A Reproducible Pipeline for JunoCam Limb Analysis and the Detection of Detached Jovian Hazes as a Function of Planetocentric Latitude

Author: *Kevin J. Kelly, MFA*
Affiliation: *PCC, ASU, NASA Jet Propulsion Laboratory, Planetary Science Group #322*
Team Lead: *Dr. Glenn Orton*
Date: January 2026

Abstract

We present a fully reproducible data-processing pipeline for the detection and classification of detached haze layers in Jupiter's atmosphere using images from the *Juno* spacecraft's visible-light camera, *JunoCam*. The system automates every stage from raw image ingestion to scientific analysis, enabling other researchers to replicate and extend the work. By integrating the USGS ISIS3 software and NASA's SPICE toolkit through Python wrappers, the pipeline converts raw .IMG files into geolocated image cubes, traces the planetary limb, rectifies the limb geometry, extracts perpendicular brightness profiles, and classifies detached haze layers. All code, parameters, and software versions are archived and version-pinned to ensure complete reproducibility.

Applying a brightness-gradient–based peak detection algorithm to limb profiles from a single JunoCam image, we find that detached haze is not uniformly present along the limb. Instead, its occurrence increases sharply toward higher planetocentric latitudes, reaching near-continuous presence above approximately 31–33°.

Introduction

Understanding Jupiter's upper-atmospheric haze structure is critical for constraining photochemical and dynamical processes within the planet's stratosphere. *JunoCam*, aboard NASA's *Juno* mission, provides high-resolution color imaging that captures the visible limb of Jupiter across multiple perijoves. Variations in haze brightness and apparent altitude along this limb reveal the distribution and dynamics of aerosols suspended above the main cloud deck.

Previous work (Kelly 2022) established a semi-manual procedure to measure the Jovian limb, derive latitude–longitude positions through ISIS + SPICE, and identify detached haze layers through visual inspection of brightness profiles. The present work extends that foundation by engineering a **modular, reproducible pipeline** that performs each step programmatically, producing verifiable scientific outputs and a clear methodological record suitable for independent validation.

Objectives

1. To create an end-to-end, reproducible workflow for JunoCam limb analysis using open-source software.

2. To automate haze detection and classification using pixel-intensity cross-sections perpendicular to the Jovian limb.
3. To provide a reusable research template for planetary-image pipelines adhering to open-science and reproducibility principles.
4. To determine whether the occurrence, altitude, or optical strength of detached haze layers varies as a function of planetocentric latitude, and to assess potential correlations with zonal or meridional atmospheric dynamics.
5. To translate pixel-scale haze measurements into physical distances (slant distance and altitude above the 1-bar reference level), enabling quantitative comparisons with previous studies and potential compositional modeling.

Methods Overview

All processing and analysis stages are executed on a per-IMG basis. For a given JunoCam image, all framelets, limb fragments, diagnostic plots, and statistical summaries are generated within IMG-specific directories. This design ensures full reproducibility, prevents output collisions when processing multiple images, and allows cross-image aggregation to be performed as a distinct analytical step outside the scope of this study.

The haze classification pipeline consists of a sequence of modular Python stages, each of which consumes the outputs of the preceding stage while remaining independently executable for diagnostic or exploratory analysis.

Stage	Module	Description	Primary Output
01	framelets_and_spice.py	Ingests raw JunoCam IMG and LBL products, converts them to ISIS-compatible .CUB files, and attaches full SPICE geometry for subsequent analysis.	SPICE-initialized .CUB files
02	trace_limb_polyline.py	Detects and traces the planetary limb using gradient-based edge detection, producing continuous polyline representation of Jupiter's limb curvature.	Limb coordinate sets (.csv), visual verification (.png)
03	rectify_limb.py	Rectifies limb segments into a linearized reference frame and extracts perpendicular brightness cross-sections spanning the atmospheric limb.	Rectified limb images (.tiff) and cross-section data (.csv)
04	plot_perpendiculars.py	Computes brightness gradients along perpendicular limb fragment cross-sections and prepares intermediate diagnostic outputs.	Intermediate diagnostic plots and exploratory analysis outputs (.csv)
05	graph_profiles.py	Generates diagnostic plots of rectified brightness profiles.	Diagnostic profile graphs (.png) for visual verification
06	peakfinder.py	Identifies primary limb peaks and secondary detached-haze brightness gradients along rectified limb profiles.	Annotated limb-fragment classification tables (.csv) with peak locations, amplitudes, and haze flags
07	haze_analysis.py	Aggregates fragment-level classification across all framelets and bins detections by first-available SPICE latitude to quantify detached haze as a function of latitude	Latitude-binned occurrence tables (.csv) and summary analysis figures (.png)

Note well: All terminal commands are executed from the project's root directory.

Stage 0 — Computational Environment Setup

Planetary image processing research must be reproducible at the binary, numerical, and geometric-kernel level. The JunoCam haze-layer pipeline presented in this paper is executed entirely within a controlled Linux environment using the USGS Integrated Software for Imagers and Spectrometers (ISIS3). To achieve true reproducibility, all software, all SPICE kernels, and all environment variables are explicitly defined here.

ISIS does not run natively on Windows, and attempting to do so leads to inconsistent geometry, missing libraries, and broken path resolution. Therefore, all Windows systems must run ISIS inside a **WSL2** (Windows Subsystem for Linux 2) Ubuntu environment. Native Linux installations are fully supported.

The steps below describe the exact procedure used in this study to create a stable, deterministic environment for all analysis stages — from raw IMG/LBL ingestion to limb tracing, normal construction, brightness-profile extraction, and haze-layer rectification.

0.1 Operating System Requirements

This pipeline requires:

- Linux distributions. We use Debian 13, Ubuntu 22.04 and 24.04 LTS, running either:
    - natively on Linux, or
    - inside WSL2 on Windows 11
- WSL1 is not supported (no FUSE, no Linux kernel, broken symlink semantics, and missing system calls)
- At the time of publication, Windows-native execution of ISIS3 is impossible (ISIS relies on GLIBC, POSIX filesystem semantics, symbolic links, and Linux-style shared objects)

This paper assumes the user is operating inside a WSL2 Ubuntu terminal or native Linux shell.

0.2 Installing Miniforge and Mamba

ISIS builds distributed by USGS are optimized for the **conda-forge** ecosystem. The reproducible environment begins by installing *Miniforge*, a minimal conda-forge-only distribution.

Download Miniforge (Linux x86_64):

```
wget https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh
```

Install interactively:

```
bash Miniforge3-Linux-x86_64.sh
```

Load conda into the current shell:

```
source ~/.bashrc
```

The environment must use **mamba**, the accelerated package solver:

```
mamba --version
```

If mamba is not available, install it:

```
conda install mamba -n base -c conda-forge
```

0.3 Creating the ISIS Environment

A dedicated environment isolates all versions and prevents conflicts downstream:

```
mamba create -n isis-8.3.0 python=3.11
mamba activate isis-8.3.0
```

After successfully creating your ISIS environment, install ISIS3 from the official USGS channel:

```
mamba install -c usgs-astrogeology isis
```

This places ISIS under:

```
$CONDA_PREFIX/isis
```

and automatically defines the required environment variables (ISISROOT, ISISDATA) on activation.

