Draft version November 7, 2025
Typeset using LaTeX **default** style in AASTeX7.0.1

Cutana: A High-Performance Tool for Astronomical Image Cutout Generation at Petabyte Scale

Pablo Gómez [1], Laslo Erik Ruhberg [2], Kristin Anett Remmelgas,[1] and David O'Ryan [1]

[1] *European Space Agency (ESA), European Space Astronomy Centre (ESAC), Camino Bajo del Castillo s/n, 28962, Villanueva de la Cañada, Madrid, Spain*

[2] *Astronomisches Rechen-Institut (ARI), Zentrum für Astronomie, Universität Heidelberg, Mönchhofstr. 12-14, 69120 Heidelberg, Germany*

ABSTRACT

The *Euclid* Quick Data Release 1 (Q1) encompasses 30 million sources across 63.1 square degrees, marking the beginning of petabyte-scale data delivery through Data Release 1 (DR1) and subsequent releases. Systematic exploitation of such datasets requires extracting millions of source-specific cutouts, yet standard tools like `Astropy`'s Cutout2D process sources individually, creating bottlenecks for large catalogues. We introduce `Cutana`, a memory-efficient software tool optimised for batch processing in both local and cloud-native environments. `Cutana` employs vectorised `NumPy` operations to extract cutout batches simultaneously from FITS tiles, implements automated memory-aware scheduling, and supports both Zarr and FITS output formats with multiple common normalisation schemes (asinh, log, zscale). `Cutana` outperforms `Astropy` in all tested Q1 subset scenarios achieving near linear scaling and processing thousands of cutouts per second. On just four worker threads, Cutana can process all of Q1 in under four hours. The tool includes an `ipywidget` interface for parameter configuration and real-time monitoring. Integration with ESA Datalabs is underway for the *Euclid* DR1 release, with open-source release[a)] pending ESA open-source licensing processes.

Keywords: Astronomy software (1855) — Astronomy data analysis (1858)

1. INTRODUCTION

Modern astronomical surveys are entering an unprecedented era of data generation. Telescopes like ESA's *Euclid* mission (Euclid Collaboration et al. 2025a) and the Vera C. Rubin Observatory (Ž. Ivezić et al. 2019) will produce petabyte-scale datasets of billions of sources. *Euclid*'s first Quick Data Release (Q1) already provides 63.1 square degrees of observations containing approximately 30 million objects (Euclid Collaboration et al. 2025b), foreshadowing the massive data volumes expected from the full survey. These vast archives harbour rare and scientifically valuable phenomena, as demonstrated by recent large-scale searches such as the identification of astrophysical anomalies in 99.6 million cutouts from the *Hubble* Legacy Archive using the `AnomalyMatch` method (D. O'Ryan & P. Gómez 2025).

Maximising scientific return from these datasets fundamentally relies on generating source-specific image cutouts for detailed analysis. However, the computational infrastructure required to process such volumes presents significant challenges. Modern computational platforms like ESA Datalabs provide Kubernetes clusters and computational grids optimised for large-scale processing, yet these environments require following high-performance computing practices such as careful memory management, and efficient compute utilisation, which traditional astronomical tools were not specifically designed to address.

The community's standard approach using `Astropy` processes sources individually through its Cutout2D implementation, creating substantial overhead when handling millions of objects. While `Astropy` provides robust world coordinate system handling and FITS manipulation, its single-source processing paradigm can introduce bottlenecks. This leaves topics such as parallelisation and memory management to the user. This typically involves custom scripts wrapping `Astropy` with parallelisation, yet these often encounter memory management challenges in constrained environments, particularly when processing large FITS files with multiple workers.

[a)] https://github.com/ESA/Cutana

Hence, we present `Cutana`, a purpose-built tool designed to address these limitations through vectorised, load-balanced batch processing while maintaining compatibility with astronomical data standards and formats. The software combines efficient algorithms with dynamic resource management, making it suitable for deployment in both traditional computing environments and modern containerised infrastructure. A streamlined graphical user interface enables rapid cutout creation.

