Journal of Applied Crystallography

ISSN 0021-8898

The Expand-Maximize-Compress Single Particle Imaging Algorithm

Kartik Ayyer,[a] Ti-Yen Lan,[b] N. Duane Loh[c]* and Veit Elser[b]

[a]Center for Free-Electron Laser Science, Deutsches Elektronen-Synchrotron DESY, Notkestraße 85, 22607 Hamburg, Germany, [b]Second affiliation address, and [c]Centre for Bio-imaging Sciences, National University of Singapore, 117543, Singapore.. Correspondence e-mail: duaneloh@nus.edu.sg

TODO: Abstract goes here.

1. Introduction

TODO: Some mumbo jumbo about how SPI will revolutionize the world.

The goal of single-particle imaging (SPI) is to assemble the three-dimensional (3D) structure of an object from incomplete data collected from very many of its copies. With X-ray Free-electron Laser (XFEL) based SPI, noisy and un-oriented two-dimensional (2D) diffraction patterns of many individual copies of the object are computationally assembled into a 3D diffraction volume. The object's 3D electron density distribution is then recovered from this diffraction volume via de novo phase-retrieval.

TODO: Include references that discusses XFEL SPI.

1.1. Purpose of this software package

What is EMC? Reference 1D and 3D EMC paper. Expectation maximization in the space of unknown orientations? Features are shot-noise-limited diffraction patterns. How does EMC compare with other methods? Extensions of this algorithm have been successfully adapted for tomography and crystallography [...]. Describe what is different about these applications.

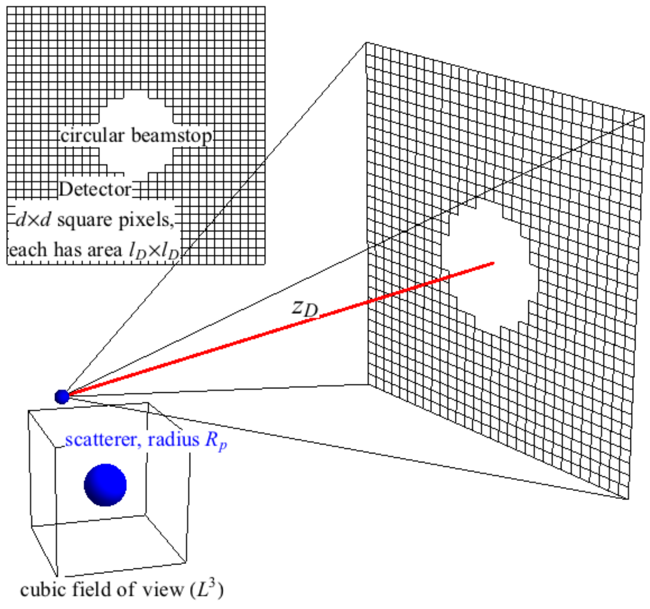
circular beamstop
Detector
$d\times d$ square pixels,
each has area $l_D\times l_D$
z_D
scatterer, radius R_p
cubic field of view (L^3)

Figure 1
Experimental geometry of single-particle imaging adopted in the data stream simulator. Notably, we implement a square detector comprising square pixels. The scatterer, while depicted here as a sphere of radius R_p, is typically an electron density map sampled from a Protein Data Bank file.

2. Structure of this package

This software package uses the EMC algorithm to reconstruct a 3D diffraction volume from noisy, randomly-oriented single particle diffraction patterns. These patterns could be from simulations or actual single-particle experiments, where the minimum input are: a configuration file, a file with detector coordinates and pixel status, and a sparse representation of the photon data from diffraction patterns flowchart. Here's an example of the configuration file that simulates a single-particle imaging data stream and reconstructs the 3D intensities afterwards:

2.1. Typical experiment and key parameters

TODO: Briefly describe the key parameters, which will be later echoed in config.ini. Also state choice of crystallographic spatial frequency, discuss dimensionless radius, speckle sampling, solid angle correction.

2.2. Data stream simulator

TODO: Short overview of data stream simulator: why this is convenient and any key design choices we made.