> ***Note:*** *Some ISIS3 builds do not ship the legacy isisversion binary.*
> *This is expected and not an error. Validation is instead performed via the command spiceinit -help and the presence of the full ISISROOT tree.*

0.4 Ensuring Correct Environment Variables

On each activation use the following commands:

```
conda activate isis-8.3.0
echo $ISISROOT
echo $ISISDATA
```

Expected:

```
ISISROOT = /home/<user>/miniforge3/envs/isis-8.3.0
ISISDATA = /home/<user>/.Isis/isis_data
```

To verify the successful installation run:

```
ls "$ISISROOT/scripts" | head
ls "$ISISROOT/lib" | head
```

Presence of downloadIsisData, ISIS plugin libraries, and instrument models confirms a correct install.

0.5 Downloading Offline SPICE Kernels (Required for Reproducibility)

This pipeline intentionally avoids reliance on remote NAIF servers or the USGS ALE SPICE service. Instead, all kernels are downloaded locally using the official ISIS data

downloader. To install these kernels, run the following commands in your activated conda environment:

```
downloadIsisData base "$ISISDATA"
downloadIsisData juno "$ISISDATA"
```

Approximate sizes at the time of publication:

- **base:** ~27 GB
- **juno:** ~46 GB

After downloading the kernels, these values can be verified with:

```
du -sh "$ISISDATA/base"
du -sh "$ISISDATA/juno"
```

Directory structure should contain:

```
~/.Isis/isis_data/base/kernels/
~/.Isis/isis_data/juno/kernels/
```

While you can download these kernels using NAIF's online server, using local kernels guarantees:

- deterministic geometry
- identical limb solutions
- reproducible normals
- stable haze-layer sampling
- replicable results years later regardless of NAIF server changes

These are the core, central reproducibility features of our classification system.

0.6 Verification Tests

Downloading these kernels will take time. After successfully downloading the required kernels, run the following commands to verify that ISIS can initialize geometry and resolve kernels:

```
spiceinit -help
```

Expected output includes the full set of SPICE parameters and server URL:

```
URL = https://astrogeology.usgs.gov/apis/ale/v0.9.1/spiceserver/
```

Next, test the file tree:

```
ls "$ISISDATA"
```

Expected:

```
base
juno
```

Test the ISISROOT path:

```
ls "$ISISROOT"
```

Expected: full ISIS tree (bin, lib, scripts, plugins, etc.)

> ***Note:*** *As mentioned above, some distributions do not include isisversion. This does not affect the correctness or completeness of the installation.*

0.7 Environment Lock for Archival Reproduction

To guarantee future identical reconstruction, you can lock your environment with the following command:

```
conda env export --no-builds > isis_environment.lock.yml
```

This environment lockfile is included in the project repository and ensures that any researcher can reproduce:

- identical ISIS binaries
- identical library versions
- identical SPICE geometry
- identical haze-layer extraction behavior

With the operating system, conda environment, ISIS installation, and SPICE kernel archive now fully configured, the computational environment is complete and reproducible. Every remaining stage of the pipeline—image ingestion, limb tracing, geometric rectification, normal construction, and haze-layer analysis—depends on this foundation. From this point forward, all commands operate inside the activated isis-8.3.0 environment, guaranteeing that every result presented in this paper can be reproduced exactly, on any machine, using the same software versions and the same local kernel set. Having established this controlled environment, we now proceed to Stage 1: ingesting raw JunoCam images and attaching the full SPICE geometry required for scientific analysis.

Stage 1 — Data Ingestion and SPICE Initialization

1.1 Data Acquisition from the Planetary Data System

Raw JunoCam images are publicly available through NASA's **Planetary Data System (PDS) Atmospheres Node**, which hosts perijove datasets under the *Juno Archive Page* at https://pds-atmospheres.nmsu.edu/. For users unfamiliar with the directory structure, the procedure for locating and downloading JunoCam .IMG and corresponding .LBL files is described in detail by Brueshaber (2021). In summary:

1. Navigate to the PDS Atmospheres Node → Juno Archive Page → Color Camera → Online Data Volumes→ JunoCam.
2. Choose the desired orbit, then follow the path: DATA/RDR/JUPITER/ORBIT_XX/ to view available .IMG and .LBL pairs.

3. For automated download, use wget with recursion disabled beyond the orbit directory. Example:

```
wget -r -np -A "JNCR*"
https://pds-imaging.jpl.nasa.gov/data/juno/JNOJNC_0012/DATA/RDR/JUPITER/ORBIT_14/
```

   The specific orbit directory here corresponds to Perijove 14; users substituting other perijoves should update the ORBIT_XX path accordingly

4. Ensure that for every .IMG file, its matching .LBL file is also downloaded, as the label contains spacecraft position, sub-spacecraft latitude/longitude, and timing metadata required by ISIS3 for ingestion.

Following Brueshaber (2021), datasets may be organized by orbit and framelet; however, in this implementation, all raw .IMG and .LBL files are stored in the directory /juno/data/raw/ for simplicity. Each image–label pair can be processed independently through the pipeline.

In this demonstration, we apply the pipeline to *JunoCam* image **JNCR_2018197_14C00024_V01.IMG** and its corresponding .LBL label file, acquired during Perijove 14 on July 16 2018. This dataset captures Jupiter's northern limb at a sub-spacecraft latitude of approximately 13.9° N and serves as the working example throughout the subsequent stages of the pipeline.

Users may substitute other perijove datasets following the same retrieval procedure described above; however, the example image is retained here for clarity and reproducibility.

1.2 Ingestion into ISIS

Once the raw data are available locally, each .IMG file is converted into an ISIS-compatible .CUB file, and SPICE geometry is attached to the newly formatted image.

The ingestion process is now performed by running `01.framelets_and_spice.py`, which ingests each JunoCam .IMG/.LBL pair and breaks each .IMG into its corresponding .CUB framelets, attaching SPICE geometry to each framelet using locally cached kernels, and records a clean manifest for reproducibility. Each manifest file documents every kernel attached to that cube, expanding $ISISDATA variables into full absolute paths (e.g., /miniconda/envs/juno-isis/isisdata/juno/kernels/spk/juno_rec_180620_180812_180821.bsp).

Once all cubes have been SPICE-initialized and their manifests generated, the system proceeds to the limb-tracing stage. To execute, type:

```
python src/01.framelets_and_spice.py \
  --img data/JNCR_2018197_14C00024_V01/raw/JNCR_2018197_14C00024_V01.IMG \
  --outdir data/JNCR_2018197_14C00024_V01/cub/stage_01_framelets
```

All outputs are written to IMG-scoped directories under `data/<IMG_NAME>/`, ensuring that multiple images can be processed independently without file collisions.

Stage 2 — Limb Detection and Tracing

With SPICE geometry successfully attached, the next stage identifies Jupiter's planetary limb in each JunoCam framelet. The module *02.trace_limb_polyline.py* combines SPICE-derived camera geometry with image-based edge detection to define the precise boundary between the planet's atmosphere and surrounding space.

