l_bnl_compress: Lossy but not lossy compression, a Python script to compress MX data

Herbert J. Bernstein[1], Jean Jakoncic[2]

[1]Fresh Pond Research Institute, c/o NSLS-II, Bldg 745, Brookhaven National Laboratory, Upton, New York, 11973-5000 USA

[2]NSLS-II, Bldg 745, Brookhaven National Laboratory, Upton, New York, 11973-5000 USA

For a copy of the code and talk see http://github.com/nsls-ii-mx/l_bnl_compress

Lombard, IL ACA Meeting, 18 – 22 July 2025

BERKELEY LAB DIALS Diffraction Integration for Advanced Light Sources Brookhaven National Laboratory

For code and a copy of this talk:

http://github.com/nsls-ii-mx/l_bnl_compress

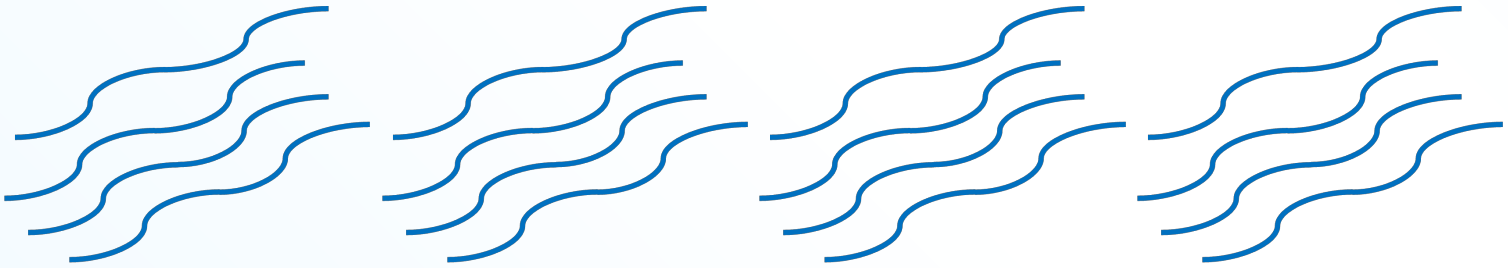
- Why?
- How?
- Example
- Estimating Compression Ratios
- Parallel Operation
- Usage <– try to get here today
- Installation Prerequisites
- Installation Process

Why?

- When X-ray crystallography was invented the problems being presented involved very few atomic positions to be determined from a modest number of Bragg reflections in vastly over-determined systems. Now we face problems with hundreds of thousands to millions of reflections in seriously under-determined systems. The volumes of data and the rate at which they arrive strain the storage capacity of our computer systems and networks.

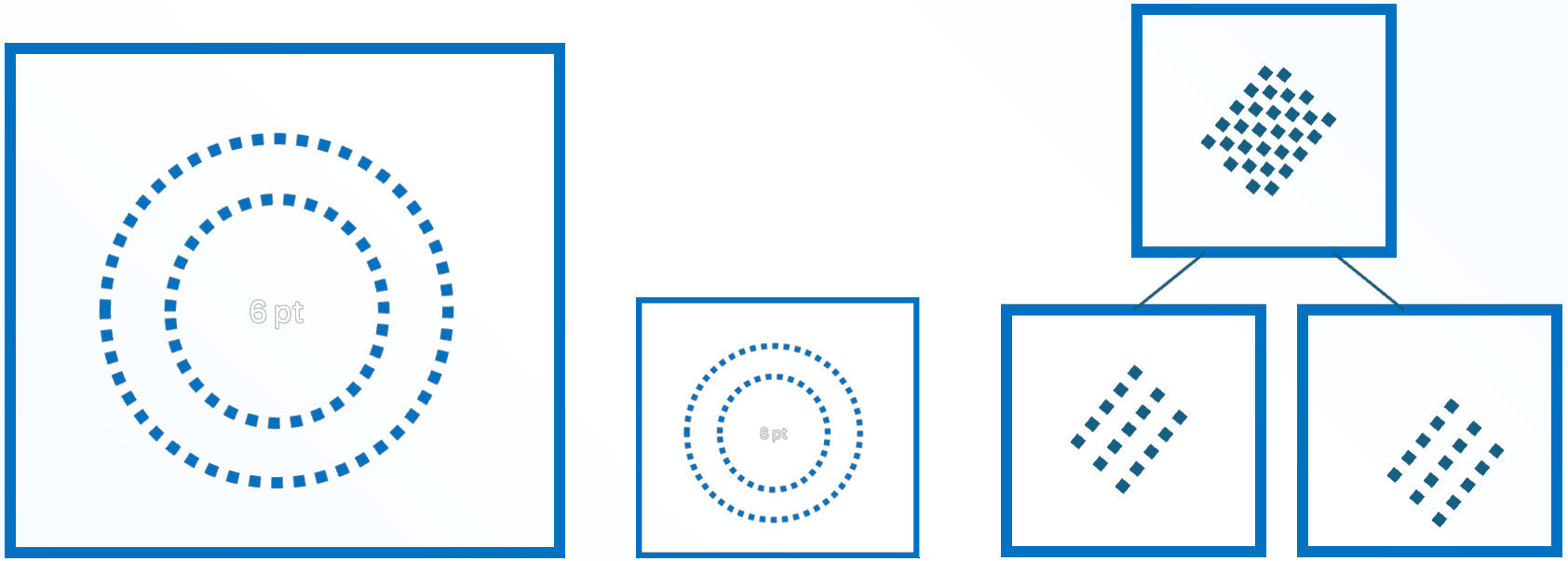
- As recently as the mid 1990's to the early 2000's we all thought it would be sufficient to carefully organize our data in a way the retained all the information content (intensities and locations) of all the Bragg refections, and just do **faithful compression**.