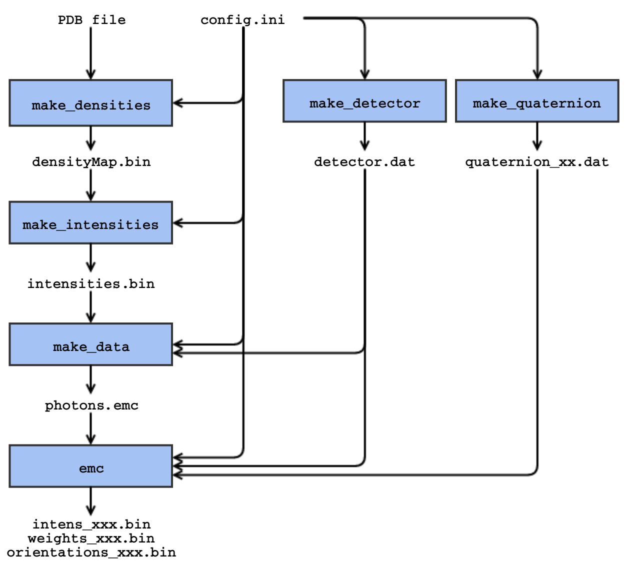

Figure 2
Flowchart of how to simulate a data set and perform a reconstruction starting from a sample PDB file and a configuration file with information about the experimental setup.

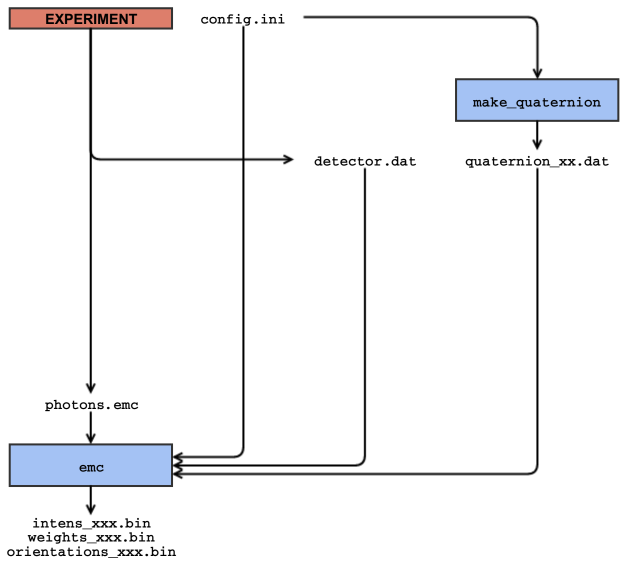

Figure 3
Flowchart of how to process experimental data in sparse format. Information about the experimental parameters is placed in the configuration file and the detector geometry is in the detector file. The formats of all three input files is described in Section 2.5.

2.3. Implementing the EMC algorithm

The EMC algorithm (Loh & Elser, 2009) is implemented here with hybrid MPI+OpenMP, and hence suitable for shared and/or distributed memory systems. In this section, we describe this implementation and an extension to deal with high signal data.

In the current version, the code assumes a Poisson probability model for the number of photons in a pixel. This means that if the mean number of photons at a pixel is W, the probability of getting K photons is given by

$$P(K|W) = \frac{W^K e^{-W}}{K!} \tag{1}$$

Gaussian noise models have been used in situations with bright, but noisy data (Loh *et al.*, 2010; Ekeberg *et al.*, 2015), but if single photons can be accurately counted, the noise model will be Poissonian. Let the number of photons at pixel number t in data frame d be K_{dt}. Also let, for a given orientation r, the predicted mean intensity at the same pixel be W_{rt}. Since an independent Poisson process occurs at each pixel, the probability of that frame being generated by that particular tomogram is

$$R_{dr} = \prod_t \frac{W_{rt}^{K_{dt}} e^{-W_{rt}}}{K_{dt}!} \tag{2}$$

But since the particle must have some orientation, the probability of frame d having orientation r is obtained by normalizing over all orientations.

$$P_{dr} = \frac{R_{dr}}{\sum\limits_r R_{dr}} \tag{3}$$

Using the principle of expectation maximization, one can calculate the update rule on the tomograms W_{rt} given the probability distributions P_{dr} as

$$W'_{rt} = \frac{\sum\limits_d P_{dr} K_{dt}}{\sum\limits_d P_{dr}} \tag{4}$$

The most time-consuming step of each iteration is the calculation of Eq. 2. This involves comparing all the tomograms with all the data frames for each pixel which has at least one photon. The code is parallelized over orientations, so each MPI and OpenMP rank performs the calculation for a subset of orientations. At the start of the iterations, each MPI rank gets a copy of the current 3D intensity model. Each MPI and OpenMP rank then calculate the relevant tomograms, W_{rt}, on the fly and calculate R_{dr} for that orientation using Eq. 2. In order to perform the normalization operation of Eq. 3, reduction operations across all ranks. The P_{dr} array is then used to calculate updated tomograms for each r, and then merged to obtain an updated 3D model for each MPI rank. These models are then reduced to obtain the final updated model.