We trace the two-dimensional planetary limb — the boundary separating Jupiter's illuminated atmosphere from the blackness of space — by computing brightness gradients in grayscale intensity between neighboring pixels. The resulting limb, represented as a dense set of subpixel coordinates (referred to as a *polyline*), captures the precise geometric curvature of the planet's visible edge. This polyline is then fitted with a smooth polynomial curve that can be rectified in later stages for quantitative haze-layer analysis. Collectively, these limb traces establish the geometric foundation for all subsequent measurements of atmospheric scattering height and curvature.

Each SPICE-attached framelet (.cub) is exported as a 16-bit TIFF (science fidelity) and an 8-bit, linearly stretched PNG (for quick-look reference). The gradient-based search seeds a starting limb-point and iteratively proposes subsequent points along the edge by maximizing brightness differential across a small angular neighborhood. The resulting (x,y) pixel coordinates of the polyline are saved as `<cub>_LIMBENDPOINTS.csv`, while an overlay with cyan limb points is saved as `<cub>_OVERLAY.png` in the same directory. Additionally, batch processing accepts a file glob of .CUB inputs, enabling full-image coverage.

The seeding step no longer assumes a fixed limb location. Instead, the algorithm identifies the global maximum of the Sobel gradient magnitude within each framelet, providing orbit-agnostic initialization and ensuring compatibility with any JunoCam observation geometry. This update replaces the earlier orientation-specific heuristic (cf. Wen, 2021) and improves cross-orbit generalization, detecting both upper and lower limb fragments even in oblique lighting geometries.

2.1 Processing Scope and Assumptions

All limb tracing and rectification steps are performed on a per-framelet basis under fixed geometric assumptions. Sampling density, smoothing parameters, and perpendicular construction are held constant across all framelets within a given IMG to ensure comparability. No framelet-level tuning is performed, and all parameters are documented explicitly to support reproducibility.

2.2 Execution Example

```
python src/02.trace_limb_polyline.py \
  --cub-dir data/JNCR_2018197_14C00024_V01/cub/stage_01_framelets \
  --outdir  data/JNCR_2018197_14C00024_V01/cub/stage_02_traces
```

Artifacts produced in this stage consist of SPICE-referenced limb polyline coordinate sets and verification overlays, which constitute the sole inputs to the rectification stage and are written to IMG-scoped output directories for reproducibility.

2.3 Limb Detection Results

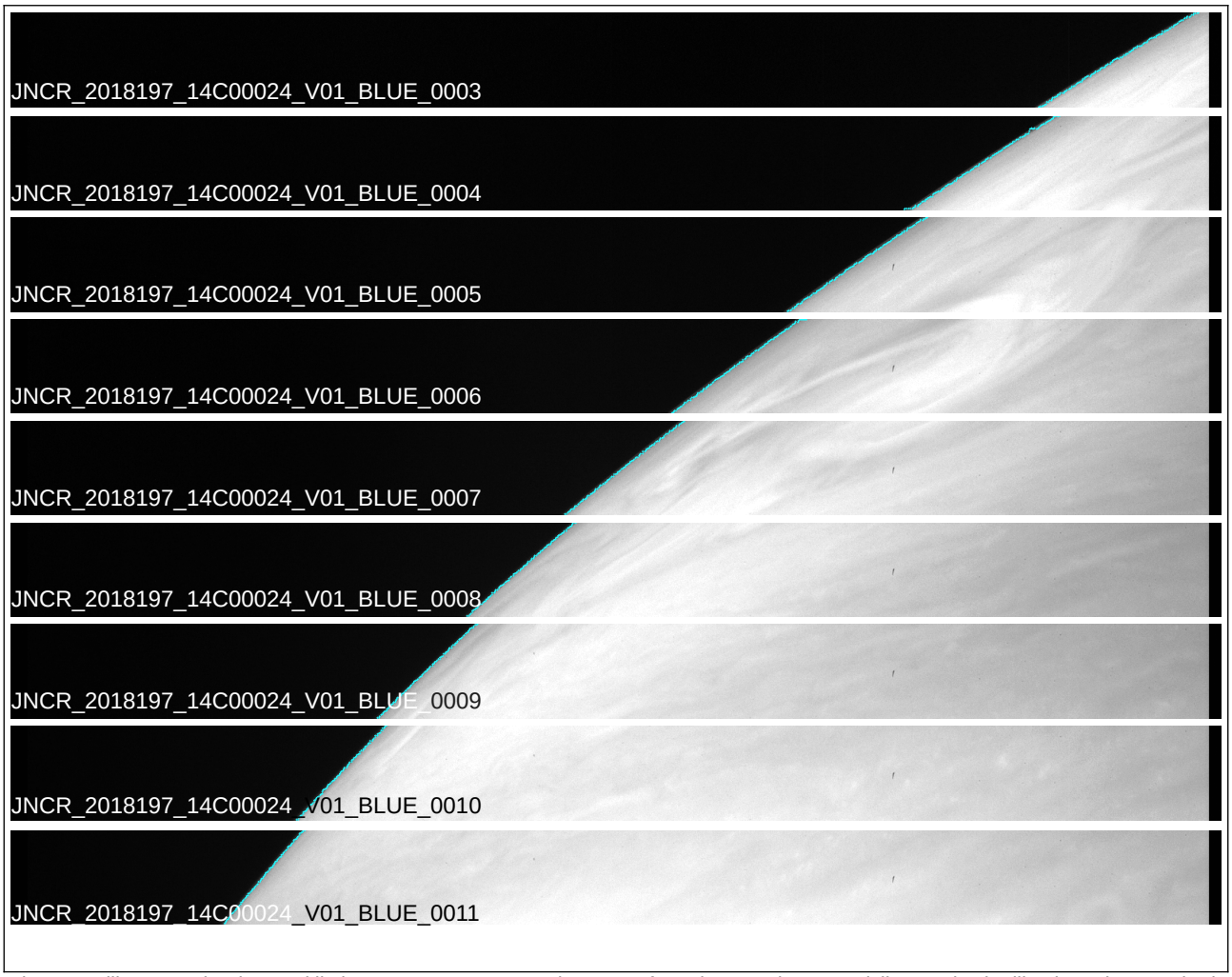


Figure 2.2 illustrates the detected limb traces across successive BLUE framelets. Each cyan polyline marks the illuminated atmospheric boundary. Consistency across framelets confirms both algorithmic stability and photometric reliability.

Each run outputs a corresponding .TIF, .PNG, *_OVERLAY.png*, and *_LIMBENDPOINTS.csv* file for every processed framelet. This updated version of the limb-tracing module marks a major improvement over prior implementations that only detected the *upper* limb. By enabling bidirectional tracing from an adaptive gradient seed, the algorithm successfully identifies both top *and* bottom limb segments, producing accurate overlays across a wide range of viewing geometries. In cases where Jupiter's disc fills the entire frame (no visible limb-space), the algorithm may generate false positives on internal gradients. These are easily identified in post-processing (see figure 2.3, below) as subsequent tangent-normal sampling yields non-null results across the entire image—a clear indicator that no limb boundary exists. Future iterations may integrate a null-value rejection filter to automatically exclude such false detections.

Figure 2.3 — False Positive: JNCR_2018197_14C00024_V01_BLUE_0028

Stage 3 — Limb Rectification and Perpendicular Sampling Geometry

The limb traces produced in Stage 2 define the precise two-dimensional boundary between Jupiter’s illuminated atmosphere and surrounding space. However, the curvature of this boundary varies across each framelet, making direct comparison of brightness profiles difficult in raw image coordinates.

