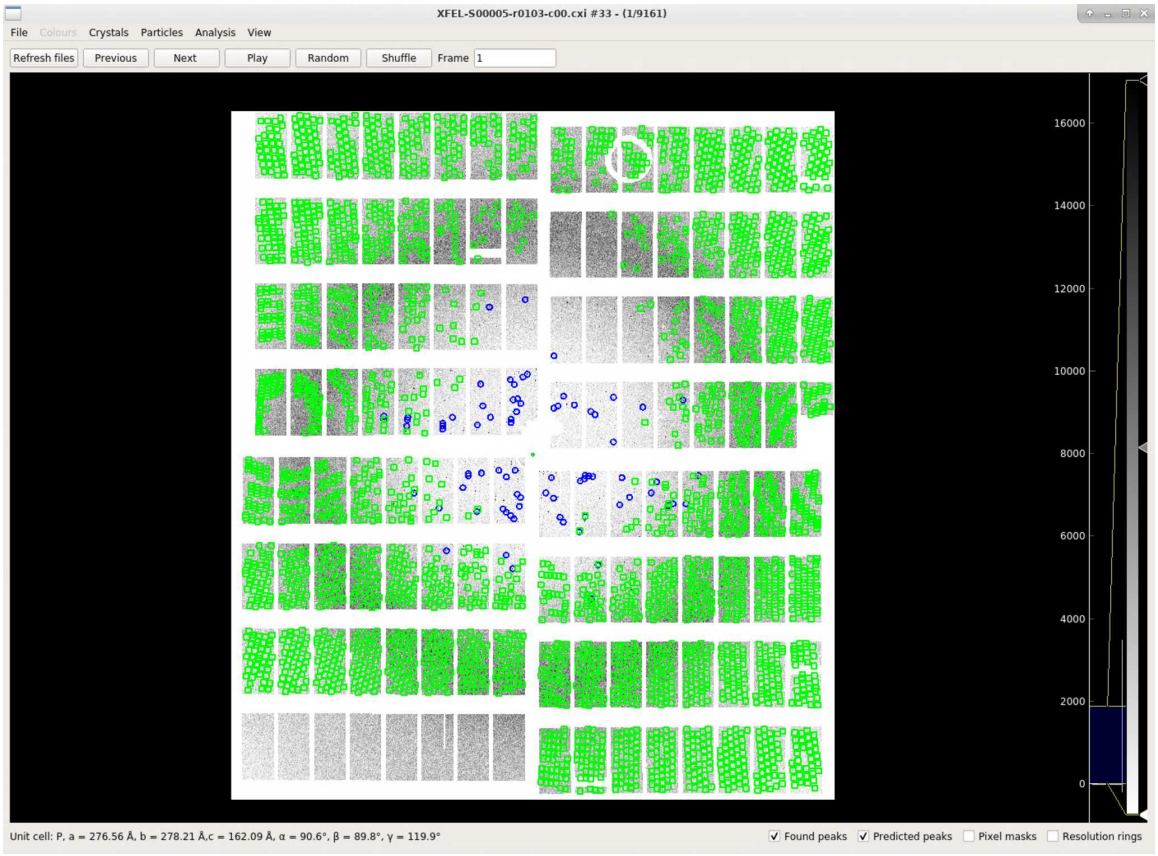

Figure 48. Cxiview

The output of *cxiview* for initial indexing of run 103 from EXFEL August 2018. The fact that the predicted peaks (green boxes, referred to as reflections in the text) don't appear in the center of the image are a sign the indexing parameters are not correct (see section 4.2.1).

geometry file with the mean value. It is also possible to click on the graph and update to the clicked values, but some versions of *detector-shift* have an error in the code that makes it look like it will update with the clicked location but actually update with the mean values because the "global" keyword is missing in the update function. There is also a script *move-entire-detector* that can update geometry if *detector-shift* is not working correctly.

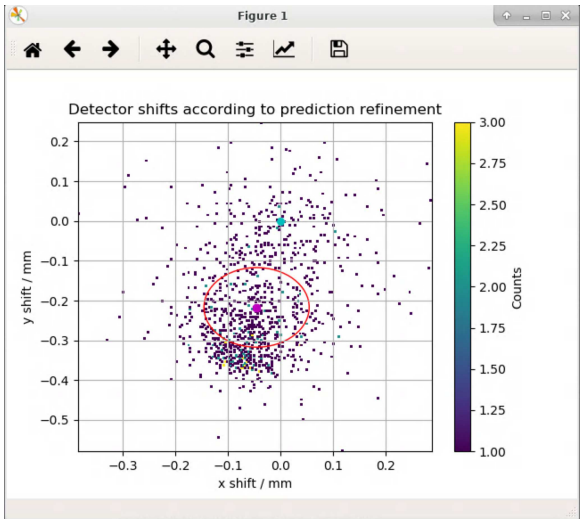

Figure 49. Detector-shift

The output of *detector-shift* for initial indexing of run 103 from EXFEL August 2018. *Detector-shift* plots the predicted beam center shifts from a stream file with each point colored by the number of patterns with that value (2D histogram). The pink dot shows the mean detector shifts and the red circle the standard deviation.

The output for *detector-shift* for run 103 is shown in fig. 49. The plot is a bit noisy, but indicates that the center of the detector is probably off. Optimization of the beam center and geometry for this dataset is described in section 4.2.3.

The tools presented so far work well and have many conveniences (see table 9). However, because they parse the stream file each time they are run, the loading speed is dependent on the size of the stream file. Numbers for the sample stream file are shown in the middle column of table 10. The final column of the table gives values for a stream file covering roughly 1/3 of the indexed PSI patterns from EXFEL August 2018.

The large stream file shows that even for just 1/3 of the indexed PSI patterns from EXFEL August 2018, many of the visualization tools take a long time to load, with *Cxiview* taking over 40 minutes. *DatView* addresses the loading time limitation

Table 9. Visualization tools

Features	DatView	Cell_Explorer	CxiView	DetectorShift
Synced Plots	Any Parameters	Cell Parameters		
Pattern Viewer	Yes		Yes	
2D histograms	Any Parameters			Detector Shifts
Update Geometry				Yes
Export	dat file, list file, stream file			
Other Plots	Scatter, pixel, aggregated			
Selections	Any plot, typed, item viewer	histograms		
Loads Stream File		Yes	Yes	Yes

A comparison of capabilities of visualization tools. In *DatView*, stream file export is only available if the data table was generated from a stream file. Also, *DatView* does not directly load stream files. Instead, a preprocessing step converts the stream file to a dat file to reduce loading time (see table 10).

of these programs and additionally allows subset selection. For this dataset, subset selection allowed fast creation of the large stream file containing indexed frames from pulses 1-40 from the full dataset which was too large to use with *geoptimiser* and *partialator* (see chapter 5).

4.2 DatView for Indexing Optimization

This section describes indexing optimization using *DatView* for PSI EXFEL August 2018 data. Indexing optimization began with run 103 as an arbitrary choice of a run that was large enough to be useful for statistics (9,178 hits, see beginning of section 4.1) without being too large to easily work with (less than an hour to index with the EXFEL computer cluster). As described in section 4.1, the initial indexing rate

Table 10. Time to appearance of graphical user interface

	Small Stream File	Large Stream File
Indexed	1038	94990
Frames	9162	94990
File Size	257 MB	71 GB
Howmany_indexed	0.215 ± 0.064 sec	41.535 ± 1.527 sec
Cell_explorer	3.870 ± 0.186 sec	9 min 22.343 ± 16.452 sec
Cxiview	12.144 ± 0.609 sec	42 min 39.081 sec
Detector-shift	5.755 ±0.116 sec	13 min 15.916 ± 2.309 sec
New Software Developed for Visualizing Large SFX Datasets		
Datgen	22.027 ± 0.397 sec	14 min 32.690 ± 21.516 sec
Datview	5.44 ± 1.435 sec	10.605 ± 0.847 sec

Time to appearance of graphical user interface of common *CrystFEL* tools for two stream files from EXFEL August 2018. All values are avearged over 5 repetitions with the exception of *cxiview* for the large stream file which was only run once.

for run 103 was low, and from figure 48 a likely problem with indexing parameters was the integration radii. Indexing optimization therefore began with the integration radius as described in the next section (4.2.1).

4.2.1 Integration Radii

The initial indexing command used for run 103 of EXFEL August 2018 was:

```
indexamajig −i split−events−r0103.lst00 −o r0103−cell.stream00\
−g [...]/natasha/indexing/dec2018/agipd_185.geom\
−−indexing=mosflm−latt−nocell,mosflm−nolatt−cell,mosflm,dirax,asdf\
−j 80 −−tolerance=5,5,5,2 −−peaks=cxi\
−p [...]scratch/natasha/indexing/ps1.cell −−nomulti
```

The integration radii was not given a specific value. According to the web page, the default integration radii is 457, which is definitely too big. The detector distance was 185 mm, so spacing between peaks is no bigger than 5. Therefore, integration radii of 234, 235, 245, and 345 were all tested. *Datcompare.py* was used to combine the

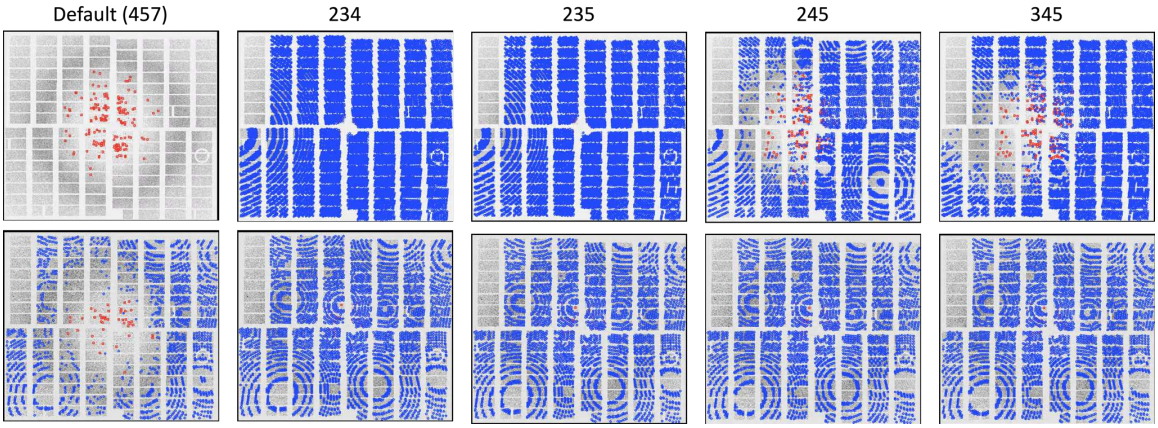

Figure 50. Integration radii affect on indexing

With *DatView*'s comparison mode, the default order is sorted by frame and then by input file. This makes it easy to see how the indexing solution changes for a single frame under different conditions in the item viewer. In this figure, each row is a frame and each column a different integration radii used for indexing, showing that reflections are missing in the middle for larger integration radii.

five different indexing conditions into a table for *datview.py*. *Datcompare.py* output is sorted first by the equivalency parameter(s), in this case the image file name and event number, and then by the input table. This means that when viewing items in the item viewer, a frame is displayed with each indexing condition before the next frame is displayed. Results for two frames are shown in fig. 50, showing that an inner radius of 4 (final two columns) is too big and results in the center of the image having no peaks.

Normal plots like histograms show all data in comparison mode. In comparison mode, it's useful to change the histogram color scheme to the comparison group, as shown in fig. 51. To evaluate the indexing rate for different values, a plot of the indexing method was used to select only indexed patterns and the compared group is plotted. The full color then shows the indexed patterns and the semi-transparent full

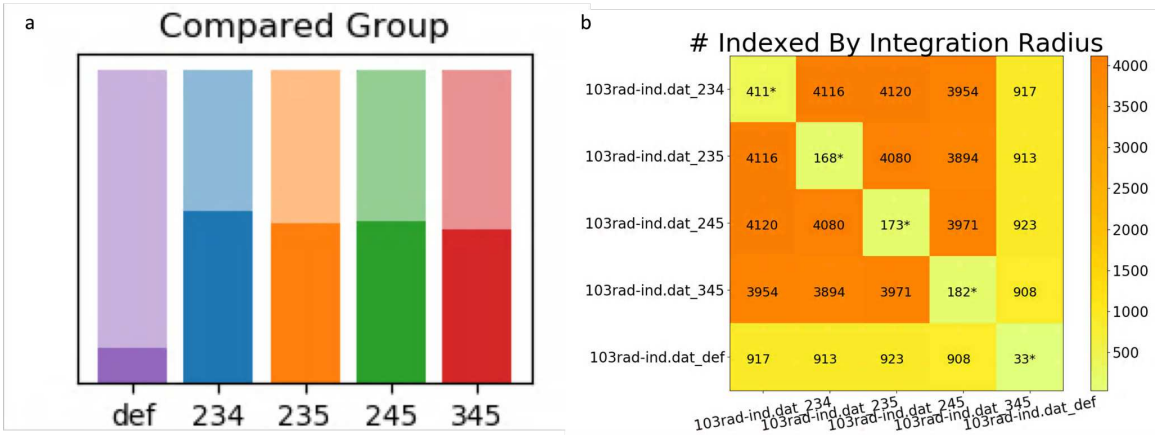

Figure 51. Integration radii affect on indexing rates

(a) With *DatView*'s comparison mode, histograms can be colored by the input file, in this case corresponding to the integration radii used during processing (def is default meaning 457). Indexed patterns are selected, so the number of indexed patterns is shown in full color and the number of hits semi-transparent. Integration radii 234 has the most indexed patterns. (b) A script available with *DatView*, *setheatmap.py*, shows the intersection of different input files, with the diagonal showing the number unique to each condition. This shows that there are some frames unique to every integration radii indexing condition.

dataset is the number of frames. Using more appropriate integration radii boosted the indexing rate with 234 looking the highest.

An interesting question to consider when looking at indexing rates is whether the same set of patterns is indexed under every condition. An external script, *setheatmap.py* shows the number of entries that are in common between input tables and the number unique to an input table along the diagonal. Every indexing condition had some patterns only indexable under that condition, even the default 457 radii. With the information so far, and the general trend that more patterns is better, an integration radii of 234 would be reasonable.

While the information available in *DatView* favors 234, it is still worth checking traditional merging statistics. A $CC_{1/2}$ plot is shown in fig. 52. The default 457

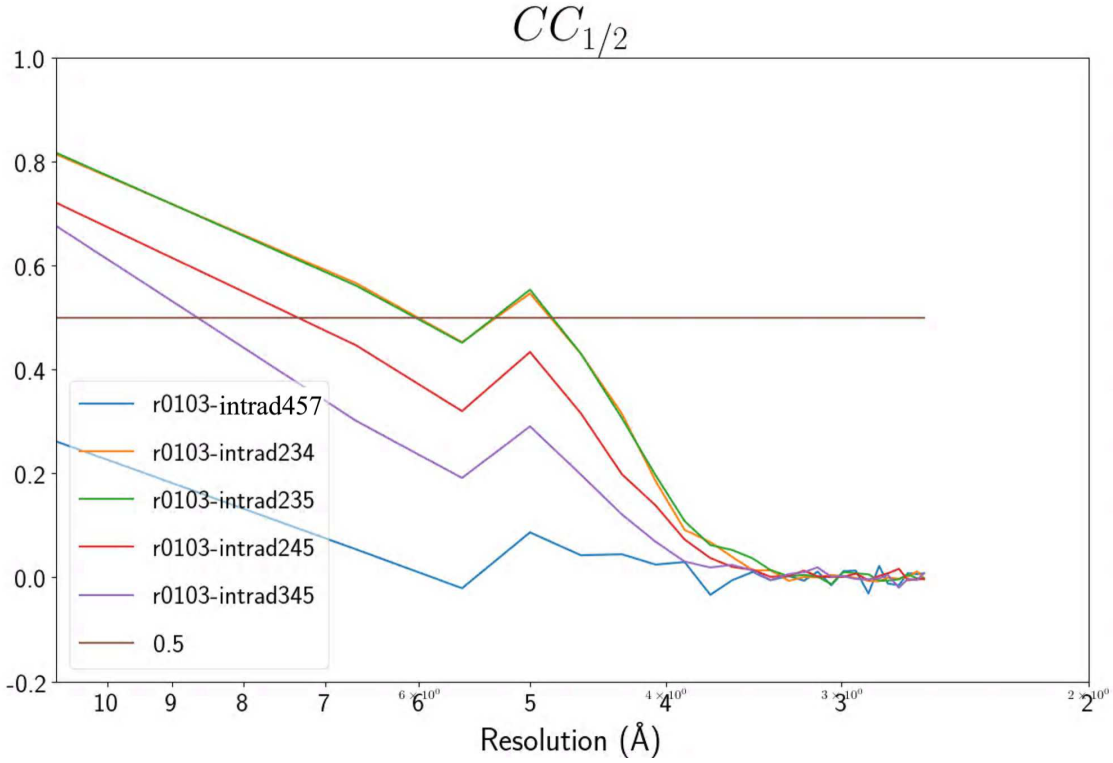

Figure 52. CC 1/2 by integration radii

$CC_{1/2}$ merging statistics for run 103 of EXFEL August 2018 by integration radii. r0103-cell used the *CrystFEL* default integration radii of 457.

indexing (blue line) has the lowest CC. This is not surprising since it had the fewest patterns and also few reflections at low resolution. Few reflections at low resolutions also explains why 345 (purple line) and 245 (red line) had visibly lower CC values. The 234 (orange) and 235 (green) lines overlap. Since the $CC_{1/2}$ plot is similar, 234 is used for further optimizations since it had more patterns (4,996 compared to 4,664 for integration radius 235).

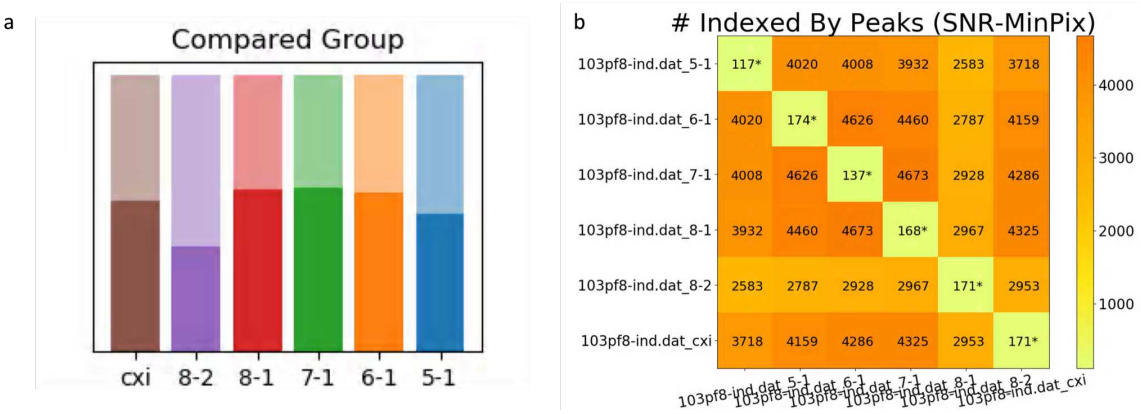

Figure 53. Peak finding affect on indexing rates

Similar to fig. 51, (a) shows a histogram in *DatView*'s comparison mode colored by the comparison group with indexed patterns selected. The comparison groups are 1) cxi, using peaks found during hit finding, stored in the CXI files, with $SNR = 8$ and min-pixels = 1. 2-6) using peak finder 8 with *indexamajig* where the first number is the SNR and the second number is the minimum number of pixels. From left to right, the stringency of the hit finding decreases. (b) The *setheatmap.py* output for the different peak finding conditions. The diagonal shows the number unique to each condition.

4.2.2 Peak Finding

The next parameter considered was the peak finding conditions. The initial indexing conditions used peaks as found by *Cheetah* and stored in the CXI files. However, *indexamajig* at this point had the capability of running *peakfinder8*. In looking at images, it seemed like some peaks were being missed. So several peak finding conditions were compared. In the legends, the two numbers separated by a dash are the signal to noise ratio and minimum number of pixels. *Cheetah* was using a signal to noise ratio of 8 and a minimum number of pixels = 1. With *indexamajig* a stricter condition of 8-2 was tried, then with descending stringency 8-1, 7-1, 6-1, and 5-1. The results were processed with *datcompare.py* and are shown in fig. 53.

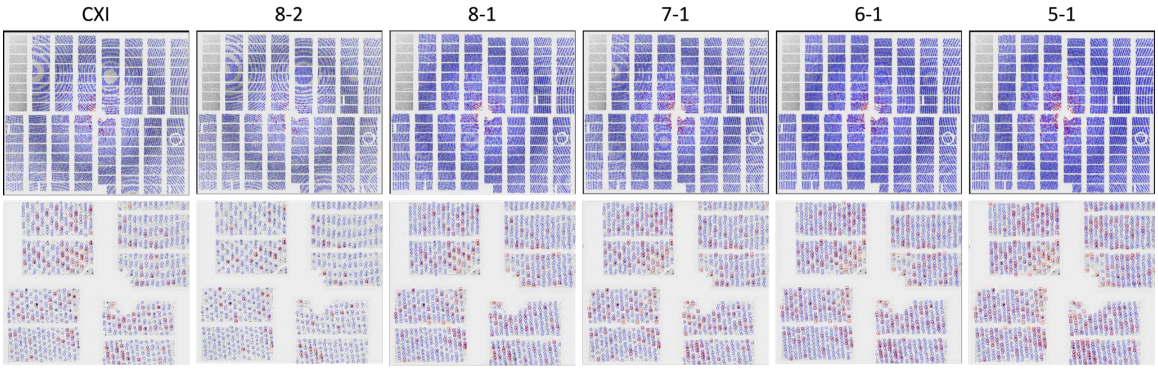

Figure 54. Peak finding affect on indexing

Similar to fig. 50, the columns in the figure are different peak finding conditions, but the second row is a zoom in around the beam stop of the frame shown in the first row. The peak finding conditions use the peaks found during hit finding (cxi), or peak finder 8 with *indexamajig* where the two numbers are the SNR and min number of pixels. From left to right (not counting cxi), the stringency of peak finding decreases.

Similar to fig. 51, figure 53 has the indexed patterns selected and shows the compared group in part a. The brown column uses the *Cheetah* results stored in the CXI file, and the following columns go from most stringent to least stringent. Interestingly, both very strict (SNR=8, min-pixels=2) and very loose (SNR=5, min-pixels=1) have lower indexing rates. The SNR=8 and SNR=7 both with min-pixels=1 (red and green columns) had similar rates, but in comparing the number unique to each subset SNR=7 had slightly more patterns (fig. 53b).

A comparison of a frame under different peak finding conditions is shown in fig. 54. Looser hit finding resulted in more reflections, suggesting over-prediction. The profile radius (since it is a measure of indexing accuracy giving the distance between peaks and corresponding reflections, see section 3.1.3) was also compared for the different peak finding conditions with results plotted in 55.

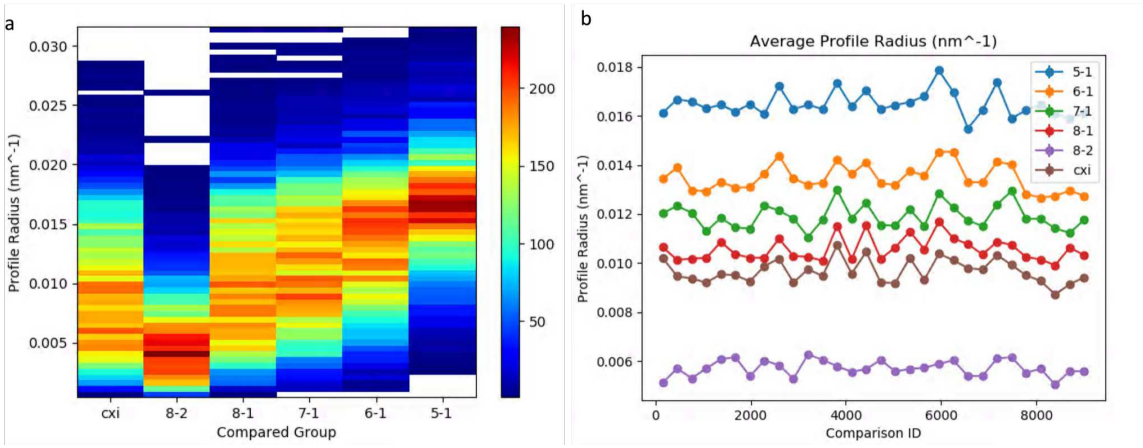

Figure 55. Peak finding affect on profile radius

The peak finding conditions use the peaks found during hit finding (cxi), or peak finder 8 with *indexamajig* where the two numbers are the SNR and min number of pixels. From left to right (not counting cxi), the stringency of peak finding decreases. (a) A 2D histogram showing the distribution of profile radii by peak finding. (b) An aggregated plot showing the average profile radius in each bin. The comparison ID is a unique identifier for each frame, and is data collection order. So, the plot shows the average profile radii over data collection order with different lines for each peak finding condition.

Unfortunately, with different peak finding conditions, the peak-reflection pairs are very different for each condition. The therefore unsurprising trend in fig. 55 is that stricter peak finding (fewer peak-reflection pairs) had lower profile radii than looser hit finding (many peak-reflection pairs). It is interesting to realize that some other parameters must have varied between *Cheetah*'s SNR=8, min-pixels=1 and *indexamajig*'s SNR=8, min-pixels=1 since in fig. 55 the brown *Cheetah* line is visibly better than the red *indexamajig* line.

To complete comparison of the various peak finding options, the data was merged and $CC_{1/2}$ plots are shown in fig. 56. Figure 56a compares the merges without any other processing. The strictest condition (8-2, orange line) is visibly worse. However, the strictest condition also had the fewest patterns. It would be helpful to compare

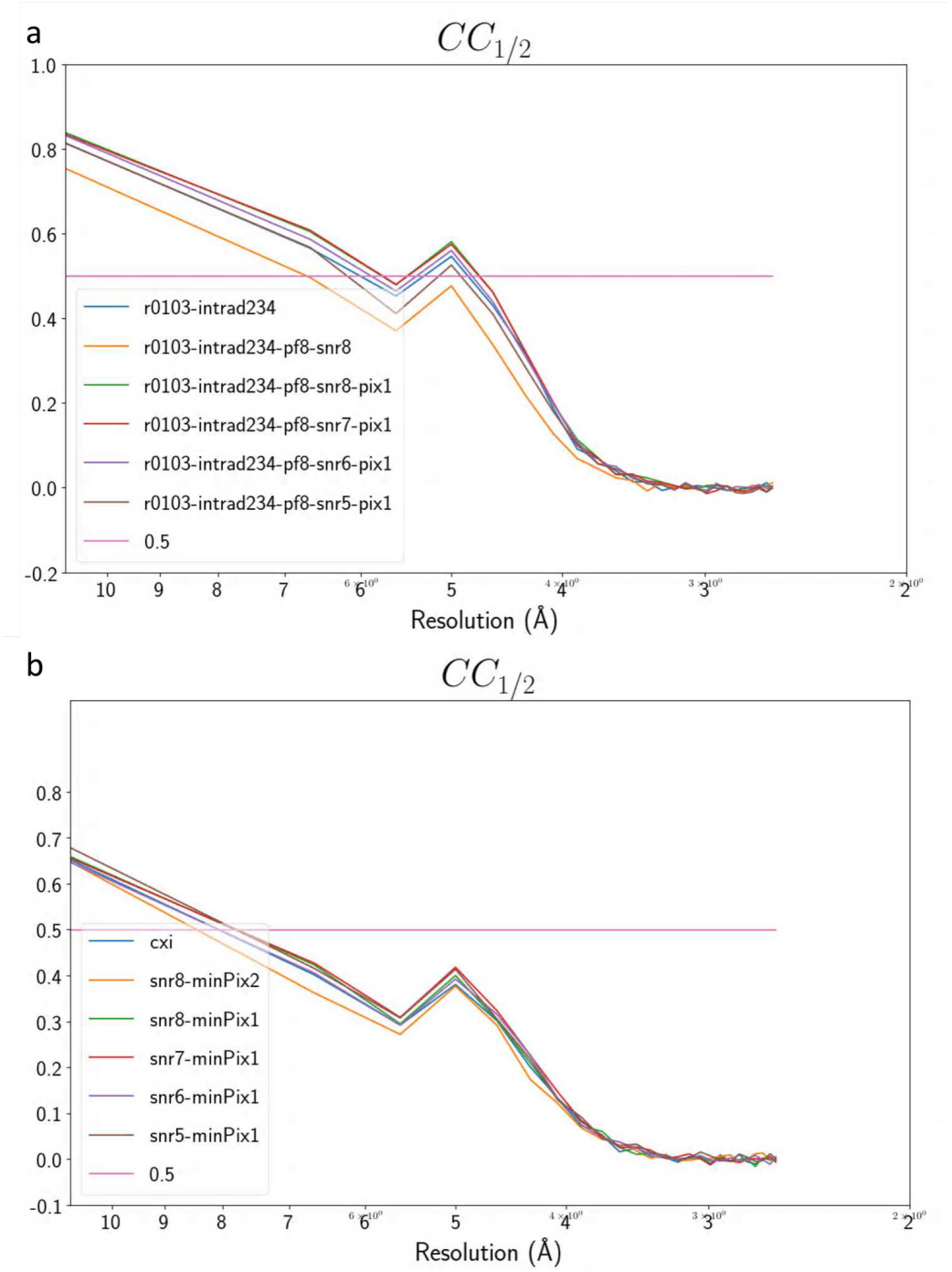

Figure 56. Peak finding affect on CC 1/2

The peak finding conditions use the peaks found during hit finding (cxi), or peak finder 8 with *indexamajig* where the two numbers are the SNR and min number of pixels. (a) The $CC_{1/2}$ merging statistics for each peak finding condition. (b) The merging statistics for just the set of 2,160 patterns indexed for all peak finding conditions.

the $CC_{1/2}$ plot for only the patterns indexed under every condition to ensure that the strictest condition really is worse and is not just impacted by the number of patterns.

When the GUI is launched with input from *datcompare.py*, additional filters are available from the filter section of the control panel. One of them is the "All Between" filter which keeps every condition for a frame if all conditions for the frame are between the given values. Using an "All Between" filter on unit cell volume initiates the values at the min - 1 and the max + 1, thus selecting all indexed patterns since unindexed patterns have a value of -1 which is not considered when calculating the minimum. This results in a selection of 12,960 items. Since there are six conditions, that means 2,160 patterns were indexed under all six conditions.

This set of patterns was then exported in stream file format with partitioning by comparison group. Partitioning means the output is split into multiple files or bins, so using the comparison group field means there are six output stream files. Each output file contains 2,160 patterns from a single indexing condition. These were then merged and $CC_{1/2}$ calculated and shown in fig. 56b. Interestingly, the orange line corresponding to the strictest hit finding is still the worst, although not by as much. With the red line corresponding to SNR=7 and min-pixels=1 appearing the best or at least overlapping with SNR=8 and min-pixels=1 in the $CC_{1/2}$ plots of fig. 56 and having a slightly higher rate, it was used for further optimizations.

4.2.3 Geometry

Geometry optimization at experiments is often performed by Dr. Oleksandr Yefanov, author of *geoptimiser* (Yefanov et al. 2015). For EXFEL August 2018, there were two possible geometry files. One was optimized on August 31, 2018 at the

experiment following this one, the other provided in December 2018 from a later experiment. Initial indexing used the latest (December) geometry file. However, Dr. Oleksandr Yefanov had reported a higher indexing rate than the rates observed in indexing optimization so far. After locating the stream file with Dr. Oleksandr Yefanov's best indexing results, the August geometry file was located.

Indexing was then run with the current optimized parameters and the August geometry file and Dr. Oleksandr Yefanov's parameters with the August geometry file, referred to in legends as Aug31 and Aug31pm respectively. Also, the detector shift results plotted in fig. 49 were generated at this point. Dec18 is the initial geometry file from December 2018, dec18ds is after one iteration of *detector-shift* where it was intended to do the visible shift instead of the mean shift but the *detector-shift* script was an incorrect version that updated with the mean shift. Dec18ds2 ran the *detector-shift* script again on the results of dec18ds. The final detector shifts for the five geometry conditions are shown in fig. 57b.

A comparison of the indexing rates by geometry file is shown in fig. 57a. The indexing optimizations so far did aid in indexing rates since the final two columns in the histogram comparing the August geometry with the optimized parameters (blue) and Oleksandr's parameters (yellow) shows the optimized with a higher rate. Also, comparing the the December geometries (green, red, and purple lines), it is clear that the incorrect beam center had a large impact on the indexing rate.

As a point of interest, in EXFEL March 2018, the indexing rate with PSI was high at the detector distance of 310 mm but the rate at 160 mm was so low that no trend in the *detector-shift* plots was visible. On the hunch that the beam center was to blame, with only 200 indexed PSII patterns at 185 mm, the *detector-shift* plot was able to be used to update the geometry file and get it close enough that the indexing

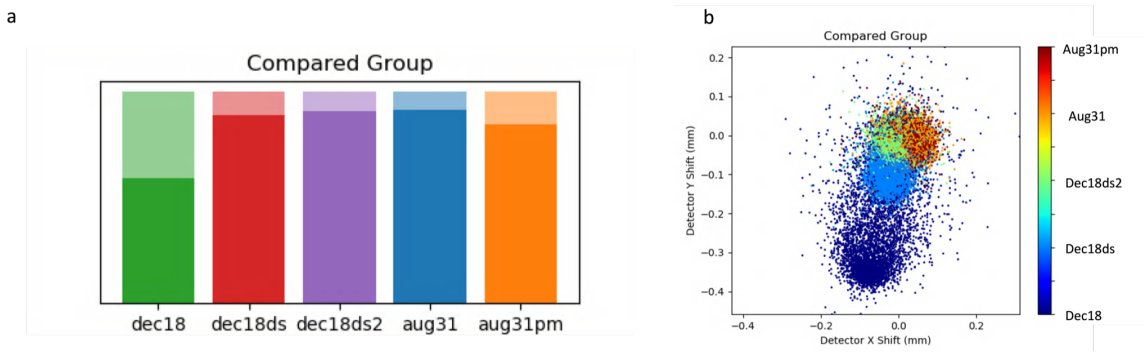

Figure 57. Geometry file affect on indexing rates

Two geometry files were provided by Dr. Oleksandr Yefanov, one on August 31 and one on December 18 (both in 2018). Indexing results so far have used the December geometry as provided with the camera length shifted to match this experiment. However, at the EXFEL, a change in camera length is usually accompanied by a change in beam center as the rails are not perfectly parallel to the beam. December 18 ds stands for the December 18 geometry file after the detector shift script updated the geometry file. December 18 ds2 was created after reindexing with December 18 ds and applying the detector shift script a second time. August 31 already had the correct camera length. August 31 pm stands for the August 31 geometry file with the indexing parameters used by Dr. Oleksandr Yefanov when he first tested it on this dataset. (a) Histograms showing the amount indexed with each the geometry file (b) The scatter plot showing the detector shifts, colored by geometry file.

rate for PSI jumped up to around 60% allowing direct optimization of the geometry for those runs. The take-away message is that low indexing rates at the EXFEL in particular are more likely related to a change in beam center with a change in detector distance than the quality of data.

The geometry files were also compared for their effect on the profile radius and $CC_{1/2}$ as indications of their contribution to the data quality. At first glance, the August geometry file with Dr. Oleksandr Yefanov's parameters (orange line in fig. 58a) appears best. However, recall from the peak finding section that the number of found peaks has a strong influence on the profile radius and Dr. Oleksandr Yefanov's parameters used peaks from *Cheetah* instead of *indexamajig* as was done with all

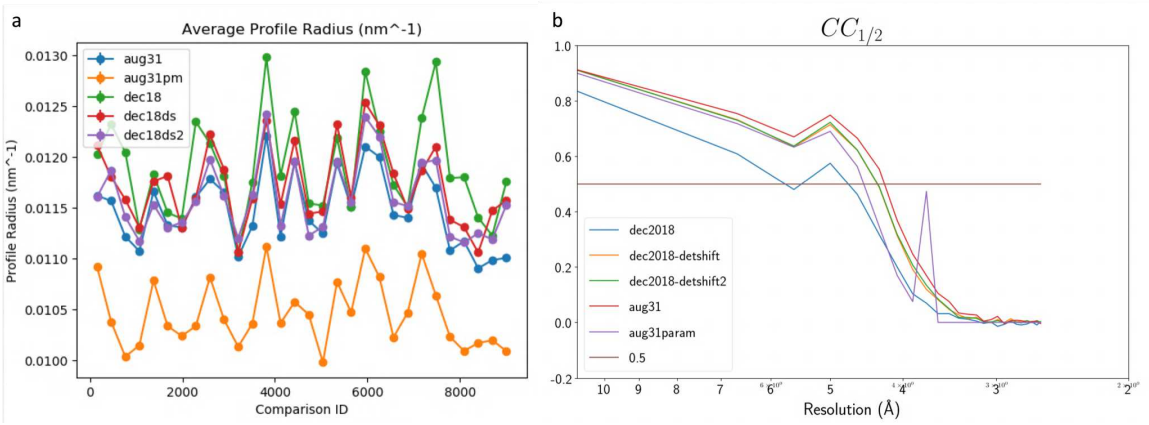

Figure 58. Geometry file affect on profile radius and CC 1/2

See caption on fig. 57 for details on the different geometry groups. (a) The profile radius by geometry used. August 31 parameters (aug31pm) used the CXI peak list whereas all other conditions used *indexamajig* with peak finder 8, SNR 7, and min peaks = 1. As expected for fewer peaks, the profile radius of August 31 parameters is visibly lower than the other four. (b) The $CC_{1/2}$ plot for the different geometry files.

other conditions. The $CC_{1/2}$ plot in fig. 58b shows the initial December geometry with the lowest values, followed by Oleksandr's indexing parameters with the August 31 geometry, then the two December geometry updates, and finally the optimized parameters with August 31 as the best.

With a good indexing rate and reasonable indexing optimization, the entire PSI dataset was processed with the optimized parameters and the best December geometry (detector shifted twice) and August geometry. The effect on the profile radius and diffraction resolution limit is shown in fig. 60. The histograms have selected as a filter the minimum profile radius or maximum diffraction resolution limit in each pair. The August geometry has the better profile radius for more patterns than December geometry but not by very much. The trend with diffraction resolution limit is much stronger with the better diffraction resolution limit more often observed from the December geometry.