In many experimental situations, the incident fluence varies from shot-to-shot. Thus, the tomograms would be scaled differently for each data frame. One can enable the recovery of these scale factors using the update rule described in Appendix C.

We also find that if the signal is too strong, the reconstructions get stuck in local maxima by having all the orientations overlap each other. This effect is similar to what is observed if the background is too high (Ayyer *et al.*, 2015). However, we also observe that the reconstruction is stable around the true solution, and only falters when one starts from a random initial guess. This problem can be avoided by using the deterministic annealing variant of expectation maximization (Ueda &

Nakano, 1998). In the EMC case, this is implemented by raising R_{dr} calculated in Eq. 2 to a small power β, and then normalizing. This has the effect of broadening the distribution and results in a rotationally blurred, but stable reconstruction. The power β can then be raised gradually in a manner similar to simulated annealing to slowly improve the reconstruction. An example of this is shown in Section 3.1, where β starts at 0.001 and is raised by a factor of 2 every 10 iterations 8 times.

2.4. Modules and convenience utilities

2.4.1. Modules Here are the essential modules for simulating a data stream from a single-particle imaging experiment. By default, these modules uses parameters listed in a single `config.ini` configuration file, although point each module to different configuration files for each is straightforward as well.

1. **make_detector.py.** Creates a detector file using the experimental parameters specified in the configuration file. The format of this file is specified in Section 2.5.2.
2. **make_densities.py.** Creates an electron density map from a PDB file, given the resolution and field of view expected from the configuration file.
3. **make_intensities.py**. Creates a set of 3D diffraction intensities from an electron density map and the experimental parameters found in the configuration file.
4. **make_data**. Simulates a sparse photon diffraction pattern using a 3D diffraction volume (e.g. the one generated by **make_intensities.py**), and the configuration file. By default these photon data are saved as a binary file, `photons.emc`, detailed in Section 2.5.3.
5. **make_quaternion**. Creates a list of quasi-uniform rotation group samples based on a refinement scheme of the 600-cell, as described in Loh & Elser (2009). These samples are used by the **emc** reconstruction algorithm, as instructed by the `config.ini` configuration file.

2.4.2. Convenience utilities Several convenience utilities are included to help prepare the data for or view the results from the EMC reconstruction algorithm. These utilities are briefly described here.

1. **init_new_recon.py.** This Python utility compiles the C executables in the package, and makes them plus other the rest of the utilities available in a newly initialized reconstruction sub-directory.
2. **sim_setup.py.** This Python utility simulates a single-particle data stream using the parameters listed in the configuration file. This utility, in turn, calls the following modules listed above: **make_densities.py**, **make_intensities.py**, **make_data.py**, and **make_quaternions.py**.
3. **make_powder.py**. Makes a virtual powder pattern from a large stack of diffraction patterns stored in the sparse photon format adopted in this package.
4. **run_emc.py**. Starts the EMC reconstruction by calling its MPI+OpenMP implementation in C. Includes a few convenience operations like calling the **make_quaternion** module to increase the sampling of the rotation group just before starting a reconstruction.
5. **autoplot.py**. Renders the results of the EMC reconstruction, including the diagnostics it generates, with the option of automatically updating the plots when newer intensities become available.

2.5. Configuration and data formats

2.5.1. Configuration file The configuration file is used to pass parameters and file names to the main reconstruction code as well as to the various utilities. The file has the standard `key = value` format with the parameters for different modules grouped by module names in square brackets. There is a global `[parameters]` module, (change to SECTION to avoid confusing with compute modules) containing information about the experimental setup. A typical configuration file is shown in Fig. 4, which corresponds to the first simulation case in Table 1. This default file also shows the use of special keywords used to point to other configuration file parameters (eg. `in_photons_file`). The `[parameters]` module is described below. For other modules, refer to Section 2.4.