Stage 3 transforms each detected limb segment into a limb-aligned coordinate system in which the curved atmospheric boundary is straightened and placed into a common reference frame. This rectification enables consistent, repeatable sampling of brightness variations perpendicular to the limb, corresponding to changes in scattering with altitude.

3.1 Limb-Aligned Rectification

Each framelet’s limb polyline is resampled at regular intervals and locally approximated as a smooth curve. Using this curve, the original image is re-projected into a rectified coordinate system in which the limb appears as a straight horizontal feature (Figure 3.1). In this rectified image, horizontal position corresponds to distance along the limb, while vertical position corresponds to distance normal to the atmospheric boundary.

Rectification is performed independently for each framelet and restricted to the stable limb region, excluding end segments where the tangent orientation becomes ill-defined near image boundaries. This resulting rectified product is a narrow strip centered on the limb that preserves the local photometric structure across the atmosphere–space interface.

The limb-aligned rectification strategy employed here follows the general methodology described by Eichstädt (2020), who introduced image-space rectification techniques for analyzing Jupiter’s limb structure in JunoCam observations. We have adapted this process for fully automated batch processing and reversible coordinate mapping.

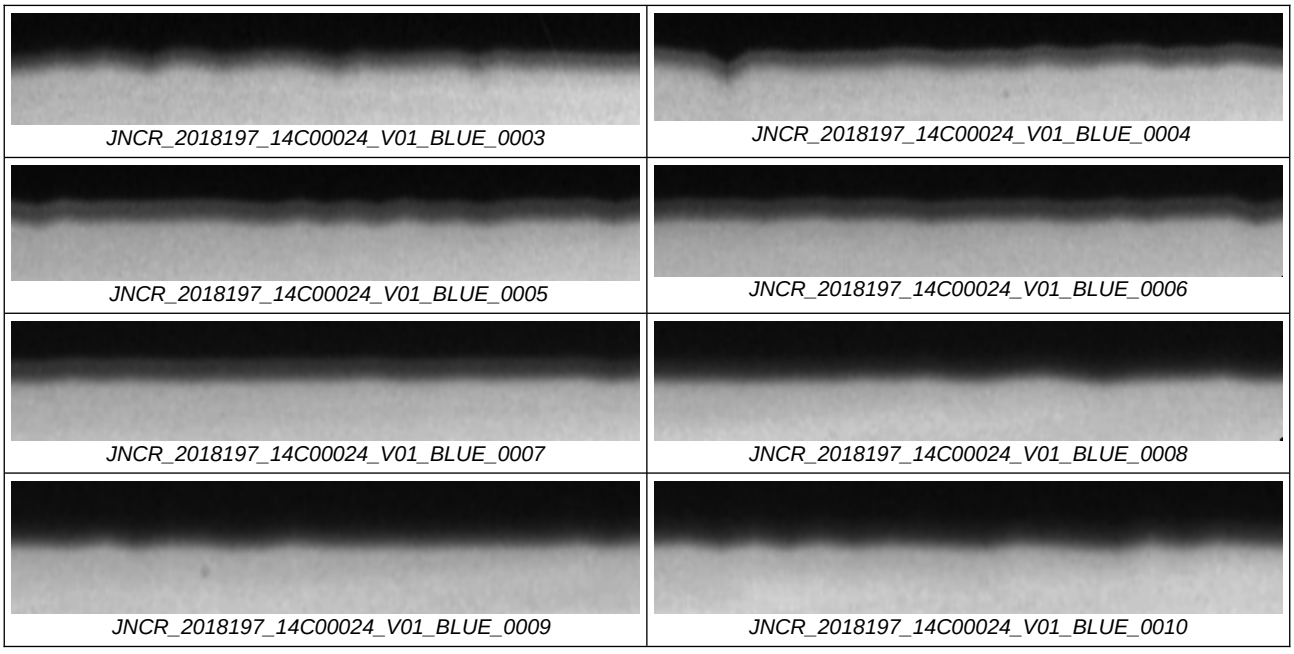


Figure 3.1 illustrates the geometric rectification process that transforms the curved limb into a flat, horizontally aligned reference frame, isolating the usable limb segment for brightness-altitude analysis.

3.2 Reversible Mapping Between Rectified & Original Coordinates

The rectification process is implemented as a fully reversible geometric transformation. For every pixel in the rectified image, the corresponding pixel coordinates in the original ISIS-calibrated framelet are recorded and stored in an explicit mapping table.

This rectified-to-original coordinate map ensures that any measurement performed in rectified space can be traced directly back to the original image pixels and associated SPICE geometry. As a result, no scientific interpretation relies solely on transformed imagery; all derived quantities remain physically anchored to the source observation.

For each framelet, Stage 3 produces:

- a 16-bit rectified limb image (*_RECTIFIED.tif) used for all quantitative sampling,
- an 8-bit visualization image (*_RECTIFIED.png) for diagnostic and publication figures, and
- a dense rectified-to-original coordinate mapping table (*_RECTIFIED_MAPPING.csv).

These products form the terminal outputs of Stage 3 and serve as the sole inputs to subsequent brightness-profile analysis.

3.3 Perpendicular Brightness Sampling Geometry

With the limb geometry straightened, sampling paths perpendicular to the atmospheric boundary are defined directly in rectified space. Sampling lines are placed at uniform horizontal intervals along the rectified limb and extend vertically across the limb transition, probing from the dark background of space into Jupiter’s upper atmosphere.

For each sampling position, two limb edges are identified in rectified coordinates, and their midpoint defines the local limb’s centerline. From this midpoint, a vertical sampling path is constructed that spans a fixed number of pixels both outward into space and inward toward the planet. These rectified limb fragments are parallel, evenly spaced, and orthogonal to the limb by construction, independent of the original viewing geometry.

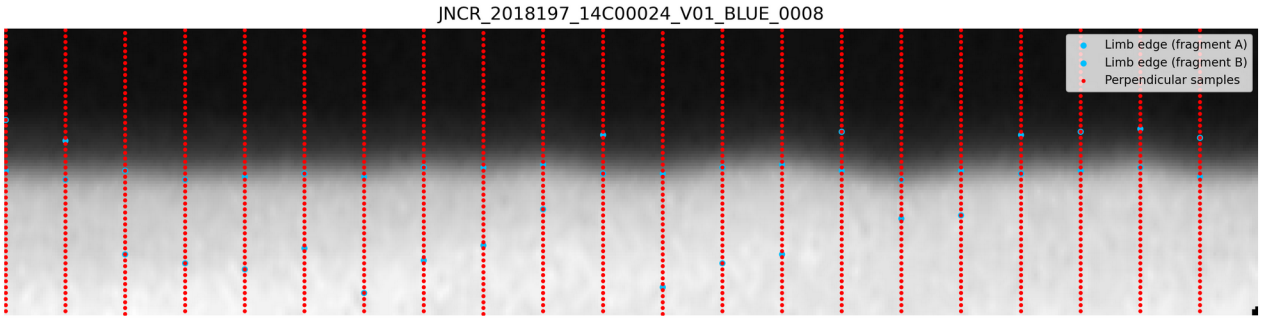


Figure 3.3. Rectified limb image with detected limb edges (blue) and perpendicular brightness sampling paths (red).