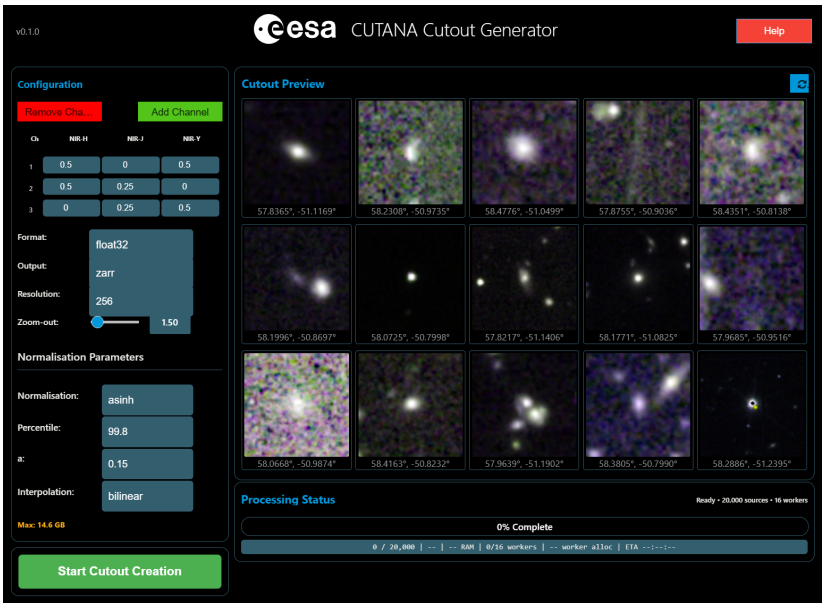

Figure 1. `Cutana`'s ipywidget-based interface within ESA Datalabs showing configuration panel (left) for source catalogue, output format, normalisation, and processing parameters, alongside real-time cutout preview grid (right) enabling validation of settings before large-scale processing on *Euclid* Q1 data.

2. IMPLEMENTATION

`Cutana` is implemented as a modular Python package with distinct components for data processing, I/O operations, and user interaction. The architecture separates the core batch processing engine from the graphical interface, enabling both interactive usage through Jupyter notebooks and automated execution in pipeline environments.

The backend employs a multi-stage pipeline optimised for astronomical image processing. The input source catalogues provide celestial coordinates (right ascension, declination), object sizes, and FITS file paths. Rather than processing sources individually, `Cutana` groups sources by their parent tiles, enabling efficient batch extraction through vectorised `NumPy` operations. The pipeline stages include: (1) catalogue validation and source grouping by tile, (2) tile-based batch processing with memory-mapped FITS access, (3) coordinate transformation and cutout extraction using optimised array slicing, (4) normalisation and format conversion, and (5) metadata generation and output packaging.

The system implements dynamic load balancing through continuous memory monitoring, adjusting worker processes based on available resources with a configurable safety margin (default 15%). This approach prevents memory exhaustion while maximising throughput, particularly critical in containerised environments with strict resource limits. `Cutana` supports both traditional FITS and the cloud-optimised Zarr format (A. Miles et al. 2025). Zarr provides efficient storage for multidimensional arrays through chunked and compressed storage that enables parallel I/O operations, making it ideal for cloud computing environments. Output data includes comprehensive metadata in parquet format. The metadata preserves source identifiers, coordinates, processing timestamps, and array indices, enabling rapid source retrieval from batch-processed archives.

The tool leverages two additional open-source packages developed by our team: `fitsbolt`[3] for high-performance FITS manipulation and normalisation operations, and `images-to-zarr`[4] for efficient conversion between image formats

[3] https://github.com/Lasloruhberg/fitsbolt/
[4] https://github.com/gomezzz/images_to_zarr/

Table 1. Framework Comparison: `Astropy` vs `Cutana` Performance

Tiles - Channels (i.e. FITS files) - Total # of sources	Tool (Threads / Workers)	Runtime (s)	Sources/sec
8 Tiles - 1 Channel - 200,000	`Astropy` (1 Thread)	659.9	303.1
8 Tiles - 1 Channel - 200,000	`Cutana` (1 Worker)	227.7	878.3
8 Tiles - 1 Channel - 200,000	`Astropy` (4 Threads)	626.1	319.4
8 Tiles - 1 Channel - 200,000	`Cutana` (4 Workers)	88.3	2264.0
32 Tiles - 4 Channel - 32,000	`Astropy` (1 Thread)	208.0	153.9
32 Tiles - 4 Channel - 32,000	`Cutana` (1 Worker)	529.2	60.5
32 Tiles - 4 Channel - 32,000	`Astropy` (4 Threads)	98.3	325.5
32 Tiles - 4 Channel - 32,000	`Cutana` (4 Workers)	69.6	460.0

and Zarr archives. These tools provide optimised implementations of critical operations, contributing to `Cutana`'s overall performance advantages.

