Astrelle Sky Simulator

Technical Manual and User Guide

Author: Srujan Kulkarni

Affiliation: RV College of Engineering, Bengaluru

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern astronomy relies on the acquisition of digital images of the sky using sensitive detectors, typically Charge-Coupled Devices (CCDs) or CMOS sensors. The standard data format for these images is the Flexible Image Transport System (FITS), a digital file format that stores not only the multi-dimensional pixel data but also a wealth of metadata in a human-readable header.

The recent and rapid deployment of satellite mega-constellations in Low Earth Orbit (LEO) has introduced a significant and pervasive source of contamination in these astronomical images. When a satellite passes through a telescope's field of view during an observation, its reflected sunlight creates a bright, linear trail. These trails can obscure or saturate the light from faint astronomical sources, corrupt photometric measurements, and trigger false positives in automated searches for transient events like supernovae or asteroids [11].

The development and validation of algorithms to detect and mitigate these trails require large, diverse, and well-characterized datasets. Astrelle is a high-fidelity sky simulation software designed to generate such datasets. It creates realistic synthetic FITS images of the sky as observed from a specific location, like the Devasthal observatory. By modeling the entire photon-to-pixel pipeline from first principles, Astrelle provides an invaluable "ground-truth" dataset for training and testing detection algorithms. This manual details the software's functionality, the physical models it employs, and its implementation.

Chapter 2

Installation

2.1 System Dependencies

Astrelle is designed for Debian-based Linux systems like Ubuntu. The core application relies on a number of standard scientific Python libraries that are best installed through the system's package manager to ensure compatibility with other system components. These dependencies are:

- python3-numpy
- python3-pandas
- python3-scipy
- python3-astropy
- python3-astroquery
- python3-pyvo
- python3-matplotlib
- python3-requests

2.2 Python-Specific Dependencies

A few required libraries are not available in the standard system repositories or need to be specific versions. These must be installed with Python's package manager, `pip`.

- **PyQt6:** For the graphical user interface.
- **skyfield:** For high-precision satellite orbit propagation.

2.3 Installation Steps (Debian/Ubuntu) (MODIFIED)

Astrelle uses a hybrid installation process. The main application is installed via a `.deb` package, which handles system dependencies, and then a few specific Python libraries are installed using `pip`.

1. **Download the Package**: Obtain the latest `astrelle_*.deb` file from the **project's GitHub repository**. The package is typically found on the "Releases" page.

2. **Install the Core Application**: Open a terminal and run the following command. This will install Astrelle and its core system dependencies.

```
sudo dpkg -i astrelle_*.deb

```

   You may see an error about missing dependencies. Run the following command to fix this and install the core libraries from the system repositories:

```
sudo apt-get install -f

```

3. **Install Python-specific Libraries**: Install the remaining libraries with `pip`.

```
pip3 install PyQt6 skyfield

```

2.4 Launching the Application

Once installed, you can launch the application in two ways:

- **From the Application Menu**: Open your desktop's application menu and search for "Astrelle Sky Simulator".

- **From the Terminal**: Open a terminal and run the command:

```
astrelle-gui

```

Chapter 3

User Guide

This chapter provides a detailed walkthrough of the Astrelle Graphical User Interface (GUI). We will cover each component of the main window and explain how to configure and run a simulation.

3.1 Main Window Overview

When you launch Astrelle, you are presented with the main window, which is organized into three key areas for an efficient workflow.

- **Control Panel (Left):** This is where you configure all simulation parameters. Settings are grouped into collapsible sections.
- **Image Viewer (Top Right):** After a simulation is complete, the final FITS image is displayed here.
- **Log Window (Bottom Right):** This window provides real-time feedback, status updates, query results, and error messages during the simulation process.