Each sampled rectified pixel is mapped back to its original framelet coordinates using the rectified-to-original mapping table, and its corresponding 16-bit pixel value is retrieved from the ISIS-calibrated TIFF image. For each framelet, a single consolidated table is produced containing rectified coordinates, original image coordinates, and pixel intensity values for every sampled location.

In addition to these numerical outputs, a diagnostic overlay image is generated for each rectified framelet (Figure 3.3), displaying the rectified limb strip with detected limb edges and perpendicular sampling paths superimposed. These overlays serve both as validation artifacts and as publication figures illustrating the sampling geometry.

3.4 Association of Perpendicular Samples with SPICE Geometry

To associate each sampled pixel with physical planetary coordinates, spacecraft geometry is retrieved using the SPICE kernel system through the ISIS `campt` tool. For each sampled pixel location, SPICE is queried using the original unrectified image geometry, ensuring that physical coordinates are derived directly from calibrated spacecraft pointing and timing information.

For every sampling point along each perpendicular brightness profile, ancillary metadata—including planetocentric latitude, planetocentric longitude, spacecraft altitude, emission angle, incidence angle, phase angle, and slant distance—are retrieved and appended to the sampling records. These parameters are stored alongside the pixel brightness values without altering the sampling order established in rectified space.

This approach decouples geometric sampling from physical interpretation: brightness profiles are constructed and ordered according to their limb-normal distance in rectified image coordinates, while each sample is independently associated with its original corresponding physical location on Jupiter. As a result, brightness-gradient analysis and peak detection are performed in a stable geometric coordinate system, while all reported results can be expressed in physically meaningful planetary coordinates.

At the conclusion of Stage 3, each valid perpendicular brightness profile consists of a sequence of pixel samples ordered by limb-normal distance and annotated with full SPICE-derived physical geometry. These profiles provide the foundation for brightness-gradient analysis, detached haze detection, and subsequent statistical analysis as a function of planetocentric latitude. They constitute the sole inputs to downstream diagnostic and classification stages and are written to IMG-scoped output directories for reproducibility

3.5 Execution Example

```
python src/03.rectify_limb.py \
  --indir data/JNCR_2018197_14C00024_V01/cub/stage_02_trace_polyline \
  --imagedir data/JNCR_2018197_14C00024_V01/cub/stage_01_framelets \
  --outdir data/JNCR_2018197_14C00024_V01/cub/stage_03_rectified
```

Stage 4 — Brightness-Gradient Visualization & Analysis

Stage 4 constitutes the core analytical step of the Juno Haze Classification System, transforming geometrically rectified, SPICE-annotated limb samples into physically interpretable diagnostics for the detection and classification of detached haze layers. At this stage, the pipeline transitions from purely geometric processing to scientific

interpretation, while remaining fully algorithmic, reproducible, and conservative with respect to geometric limitations near the limb.

This stage replaces earlier visualization-first workflows with a unified, physics-aware peak-detection system implemented in a single module (`06.peakfinder.py`). Diagnostic plots are generated directly from the same logic that performs classification, ensuring that all visualizations reflect the true decision path of the algorithm.

4.1 Construction of Limb-Normal Brightness-Gradient Profiles

For each rectified limb fragment produced in Stage 3, brightness is sampled along paths oriented perpendicular to the limb, yielding one-dimensional brightness profiles ordered by limb-normal distance (dy). These profiles probe the vertical structure of Jupiter's upper atmosphere, extending from optically thin space, across the tangent limb, and into the upper cloud deck.

Raw pixel intensities are extracted from calibrated ISIS products and assembled into ordered brightness sequences. To suppress slowly varying background illumination and emphasize vertical structure, first-order brightness gradients (ΔBrightness) are computed along each profile. This transformation sharpens transitions associated with atmospheric boundaries and highlights secondary scattering layers above the cloud deck.

Internally, all profiles are constructed and analyzed strictly in dy-ordered space. This preserves uniform sampling and avoids ambiguities introduced by nonuniform physical spacing in latitude or slant distance. Where available, SPICE-derived physical coordinates (planetocentric latitude, longitude, slant distance, emission angle) are attached to each sampling point without altering the dy-based ordering.

For visualization purposes only, profiles may be plotted with the horizontal axis labeled in planetocentric latitude. Importantly, this relabeling does not alter the underlying index-space ordering used for peak detection or classification. The plotted latitude axis therefore serves as a physical annotation rather than a computational coordinate.

4.2 Smoothing and Noise Suppression

Brightness-gradient profiles often exhibit pixel-scale noise arising from photon statistics, local albedo variations, and residual detector artifacts. To suppress this noise while preserving physically meaningful extrema, all ΔBrightness profiles are smoothed using a Savitzky–Golay filter (quadratic polynomial, window size of 5 samples).

This window size represents the minimum smoothing scale capable of removing high-frequency noise without suppressing secondary maxima or shoulder-like features associated with detached haze layers. The smoothing parameters are fixed and applied uniformly across all profiles. No adaptive or profile-specific tuning is performed.

Both raw and smoothed profiles are retained. Raw profiles are preserved for archival integrity, sanity checks, and archival transparency, while smoothed profiles are used exclusively for peak detection and classification.

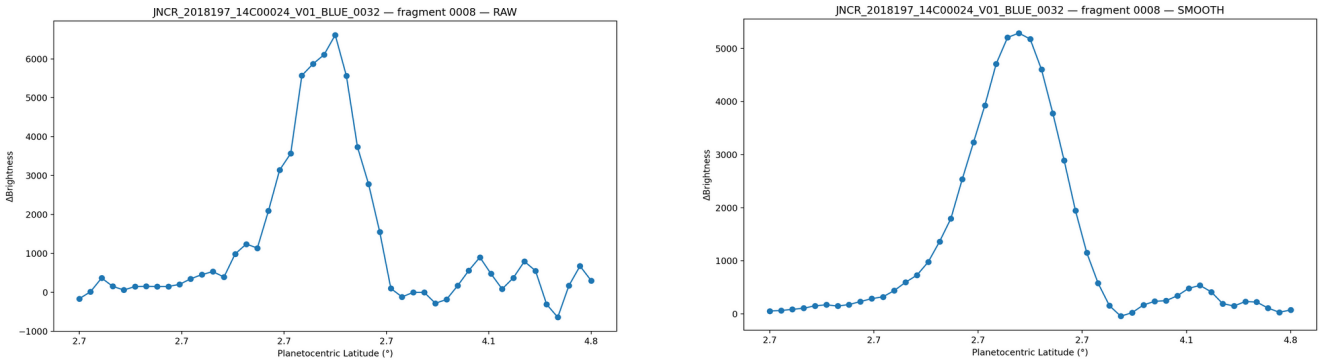


Figure 4.2 — Raw and Smoothed Limb-Normal ΔBrightness Profiles Illustrating Noise Suppression and Feature Preservation along the Jovian limb

4.3 Peak Detection Logic and Physical Constraints

Automated peak detection is performed on the smoothed ΔBrightness(dy) profiles using a constrained local-extrema search. The algorithm identifies candidate local maxima and classifies them according to physically motivated rules rather than purely mathematical prominence.