- Now, especially in a era of tight budgets, we need to move to **lossy compression** and carefully choose which information to retain and which information to discard



How? Sum and Bin

- l_bnl_compress.py is a Python script implementing the lossy compressions described in Bernstein *et al.* (2024) for macromolecular crystallographic diffraction data. The lossy compressions use **pixel-by-pixel binning**, **image-by-image summing** for preprocessingl



How? JPEG-2000 and HCompress

- I_bnl_compress.py is a Python script implementing the lossy compressions described in Bernstein *et al.* (2024) for macromolecular crystallographic diffraction data. In addition to **pixel-by-pixel binning**, **image-by-image summing**, the lossy compressions use **JPEG-2000 Daubechies (DB) wavelet compression** from the movie industry, and **HCompress Haar (now known as DB0) wavelet compression** from astronomy which are combined with the usual lossless MX compressions. In its current version, it is intended to compress hdf5 files from Dectris Eiger pixel array detectors but can be updated to compress data from other detectors and formats.

- Depending on the data collection parameters (frame oscillation width) and number of reflections, we recommend starting with moderate achieved total compression ratio of 50. In many cases, at moderate levels of loss of image detail, essential crystallographic details are retained even though in terms of image detail the compressions achieved are lossy.

- When combining effects of frame summing, pixel binning and either JPEG-2000 or Hcompress compression, frames are always pre-processed first (summing and or binning) to increase intensity of peaks, that systematically results in preserved reflection integrated intensities during data reduction.

A Thermolysin Compression Example

- The data from a 2000-frame thermolysin dataset was collected as rotation data on an Eiger 9m at the BNL AMX beamline at 0.920119 Ångstrom wavelength and a 200 millimeter detector distance. The data was organized by Dectris Filewriter as a master file and as four data files with 500 frames each in hdf5 format, compressed by the commonly-used lossless bslz4 compression. The sizes of the original files are a total of 3,566,315,563 bytes:

bytes	name
81,904,228	tlys-1023_14689_master.h5
855,324,052	tlys-1023_14689_data_000004.h5
862,534,593	tlys-1023_14689_data_000003.h5
905,555,017	tlys-1023_14689_data_000002.h5
860,997,673	tlys-1023_14689_data_000001.h5

Table 1: File sizes with bslz4 compression in Tlys1023

A Thermolysin Compression Example (continued)

- We applied lossy compression to this data with four sets of options, each with 2 by 2 pixel binning, 2 frame summing and with Hcompress options H12 and H16 or JPEG-2000 options J120 and J360. To allow more parallelism in processing, data files with only 100 frames each were produced with a final bslz4 compression. The total compressed file sizes and compression ratio relative to lz4 lossless compression and 2 bytes/pixel raw uncompressed 40,666,360,000-byte images are:

bytes	compression	extra compression	total compression
364821870	b2s2h16	10	111
527639659	b2s2j360	7	77
664234244	b2s2h12	5	61
679465102	b2s2j120	5	60

Table 2: extra compression ratios and full compression ratios for Tlys1023

A Thermolysin Compression Example (continued)

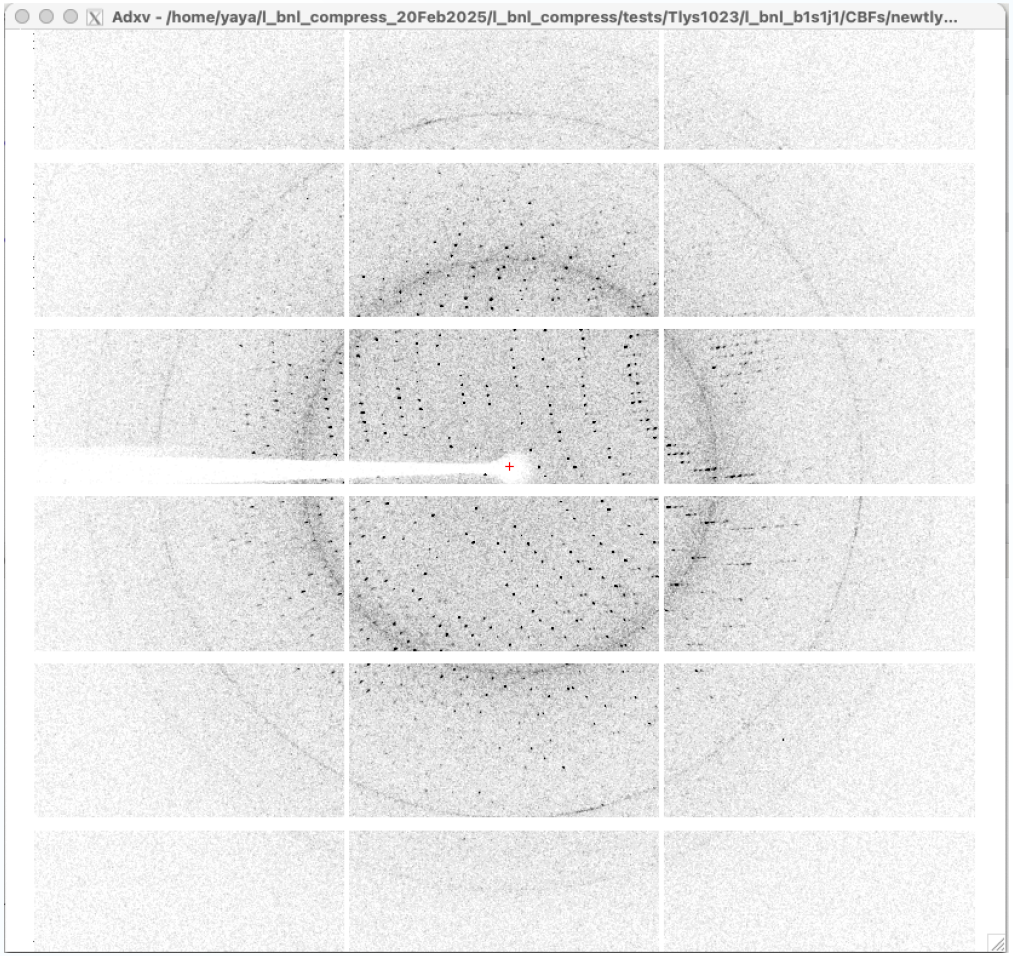


Fig. 1: Frame 1000 from Tlys1023

A Thermolysin Compression Example (continued)