The basic parameters of the experiment are:

- `detd`: Detector distance in mm
- `lambda`: Wavelength in Å
- `detsize`: Detector size (assuming square detector) in pixels
- `pixsize`: Pixel size in mm
- `stoprad`: Radius of beamstop in pixels
- `polarization`: Polarization direction (can be x, y, or none)

```
[parameters]
detd = 300
lambda = 6.2
detsize = 150
pixsize = 0.512
stoprad = 10
polarization = x

[make_densities]
in_pdb_file = aux/4BED.pdb
scatt_dir = aux/henke_table
out_density_file = data/densityMap.bin

[make_intensities]
in_density_file = make_densities:::out_density_file
out_intensity_file = data/intensities.bin

[make_detector]
out_detector_file = data/det_sim.dat

[make_data]
num_data = 300000
fluence = 1e10
in_detector_file = make_detector:::out_detector_file
in_intensity_file = make_intensities:::out_intensity_file
out_photons_file = data/photons.emc

[make_quaternion]
num_div = 6
out_quat_file = aux/quat_06.dat

[emc]
in_photons_file = make_data:::out_photons_file
in_detector_file = make_detector:::out_detector_file
in_quat_file = make_quaternion:::out_quat_file
out_folder = data/
log_file = EMC.log
need_scaling = 1
alpha = 0.
beta = 1.
```

Figure 4
Typical configuration file describing various parameters used to perform a basic simulation and reconstruction using the KLH1 (`4BED.pdb`) molecule. Compare these with numbers from Table 1.

2.5.2. Detector file The detector file is an ASCII (human readable) file which describes various properties of the detector. The first line of the file contains a single number which represents the number of pixels. Following that, for each pixel, there are five columns of numbers. The first three columns give the 3D coordinates of the detector pixel in voxels, where the voxels refer to the 3D grid containing the intensity model. The next column gives the product of the polarization and solid angle corrections for that pixel. The last column is an 8-byte unsigned integer whose value is used by the EMC code as well as other utilities to categorize the pixel. Currently, there are three categories:

- 0: Good pixel, used to determine the orientation of a given frame.
- 1: These pixels will not be used to determine the orientation, but will still be merged into the 3D grid using the orientations calculated from category 0 pixels.
- 2: Bad pixel. Pixels whose value will be used neither to determine the orientation nor to calculate the merged 3D intensities.

2.5.3. Photons file (emc format) Since the data in many relevant experiments utilizing the EMC algorithm have very few photons/frame, a sparse binary format is used to store the data. Instead of storing the number of counts at each pixel, only the locations of non-zero counts are stored. since most of the non-zero counts are ones, only the locations of those are stored. For the rest, two numbers are stored, the location and photon count.

The header resides in the first 1024 bytes. The first 4 bytes are a 32 bit integer giving the number of data frames (`num_data`) contained in the file. The next 4 bytes are also a 32 bit integer giving the number of pixels in the detector file used to number the pixels. The next 1016 bytes are currently empty (filled with zeros).

After this, there are `num_data` 32 bit integers giving the number of one-photon events in each frame (`ones`). Similarily, there are `num_data` integers giving the number of multi-photon events (`multi`). The total number of single photon events in all the frames is the sum of all numbers in the `ones` array (S_o). Similarly, let S_m be the total number of multiple photon events. The file contains S_o 32 bit integers giving the locations of the single photon pixels, and then S_m integers with the locations of the multiple photon pixels. This is followed by S_m integers giving the number of photons in each of those multiple photon pixels.

3. Example reconstructions of simulated experiments

We exemplify our package with three simulated single-particle reconstructions using the specifications of Atomic Molecular and Optical Science (AMO) (Ferguson *et al.*, 2015) and Coherent X-ray Imaging (CXI) (Liang *et al.*, 2015) endstations at Linac Coherent Light Source (LCLS) (Emma *et al.*, 2010). For these experiments chose to simulate the reconstruction of keyhole limpet hemocyanin (KLH) 1 didecamer (Gatsogiannis & Markl, 2009) and four-layer Tobacco Mosaic Virus (TMV) (Bhyravbhatla *et al.*, 1998) at AMO and CXI respectively. Notably, the choices in Table 1 yield and average of $\sim$ 100 photons per single-particle diffraction frame.

Table 1
Parameters for EMC reconstructions of simulated single-particle imaging.

