cctbx Spotfinder: a faster software pipeline for crystal positioning

Nicholas K. Sauter[a]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

Correspondence email: NKSauter@LBL.Gov

Synopsis

This article documents the program `distl.signal_strength`, used to locate candidate Bragg spots on X-ray diffraction images for macromolecular crystallography. While standalone analysis requires about 3 seconds/image on typical Linux systems, an order of magnitude increase in the overall throughput can be achieved using concurrent multiprocessing within a client/server implementation. The program is thus suitable for the rapid location of the best-diffracting positions within the crystal sample using a low-dose X-ray probe.

Introduction

The development of microfocused X-ray beams at synchrotron facility endstations has made it possible to obtain higher-signal, lower mosaicity datasets from small crystal samples (Fischetti *et al.*, 2009). With small crystal dimensions of order 5 μm (matched to the diameter of present microbeams), it may be necessary to examine numerous specimens in order to assemble a complete dataset, making it highly desirable to automate the process of centering each sample in the microbeam. Various approaches have been proposed (Pothineni *et al.*, 2006; Song *et al.*, 2007), and it has been possible to automatically center the sample-holding loop (or other device) using videomicroscopy. However, it has been more difficult to visually identify small crystals within the loop, thus requiring the more robust approach of scanning the sample (translating it with respect to the microbeam), so as to collect a low-dose diffraction image at each translational position. It has been noted that such X-ray based autocentering may take up to 10 minutes for the smallest crystals (Song *et al.*, 2007), which presumably require very fine rastering to locate. This outcome could be improved dramatically by the use of a photon-counting pixel array detector such as the Pilatus-6M that supports a framing rate of 10 Hz. An accompanying challenge is to write software that can quantify the measured signal at a reasonably high turnaround rate, so that the diffraction analysis can keep pace with the rapid data acquisition needed for fine-grained coverage of the sample.

The standalone spotfinder program and client/server pair described below are meant to provide a toolbox for rapid diffraction analysis within the beamline computing environment. The software architecture is the same as described for the cctbx package (http://cctbx.sf.net; Grosse-Kunstleve *et al.*, 2002), with a high-level Python scripting interface that is meant to encourage the reuse and adaptation of code as the problems change. One example is that new detector hardware types can be readily supported as they are introduced.

Installation

The spotfinder program is bundled and distributed with the packages *LABELIT* and *PHENIX*, and can therefore be downloaded from either Web site (http://cci.lbl.gov/labelit/ or http://www.phenix-online.org/). Newest-version *PHENIX* installers are released on an almost daily basis, while *LABELIT* releases currently cycle every few months. Follow the installation instructions, and source the appropriate setup file (given in the message printed by the installer) to place the command-line dispatchers on path.

Command line reference

Use the command `distl.signal_strength` to produce a quick summary of the diffraction characteristics from a single diffraction exposure:

Usage:
```
      distl.signal_strength image_filename [parameter=value [parameter2=value2
...]]
```
Example:
```
      distl.signal_strength lysozyme_001.img distl.res.outer=2.0
```

Optional command line parameters change the program operation as follows (Default values are shown):

`distl.res.outer=None`

If specified, this is the outer (high) resolution limit to be used for the diffraction analysis, in Ångstroms. This option is useful for two reasons. Firstly, if a large number of images from the same crystal specimen are to be examined (in order to locate the best-diffracting position), then placing a uniform limit on the resolution range permits the number of observed Bragg spots to act as a good proxy of the diffraction strength. Secondly, areas on the image that are outside the outer resolution limit are not analyzed, potentially producing a substantial speedup in program throughput. [Note: areas on the stored image that are not part of the active detector are already excluded from analysis. Examples are the corner areas on circular image plates (such as the Mar), the inactive pixel stripes between fiber-optic tapers on CCD detectors (such as the ADSC), and the inactive pixel stripes between modules of the Pilatus detector.] Camera parameters such as the sample-to-detector distance and swing-arm angle are taken from the file header, to calculate the resolution at each pixel.

`distl.res.inner=None`

If specified, this is the inner (low) resolution limit to be used for the diffraction analysis, in Ångstroms. Excluding the low-resolution Bragg spots may be a potential workaround if there is severe beam leakage around the beam stop masquerading as a Bragg signal. However, the built-in heuristics that filter out unusually-large signal areas and very intense signals will normally exclude such artifacts anyway. The inner resolution cutoff has no effect on overall program throughput, as the filter is applied after the image is analyzed. This program option is only recommended for unusual situations.