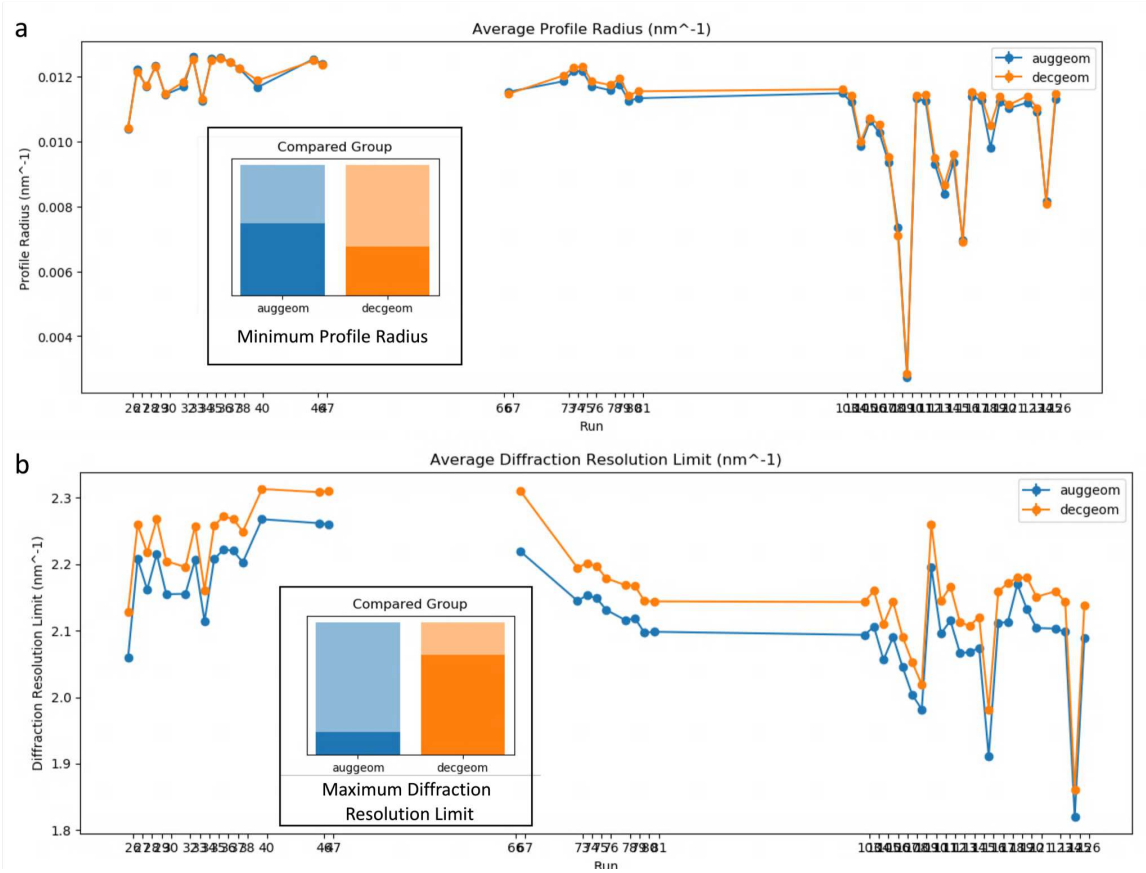

Figure 59. Geometry file affect on profile radius and resolution by run

A comparison of the August 31 and December 2018 prediction refined twice geometry files for all PSI runs for EXFEL August 2018. (a) the August 31 geometry tended to have better profile radii per patten. The histogram shows the result after selecting the condition with the minimum profile radius for each pair. (b) the August 31 geometry had a lower diffraction resolution limit for most patterns. The histogram shows the result after selecting the condition with the maximum diffraction resolution limit (nm^{-1}) for each pair.

This raises the question of what makes the difference between the two geometry files. A summary of statistics related to the geometry files is shown in table 11. While panel positions may also be different between the two files, a direct comparison of geometry files is not straightforward. On the surface, though, one big difference is in the camera length. The December geometry having a higher diffraction resolution limit has to do with its camera length being smaller than the August camera length so the same position on the detector is seen as a higher resolution in December geometry than in August geometry.

Table 11. PS1 statistics by geometry file

	August Geometry	December Geometry
Total Processed	310449	310433
Total Indexed	280771	280435
Pulses 1-40 Processed	104532	104527
Pulses 1-40 Indexed	94990	95034
Camera length	0.187	0.1833
Photon energy	9300	9350

Statistics for all PSI runs from EXFEL August 2018 for the two different geometry files. There were 120 pulses per train, so taking pulses 1-40 corresponds to roughly 1/3 of the data. It is also the dark data before the laser pulse which occurs between pulses 40 and 41.

With the change in camera length resulting in a higher diffraction resolution limit with the December geometry, it's reasonable to assume that the December geometry would also have better CC since the same values would be shifted to a higher resolution. Calculating the CC requires merging all the data, and unfortunately the 310,000+ dataset was too big to process with *partialator* (see chapter 5) as mentioned in section 4.1. *DatView* was used to select the first 40 pulses from the entire dataset and export them as the large stream file used in table 10. Since this data was laser illuminated between the 40th and 41st pulse, this corresponds to the dark PSI data. $CC_{1/2}$ for

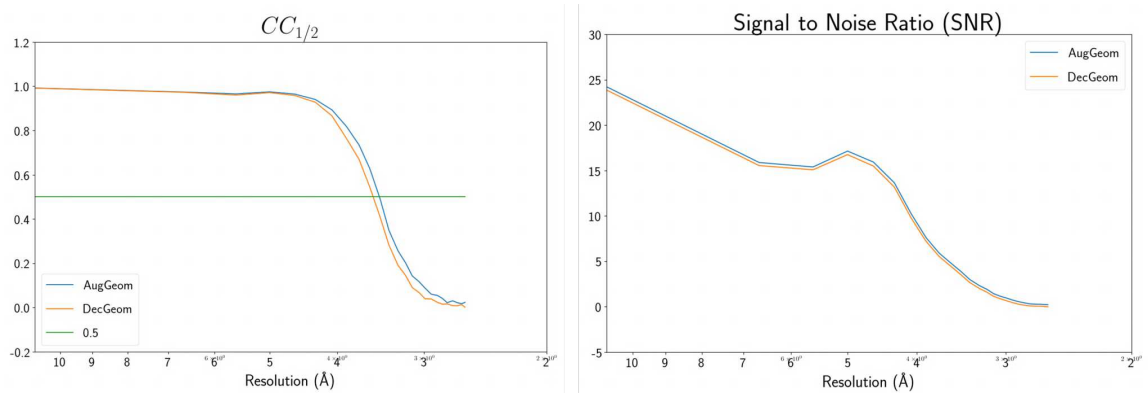

Figure 60. Geometry file affect on merging statistics

Using August 31 and December 2018 prediction refined twice geometry for all PSI runs pulses 1-40 for EXFEL August 2018, (a) $CC_{1/2}$ plot and (b) SNR plot.

these smaller stream files are shown in figure 60. Surprisingly, the August geometry provides better statistics despite having the longer camera length, suggesting it is more appropriate for this dataset.

In observing some of the patterns so far, both over-prediction of reflections and a large profile radius have caused concern that the indexing isn't accurate enough for how small the integration radius has to be. The measured profile radius seems like it would be more related to the geometry and panel positions than other indexing parameters. However, updating the panel positions with *geoptimiser* works best with good indexing results. If there is over-prediction, then the average shifts won't necessarily fix the problem. As a curiosity, since PSI has sufficient peaks, each quadrant was indexed individually with the idea that whole quadrant shifts would be easier to detect with prediction refinement when only considering one quadrant at a time.

Figure 61 shows how the frames look in comparison mode. For many cases, indexing a single quadrant reduces the number of reflections compared to considering all quadrants at once. The predicted detector shifts over time are plotted in fig. 62.

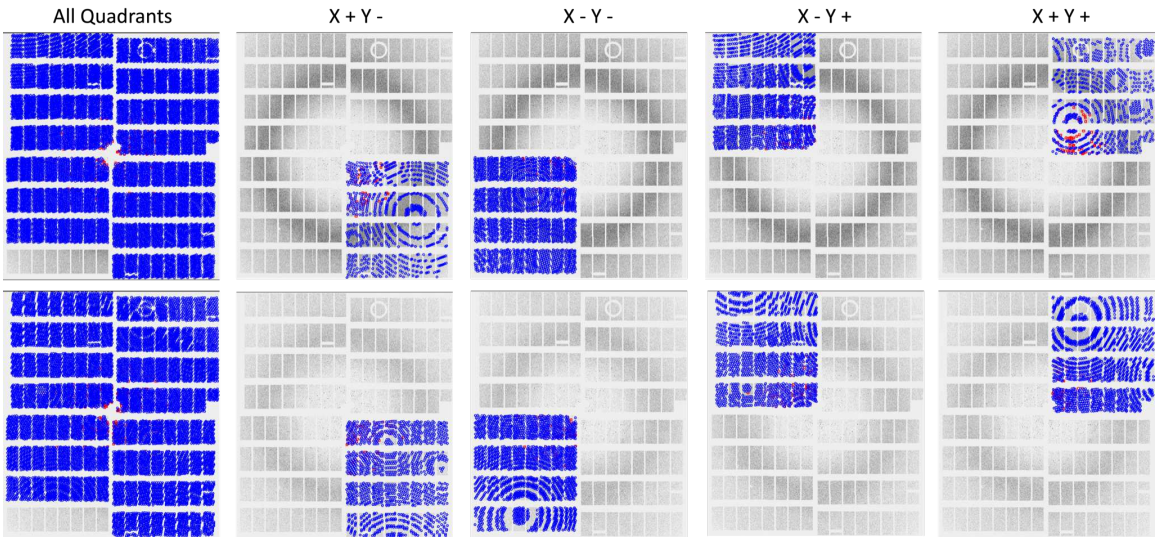

Figure 61. Geometry quadrant affect on indexing

Starting with August 31 geometry after one round of *geoptimiser* from pulses 1-40, four new geometry files were created each with only one valid quadrant. Indexing was done with each geometry file, with results for two frames (rows) shown in the figure.

Two problems arose with this analysis. First, looking at the frames in fig. 61, is it really appropriate to shift quadrants if each of them indexed with possibly very different unit cells or orientations. Second, looking at fig. 62 for some areas of the plot there is a very clear trend but by the end of the plot most lines are overlapping. Since this data covered three shifts, the major jumps are likely related to changes between shifts. The beam center is expected to vary between shifts and sometimes between runs. In that case, multiple geometry files are created and refined for different sets of data. However, the changes shown in fig. 62 are small enough (notice the y axis is in 100ths of a mm) that indexing corrects for them. The problem is quadrant positions should not change between shifts like they are doing in fig. 62.

So, while the analysis by quadrant is interesting, there are too many unknowns to apply it immediately to this data. Therefore, the best indexing condition for

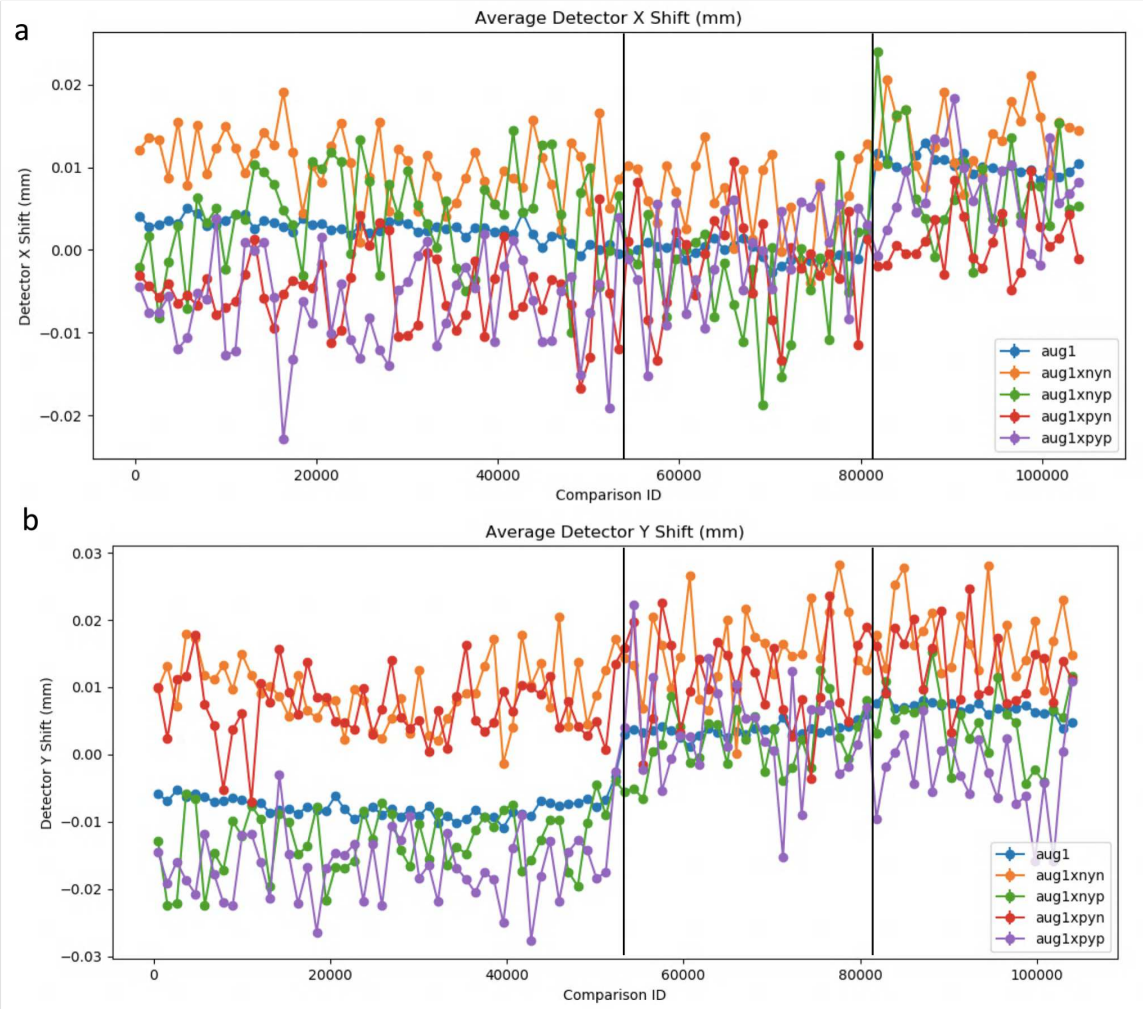

Figure 62. Geometry quadrant affect on detector shifts

Starting with August 31 geometry after one round of *geoptimiser* from pulses 1-40, four new geometry files were created each with only one valid quadrant. Indexing was done with each geometry file. The plots show the predicted detector shifts in X (top) and y (bottom) for the full geometry file and each quadrant with n standing for negative and p standing for positive. Note the jumps around 80,000 in the x shift and between 40,000 and 60,000 in the y shift correspond roughly to breaks between days of data collection.

this dataset to date uses an integration radius of 234, the August geometry, and *indexamajig* for peak finding with *peakfinder8* with SNR=7 and min-pixels=1. It's worth noting that fixing the profile radius was attempted but resulted in no reflections being output in the stream file although the indexing rate remained high. Perhaps no reflections met the fixed profile radius enforcement. A photon energy screen was also tried since the photon energy at EXFEL is not stored per shot, but the results were inconclusive. An interesting further study would be applying a finer photon energy screen and merging patterns from whichever photon energy minimized the profile radius. However, it would be better to apply the screen to a sample with more reliable indexing first as a proof of principle.

4.3 Photosystem II

The previous sections have followed indexing optimization for a PSI dataset from EXFEL August 2018. This section returns to data analysis of PSII datasets. *DatView* has not been directly used for indexing optimization for PSII datasets. However, results from the indexing optimizations presented in section 3.3 are reviewed with *DatView* in section 4.3.1. Sections 4.3.2 and 4.3.3 introduce subsets of PSII data from LCLS November 2016. As briefly mentioned at the beginning of this chapter, one of the motivations for the development of *DatView* was determining what range of unit cells was combinable from the oscillating unit cells in LCLS November 2016. Section 4.3.2 introduces subsets based on the unit cell and diffraction resolution limit that are used again in chapters 5 and 6. Section 4.3.3 gives results from a reanalysis with subsets generated from *DatView*.

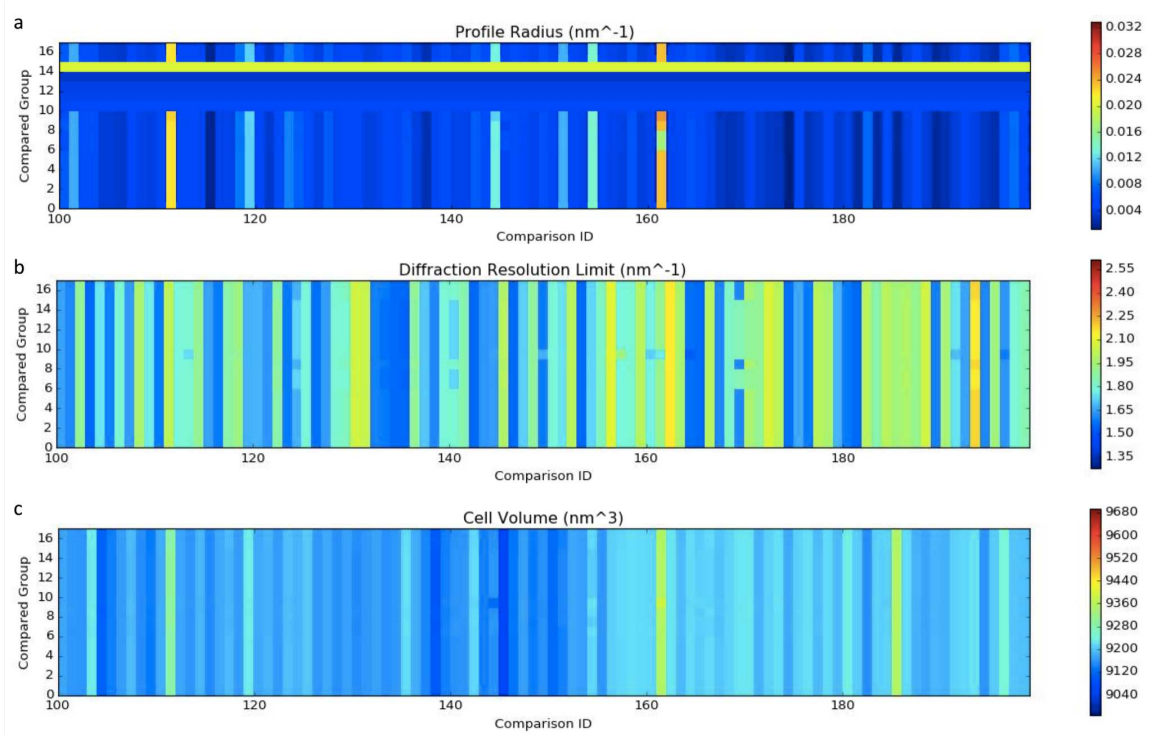

Figure 63. Indexing optimization comparison plots

For the indexing optimizations described in section 3.3, pixel plots for 100 of the 2,000 best patterns showing from top to bottom the change in (a) cell volume, (b) diffraction resolution limit, and (c) profile radius per pattern. Each row is an indexing condition (compared group) with numbers matching those given in table 7. Each column is a pattern (comparison id).

4.3.1 Photosystem II Indexing Conditions

Section 3.3 screened 18 indexing conditions (see table 7) using the 2,000 PSII patterns with the highest diffraction resolution limit from LCLS June 2012. The 18 indexing conditions were processed with *datcompare.py* and visualized with *DatView*.

Figure 63 shows the unit cell volume, diffraction resolution limit, and profile radius changes over the 18 conditions (y axis) for 100 patterns (x axis) as pixel plots. It's interesting to note that for most frames, the parameters change with each

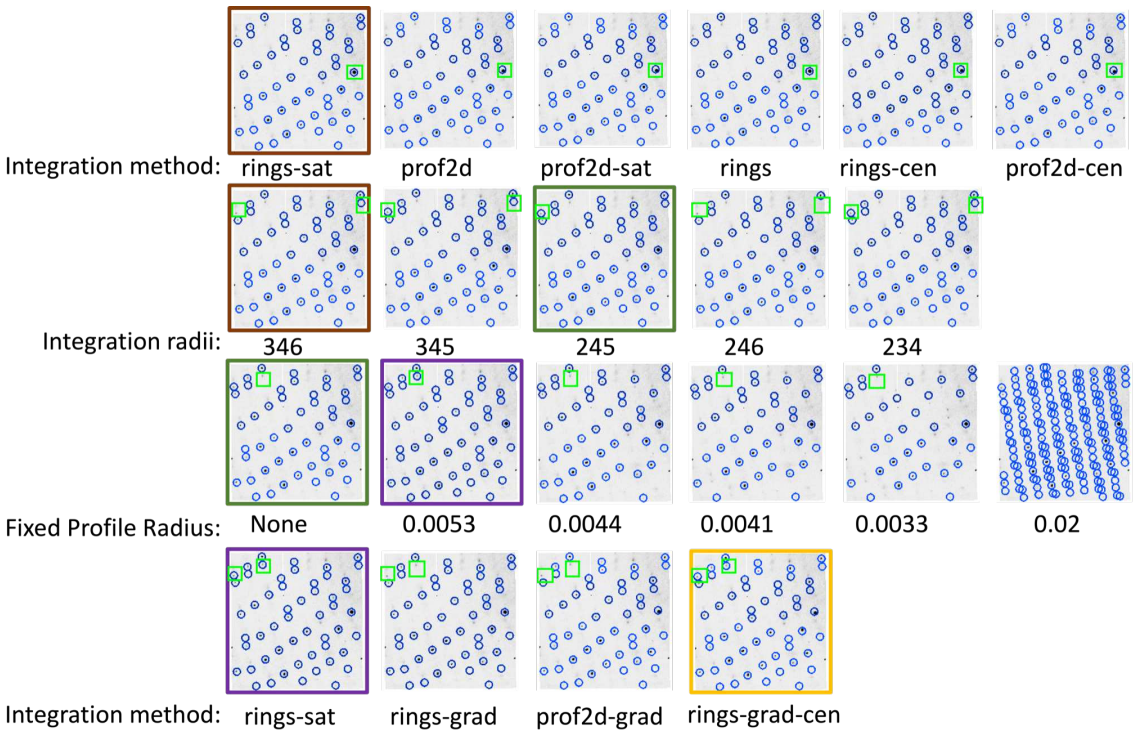

Figure 64. Indexing optimization affect on indexing

For the indexing optimizations described in section 3.3, a single asic for a single pattern showing the change in reflections for each of the 18 conditions (see table 7). Each row in the figure corresponds to a parameter being changed (row labels) and the value for that parameter is given below the pattern. Green squares highlight peaks that are changing in a particular row. Borders indicate duplicated patterns. The brown border is from the initial indexing (condition 0) used as the reference for both the first and second rows. The green border for integration radius 245 (condition 7) was used as the reference for the profile radius screen (all of which used integration radius 245 instead of the original 346). The purple border for fixed profile radius 0.0053 (condition 10) was used as reference for the final row. The gold border for rings-grad-cen (condition 17) is the condition used for further analysis.

different indexing condition. Figure 64 shows a single asic from a single frame for the different indexing condition with colored boxes showing the ones that are duplicated for comparison. Green highlights show peaks that change across a row.

The takeaway message from this brief review of PSII indexing optimization is that seemingly small changes in indexing parameters can change not only the integration of peaks but also the indexing solution for a frame, in turn changing statistics such as the unit cell volume or diffraction resolution limit.

4.3.2 Unit Cell and Resolution Subsets

The remainder of the chapter uses PSII data from LCLS November 2016. Recall that unit cell oscillations have been observed in this dataset (see section 1.4.2). Seven initial subsets were created based on fig. 65a. Three of them targeted the three visible clusters at small cell low resolution (SL), small cell high resolution (SH), and big cell low resolution (BL). The four others considered only one parameter with low resolution (LR), high resolution (HR), small cell (SC) and big cell (BC). The cutoff criteria are given in table 12 and were chosen to give about 20,000 patterns in each bin.

One limitation of this analysis is that the indexing is no-cell indexing. While the a axis was limited to a PSII range of 125-145 Å for all criteria, the other unit cell parameters were not limited. This means that some patterns included in merges may not have even been in the correct lattice. This is one motivation for the reanalysis presented in section 4.3.3. However, the majority of the analysis was done on these initial seven subsets since the lack of unit cell restrictions was not noticed until later.

Figure 65b shows initial $CC_{1/2}$ values for the seven subsets. Unsurprisingly, the low resolution subsets only go to low resolution. The small cell (SC), small cell high resolution (SH), and high resolution (HR) subsets are the more interesting. The high

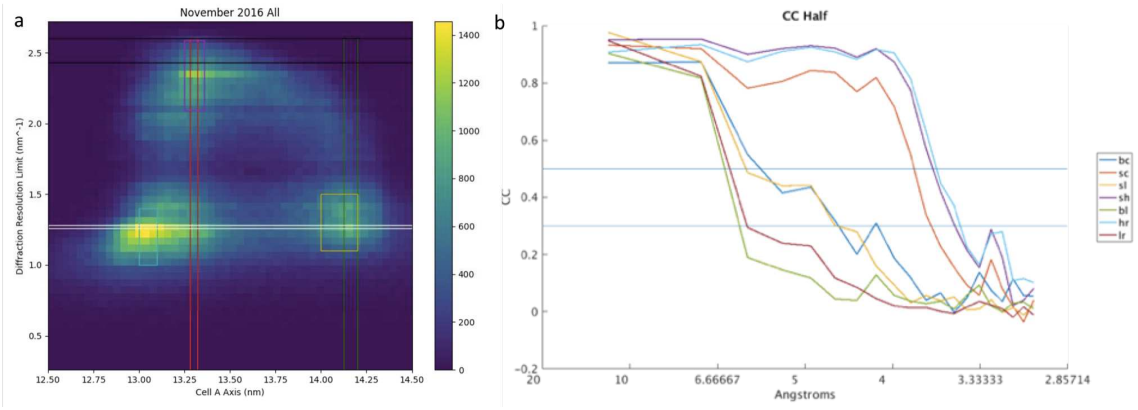

Figure 65. Unit cell and resolution subsets

For LCLS November 2016 PSII data, a large unit cell variation was observed. To determine whether it was better to take a small range of unit cell or a larger range with just higher resolution seven subsets were created, with details in table 12. Three of the sets target the major clusters in (a) with sl referring to the small cell low resolution cluster (teal in (a), yellow in (b)), sh referring to the small cell high resolution cluster (purple in (a) and (b)), and bl referring to the big cell low resolution (yellow in (a), green in (b)). The other four subsets take low or high, unit cell or resolution. hr is high resolution (black in (a), light blue in (b)), lr is low resolution (white in (a), dark red in (b)), sc is small cell (red in (a) and (b)), and bc is big cell (green in (a), blue in (b)).

resolution dataset does better than the small cell high resolution dataset although not by much. This is a little surprising since it implies it's better to include high resolution patterns regardless of the spread in unit cell. However, that trend is reversed in the reanalysis in section 4.3.3.

4.3.3 Additional Subsets

After *DatView* was more fully developped, LCLS November 2016 PSII data was reexamined to further explore subset selection in general. The previous subsets were based solely on the unit cell a axis and/or diffraction resolution limit and covered all

Table 12. Unit cell and resolution subsets

Dataset	A $\geq$	A $\leq$	R $>$	R $<$	Count
Small Cell (SC)	13.28	13.32	-	-	20373
Big Cell (BC)	14.125	14.2	-	-	20332
Small Cell Low Res (SL)	13	13.1	1	1.4	20017
Small Cell High Res (SH)	13.25	13.36	2.09	-	20291
Big Cell, Low Res (BL)	14	14.2	1.1	1.4	20461
High Res (HR)	12.5	14.5	2.42	-	19889
Low Res (LR)	12.5	14.5	1.25	1.29	19890

Cutoffs for the seven unit cell resolution subsets. When no restraint is given, all values are accepted and the full range is different than that shown in fig. 65a which was trimmed to show the range most relevant to PSII. Count is the number of patterns matching the criteria. Resolutions are in nm^{-1}

indexed patterns. Reanalysis began with the set of orthorhombic patterns within ± 10 Å of 135, 228, and 310 for the a, b and c axes respectively.

Three of the previous subsets were recalculated: the high resolution, small unit cell, and small unit cell high resolution datasets. The other four subsets were dropped because they were much lower resolution and aren't expected to improve merge quality. Three additional subsets based on correlations were created based on results from chapter 7 and literature. An overview of the subset generation is presented in fig. 66, summarized in table 13, and described in the following paragraphs.

The unit cell criteria is similar to a common unit cell criteria examined in traditional crystallography with programs like *BLEND* (Foadi et al. 2013). The diffraction resolution limit criteria remains in part because the previous analysis included it and it was the best, and also because results in chapter 7 showed it was similar to the "best pattern" criteria used in (Ayyer et al. 2016). The combination subset was also maintained, although using the total unit cell volume instead of just the unit cell a axis.

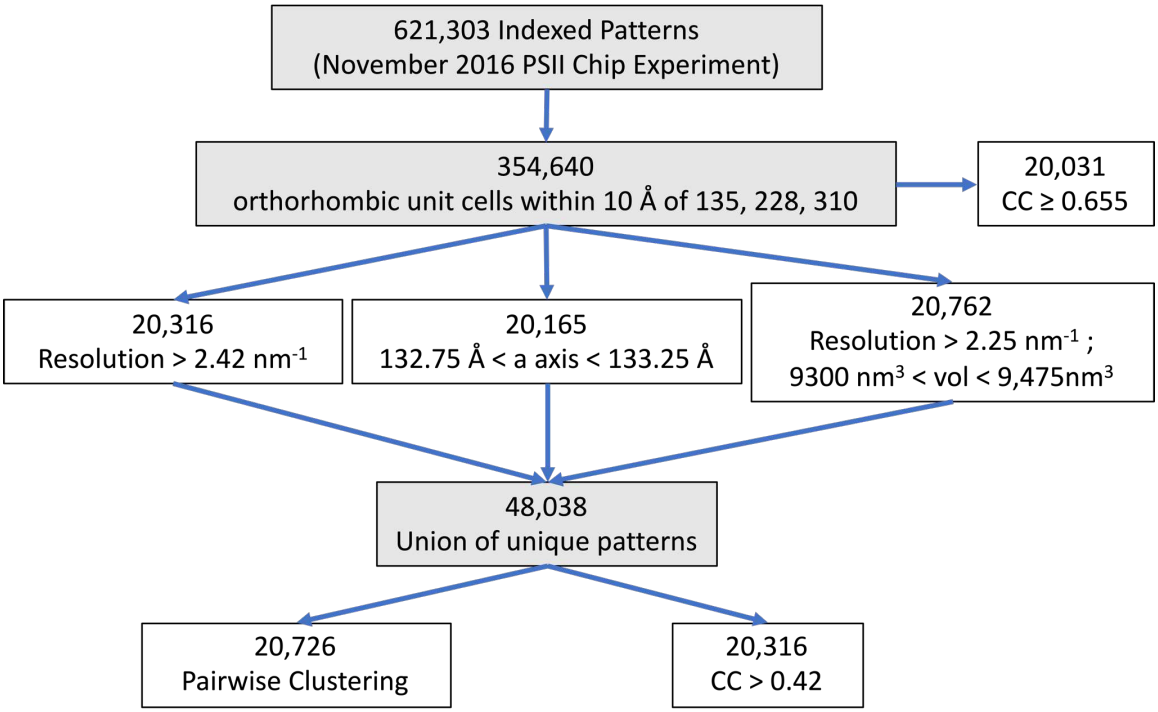

Figure 66. Additional subsets overview

A flow chart showing the 6 subsets (white background) in the reanalysis of LCLS November 2016 PSII data. First, the initial set of patterns was limited to the 354,640 patterns with orthorhombic unit cells within 10 Å of 135, 228, and 310 for the a, b, and c axes respectively. Then, the high resolution, small unit cell, and small unit cell high resolution subsets were recreated similar to the initial sets. The union of these sets was used for pairwise correlation and correlation to merge analysis to create the two subsets on the final row. A merge was also created from the 354,640 patterns for the correlation to merge set appearing on the second row.

The union of the three basic unit cell subsets was merged with *process_hkl* (see chapter 5) because *process_hkl* can output the per pattern scale and correlation to the merge. This correlation to merge value was used to create the CC small dataset, standing for the cross correlation to the small merge. This is similar to the relative anomalous correlation coefficient used in (Liu et al. 2012; Liu, Liu, and Hendrickson 2013) and was also chosen because of results in chapter 7.

Table 13. Additional subsets

Subset	Criteria	Matching	Average $CC_{1/2}$	High Res $CC_{1/2}$
Resolution	resolution $\geq$ 2.42 nm^{-1}	20,316	0.814	0.326
Unit Cell	132.75 Å < a axis < 133.25 Å	20,165	0.727	0.187
Res. Cell	9,300 nm^3 < cell volume < 9,475 nm^3, resolution $\geq$ 2.25 nm^{-1}	20,742	0.827	0.364
CC Small	CC $\geq$ 0.43	20,468	0.835	0.367
CC All	CC $\geq$ 0.655	20,031	0.560	0.072
CC Pair	-0.42 < xy angle < -0.075, -0.32 < z angle < 0.14	20,726	0.802	0.278

Cutoffs for the six subsets in the reanalysis, shown in fig. 66. The Average $CC_{1/2}$ was calculated with 30 resolution shells from 20 Å to 3.5 Å, and the $CC_{1/2}$ of the highest resolution shell is from 3.50-3.54 Å. Count is the original number matching, which was randomly reduced to 20,000 for further analysis.

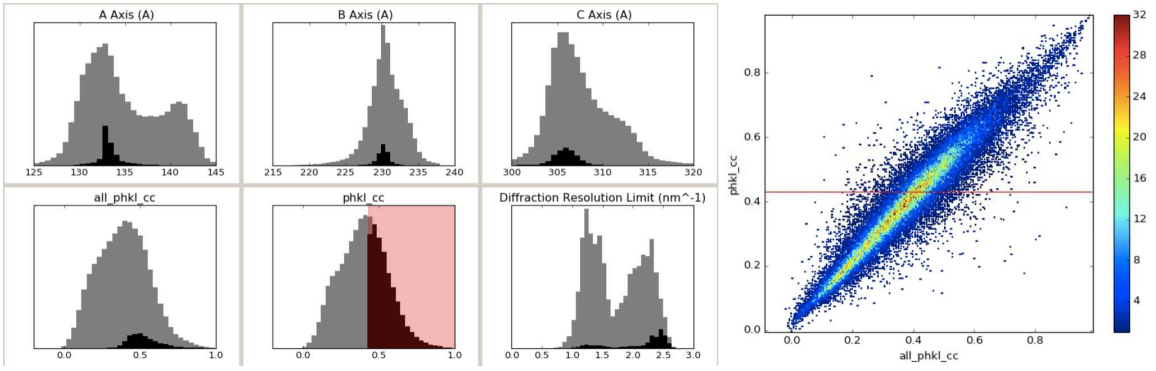

Figure 67. Correlation to merge subsets

Histograms for the correlation to merge all subset with the patterns for correlation to merge small selected. The 2D histogram shows the change in CC by pattern for comparison to merge all (x axis) and comparison to merge small (y axis).

One weakness of the CC Small subset is it only considers the 48,038 patterns from previous criteria. For comparison, *process_hkl* was also run on the 354,640 patterns with reasonable PSII unit cells. The best correlating patterns were again selected, and this subset is referred to as CC All because it was the correlation to the merge with all reasonable patterns. A comparison of patterns included in the CC Small versus CC All datasets is shown in fig. 67. In the histograms, the 354,640 reasonable patterns are the semi-transparent full dataset and the selection is the CC All dataset. The first histogram on the second row shows the CC to the all merge. The CC small values remain similar but there are many more patterns that correlate just as well or better (therefore the cutoff for the CC all subset is higher than that of the CC small subset to keep the number of selected patterns similar). The 2D histogram in the figure shows the change in CC by frame when correlated with the small merge (y axis) or all merge (x axis). The trend is overall linear, but shows that the merge small and merge all are different enough to alter the CC values for many frames.

Pattern selection in literature has also used clustering on the results of pairwise correlations, like the dendrograms used in *BLEND* (Foadi et al. 2013). However, pairwise clustering is order N^2 in both memory and time, meaning that the number of calculations and space to store them increases as the square of the number of patterns considered. Running pairwise correlation on the 354,640 reasonable patterns was therefore not feasible with the available computational resources.

Instead, clustering software from (Diederichs 2017) was run on the 48,038 union subset. This software takes as input a resolution range for the comparison and the number of dimensions to cluster in. Clustering was performed in three dimensions with a resolution range of 3.5 to 5. With three dimensions, the x, y, and z values and the $x - y$ angle and xy plane to z angle are all output for each pattern. The visible

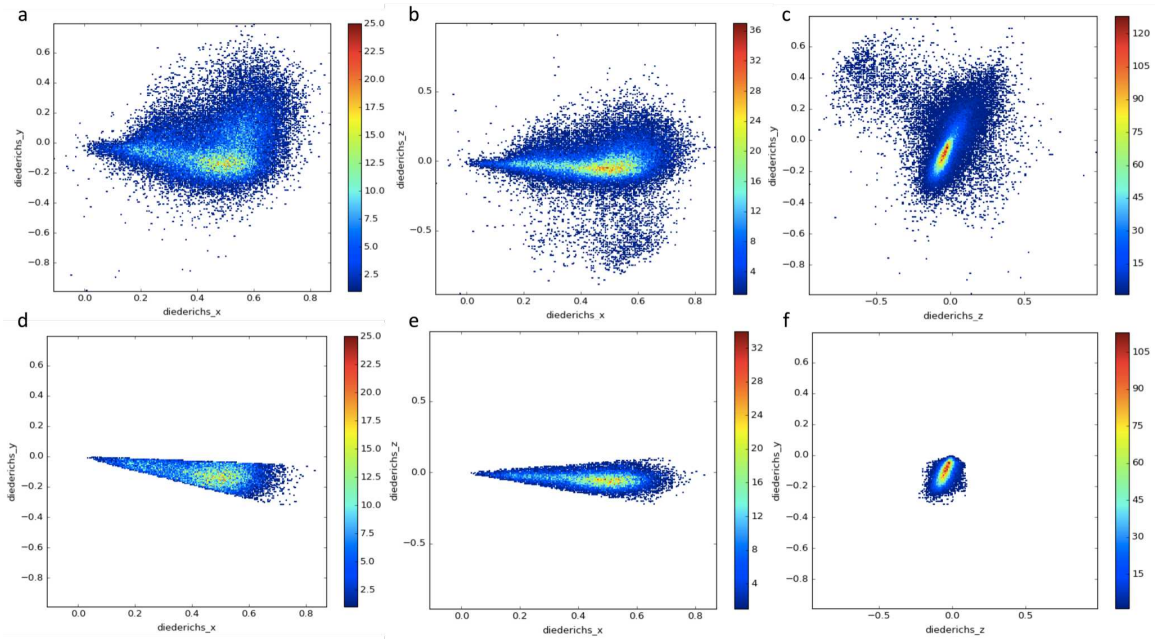

Figure 68. Output of pairwise clustering

Pairwise clustering with Diederich's algorithms was performed in 3 dimensions on the union on unique patterns. The plots show the output of clustering where each point is colored by the number of patterns. The clustering algorithm determines x, y, and z values for each pattern which are plotted on the xy, xz, and zy planes before selection (a-c) and after selecting (d-f). Selection was made on angles (between the x and y values and the xy plane and the z value), see table 13. Selections were made visually to select the largest cluster.

difference between the points corresponds to the difference between the patterns. None of these input parameters were optimized, and the output warned that a higher number of dimensions should be used. So, these results should be considered preliminary and much more work is required to fully utilize this software.

The clustering results are shown in fig. 68 along with the areas selected for the subset. According to (Diederichs 2017), the differences in angle correspond to systematic error between patterns and the length of the vector to random error between patterns. Therefore, the angles were used as selection criteria with the largest visual cluster targeted.