	AMO (low)	CXI	AMO (high)
photon energy (keV)	2.0	7.0	2.0
λ, photon wavelength (Å)	6.2	1.77	6.2
z_D, detector distance (mm)	300	350	290
d, detector size (pixel)	150	150	150
l_D, pixel side length (mm)	0.512	0.751	0.512
L, full field of view (nm)	363	82.4	351
beamstop radius (pixel)	10.0	8.0	10.0
fluence (photons/um^2)	$1 \times 10^{10\dagger}$	1×10^{12}	$\mathbf{3.1 \times 10^{12}}$
a, half-period resolution‡(nm)	2.45	0.56	2.5
particle	KLH1§	TMV¶	KLH1§
mass (kDa)	8000	18	8000
R_p, particle radius (nm)	18.9	9.3	18.9
$\widetilde{R}^{\|}$, dimensionless radius	7.7	16.6	7.6
σ, speckle sampling**	9.6	4.45	9.2
N, mean photons/frame	90	90	$\mathbf{2.8 \times 10^4}$
number of frames	3×10^5	5×10^5	1×10^5
max. quaternion sampling††	9	16	9

† Estimated from Loh *et al.* (2013).
‡ Resolution defined from detector's edge.
§ Keyhole Limpet Hemocyanin 1.
¶ Four-layer Tobacco Mosaic Virus.
$^{\|}$ Dimensionless radius, R_p/a.
**Defined as $\widetilde{R} = L/(2R_p)$. See appendix A.
††Sampling and criterion defined in Loh & Elser (2009).

The simulation parameters are shown in Table 1. The detectors here had 1024×1024-pixels, taking on the pixel pitch of the pnCCD(Strüder *et al.*, 2010) and CSPADcitation needed detectors. We decrease the beam fluence to obtain mean photon counts $N \sim 90$ for the first two simulations, mimicking realistic losses from imperfect beam transmission, optics, and cleanup apertures (Loh *et al.*, 2013).

For data sufficiency we use the signal-to-noise ratio parameter defined in Eq. 37 of Loh & Elser (2009),

$$S = \sqrt{\frac{NM_{data}}{M_{rot}}}\ , \tag{5}$$

to estimate the required number of frames M_{data} for $S \sim 50$, where M_{rot} is the number of rotation samplings. Assuming the diffraction frames are uniformly distributed in orientation space, S^2 can be interpreted as the average number of photons per orientation.

The third simulation of of Table 1 was designed to demonstrate how deterministic annealing method can deal with convergence issues caused by very high signal. Most of the parameters were kept identical to the low fluence AMO simulation, but the fluence was up-adjusted to receive 1 mJ x-ray pulses (within an order of magnitude of design specifications (Emma *et al.*, 2010)).

3.1. Discussions on simulated reconstructions

TODO: Describe how we know a reconstruction converges and monitor its progress. Describe how we nurse reconstructions to higher rotation group sampling. Describe how we guide a "heating" up of a beta parameter reconstruction. Describe how we interpret the most-likely orientations plot.