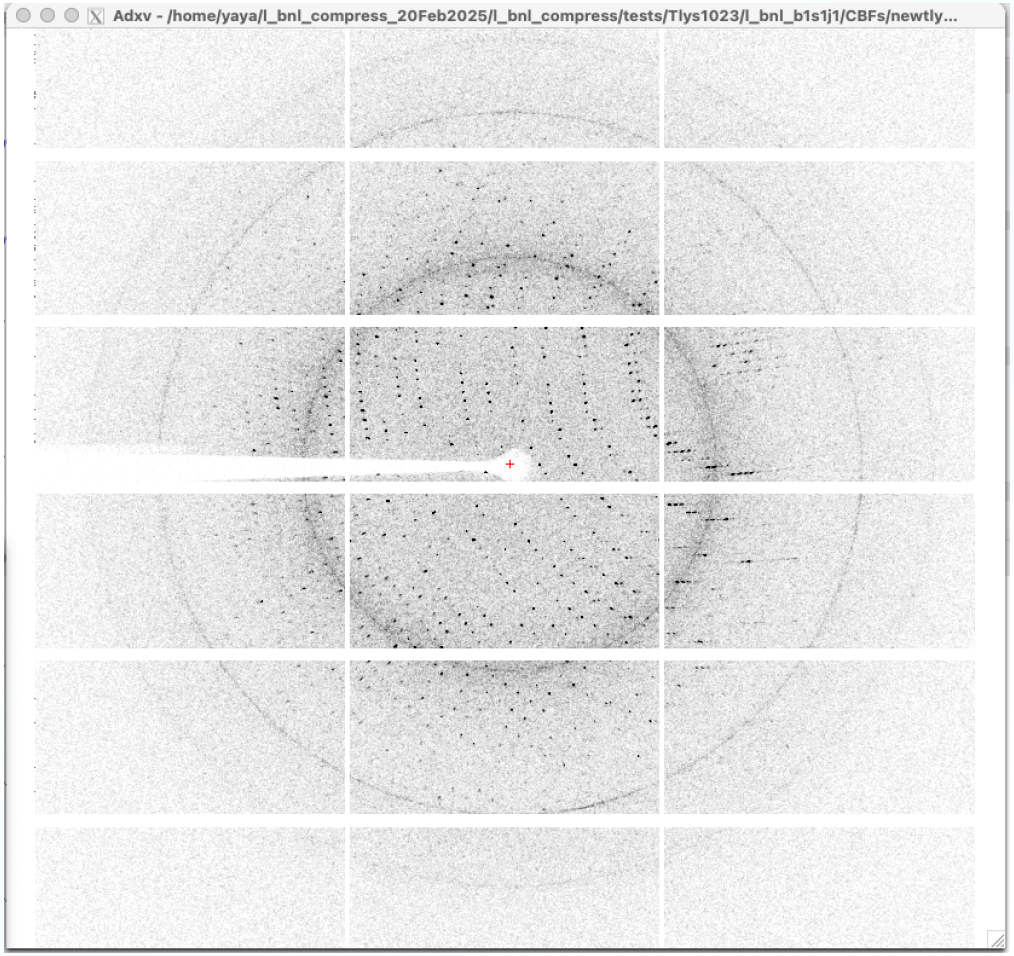


Fig. 2: Frame 1001 from Tlys1023

A Thermolysin Compression Example (continued)

- With these compressions the combination of pixels in original frames 1000 and 1001 generates frame 500 in each compressed dataset, as shown by `adxv` in Figs. 1, 2, 3, 4, 5 and 6.