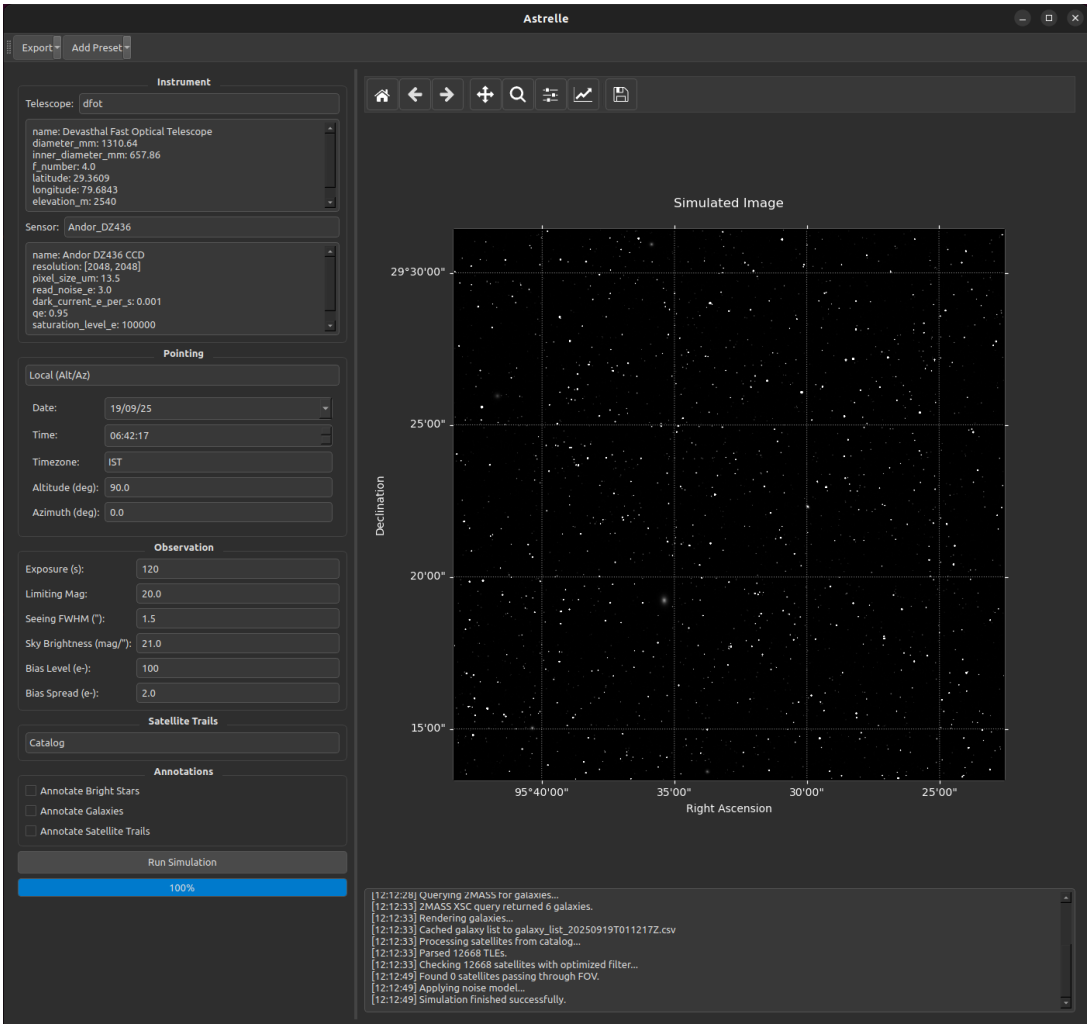

Figure 3.1: The main window of the Astrelle Sky Simulator, showing the Control Panel, Image Viewer, and Log Window.

3.2 The Control Panel: Configuring a Simulation

The control panel on the left contains all the settings required to define your simulation. These are logically grouped into collapsible sections.

3.2.1 Instrument Configuration

This section allows you to select the telescope and sensor combination for the simulation.

- **Preset:** Choose from a list of pre-defined instrument setups. Astrelle uses a JSON file (`presets.json`) to manage these. You can add your own instruments by editing this file. The default preset is for the Devasthal Fast Optical Telescope (DFOT).