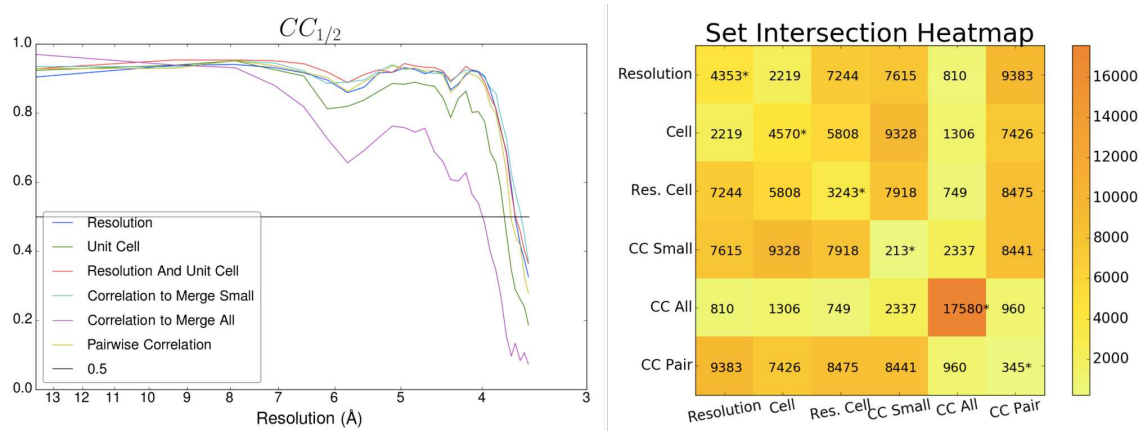

Figure 69. Additional subsets statistics

$CC_{1/2}$ plot and *setheatmap.py* plot (showing the intersection of different input files, with the diagonal showing the number unique to each condition) for the additional subsets described in table 13.

The $CC_{1/2}$ and *setheatmap.py* results for the six clusters are shown in fig. 69, and overall statistics were given in table 13. In the reanalysis, the unit cell and resolution combo dataset had better CC values than the high resolution only dataset, reversing the trend observed in the initial subsets. Since the reanalysis limited the starting patterns to PSII unit cell parameters, it is more reliable. The current outlook for unit cell versus diffraction resolution limit criteria is that the range of unit cell does matter, since the combo subset did better than just the resolution subset, but that keeping the higher resolution patterns helps since both the high resolution and combo datasets outperformed the unit cell subset.

Including the CC subsets, it's interesting that the best subset overall was actually the correlation to the small merge and the worst subset was correlation to the all merge. With a merge starting with a good subset of patterns, the CC is therefore useful, but a merge considering all patterns may be nosier and therefore less useful. Or, the correlation to merge, since the resolution considered is unspecified, could be dominated by low resolution high intensity peaks, and more control over the resolution

range considered, for example targeting only high resolution peaks, could be more useful. The pairwise correlation subset was intermediate, but since the analysis was not optimized it is hard to directly compare.

This analysis considers the case were only a subset of the data is processed. In that case, these results and those presented in chapter 7 suggest the criteria used to make the selection can have a big impact. However, many datasets can be processed as a whole and that is not compared in this analysis. In general, including all possible patterns has given better statistics than including only a subset, with rare exceptions.

This chapter has presented *DatView* as a visualization tool to improve the loading speed of large datasets and export smaller datasets for further analysis. *DatView*'s utility in indexing optimization was shown in section 4.2. Subset selection was covered in sections 4.3.2 and 4.3.3. The subsets of section 4.3.2 are used throughout chapters 5 and 6. Subset selection for continuous diffuse scattering analysis is a major theme of chapter 7.

Chapter 5

MERGING

Merging is the process of combining indexing results from individual crystal lattices into a single whole. Usually, merging results in a table of Miller indices with a single integrated value for each index. Merging in 3 dimensions results in a 3D volume including values between Bragg peaks. Merging in 3 dimensions is described in chapter 7. This chapter focuses on merging of Bragg data.

There are two *CrystFEL* programs for merging a .stream file into a .hkl table. The older program is *process_hkl* and the newer program is *partialator*. *Partialator* must load the entire dataset into memory so it is limited in the size of stream files it can process. However, it has more options, its scaling algorithm is recommended over *process_hkl*'s, and it has conveniences like automatically producing half merges. Both programs can output statistics, but only *process_hkl* outputs the per-pattern scale and CC to the final merge. The next section of this chapter describes some common options of the merging programs in the context of optimizing merging. Merging statistics calculated with *CrystFEL* programs have been previously described in section 3.2.

5.1 Bragg Merging Optimization

Merging conditions were most fully screened with LCLS June 2012 PSII data (indexing conditions given in section 3.3). Section 5.1.1 describes the push-resolution and scaling options for *CrystFEL* merging programs and compares merges from two

of the indexing conditions. Section 5.1.2 compares the unit cell and resolution subsets of LCLS November 2016 (see section 4.3.2) with different merging parameters.

5.1.1 Indexing Affect on Merging

CrystFEL merging programs have a push-resolution (push-res) option. By default, all resolutions from all input patterns are included in the merge. The push-res option allows changing the resolution included for each pattern based on that pattern's diffraction resolution limit. A push-res value of 0 includes reflections up to the pattern's diffraction resolution limit (as calculated by *CrystFEL*, see section 3.1.2) but no further. Negative push-res values use less than the apparent diffraction resolution limit, and positive push-res values go past the pattern's diffraction resolution limit. A positive push-res value may be used because the diffraction resolution limit calculated by *CrystFEL* is conservative and can be impacted peak finding parameters. Push-res is also available at the indexing step which would prevent reflections outside of range from being stored at all. Applying push-res at merging is usually preferred because it reduces analysis time compared to rerunning indexing for each push-res condition.

Some push-res screens are shown in fig. 70. The basic trend follows expectations with increasing push-res values having data at higher resolutions. As in chapter 3, it's worth noting that the results will appear different for different ranges and bins of data. In fig. 70b, a push-res value of 1.0 is distinguishable from a push-res value of 1.5 or None. However, in fig. 70c, the three values are indistinguishable. Because fig. 70c was used to determine a push-res limit for future analysis, a value of 0.5 was picked under the assumption that the final three values were equivalent.

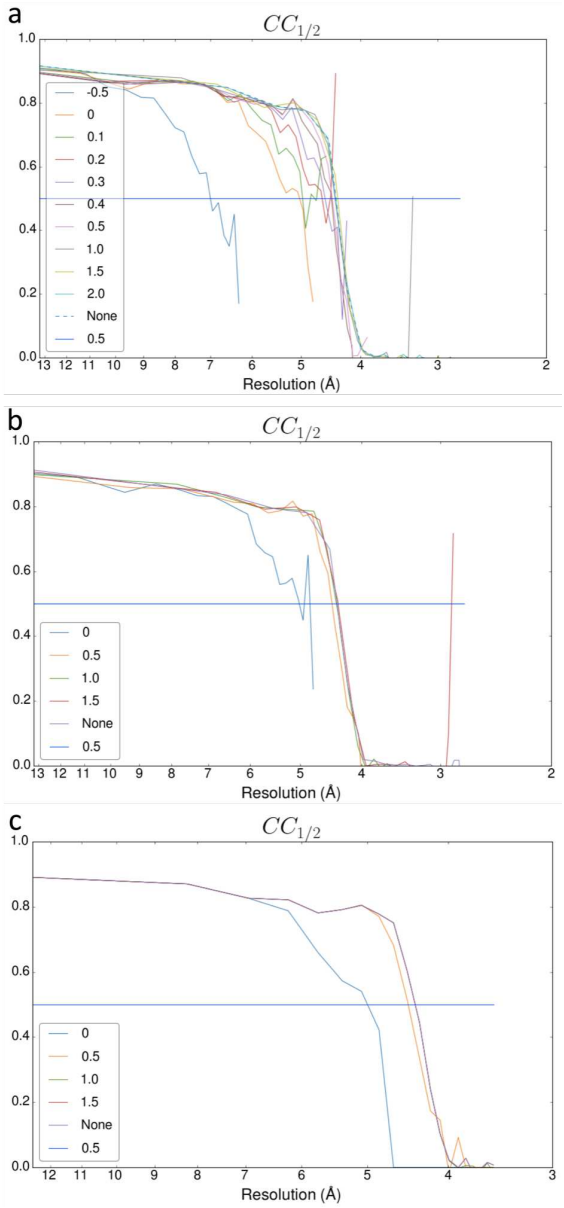

Figure 70. Push-res screen

Different push-res settings were screened with *process_hkl* on the best 2,000 pattern described in section 3.3, (see also table 7). (a) A push-res screen for indexing condition 7. Statistics were calculated in 20 shells over the data range. (b) A push-res screen for indexing condition 17. Statistics were calculated in 20 shells over the data range. (c) Same as (b) except the 20 shells were calculated over the range 20 Å to 3.5 Å.

Table 14. Push-res Screen Statistics

Push-res	SNR	Rsplit	CC 1/2	CC *
None	2.055195	29.11	**0.9368891**	**0.9835733**
0	**3.697593**	**25.12**	0.9082246	0.9756564
0.5	2.577588	26.23	0.9307153	0.9818933
1	1.999293	29.89	0.9361165	0.9833638
1.5	2.055234	29.11	0.9368887	0.9835732

Statistics for the push-res screen for indexing condition 17 best 2,000 patterns with statistics calculated in 20 shells over the range 20 Å to 3.5 Å. Best statistics in column are bold.

The choice of push-res=0.5 for future screening was also supported by the overall statistics in table 14. While the CC values were best with no push-res or push-res=1.5, the R_{split} and SNR values get worse with increasing push-res. The question of which statistic(s) relate to electron density quality when comparing parameters is still an open question. Push-res=0.5 is a comprise between the various statistics, following the precedent of using indexing condition 17.

Process_hkl and *partialator* have different scaling algorithms. In *process_hkl*, the data is processed a second time and scaled to the initial model (White et al. 2012). In *partialator*, the scaling algorithm has been improved and multiple passes can be performed (White et al. 2016). For comparisons on this dataset, scaling is treated as either on or off, and the number of iterations of scaling in *partialator* is always set to 1.

Partialator and *process_hkl* were compared for indexing conditions 0 and 17 for all LCLS June 2012 PSII light data (see table 7). In fig. 71a,c,e, no scaling is used and three parameters are considered: 1) *partialator* (dashed lines) or *process_hkl* (solid lines), 2) push-res = None (first 4) or 0.5 (last 4) and 3) indexing condition 17 or 0. The combination of solid and dashed lines shows that without scaling, *partialator* and *process_hkl* are very similar if not identical. Figure 71b,d,f is similar except that all

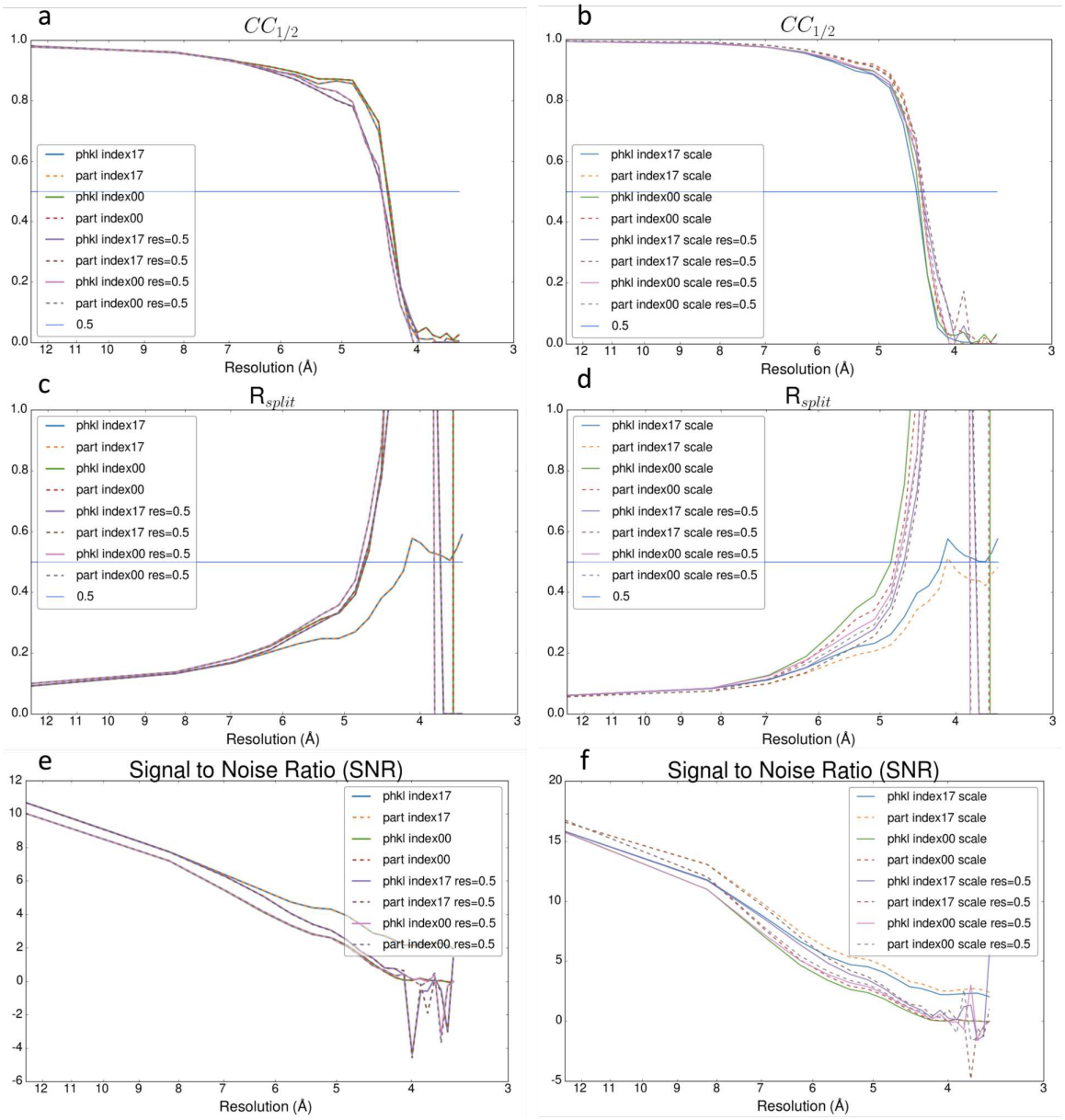

Figure 71. Partialator versus process_hkl

Statistics calculated in 20 shells from 20 Å to 3.5 Å for all light PSII patterns from LCLS June 2012 processed using indexing condition 0 or 17 (see table 7) with *process_hkl* (phkl, solid lines) or *partialator* (part, dashed lines). The first four lines are without push-res, and the last four lines are with push-res = 0.5. (a,c,e) No scaling. Under these conditions, *partialator* and *process_hkl* are nearly identical. (b,d,f) Scaling. *Partialator* (dashed lines) performs better than *process_hkl* (solid lines) for each condition.

8 conditions are scaled. Here, the trend with every statistic is that *partialator* has better results than *process_hkl*.

Merges were also run for the dark alternating patterns. With *partialator*, the light and dark are given together to *partialator* along with a file describing which group each pattern belongs to. This allows the patterns to be scaled equally and then split just before merging. For *process_hkl*, light and dark are processed separately. Table 15 gives the overall statistics for the 16 merges of the light dataset (first half) and the 16 merges of the dark dataset (second half). Each half has the best three values bolded and is sorted by CC^*.

From the table, a few trends are noticeable. First, scaling is generally better than not scaling, even with *process_hkl*. Second, *partialator* with scaling is better than *process_hkl* with scaling. Third, no push-res is usually better than push-res=0.5. Fourth, *partialator* and *process_hkl* are not identical with no scaling and push-res since the statistics vary a little. As in table 8, a comparison between indexing conditions 0 and 17 depends on which statistic is examined with indexing condition 0 having the best CC values but indexing condition 17 having the best SNR and R_{split}. However, comparing indexing conditions does depend on the merging conditions since for light, indexing condition 17 had a better CC than indexing condition 0 with *partialator* scaling, push-res=0.5 but for dark the trend is reversed. Therefore, the question of which indexing condition is continued in section 6.2.

5.1.2 Subset Affect on Merging

Partialator is more convenient to run than *process_hkl* because it automatically produces half merges for statistics. With the results in section 5.1.1 showing the without

Table 15. Merging screen statistics

Indexing	Partialator	Scale	PushRes	SNR	R-Split	CC 1/2	CC *
0	yes	yes		2.869286	**12.81**	**0.9954733**	**0.9988651**
17	yes	yes		**5.134514**	**10.01**	**0.9950649**	**0.9987624**
17	yes	yes	0.5	**5.116542**	**11.86**	**0.9934534**	**0.9983566**
0	yes	yes	0.5	4.461673	13.65	0.9930472	0.9982542
17		yes		4.589481	15.5	0.9927783	0.9981864
0		yes		2.617279	21.76	0.9921279	0.9980222
0		yes	0.5	4.139582	18.53	0.9849788	0.9962091
17				3.845181	17.13	0.9848923	0.9961871
17	yes			3.845181	17.13	0.9848923	0.9961871
17		yes	0.5	**4.684846**	17.4	0.9843188	0.9960409
0				2.10292	22.5	0.9824444	0.9955624
0	yes			2.10292	22.5	0.9824444	0.9955624
0			0.5	3.111733	30.28	0.9262322	0.9806649
0	yes		0.5	3.114466	30.2	0.926095	0.9806272
17			0.5	3.380757	27.92	0.9251559	0.9803689
17	yes		0.5	3.407241	27.9	0.9242782	0.9801272
0	yes	yes		2.901646	**12.53**	**0.9960058**	**0.998999**
17	yes	yes		**5.161059**	**9.71**	**0.995088**	**0.9987682**
0	yes	yes	0.5	4.592239	13.27	**0.9938827**	**0.9984648**
17	yes	yes	0.5	**5.200327**	**11.54**	0.993615	0.9983974
17		yes		4.633009	15.13	0.9930836	0.9982634
0		yes		2.645338	21.14	0.9930586	0.9982571
17				3.857275	16.91	0.9860147	0.9964728
17	yes			3.857275	16.91	0.9860147	0.9964728
0		yes	0.5	4.264271	18.27	0.9858328	0.9964266
17		yes	0.5	**4.873651**	17.23	0.9851685	0.9962574
0				2.10725	22.06	0.984157	0.9959996
0	yes			2.10725	22.06	0.984157	0.9959996
17			0.5	3.586901	28.39	0.9240806	0.9800727
17	yes		0.5	3.534214	28.37	0.9234105	0.9798879
0			0.5	3.310695	30.92	0.921128	0.9792573
0	yes		0.5	3.329659	30.81	0.9209268	0.9792017

Statistics for all light (first half of table) and dark (second half of table) PSII patterns from LCLS June 2012 processed using indexing condition 0 or 17 (see table 7) with *process_hkl* or *partialator* (second column), with none or one iteration of scaling (third column) and without push-res or with push-res = 0.5 (fourth column), with light and dark sorted separately on CC* (final column) with the best 3 values (for light and dark separately) in each statistic column bolded.

scaling *process_hkl* and *partialator* appear identical and with scaling *partialator* is better than *process_hkl* only *partialator* is used for merging in this section.

Ideally, *partialator* would be run on the union of the seven subsets given in table 12 with the external configuration file assigning each pattern to its appropriate subset(s). Unfortunately, *partialator* only supports each pattern belonging to a single subset, and the subsets in table 12 overlap. So, *partialator* was run separately for each subset.

Four different conditions were considered toggling scaling and push-res. A push-res value of 0.5 was kept as an arbitrary decision to keep things similar to previous analyses. Figure 72 shows the CC for all 28 merges with coloring by subset and line style by merge. One nice feature of the graph is that the overall trend between subsets remained the same regardless of merging condition. This is preferable to the trend of indexing conditions in table 15 where merging conditions could change the order because it supports the assumption that subset and merging parameters are independent from each other.

Simplified versions of fig. 72 are given in fig. 73. fig. 73a shows that scaling is better than not scaling for relevant resolutions with all seven subsets. fig. 73b compare push-res=none with push-res=0.5. Here there isn't as clear a trend although push-res usually doesn't help at high resolutions.

As in section 4.3.2, the high resolution subset appears the best, at least at high resolutions, regardless of the merging parameters. However, as mentioned then, these subsets contained a wide range of unit cells. The 28 merges described here are compared again in section 6.3.

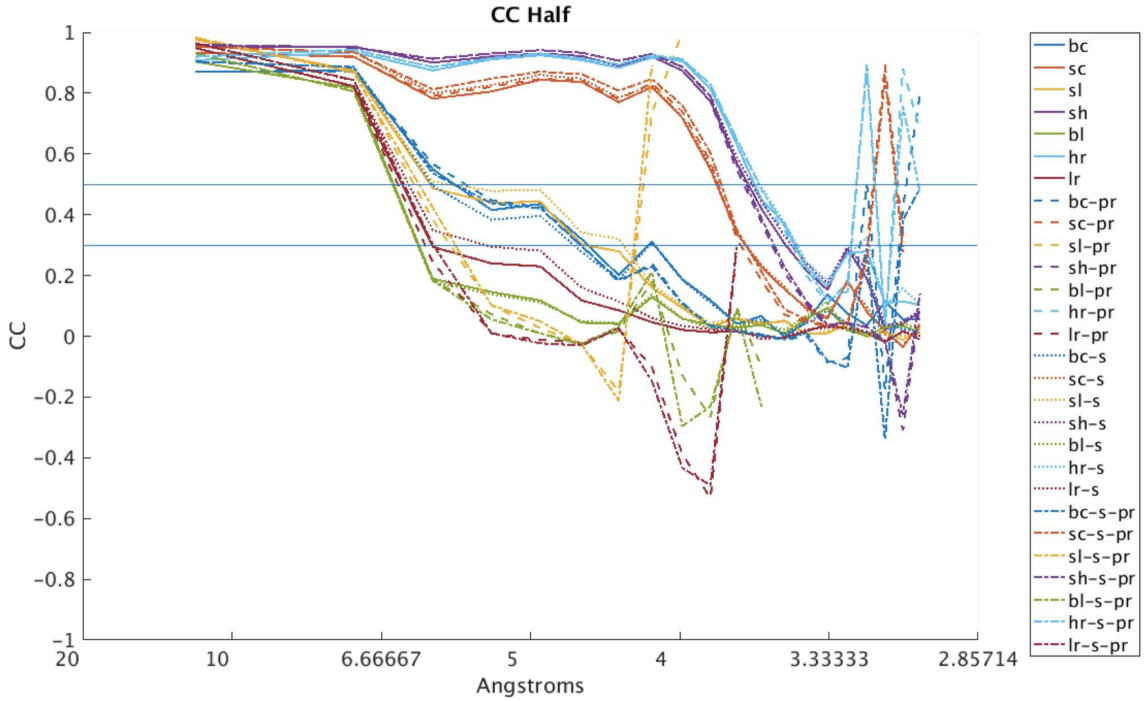

Figure 72. Merging comparisons by subset

For the seven subsets (Big Cell (BC), Small Cell (SC), Small Cell Low Resolution (SL), Small Cell High Resolution (SH), Big Cell Low resolution (BL) High Resolution (HR) and Low Resolution (LR)) described in table 12, merging statistics with *partialator*. Solid lines had no scaling and no push-res, Dashed lines had push-res=0.5, dotted lines had scaling, and dot-dash lines had both. Dark blue is big cell, red is small cell, yellow is small cell low resolution, purple is small cell high resolution, green is big cell low resolution, cyan is high resolution, and dark red is low resolution. Statistics were calculated from 20 Å to 3.0 Å.

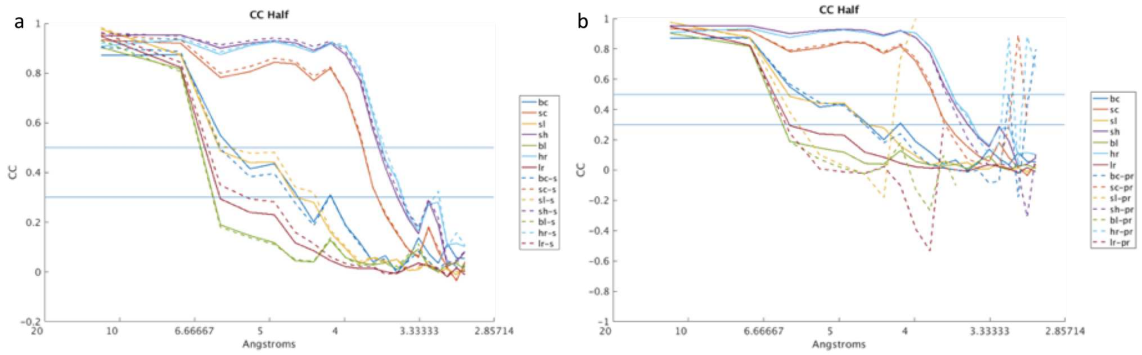

Figure 73. Merging comparisons by subset

A simplified version of figure 72 with (a) comparing only not scaled (solid) with scaled (dashed) and (b) comparing no push-res (solid) with push-res (dashed). Scaling seems generally better, whereas push-res depends on the resolution and subset.

Chapter 6

PHASING AND REFINEMENT

The final steps in a traditional crystallographic analysis pipeline are phasing and refinement. The .mtz files from merging contain intensities at each Miller Index. These intensities, together with phases, are inverse Fourier Transformed to obtain an electron density. Several methods of obtaining phases exist. In this dissertation, molecular replacement is used. Molecular replacement uses phases from an existing model for an initial estimate. Phases are then refined with refinement programs that iteratively update positions of atoms and phases to better match.

There are two main software suites that contain many crystallographic analysis programs and a GUI interface: *ccp4* (Winn et al. 2011) and *phenix* (Adams et al. 2010). The analysis presented here uses *phenix*, specifically *phenix.phaser* and *phenix.refine*. It is not intended to be a comprehensive review of crystallographic data analysis, but instead focuses on the impact of SFX analysis and subsets on phasing and refinement statistics. Section 6.1 introduces the phasing and refinement statistics used. Then section 6.2 covers phasing and refinement screens comparing the indexing results introduced in section 3.3. Finally, phasing and refinement output of the subsets given in section 4.3.2 are compared in section 6.3.

6.1 Output

Phenix.phaser has two interfaces: a simple one and a full one. It takes as a minimum input a PDB search model and a .mtz intensities file. The PDB search file

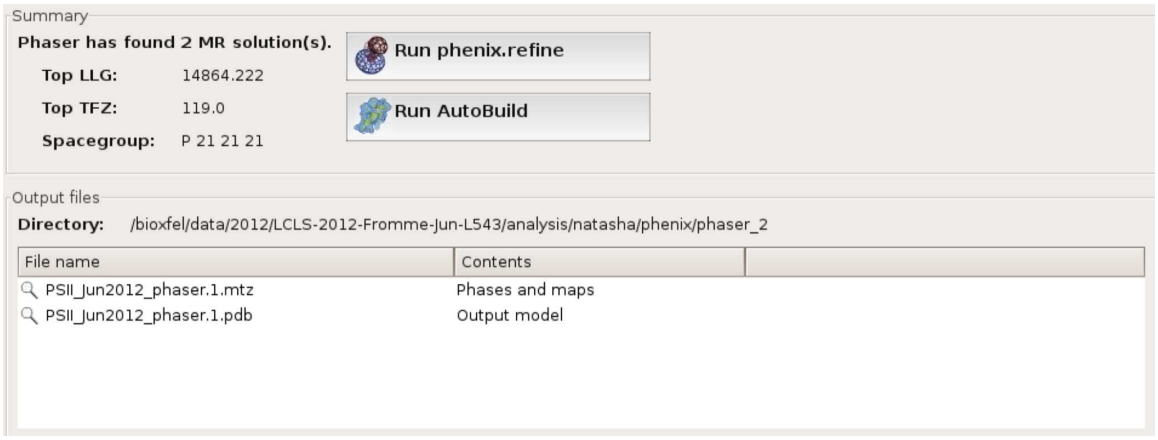

Figure 74. Phaser output

A screen shot of *Phaser* output for LCLS June 2012 PSII light data merged with *partialator* defaults (no scaling, no push-res) using MR.1.pdb as a search model.

can be used for the sequence, requiring a percent sequence identity argument, or a separate sequence file can be provided. The full interface provides more options, such as the option to keep the ligands in the search model which is normally off. PSII must be phased with ligands, so only the full interface can be used.

A screenshot of typical *phenix.phaser* output is shown in fig. 74. There are two main statistics, the LLG (log likelihood gain) and TFZ (translation function z score) which are used internally to compare the best placements. Higher values are better for both statistics, but they are not necessarily comparable between different *phenix.phaser* outputs. Also output are an .mtz and .pdb file for use with refinement. When multiple solutions are found, multiple .mtz and .pdb files can be output.

Refinement takes as minimum input files an .mtz and a .pdb file. For PSII, a .cif file may also be needed that gives constraints for ligands. Refinement is iterative and users can select which refinement algorithms to run. By default, the refinement algorithms are XYZ coordinates, real space, occupancies, and individual B-factors. Users can also select a resolution range.

Before and after refinement:

	Starting	Final
R-work	0.2743	0.2327
R-free	0.2598	0.2574
Bonds	0.024	0.005
Angles	2.955	0.781

X-ray statistics by resolution bin:

	R-work	R-free	%complete	FOM	Phase erro	Scale facto	#work	#test
62.0543 - 6.1320	0.1951	0.2104	99.9%	0.77	26.71	1.00	22000	178
6.1320 - 4.8677	0.2700	0.3300	100.0%	0.56	47.22	1.02	21442	174
4.8677 - 4.2526	0.3203	0.3561	99.7%	0.58	43.13	1.00	21255	174
4.2526 - 3.8638	0.5240	0.4371	36.0%	0.52	54.42	0.65	7656	61

Figure 75. Refinement output

A screen shot of *phenix.refine* output for LCLS June 2012 PSII light data merged with *partialator* defaults (no scaling, no push-res) using MR.1.pdb as a search model and default refinement settings.

The main output window for refinement is shown in fig. 75. The crystallographic R factor is defined as

$$R = \frac{\sum ||F_{obs}| - |F_{calc}||}{\sum |F_{obs}|} \tag{6.1}$$

where F_{obs} and F_{calc} are the observed (from measured intensities and current phases) and calculated (from atomic positions) structure factors respectively. The structure factor contains both an amplitude (proportional to the measured intensity values) and phase. The R factor is therefore a measure of how well the atomic positions fit the Bragg reflection data. R values of 0.4 - 0.6 can be obtained from a random model (Laskowski and Swaminathan 2013) and therefore models with high R factors are not reliable.

The reflections in the dataset are randomly divided between working reflections and free reflections. The R factor calculated from the working reflections (R_{work}) is used by the program as an optimization function for refinement algorithms. R_{work} should

be lower after refinement. The portion of the reflections that are randomly excluded from the optimization are called free reflections. The flags for which reflections are free are stored in .mtz format and can be passed to refinement so that R_{free} (R factor calculated over the free reflections) can be compared between different refinements. R_{free} is usually higher than R_{work}, but should also decrease during refinement. If R_{free} increases, that is a sign of overfitting (meaning the electron density is biased by the starting model). Overfitting can occur when a high resolution model is used with low resolution data.

6.2 SFX Analysis Influences on Phasing and Refinement

This section focuses on phasing and refinement of LCLS June 2012 PSII data, with an emphasis on indexing conditions 0 and 17 that were introduced in section 3.3 and are summarized in table 7. Three different PDB files were screened for phasing. The highest resolution model of PSII is PDB entry 3wu2 (Umena et al. 2011) which is an update from the initial deposit (PDB entry 3arc). However, 3wu2 used PSII from a different organism with different crystal additives and packing compared to the current data. An internal PDB (MR.1.pdb) and .cif file appropriate to the crystals used in this dataset was provided by Dr. Raimund Fromme. However, 3wu2 was also used for reproducibility.

Two variations of 3wu2 were created with waters (because the current data is not high enough resolution) and some ligands removed. In the first version, HTG (HEPTYL 1-THIOHEXOPYRANOSIDE), RRX ((3R)-beta,beta-caroten-3-ol) and UNL (unknown ligand) were removed because they were not included in .cif file for this dataset. The second version also had those ligands removed, and additionally

removed GOL (GLYCEROL) and LMG (1,2-DISTEAROYL-MONOGALACTOSYL-DIGLYCERIDE) because those ligands were not in MR.1.pdb (see the text based comparison of the files in table 27 in appendix B).

Table 16. Search model comparison

Chain		3wu2			3arc	
	Alignment	Atoms	RMSD	Alignment	Atoms	RMSD
A	1779.6	334	0.044	1779.6	334	0.001
B	2674.1	504	0.074	2688.5	504	0.001
C	2400.8	451	0.051	2404.4	451	0.001
D	1806.9	341	0.055	1833.9	342	0.001
E	417.2	81	0.164	420.8	81	0.001
F	177.6	34	0.121	177.6	34	0.001
H	325.1	63	0.055	334.5	65	0.001
I	193.2	36	0.167	195.6	38	0.164
J	188.4	36	0.345	193.8	38	0.001
K	186.8	37	0.064	192.8	37	0.001
L	188.4	37	0.101	188.4	37	0.001
M	160	33	0.083	167.2	34	0.008
O	1217.5	239	0.16	1223.5	243	0.001
T	157.1	30	0.103	157.1	30	0.001
U	492.8	97	0.101	492.8	97	0.001
V	692.7	137	0.057	707.7	137	0.001
Y	137.2	27	0.117	143.8	29	0.001
X	191.7	38	0.107	192.9	39	0.001
Z	309.3	62	0.231	319.5	62	0.001
All but XYZ		2490	0.099		2502	0.022

Output from *chimera*'s three dimensional alignment tool by chain for MR.1.pdb to PDB entry 3wu2 (first three columns) or PDB entry 3arc (second three columns). Only one monomer (capital chains) is considered. The calculation is performed with all paired atoms, and a subset of pruned ones. The above table shows the results for the pruned pairs, which was identical to the subset (no atoms were pruned) except for comparison to 3wu2 O chain (total pairs was 343 with RMSD of 0.392) and 3wu2 with all chains but X, Y, and Z (total pairs was 2494 with RMSD 0.149). The alignment score is sequence alignment.

In addition to the text based comparison, *chimera* (Pettersen et al. 2004) was used to align MR.1.pdb with wu2 and 3arc. *Chimera* operates on chains and each chain

was compared individually. Due to the large number of chains, it was not possible to select all chains for simultaneous comparison in *chimera*, so all the chains except X, Y, and Z were compared. *Chimera* outputs a sequence alignment score, the number of atoms used, and the RMSD. Results are summarized in table 16. From table 16, MR.1.pdb is almost identical to 3arc (the previous version of 3wu2). It's interesting to note how different 3wu2 is from MR.1.pdb (and therefore 3arc) suggesting there were many changes between the two versions.

All three PDBs were used for initial phasing. The light and dark merges from indexing conditions 0 and 17 merged with *partialator* scaling were used initially because the merging condition with *partialator* scaling had better CC values (see table 15). Rigid body refinement (with each monomer defined as a rigid body) was done to get R factors for comparison.

The refinement resolution range was limited to 20 Å to 4.3 Å. The high resolution cutoff was chosen based on the CC^* values for the four merges, given in table 17. Phasing and refinement statistics are summarized in table 18.

As an arbitrary convention, the electron density of pheophytin 409 from chain A is always used for electron density images. It is convenient for comparison of electron density quality because changes between the light and dark structures are not expected and the hollow ring is easily comparable. Figure 76 shows the electron densities for the 12 conditions of table 18.

The visible difference is between the indexing conditions (top and bottom halves of the figure) rather than the search model (columns of the figure). Following the R

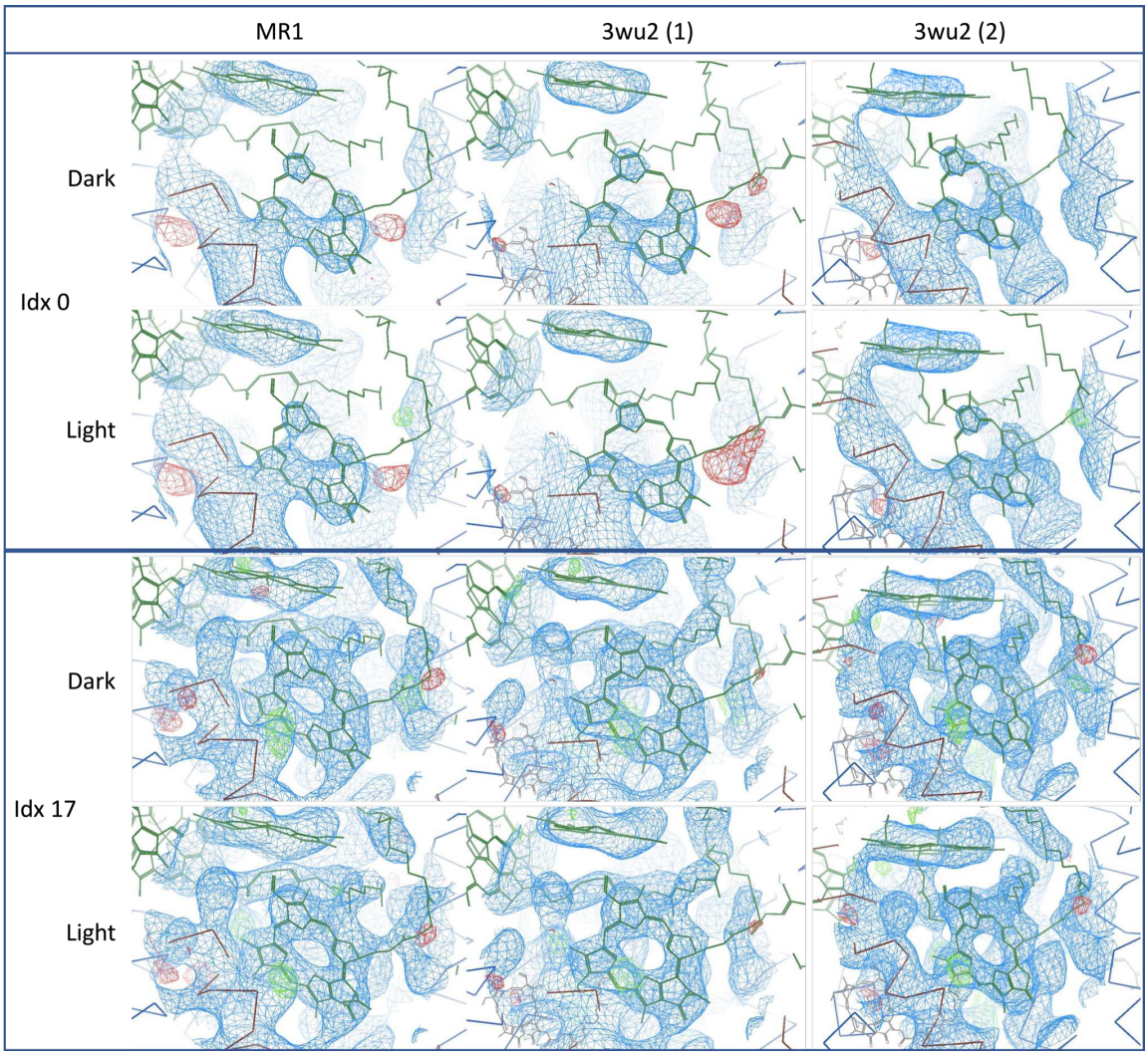

Figure 76. Electron densities by dataset, indexing, and search model

A view of the electron density around the pheophytin molecule of chain A with indexing condition 0 (top box) or 17 (bottom box) (see table 7 for indexing conditions), dark (first rows within each box) or light datasets (second rows within each box) and search model MR.1.pdb (first column), PDB entry 3WU2 with waters, RRX, UNL, and HTG removed (second column) and 3wu2 with waters, RRX, HTG, UNL, GOL, and LMG removed (third column). All electron densities come from refining with rigid body monomers (see table 18 for details). Phasing was performed with ligands.