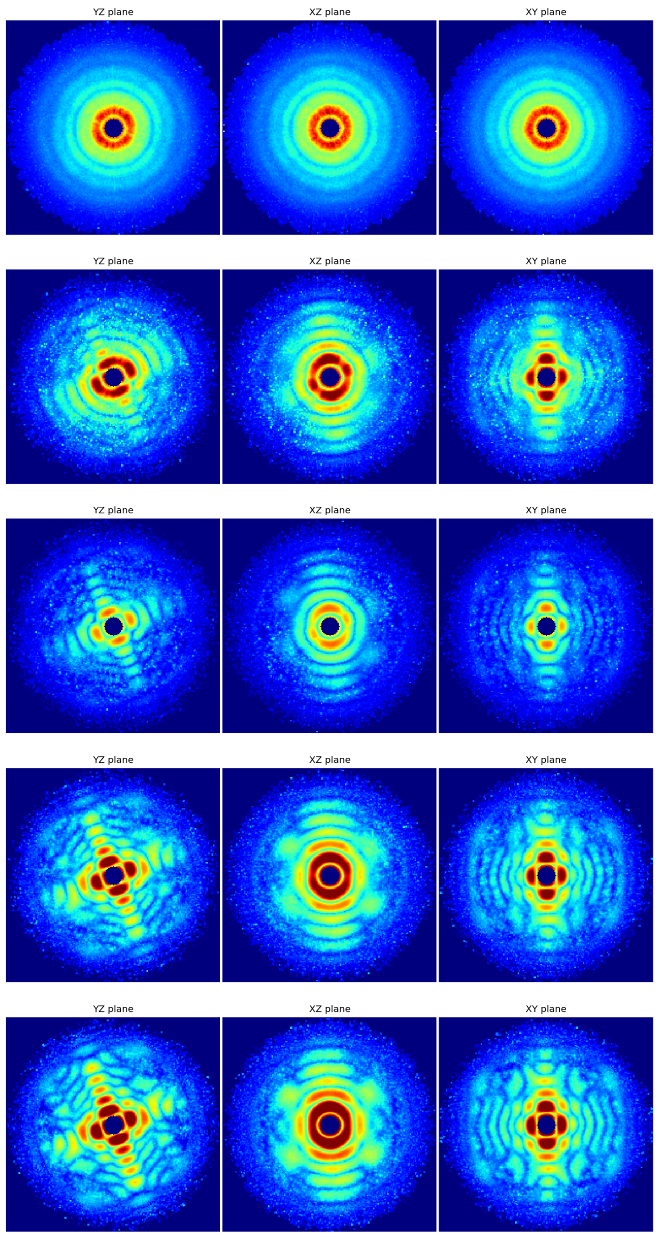

Figure 5
Convergence of diffraction speckle features in a simulated AMO single-particle experiment (parameters listed in Table 1). In each row we render central slices of the 3D diffraction intensities recovered from KLH1 during an EMC reconstruction, for iterations 1,5,10,15, and 30 in descending row order. These figures were generated by the `autoplot.py` utility of Section 2.4.2.

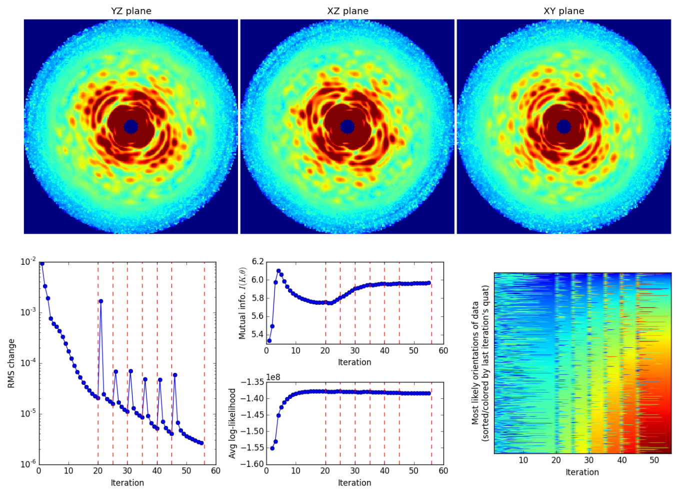

Figure 6
The simulated reconstruction of TMV at the CXI endstation of LCLS (see Table 1). Show here are the central sections of the reconstructed 3D diffraction volume of TMV after 55 iterations. With 88 Intel Xeon X7542 (2.67 GHz) cores this full reconstruction took 7 hours, spending 15 minutes for each of the slowest refinement iterations using 204,960 rotation group samples.

Figure 7
A simulated reconstruction at the AMO endstation at LCLS with high photon fluence (see Table 1).

4. Conclusions and future work

TODO:

Appendix A
Computing speckle sampling on the detector

We use a spherical approximate to estimate the size of diffraction speckles from a scatterer. A sphere of radius R_p, produces diffraction intensities

$$I(\widetilde{q}) = \left|\frac{\sin(\widetilde{q}) - \widetilde{q}\cos(\widetilde{q})}{\widetilde{q}^3}\right|^2 , \tag{6}$$

with dimensionless resolution $\widetilde{q} = 2\pi\widehat{q}R_p$, where we define $\widehat{q} = 2\sin(\phi/2)/\lambda$ as the spatial resolution commonly used in structural biology. Here, ϕ and λ are the scattering angles and photon wavelength respectively (see Figure 8). The width of a diffraction speckle of this spherical approximate is the separation in $\Delta\widehat{q}$ between the zeros of (6). These zeros occur when

$$\tan(\widetilde{q}) = \widetilde{q} , \tag{7}$$

which approaches $\widetilde{q} \to (2n+1)\pi/2$, where $n \in Z$, for large $\widetilde{q}$. As a result, for large $\widetilde{q}$ the separation between the zeros of (6) $\Delta\widetilde{q} \to \pi$, which results in a speckle width in spatial frequency of

$$\Delta\widehat{q} \to 1/(2R_p) . \tag{8}$$

Referring to Figure 1, the coarsest spacings in spatial frequency occur at small scattering angles and is inversely proportional to the field of view L, or $\Delta\widehat{q}_{\text{min}} \sim 1/L$.