The dominant limb peak—corresponding to the sharp brightness transition at the atmospheric boundary—is identified as the **primary** peak. Candidate secondary peaks are only evaluated as potential detached haze signatures if they satisfy the following constraints:

1. **Inboard Requirement**: Detached haze features must occur interior to the primary limb peak. Any extrema occurring at dy ≥ dy_primary are excluded. This is a known and defensible geometric limitation inherent to limb-normal sampling and is consistent with the physical expectation that detached hazes reside above the cloud deck but below the tangent limb.
2. **Separation Constraints**: Secondary peaks must be separated from the primary peak by a minimum and maximum distance in dy index space, eliminating spurious noise near the limb transition and irrelevant interior structure.
3. **Near-Limb Atmospheric Constraint**: Secondary candidates must lie within the outer portion of the atmospheric column sampled. Features occurring too deep in the interior are rejected as unlikely to represent detached haze.
4. **Absolute and Relative Brightness Thresholds**: Candidate secondary peaks must exceed both an absolute ΔBrightness floor and a fractional threshold relative to the primary peak. This combination suppresses low-amplitude noise while retaining physically plausible weak haze layers.
5. **Valley (Shoulder) Test**: To prevent misclassification of post-limb decay shoulders or monotonic slopes, each secondary candidate must exhibit a sufficient local valley between itself and the primary peak. This allows the algorithm to accept both resolved secondary maxima and physically meaningful shoulder-like features while rejecting simple brightness decay.

Among candidates that pass all constraints, the strongest secondary feature is selected as the detached haze indicator for that limb fragment.

4.4 Geometric Limitations and SPICE Association

A critical geometric limitation arises near the tangent limb: not all detected atmospheric structure corresponds to a unique or stable intercept with Jupiter's reference shape model. In these regions, SPICE geometry cannot assign a reliable planetocentric latitude, even though optically detectable haze structure is present.

As a result, none of the detected peaks—primary or secondary—necessarily coincide with sampling points that have valid SPICE latitude solutions. This is not a failure of the pipeline and not a coding error. It is an inherent consequence of limb-viewing geometry, in which the line of sight intersects extended atmospheric structure without intersecting the modeled planetary surface.

To preserve scientific integrity, the pipeline does not interpolate, extrapolate, or infer latitude values for such detections. Instead, haze detections are explicitly labeled as either:

- **SPICE-resolved:** Secondary features whose sampling points possess valid planetocentric latitude values.
- **Optical-only:** Secondary features detected in ΔBrightness profiles that occur prior to the first valid SPICE surface intercept.

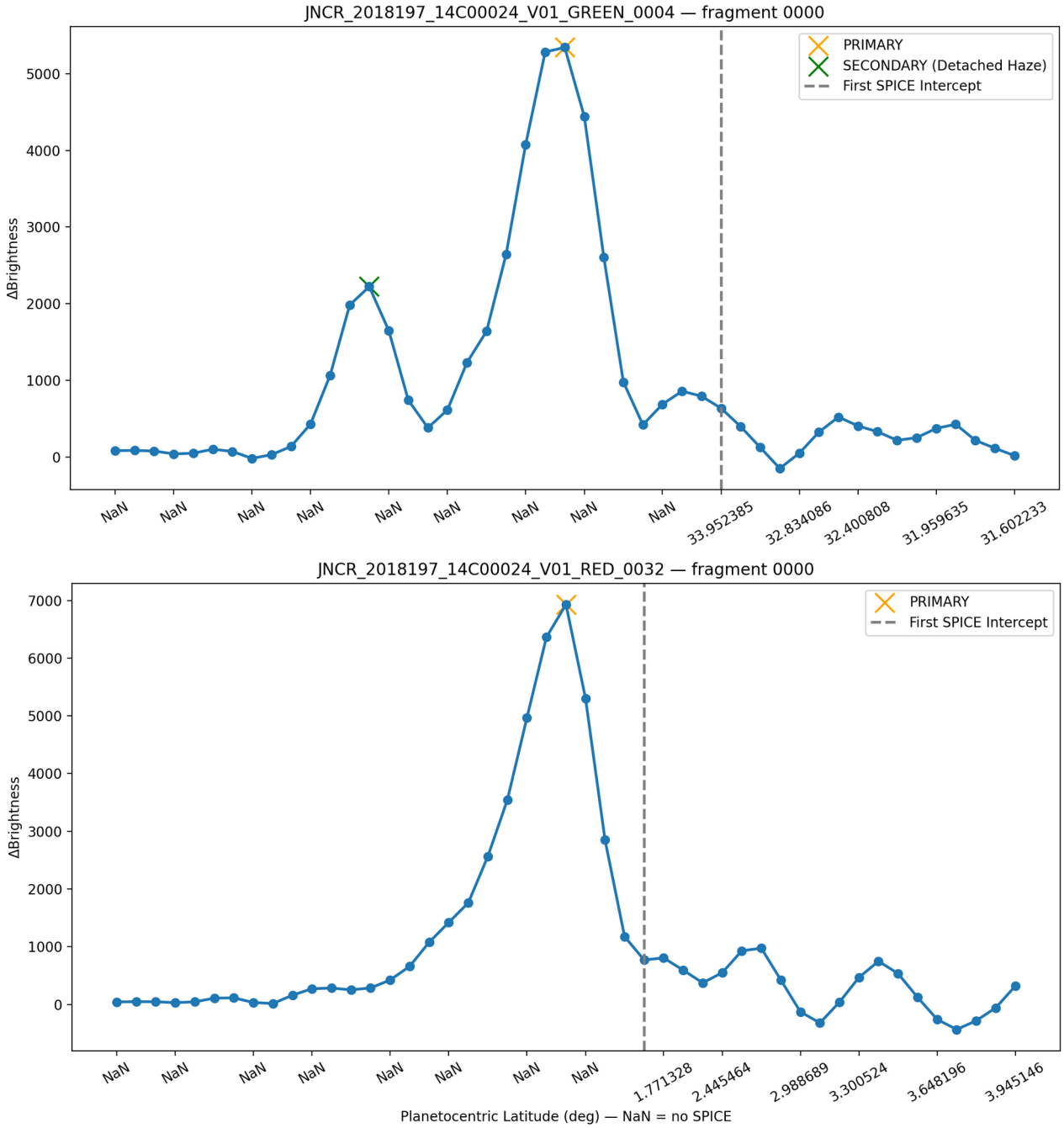
A vertical marker indicating the first valid SPICE intercept is included in all diagnostic plots. ΔBrightness structure occurring prior to this intercept represents optically detected haze for which no planetocentric latitude can be assigned. These detections are preserved rather than interpolated.

4.5 Visualization and Diagnostic Outputs

For each limb fragment, a diagnostic plot is generated showing the smoothed ΔBrightness profile, the identified primary limb peak, any detected secondary haze peak, and the location of the first valid SPICE intercept. Figure 4.5 presents representative examples illustrating both profiles: one with a single dominant limb peak and no detected secondary brightness gradient, and another in which secondary peaks are detected.