Table 17. Resolution cutoffs by CC

Å	Dark idx 0	Dark idx 17	Light idx 0	Light idx 17
12.60	0.9987028	0.9982348	0.9982192	0.9984761
8.22	0.9976218	0.9970216	0.9970965	0.9975919
6.94	0.9953603	0.9956011	0.9950104	0.9950859
6.22	0.991405	0.9912746	0.9914245	0.9911147
5.73	0.9858594	0.9854525	0.9853753	0.9851656
5.36	0.9811704	0.9798526	0.9791442	0.9799826
5.08	0.9800765	0.9791275	0.9768475	0.9784833
4.84	0.9684824	0.9652283	0.9684639	0.9700434
4.65	0.9462533	0.9392322	0.9441836	0.9480972
4.48	0.9051785	0.8932211	0.8778869	0.890637
4.33	0.6810695	**0.6879664**	**0.6921353**	**0.704891**
4.20	**0.5121695**	0.4275852	0.4395869	0.4656308
4.09	0.1062756	0.3344954	0.2837345	0.2424301
3.99	0.178895	0.1690236	0.1167423	0.3035376
3.89	0.2695731	nan	0.0377545	0.2666882
3.81	nan	0.045789	nan	0.2363195
3.73	nan	0.0633071	nan	0.0839059
3.66	nan	0.182856	0.1732841	0.1967832
3.59	nan	nan	nan	0.0827375
3.53	0.154734	nan	0.1016588	0.2411876

$CC*$ values for the full light and dark PSII datasets from LCLS June 2012 merged with *partialator* scaling. For indexing conditions, see table 7. The last value greater than 0.5 in each column is bolded, motivating 4.33 Å as the cutoff for future refinements of this data.

values, the electron densities from indexing condition 17 look better than the electron densities from indexing condition 0.

6.2.1 L Test

However, indexing condition 17 gave a warning during phasing that intensity moments suggested the possibility of twinning. Physical twinning is not possible in the space group ($P2_12_12_1$). The L-test (Padilla and Yeates 2003) is an intensity statistic

Table 18. Phasing and refinement statistics by search model

PDB	State	Idx	LLG	TFZ	Init R_{work}	Final R_{work}	Init R_{free}	Final R_{free}	WRN
MR1	Dark	0	14706.7	**90.2**	0.2698	0.2641	0.2553	0.2526	BW
		17	9285.3	81.1	0.2651	0.26	**0.245**	**0.2382**	BW
	Light	0	14787.4	89	0.2694	0.2641	0.2522	0.2465	BW
		17	9409.4	80.3	0.2644	0.2594	0.2506	0.2417	BW
3wu2 (1)	Dark	0	16185.7	85.6	0.2691	0.2634	0.2768	0.2737	B
		17	9951.5	76.1	0.2647	0.26	0.2677	0.26331	B
	Light	0	**16308.4**	85.7	0.2681	0.2629	0.2691	0.2706	BF
		17	10079.7	76.7	0.2638	0.2592	0.2678	0.2623	B
3wu2 (2)	Dark	0	15758.1	82.6	0.2689	0.2633	0.2679	0.2663	B
		17	9708.7	73.8	0.2638	0.2592	0.2573	0.2463	BW
	Light	0	15891.8	82.7	0.2684	0.2631	0.2649	0.2612	BW
		17	9858.9	73.5	**0.2633**	**0.2585**	0.261	0.253	BW

Phasing and refinement output. The first column is the search model used for phasing, either MR.1.pdb, PDB 3wu2 with waters, RRX, UNL, and HTG removed (1), or 3wu2 with waters, RRX, HTG, UNL, GOL, and LMG removed (2). The second column is dark or light dataset. The third column is the indexing condition (see table 7). LLG (rounded to 1 digit) and TFZ are output from *phenix.phaser* with ligands included. R values come from rigid body refinement with monomers. WRN stands for warnings with B for high bonds / angles, W for R_{work} better than R_{free} and F for R_{free} increasing during refinement. Best values in each output column are bolded.

that is insensitive to anisotropic diffraction and pseudo-centering. L is defined as:

$$L = \frac{I(h_1) - I(h_2)}{I(h_1) + I(h_2)} \tag{6.2}$$

where I is the intensity of a measurement and h_1 and h_2 are neighboring reflections that are not related by twin laws. $|L|$ (x axis) is plotted against the cumulative probability distribution of $|L|$ (y axis) to detect anomalies. The expected distribution for an acentric crystal without twinning is a straight line, and an acentric crystal with twinning is a curved line.

In SFX data analysis, the L test has been used to compare PSII datasets (Wang et al. 2017) and identify errors in analysis (such as incorrect handling of negative intensities (Lyubimov, Uervirojnangkoorn, Zeldin, Zhou, et al. 2016)). *Phenix.xtriage*

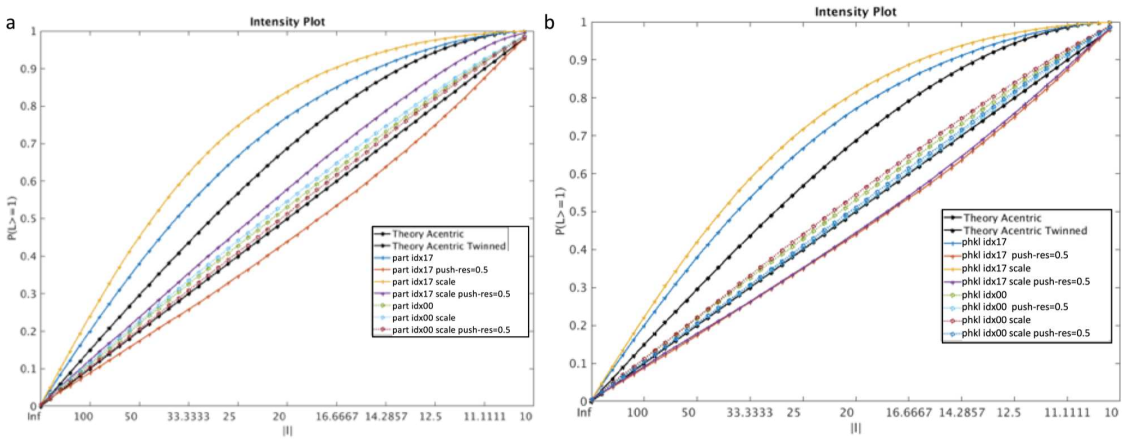

Figure 77. L-Test plots for full dataset

L-Test plots from *Xtriage* for the 16 merge screen conditions for light data (top half of table 15). The straight black line is expected for non-twinned data and the curved black line is for perfectly twinned data. (a) *partialator* results and (b) *process_hkl* results. *Partialator* results for condition 00 with push-res were not available because *Xtriage* had an error while processing. Indexing condition 17 is solid lines and indexing condition 0 is dotted lines.

was used to generate the L test plot for the 16 merges from the light dataset (see table 15).

Figure 77 shows the L-Test results with indexing condition 17 as solid lines and indexing condition 0 as dotted lines. Indexing condition 17 has odd L-Test results with both *partialator* (fig. 77a) and *process_hkl* (fig. 77b). The solid yellow (scaled) and blue (non-scaled) lines are both super twinned. Using push-resolution improved the results with the red (push-res=0) and purple (scaled, push-res=0) appearing less twinned than than the corresponding yellow and blue lines, but in three of the four cases those lines are undertwinned and in all four cases they are outside the range of indexing condition 0.

One of the indexing parameters that differed between indexing conditions 0 and 17 must cause the abnormal results since indexing condition 17 has abnormal results with

all merging conditions and indexing condition 0 does not. To detect the point where the L test results begin to deviate from normal, L-test plots were created for the 18 indexing conditions given in table 7. Figure 78 shows the results with part a showing the integration method screen, part b showing the integration radii screen, part c showing the profile radius, and part d showing the three points were a parameter was changed between indexing conditions 0 and 17.

From fig. 78, the parameter with the largest impact on the L test was the integration method. However, the figure also shows that the L test deviates much less when only 2,000 patterns are used. To verify that the integration method caused the abnormal intensity statistics, two additional indexing conditions were tested. Both used the integration radius of 245 and a fixed profile radius of 0.0053 to match indexing condition 17. However, the integration method was either rings or rings-cen (indexing condition 17 had an integration method of rings-cen-grad). The two new indexing conditions were merged with all four possible merging conditions.

An L test comparison of indexing conditions 0 and 17 with the two new indexing conditions is shown in fig. 79 with the first column considering all resolutions and the second column limiting the resolution to 20 Å to 4.3 Å. Figure 79 verifies that integration methods including the 'cen' flag have abnormal intensity distributions compared to those without that flag.

6.2.2 Integration Method Comparison

All 32 conditions were phased with the second 3wu2 variation (with waters, RRX, UNL, HTG, GOL, and LMG). Refinements were done with rigid body monomers.

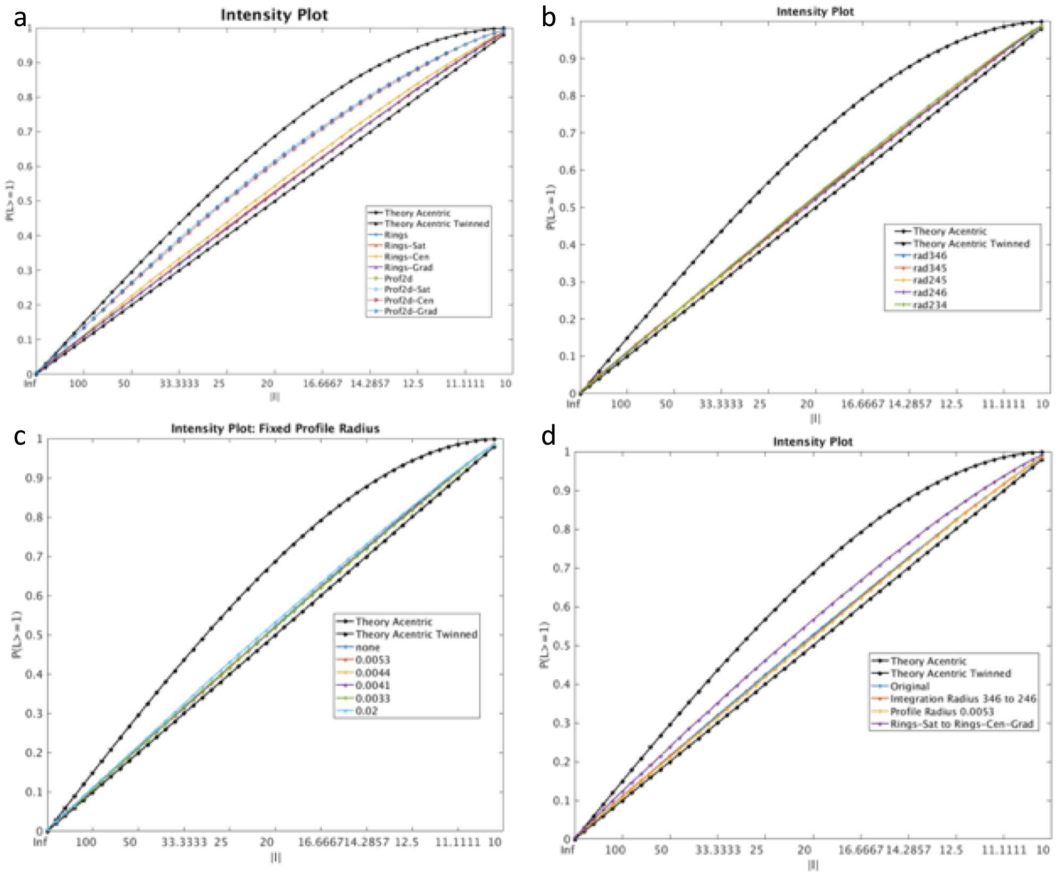

Figure 78. L-Test plots for indexing screen

L-Test plots from *XTriage* for the 18 indexing screen conditions for light data (see table 7). Note that since indexing screening only used 2,000 patterns, the statistics aren't as reliable. (a) Changes in L-Test results by integration method. Prof2d is dotted lines and rings is solid lines, showing prof2d appears more twinned than rings. (b) Changes in L-Test results by integration radii, showing integration radii has little affect on the test. (c) Changes in L-Test results by fixed profile radius, showing little impact. (d) A summary of the three points where a parameter was before screening the next set of parameters, showing the change in L-Test results occurred when changing the integration method.

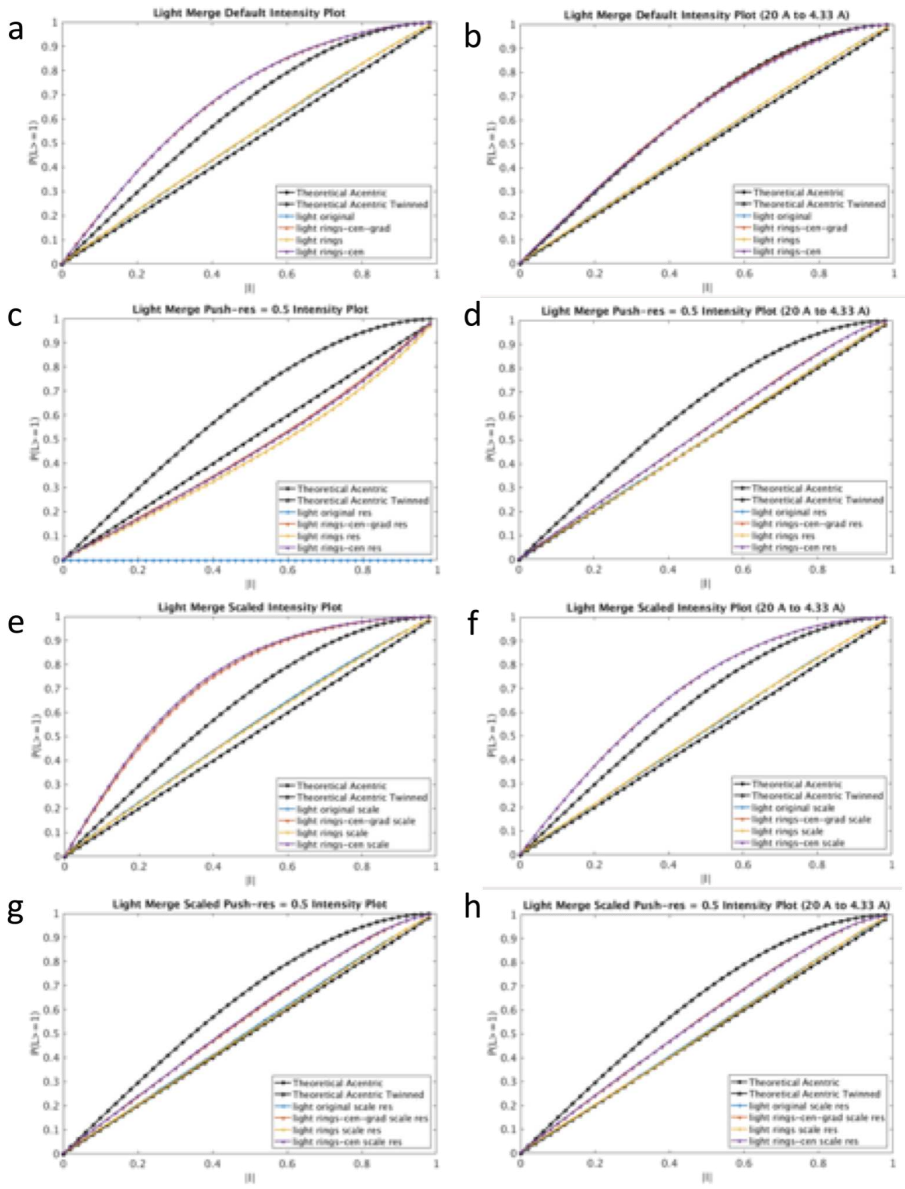

Figure 79. L-Test plots for full datasets by integration method

L-Test plots from *XTriage* for all light data with merged with *partialator*. The four rows are four different merging parameters with (a,b) from default parameters, (c,d) with push-res = 0.5, (e,f) with scaling, and (g,h) with both scaling and push-res = 0.5. The two columns are from *Xtriage* with the full dataset (a,c,e,g) and *Xtriage* limited in resolution to 20 Å to 4.33 Å (b,d,f,h). Each plot has four different integration methods plotted with indexing condition 0 blue (rings-sat), indexing condition 17 red (rings-cen-grad) and the final two conditions matching condition 17 except for integration method with yellow using rings and purple rings-cen.

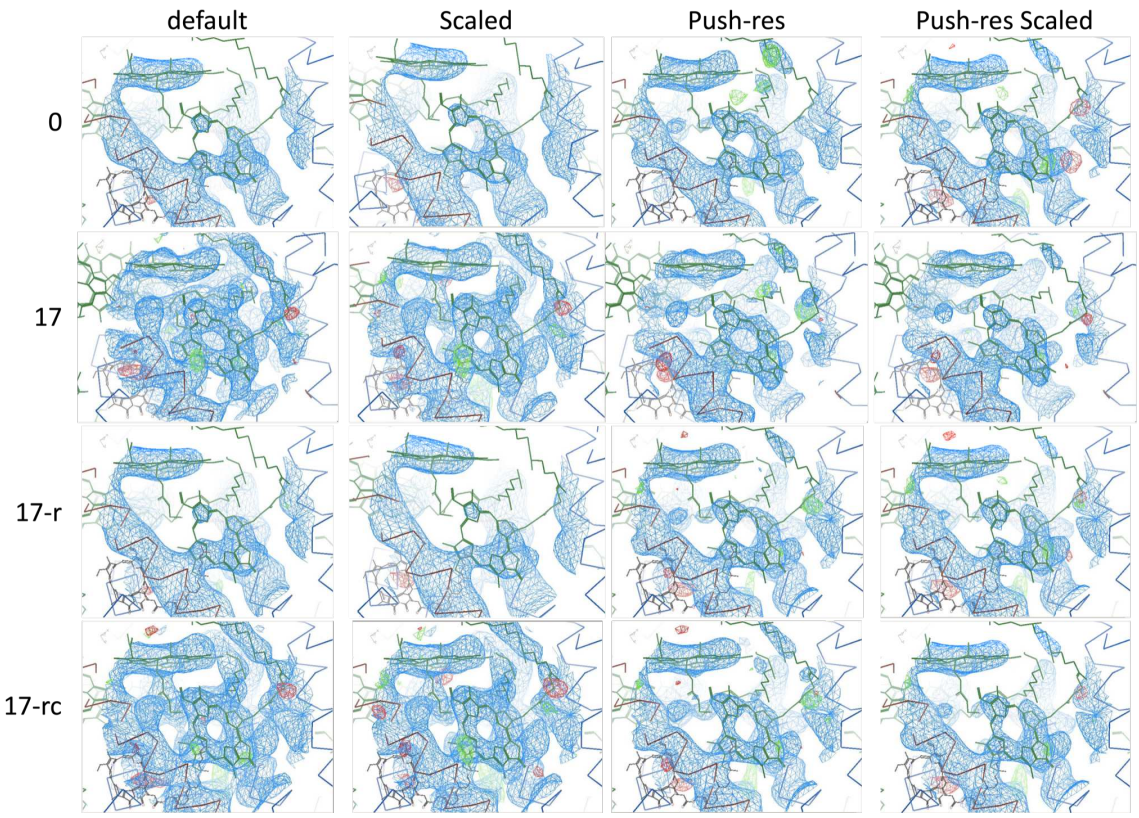

Figure 80. Electron densities by integration, and merge (dark)

A view of the electron density around the pheophytin molecule of chain A for dark data summarized in table 19. Each row is an integration method with indexing condition 0, 17, 17 with integration method rings (17-r) or 17 with integration method rings-cen (17-rc) (see table 7 for indexing conditions 0 and 17). Each column is a merging condition with *partialator*.

Results are given in table 19 with the first half of the table for the 16 dark datasets and the second half for the 16 light datasets.

Figures of the electron densities around the pheophytin molecule of chain A were generated for all 32 results. Figure 80 shows the 16 dark datasets and fig. 81 shows the 16 light datasets. Indexing conditions form the four rows, and merging conditions form the four columns in the figures.

Table 19. Integration method comparison

		Merge		Phase			Init	Final	Init	Final	
	Idx	R	S	LLG	TFZ	WRN	R_{work}	R_{work}	R_{free}	R_{free}	WRN
D	0			16279.2	85.7		2660	2599	2685	2644	B
		0.5		13833.8	82.6		2802	2750	2707	2707	BWF
			Y	15758.1	82.6		2689	2633	2679	2663	B
		0.5	Y	14503.1	82.6		2791	2733	2743	2729	BW
	17			10625.3	79	T	2585	2532	2577	2496	BW
		0.5		13596.9	80		2746	2690	2746	2665	BW
			Y	9708.7	73.8	T	2638	2592	2573	**2463**	BW
		0.5	Y	14320.7	79.8		2647	2589	2636	2529	BW
	17-r			16187.9	85		2634	2574	2641	2606	B
		0.5		13943.7	80.5		2824	2766	2757	2716	BW
			Y	16019.9	81.7		2665	2608	2633	2578	BW
		0.5	Y	14618.3	80.8		2737	2677	2740	2680	B
	17-rc			10664.9	79.5	T	2578	2528	**2565**	2519	BW
		0.5		13592.5	80.5		2758	2703	2790	2728	B
			Y	9707.4	73.9	T	2632	2584	2627	2574	BW
		0.5	Y	14341.7	80.7		2640	2582	2652	2583	B
L	0			**16523.4**	**86.1**		2648	2591	2640	2600	B
		0.5		14306.6	82.5		2841	2774	2845	2789	B
			Y	15891.8	82.7		2684	2631	2649	2612	BW
		0.5	Y	14891.9	81.5		2776	2715	2757	2692	BW
	17			10882.1	79	T	2579	2524	2567	2507	BW
		0.5		13878.5	81.2		2731	2673	2714	2669	BW
			Y	9858.9	73.5	T	2633	2585	2610	2530	BW
		0.5	Y	14626	80.9		2645	2584	2671	2610	B
	17-r			16437.7	85.2		2629	2567	2653	2632	B
		0.5		14065.2	81.7		2827	2766	2889	2847	B
			Y	16174.6	82.1		2654	2595	2679	2627	B
		0.5	Y	15023.7	81.8		2735	2672	2781	2746	B
	17-rc			10908	78.7	T	**2577**	**2523**	2600	2546	B
		0.5		13905.1	81.2		2788	2735	2765	2750	B
			Y	9925	73.1	T	2624	2577	2647	2573	BW
		0.5	Y	14728.9	80.7		2638	2578	2660	2620	B

Phasing and refinement statistics of the indexing and merging screen for LCLS June 2012 PSII data. The top half of the table is dark (D), the bottom half light (L). Idx is indexing condition with 0 and 17 defined in table 7 and 17-r and 17-rc variations of indexing condition 17 where the integration method is rings and rings-cen respectively. The merging columns are R for for push-res and S column for scaling. LLG values are rounded to a single digit. Refinement was rigid body monomer refinement. R values are e^{-4} (so 2577 is 0.2577). Warnings (WRN) for phaser are T for intensity statistics suggesting twinning and for refinement are B for high bonds / angles, W for R_{work} better than R_{free} and F for R_{free} increasing during refinement. Best values in statistical columns are bolded.

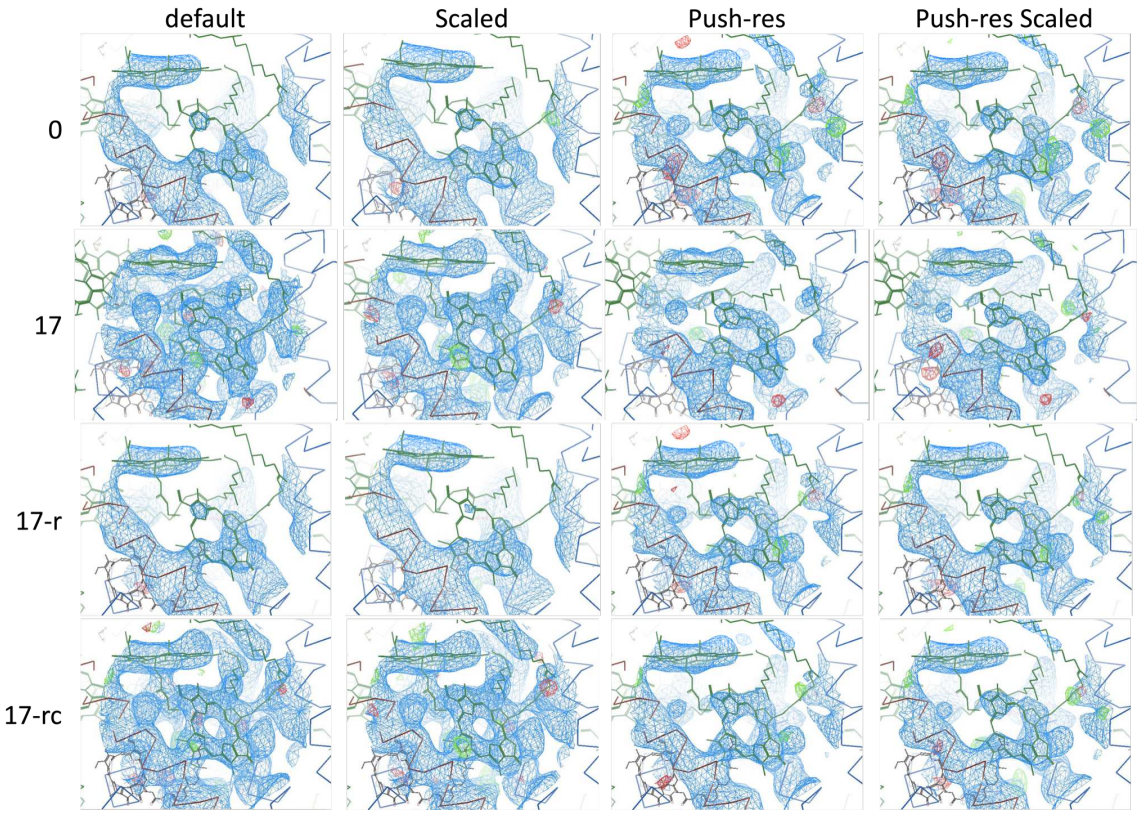

Figure 81. Electron densities by integration, and merge (light)

A view of the electron density around the pheophytin molecule of chain A for light data summarized in table 19. Each row is an integration method with indexing condition 0, 17, 17 with integration method rings (17-r) or 17 with integration method rings-cen (17-rc) (see table 7 for indexing conditions 0 and 17). Each column is a merging condition with *partialator*.

The first thing to note is that, following the L test trend, indexing conditions 17 and condition 17 with rings-cen had warning during phasing that the intensity statistics suggested twinning when push resolution was not used with merging. Although with push-res they still have unusual L Test results, it did not trigger *phenix.phaser*'s warning. However, the improvement with push-res comes at a price. From table 19 push-res datasets had higher R values than the corresponding non push-res datasets with the only exception the R_{work} of indexing condition 17 with *partialator* scaling.

Trends in the electron densities are harder to see. Certainly, the electron densities that came from indexing condition 17 look good, perhaps even the best. However for dark data, *partialator* push-res scaled on indexing condition 0 also has a complete ring (fig. 80). Yet for light data, the push-res scaled condition never has a complete ring and the only complete ring without a warning is from indexing condition 17 rings-cen with push-res. For dark data, push-res seems to improve the density, but for light data the trend isn't as clear.

This section is the final section for the LCLS June 2012 dataset. The major take-away message from this analysis is that indexing and merging parameters such as using rings-cen for the integration method or push-res at merging can impact all later stages of analysis. The results presented in this dissertation are limited in that they deal only with one PSII dataset and have not been tried on other PSII datasets or other proteins. However, for this dataset, an integration method of rings-cen causes unusual intensity distributions. In comparing statistics, R_{split} and SNR were better for indexing condition 17 but indexing condition 17 has an odd intensity distribution. Therefore, in this case, the CC was the better statistic for comparing indexing results.

6.3 Subset Comparison

This section returns to the analysis of LCLS November 2016 dataset with the seven unit cell subsets introduced in section 4.3.2 and summarized in table 12. Phasing was done for each of the 28 .mtz files (from the seven subsets and four merging conditions) with the second variation of 3wu2 (with waters, RRX, UNL, HTG, GOL, and LMG removed). Refinement was done with rigid body monomers with different cutoffs for

each subset based on CC values. Statistics are given in table 20 and electron densities are shown in fig. 82

Each subset has its own trends for the best merging condition. For the big cell low resolution data, default merging had the best statistics in all categories, but for the low resolution dataset which had a similar resolution, best values were scattered across many conditions with a slight preference for the scaled dataset. The big cell and small cell low resolution datasets also had similar resolutions but for the big-cell dataset no trend is visible and for the small cell low resolution dataset phasing did best with scaling and refinement did best with push-res. The high resolution and small cell high resolution have better R values in general with scaling, but the similar small cell dataset had better values with both scaling and push-res.

Comparing overall R values, the trend follows that of the CC with the high resolution refinements having the best R_{free} final values, followed by small cell high resolution, and then the remaining datasets which all have similar values. Interestingly, the small cell dataset does not have particularly different values than the other five datasets.

In summary, for LCLS November 2016, initial analysis suggests that including more high resolution patterns at the expense of a larger unit cell distribution is better than including a smaller unit cell distribution. However, as mentioned before, this initial analysis had a large range of unit cells.

Table 20. Subset comparison

	Merge		Phase				Init	Final	Init	Final
	R	S	LLG	TFZ	Res	R_{work}	R_{work}	R_{free}	R_{free}	WRN
BC			6797.5	**86.9**	4	3453	3187	**3395**	3123	BW
BC	0.5		**7329.7**	78.5	4	3466	3198	3425	3120	BW
BC		Y	6390.7	76.4	4	**3446**	3185	3438	**3050**	BW
BC	0.5	Y	7065.8	77.5	4	3447	**3181**	3412	3103	BW
BL			**3730.2**	**59.8**	5	**3343**	**3111**	**3429**	**3215**	B
BL	0.5		2841.3	51.8	5	3410	3295	3635	3467	B
BL		Y	3516.2	57.5	5	3375	3147	3450	3255	
BL	0.5	Y	2687.2	50.4	5	3397	3169	3529	3320	B
HR			18899.8	**135.5**	3.33	3214	2945	3143	2833	BW
HR	0.5		**19319.4**	133.8	3.33	3216	2947	3137	2835	BW
HR		Y	18894.3	134.6	3.33	**3185**	**2913**	3101	**2817**	BW
HR	0.5	Y	19114.1	132.1	3.33	3186	2915	**3100**	2817	BW
LR			3199.4	57.3	5	3428	3263	**3308**	3314	BF
LR	0.5		2141.3	45.9	5	3434	3297	3376	3294	BW
LR		Y	**3419.6**	**59.3**	5	3427	**3245**	3313	3288	B
LR	0.5	Y	2211.1	39.2	5	**3420**	3261	3323	**3258**	BW
SC			13910.3	124.5	3.33	3452	3207	3465	3290	B
SC	0.5		16226.7	130	3.33	3465	3193	3469	3267	B
SC		Y	13826	124.7	3.33	**3428**	3183	3450	3264	B
SC	0.5	Y	**16274.6**	**130.3**	3.33	3438	**3149**	**3438**	**3204**	B
SH			17987	**134.7**	3.33	3337	3023	3306	2987	BW
SH	0.5		18666	132.1	3.33	3357	3044	3321	3000	BW
SH		Y	18037.2	134.4	3.33	**3299**	**2979**	3284	**2947**	BW
SH	0.5	Y	**18675.4**	132.6	3.33	3313	3000	**3273**	2952	BW
SL			8433.9	87.1	4	3293	3167	3373	3175	B
SL	0.5		4373.8	50.4	4	3068	**2949**	**3113**	**3008**	B
SL		Y	**8541.1**	**87.5**	4	3238	3150	3378	3209	B
SL	0.5	Y	4490.2	50.4	4	**3065**	2957	3193	3072	B

Phasing and refinement statistics for the subsets of LCLS November 2016 PSII data listed in table 12. Merging was done with *partialator* with the R column for push-res and the S column for scaling. Phasing was done with *phenix.phaser*. LLG values are rounded to a single digit. Refinement was rigid body monomer refinement with *phenix.refine* with the cutoff determined by dataset as shown in the Res column (Å). R values are e^{-4} (so 2577 is 0.2577). Warnings (WRN) are B for high bonds / angles, W for R_{work} better than R_{free} and F for R_{free} increasing during refinement. Best values in each group of four are bolded.

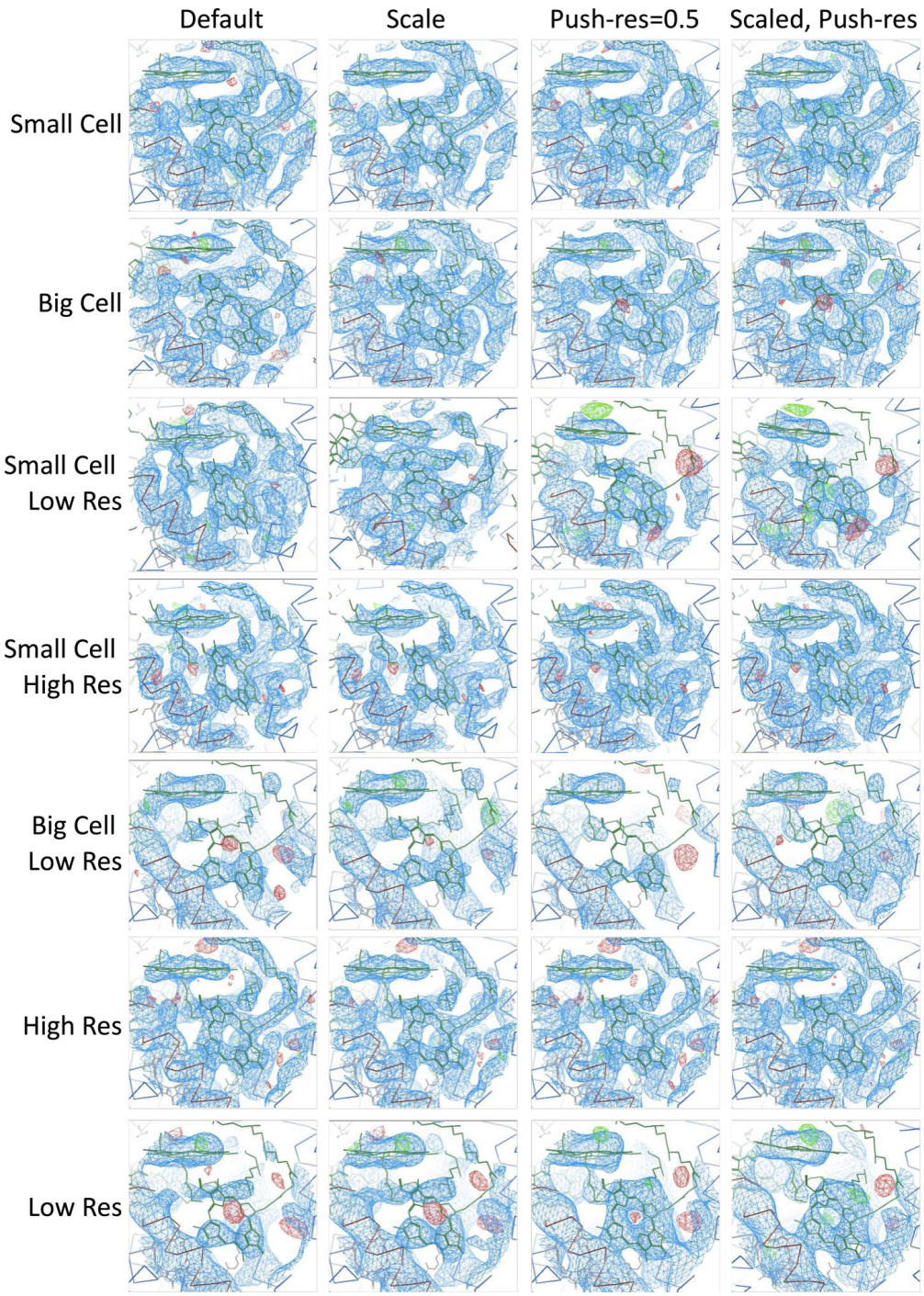

Figure 82. Electron densities by subset

A view of the electron density around the pheophytin molecule of chain A for subsets listed in table 12. Each row is a subset. Each column is a merging condition with *partialator*.

Chapter 7

CONTINUOUS DIFFUSE SCATTERING ANALYSIS

In (Ayyer et al. 2016), a 4.5 Å PSII dataset (LCLS November 2014) was extended to 3.5 Å using continuous diffuse scattering. The theory behind the analysis is that small random translational shifts of the PSII dimer occur in the crystal leading to the Fourier Transform of the four PSII dimers in the unit cell generating a continuous diffraction pattern. Combining all the pixels in the image into a 3D volume (using the orientations determined during indexing of the Bragg peaks) provides enough voxels of information to directly phase the data and compute the inverse Fourier Transform to reconstruct the PSII dimer. However, the Bragg data at low resolutions are not part of the continuous diffuse scattering and must either be masked or removed.

This chapter covers the work done with continuous diffuse scattering analysis on other PSII datasets. Section 7.1 introduces the software and some overall concepts. Section 7.2 then describes initial work using the software for LCLS October 2015 PSII dataset. Section 7.3 describes parameter screens used after the basic functionality of the software seemed to be working correctly. However, the results were not as good as those published, leading to a more in depth study of the differences between the current analysis and the analysis presented in the paper. Other than the different starting dataset, there were two primary hypothesis of how the current analysis could be improved. First, the paper used only a subset of the available data in the analysis. However, the script used to generate that subset was not available. Section 7.4 describes several different criteria used for creating subsets and the affect on continuous diffuse scattering analysis. The second hypothesis was background

correction. It's known that background correction and masks need to be very accurate for the analysis to work, and it had not been optimized for these datasets which extend to higher resolution than the Ayyer et al. 2016 data. Since background correction was an active area of research at CFEL, work on background correction with these datasets is primarily testing the effect of different updates. Section 7.5 shows background correction results with LCLS August 2016 PSII data that help identify the presence of nozzle shadows.

7.1 Overview

Crystallography can be thought of as finding a solution to a set of mathematical constraints. The solution is the placement of atoms in the unit cell. Constraints can be in real space or reciprocal space. Reciprocal space is the Fourier Transform of real space, and the constraints are the experimental intensities. Real space constraints can involve simple things like electron density can't be negative or the sequence information of the protein. It can get more complex by favoring statistics common in the PDB or by using a support where electron density must be inside the support and not outside of it.

Iterative phasing alternates between real space and reciprocal space, enforcing the constraints are met. Figure 83 gives the general idea. Random phases are assigned to the measured intensities, and then an inverse Fourier Transform is calculated (green line). The result probably does not meet the real space constraints. So, the next step is to enforce the real space constraints. For example, a non-negativity constraint may be met by setting all negative values to 0 (light blue line). Next, the Fourier Transform is applied to bring the data back to reciprocal space (purple line). Now

the intensities no longer match the measured values. So, the intensities are updated to match the reciprocal space constraints (purple line). The process is repeated for many iterations until a solution that matches both sets of constraints is found.

Iterative phasing algorithms are essentially search algorithms. Searching all possible solutions becomes computationally intractable with larger numbers of atoms. Search algorithms attempt to sample the possible solutions and find the true minimum or maximum. However, since not all solutions can be tested, the solution found may be a local minimum/maximum instead of the global one. The solution found will be impacted by the initial random phases. Figure 84a shows a visual representation of a local minimum, where a solution is not found because the algorithm is stuck alternating between the same two possibilities. Iterative phasing algorithms are typically run several times to reduce the bias of the initial random phases.