`distl.minimum_signal_height=[1.5 for CCDs; 2.5 for pixel-array detectors]`

This parameter determines whether a given pixel is classified as background or signal. Active pixels are classified within non-overlapping 100×100-pixel tiles. For each tile, the best-fit plane is chosen to represent the background level. Pixel heights are distributed above and below this plane with a normal distribution whose standard deviation σ is readily calculated. Pixels higher than `minimum_signal_height`×σ above the background are classified as signal. Two successive rounds (now with 50×50-pixel tiles) are employed to remove the signal pixels from the background level. With CCD detectors the longstanding default of 1.5σ has been a highly successful compromise between increased sensitivity (favoring a lower cutoff) and better noise discrimination (favoring a higher cutoff; any cutoff lower than 1.25σ produces numerous false Bragg spots). For pixel-array detectors, which lack any significant point-spread function, 2.5σ gives better results and is now the pixel-array default. Images from very long unit cells (viruses, ribosomes) potentially have close spots that run into each other, without being separated down to the baseline. The heuristic rules in `distl` will reject these close signals, and in severe cases this will interfere with the ability to index the lattice. The solution is to raise the cutoff level to as high as 5σ, thus correctly separating the close spots. It is therefore important to consider raising the `minimum_signal_height` if the crystals have a large unit cell. At present there is no automatic way to do this; the program has no way to know the cell length ahead of time. In the future it will be beneficial to add some preliminary noise analysis to provide more sophisticated guidance for spot detection.

```
distl.minimum_spot_area=None [5 for pixel-array detectors, 10 otherwise]
```

If specified, this is the minimum number of contiguous pixels that is considered to be a spot. For image plate and CCD detectors the hard-coded default is 10 pixels/spot. With pixel-array technology, spots are much narrower due to the sharp point-spread function, but they are still generally distributed over a few pixels. The pixel-array default is therefore set to 5 pixels/spot. Synchrotron beamlines experimenting with new detectors or optics should test this parameter to find the optimum setting!

```
verbose=False
```

If True, the program will output detailed statistics on all Bragg spots and show signal pixel locations. This is potentially useful for beamline commissioning.

```
pdf_output=None
```

If specified, this is the filename (*.pdf) for an optional PDF-format graphical output showing the image and associated Bragg signals. This visual output is the most direct way to understand the "big picture" of what the program results mean. Comparing one's own visual impression of the diffraction image with the marked up spotfinder results will reveal whether the program has missed real Bragg spots (false negatives) or overinterpreted bad signals (false positives). The contrast level is pre-set internally. Pink stripes are used to color-code the inactive areas between detector modules (for the Pilatus), and red squares (again for the Pilatus) are inactive pixels, which should be the same on every image for a given instrument. Within Bragg spots, color circles indicate that the spot has been tagged as a "good Bragg spot candidate" (see below). Pink circles tag the position of the maximum pixel, and red circles show the spot center of mass.

```
--help
```

Produces an informative program synopsis.

Program operation and output

Computational steps performed by the program have already been described in several publications (Zhang *et al.*, 2006; Sauter *et al.*, 2004; Sauter & Zwart, 2009; Sauter & Poon, 2010). Typical results written to *stdout* are as follows:

```
                    File : 0_clmtr_edge_403.cbf
              Spot Total :    177
     In-Resolution Total :    170
   Good Bragg Candidates :    157
               Ice Rings :      1
     Method 1 Resolution :   2.55
     Method 2 Resolution :   2.08
       Maximum unit cell :  108.9
%Saturation, Top 50 Peaks :   0.09
In-Resolution Ovrld Spots :      0

Bin population cutoff for method 2 resolution: 20%

Number of focus spots on image #403 within the input resolution range: 170
Total integrated signal, pixel-ADC units above local background (just the good Bragg candidates) 147322
Signals range from 37.2 to 11773.1 with mean integrated signal 997.2
Saturations range from 0.0% to 0.7% with mean saturation 0.0%
```

Three types of spot count are listed:

- "Spot Total" is the number of separate spots that rise above the minimum thresholds for signal height and spot area. A graph of average pixel intensity vs. resolution is examined to prefilter likely ice rings.
- "In-Resolution Total" is the subset remaining after high- and low-resolution filters are applied. The command-line distl.res filter is applied here (if given), as well as the "Method 2 Resolution" filter described below.
- "Good Bragg Candidates" are the spots remaining after the application of several spot-quality heuristics:
    - A histogram of spot count vs. resolution is used to implement a second filter for ice rings.
    - Spots with ill-defined profiles having more than two signal maxima are not counted.
    - Outliers in intensity, area, eccentricity and skewness are thrown away.
    - Spots with too-close nearest neighbors are filtered.