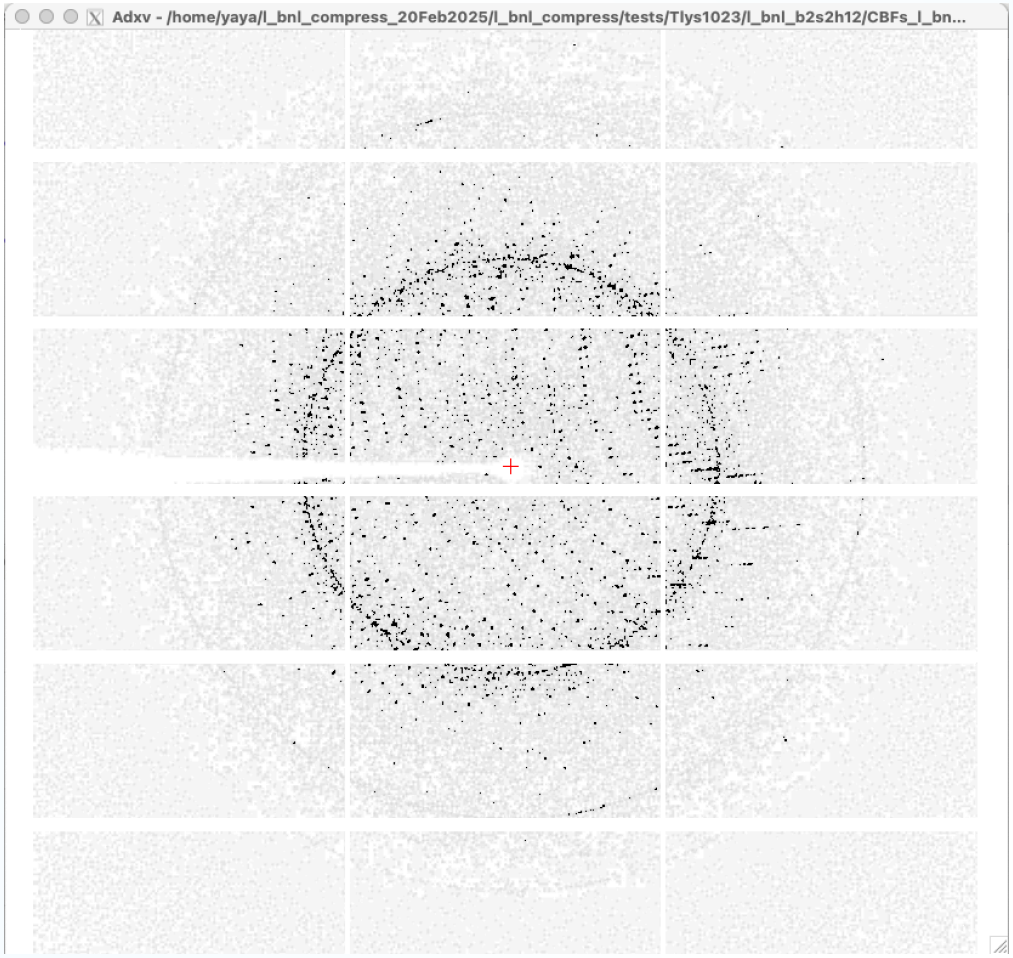


Fig. 3: Frame 500 from l_bnl_compress run with options -b 2 -s 2 -H 12 -c bslz4

A Thermolysin Compression Example (continued)

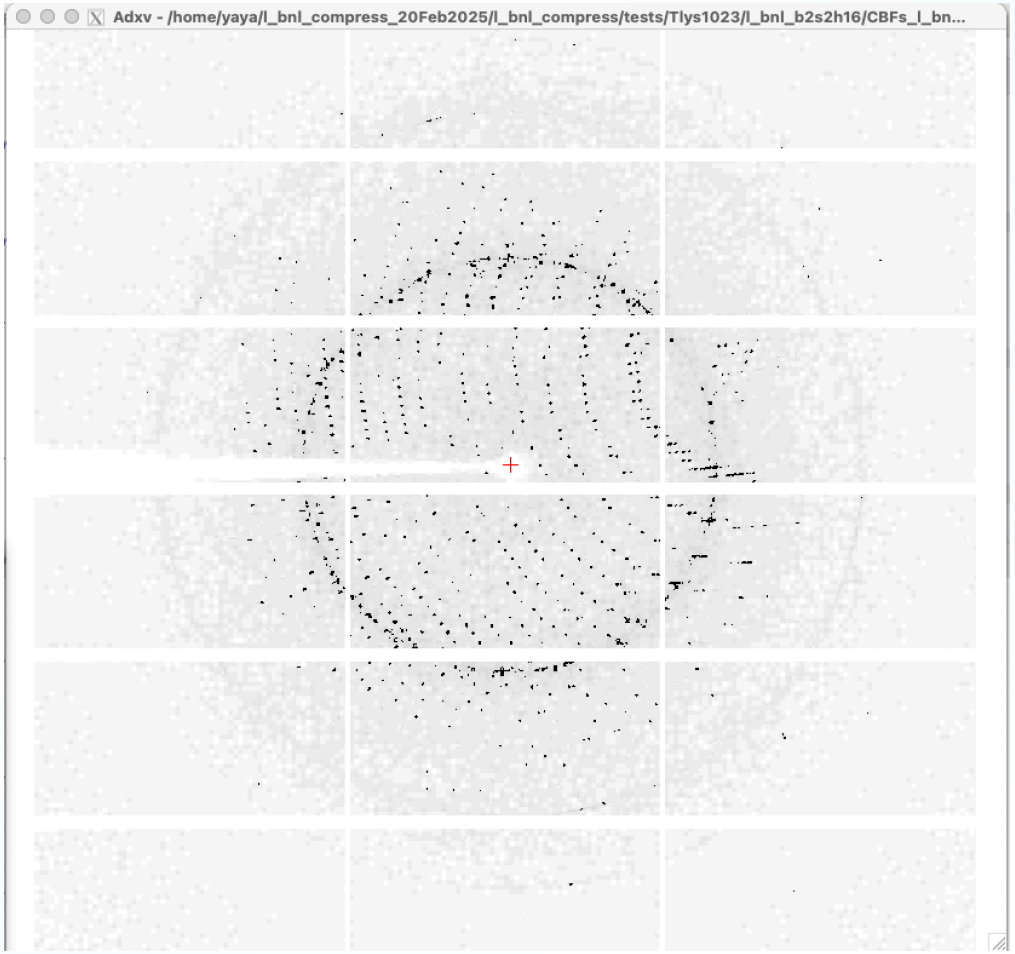


Fig. 4: Frame 500 from l_bnl_compress run with options -b 2 -s 2 -H 16 -c bslz4

A Thermolysin Compression Example (continued)