Another problem occurs when there are many possible solutions, as visually shown in fig. 84b. This problem occurs when there are not enough constraints. For a small number of atoms, there are enough constraints to directly solve a structure through iterative phasing. However, with large numbers of atoms like in protein crystallography, there are typically not enough constraints for the number of atoms. Continuous diffuse scattering that arises only from small, random translational shifts provides additional constraints in reciprocal space because all pixels can serve as constraints instead of just the intensities at Bragg peaks.

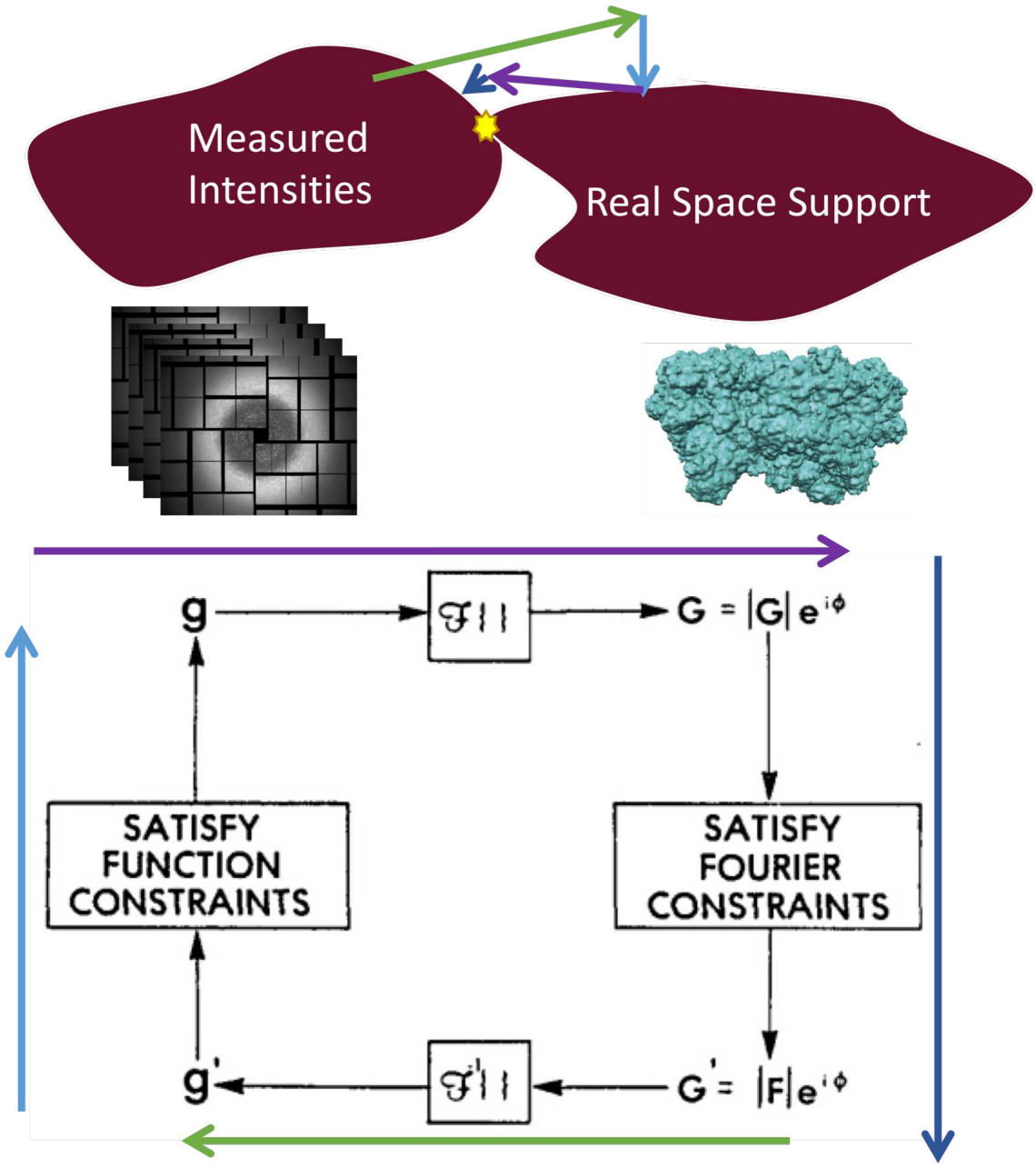

Figure 83. Iterative phasing overview

An overview of the idea behind iterative phasing. Iterative phasing combines information in Fourier space from crystallography with information in real space such as requirements that electron density cannot be negative. By iteratively enforcing the constraints of each space, a solution matching both sets of constraints may be found. The bottom half of the figure is adapted from (Fienup 1982) and the support figure comes from (Ayyer et al. 2016).

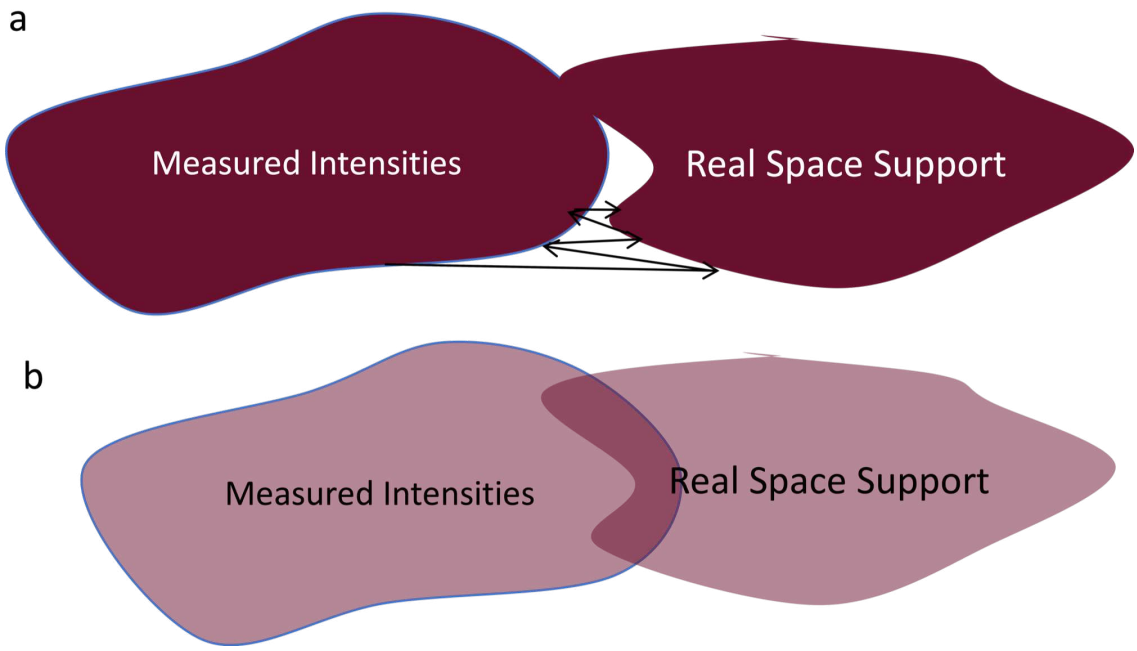

Figure 84. Iterative phasing problems

Iterative phasing can fail when at local minimums (a) where it repeats the same steps without converging. It also does not work when constraints are too loose, enabling many possible solutions (b).

7.1.1 3D Merge Software

The beginning steps of continuous diffuse scattering analysis are identical to SFX analysis: diffraction images are identified with hit finding and indexed. The first unique step in continuous diffuse scattering analysis is to merge all pixels (as opposed to only pixels with Bragg peaks) on the frame into a 3D volume to provide the continuous diffuse scattering intensity constraints. Originally, the software to create a 3D merged volume was an alternate way to merge Bragg data by first creating the volume and then integrating the spots instead of integrating the spots before merging (Yefanov et al. 2014). The software is written by Dr. Oleksandr Yefanov and has been called *intor* in the past but is now called *merge3d*. The program contains a large

number of algorithms and the one to run is determined by an external configuration file. An example configuration file is given below.

```
Ver.2
Mode = Orient_Raw
File = ../allB.stream
Geometry = ../../../cxij4915-v12-wclen.geom
%now optional parameters:
DataField = /data/data
MaskFile = ../../../maskOct15_new.h5
SomeOtherFile = param.txt
NumPoResult = 701
SubRadial = 7
SubPedestal = 0
Filter = 1 4 4
SaturationVal = 12000
NumThreads = 30
ReadPatterns = 0
SkipPatterns = 0000
%some strange parameters
%CalcVariance = 1
```

The mode parameter determines which of the program modes is run, in this case a 3D merge. The 3D merge program uses a stream file as input to determine the orientation of each pattern, which is the second line in the configuration file. The third line is a geometry file (see section 3.3.1). The % character comments out lines. DataField is the HDF5 path to the image data. MaskFile provides an additional mask (see section 1.5.1). SomeOtherFile is essentially a second configuration file that contains information about the space group and pixel size. An example param.txt is:

```
Average cell parameters: a, b, c, alfa, beta, gamma:
13.5747 22.9159 30.7809 90.0000 90.0000 90.0000
Pixel size in 3D in nm^-1:
80
```

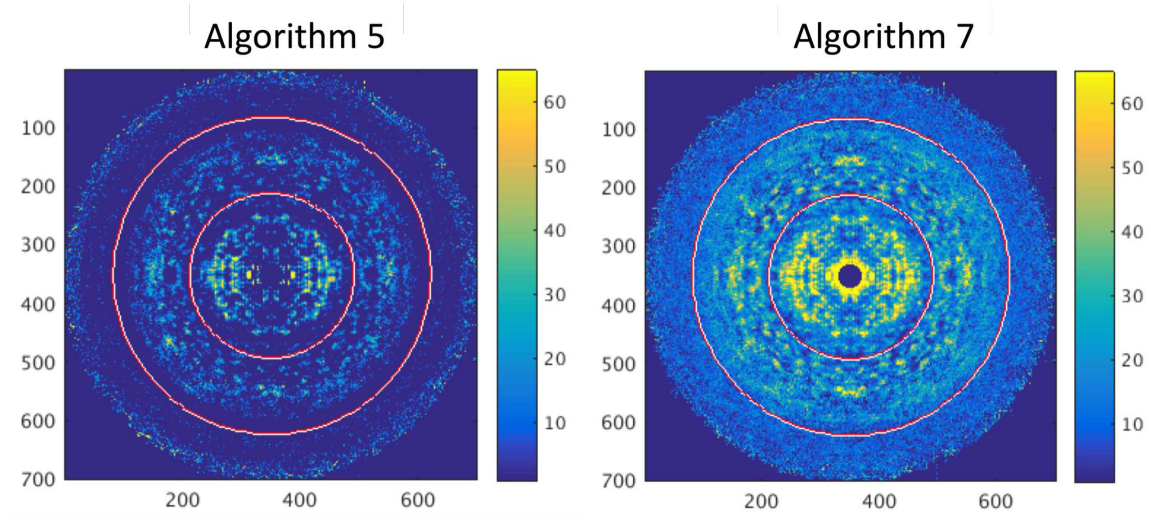

Figure 85. 3D merge background subtraction algorithms

An example showing a central plane of merges of runs 94-106 from LCLS October 2015. There are two main background subtraction algorithms. Algorithm 5 is a median radial subtraction algorithm (first column). Algorithm 7 is an initial version of the algorithm later published in (Chapman et al. 2017). The version used here is that of the merge program as it was in the git repository on March 24, 2016. The rings are at 140 pixels and 270 pixels corresponding to 5.7 Å and 2.9 Å respectively (the 80 in param.txt is 800 in $Å^{-1}$ and 800/pixels gives the value in Å. So, 800/140=5.7 Å).

NumPoResult determines the number of voxels in the result, with 701 meaning the output will be a 701x701x701 cube of voxels. The next four parameters deal with background correction of the frames. SubRadial controls radial background subtraction with different numbers referring to different algorithms. The important ones are 5 for median radial background subtraction as used in the paper, 7 for an initial version of Dr. Henry Chapman's developmental background subtraction, and 8 for the published version of Dr. Henry Chapman's background correction (Chapman et al. 2017). The difference between background subtraction algorithms 5 and 7 is shown in fig. 85.

SubPedestal is an on/off option likely related to dark calibrations (see section 1.5.2). Filter is a filter for Bragg peaks. The numbers define a tolerance parameter and the boundaries of the square to consider. SaturationVal is the saturation value of the pixels (see section 1.5.3). NumThreads is the number of computational threads. ReadPatterns and SkipPatterns determine the number of patterns to include and where in the file to start, similar to *CrystFEL* merging program options of 'start-after' and 'stop-after.' The final parameter needs to be on in order to output additional 3D volumes with other statistics that are needed to combine merges. Since merges take a single geometry file, merges must be run separately for different geometries and combined later.

The main output of the program is a 3D volume that is used in the iterative phasing software. For convenience, the central planes of the volume are output as 2D arrays for visualization. The number of patterns contributing to each pixel is also output for the slices, as shown in fig. 15a. However, the full volume with number of patterns at each pixel is not output unless the CalcVariance parameter is used. That volume is needed to combine merges since combining merges is essentially calculating a weighted average for each voxel.

7.1.2 Iterative Phasing Software

Figure 86 gives an overview of the iterative phasing software for continuous diffuse scattering analysis. Initial input to the analysis is an arbitrary system size, a 3D merge and an electron density map. The system size determines the volume of the three dimensional cube containing intensities. The value of 701 in the figure means that the 3D merge is 701x701x701. Odd numbers are typically chosen so that there is

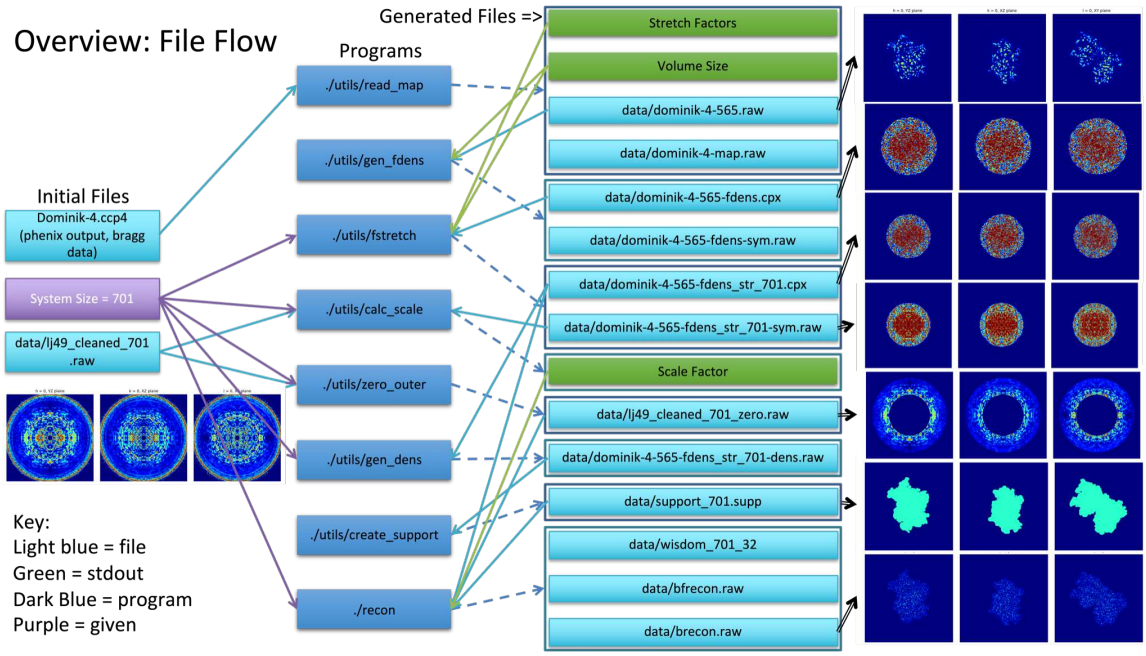

Figure 86. Resolution extension software overview

An overview of the software pipeline for resolution extension. Resolution extension begins with an existing electron density map (ccp4 format), a known system size defining the number of voxels in the three dimensional space, and the output from a 3D merge (.raw), listed on the left. The programs are listed in order in the second column with output in the third column. The fourth column shows views of the central (YZ, XZ, and XY) planes from some of the 3D volume files. The scripts and outputs are described in more detail in the text.

a central voxel. The 3D merge, in this case lj49_cleaned_701.raw, is a 3D volume containing the reciprocal space intensity constraints. The three central planes of the volume are shown below the label. LJ49 is the LCLS identifier for LCLS October 2015 beam time, and for the initial analysis this 3D merge was provided by Dr. Oleksandr Yefanov. The other file, Dominik-4.ccp4, is an electron density map provided by Dr. Kartik Ayyer for the initial analysis.

The first step in the analysis is a conversion step that changes the electron density .ccp4 format into a 3D volume of real space electron density. The script is called

read_map and produces .raw files, a naming convention meaning a 3D volume of numbers. On the far right of the figure, the 3 central planes of the map are shown, with the far right most easily identifiable as the PSII dimer.

As mentioned in the introduction, the Bragg intensities pose a problem for iterative phasing of the continuous diffuse values and must be masked or removed. To remove the Bragg intensities, the software cuts out the central sphere from the 3D merge. But the sphere has to be replaced with something, and so it is filled in with the Fourier transform of the map determined from the Bragg data. Thus the outer shell is extending the resolution of the existing inner-sphere Bragg data. In order to fit the Bragg data into the existing merge, a few steps are necessary. The script *gen_fdens* takes the Fourier transform of the volume, resulting in the second line of planes on the far right of the figure. Next, the 3D volume from the Bragg intensities needs to have the same voxel size as the 3D merge. The script *fstretch* scales the volume from its initial 565x565x565 cube (output from *gen_fdens*) into a 701x701x701 cube and also applies the space group. Now the two 3D volumes are on the same voxel scale, but not necessarily the same intensity scale. The script *calc_scale* takes both volumes and an inner and outer resolution limit to determine a scale factor relating the two volumes.

The iterative phasing software also preprocesses the merge data. The script *zero_outer* determines the shell of data that will be iteratively phased by setting any data outside that shell to 0. The result is shown as the fifth set of planes in the right of the figure.

So far, all of the processing has been building up the reciprocal space constraints (intensities) for phasing. For a real space constraint, a support is used to set the boundary of the electron density. Since PSII has been solved, the support is generated

from the existing structure with gaussian blurring. The script *gen_ dens* brings the reciprocal space volume back into real space (but now on the correct voxel scale) and the script *create_ support* blurs the result to create the second to last set of planes on the far right. Finally, all the pieces are available for iterative phasing, which is done with the program *recon* which outputs a real space and reciprocal space volume. The volume is converted back into a .ccp4 map format, and then to a .mtz format where it can be refined with traditional algorithms. The precedent set by the paper is *phenix.refine* with the 'MLHL' target function instead of the 'ML' target function, identifying the phases as experimentally determined.

7.2 Initial Analysis

This section covers initial continuous diffuse scattering analysis for LCLS October 2015. While a basic description of the software has been given in section 7.1, it's important to realize that this type of analysis is still developmental. In describing the sorts of problems encountered when using the software, this section aids in understanding what the software is doing and the motivations for later parameter screens. Also, the main source of documentation for this type of analysis is the published paper (Ayyer et al. 2016) so the preceding section and this one are valuable sources of information for anyone learning the software pipeline.

7.2.1 Merging

The first step in the continuous diffuse scattering analysis pipeline is creation of a 3D merge from stream files. For LCLS October 2015, Gihan Ketawala did the

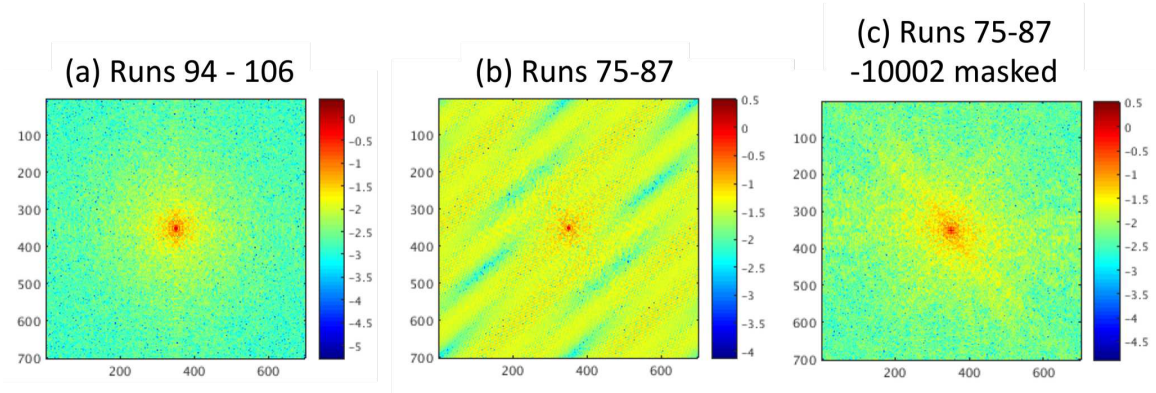

Figure 87. Autocorrelation streaks

An example showing the autocorrelation of the XY slice of merges from LCLS October 2015. (a) A merge from runs 94-106. (b) A merge from runs 75-87. (c) The same as (b) except -10002 values were set to 0. -10002 is a flag indicating no data for that particular voxel. Runs 75-87 had about half the number of frames compared to runs 94-106.

indexing and his files are used for all LCLS October 2015 continuous diffuse scattering analysis. Initial analysis focused on two groups for runs: 74-82 and 92-106.

Merges were generated for both sets of runs using the light data. LCLS October 2015 was a time resolved experiment with alternating light-dark data collection, and the light data for these runs is for the transient S4 state. The next question for a user of the software is how to tell if the merges worked. After some discussion, the autocorrelation was used to examine the slices of the central planes, based on its use in (Ayyer et al. 2016) figure 3. Figure 87a and b show the results for runs 94-106 and and 75-87 respectively. While fig. 87 looks as expected, fig .87b raised questions. Different annuli were tried, and merges were run on subsets of the runs to try to find the source of the streaks. In the end, the source of the streaks was limited to a pixel with a value of -10002. The large negative number is a flag value indicating that no information for that pixel was available (in other words, the number of patterns contributing to

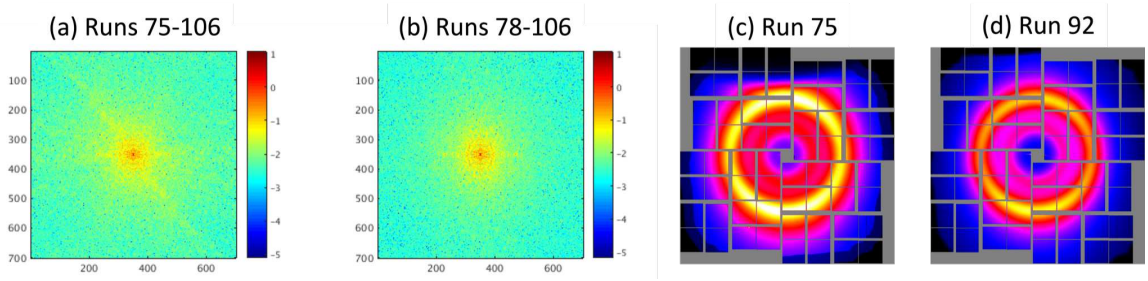

Figure 88. Autocorrelation star

An example showing the autocorrelation of the XY slice of merges from LCLS October 2015. (a) a merge from runs 75-76. (b) a similar merge but without runs 75 or 76. (c) The powder sum of run 75 showing an odd shadow. (d) the powder sum from run 92 for comparison. Detector shadows like this were introduced in section 1.4.1, in particular fig. 9.

the value at that pixel is 0). Setting all '-10002' values to 0 (or masking them) gives fig. 87c where the massive streaks are gone. A value of -10002 did not appear in the central slices of 94-106 likely because that dataset is almost twice as large.

However, fig. 87c still looks odd compared to fig. 87a. There's an almost 'star' background to it. Again, several variations of merges were tried, and the source of the star was linked to runs 75 and 76. Figure 88a,b shows the autocorrelation of merges with and without runs 75 and 76 respectively. Without those runs, the star vanishes. What makes run 75 weird? Figure 88c,d shows powder sums of runs 75 and 92 respectively. Run 75 shows an odd nozzle shadow at the top half of the detector.

After solving the streak and star autocorrelation issues, a merge was run on a combination of datasets with quinone added to the crystals (excluding problematic runs) , totaling 23,549 patterns. This merge was created only for parameter optimization, as combining different laser schemes is not appropriate when light and dark data are compared. Merges were also run on the half datasets generated with the

CrystFEL script *alternate-stream*, since comparing half merges was a known method of generating statistics (see section 3.2).

7.2.2 Phase Retrieval

For the iterative phasing software step, the electron density from the sample files was used (the Dominik-4.ccp4 file described in section 7.1.2). So, the only steps needed before the reconstruction step were cleaning of the 3D merge and determining the scale factor between the two volumes. Several possible methods for determining the outer resolution cutoff of the 3D merge were considered.

First, the presence of -10002 values indicates were the edge of available values are. Figure 89a shows a count of those pixels by radius. Note that all of the plots in that figure are generated from the XY slices because the whole volume is too large to easily work with. Another method, based on the similar method for determining the resolution of Bragg data, was the CC between the half merges. However, the question arises whether the bins should be cumulative. In other words, do the bins go 1-2, 2-3, 3-4 (not cumulative) or 1-2, 1-3, 1-4 (cumulative). Both methods were tried and are shown in fig. 89b,c.

With three different possible cutoffs and no direct way to know which was the correct one, all three were processed. They are referred to by the cutoffs with the -10002 values leading to a 340 pixel (2.3 Å) cutoff, cumulative correlation leading to a 320 pixel (2.5 Å) cutoff, and non-cumulative correlation leading to a 240 pixel (3.3 Å) cutoff.

A similar question of resolutions arises when considering the shell to use to calculate the scale factor between the merge and the Bragg electron density used to mask the

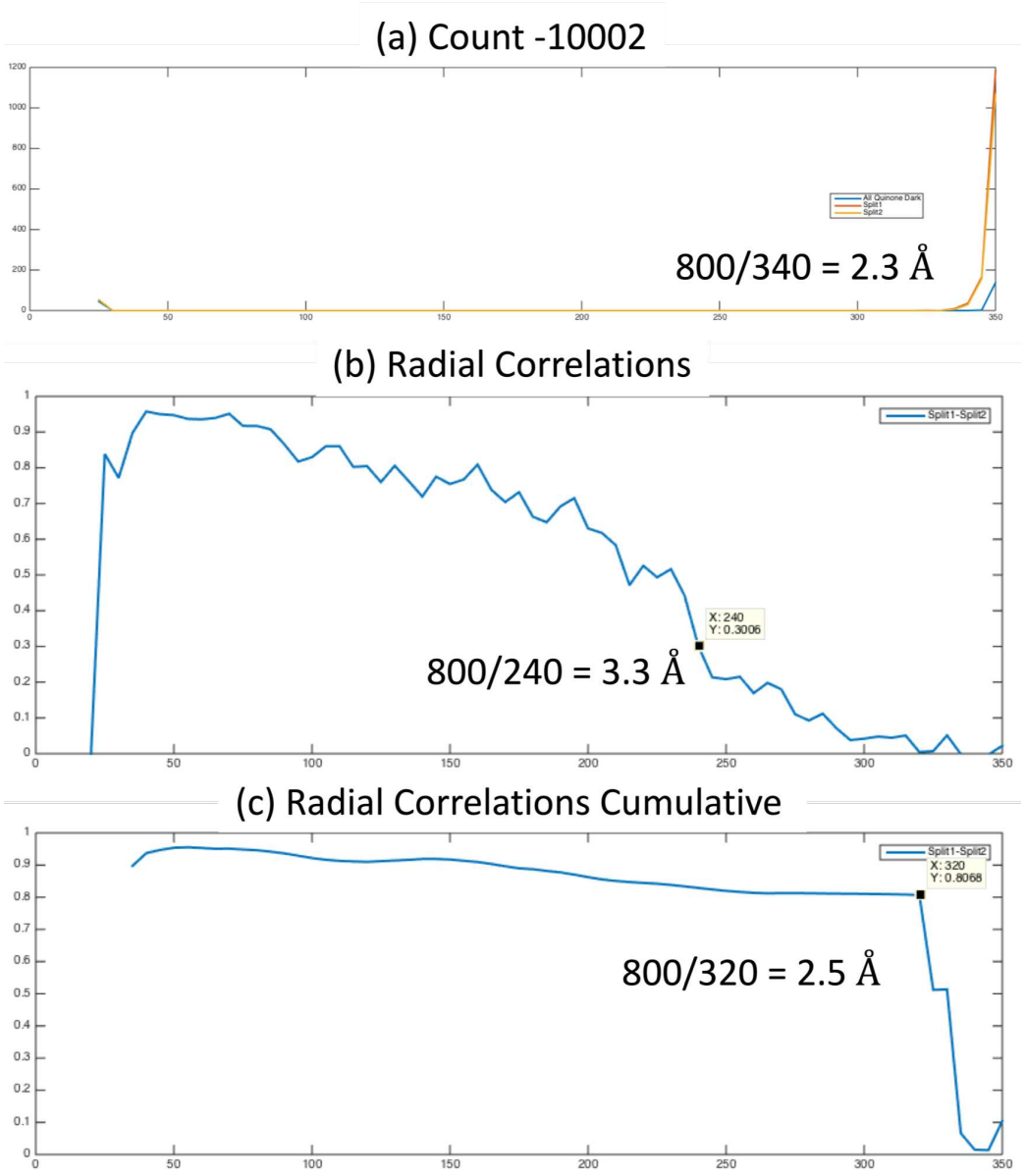

Figure 89. 3D nerge resolution

For the combined dataset of crystals with quinone from LCLS October 2015, 23,549 patterns were merged and half merges were created with 11,771 and 11,778 patterns using the *CrystFEL* script *alternate-stream*. Merges were run on the full dataset and each split and plots were generated from the XY slices. (a) The number of -10002 values (empty pixels) by radius with blue for all quinone and red and yellow for splits 1 and 2 respectively. (b) The correlation in shells between splits. (c) The cumulative correlation in shells between splits. Points on the plot are converted to Å and are the rational for the three cutoffs used later.

Bragg peaks. The outer cutoff is the resolution limit of the Bragg electron density, about 175 pixels. But the inner cutoff is less certain. A value past the edge of Bragg peaks was recommended. Several different cutoffs and the resulting scale factors are summarized in table 21, with the value 497 from the shell from 160 to 175 pixels used for further analysis.

Table 21. Scale factors by resolution cutoffs

Inside Cutoff	Outside Cutoff	Scale Factor
160	180	430.287272
160	175	**497.400681**
100	175	589.984249
35	175	646.755636
30	175	648.108452

Determining the scale factor to combine the experimental intensities with the existing map cleaned of Bragg data involves picking two cutoffs. The inside cutoff cuts out any Bragg noise from the experimental merge. The outside cutoff is determined by the edge of the previous map. The table shows scale factors based on different cutoffs with the value used bolded.

The reconstruction program was run on all three cutoffs of the merge data, and the central planes of the electron density are shown in fig. 91. Differences between the slices are hard to quantify.

The program *recon* outputs two statistical files in addition to the final volumes. One is called prtf.dat and presumably contains the phase retrieval transfer function over the percent of the radius (based on the name of the file, the data range from 0 to 1, and the start of non-zero information at roughly 0.5 which may be the inner radius of 160 divided by the total radius of 350). The PHASING.log file contains two columns with iteration and error. However, it was not recommended by Dr. Kartik Ayyer as a source of useful information. Figure 91 compares the outputs of both files for the three different merge cutoffs.

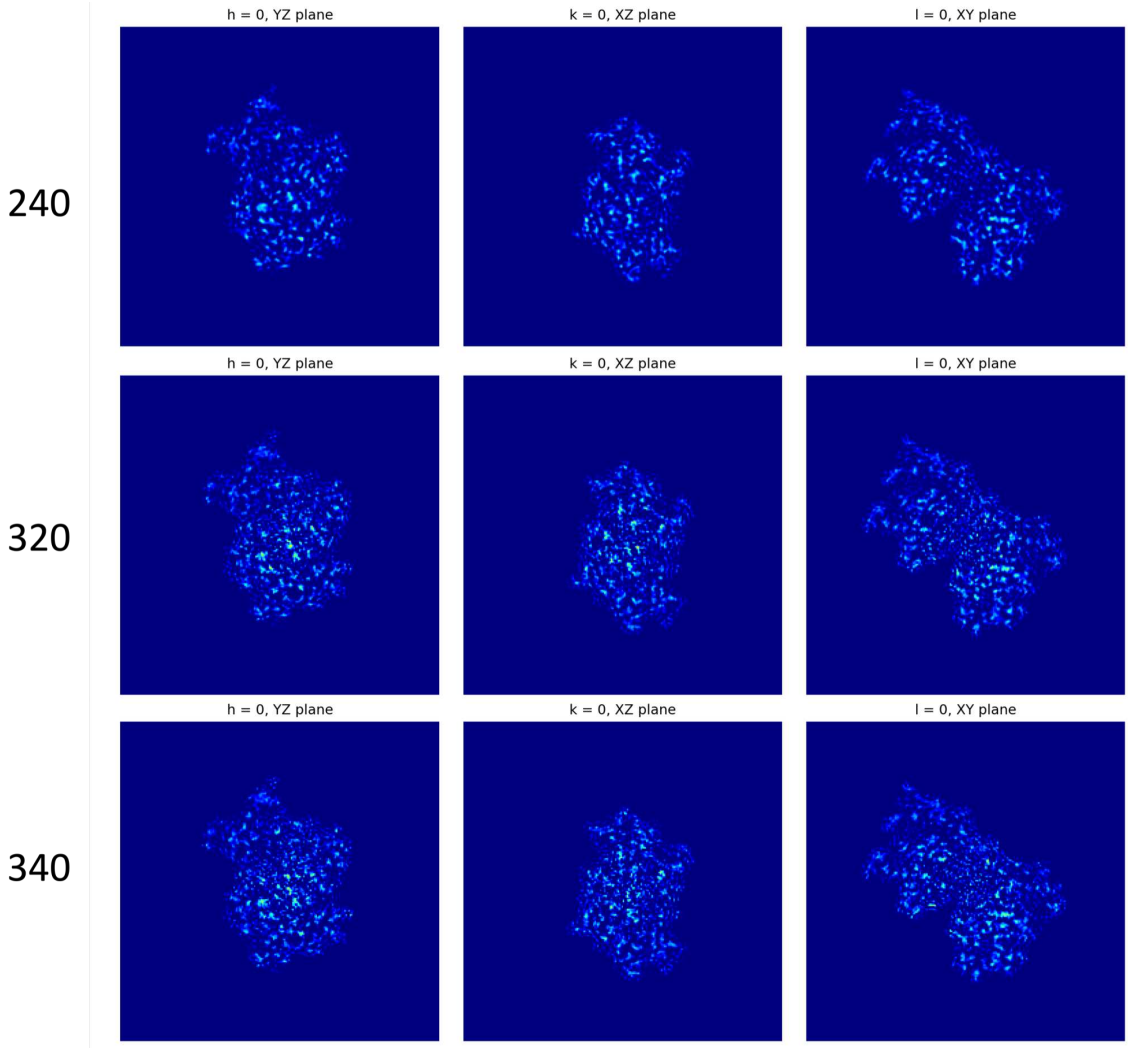

Figure 90. Resolution extension slices

The result of the iterative phasing step is a 3D volume of electron density. This image shows slices of the reconstructed electron density for the three different cutoffs from fig. 89.

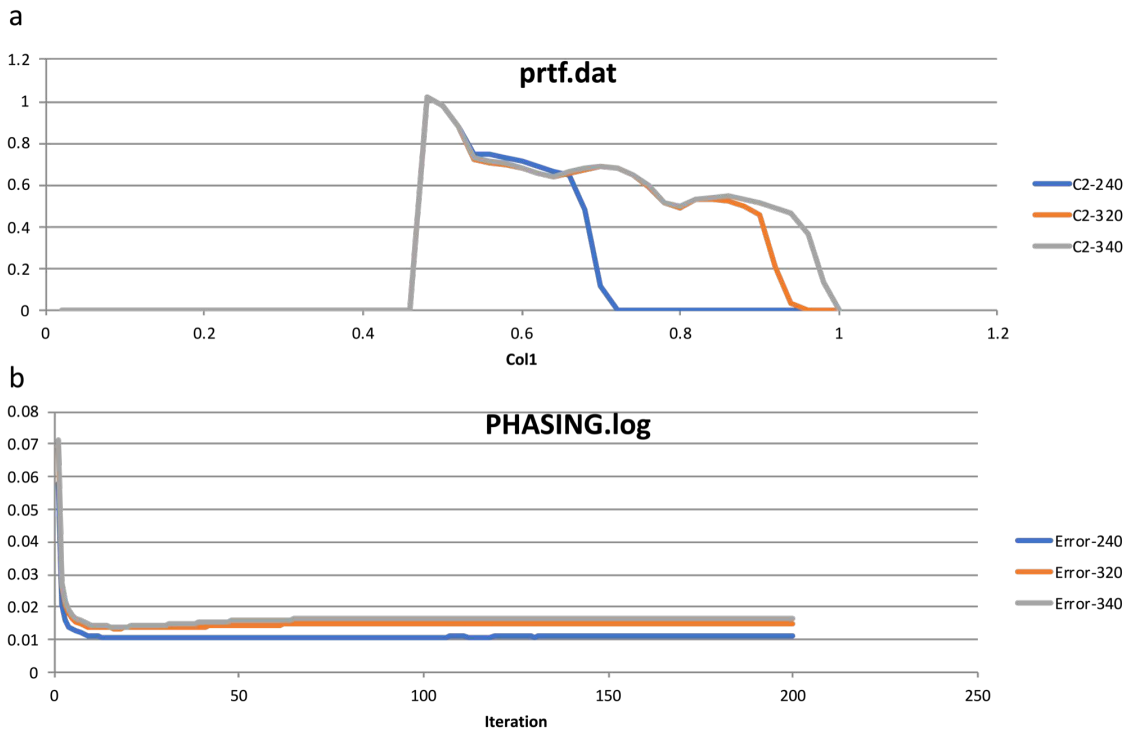

Figure 91. Resolution Extension Statistics

The two outputs from resolution extension. (a) prtf which presumably stands for phase retrieval transfer function for the three different cutoffs (in voxels) determined from fig. 89. (b) The phasing.log output for the same data.

The results of prtf.dat make sense. The smaller cutoff of 240 has no information past 240/350 on the x axis, the next largest cutoff (320) goes further, and the final cutoff (340) goes furthest. Interestingly, assuming that higher values are better, the 240 cutoff has better results around 0.6 than the two larger cutoffs. The PHASING.log results could also be read as showing the 240 cutoff had better results. However, comparing results between different cutoffs is probably not valid since smaller cutoffs have fewer pixels contributing to the final values than larger cutoffs.

7.2.3 Refinement

The electron densities for each cutoff were converted to .mtz format for refinement with *phenix.refine*. A lot of variations were tested and are summarized in fig. 92. The layout of the figure is designed to highlight particular problems identified through the set of refinements and is not in the order the refinements were initially run. The first four R values are comparing an older version of the software with an updated one. Regardless of the PDB file, the older software had better R values with 0.4457 better than 0.5869 and 0.5464 better than 0.5596. All four refinements were done with the 2.3 Å merge cutoff. This indicated that something was wrong with the new version of the software.