Limiting resolution is estimated by the two methods defined in Zhang *et al.* (2006). Method 2 is used to choose spots for *LABELIT* autoindexing. It relies on the ability to produce a histogram of spot count vs. resolution, and therefore requires a minimum number of spots (usually 25). The resolution cutoff is determined by noting the falloff in bin population at higher diffraction angles. The maximum unit cell is estimated by observing nearest-neighbor distances and assuming that the closest spacing corresponds to the largest unit cell length.

"In-resolution" spots are used to produce several signal strength metrics: "% Saturation", "In-Resolution Overloaded Spots", "Signals range" and "Saturations range".

A final "Total integrated signal" metric is computed on the "Good Bragg Candidates". This value is the summed signal height (corrected for background) over all signal pixels. ADC units are analog-to-digital units, as recorded in the raw image file.

Performance assessment for crystal positioning

To position the crystal optimally in the beam, we aim to translate the sample to the position that maximizes the total number of good Bragg spots, or average intensity of Bragg spots, within a given resolution range. Low-dose X-rays can be used to perform this raster scan over the sample, as described (Song *et al.*, 2007). Very high throughput is achieved with shutterless exposures using a Pilatus-6M detector. However, the resulting turnaround time of about 0.2 seconds/image places very high performance requirements on the computational pipeline; which typically takes about 3 seconds to process one image. The magnitude of the challenge can be assessed by profiling the spotfinding code, as is done here with a 64-bit, 2.9 GHz Xeon machine running Fedora Core 8. The processor was equipped with 32 GB RAM and 16 CPU cores, although only one core was used by the single-threaded spotfinder process:

Computational step	Typical time/image	Comments
Load the dynamic libraries	0.50 sec	Client/server removes this overhead
Read file from NFS disk	0.70 sec	Local file I/O takes only 0.04 sec
Uncompress CBF image to memory	0.27 sec	cctbx-optimized code takes only 0.09 sec
Classify pixels: background/signal	0.63 sec	
First ice ring filter	0.23 sec	
Find all spots	0.14 sec	
Second ice ring filter	0.15 sec	
Additional heuristics for good spots	0.15 sec	
Total time:	2.77 sec	

Client-server architecture

The above-listed performance data show that an order-of-magnitude improvement is needed to keep pace with Pilatus-6M acquisition. We therefore move to a client-server architecture that eliminates the need to reload the dynamic libraries for each image (since the server process is persistent), and runs with multiple processes, so that numerous images may be processed concurrently within separate cores. At present, 16-core CPUs are available for under $6000, so they are within reach of beamline operating budgets. With this scheme it is easy to achieve the desired 0.2 second/image total throughput.

Server:

```
distl.mp_spotfinder_server_read_file [parameter=value ... ]
```

The program operates as a multithreaded server. Allowed parameters are:

- `distl.port=8125` (Required) The server will listen for requests on this port.
- `distl.processors=1` (Required) The total number of processes to be forked to listen for requests.
- `distl.minimum_spot_area=` same as above, the minimum spot area in pixels.
- `distl.minimum_signal_height=` same as above, the minimum signal height.
- `distl.res.outer=` same as above, the outer resolution limit in Ångstroms.

Client:

```
distl.thin_client <filepath> <host> <port>
```

No keyword parameters are allowed. The client simply takes the filepath (must be a valid filepath on the server machine), host name (usually "localhost") and port number, and outputs the spot analysis to *stdout*.

Notes:

The server can be killed from the command line by Ctrl-C, or via the client by sending the message:

```
distl.thin_client EXIT <host> <port>
```

While the server is supposed to queue requests, too many at once can cause some requests to drop out, in a manner that is not fully characterized. Therefore in the example given (`<path to sources>/spotfinder/servers/thin_client.csh`), a `/bin/sleep` command is used to time the requests at a reasonable pace given the particular server host and number of processes. In application, it is the responsibility of the caller (the beamline data collection process) to avoid overloading the server with too many simultaneous requests.

LABELIT contains a separate program (`labelit.distl`) to perform a similar spot finding function for use in autoindexing. That implementation is not suitable for multiprocessing because it writes the spot list to disk in the current working directory under a constant file name. The file can be erased with the command `labelit.reset`.