Profiles exhibiting a single dominant brightness-gradient peak are interpreted as sampling a continuous cloud deck with no resolved detached haze layer. Profiles exhibiting a secondary peak prior to the first SPICE surface intercept indicate optically detected atmospheric structure occurring above the visible limb, consistent with a detached haze morphology. These diagnostic distinctions motivate the statistical aggregation performed in Stage 5.

In all panels, orange markers denote the PRIMARY limb peak, while green markers denote SECONDARY peaks interpreted as candidate detached haze features. The vertical dashed line indicates the first valid SPICE surface intercept. Brightness-gradient structures occurring prior to this intercept correspond to optically detected atmospheric features for which no unique SPICE geometry can be assigned.



Figures 4.5: Representative limb-normal brightness-gradient profiles illustrating single-peak and double-structure classifications.

4.6 Known Failure Modes and Algorithmic Scope

The peak-finding algorithm operates exclusively on brightness-gradient structure and geometric constraints. It has no intrinsic semantic understanding of cloud decks versus detached haze layers. In rare cases, a detached haze peak may be incorrectly selected as the primary peak when it rivals the cloud deck in ΔBrightness amplitude and occurs earlier along the sampling direction.

Such cases arise when:

- Two peaks have comparable prominence.
- The valley between them is shallow.

- Noise and smoothing subtly favor one feature.

Humans readily recognize these cases as a cloud deck plus detached haze system; the algorithm requires explicit geometric and SPICE-based rules to do the same. The constraints described above eliminate the majority of such failures, and residual ambiguities are flagged for manual review rather than silently excluded.

4.7 Outputs and Transition to Stage 5

Stage 4 writes all classification results directly back to the limb-fragment sampling tables, including:

- Primary and secondary peak indices (dy space)
- ΔBrightness amplitudes
- Associated SPICE geometry where available
- Classification flags indicating detached haze presence

These annotated tables constitute the canonical scientific records for subsequent aggregation. In Stage 5, detections are analyzed statistically across all framelets and their limb fragments to evaluate detached haze occurrence as a function of planetocentric latitude, despite the inherent geometric limitations at the limb.

Implementation details related to limb tracing are provided in Appendix A.

4.8 Execution Example

Artifacts produced in this stage consist of annotated limb-fragment classification tables containing peak locations, amplitudes, and detached-haze flags, which constitute the sole inputs to statistical aggregation and are written to IMG-scoped output directories for reproducibility.

```
python src/04.plot_rectified_perpendiculars.py \
  --indir data/JNCR_2018197_14C00024_V01/cub/stage_03_rectified \
  --framelet-dir data/JNCR_2018197_14C00024_V01/cub/stage_01_framelets \
  --outdir data/JNCR_2018197_14C00024_V01/cub/stage_04_perp_samples \
  --batch
```

Stage 5 — Analysis: Latitude-Dependent Detached Haze Occurrence

Stage 5 evaluates detached haze occurrence statistically across limb fragments rather than attempting precise spatial localization at the fragment level. Because haze signatures consistently occur prior to the first reliable SPICE surface intercept, individual detections lack uniquely assignable coordinates. Aggregation across fragments using first-available SPICE latitude therefore provides a conservative but physically meaningful measure of latitude-dependent haze occurrence. While individual limb fragments do not permit precise

altitude localization due to limited SPICE coverage prior to the limb intercept, the presence or absence of detached haze can be assessed statistically across many fragments. This stage evaluates whether detached haze occurrence varies systematically with planetocentric latitude.

5.1 Proxy Latitude Assignment

For each limb fragment, the first available SPICE-derived planetocentric latitude encountered along the sampling direction is recorded. This latitude serves as a conservative proxy for the fragment's geographic context. Fragments lacking any valid SPICE latitude are excluded from statistical aggregation.

5.2 Binning and Occurrence Metric

Fragments are grouped into latitude bins of fixed width. Within each bin, the fraction of fragments exhibiting a secondary brightness-gradient peak is computed. This fraction represents the occurrence rate of detached haze candidates within that latitude band. Minimum fragment counts per bin are enforced to suppress small-number artifacts.

5.3 Results

Detached haze is not uniformly present along the limb. Instead, its occurrence increases sharply at higher planetocentric latitudes, reaching near-continuous presence above approximately 31–33° in this observation.

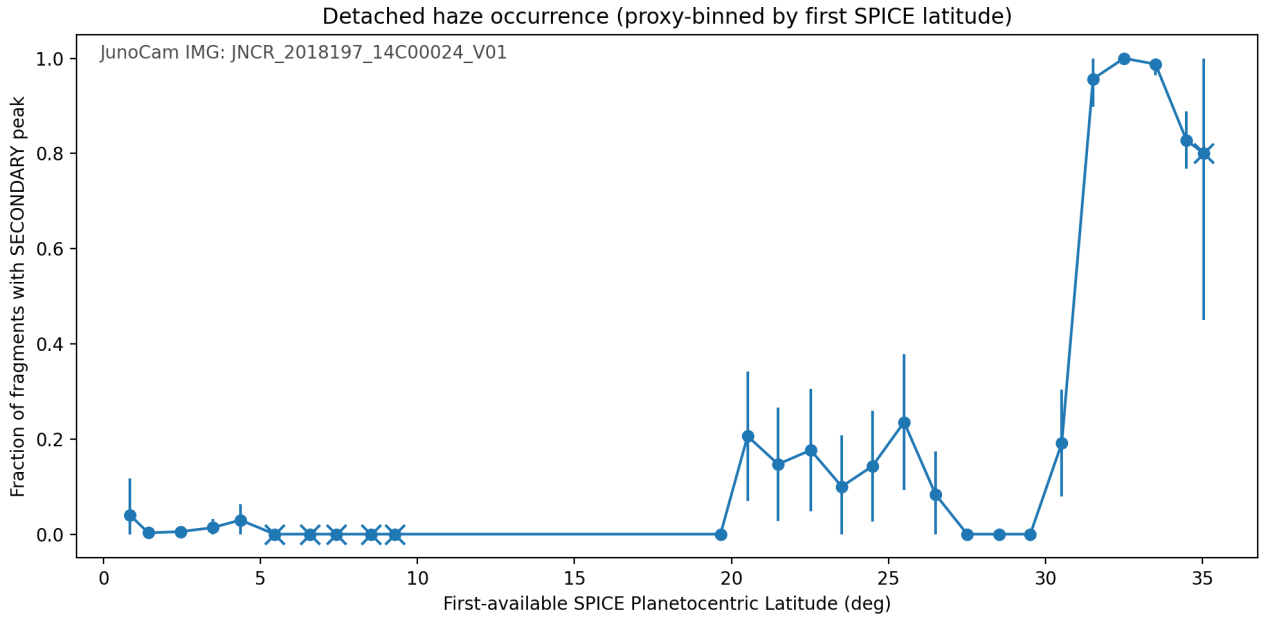


Figure 5.3 - Fraction of limb fragments exhibiting secondary brightness-gradient peaks (detached haze candidates), binned by first-available SPICE planetocentric latitude for JunoCam IMG JNCR_2018197_14C00024_V01.

5.4 Limitations and Interpretation

Because SPICE coverage does not extend to the altitude of detached haze layers, precise spatial localization is not possible at the fragment level. The observed trend therefore represents a statistical, not geometric, association. Nevertheless, the sharp transition in occurrence rate strongly supports a latitude-dependent haze structure.