The next set of three is the only comparison between the different cutoffs. The 2.3 Å merge had an R value of 0.4607. The 2.4 Å merge surprisingly had a worse R value of 0.4919 and the smallest resolution cutoff of 3.3 Å had the best R value of 0.3911. It was noticed that the edge of the .mtz file actually went to the much higher resolutions than the data (in the 1-2 Å range) regardless of the merge cutoff. So, a refinement cutoff of 3.3 Å was applied to the next set of two to compare the different cutoffs over the range of data. The 2.3 Å merge cutoff had a better R value of 0.3028 than the 3.3 Å merge cutoff's value of 0.3469. This suggested including more pixels in the merge was better.

Most of the R factors given are higher than expected. In looking at the electron density results in *Coot* (Emsley and Cowtan 2004), one reason for the problem was identified. The PDB file used, 5E7C, was that deposited with the paper. However, from fig. 93, it obviously was not in the same orientation as the current electron density maps being produced. Dr. Kartik Ayyer recommended *Chimera* (Pettersen et al.

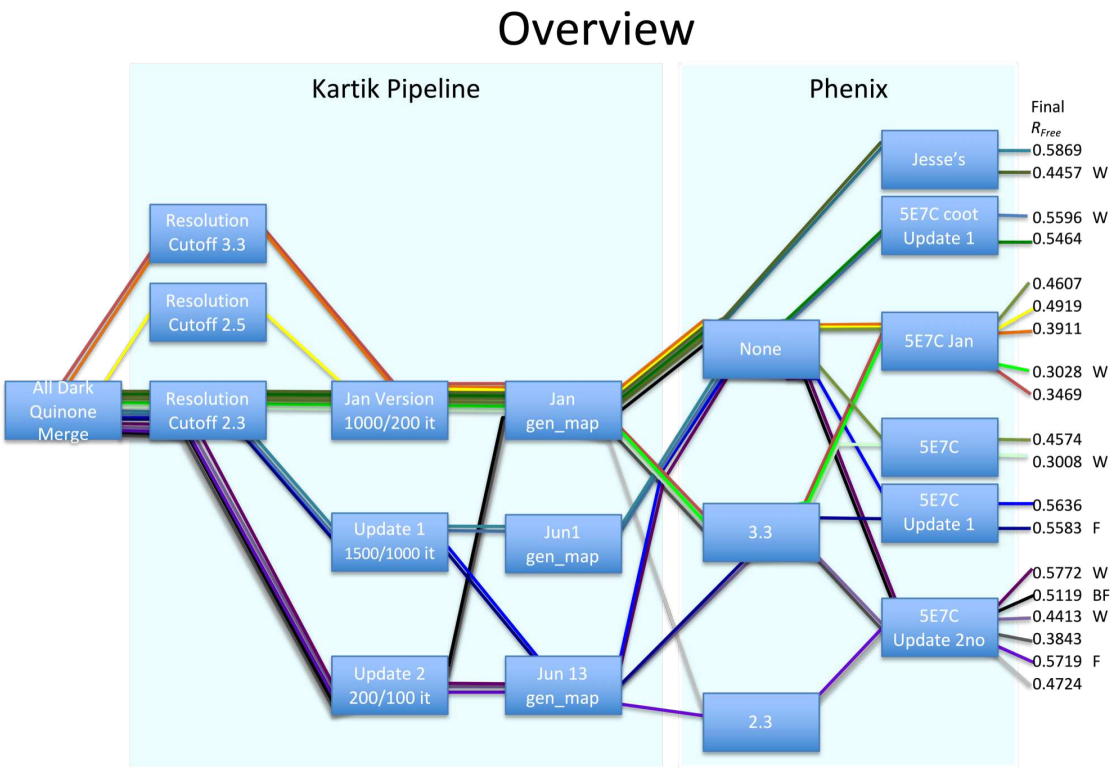

Figure 92. Debugging refinements summary

A summary of refinements identifying some of the problems shown in fig. 93. The columns are 1) the merge 2) the cutoff of the merge before phasing 3) the version of phasing software with the number of iterations (1000,200 means 1000 iterations the last 200 of which were averaged) 4) the version of software to convert the 3D volume of electron density to a map format for *phenix.refine* 5) The refinement resolution cutoff 6) the PDB used with refinement with the descriptor after 5E7C giving the version of software of the map that 5E7C was aligned to before refinement. Aligning was done with *chimera* with the exception of the 5E7C coot (second in column) where alignment was done with *coot*. 7) The final R_{free} value 6) warnings given by refinement with B for high bonds / angles, W for R_{work} better than R_{free} and F for R_{free} increasing during refinement. In general, red/orange lines come from the 3.3 Å map, yellow from 2.5 Å map, and all the rest from the 2.3 Å map. The green lines go through the oldest versions of software with darker greens having no resolution cutoffs and lighter greens have a 3.3 cutoff. Blue lines go through the intermediate software and purple lines through the latest software.

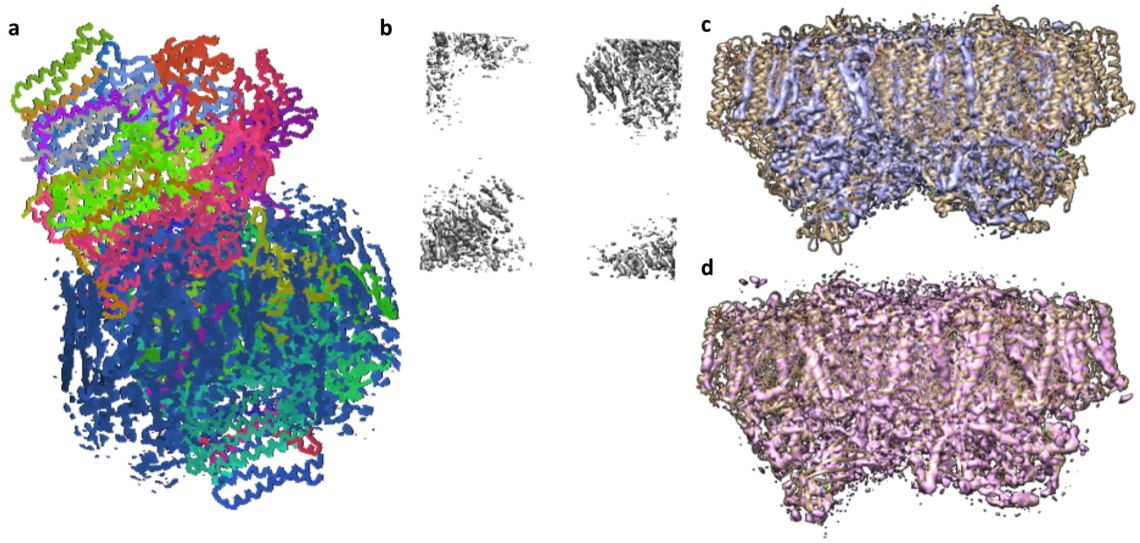

Figure 93. Refinement alignment

Refining the map produced by iterative phasing had various issues related to alignment. (a) Initially, the electron density was almost at right angles with the corresponding PDB, resulting in high R factors. (b) Attempts to align the PDB with the electron density in *chimera* initially suffered from the map opening incorrectly. (c) A third problem aligning the map and PDB occurred with a software update that required an additional undocumented scale parameter, without which the electron density was shrunk relative to the PDB, also resulting in high R factors. (d) For comparison, an early map before the software update with correct scaling.

2004) for aligning the PDB coordinates to the electron density map before refinement but that was not straightforward either. Initially, the map opened incorrectly, as seen in fig. 93b.

The next two values show the initial results before any alignment was done on the PDB file with no refinement resolution cutoff (0.4574) and a 3.3 refinement resolution cutoff (0.3008) for the 2.3 Å merge cutoff data. Surprisingly, these R values are slightly better than the corresponding R values of the aligned map (0.4607 and 0.3028 respectively).

The next set of two values tested an even newer software update, but the R values were high for no refinement resolution cutoff (0.5636) and a 3.3 Å refinement resolution cutoff (0.5583) showing that the newest software still had the high R value problem shown in the first four refinements. The final set of six refinements narrows the problem with the software update to the *gen_ map* script. Each pair has the newest *gen_ map* software followed by the oldest *gen_ map* software. The pairs are all from the 2.3 Å merge cutoff with refinement cutoffs of none, 3.3 Å and 2.3 Å respectively and in each case the old software had a better R value (compare 0.5772 to 0.5119, 0.4412 to 0.3843, and 0.5719 to 0.4724).

The problem with the newest *gen_ map* was eventually traced to an undocumented update to the script so that the script required a scale factor. The effect of this is visible in fig. 93c,d which are from the newest and oldest *gen_ map* respectively. Figure 93c is visibly shrunk compared to fig. 93d corresponding to a default scale factor of 1 instead of the correct scale factor.

With the major source of R values over 0.5 found, variations on the number of phase retrieval iterations and refinement iterations were tested and are summarized in table 22. Initially, 200 iterations of phase retrieval with the final 100 averaged were used. Another reconstrution was made with 1500 iterations and averaging after the 1000th iteration for the 2.3 Å map. The first column shows R values with the oldest *gen_ map* script. The next two columns use the newest *gen_ map* script without and with the scale factor respectively. The final column increased the number of refinement iterations from 3 to 8.

Increasing the phase retrieval iterations did not help with the incorrect *gen_ map* but did improve the R values in the last two columns. Increasing the refinement iterations to 8 helped with the 1500 iteration reconstruction but not the 200 iteration

Table 22. Software influence on R values

Map @ Refine Res	Old Gen_Map	Before Voxel Fix	After Voxel Fix	After Voxel Fix (8)
1500 @ 2.3			0.4885	
1500 @ 3.3		0.5583	0.4171	0.403
200 @ 2.3	0.4724	0.5719	0.5236	
200 @ 3.3	0.3843	0.4413	0.4465	0.4491

Maps are identified by number of iterations phasing was run for and the refinement resolution cutoff in Å. The columns are different versions of the volume to map script (*gen_map*) with the final column having more refinement cycles than the second to last.

reconstruction. Variations on the refinement algorithms were also tested and are summarized in table 23.

Table 23. Refinement and map influence on R values

XYZ	RS	Occ	I-B	RB	G-B	NCS	1500 map	200 map
X	X	X	X				0.4030	0.4491
X	X		X	X	X		0.387	0.4481
X	X		X	X	X	X	0.3792	0.4508
X			X	X			0.4208	0.4547

Refinement options used for 8 cycle refinement for the 1500/1000 iteration map and the 200/100 iteration maps (phasing resolution at 2.3 Å and refinement resolution at 3.3 Å). RS is for real space, Occ for occupancies, I-B for individual B factors, RB for rigid body, G-B for group B factors and NCS for non-crystallographic symmetry.

The 1500 iteration reconstruction always did better than the 200 iteration reconstruction. However, the best refinement parameters are unclear because they differ for the two reconstructions. The 1500 iterations reconstruction did best with the third line of algorithms, but the 200 iterations reconstruction did best with the second line of algorithms.

7.3 Resolution Parameter Screening

After the general flow of the software appeared to be working, finer screens were done. The first parameter focused on the number of phasing iterations. The initial choice of 200/100 (total iterations, iteration to begin averaging) was a quick option to see if the software input and output was working correctly. In addition to 200/100, 1000/500 and 1500/1000 were tried. The second parameter screened was the merge cutoff in steps of 0.2 Å. Finally, the refinement cutoff either matched the merge cutoff or was limited to 4 Å for comparison with other cutoffs.

Results are given in fig. 94. None of the R values are particularly good, since a value less than 0.3 is desirable. With increasing resolution, the R values also rose (see fig. 94a). In contrast to the prior result, with a constant refinement cutoff at 4.0 Å, increasing merge resolution cutoffs did not improve R values. Also in contrast to the prior result (see table 23), increased reconstruction iterations did not have better R values. Instead, the 200/100 iterations usually had the best R values.

To follow up on the R values, plots of R value by resolution for each map cutoff for the 1500/1000 iteration reconstructions are shown in fig. 95. The last value on every plot is abnormally high, possibly from plotting from the log files instead of using the *phenix* GUI. However, the more interesting feature is the jump in R values from under 0.4 to around 0.5 after 4.5 Å resolution on every plot. This corresponds to the point where only continuous diffuse scattering data is used. This feature is not unique to this analysis and is visible in supplementary figure 1 of (Ayyer et al. 2016).

At this point, it is worth discussing R values in relation to continuous diffuse scattering. The R value was described in section 6.1 and is a measure of how well the

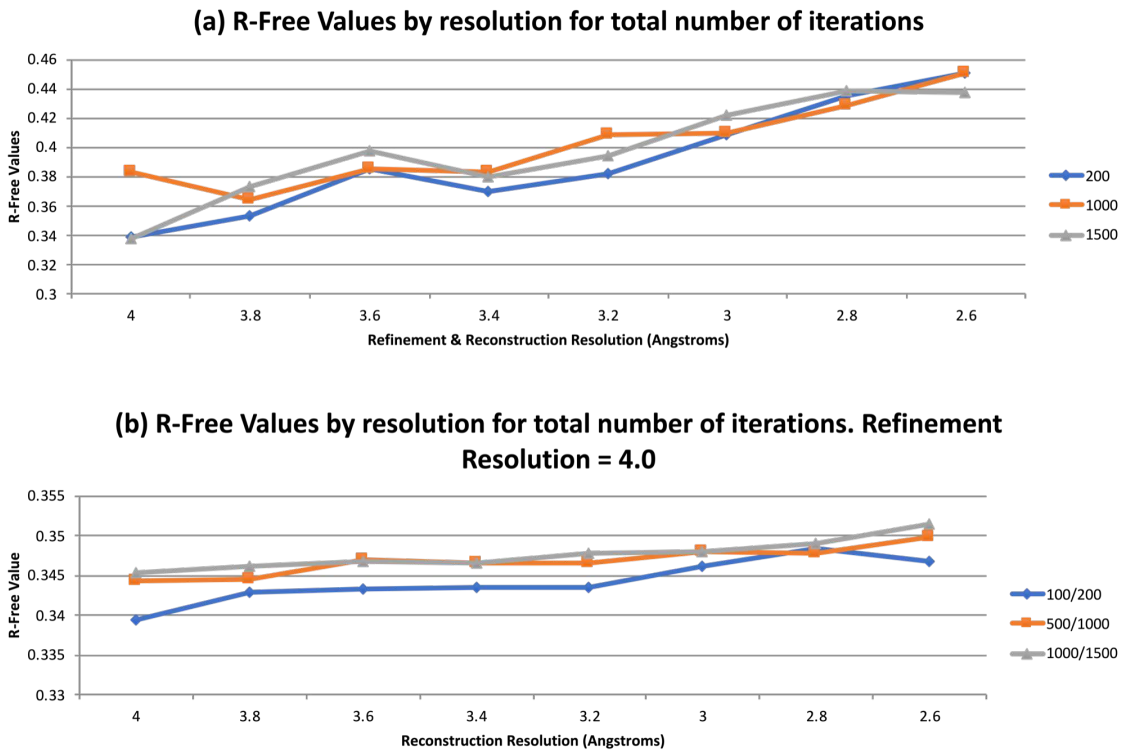

Figure 94. Phasing cutoff and iteration screen

A screen of the three iteration conditions (100/200, 500/1000 and 1000/1500 where the first number is the number of iterations without averaging and the second number is the total iterations) and phasing resolution cutoffs (4 Å to 2.6 Å with 0.2 Å intervals). (a) the final R_{free} value after refinement with a matching resolution cutoff to the edge of the map. (b) the final R_{free} value with refinement resolution of 4 Å for all conditions. Refinement was run with default settings.

observed structure factors match the ones calculated from atomic positions. R values of 0.5 can be obtained with a random model. So, one interpretation of the R value by resolution plots is that the phases determined by continuous diffuse scattering are not acccurate. However, there is an argument that the R value is not an appropriate statistic to use with continuous diffuse scattering since the R value is tied to structure factors that are in turn related to Bragg peaks. The observed structure factors used in refinement of continuous diffuse scattering do not come from Bragg data, but instead

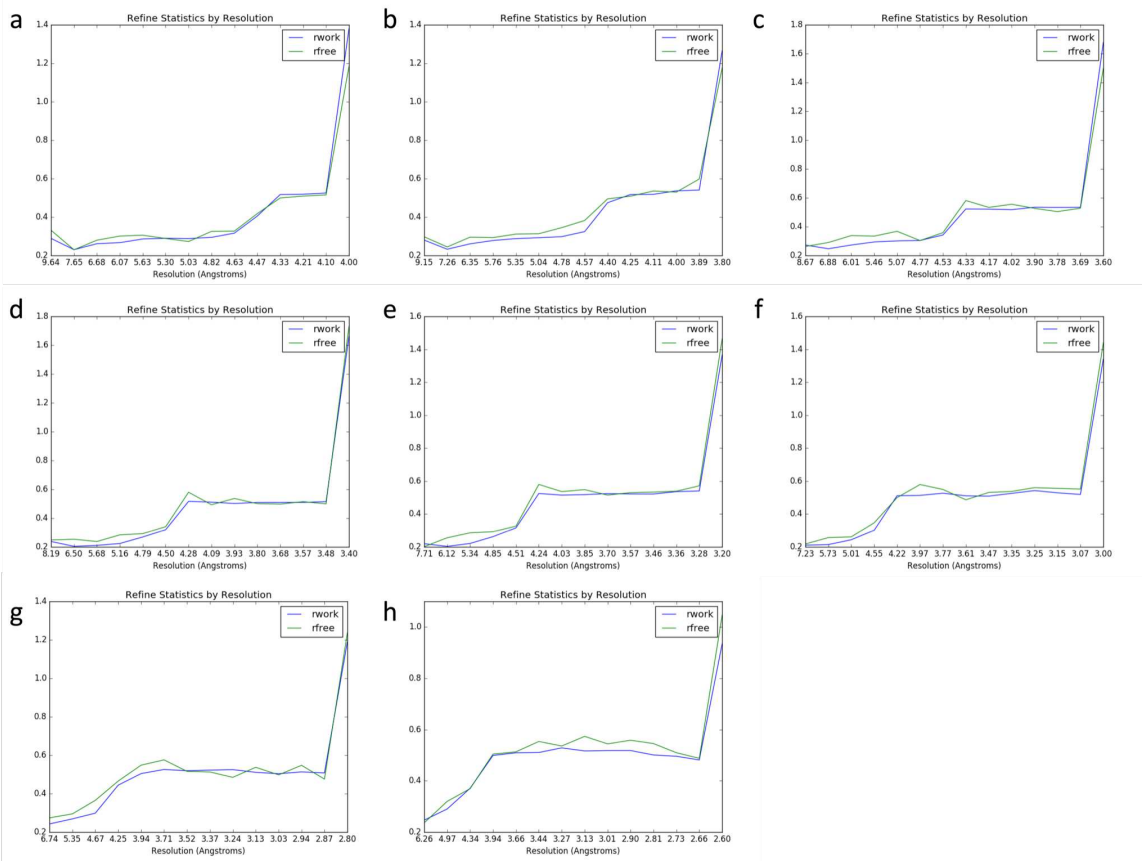

Figure 95. Phasing cutoff screen: R by resolution

For the 1500/1000 iteration map, the R values by resolution plots from *phenix* showing a jump around 4 Å on every plot to around 0.6 R values. (a) 4.0 Å (b) 3.8 Å (c) 3.6 Å (d) 3.4 Å (e) 3.2 Å (f) 3.0 Å (g) 2.8 Å and (h) 2.6 Å map and refinement resolution cutoffs.

are calculated directly from the electron density map produced by the continuous diffuse scattering phase retrieval.

Some results in this chapter support the claim that R values are not appropriate measures of continuous diffuse scattering analysis quality. For example, the fact that the completely misaligned maps (see fig. 93a) had better R values than aligned ones. Also, results presented in section 7.4.1 have similar R values for very different 3d merges (compare the R values in table 25 with the merges shown in fig. 99). Both of

those results support the conclusion that R values do not relate to continuous diffuse scattering. However, the R values did consistently show that reconstructions using an incorrect *gen_map* script were worse than those with the correct *gen_map* script, allowing the bug in that script to be identified. So, the usefulness of the R value is not certain.

However, at the time of these analyses, the usefulness of the R value as a metric had not been questioned. Results presented in this chapter will therefore still include R values when they were calculated and the direction of analysis was guided by the R values. Additional statistics are presented when useful. A takeaway message from this discussion is that more statistics are needed, particularly for intermediate results.

With the results in fig. 95, the initial conclusion was that the quality of the continuous diffuse scattering needed to be improved. This conclusion stemmed from the high R values from continuous diffuse regions (which were not questioned at the time) and the fact that increased reconstruction iterations were not improving the R values suggesting the problem was with the data rather than parameters.

Problems with the merge used so far are not entirely unexpected. After all, in section 7.2.1, it was shown that some runs could cause artifacts in the merge. Neither the data included in the merge nor the background correction of the merge had been screened or optimized. Also, unit cell variations had been noticed in the dataset. Figure 96 shows the unit cells by groups of runs along with some metadata. On the left hand side, notes from the log book identify potential factors impacting data quality. On the right hand side, the three lines of text give the range of runs, and the geometry and cell files used with indexing. Some runs did not record the camera length correctly and used a variant geometry file with the camera length (clen) fixed. The colored bars in the background show the PSII sample label and the laser scheme

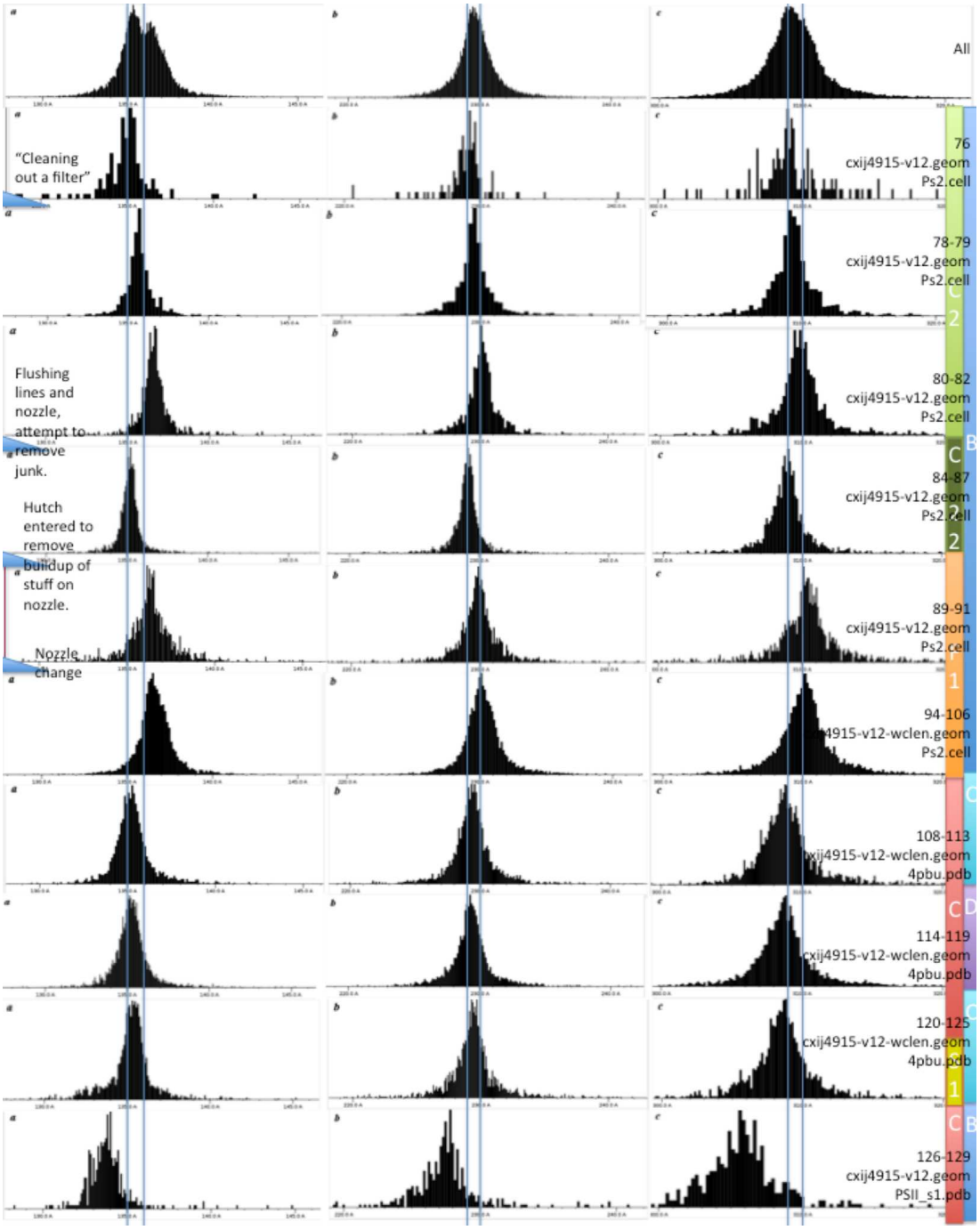

Figure 96. Unit cell variation

An overview using *cell_explorer* for LCLS October 2015. Along the right are comments from the run log. The far right shows the laser scheme as a colored bar, and the second to last bar shows the sample. The cell file and geometry used for each indexing is also written on the left. A faster method of viewing this type of information is to use *DatView* as done for the similar fig. 10.

label as the left and right columns respectively. For laser delays, B stands for '700 ; 1200; 600' (for the transient S4 state), C for '700 ; 1200 ; 2000' (for the S0 state) and D for '700 ; 1200' (for the S3 state) with all values in μs. The datasets were collected with alternating light and dark patterns except for two runs in the D laser scheme that were all dark. The *cell_ explorer* windows were manually lined up with vertical lines showing the shifts in different run groups.

Two subsets of the quinone dataset were created. One included only runs with the light (class 0) B laser scheme and the other included the C and D laser schemes. C and D were combined both because of their similar unit cells and because there were fewer patterns (compare 12,430 B class 1 (light) patterns to 9,018 patterns from C and D combined). This combination is only to get enough patterns for parameter optimization, as the light data must be separated to compare states. The CD dataset has 2,961 dark patterns, 4,365 light patterns for the S0 state, and 1,692 patterns for the S3 state.

One other possible factor of merge quality was tested. So far, merges had used algorithm 7 for background correction. The CD dataset was also run with algorithm 5. As before, the merges resolution cutoffs were screened in steps of 0.2 Å and refinement was run with matching resolution cutoff or with a resolution cutoff of 4 Å.

Interestingly, a clear trend was visible with refinement resolution cutoff at 4 Å of background subtraction algorithm 5 performing better than the background subtraction algorithm 7. However, there wasn't a clear difference between using the full dataset, only those with the B laser scheme, or the CD dataset with more consistent unit cells. The next section (7.4) presents a more in-depth analysis of the effect of subsets on merges. Section 7.5 gives a short treatment on background subtraction with a focus on detector artifacts in LCLS August 2016.

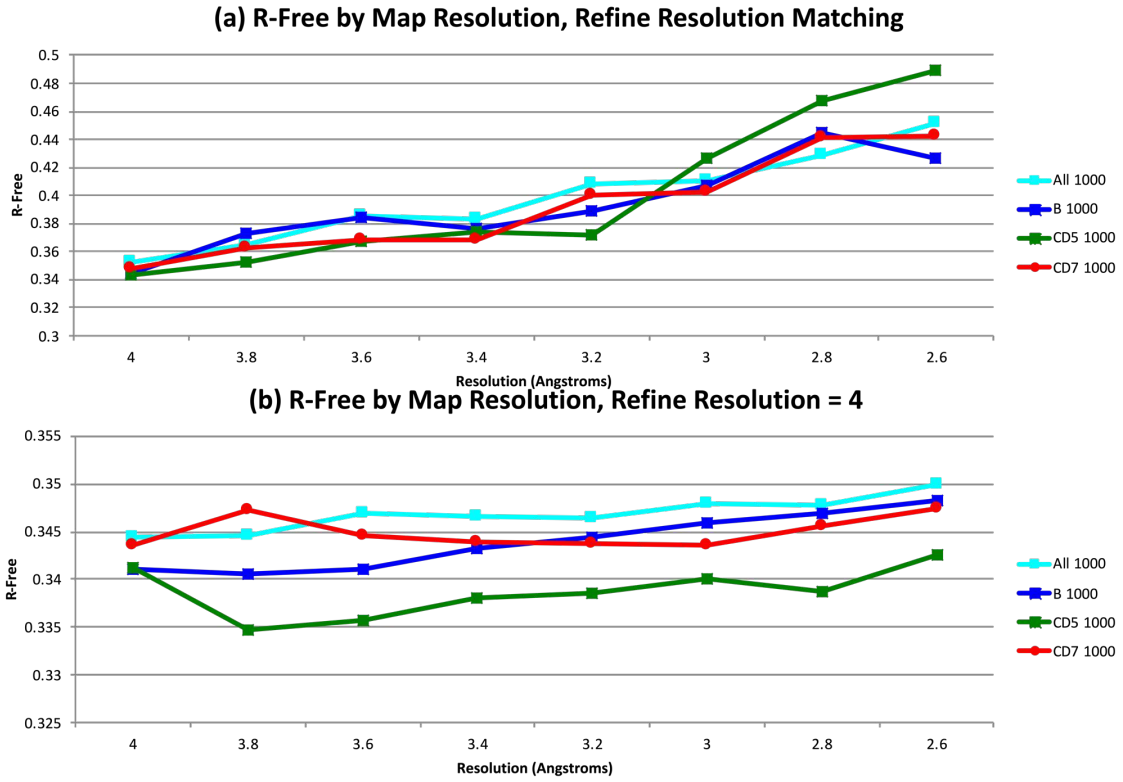

Figure 97. Phasing cutoff and laser subset screen

Using 1000/500 iteration maps for all conditions, a comparison with the all merge used to this point (teal) and subsets of the data based on laser scheme. The B subset has 12,430 alternating patterns from transient S4 laser delays. C has alternating patterns from S0 state laser delays and D has alternating patterns from S3 state laser delays with all patterns from runs with no laser. C and D runs 108-122 were combined for parameter optimization only because of their similar unit cell for a total of 9018 patterns. All the merges algorithm 7 for background subtraction except CD5 which used the algorithm 5. (a) the final R_{free} value after refinement with a matching resolution cutoff to the edge of the map. (b) the final R_{free} value with refinement resolution of 4 Å for all conditions. Refinement was run with default settings.

7.4 Subset Screening

In (Ayyer et al. 2016), only 2,848 patterns out of 25,585 indexed patterns were used for analysis. The selected patterns had strong continuous diffuse signal, however the script used to identify the patterns was not available. The initial purpose of this section was to identify parameters in the stream file that correlated with better continuous diffuse scattering as a replacement for the script. The expected outcome was that subsets with better quality would have better R values, particularly at resolutions relying solely on continuous diffuse scattering. The majority of the analysis was done for LCLS October 2015 and is presented in section 7.4.1. After LCLS August 2016, a follow up repeated portions of the analysis with that dataset, presented in section 7.4.2.

7.4.1 LCLS October 2015

The first step was identification of parameters to use for pattern selection. One obvious parameter is the unit cell. However, the unit cell has six parameters so they needed to be combined into one value. Figure 98 shows the number of patterns (y axis) within a cell axis tolerance (x axis) and angle tolerance (lines). The takeaway message from the figure is that cell axis tolerance and angle tolerance are not tightly correlated. Therefore, three variations of unit cell parameters were used. The cell axes is the distance from the average cell axes, the angle distance is the distance from the angles, and the cell distance is the distance for all six parameters.

In addition to the three unit cell criteria, four additional criteria were considered: the diffraction resolution limit and number of implausibly negative reflections from

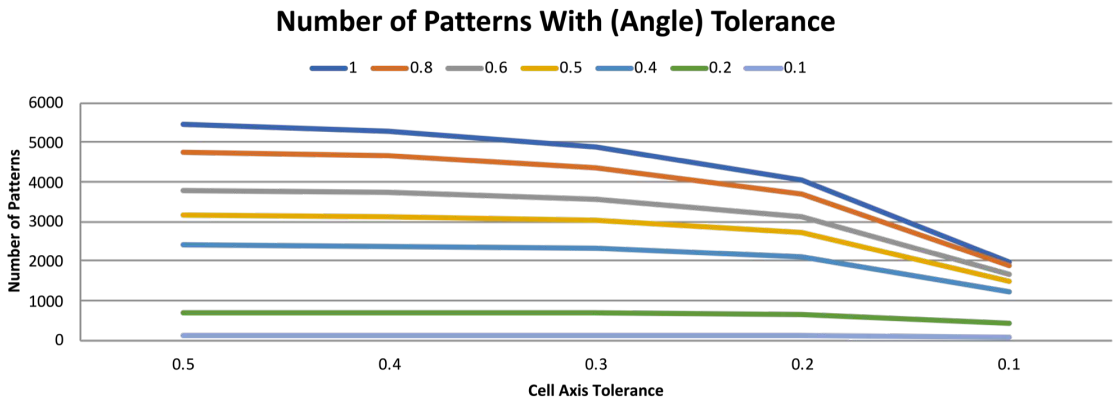

Figure 98. Phasing cutoff and iteration screen

The distance of the unit cell axes from 13.522, 22.928 and 30.86 nm and from cell angles from 90 degrees was computed per pattern. The count of patterns with distances less than or equal to the values is plotted with the count along the y axis, the cell axes distance on the x axis, and the angle distance as separate lines. This is a trimmed plot showing only cells with an average distance of less than 5 Å and angles with less than 1 degree distance.

the stream file (see sections 3.1.2 and 3.1.4) and the cross correlation and scale output by *process_hkl* (see chapter 5). A total of 9,037 indexed patterns from LCLS October 2015 runs 108-122 (combined for parameter optimization only) were split into three bins for each of the seven parameters as summarized in table 24. Cutoffs for the bins were chosen to result in roughly equal numbers of patterns in each bin, and a random selection of 3,000 was created as a control.

Each subset was merged with background subtraction algorithm 5 since it appeared better in fig. 97. The XY slices are shown in fig. 99. Each set of three in a row is the three bins created from a parameter. The trends in the merges make sense.

For the first row showing the unit cell, the merge with the most visible contrast contains the patterns with the smallest deviation from the unit cell, and as the deviation from the unit cell increases in the middle and big bins the visible merge

Table 24. Subset criteria

Parameter	1st Third	2nd Third	3rd Third
Cell	$x < 0.62$	$0.62 \leq x < 1.43$	$x \geq 1.43$
Distance	3019	3023	2995
Angle	$x < 0.58$	$0.58 \leq x < 1.35$	$x \geq 1.35$
Distance	3023	3005	3009
Cross	$x < 0.172$	$0.172 \leq x < 0.343$	$x \geq 0.343$
Correlation	3019	3005	3011
Scale	$x < 0.093$	$0.093 \leq x < 0.1536$	$x \geq 0.1536$
	3009	3010	3016
Axes	$x < 0.15$	$0.15 \leq x < 0.35$	$x \geq 0.35$
Distance	3014	3040	2983
Resolution	$x < 1.0$	$1.0 \leq x < 1.24$	$x \geq 1.24$
Limit	3064	2963	3010
Implausible	$x < 7$	$7 \leq x < 25$	$x \geq 25$
Reflections	3076	2937	3024

9,037 patterns from LCLS October 2015 runs 108-122 were split into three bins for 7 different parameters. This table shows the parameter, the edges of the bin, and the count by the three bins (thirds of data). The axes distance and angle distance are described in fig. 98. The cell distance is the distance for all six (axes and angles) values. Cross correlation and scale are output from *process_hkl* merging. The diffraction resolution limit and number of implausible reflections are indexing parameters described in chapter 3.

quality decreases. Similarly, for the angle datasets comparing unit cell angles and the axes datasets comparing only the axes, the bins for the patterns with the smallest deviations have higher contrast than the other bins which decrease in contrast as patterns have more unit cell variations.

The third row in the figure shows the CC to the merge output by *process_hkl* and the diffraction resolution limit bins. The 3D merge containing patterns with small correlation to the *process_hkl* merge had low visual contrast. The most visual contrast in the merge is observed with the bin containing patterns with the highest correlation to the *process_hkl* merge. Similarly, patterns with small diffraction resolution limits resulted in a merge with low visual contrast (recall the diffraction resolution limit is

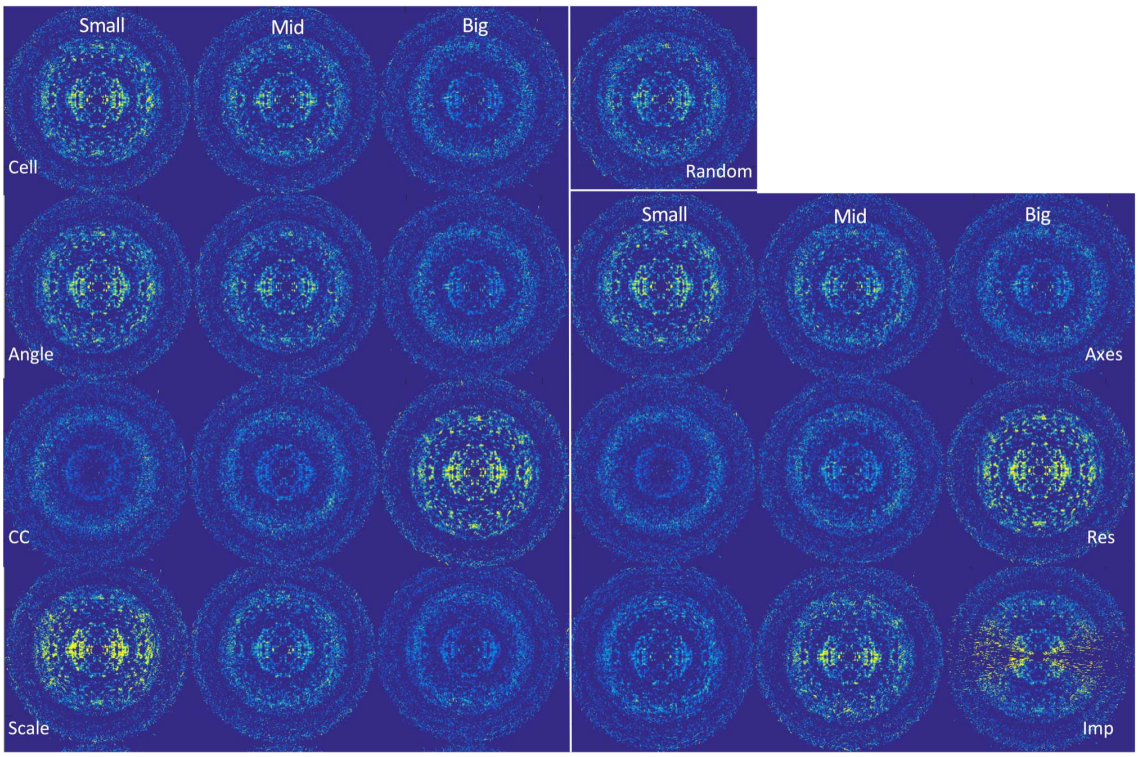

Figure 99. Subset merge slices

The XY slice of the merge for all subsets described in table 24. Each set of three in a row is a parameter, and within each set of three the first one is from the first bin (small values), the second from the middle bin (middle values) and the third from the large bin (big values). A random subset with 3,000 patterns is shown as the fourth merge in the top row for comparison. Cell is for distance from all six unit cell values, angle for angle distance, CC for cross correlation to merge, axes for cell axes distance, res for diffraction resolution limit, and Imp for number of implausible reflections.

in nm^{-1}). The merge with the best visual contrast for diffraction resolution limits is the one containing the best diffracting patterns.