The sampling ratio of the diffraction speckle is defined as:

$$\Delta\widehat{q}\,/\,\Delta\widehat{q}_{\text{min}} = \frac{L}{2R_p} = \frac{\lambda}{4R_p \sin\left(\arctan(l_D/z_D)\,/\,2\right)} , \tag{9}$$

where l_D is the width of the detector pixel, and z_D is separation between the detector and interaction region (see Figures 1 or 8). While the ideal sampling ratio of the diffraction speckles should exceed two, the time and memory required for intensity reconstruction rises rapidly when this ratio becomes excessively large (e.g. $\Delta\widehat{q}\,/\,\Delta\widehat{q}_{\text{min}} \gg 5$).

Appendix B
Solid angle correction for square pixels on planar detectors

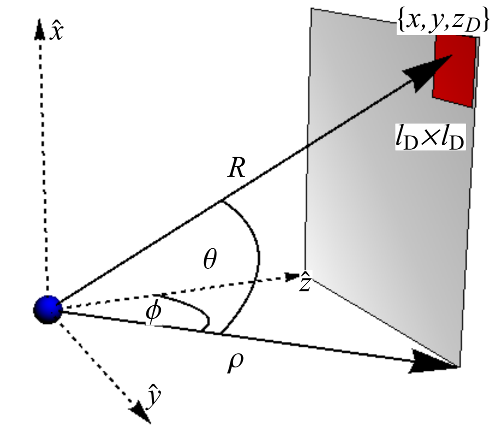

Figure 8
Setup for solid angle correction. In this section we compute the solid angle subtended by the square pixel (red) on the detector plane (gray). The scatterer (blue sphere) is set at the origin of this figure.

In this section we compute the solid angle $\Delta\Omega$ subtended by a square pixel of area $l_D \times l_D$ about the point where a scatterer sits (see Figure 8). To first approximation, the solid angle $\Delta\Omega \sim \cos(\theta)\Delta\phi \cdot \Delta\theta$. The estimate $\Delta\phi$ and $\Delta\theta$, we use the following relations. On a detector z_D from the interaction point, we consider a pixel at position $\{x, y, z_D\}$ in its spherical coordinate representation:

$$\sin(\phi) = y/\rho, \qquad \cos(\phi) = z_D/\rho , \tag{10}$$

where $\rho = \sqrt{y^2 + z_D^2}$. Differentiating $\sin(\phi)$ with respect to ϕ gives

$$\cos(\phi)\Delta\phi \approx \frac{\Delta y}{\rho}\left[1 - (y/\rho)^2\right] . \tag{11}$$

Repeating this for θ

$$\sin(\theta) = x/R, \ \cos(\theta) = \rho/R \ , \qquad (12)$$

with $R = \sqrt{x^2 + y^2 + z_D^2}$, leads to

$$\cos(\theta)\Delta\theta = \frac{\Delta x}{\rho}\left[1 - (x/R)^2\right] \ . \qquad (13)$$

Combining the two, then simplifying, we get the solid angle subtended by the square pixel as

$$\Delta\Omega = \cos(\theta)\Delta\phi\Delta\theta = \frac{l_D^2\, z_D}{R^3} = \frac{l_D^2\, z_D}{\left(x^2 + y^2 + z_D^2\right)^{3/2}} \ . \qquad (14)$$

Appendix C Frame-by-frame intensity scale factor updates

In many real world applications, the incident fluence on each particle will be different. The default implementation in Loh & Elser (2009) assumes uniform incident fluence. Here we derive the likelihood maximizing update rule employed in this package when the `need_scaling` option is turned on. The approach used is similar to that employed in Loh *et al.* (2010), except for a Poisson probability model rather than Gaussian. Let ϕ_d be a scale factor which is proportional to the fluence incident on the particle in frame d. Thus, Eq. 3 and Eq 2 become,

$$P_{dr} = \frac{R_{dr}}{\sum_r R_{dr}} \qquad (15)$$

$$\text{and } R_{dr} = \prod_t \frac{(W_{rt}\phi_d)^{K_{dt}} e^{-W_{rt}\phi_d}}{K_{dt}!} \qquad (16)$$