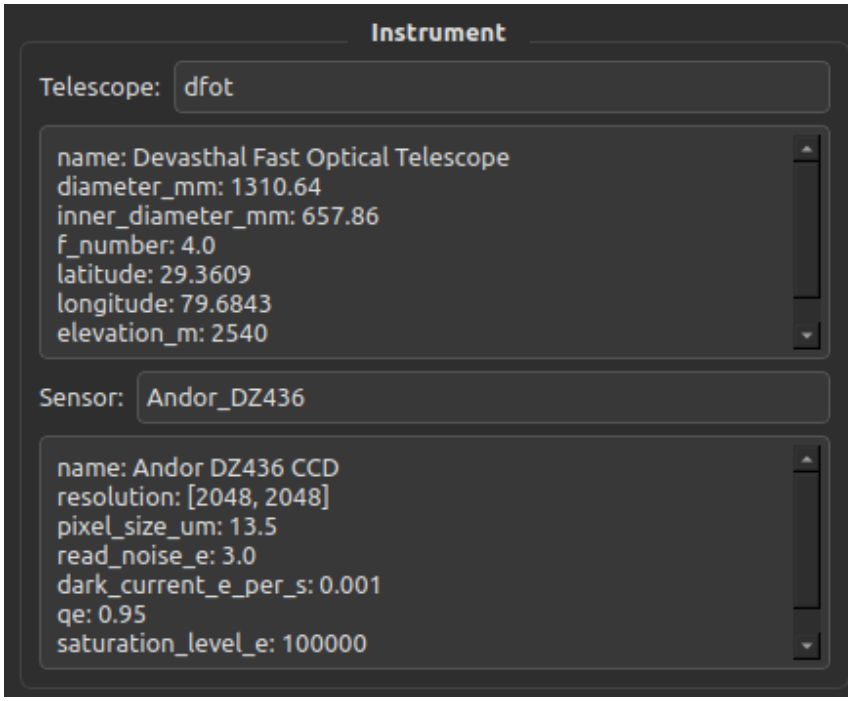

Figure 3.2: The Instrument Configuration section. Select a preset from the dropdown menu.

3.2.2 Pointing and Observation

Here, you define where the telescope is pointing and the core parameters of the observation.

- **Pointing Coordinates:** You can specify the center of the field of view in two ways:
  - **RA/Dec:** Celestial coordinates (Right Ascension and Declination).
  - **Alt/Az:** Local coordinates (Altitude and Azimuth) for a specific date and time.
- **Observation Parameters:**
  - **Exposure Time (s):** The total duration of the simulated observation in seconds.
  - **Seeing (arcsec):** The atmospheric seeing, measured as the Full Width at Half Maximum (FWHM) of the stellar PSF in arcseconds.
  - **Sky Brightness (mag/arcsec²):** The brightness of the sky background.

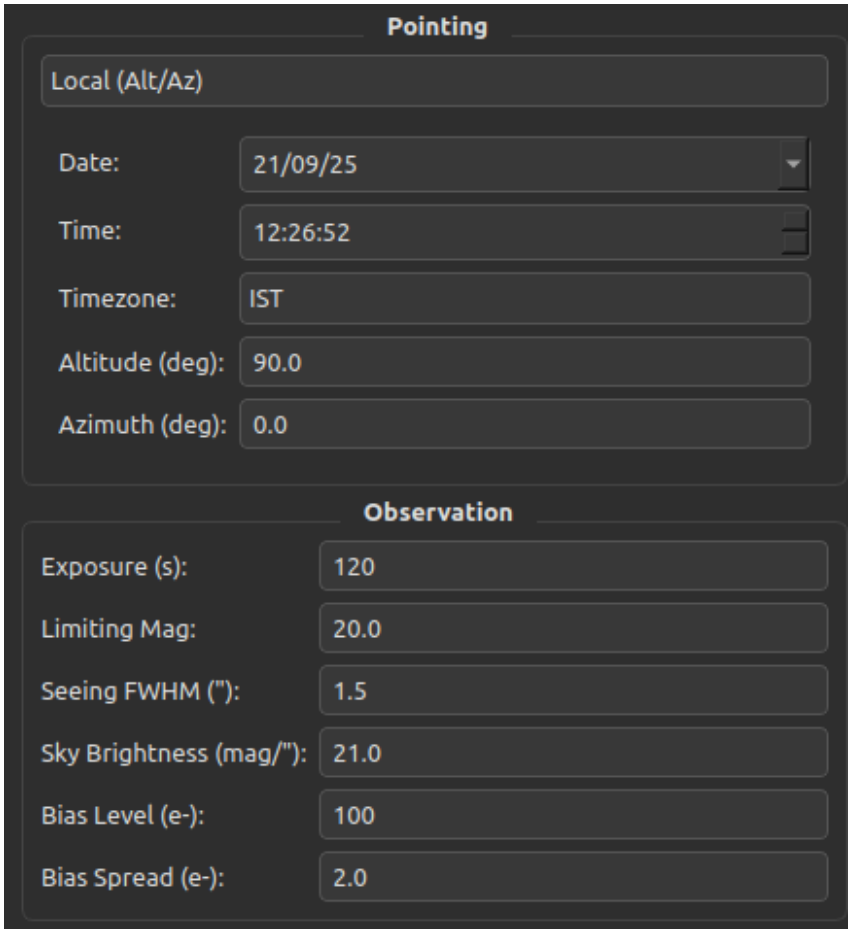

Figure 3.3: The Pointing and Observation section, where the telescope's target and exposure settings are defined.