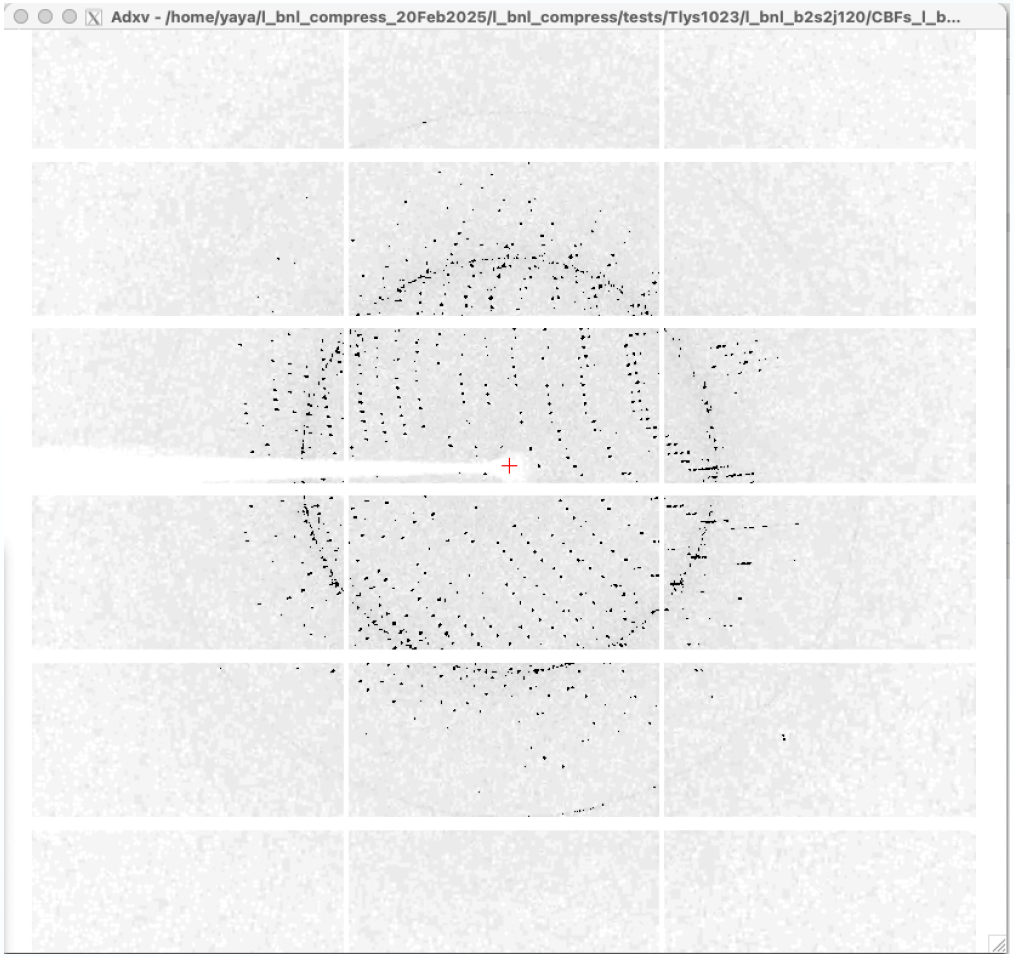


Fig. 5: Frame 500 from l_bnl_compress run with options -b 2 -s 2 -J 120 -c bslz4

A Thermolysin Compression Example (continued)

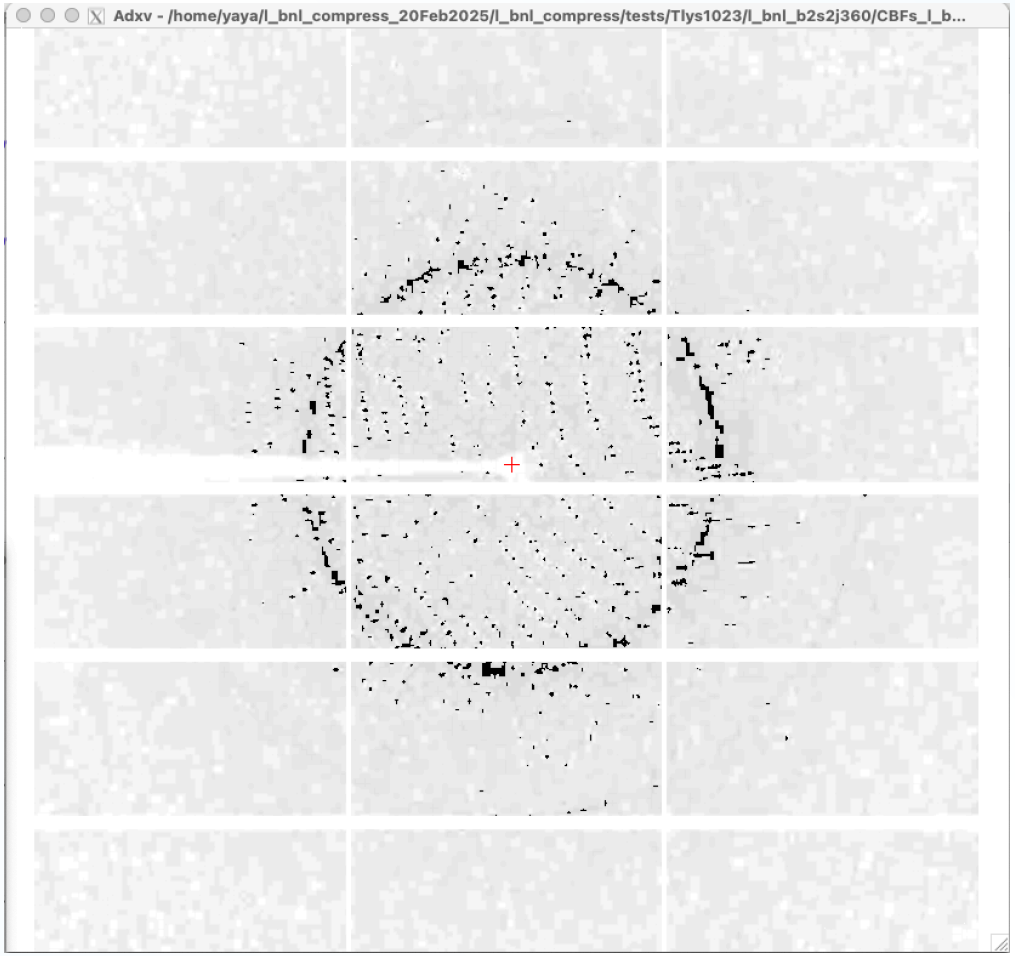


Fig. 6: Frame 500 from l_bnl_compress run with options -b 2 -s 2 -J 360 -c bslz4

Estimating Compression Ratios

- In order to facilitate the estimation of compression ratios, l_bnl_compress provides the “-q” or “–out-squash” option which saves raw Hcompress or JPEG-2000 images to files. These tend to be much smaller than the equivalent HDF5 bslz4 or CBF byte-offset files, but normal crystallographic processing pipelines are not yet ready to read such files.

Parallel Operation

- When run an a computer with sufficient memory, the program is capable of coarse-grained process-level operation by use of the -p threads option. When more than one thread is specified, the -H, -J or -K lossy compressions are broken into the specified number of threads organized around whole datablocks processed in parallel. On a small machine only 2 to 4 threads are likely to fit in memory at the same time, but on larger modern machines, a linear speed-up can be achieved, especially when both the program and the data are memory resident.