In contrast, `distl.signal_strength` and `distl.mp_spotfinder_server_read_file` perform all work in memory without any file output.

Results

Low-dose exposures were used to probe samples on a 5 × 6-position grid (Diamond, ADSC detector) or a

5 × 5-position grid (SSRL, Pilatus-6M). Images were visually inspected to produce a subjective rank for each exposure, taking into account factors such as the limiting resolution and strength of the Bragg spots. Separately, the images were analyzed with the automated spotfinder process, using an up-front resolution limit (`distl.res.outer`) of 3.0 Å.

An automated ranking scheme was developed that is consistent with the subjective rank from visual inspection. The 30 or 25 images from each sample are ranked by two criteria, the "Total Integrated Signal" in pixel-ADC units, and the count of "Good Bragg Candidates", and the rank scores based on these two criteria are averaged with equal weight. If two images receive the same average score, the tie is broken by giving a higher priority to "Total Integrated Signal". No score is developed unless there is a numerical score for "Method 2 Resolution" (although the resolution isn't actually included in the ranking); therefore images with too few spots are not scored.

Application programming interface

The spotfinder software can be accessed directly through Python code. Although the necessary interface is not formally documented, example usage is given by the program itself, within the `<path to sources>/spotfinder/` directory:

`./command_line/signal_strength.py`: Illustrates the use of command-line parameters.

`./applications/signal_strength.py`: Illustrates the core function calls, as well as the detailed parsing of the resulting spot list.

Acknowledgments

Numerous contributions from synchrotron groups helped to shape this software. Test results were contributed by Michael Soltis, Ana González and Penjit (Boom) Moorhead (Stanford Synchrotron Radiation Lightsource); Craig Ogata, Mark Hilgart and Sudhir Pothineni (GM/CA-CAT, Advanced Photon Source); John Skinner and Annie Heroux (National Synchrotron Light Source); and Alan Ashtun, Katherine McAuley, Graeme Winter and Mark Williams (Diamond Light Source, Ltd., UK). Herbert Bernstein (Dowling College) offered his expertise toward the optimization of CBF decompression, and Chris Nielson (Area Detector Systems Corp.) engaged in valuable discussions. Ralf Grosse-Kunstleve at LBNL helped to implement the overall software architecture. The financial support of NIH/NIGMS under grant number R01GM077071 is gratefully acknowledged. The work was partly supported by the US Department of Energy under Contract No. DE-AC02-05CH11231.

References

Fischetti, R.F., Xu, S., Yoder, D.W., Becker, M., Nagarajan, V., Sanishvili, R., Hilgart, M.C., Stepanov, S., Makarov, O. & Smith, J.L. (2009). Mini-beam collimator enables microcrystallography experiments on standard beamlines. *J. Synchrotron Rad.* **16**, 217-225.

Grosse-Kunstleve, R.W., Sauter, N.K., Moriarty, N.W. & Adams, P.D. (2002). The Computational Crystallography Toolbox: crystallographic algorithms in a reusable software framework. *J. Appl. Cryst.* **35**, 126-136.

Poon, B.K., Grosse-Kunstleve, R.W., Zwart, P.H. & Sauter, N.K. (2010). Detection and correction of underassigned rotational symmetry prior to structure deposition. *Acta Cryst.* D**66**, 503-513.

Pothineni, S.B., Strutz, T. & Lamzin, V.S. (2006). Automated detection and centring of cryocooled protein crystals. *Acta Cryst.* D**62**, 1358-1368.

Sauter, N.K., Grosse-Kunstleve, R.W. & Adams, P.D. (2004). Robust indexing for automatic data collection. *J. Appl. Cryst.* **37,** 399-409.

Sauter, N.K. & Zwart, P.H. (2009). Autoindexing the diffraction patterns from crystals with a pseudotranslation. *Acta Cryst.* D**65**, 553-559.

Sauter, N.K. & Poon, B.K. (2010). Autoindexing with outlier rejection and identification of superimposed lattices. *J. Appl. Cryst.* **43**, 611-616.

Song, J., Mathew, D., Jacob, S.A., Corbett, L., Moorhead, P. & Soltis, S.M. (2007). Diffraction-based automated crystal centering. *J. Synchrotron Rad.* **14**, 191-195.

Zhang, Z., Sauter, N.K., van den Bedem, H., Snell, G., and Deacon, A.M. (2006). Automated diffraction image analysis and spot searching for high-throughput crystal screening. *J. Appl. Cryst.* **39**, 112-119.