The software incorporates multiple image normalisation methods essential for astronomical analysis, powered by the `fitsbolt` library. Available stretches include asinh, log, and zscale transformations, each with configurable parameters unified under a consistent interface. The system supports channel blending for multi-band observations, enabling users to combine multiple FITS extensions with custom weights for composite visualisation or analysis. These normalisation options can be previewed in real-time through the user interface before processing.

Cutout extraction implements a configurable padding factor controlling the extraction area relative to source size. With that factor, both zoom-ins (< 1), e.g. for crowded fields, as well as padding (> 1) , i.e. larger cutout areas, can be achieved. All cutouts undergo resampling to a user-specified target resolution (default 256×256 pixels) using configurable interpolation methods including bilinear, nearest-neighbour, cubic, and Lanczos resampling to enable efficient storage as large tensors.

3. RESULTS

We evaluated `Cutana`'s performance using *Euclid* Q1 data, generating 256×256 pixel cutouts from the MER catalogue. Testing was conducted on an Intel i5-13400F system with 64 GB DDR4 RAM to ensure reproducible comparisons with `Astropy`'s Cutout2D implementation. As `Astropy` lacks native parallelisation support, we implemented single-threaded comparisons to establish baseline performance, utilising the automated parallelisation of underlying libraries via thread pinning to either one or four cores. Table 1 presents comprehensive performance measurements across two distinct test configurations varying in tile count, FITS file complexity, and source catalogue size. The primary comparison (8 tiles, 1 FITS file, 200,000 sources) demonstrates that single-threaded `Cutana` execution achieves a 2.90× improvement over `Astropy`'s memory-mapped implementation, processing 878.3 cutouts per second compared to 303.1 for `Astropy`. With four workers, `Cutana` reaches 2264.0 cutouts per second, representing a 7.47× speed-up over the baseline single-threaded `Astropy` implementation.

The performance gains derive from three primary optimisations: (1) vectorised batch processing eliminating per-source overhead, (2) efficient memory management through tile-based grouping, and (3) optimised I/O patterns minimising disk access latency. The batch processing approach proves particularly effective for dense source catalogues where many cutouts originate from the same tile, as evidenced by the 8-tile configuration achieving the highest throughput rates. `Cutana` demonstrates scaling from single to multi-threaded execution, with a scaling factor of 2.58× when moving from one to four workers for the 8 Tile Scenario. This represents a parallel efficiency of 64.5%. The sub-linear scaling primarily results from I/O contention when multiple workers access the same FITS files simultaneously. This is noticeable as thread scaling efficiency is super-linear (7.6x sources/sec with workers) for the 32 Tiles Scenario. The two test configurations reveal how number of sources per tile affects processing throughput. The dense configuration (8 tiles, 1 FITS file, 200,000 sources) achieves 878.3 cutouts/second with single-worker `Cutana`, demonstrating optimal performance when many sources originate from few tiles. In contrast, the sparse configuration (32 tiles, 4 FITS files, 32,000 sources) demonstrates 60.5 cutouts/second, a reduction attributable to increased tile-loading overhead. This behaviour reveals that `Cutana` operates most efficiently when compute-bound rather than I/O-bound, whilst `Astropy` exhibits limited performance variation across these scenarios.