Usage

```
l_bnl_compress.py [-h] [-1 FIRST_IMAGE]
[-b BIN_RANGE] [-c COMPRESSION]
[-d DATA_BLOCK_SIZE] [-H HCOMP_SCALE]
[-i INFILE]
[-J J2K_TARGET_COMPRESSION_RATIO]
[-K J2K_ALT_TARGET_COMPRESSION_RATIO]
[-l COMPRESSION_LEVEL]
[-m OUT_MASTER] [-N LAST_IMAGE] [-o OUT_FILE]
[-p THREADS] [-q OUT_SQUASH] [-s SUM_RANGE] [-S SCALE]
[-t [THREAD]] [-v] [-V]
```

Bin and sum images from a range and optionally apply JPEG-2000 or HCompress

Options

- −h, −−help show this help message and exit
- −1 FIRST_IMAGE, −−first_image FIRST_IMAGE first selected image counting from 1, defaults to 1
- −b BIN_RANGE, −−bin BIN_RANGE an integer image binning range (1 ...) to apply to each selected image, defaults to 1
- −c COMPRESSION, −−compression COMPRESSION optional compression, bslz4, bszstd, bshuf, defaults to zstd
- −d DATA_BLOCK_SIZE, −−data_block_size DATA_BLOCK_SIZE data block size in images for out_file, defaults to 100
- −H HCOMP_SCALE, −−Hcompress HCOMP_SCALE Hcompress scale compression, immediately followed by decompression
- −i INFILE, −−infile INFILE the input hdf5 file to read images from

Options

- −J J2K_TARGET_COMPRESSION_RATIO, −−J2K J2K_TARGET_COMPRESSION_RATIO JPEG-2000 target compression ratio, immediately followed by decompression
- −K J2K_ALT_TARGET_COMPRESSION_RATIO, −−J2K2 J2K_ALT_TARGET_COMPRESSION_RATIO JPEG-2000 target compression ratio, immediately followed by decompression
- −l COMPRESSION_LEVEL, −−compression_level COMPRESSION_LEVEL optional compression level for bszstd or zstd
- −m OUT_MASTER, −−out_master OUT_MASTER the output hdf5 master to which to write metadata
- −N LAST_IMAGE, −−last_image LAST_IMAGE last selected image counting from 1
- −o OUT_FILE, −−out_file OUT_FILE the output hdf5 data files out_file_?????? with a .h5 extension are files to which to write images
- −p THREADS, −−parallel THREADS the number of parallel threads with extra thread 0 used by itself to generate the new master file first

Options

- −q OUT_SQUASH, −−out_squash OUT_SQUASH the output hdf5 data file out_squash_?????? with an .h5 extension are optional files to which to write raw j2k or hcomp images
- −s SUM_RANGE, −−sum SUM_RANGE an integer image summing range (1 ...) to apply to the selected images
- −S SCALE, −−scale SCALE a non negative scaling factor to apply both to the images and satval
- −t [THREAD], −−thread [THREAD] the thread number for the action of the current invocation of l_bnl_compress, 0 to just make a new master file, otherwise between 1 and the number of datablocks/threads
- − [UINT], −−uint [UINT] clip the output above 0 and limit to 2 byte or 4 byte integers
- −v, −−verbose provide additional information
- −V, −−version report version and version_date

Notes

- The program reads a NeXus hdf5 file dataset following the Dectris FileWriter Eiger detector conventions and produces a new hdf5 file dataset following the same conventions.
- For the current release (1.0.1) the format allows direct processing by XDS. For fast_dp, conversion of the output to miniCBFs with eiger2cbf is necessary at present.

Installation Prerequisites

- This is a Python program that needs Python 3 running in a Linux-like environment. It was created on a Debian 12 system and tested on a RHEL8 environment. It is not likely to work with a Python version prior to 3.9. A suitable Python with many modules upon which l_bnl_compress.py depends can be provided by use of cctbx.python from the DIALS package.

- The modules upon which this program depends are:

- argparse " The argparse module makes it easy to write user friendly command line interfaces.

  "The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

  "As of Python $\geq$ 2.7 and $\geq$ 3.2, the argparse module is maintained within the Python standard library. For users who still need to support Python $<$ 2.7 or $<$ 3.2, it is also provided as a separate package, which tries to stay compatible with the module in the standard library, but also supports older Python versions." Python Software Foundation (2001)

Installation Prerequisites (continued)

- astropy “The astropy package contains key functionality and common tools needed for performing astronomy and astrophysics with Python. It is at the core of the Astropy Project, which aims to enable the community to develop a robust ecosystem of affiliated packages covering a broad range of needs for astronomical research, data processing, and data analysis.” The Astropy Collaboration *et al.* (2013) Astropy Collaboration *et al.* (2018) Astropy Collaboration *et al.* (2022)

- glymur “Glymur is an interface to the OpenJPEG library which allows one to read and write JPEG 2000 files from Python. Glymur supports both reading and writing of JPEG 2000 images. There was a historical limitation of glymur where it could not write images that did not fit into memory, but that limitation has been removed.

  “In regards to metadata, most JP2 boxes are properly interpreted. Certain optional JP2 boxes can also be written, including XML boxes and XMP UUIDs. There is incomplete support for reading JPX metadata.

  “The current version of glymur is supported on Python versions 3.10, 3.11, 3.12, and 3.13. You should have at least version 2.4.0 of OpenJPEG.

  “For more information about OpenJPEG, please consult `http://www.openjpeg.org`. ... ” Evans (2013)

Installation Prerequisites (continued)

- hdf5plugin “hdf5plugin provides HDF5 compression filters (namely: Blosc, Blosc2, BitShuffle, BZip2, FciDecomp, LZ4, Sperr, SZ, SZ3, Zfp, ZStd) and makes them usable from h5py.