The final row in the figure is odd. For the bins created from the *process_hkl* scale, the merge with the most visual contrast contains the patterns that were scaled the least. That part of the trend makes sense. However, that merge appears to be brighter in the horizontal direction than in the vertical direction. This is the inverse of the merge from patterns containing the highest number of implausibly negative

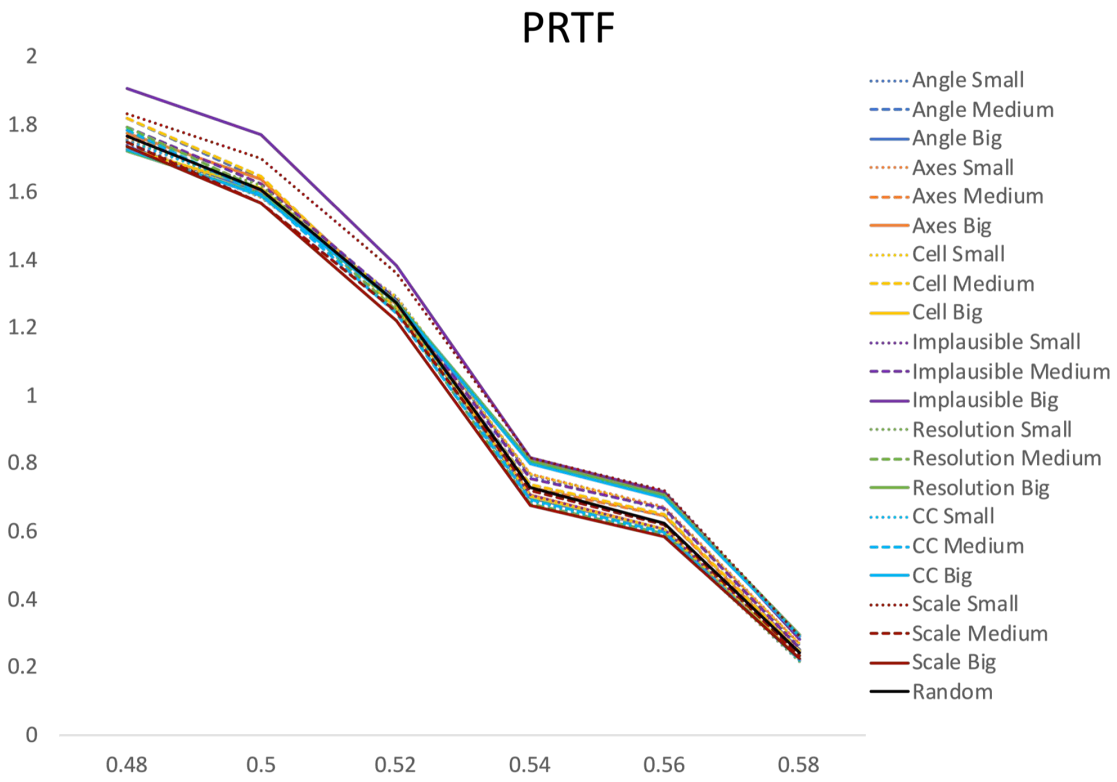

Figure 100. Subset PRTF

The PRTF output for the subsets described in table 24.

reflections were data is missing in the horizontal direction and present in the vertical direction. This suggests that both of these parameters relate to the orientation of the pattern. This could be related to having more peaks in the direction of the longest unit cell axis. The proximity of the peaks may impact the integration, resulting in more implausibly negative reflections. The scale trend would then be inverted because patterns with many implausibly negative reflections would have higher scale factors. The merge from patterns with the highest number of implausibly negative reflections is missing the most data, and the merge with the smallest scale values has the second most missing data.

All merges were used for reconstruction. A comparison of the prtf output is shown in fig. 100. Surprisingly, the highest values come from the bin with the large numbers of implausibly negative reflections, which as shown in fig. 99 had missing cones of data.

Table 25. Subset R values

Dataset	Small	Medium	Big
Cell	0.3404	0.3396	0.3424
Angle	0.3399	0.3404	0.3427
CC	0.3425	0.3411	0.3425
Scale	0.3391	0.3402	0.3413
Axes	0.3418	0.3396	0.3392
Resolution	0.3432	0.3372	0.3395
Implausible	0.3391	0.3387	0.3427

Final R_{free} values for the subsets described in table 24. The random subset had a value of 0.3412.

However, despite obvious differences in visual merge quality, the R factors from refinement were all very similar, as shown in table 25. There are several possible explanations for this. As discussed above, the R values may not be an appropriate statistic for continuous diffuse scattering. Another possibility is that the shell of data used for continuous diffuse scattering analysis was not significantly different for the different merges. Recall that only the outer shell of continuous diffuse scattering data is phased because the inner sphere of the merge contains Bragg peaks. In this case, only the shell from 4.5 to 4 Å was iteratively phased. Another explanation is that visual merge quality is not an indicator of continuous diffuse scattering data quality. Or, something is wrong with the phasing step so it doesn't matter what data is used.

A possible problem with the phasing step is suggested by the high performance of the implausibly negative reflections large bin that had missing cones of data and yet had high values of the prtf output. The phase retrieval could have appeared better for

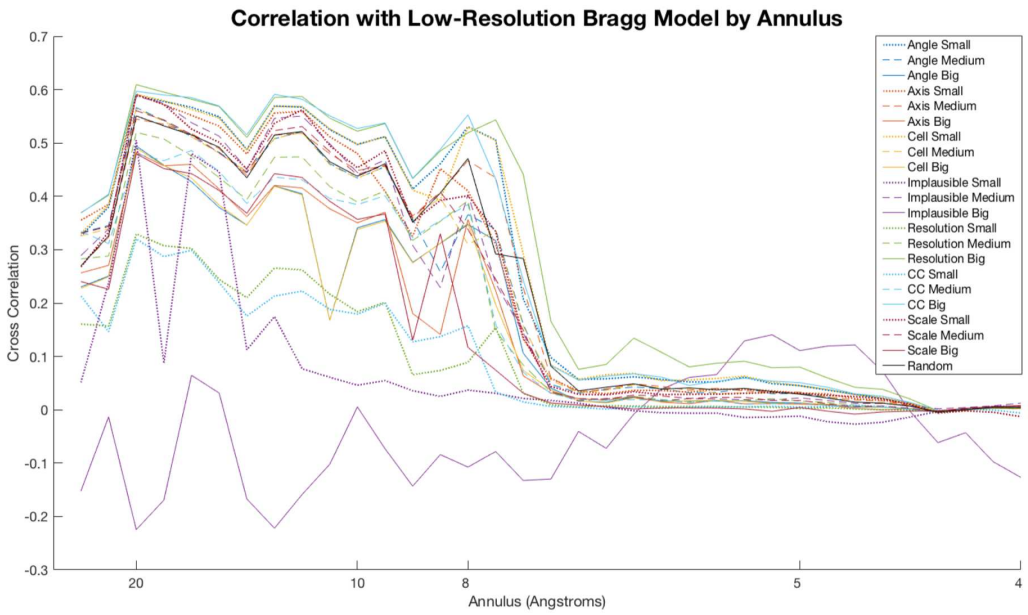

Figure 101. Subset correlation to Bragg model volume

The cross correlation of the 3D merge volume to the low resolution Bragg volume by annulus for the subsets described in table 24.

that dataset because it was missing so much data and therefore had fewer constraints. The phase retrieval step could be reinforcing the low resolution support instead of extending the resolution of the dataset. This hypothesis is supported by the fact that the scale bin with the smallest values which performed second best by prtf also has the second most missing data.

With the R values largely similar and the prtf apparently biased by missing data, it's desirable to have a numerical value that relates to the visual merge quality. This would allow quantitative comparison of the merges shown in fig. 99. The cross correlation of the 3D merge to the Bragg volume used to remove the Bragg peaks was computed for all the subsets and shown in fig. 101. It matches the trend of visual contrast well with the merge with the highest diffraction resolution limits appearing

the best at higher resolutions (solid green line) and equivalent to the merge with patterns that had a large CC (solid light blue) to the *process_hkl* merge at lower resolutions. The merge with the patterns with the highest number of implausibly negative reflections (solid purple line) has the least correlation which makes sense since it has the most missing data.

The major result of the analysis is that the diffraction resolution limit is favored as a pattern selection criterion. While the CC also performs well, it relies on a second step of merging. Likewise, the scale will be impacted by merging. The number of implausibly negative reflections, while an interesting set, has orientation bias. The unit cell criteria may be biased by having to choose an average unit cell and didn't have as high a CC with the Bragg data as the CC and diffraction resolution limit subsets.

7.4.2 LCLS August 2016

The analysis in section 7.4.1 was repeated for LCLS August 2016 with a few changes. First, all light data (for the transient S4 state) from LCLS August 2016 was used so that each subset was roughly 10,000 patterns instead of 3,000. It was hoped that additional patterns would reduce the number of empty values. Second, the pulse duration was recorded for each shot for this experiment, so binning was done by pulse duration to see if different pulse lengths had an influence. Third, LCLS October 2015 had been indexed with a fixed profile radius, but LCLS August 2016 was indexed without a fixed profile radius. The profile radius was therefore also added as a criterion. Finally, the distribution of pattern implausibly negative reflections was different (there were different indexing parameters) for this dataset with almost

half the data having 0 implausibly negative reflections. Therefore, only two bins of implausibly negative reflections were created, each limited to 9,740 patterns and randomly selected from the set with 0 implausibly negative reflections or the set with at least one implausibly negative reflection. The criterion, bin edges, and count in each bin are given in table 26.

The XY slice for each merge is shown in fig. 102. One immediately noticeable fact is that the merges are all surrounded by bright yellow halos. This is later identified with problematic runs and background correction in section 7.5. Another difference when compared to LCLS October 2015 is that the bin with implausibly negative reflections does not appear to have missing cones of data. As before, the merge with 0 implausibly negative reflections has low contrast. The cell, angle, axes, scale, diffraction resolution limit, and CC criterion all follow the same trends as before.

Interestingly, the pulse duration bins don't show a difference. This suggests that the pulse duration did not make a significant impact on the continuous diffuse scattering quality. The result isn't entirely surprising since the range of pulse durations was small, from 7 to 23 fs. More surprising is the profile radius where patterns with a larger profile radius had the strongest visual contrast in the merge. Since the profile radius is a measure of indexing accuracy, it is intuitive to assume that better indexed patterns would have a more accurate merge. A possible explanation is that the profile radius is larger for patterns with more diffuse Bragg peaks and thus larger profile radii correspond to patterns with more diffuse scattering.

Rather than compute the CC of each merge with the Fourier Transform of the Bragg data, the CC was computed for half merges. This is preferable because a Bragg

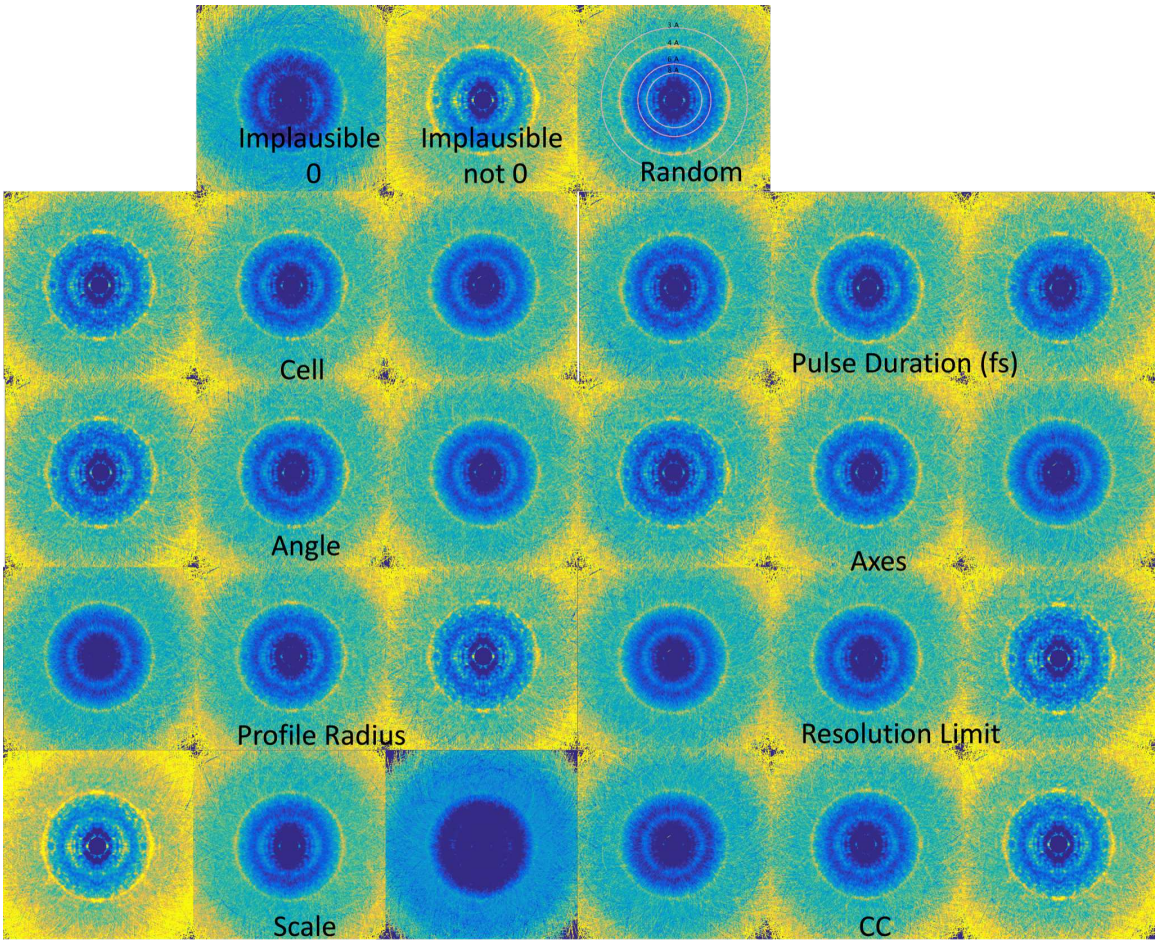

Figure 102. Subset merge slices

The XY slice of the merges for all subsets (see table 26 for selection constraints). Each set of three in a row is a parameter, and within each set of three the first one is from the first bin (small values), the second from the middle bin (middle values) and the third from the large bin (big values). A random subset is shown in the top row for comparison. Cell is for distance from all six unit cell values, angle for angle distance, CC for cross correlation to merge, axes for cell axes distance, res for diffraction resolution limit, and Imp for number of implausible reflections. The source of the yellow halos and bright yellow ring are discussed section 7.5.

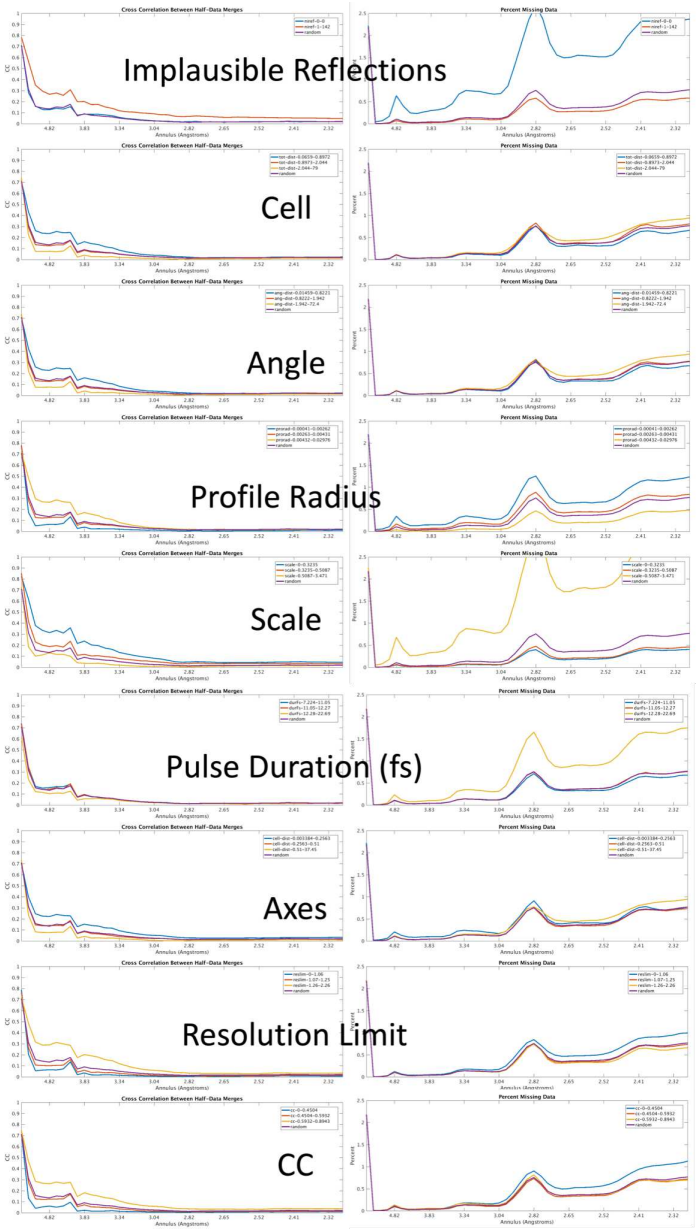

Figure 103. Subset merge plots

A pair of plots is shown for each criteria used for the subsets described in table 26 with the plot on the left showing the correlation between merges of half datasets by annulus and the plot on the right showing the percentage of missing data (-10002 flags) by annulus. The blue line is always the bin with the smallest values, red the bin with the middle values, and yellow the bin with the big values. Purple is the random subset and shown on every plot.

Table 26. Subset criteria

Parameter	Min	Max	Count
Implausible	0	0	9740
	1	142	9740
Cell	0.0659	0.8972	9740
	0.8973	2.044	9740
	2.044	79	9740
Angle	0.01459	0.8221	9740
	0.8222	1.942	9740
	1.942	72.4	9740
Profile Radius	0.00041	0.00262	9797
	0.00263	0.00431	9766
	0.00432	0.02976	9657
Scale	0	0.3235	9740
	0.3235	0.5087	9740
	0.5087	3.471	9740
Pulse Dur (fs)	7.224	11.05	9857
	11.05	12.27	9904
	12.28	22.69	8155
Axes	0.003384	0.2563	9740
	0.2563	0.51	9740
	0.51	37.45	9740
Resolution Limit	0	1.06	9871
	1.07	1.25	9946
	1.26	2.26	9403
CC	0	0.4504	9740
	0.4504	0.5932	9740
	0.5932	0.8943	9740

For LCLS August 2016, the subset bins. Roughly 30,000 "dark" patterns were split by thirds. Since 14,104 patterns had 0 implausibly negative reflections, that data was only split into two bins (0 and not 0) and a random subset of 9,740 patterns was selected from each bin to approximate the size of the other bins. A random subset was also created for comparison.

model for this dataset had not been created yet (only 3D merges had been generated).

Also, Bragg models are limited in resolution and may be biased by prior processing.

Results are grouped by criterion and given in fig. 103. The percentage of missing data

is also calculated and shown on the right of each pair. The CC between half merges correlates well with the visual quality of the merge.

As with LCLS October 2015, the diffraction resolution limit appears the best candidate for pattern selection for continuous diffuse merge quality. A further analysis was done to determine the best cutoff. The full stream file was sorted by diffraction resolution limit and the number of patterns included into the merged systematically increased. Including more patterns from the sorted stream file means the added patterns have lower diffraction resolution. Cross correlations between half merges were calculated in shells and results are given in fig. 104.

The coloring in fig. 104 is by rank with the best value in each column brightest yellow and the worst darkest blue. The results show that after around 24,000 patterns, including more patterns makes things worse. This is a particularly interesting result because usually including more patterns makes things better (see section 1.7.1).

7.5 Background Correction

This section describes background correction work done for LCLS August 2016 data. As mentioned in section 7.1.1, Dr. Oleksandr Yefanov's 3D merge software (Yefanov et al. 2014) has multiple modes. In another mode, it stores results from background correction frame by frame. For each frame it stores a graph by resolution with the average measured intensity, the raw background, the smoothed background curve, and the intensity after the smoothed curve is subtracted from the measured intensity. Additionally, it outputs the background corrected frame.

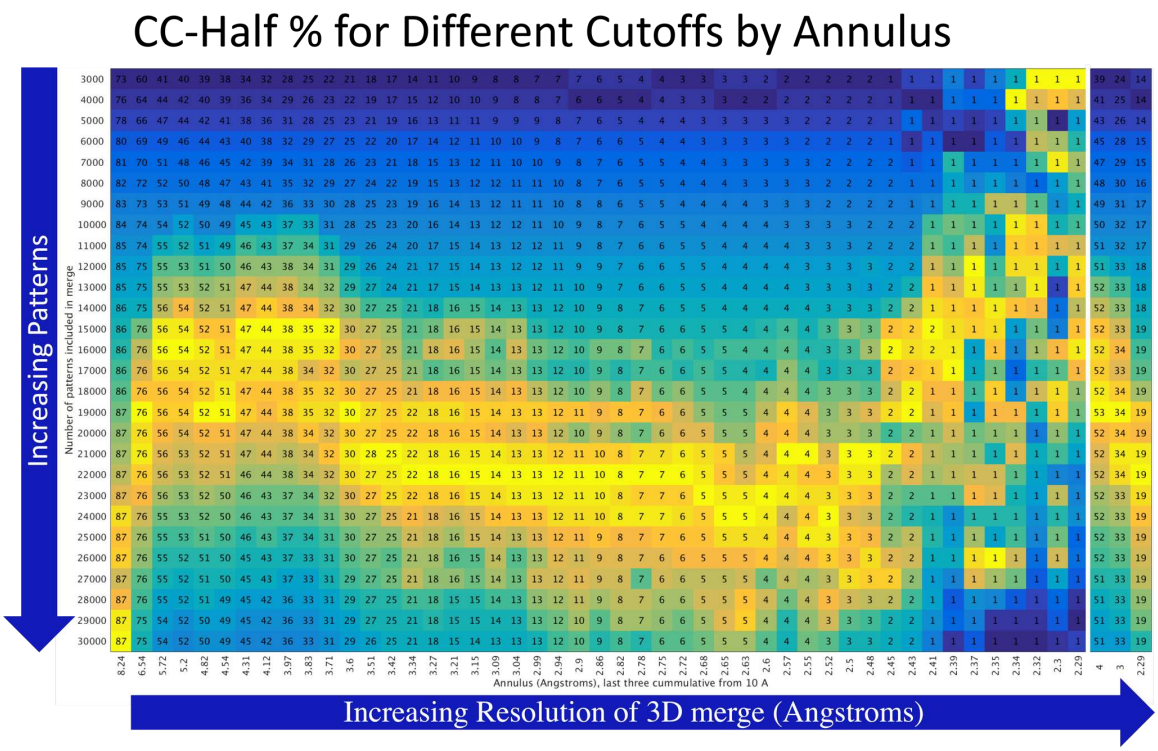

Figure 104. CC with increasing patterns

The cross correlation between half dataset merges for LCLS August 2016 with a stream file sorted by diffraction resolution limit. Down the y axis, increasing numbers of patterns from starting at the highest resolution patterns are included in the merge. Along the bottom is various annulus with increasing resolution. The final three columns are cumulative going to 4, 3, and 2.29 Å respectively. The CC % (raw value multiplied by 100 for display) is shown in each cell. The coloring is by rank with the best value in each row colored brightest yellow and the worst value in each row colored darkest blue.

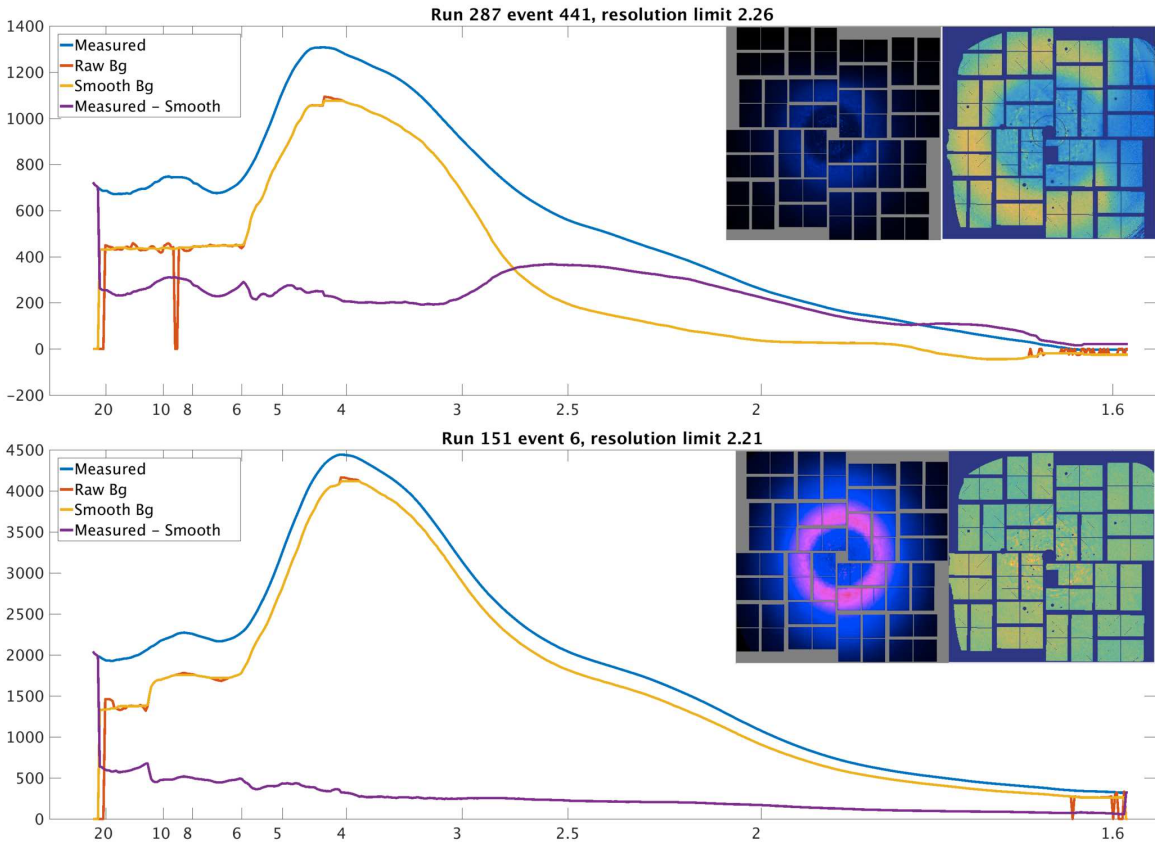

Figure 105. Radial background results by event

Results from the developmental background correction algorithm from March 2017 for two events from LCLS August 2016 data. The plots show, by radius, the measured value (blue), the initial calculation of background (red) determined by the background correction algorithm, the smoothed line of background (yellow) and the result after background correction (purple). The two pictures are a view of the event with *HDFsee* (right) and after background correction with *matlab* (left). The top event shows a detector shadow on the right because the image is transposed relative to the laboratory frame. In physical space, the nozzle shadow is on the top of the detector..

Results for an event from run 287 and run 151 are shown in fig. 105. The results from run 441 identify a nozzle shadow that was not obvious from the original image (generated with *hdfsee*). These plots were generated for a set of frames covering the range of diffraction resolution limits and for a set of frames containing a single frame from each run. They allowed identifying which runs contained nozzle shadowing, and a mask was created specifically for those frames. This analysis also identified one quadrant brighter than the rest, as described in section 1.5.

Merges were run separately for nozzle shadowed runs since they required a different mask file. The merges from shadowed and non-shadowed runs were combined with weighted averages, and the final analysis in section 7.4.2 repeated. Figure 106 shows the merge slices. Importantly, the yellow halos from before are now gone, showing that they came from including nozzle shadowed patterns without masks. The bright yellow ring at the inner edge of the yellow halos is also gone, indicating that it was probably the edge of the nozzle shadow.

The CC between half merges was also recalculated and results are shown in fig. 107. This time at low resolutions including all patterns was better. However, for higher resolutions, the number of patterns was best from 16,230 to 20,454 patterns.

Overall, this chapter has described continuous diffuse scattering analysis for LCLS October 2015 and LCLS August 2016 datasets. The main takeaway from this chapter is that there is still a lot of work to be done to use continuous diffuse scattering for resolution extension. The intermediate results in this chapter show that additional statistics are needed, especially for intermediate steps in the analysis. However, the CC between half merges appears to be a good candidate statistic that relates to visual merge contrast. Pattern selection does have an impact on visual merge quality with

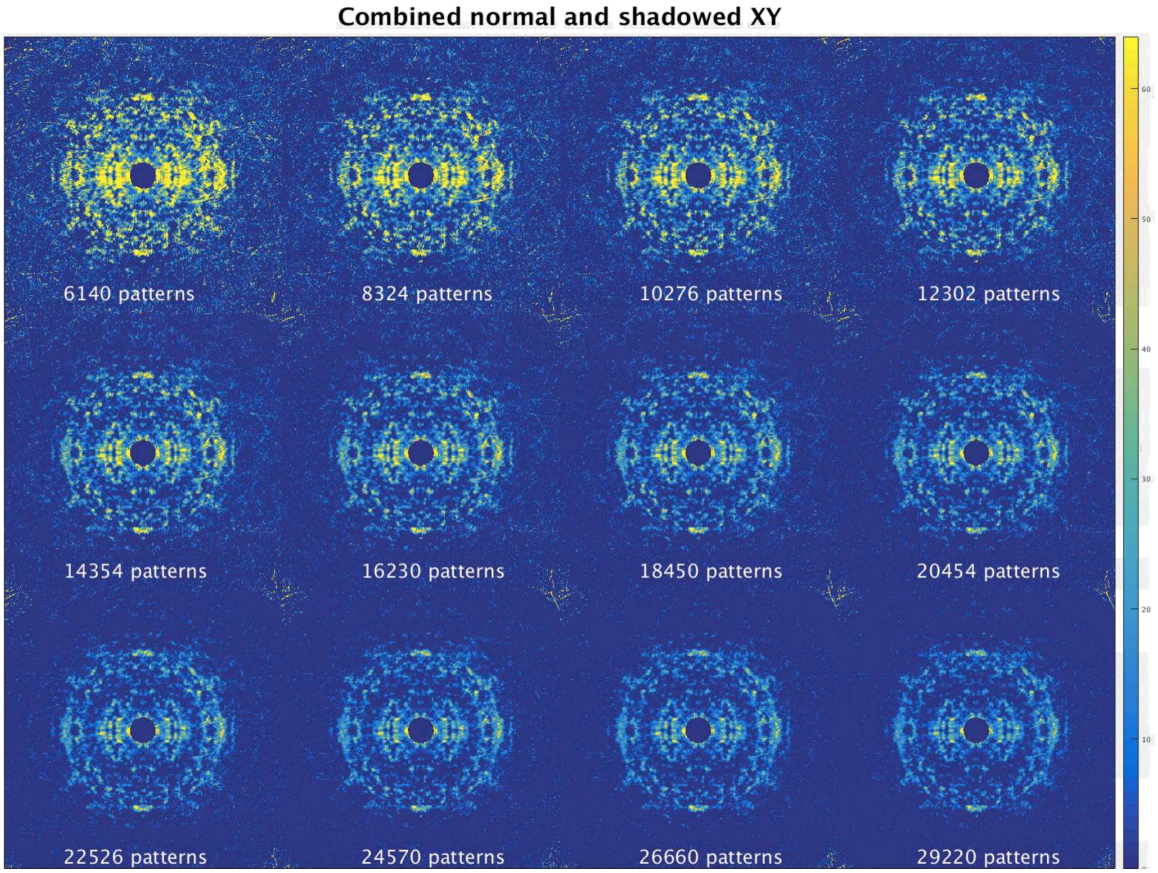

Figure 106. XY slices for increasing number of patterns

Runs with detector shadows were merged with appropriate masks separately from the remainder of the runs. The normal and shadowed sets were then combined. Results come from a stream file sorted on resolution limit, so increasing numbers of patterns corresponds to including patterns with lower resolution limits. The results are presented left to right in rows, so the number of patterns increases as across each row, and then continues on the next row from 6,140 patterns in the top left to 29,220 patterns in the bottom right

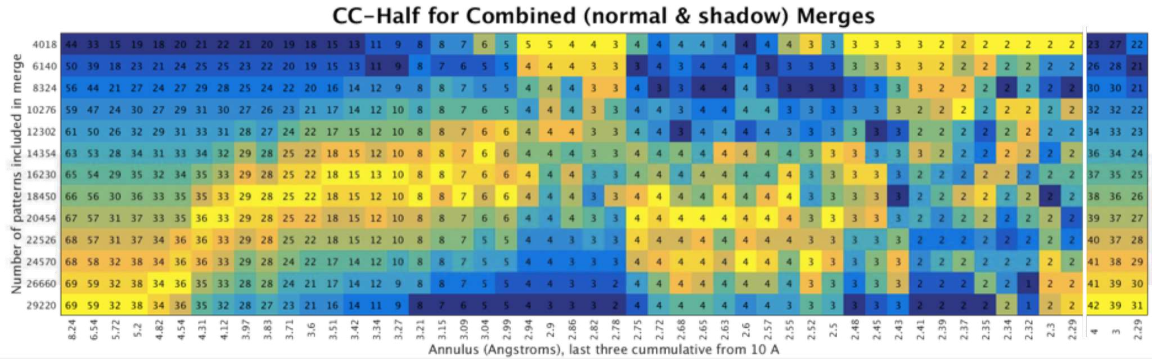

Figure 107. CC with increasing patterns background corrected

Similar to fig. 104, the cross correlation between half dataset merges for LCLS August 2016 with a stream file sorted by diffraction resolution limit. However, this data was processed with masks for shadowed runs (slices are shown in fig. 106. Down the y axis, increasing numbers of patterns from starting at the highest resolution patterns are included in the merge. Along the bottom is various annulus with increasing resolution. The final three columns are cumulative going to 4, 3, and 2.29 Å respectively. The CC % (raw value multiplied by 100 for display) is shown in each cell. The coloring is by rank with the best value in each row colored brightest yellow and the worst value in each row colored darkest blue.

the diffraction resolution limit output by *CrystFEL* as the most promising criterion. Including only a subset of patterns instead of the full dataset does improve the CC of the half merges at higher resolutions. Finally, careful attention must be paid to detector artifacts that can bias the merges.

Chapter 8

CONCLUSIONS AND FUTURE OUTLOOK

The subset of data and the selection parameters used are shown to affect data processing in several sections of the thesis. Sections 4.3.2, 5.1.2, and 6.3 show that including more high resolution patterns was preferable to having a narrow unit cell distribution. However, the initial set of data had no unit cell constraints. The second subset analysis in section 4.3.3 showed that a combination of unit cell and resolution constraints was better than only a resolution constraint, but correlation to a high quality merge was best. Further work is necessary to examine the effect of the resolution range used when calculating correlations. Further work is also necessary to show the general applicability of the results to other proteins.

Section 7.4 examined the effectiveness of several indexing parameters in selecting patterns for continuous diffuse scattering analysis. The results showed that the subset of patterns used had a visible effect on the 3D merges and favors the diffraction resolution limit as a criterion for selecting the best patterns. It introduced the CC between half-dataset 3D merges as a statistic correlating with visual merge contrast, and showed that including more low resolution patterns decreased the CC at higher resolutions. It also identified the impact of detector artifacts such as nozzle shadowing on continuous diffuse merges and showed that masking improves the results. Further work on continuous diffuse scattering analysis is necessary to identify useful statistics for phasing and refinement, and to extend the technique to other proteins. Scientists in Dr. Henry Chapman's group at CFEL are currently working on refinement algorithms and statistics for improving data analysis of continuous diffuse scattering.

Sections 3.3, 4.3.1, 5.1.1, and 6.2 show the effect of indexing and merging parameters on results. The major conclusion from the results is that an integration radius of rings-cen causes the intensities to appear twinned. Further work is necessary to determine if this result is general and can be applied to other datasets and proteins.

Section 2.3 introduced a script for improving *Cheetah* processing speeds. This has proved especially useful at the EXFEL, reducing processing times to roughly 30 minutes per run depending on the computational resources available. Future work in this area may be required as *Cheetah* or beam line software changes.

Finally, chapter 4 introduced the novel software *DatView*. *DatView* improves visualization of large datasets by reducing loading time and combining many visualization tools into a single interface that synchronizes the selection across all plots. It also provides important new functionalities for subset creation and export and individual item comparison. Finally, it is extendable through an external configuration file for use in fields outside of SFX. *DatView* opens up possibilities for future works. First, a more thorough comparison of various subsets is possible through *DatView*. Second, *DatView* allows per-pattern optimization allowing each pattern to have optimized parameters. Work examining combined parameter datasets on model datasets is necessary to determine the utility of the approach.

Aboalbiss. 2009. *Bragg's Law,* November 14. Accessed April 24, 2019. https://upload.wikimedia.org/wikipedia/commons/5/58/Bragg's_Law.PNG.

Adams, Paul D, Pavel V Afonine, Gábor Bunkóczi, Vincent B Chen, Ian W Davis, Nathaniel Echols, Jeffrey J Headd, L-W Hung, Gary J Kapral, Ralf W Grosse-Kunstleve, et al. 2010. "PHENIX: a comprehensive Python-based system for macromolecular structure solution." *Acta Crystallographica Section D: Biological Crystallography* 66 (2): 213–221.

Andruszkow, J, B Aune, V Ayvazyan, N Baboi, R Bakker, V Balakin, D Barni, A Bazhan, M Bernard, A Bosotti, et al. 2000. "First observation of self-amplified spontaneous emission in a free-electron laser at 109 nm wavelength." *Physical Review Letters* 85 (18): 3825.

Arvai, A. 2012. *ADXV–a program to display X-ray diffraction images.*

Assmann, Greta, Wolfgang Brehm, and Kay Diederichs. 2016. "Identification of rogue datasets in serial crystallography." *Journal of applied crystallography* 49 (3): 1021–1028.

Ayvazyan, V, N Baboi, I Bohnet, R Brinkmann, M Castellano, P Castro, L Catani, S Choroba, A Cianchi, M Dohlus, et al. 2002. "A new powerful source for coherent VUV radiation: Demonstration of exponential growth and saturation at the TTF free-electron laser." *The European Physical Journal D-Atomic, Molecular, Optical and Plasma Physics* 20 (1): 149–156.

Ayyer, Kartik, Oleksandr M Yefanov, Dominik Oberthür, Shatabdi Roy-Chowdhury, Lorenzo Galli, Valerio Mariani, Shibom Basu, Jesse Coe, Chelsie E Conrad, Raimund Fromme, et al. 2016. "Macromolecular diffractive imaging using imperfect crystals." *Nature* 530 (7589): 202.

Barends, Thomas, Thomas A White, Anton Barty, Lutz Foucar, Marc Messerschmidt, Roberto Alonso-Mori, Sabine Botha, Henry Chapman, R Bruce Doak, Lorenzo Galli, et al. 2015. "Effects of self-seeding and crystal post-selection on the quality of Monte Carlo-integrated SFX data." *Journal of synchrotron radiation* 22 (3): 644–652.

Barty, Anton, Richard A Kirian, Filipe RNC Maia, Max Hantke, Chun Hong Yoon, Thomas A White, and Henry Chapman. 2014. "Cheetah: software for high-

throughput reduction and analysis of serial femtosecond X-ray diffraction data." *Journal of applied crystallography* 47 (3): 1118–1131.