In expectation maximization, one would like to find updates for the intensity tomograms, W' and ϕ' which maximize the total log-likelihood Q given by

$$Q(W', \phi') = \sum_d \sum_r \sum_t P_{dr}\left[K_{dt} \log(W'_{rt}\phi'_d) - W'_{rt}\phi'_d\right]$$

Here, P_{dr} are the probabilities calculated using the current models for W and ϕ. Unfortunately, an analytical update rule for both these quantities simultaneously which maximizes Q is not available. We use the strategy of updating one while keeping the other constant. Setting partial derivatives with respect to W' and ϕ' equal to 0, we obtain

$$W'_{rt} = \frac{\sum_d P_{dr} K_{dt}}{\sum_d P_{dr}\phi_d} \qquad (17)$$

$$\phi'_d = \frac{\sum_t K_{dt}}{\sum_{rt} P_{dr} W_{rt}} \qquad (18)$$

This modification to the update rule in Eq. 4 is used when the user expects variable incident fluence on the particle.

Acknowledgements N.D.L. would like to thank the support of the Lee Kuan Yew Endowment fund, and the assistance by the IT facility at the Centre for Bio-Imaging Sciences.

References

Ayyer, K., Geloni, G., Kocharyan, V., Saldin, E., Serkez, S., Yefanov, O. & Zagorodnov, I. (2015). *Structural Dynamics*, **2**(4), 041702.

Bhyravbhatla, B., Watowich, S. J. & Caspar, D. L. (1998). *Biophysical journal*, **74**(1), 604–615.

Ekeberg, T., Svenda, M., Abergel, C., Maia, F. R., Seltzer, V., Claverie, J.-M., Hantke, M., Jönsson, O., Nettelblad, C., van der Schot, G. *et al.* (2015). *Physical review letters*, **114**(9), 098102.

Emma, P., Akre, R., Arthur, J., Bionta, R. M., Bostedt, C., Bozek, J., Brachmann, A., Bucksbaum, P. H., Coffee, R., Decker, F. J., Ding, Y., Dowell, D., Edstrom, S., Fisher, A., Frisch, J., Gilevich, S., Hastings, J., Hays, G., Hering, P., Huang, Z., Iverson, R., Loos, H., Messerschmidt, M., Miahnahri, A., Moeller, S., Nuhn, H.-D., Pile, D., Ratner, D., Rzepiela, J., Schultz, D., Smith, T., Stefan, P., Tompkins, H., Turner, J., Welch, J., White, W., Wu, J., Yocky, G. & Galayda, J. N. (2010). *Nat. Photonics*, **4**, 641–647.

Ferguson, K. R., Bucher, M., Bozek, J. D., Carron, S., Castagna, J.-C., Coffee, R., Curiel, G. I., Holmes, M., Krzywinski, J., Messerschmidt, M. *et al.* (2015). *Journal of synchrotron radiation*, **22**(3), 0–0.

Gatsogiannis, C. & Markl, J. (2009). *Journal of molecular biology*, **385**(3), 963–983.

Liang, M., Williams, G. J., Messerschmidt, M., Seibert, M. M., Montanez, P. A., Hayes, M., Milathianaki, D., Aquila, A., Hunter, M. S., Koglin, J. E. *et al.* (2015). *Journal of synchrotron radiation*, **22**(3), 0–0.

Loh, N., Bogan, M., Elser, V., Barty, A., Boutet, S., Bajt, S., Hajdu, J., Ekeberg, T., Maia, F. R., Schulz, J. *et al.* (2010). *Physical review letters*, **104**(22), 225501.

Loh, N. D., Starodub, D., Lomb, L., Hampton, C. Y., Martin, A. V., Sierra, R. G., Barty, A., Aquila, A., Schulz, J., Steinbrener, J. F. *et al.* (2013). *Opt. Express*, **21**(10), 12385–12394.

Loh, N.-T. D. & Elser, V. (2009). *Physical Review E*, **80**(2), 026705.

Strüder, L., Epp, S., Rolles, D., Hartmann, R., Holl, P., Lutz, G., Soltau, H., Eckart, R., Reich, C., Heinzinger, K., Richter, R., Foucar, L., Schlichting, I., Ullrich, J. & Shoeman, R. L. (2010). *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.* **614**(3), 483–496.

Ueda, N. & Nakano, R. (1998). *Neural Networks*, **11**(2), 271–282.