  “Supported operating systems: Linux, Windows, macOS. Supported versions of Python: *geq*3.8 Supported architectures: All. Specific optimizations are available for x86 family, arm64 and ppc64le. hdf5plugin provides a generic way to enable the use of the provided HDF5 compression filters with h5py that can be installed via pip or conda.”
  European Synchrotron Radiation Facility (2016)

- h5py ‘The h5py package is a Pythonic interface to the HDF5 binary data format.

  “HDF5 lets you store huge amounts of numerical data, and easily manipulate that data from NumPy. For example, you can slice into multi-terabyte datasets stored on disk, as if they were real NumPy arrays. Thousands of datasets can be stored in a single file, categorized and tagged however you want.” Collette (2014)

- io “The io module provides Python’s main facilities for dealing with various types of I/O. There are three main types of I/O: text I/O, binary I/O and raw I/O. These are generic categories, and various backing stores can be used for each of them. A concrete object belonging to any of these categories is called a file object. Other common terms are stream and file-like object.”

Installation Prerequisites (continued)

- numpy “NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.” Numpy Team (2025) Harris *et al.* (2020)
- os (part of the base level Python installation)
- PIL “Pillow is the friendly PIL fork by Jeffrey A. Clark and contributors. PIL is the Python Imaging Library by Fredrik Lundh and contributors” Jeffrey A. Clark and contributors (2010)

Installation Prerequisites (continued)

- queue “The queue module implements multi-producer, multi-consumer queues. It is especially useful in threaded programming when information must be exchanged safely between multiple threads. The Queue class in this module implements all the required locking semantics.
  “The module implements three types of queue, which differ only in the order in which the entries are retrieved. In a FIFO queue, the first tasks added are the first retrieved. In a LIFO queue, the most recently added entry is the first retrieved (operating like a stack). With a priority queue, the entries are kept sorted (using the heapq module) and the lowest valued entry is retrieved first.”

- sys (part of the base level Python installation)

- skimage (scikit-image) “... is a collection of algorithms for image processing. It is available free of charge and free of restriction. ” scikit - image development team (2022) Van der Walt *et al.* (2014)

Installation Prerequisites (continued)

- string “The built-in string class provides the ability to do complex variable substitutions and value formatting via the format() method described in PEP 3101. The Formatter class in the string module allows you to create and customize your own string formatting behaviors using the same implementation as the built-in format() method.”
- subprocess “Subprocess management The subprocess module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes.”
- thread “Strictly type-safe and Wraps around the Python threading library and provides extra functionality Fully compatible with the threading library, this project hopes to provide a more out-of-the-box solution with multi-threaded processing and fetching values from a completed thread, etc.” Xiang (2024)

Installation Prerequisites (continued)

- threading “Thread-based parallelism. This module constructs higher-level threading interfaces on top of the lower level _thread module.”

- tifffile “ ... a Python library to store NumPy arrays in TIFF (Tagged Image File Format) files, and read image and metadata from TIFF-like files used in bioimaging. Image and metadata can be read from TIFF, BigTIFF, OME-TIFF, GeoTIFF, Adobe DNG, ZIF (Zoomable Image File Format), MetaMorph STK, Zeiss LSM, ImageJ hyperstack, Micro-Manager MMStack and NDTiff, SGI, NIHImage, Olympus FluoView and SIS, ScanImage, Molecular Dynamics GEL, Aperio SVS, Leica SCN, Roche BIF, PerkinElmer QPTIFF (QPI, PKI), Hamamatsu NDPI, Argos AVS, and Philips DP formatted files.

  “Image data can be read as NumPy arrays or Zarr 2 arrays/groups from strips, tiles, pages (IFDs), SubIFDs, higher-order series, and pyramidal levels.

  “Image data can be written to TIFF, BigTIFF, OME-TIFF, and ImageJ hyperstack compatible files in multi-page, volumetric, pyramidal, memory-mappable, tiled, predicted, or compressed form.” Gohlke (2025)

- time “Time access and conversions This module provides various time-related functions. For related functionality, see also the datetime and calendar modules.”

Installation Prerequisites (continued)

- warnings “Warning control. Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn’t warrant raising an exception and terminating the program. For example, one might want to issue a warning when a program uses an obsolete module.”

Installation Process

- Under recent Python 3 releases, the safest approach to installing modules is to use pip or pipx. For recent debian releases, only use of pipx is permitted Jonas Smedegaard (2025). In that case, assuming pipx has been installed at the Debian system level, in the following examples, running as a user, substitute pipx for pip. For example, in an older system, to install astropy, which is needed to support Hcompress we execute:

```
cctbx.python -m pip install astropy
Collecting astropy
  Downloading astropy-6.1.2-cp311-cp311-macosx_11_0_arm64.whl.metadata (10 kB)
Requirement already satisfied: numpy>=1.23 in

/opt/homebrew/lib/python3.11/site-packages (from astropy) (1.26.2)

Collecting pyerfa>=2.0.1.1 (from astropy)
  Downloading pyerfa-2.0.1.4-cp39-abi3-macosx_11_0_arm64.whl.metadata (5.7 kB)
```

Installation Process (continued)

Collecting astropy-iers-data>=0.2024.7.1.0.34.3 (from astropy)

Downloading astropy_iers_data-0.2024.8.5.0.32.23
-py3-none-any.whl.metadata (5.1 kB)

Requirement already satisfied: PyYAML>=3.13 in

/opt/homebrew/lib/python3.11/site-packages (from astropy) (6.0.1)

Requirement already satisfied: packaging>=19.0 in

/opt/homebrew/lib/python3.11/site-packages (from astropy) (23.2)

Downloading astropy-6.1.2-cp311-cp311-macosx_11_0_arm64.whl (6.4 MB)
...
Downloading astropy_iers_data -0.2024.8.5.0.32.23-py3-none-any.whl (1.9 MB) ...
Downloading pyerfa-2.0.1.4-cp39-abi3-macosx_11_0_arm64.whl (329 kB) ...