5.5 Execution Example

Artifacts produced in this stage consist of latitude-binned occurrence tables and summary figures representing the final analytical outputs of the pipeline, written to IMG-scoped output directories for reproducibility. To reproduce, execute from the root directory with the following prompt:

```
python src/07.haze_analysis.py \
  --stage6-root data/JNCR_2018197_14C00024_V01/cub/stage_06_peaks \
  --outdir      data/JNCR_2018197_14C00024_V01/analysis/stage_05 \
  --bin-width-deg 1.0 \
  --min-fragments-per-bin 15
```

Stage 6 — Methods Overview

This study employs a modular image-analysis pipeline designed to detect and classify detached haze features along the Jovian limb in JunoCam imagery. All processing is performed on a per-IMG basis, with each stage consuming the outputs of the preceding step while remaining independently executable for diagnostic purposes. The pipeline consists of: (1) ISIS-based ingestion and SPICE initialization, (2) sub-pixel limb tracing, (3) rectification of curved limb segments, (4) brightness-gradient analysis and peak classification, and (5) statistical aggregation by proxy latitude. Detailed descriptions of each stage, including execution commands and generated artifacts, are provided inline in the subsequent sections.

Stage 7 — Discussion

The results presented here demonstrate a clear latitude-dependent transition in detached haze occurrence along the observed limb. Although individual limb fragments lack precise SPICE geometry at the haze altitude, the aggregate behavior across fragments and framelets reveals a statistically robust pattern: detached haze features are rare or absent at low planetocentric latitudes and become increasingly prevalent toward higher latitudes, approaching near-continuous occurrence above approximately 31–33° in this observation.

This conclusion is deliberately conservative. Detached haze identification relies solely on brightness-gradient structure and geometric consistency, without assuming a priori vertical placement or atmospheric composition. The absence of SPICE surface intercepts for many haze-adjacent samples precludes precise altitude assignment at the fragment level. However, because first-available SPICE intercept latitude varies smoothly across fragments and framelets, it provides a meaningful proxy for large-scale latitudinal trends when analyzed statistically.

Importantly, this approach avoids over-interpretation of individual profiles while preserving sensitivity to genuine atmospheric structure. The observed transition therefore supports the hypothesis that detached haze occurrence is not uniform along the limb, but instead exhibits a strong dependence on planetocentric latitude. Future work incorporating multi-

image aggregation or improved limb geometry solutions may further refine altitude constraints, but such refinements are not required to establish the primary latitudinal trend detected here.

Stage 8 — Reproducibility and Data Availability

All analysis code used in this study—including limb tracing, rectification, peak finding, and statistical aggregation—are publicly available in a GitHub repository at https://github.com/kevinkell-y/juno-haze-classification. The repository contains versioned source code, execution instructions, and directory conventions required to reproduce all figures and tables in this paper. Intermediate products generated during processing are preserved in IMG-specific directories to ensure reproducibility and facilitate independent inspection. Processed data products (rectified limb profiles, classification tables, and diagnostic figures) are derived from publicly available JunoCam images distributed by NASA's Planetary Data System. Instructions to reproduce each pipeline stage are provided inline in the Methods section.

Each JunoCam cube's label and accompanying manifest file record the complete set of SPICE kernels used during initialization (LSK, PCK, SPK, CK, FK, IK, SCLK). These kernel sets were drawn from locally synchronized ISIS and Juno kernel repositories and are publicly available online. The manifest files stored alongside each .cub constitute the definitive provenance record for all geometry used in this study.

References

Brueshaber, K. (2021). *JunoCam Image Processing Document, Revision 2*. NASA Jet Propulsion Laboratory, Planetary Science Group Technical Report #3222.

Eichstädt, G., et al. (2021). *JunoCam Limb Geometry and Imaging Characteristics*.

Navigation and Ancillary Information Facility (NAIF). (2021). *SPICE Toolkit Documentation*. https://naif.jpl.nasa.gov/naif/

Kelly, K. (2021). *MSP Final Report*. NASA Jet Propulsion Laboratory.

Appendix A — Limb-Tracing Algorithm and Implementation

This appendix supports Stage 2 (Limb Detection and Tracing) described in Section 2 of the main text.

The limb-tracing module (02.trace_limb_polyline.py) combines SPICE-registered camera geometry with grayscale-gradient analysis to delineate Jupiter's planetary edge.
Below is a high-level pseudocode description of the complete process.

A.1 Pre-processing

Input: SPICE-attached framelet F.cub
Output: Grayscale array I(x, y)

1. Convert .cub → 16-bit TIFF and 8-bit PNG using isis2std.
2. Read PNG and normalize pixel intensities to [0, 1].
3. Apply a 3×3 Gaussian blur to suppress single-pixel noise.

A.2 Gradient-field computation

4. Compute Sobel derivatives:
   $G_x = dI/dx$, $G_y = dI/dy$
5. Compute gradient magnitude and orientation:
   $M = \sqrt{(G_x^2 + G_y^2)}$
   $\Theta = \arctan2(G_y, G_x)$
6. Identify global maximum of M as initial limb seed P_0.

This replaces the earlier fixed-orientation seeding heuristic (Wen 2021) with a geometry-agnostic initialization that adapts to any illumination direction.

A.3 Bidirectional edge propagation

7. Initialize limb polyline $L = \{P_0\}$.
8. For each propagation direction $D \in \{\text{forward, reverse}\}$:
   repeat
   - Sample candidate points Q within a narrow angular window ($\pm\theta$) around $\Theta(P_0)$.
   - Select Q maximizing local gradient magnitude M(Q).
   - Append Q to L if distance(P_0,Q) < δ_{max}.
   - Update orientation $\Theta(P_0) \leftarrow \Theta(Q)$; set $P_0 \leftarrow Q$.

   until gradient < τ or boundary reached

Typical parameters:
$\theta \approx 10–15°$, $\delta_{max} \approx 3$ pixels, $\tau \approx 0.1 \times M_{max}$.
Both directions are traced independently and then merged, yielding a continuous limb polyline.

A.4 Polyline smoothing and export

9. Apply cubic spline or low-pass polynomial fit to L.
10. Save limb coordinates → <CUB>_LIMBENDPOINTS.csv
11. Overlay cyan markers on source image → <CUB>_OVERLAY.png

The resulting CSV provides ordered (x, y) positions along the detected planetary edge at sub-pixel precision.
Each overlay PNG offers a rapid-visual confirmation of limb detection quality.

A.5 Performance and false-positive control

- **Runtime:** ≈ 0.5–1 s per 1600×128 framelet on a modern CPU.
- **Memory footprint:** < 200 MB per image.
- **False positives:** Occur when the scene lacks dark-to-bright transitions (planet fills frame).
  These cases are flagged during later tangent-normal sampling when no null-space data are detected.

A.6 Scientific use

The extracted limb polylines define the instantaneous geometric curvature of Jupiter’s visible atmosphere. They serve as the spatial reference for subsequent rectification and altitude mapping, enabling quantitative analysis of detached haze layers, scattering gradients, and curvature-dependent optical depth.

No assumption is made that the rectified limb represents a true planetary vertical; rather, rectification is performed in image-coordinate space to stabilize sampling geometry for comparative brightness-gradient analysis.