Bernstein, HJ, and AP Hammersley. 2006. "Specification of the Crystallographic Binary File (CBF/imgCIF)." *International Tables for Crystallography.*

Blankenship, Robert E. 2014. *Molecular mechanisms of photosynthesis.* John Wiley & Sons.

Bracewell, Ronald Newbold, and Ronald N Bracewell. 1986. *The Fourier transform and its applications.* Vol. 31999. McGraw-Hill New York.

Brehm, Wolfgang, and Kay Diederichs. 2014. "Breaking the indexing ambiguity in serial crystallography." *Acta Crystallographica Section D: Biological Crystallography* 70 (1): 101–109.

Chapman, Henry N, Petra Fromme, Anton Barty, Thomas A White, Richard A Kirian, Andrew Aquila, Mark S Hunter, Joachim Schulz, Daniel P DePonte, Uwe Weierstall, et al. 2011. "Femtosecond X-ray protein nanocrystallography." *Nature* 470 (7332): 73.

Chapman, Henry N, Oleksandr M Yefanov, Kartik Ayyer, Thomas A White, Anton Barty, Andrew Morgan, Valerio Mariani, Dominik Oberthuer, and Kanupriya Pande. 2017. "Continuous diffraction of molecules and disordered molecular crystals." *Journal of applied crystallography* 50 (4): 1084–1103.

Conrad, Chelsie E, Shibom Basu, Daniel James, Dingjie Wang, Alexander Schaffer, Shatabdi Roy-Chowdhury, Nadia A Zatsepin, Andrew Aquila, Jesse Coe, Cornelius Gati, et al. 2015. "A novel inert crystal delivery medium for serial femtosecond crystallography." *IUCrJ* 2 (4): 421–430.

Damiani, D, M Dubrovin, I Gaponenko, W Kroeger, TJ Lane, A Mitra, CP O'Grady, A Salnikov, A Sanchez-Gonzalez, D Schneider, et al. 2016. "Linac Coherent Light Source data analysis using psana." *Journal of Applied Crystallography* 49 (2): 672–679.

Dau, Holger, and Michael Haumann. 2008. "The manganese complex of photosystem II in its reaction cycle?basic framework and possible realization at the atomic level." *Coordination Chemistry Reviews* 252 (3-4): 273–295.

DePonte, DP, Uwe Weierstall, Kevin Schmidt, J Warner, D Starodub, JCH Spence, and RB Doak. 2008. "Gas dynamic virtual nozzle for generation of microscopic droplet streams." *Journal of Physics D: Applied Physics* 41 (19): 195505.

Diederichs, Kay. 2017. "Dissecting random and systematic differences between noisy composite data sets." *Acta Crystallographica Section D: Structural Biology* 73 (4): 286–293.

Diederichs, Kay, and P Andrew Karplus. 2013. "Better models by discarding data?" *Acta Crystallographica Section D: Biological Crystallography* 69 (7): 1215–1222.

Duisenberg, Albert JM. 1992. "Indexing in single-crystal diffractometry with an obstinate list of reflections." *Journal of applied crystallography* 25 (2): 92–96.

Emma, Paul, R Akre, J Arthur, R Bionta, C Bostedt, J Bozek, A Brachmann, P Bucksbaum, Ryan Coffee, F-J Decker, et al. 2010. "First lasing and operation of an ångstrom-wavelength free-electron laser." *nature photonics* 4 (9): 641.

Emsley, Paul, and Kevin Cowtan. 2004. "Coot: model-building tools for molecular graphics." *Acta Crystallographica Section D: Biological Crystallography* 60 (12): 2126–2132.

Fangohr, Hans, Cyril Danilevski, Jolanta Sztuk-Dambietz, Gabriele Giovanetti, Martin Teichmann, Sandor Brockhauser, Leonce Mekinda, Thomas Michelat, Wajid Ehsan, Gero Flucke, et al. 2018. "Data analysis support in Karabo at European XFEL."

Fienup, James R. 1982. "Phase retrieval algorithms: a comparison." *Applied optics* 21 (15): 2758–2769.

Foadi, James, Pierre Aller, Yilmaz Alguel, Alex Cameron, Danny Axford, Robin L Owen, Wes Armour, David G Waterman, So Iwata, and Gwyndaf Evans. 2013. "Clustering procedures for the optimal selection of data sets from multiple crystals in macromolecular crystallography." *Acta Crystallographica Section D: Biological Crystallography* 69 (8): 1617–1632.

Fromme, Petra, and Ingo Grotjohann. 2008. "Overview of photosynthesis."

Ginn, Helen Mary, and David Ian Stuart. 2017. "The slip-and-slide algorithm: a refinement protocol for detector geometry." *Journal of synchrotron radiation* 24 (6): 1152–1162.

Grünbein, Marie Luise, and Gabriela Nass Kovacs. 2019. "Sample delivery for serial crystallography at free-electron lasers and synchrotrons." *Acta Crystallographica Section D: Structural Biology* 75 (2).

Guo, Gongrui, Martin R Fuchs, Wuxian Shi, John Skinner, Evanna Berman, Craig M Ogata, Wayne A Hendrickson, Sean McSweeney, and Qun Liu. 2018. "Sample manipulation and data assembly for robust microcrystal synchrotron crystallography." *IUCrJ* 5 (3): 238–246.

Hart, Philip, Sébastien Boutet, Gabriella Carini, Mikhail Dubrovin, Brian Duda, David Fritz, Gunther Haller, Ryan Herbst, Sven Herrmann, Chris Kenney, et al. 2012. "The CSPAD megapixel x-ray camera at LCLS." In *X-Ray free-electron lasers: beam diagnostics, beamline instrumentation, and applications,* 8504:85040C. International Society for Optics and Photonics.

Hattne, Johan, Nathaniel Echols, Rosalie Tran, Jan Kern, Richard J Gildea, Aaron S Brewster, Roberto Alonso-Mori, Carina Glöckner, Julia Hellmich, Hartawan Laksmono, et al. 2014. "Accurate macromolecular structures using minimal measurements from X-ray free-electron lasers." *Nature methods* 11 (5): 545.

Henrich, B, J Becker, R Dinapoli, P Goettlicher, H Graafsma, H Hirsemann, R Klanner, H Krueger, R Mazzocco, A Mozzanica, et al. 2011. "The adaptive gain integrating pixel detector AGIPD a detector for the European XFEL." *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 633:S11–S14.

Henrich, B, A Bergamaschi, C Broennimann, R Dinapoli, EF Eikenberry, I Johnson, M Kobas, P Kraft, A Mozzanica, and B Schmitt. 2009. "PILATUS: A single photon counting pixel detector for X-ray applications." *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 607 (1): 247–249.

Holton, James M, and Kenneth A Frankel. 2010. "The minimum crystal size needed for a complete diffraction data set." *Acta Crystallographica Section D: Biological Crystallography* 66 (4): 393–408.

Huang, Zhirong, and Ingolf Lindau. 2012. "Free-electron lasers: SACLA hard-X-ray compact FEL." *Nature Photonics* 6 (8): 505.

Jeffrey, Phil. 2006. *X-ray Data Collection Course,* February. http://xray0.princeton.edu/~phil/Facility/Guides/XrayDataCollection.html.

Jordan, Patrick, Petra Fromme, Horst Tobias Witt, Olaf Klukas, Wolfram Saenger, and Norbert Krauß. 2001. "Three-dimensional structure of cyanobacterial photosystem I at 2.5 Å resolution." *Nature* 411 (6840): 909.

Kabsch, Wolfgang. 2010. "Xds." *Acta Crystallographica Section D: Biological Crystallography* 66 (2): 125–132.

———. 2014. "Processing of X-ray snapshots from crystals in random orientations." *Acta Crystallographica Section D: Biological Crystallography* 70 (8): 2204–2216.

Karplus, P Andrew, and Kay Diederichs. 2012. "Linking crystallographic model and data quality." *Science* 336 (6084): 1030–1033.

———. 2015. "Assessing and maximizing data quality in macromolecular crystallography." *Current opinion in structural biology* 34:60–68.

Kern, Jan, Ruchira Chatterjee, Iris D Young, Franklin D Fuller, Louise Lassalle, Mohamed Ibrahim, Sheraz Gul, Thomas Fransson, Aaron S Brewster, Roberto Alonso-Mori, et al. 2018. "Structures of the intermediates of Kok?s photosynthetic water oxidation clock." *Nature* 563 (7731): 421.

Kirian, Richard, and Nadia Zatsepin. 2015. *Cheetah Documentation - all keywords,* July. https://www.bioxfel.org/resources/cheetah_documentation_keywords/about.

Ko, In, Heung-Sik Kang, Hoon Heo, Changbum Kim, Gyujin Kim, Chang-Ki Min, Haeryong Yang, Soung Baek, Hyo-Jin Choi, Geonyeong Mun, et al. 2017. "Construction and commissioning of PAL-XFEL facility." *Applied Sciences* 7 (5): 479.

Krejcik, Patrick, FJ Decker, Y Ding, J Frisch, Z Huang, J Lewandowski, H Loos, J Turner, MH Wang, J Wang, et al. 2013. "Commissioning the new LCLS X-band transverse deflecting cavity with femtosecond resolution." *Proceedings of IBIC2013, Oxford, UK.*

Krewald, Vera, Marius Retegan, Nicholas Cox, Johannes Messinger, Wolfgang Lubitz, Serena DeBeer, Frank Neese, and Dimitrios A Pantazis. 2015. "Metal oxidation states in biological water splitting." *Chemical science* 6 (3): 1676–1695.

Kupitz, Christopher, Shibom Basu, Ingo Grotjohann, Raimund Fromme, Nadia A Zatsepin, Kimberly N Rendek, Mark S Hunter, Robert L Shoeman, Thomas A White, Dingjie Wang, et al. 2014. "Serial time-resolved crystallography of photosystem II using a femtosecond X-ray laser." *Nature* 513 (7517): 261.

Laskowski, RA, and GJ Swaminathan. 2013. "Problems of protein three-dimensional structures."

Leslie, Andrew GW. 2006. "The integration of macromolecular diffraction data." *Acta Crystallographica Section D: Biological Crystallography* 62 (1): 48–57.

Li, Chufeng, Xuanxuan Li, Richard Kirian, John CH Spence, Haiguang Liu, and Nadia A Zatsepin. 2019. "SPIND: a reference-based auto-indexing algorithm for sparse serial crystallography data." *IUCrJ* 6 (1).

Liu, Haiguang, and John CH Spence. 2014. "The indexing ambiguity in serial femtosecond crystallography (SFX) resolved using an expectation maximization algorithm." *IUCrJ* 1 (6): 393–401.

Liu, Qun, Tassadite Dahmane, Zhen Zhang, Zahra Assur, Julia Brasch, Lawrence Shapiro, Filippo Mancia, and Wayne A Hendrickson. 2012. "Structures from anomalous diffraction of native biological macromolecules." *Science* 336 (6084): 1033–1037.

Liu, Qun, Youzhong Guo, Yanqi Chang, Zheng Cai, Zahra Assur, Filippo Mancia, Mark I Greene, and Wayne A Hendrickson. 2014. "Multi-crystal native SAD analysis at 6 keV." *Acta Crystallographica Section D: Biological Crystallography* 70 (10): 2544–2557.

Liu, Qun, Qinglian Liu, and Wayne A Hendrickson. 2013. "Robust structural analysis of native biological macromolecules from multi-crystal anomalous diffraction data." *Acta Crystallographica Section D* 69 (7): 1314–1332.

Liu, Qun, Zhen Zhang, and Wayne A Hendrickson. 2011. "Multi-crystal anomalous diffraction for low-resolution macromolecular phasing." *Acta Crystallographica Section D: Biological Crystallography* 67 (1): 45–59.

Lyubimov, Artem Y, Monarin Uervirojnangkoorn, Oliver B Zeldin, Aaron S Brewster, Thomas D Murray, Nicholas K Sauter, James M Berger, William I Weis, and Axel T Brunger. 2016. "IOTA: integration optimization, triage and analysis tool for the processing of XFEL diffraction images." *Journal of applied crystallography* 49 (3): 1057–1064.

Lyubimov, Artem Y, Monarin Uervirojnangkoorn, Oliver B Zeldin, Qiangjun Zhou, Minglei Zhao, Aaron S Brewster, Tara Michels-Clark, James M Holton, Nicholas K Sauter, William I Weis, et al. 2016. "Advances in X-ray free electron laser (XFEL) diffraction data processing applied to the crystal structure of the synaptotagmin-1/SNARE complex." *Elife* 5:e18740.

Mancuso, Adrian. *Overview of the European XFEL and the SPB/SFX Instrumet: Opportunities for microfluidic sample delivery (and more).* https://www.xfel.eu/

sites/sites_custom/site_xfel/content/e35152/e35161/e59605/e59607/e59608/e59616/xfel_file59619/Mancuso_XFEL_and_SPB-SFX_for_Microfluidics_Workshop_eng.pdf. Accessed: 2019-05-06.

Mariani, Valerio, Andrew Morgan, Chun Hong Yoon, Thomas J Lane, Thomas A White, Christopher O'Grady, Manuela Kuhn, Steve Aplin, Jason Koglin, Anton Barty, et al. 2016. "OnDA: online data analysis and feedback for serial X-ray imaging." *Journal of applied crystallography* 49 (3): 1073–1080.

Martin-Garcia, Jose M, Chelsie E Conrad, Garrett Nelson, Natasha Stander, Nadia A Zatsepin, James Zook, Lan Zhu, James Geiger, Eugene Chun, David Kissick, et al. 2017. "Serial millisecond crystallography of membrane and soluble protein microcrystals using synchrotron radiation." *IUCrJ* 4 (4): 439–454.

Milne, Christopher, Thomas Schietinger, Masamitsu Aiba, Arturo Alarcon, Jürgen Alex, Alexander Anghel, Vladimir Arsov, Carl Beard, Paul Beaud, Simona Bettoni, et al. 2017. "SwissFEL: the Swiss X-ray free electron laser." *Applied Sciences* 7 (7): 720.

Milton, SV, E Gluskin, ND Arnold, C Benson, W Berg, SG Biedron, M Borland, Y-C Chae, RJ Dejus, PK Den Hartog, et al. 2001. "Exponential gain and saturation of a self-amplified spontaneous emission free-electron laser." *Science* 292 (5524): 2037–2041.

Neutze, Richard, Remco Wouts, David van der Spoel, Edgar Weckert, and Janos Hajdu. 2000. "Potential for biomolecular imaging with femtosecond X-ray pulses." *Nature* 406 (6797): 752.

Oberthuer, Dominik, Juraj Knoška, Max O Wiedorn, Kenneth R Beyerlein, David A Bushnell, Elena G Kovaleva, Michael Heymann, Lars Gumprecht, Richard A Kirian, Anton Barty, et al. 2017. "Double-flow focused liquid injector for efficient serial femtosecond crystallography." *Scientific reports* 7:44628.

Padilla, Jennifer E, and Todd O Yeates. 2003. "A statistic for local intensity differences: robustness to anisotropy and pseudo-centering and utility for detecting twinning." *Acta Crystallographica Section D: Biological Crystallography* 59 (7): 1124–1130.

Peck, Ariana, Frédéric Poitevin, and Thomas J Lane. 2018. "Intermolecular correlations are necessary to explain diffuse scattering from protein crystals." *IUCrJ* 5 (2): 211–222.

Pettersen, Eric F, Thomas D Goddard, Conrad C Huang, Gregory S Couch, Daniel M Greenblatt, Elaine C Meng, and Thomas E Ferrin. 2004. "UCSF Chimera?a visu-

alization system for exploratory research and analysis." *Journal of computational chemistry* 25 (13): 1605–1612.

Roedig, Philip, Helen M Ginn, Tim Pakendorf, Geoff Sutton, Karl Harlos, Thomas S Walter, Jan Meyer, Pontus Fischer, Ramona Duman, Ismo Vartiainen, et al. 2017. "High-speed fixed-target serial virus crystallography." *nAture methods* 14 (8): 805.

Rupp, Bernhard. 2009. *Biomolecular crystallography: principles, practice, and application to structural biology.* Garland Science.

Shin, Hocheol, Seungnam Kim, and Chun Hong Yoon. 2018. "Data Analysis using Psocake at PAL-XFEL." *Journal of the Korean Physical Society* 73 (1): 16–20.

Stander, Natasha, Petra Fromme, and Nadia Zatsepin. Accepted. "DatView: a graphical user interface for visualizing and querying large datasets in serial femtosecond crystallography." *Journal of applied crystallography.*

Suga, Michihiro, Fusamichi Akita, Kunio Hirata, Go Ueno, Hironori Murakami, Yoshiki Nakajima, Tetsuya Shimizu, Keitaro Yamashita, Masaki Yamamoto, Hideo Ago, et al. 2015. "Native structure of photosystem II at 1.95 Å resolution viewed by femtosecond X-ray pulses." *Nature* 517 (7532): 99.

Suga, Michihiro, Fusamichi Akita, Michihiro Sugahara, Minoru Kubo, Yoshiki Nakajima, Takanori Nakane, Keitaro Yamashita, Yasufumi Umena, Makoto Nakabayashi, Takahiro Yamane, et al. 2017. "Light-induced structural changes and the site of O= O bond formation in PSII caught by XFEL." *Nature* 543 (7643): 131.

Tschentscher, Thomas, Christian Bressler, Jan Grünert, Anders Madsen, Adrian Mancuso, Michael Meyer, Andreas Scherz, Harald Sinn, and Ulf Zastrau. 2017. "Photon beam transport and scientific instruments at the European XFEL." *Applied Sciences* 7 (6): 592.

Turner, JL, R Akre, A Brachmann, F-J Decker, YT Ding, P Emma, Y Feng, A Fisher, J Frisch, A Gilevich, et al. 2016. "FEL beam stability in the LCLS." In *Conf. Proc. C110328: 2423-2425, 2011.* SLAC-PUB-16660. SLAC National Accelerator Lab., Menlo Park, CA (United States).

Umena, Yasufumi, Keisuke Kawakami, Jian-Ren Shen, and Nobuo Kamiya. 2011. "Crystal structure of oxygen-evolving photosystem II at a resolution of 1.9 Å." *Nature* 473 (7345): 55.

Victorgrigas. 2012. *Wikimedia Foundation Servers,* July 16, CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0). Accessed May 6, 2019. https://commons.wikimedia.org/wiki/File:Wikimedia_Foundation_Servers-8055_35.jpg.

Wall, Michael E, Alexander M Wolff, and James S Fraser. 2018. "Bringing diffuse X-ray scattering into focus." *Current opinion in structural biology* 50:109–116.

Wang, Jimin, Mikhail Askerka, Gary W Brudvig, and Victor S Batista. 2017. "Insights into photosystem II from isomorphous difference Fourier maps of femtosecond X-ray diffraction data and quantum mechanics/molecular mechanics structural models." *ACS energy letters* 2 (2): 397–407.

Weierstall, Uwe, Daniel James, Chong Wang, Thomas A White, Dingjie Wang, Wei Liu, John CH Spence, R Bruce Doak, Garrett Nelson, Petra Fromme, et al. 2014. "Lipidic cubic phase injector facilitates membrane protein serial femtosecond crystallography." *Nature communications* 5:3309.

White, Thomas A. 2019. "Processing serial crystallography data with CrystFEL: a step-by-step guide." *Acta Crystallographica Section D: Structural Biology* 75 (2).

White, Thomas A, Anton Barty, Francesco Stellato, James M Holton, Richard A Kirian, Nadia A Zatsepin, and Henry N Chapman. 2013. "Crystallographic data processing for free-electron laser sources." *Acta Crystallographica Section D: Biological Crystallography* 69 (7): 1231–1240.

White, Thomas A, Richard A Kirian, Andrew V Martin, Andrew Aquila, Karol Nass, Anton Barty, and Henry N Chapman. 2012. "CrystFEL: a software suite for snapshot serial crystallography." *Journal of applied crystallography* 45 (2): 335–341.

White, Thomas A, Valerio Mariani, Wolfgang Brehm, Oleksandr Yefanov, Anton Barty, Kenneth R Beyerlein, Fedor Chervinskii, Lorenzo Galli, Cornelius Gati, Takanori Nakane, et al. 2016. "Recent developments in CrystFEL." *Journal of applied crystallography* 49 (2): 680–689.

Winn, Martyn D, Charles C Ballard, Kevin D Cowtan, Eleanor J Dodson, Paul Emsley, Phil R Evans, Ronan M Keegan, Eugene B Krissinel, Andrew GW Leslie, Airlie McCoy, et al. 2011. "Overview of the CCP4 suite and current developments." *Acta Crystallographica Section D: Biological Crystallography* 67 (4): 235–242.

Yefanov, Oleksandr, Cornelius Gati, Gleb Bourenkov, Richard A Kirian, Thomas A White, John CH Spence, Henry N Chapman, and Anton Barty. 2014. "Mapping the

continuous reciprocal space intensity distribution of X-ray serial crystallography." *Phil. Trans. R. Soc. B* 369 (1647): 20130333.

Yefanov, Oleksandr, Valerio Mariani, Cornelius Gati, Thomas A White, Henry N Chapman, and Anton Barty. 2015. "Accurate determination of segmented X-ray detector geometry." *Optics express* 23 (22): 28459–28470.

Young, Iris D, Mohamed Ibrahim, Ruchira Chatterjee, Sheraz Gul, Franklin D Fuller, Sergey Koroidov, Aaron S Brewster, Rosalie Tran, Roberto Alonso-Mori, Thomas Kroll, et al. 2016. "Structure of photosystem II and substrate binding at room temperature." *Nature* 540 (7633): 453.

Zander, Ulrich, Michele Cianci, Nicolas Foos, Catarina S Silva, Luca Mazzei, Chloe Zubieta, Alejandro De Maria, and Max H Nanao. 2016. "Merging of synchrotron serial crystallographic data by a genetic algorithm." *Acta Crystallographica Section D: Structural Biology* 72 (9): 1026–1035.

Zeldin, Oliver B, Aaron S Brewster, Johan Hattne, Monarin Uervirojnangkoorn, Artem Y Lyubimov, Qiangjun Zhou, Minglei Zhao, William I Weis, Nicholas K Sauter, and Axel T Brunger. 2015. "Data Exploration Toolkit for serial diffraction experiments." *Acta Crystallographica Section D: Biological Crystallography* 71 (2): 352–356.

Zouni, Athina, Horst-Tobias Witt, Jan Kern, Petra Fromme, Norbert Krauss, Wolfram Saenger, and Peter Orth. 2001. "Crystal structure of photosystem II from Synechococcus elongatus at 3.8 Å resolution." *Nature* 409 (6821): 739.

APPENDIX A

DATASETS

This appendix summarizes work done by dataset. Analysts of the datasets are referred to the experimental log books, google spreadsheets, and other spreadsheets available with the data at Arizona State University (ASU) for more comprehensive statistics and information.

A.1 Photosystem II comparisons

The PSII all data comparison was an attempt to compare the many PSII datasets. Indexing was therefore identical for all datasets with the exception that the geometry file with each dataset was used. The indexing command was:

```
indexamajig −i \$INLST \
    −o \$STREAM −j 4\
    −g ${GEOM} \
    −−peaks=hdf5 −−int−radius=2,4,5 \
    −−indexing=mosflm−latt−nocell,mosflm−nolatt−cell,mosflm,dirax \
    −p ${CELL} −−tolerance=10,10,10,1 \
    −−integration=rings −−multi
```

Indexing was performed with *CrystFEL* version 0.7.0. Note that because the same indexing command was used for all datasets, indexing is not optimal for any of the datasets and better statistics may come from more optimized indexing. The cell file for all datasets was:

```
CrystFEL unit cell file version 1.0

lattice_type = orthorhombic
centering = P

a = 133 A
b = 228 A
c = 308 A

al = 90.00 deg
be = 90.00 deg
ga = 90.00 deg
```

and a large tolerance was used to allow for the variation in the chip datasets. The all PSII dataset is used to generate tables 1 and 2, and figures 3, 31, 32, 33, 34, and 45. For LCLS January 2012 both 'a' and 'b' hit tags were indexed since most runs

were only one or the other but there are about 8,000 duplicated patterns. For LCLS June 2012 only 'a' hits were indexed since it covered all runs and usually had more patterns. The laser scheme for LCLS November 2014 is an assumption based on the electronic log book.

A.2 LCLS January 2012

LCLS January 2012 was the primary dataset used in (Kupitz et al. 2014). Data was reindexed with *CrystFEL* version 0.6.2 and geometry optimization done (see fig. 38). The changes in *CrystFEL* and geometry file resulted in an improved indexing rate of around 60% compared to the prior rate of around 40% estimated from the stream files available at ASU. Data was reindexed again as part of the all PSII dataset with *CrystFEL* 0.7.0.

A.3 LCLS June 2012

LCLS June 2012 was part of (Kupitz et al. 2014) for unit cell comparisons. It targets the transient S4 state instead of the S3 state and prior results at ASU had about 8,000 light and 8,000 dark patterns corresponding a 40% indexing rate. Reindexing with *CrystFEL* version 0.6.2 increased the indexing rate to around 90% with around 20,000 light and 20,000 dark patterns. This dataset was used for indexing optimization and results from it are presented throughout this dissertation. The indexing screen is given in section 3.3 with additional results in section 4.3.1. Merging results are in section 5.1.1, and phasing and refinement results in section 6.2. The dataset was also reindexed as part of the all PSII dataset with *CrystFEL* 0.7.0.

A.4 LCLS November 2014

LCLS November 2014 was a screening beam time for an agarose jet. There are a few hundred agarose patterns, but the majority of the dataset comes from liquid jet. From the electronic log book, it appears that data was time resolved with a single laser flash at 500 μs. This dataset was used in (Ayyer et al. 2016), but that analysis was done by CFEL and is not fully stored at ASU. This dataset was reindexed as part of the all PSII dataset with *CrystFEL* 0.7.0.

A.5 LCLS October 2015

LCLS October 2015 is the first beam time with both onsite and offsite contributions to analysis. Onsite work was primarily training. Offsite analysis focused on continuous diffuse scattering and forms the bulk of chapter 7. The dataset can be viewed as a screening dataset with many variations of laser scheme, pulse duration, and quinone addition. The major challenge with this dataset is that with so many conditions it is hard to get enough patterns from any one condition to make meaningful comparisons.

A.6 APS August 2016

Cheetah-cbf (see section 2.2) was developed for this beam time and used onsite. However, with only a few computers instead of a computational cluster, *cheetah-cbf* was unable to keep up with data collection. All CBF files were copied to ASU, and *cheetah-cbf* was run at ASU (see section 2.1.2). The results were used for further analysis by Jose Martin and published in (Martin-Garcia et al. 2017).

A.7 LCLS August 2016

LCLS August 2016 was a combined diffraction and spectroscopy experiment examining the most promising LCLS October 2016 condition with a laser scheme of '700 ; 1300 ; 600' and a pulse duration of 15 femtosecond. The XTCAV system was used to record the shot-to-shot pulse durations. Analysis of this dataset was delayed since it was so close to other beam times, and LCLS November 2016 took priority. Work done on this dataset is presented in chapter 7 and was primarily early continuous diffuse scattering analysis and background corrections. A challenge with this dataset is that many later runs had a nozzle shadow that wasn't detected until much later. Also, the shorter pulse duration meant less intensity so the patterns may not be as strong in comparison to other datasets.

A.8 LCLS November 2016

LCLS November 2016 was a CFEL fixed-target chip proof-of-principle experiment that actually included many proteins. Only PSII runs were copied to ASU. The primary challenge with LCLS November 2016 is the oscillating unit cells. This dataset is the basis for the subset analysis used in sections 4.3.2, 4.3.3, 5.1.2, and 6.3.

A.9 LCLS September 2017

LCLS September 2017 was a proof-of-principle for time-resolved fixed-target chip datasets. The major challenge with the dataset is fogging means the identification of light and dark data is uncertain. Analysis from this dataset is included in section 1.4.2, and work included quality control of the hit finding, indexing, and detailed notes by run and by groups such as sample and chips. Note that a major change in *Cheetah* means that output was split into CXI and H5 files and the H5 files in this experiment are missing information for the last event due to a *Cheetah* error.

A.10 European XFEL November 2017

EXFEL November 2017 was a single-flash time-resolved study of PSI. The dark data has been prepared for a publication that is currently being reviewed. The major contribution with this experiment was onsite logging.

A.11 LCLS March 2018

LCLS March 2018 was a fixed-target PSII experiment with the addition of a tape-drive to clear debris and prevent fogging. However, issues with the beam limited data collection and the strong background from the chip makes the dataset difficult to use for continuous diffuse scattering analysis. *Cheetah-parallel* was developed at this beam time (see section 2.3).

A.12 European XFEL August 2018

EXFEL August 2018 was a follow up to EXFEL November 2017 focused on time resolved PSI data collection. Onsite work adapted *cheetah-parallel* for use with the European XFEL (see section 2.3). Offsite analysis included rerunning dark calibrations and hit finding for all runs and the indexing optimizations presented in chapter 4.

A.13 PAL XFEL November 2018

PAL XFEL November 2018 was a spectroscopy experiment led by Ganesh Subramanian. Onsite work adapted *DatView* for spectroscopy data (Stander, Fromme, and Zatsepin, Accepted).

A.14 LCLS December 2018

LCLS December 2018 was primarily a spectroscopy experiment, although one shift collected simultaneous diffraction data for PSII. Hit finding was done onsite and copied to ASU, and initial indexing was done as part of the all PSII dataset with *CrystFEL* 0.7.0.

A.15 European XFEL March 2019

EXFEL March 2019 was initially intended as a follow up for EXFEL August 2018 to get more time-resolved patterns. However, beam difficulties meant that one shift was lost, and the remainder of the beam time was spent screening several conditions and testing jets. Onsite analysis included hit finding and indexing. Offsite analysis has not been a priority since, like LCLS October 2015, the beam time is more of a screening beam time and there aren't a lot of patterns in any single condition.

APPENDIX B

TEXT BASED PDB COMPARISON

A text based comparison of MR.1.pdb and PDB entry 3wu2 showing the difference between the residue, chain, and ID fields. The alignment in the table was done manually and does not necessarily correspond to structural alignment. Residues not found in the CIF file provided with MR.1.pdb are bolded. That CIF file contained LHG, PHO, DGD, LMT, CLA, PL9, BCT, BCR, and SQD.

Table 27. MR.1.pdb text compared with 3wu2

MR1	3wu2	MR1	3wu2	MR1	3wu2
SQD a 659	SQD a 401		**GOL B 634**		**GOL D 415**
	LMT a 402		**GOL B 635**	THR e 4	
OEX a 601	OEX a 404		**GOL B 636**	THR e 5	
FE2 a 603	FE2 a 405		**GOL B 637**	LHG e 772	
CL a 679	CL a 406		**GOL B 638**	LMT e 787	
CL a 680	CL a 407	CA c 901	CA c 901	LHG E 772	LHG E 101
	BCT a 408	CLA c 628	CLA c 902	LMT E 787	
CLA a 604	CLA a 409	CLA c 629	CLA c 903	HEM f 641	HEM f 101
CLA a 606	CLA a 410	CLA c 630	CLA c 904		SQD f 102
CLA a 607	CLA a 411	CLA c 631	CLA c 905	CA f 796	CA f 103
PHO a 608	PHO a 412	CLA c 632	CLA c 906		**GOL f 104**
	PHO a 413	CLA c 633	CLA c 907	HEM F 641	HEM F 101
CLA a 610	CLA a 414	CLA c 634	CLA c 908		LMT F 102
BCR a 645	BCR a 415	CLA c 635	CLA c 909	CA F 796	CA F 103
SQD a 667	SQD a 416	CLA c 636	CLA c 910	LEU h 65	
	LHG a 417	CLA c 637	CLA c 911	GLY h 66	
	LMG a 418	CLA c 638	CLA c 912	BCR h 650	
PL9 a 713	PL9 a 419	CLA c 639	CLA c 913	DGD h 663	DGD h 102
	GOL a 422	CLA c 640	CLA c 914		**GOL h 103**
	GOL a 423	BCR c 655	BCR c 915	LEU H 65	
	GOL a 424		BCR c 916	GLY H 66	
OEX A 601	OEX A 401	DGD c 657	DGD c 917	BCR H 650	
FE2 A 603	FE2 A 402	DGD c 660	DGD c 918	DGD H 663	DGD H 102
CL A 679	CL A 403	DGD c 661	DGD c 919	MET i 1	FME i 1
CL A 680	CL A 404		**LMG c 920**	MET I 1	FME I 1
CLA A 604	CLA A 405		**LMG c 921**	LEU I 37	
CLA A 607	CLA A 406	LMT c 758	LMT c 922	GLU I 38	
	CLA A 407		**GOL c 927**	MET j 1	
PHO A 608	PHO A 408		**GOL c 928**	BCR j 652	
PHO A 609	PHO A 409		**GOL c 929**	MG j 771	MG j 101
CLA A 610	CLA A 410		**GOL c 930**	SER J 3	
BCR A 645	BCR A 411	CLA C 628	CLA C 501	GLU J 4	
SQD A 659	SQD A 412	CLA C 629	CLA C 502		MG J 101
	LMG A 413	CLA C 630	CLA C 503		LMT J 102
PL9 A 713	PL9 A 414	CLA C 631	CLA C 504	BCR k 653	BCR k 101

Continued on next page

Table 27 – continued from previous page

MR1	3wu2	MR1	3wu2	MR1	3wu2
SQD A 667	SQD A 418	CLA C 632	CLA C 505	BCR k 654	BCR k 102
LMT A 730	LMT A 419	CLA C 633	CLA C 506	BCR K 652	BCR K 101
	GOL A 421	CLA C 634	CLA C 507	BCR K 653	BCR K 102
	GOL A 422	CLA C 635	CLA C 508	LHG l 694	LHG l 101
	GOL A 423	CLA C 636	CLA C 509		**GOL l 102**
BCT A 681		CLA C 637	CLA C 510	LHG L 694	LHG L 101
GLU b 485		CLA C 638	CLA C 511	SQD L 668	SQD L 103
LEU b 486		CLA C 639	CLA C 512		**GOL L 104**
ARG b 505		CLA C 640	CLA C 513	MET m 1	FME m 1
CA b 803	CA b 603	BCR C 654	BCR C 514	LMT m 742	LMT m 101
CLA b 620	CLA b 604	BCR C 655	BCR C 515	LMT m 759	LMT m 102
CLA b 621	CLA b 605	DGD C 657	DGD C 516	MET M 1	FME M 1
CLA b 622	CLA b 606	DGD C 660	DGD C 517	LYS M 34	
CLA b 623	CLA b 607	DGD C 661	DGD C 518	LMT M 742	LMT M 101
CLA b 624	CLA b 608		**LMG C 519**	LMT M 759	LMT M 102
CLA b 625	CLA b 609	LMT C 758	LMT C 520	ARG o 60	
CLA b 626	CLA b 610		**GOL C 524**	GLN o 61	
CLA b 627	CLA b 611		**GOL C 525**	CA o 767	CA o 301
BCR b 646	BCR b 620		**GOL C 526**		GLN O 3
BCR b 648	BCR b 621	HIS d 336		CA O 767	CA O 301
BCR b 649	BCR b 622		HSK d 336		SO4 O 302
	LMG b 623	CLA d 605	CLA d 402		GOL O 304
LMT b 791	LMT b 624	PHO d 609		MET t 1	FME t 1
	LMT b 625	CLA d 611	CLA d 403	BCR t 647	BCR t 101
	GOL b 632	BCR d 651	BCR d 404		LMT t 102
	GOL b 633	LHG d 664	LHG d 407	MET T 1	FME T 1
	GOL b 634	BCT d 681		BCR T 647	BCR T 101
	GOL b 635	LHG d 702	LHG d 408	CL U 808	
	GOL b 636	PL9 d 712	PL9 d 405	HEM v 642	HEM v 201
CA B 803	CA B 601	LHG d 714	LHG d 409	CL v 808	
CLA B 618	CLA B 602	DGD d 755	DGD d 406		GOL v 202
CLA B 619	CLA B 603		**LMG d 410**		GOL v 203
CLA B 620	CLA B 604	SQD d 768			GOL v 204
CLA B 621	CLA B 605	GLU D 11		HEM V 642	HEM V 201
CLA B 622	CLA B 606	HIS D 336			GOL V 203
CLA B 623	CLA B 607		HSK D 336		GOL V 204
CLA B 624	CLA B 608		BCT D 401		GOL V 205
CLA B 625	CLA B 609	CLA D 605	CLA D 402	SER x 40	
CLA B 626	CLA B 610	CLA D 606	CLA D 403	SER X 40	
CLA B 627	CLA B 611	CLA D 611	BCR D 404	VAL y 18	
BCR B 646	BCR B 618	BCR D 651		VAL Y 18	

Continued on next page

Table 27 – continued from previous page

MR1	3wu2	MR1	3wu2	MR1	3wu2
BCR B 648	BCR B 619	PL9 D 712	PL9 D 405	ILE Y 19	
BCR B 649	BCR B 620	DGD D 755	DGD D 406	MET z 1	
SQD B 668	SQD B 621	SQD D 768	SQD D 407	VAL z 62	
	LMG B 622	LHG D 664	LHG D 408		LMT z 101
LMT B 730	LMT B 623	LHG D 702	LHG D 409		LMG Z 101
LMT B 791		LHG D 714	LHG D 410		LMT Z 102
	GOL B 633		**LMG D 411**		

APPENDIX C

COMBINING POWDERS

The following describes how powders are combined in *Cheetah* parallel (see section 2.3).

From libcheetah/powder.cpp

```
[line 316] powderBuffer[i] /= powder_counter[i];
```

Then the following fields are averages from their corresponding powderBuffer sum arrays.

- data/non_assembled_detector_and_photon_corrected
- data/non_assembled_detector_corrected
- data/radial_average_detector_and_photon_corrected
- data/radial_average_detector_corrected

So, combining them across multiple files is roughly the weighted average (recognizing the 'data/nframes' may not be equivalent to powder_counter[i] for every pixel):

```
for filetocombine
    sum+= file['data/nframes'][0]*np.array(
          file['data/one_of_the_average_fields'])
    N += file['data/nframes'][0]
final=sum / N
```

The fields ending in _sigma seem to come from libcheetah/powder.cpp line 339:

```
powderSigmaBuffer[i] = sqrt(powderSquaredBuffer[i]/powder_counter[i] -
            powderBuffer[i]*powderBuffer[i]);
```

So to combine sigmas, powderSquaredBuffer needs to be reconstructed. Algebraically, given powderSigmaBuffer[i], powder_counter[i] roughly equal to nframes, and the corresponding average, then

```
powdersquaredbuffer= n * (powderSigmaBuffer * powderSigmaBuffer
        + average * average)
```

And since powder squared buffer is itself just a sum from libcheetah/powder.cpp lines 82 and 96:

```
buffer[i] = ((double) data[i])*data[i];

powder\_squared[i] += buffer[i];
```

Then to combine multiple files sigma fields the complete sqrsum and sum fields must be reconstructed:

```
for filetocombine
    sum+= file['data/nframes'][0]*np.array(
                file['data/one_of_the_average_fields'])
    N += file['data/nframes'][0]
   sqrsum +=file['data/nframes'][0] * (powderSigmaBuffer *
            powderSigmaBuffer +
            average * average)

finalsum=sum / N
finalsigma=np.sqrt(sqrsum/N - finalsum*finalsum)
```

The field 'data/peakpowder' seems to be a raw sum from libcheetah/powder.cpp 371-384 since it is the value of detector->powderPeaks[powderClass] which is only updated in addToPowder function (line 150 in powder.cpp). So both that field and nframes is just the sum across files.