Installing collected packages: pyerfa, astropy-iers-data, astropy ...

Installation Process (continued)

For a recent Debian 12, on the other hand, working as a user, not root, we can install astropy in the DIALS version of Python 3.12 with:

```
cctbx.python -m pipx install astropy
  installed package astropy 7.0.1, installed using Python 3.12.9
  These apps are now globally available
    - fits2bitmap
    - fitscheck
    - fitsdiff
    - fitsheader
    - fitsinfo
    - samp_hub
    - showtable
    - volint
    - wcslint
done!
```

Installation Process (continued)

When using pip or pipx to install modules, it is important to install module versions that are consistent with the operating system, the Python version, and the hdf5 library version already used. Therefore, you should not replace any of the modules already installed with DIALS, but when installing modules that did not get installed with DIALS it may be necessary to install from source rather that from wheel binaries. For example to install h5py from source using pip:

Installation Process (continued)

cctbx.python -m pip install –no-binary=h5py h5py

One very reliable way to get a version of Python with many of the dependencies already installed is to start with a recent DIALS installation (see `https://dials.github.io/installation.html`, followed by the additional module installs with pip. Be warned that the process may well ask for additional dependencies. For example, when doing the install on a linux system in late February 2025, the necessary steps after downloading a DIALS installer in `dials-linux-x86_64-conda3.tar` were:

tar -xvf dials-linux-x86_64-conda3.tar
cd dials-installer-dev
./install –prefix=$HOME
cd $HOME/dials-dev20250221
. dials_env.sh

Installation Process (continued)

cctbx.python l_bnl_compress.py –help

to see what was missing next. It turned out to be scikit-image, so we ran

cctbx.python -m pip install scikit-image

That worked and we followed up with

```
cctbx.python l_bnl_compress.py --help
cctbx.python -m pip install astropy
cctbx.python l_bnl_compress.py --help
cctbx.python -m pip install glymur
cctbx.python l_bnl_compress.py --help
cctbx.python -m pip install numcodecs
cctbx.python l_bnl_compress.py --help
```

Acknowledgements

- Our thanks to Alexei Soares and Kimberly Horvat for their work on the research in Bernstein *et al.* (2024) on which this software package is based.

  We gratefully acknowledge thoughtful comments and suggestions by Robert M. Sweet and Frances C. Bernstein.

  The entire community owes a debt to James Holton for his creative, useful, and constructive comments and suggestions on the subject of lossy compression for macromolecular crystallography since 2011. He helped us all to break free of the past and look dispassionately at what modern realities require.

  We thank Dr. Karen Anderson for generously allowing us to use her HIV reverse transcriptase data to benchmark our compression algorithms, and for her comments regarding the ways that small changes in data quality could affect the interpretability of data in the vicinity of key areas of the protein that are biologically significant.

Funding

- This work utilized the AMX beamline from the Center for BioMolecular Structure (CBMS) which is primarily supported by the National Institutes of Health, National Institute of General Medical Sciences (NIGMS) through a Center Core P30 Grant (P30GM133893), and by the DOE Office of Biological and Environmental Research (KP1607011). As part of NSLS-II, a national user facility at Brookhaven National Laboratory, work performed at the CBMS is supported in part by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences Program under contract number DE-SC0012704. For H. Bernstein, this work was funded in part by Grant No. 1R24GM154040-01 from the NIH and the University of California Regents' Prime contract No. DE-AC02-05CH11231 from the Department of Energy for Lawrence Berkeley National Laboratory.

Conflicts of Interest

- There are no conflicts of interest to be declared.

Data Availability

- In addition to the data supporting Bernstein *et al.* (2024) in zenodo `https://zenodo.org/records/12701763` the code for this application and additional test data are available in github `http://github.com/nsls-ii-mx/l_bnl_compress`

References I

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L. *et al.* (2022). *Astrophysical J.* 935(2), 167.

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M. *et al.* (2018). *Astronomical J.* 156(3), 123.

Bernstein, H. J., Soares, A. S., Horvat, K. and Jakoncic, J. (2024). *J. Sync. Rad.* 32(2).

Collette, A. (2014). Hdf5 for python. https://docs.h5py.org/en/stable/.

European Synchrotron Radiation Facility (2016). hdf5plugin. http://www.silx.org/doc/hdf5plugin/latest/.

Evans, J. (2013). Glymur: a Python interface for JPEG 2000. https://glymur.readthedocs.io/en/latest/introduction.html.

Gohlke, C. (2025). tifffile, read and write tiff files. https://github.com/cgohlke/tifffile.

References II

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern1, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe1, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbas, H., Gohlke, C. and Oliphant, T. E. (2020). *Nature*, 585(7825), 357–362.

Jeffrey A. Clark and contributors (2010). Pillow is the friendly PIL fork. `https://pillow.readthedocs.io/en/stable/`.

Jonas Smedegaard (2025). Package: pipx (0.12.1.0-1) execute binaries from Python packages in isolated environments. `https://packages.debian.org/buster/pipx`.

Numpy Team (2025). NumPy documentatiom. `https://numpy.org/doc/stable/`.

Python Software Foundation (2001). argparse. `https://docs.python.org/3/library/argparse.html`.

References III

scikit - image development team (2022). scikit - image Image processing in Python. https://scikit-image.org.

The Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., , Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., Günther, H. M., Deil1, C., Woillez, J., Conseil, S., Kramer, R., Turner, J. E. H., Singer, L., Fox, R., Weaver, B. A., Zabalza, V., Edwards, Z. I., Bostroem, K. A., Burke, D. J., Casey, A. R., Crawford, S. M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P. L., Pierfederic, F., Pontzen, A., Ptak, A., Refsda, B., Servillat, M. and Streicher, O. (2013). *Astronomy & Astrophysics*, 558, A33.

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E. and Yu, T. (2014). *PeerJ*, 2, e453.

Xiang, J. (2024). thread – a python threading library extension. https://pypi.org/project/thread/.