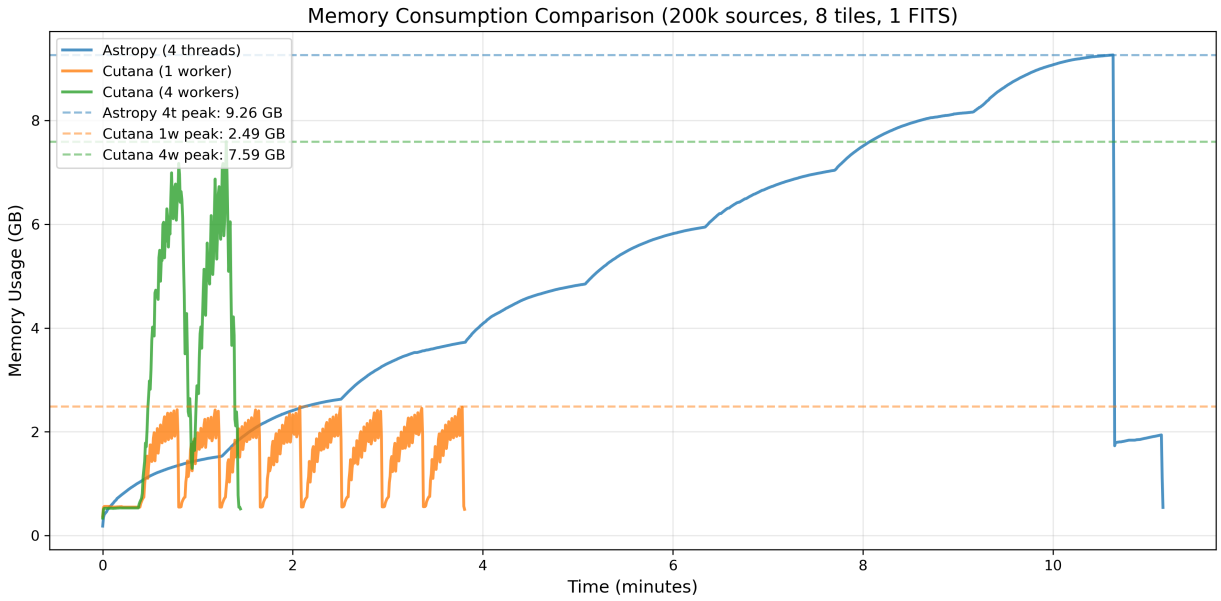

Figure 2. Memory consumption comparison processing 200,000 sources across 8 tiles. `Astropy` (blue) exhibits monotonic growth, whilst `Cutana` (orange: 1 worker, green: 4 workers) maintains bounded memory through tile-wise recycling. The sawtooth pattern reflects controlled loading cycles preventing exhaustion whilst maintaining throughput.

Figure 2 illustrates the critical distinction between `Cutana`'s dynamic memory management and `Astropy`'s linear accumulation pattern. When processing 200,000 sources with naive parallelisation, `Astropy` exhibits continuous memory growth, reaching 9.26 GB before completion. This behaviour stems from per-source processing accumulating intermediate arrays and Python objects without garbage collection opportunities between cutouts.

In contrast, `Cutana`'s tile-based architecture with worker process recycling maintains bounded memory consumption regardless of catalogue size. Single-worker execution peaks at 2.49 GB, whilst four-worker parallel processing reaches 7.59 GB. The characteristic sawtooth pattern in `Cutana`'s memory profile reflects deliberate batch processing cycles: each worker receives a job for ideally one tile (minimum and maximum numbers of sources can be specified to minimise overhead), processes all associated sources in batches, writes outputs, and terminates before a new one worker is spawned for the next job. This memory-efficient design proves essential for containerised deployments where resource limits are strictly enforced.

Peak memory usage of 2.49 GB per worker during tile loading phases, with average footprint of 1.8 GB during steady-state processing, enables predictable resource allocation. This controlled memory behaviour, combined with automatic management of number of active workers, prevents out-of-memory failures that commonly plague large-scale processing jobs whilst maintaining near-optimal throughput through intelligent load balancing.

4. CONCLUSION

`Cutana` addresses the critical need for scalable, memory-efficient cutout generation in the era of petabyte-scale astronomical surveys. By implementing batch processing, dynamic load balancing, and support for efficient data formats, the software tool enables efficient processing of massive datasets in state-of-the-art computational environments. The demonstrated performance improvements combined with robust memory management make `Cutana` suitable for integration into survey data processing pipelines, as evidenced by its deployment within ESA Datalabs for *Euclid* data releases.

As astronomical data volumes continue to grow, tools like `Cutana` become essential infrastructure for scientific discovery. By bridging the gap between traditional astronomical software and modern computational paradigms, we enable the community to fully exploit the scientific potential of current and future surveys, ultimately advancing our understanding of the cosmos through efficient data exploration and analysis.

ACKNOWLEDGMENTS

We thank the *Euclid* Consortium for providing access to Q1 data and valuable feedback during development. We acknowledge ESA Datalabs for computational resources and platform support. Software development and manuscript preparation made use of AI tools.

REFERENCES

Euclid Collaboration, Mellier, Y., Abdurro'uf, et al. 2025a, A&A, 697, A1, doi: 10.1051/0004-6361/202450810

Euclid Collaboration, Aussel, H., Tereno, I., et al. 2025b, Euclid Quick Data Release (Q1) – Data release overview, https://arxiv.org/abs/2503.15302

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, ApJ, 873, 111, doi: 10.3847/1538-4357/ab042c

Miles, A., jakirkham, Stansby, D., et al. 2025, zarr-developers/zarr-python: v3.1.3, v3.1.3 Zenodo, doi: 10.5281/zenodo.17155639

O'Ryan, D., & Gómez, P. 2025, arXiv e-prints, arXiv:2505.03508, doi: 10.48550/arXiv.2505.03508