3.2.3 Satellite Simulation

This section controls how satellite trails are generated.

- **Simulation Mode:**
  - **Catalog Mode:** This is the default mode. Astrelle will automatically download the latest Two-Line Element (TLE) data from CelesTrak and simulate trails for any real satellites that pass through the field of view during the exposure.
  - **Synthetic Mode:** This mode allows you to create artificial satellite trails. It is useful for testing detection algorithms with specific, user-defined trail characteristics (e.g., brightness, speed, angle). When selected, additional fields will appear to configure the synthetic trail.

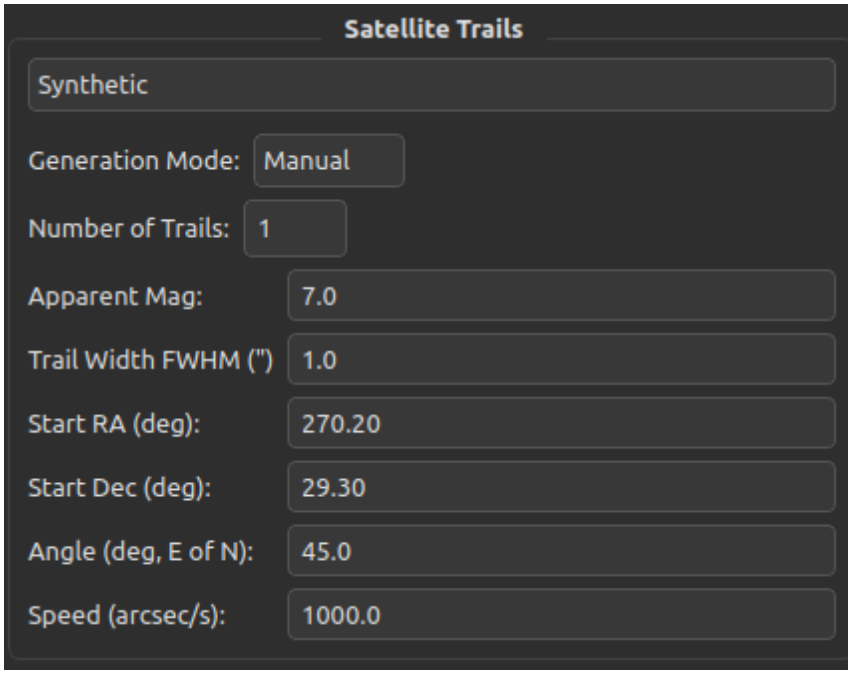

Figure 3.4: The Satellite Simulation section, allowing a choice between real-world (Catalog) and user-defined (Synthetic) trails.

3.2.4 Running the Simulation and Viewing Output

Once all parameters are set, you can start the simulation.

- **Run Simulation Button:** Click this button at the bottom of the control panel to begin the simulation.
- **Monitoring Progress:** Watch the **Log Window** for real-time updates. It will show progress on downloading catalogs, calculating positions, rendering objects, and generating noise. Any errors will also be reported here.
- **Viewing the Image:** Upon successful completion, the final simulated image will appear in the **Image Viewer**. You can use your mouse to pan and zoom.
- **Exporting Data:** Buttons below the image viewer allow you to export the data. The primary output is a FITS file, but the image can also be saved as a PNG for quick viewing. You can also export ground-truth data (lists of stars, galaxies, and satellites) as CSV files.

Chapter 4

Tutorial: Simulation Examples

This chapter provides step-by-step examples for common use cases.

4.1 Example 1: Simulating a Star Field using Local Coordinates

Let's perform a simple 60-second simulation pointing at the eastern sky using local Altitude/Azimuth coordinates.

1. **Instrument:** Select the default `DFOT` preset.
2. **Pointing:** Choose the **Alt/Az** coordinate system. Set Altitude to `30` and Azimuth to `90`.
3. **Observation:** Set Exposure Time to `60` s, Seeing to `1.8` arcsec, and Sky Brightness to `20.0` mag/arcsec^2.
4. **Satellites:** Keep the default **Catalog Mode** selected. The software will check for real satellites but for this short exposure, it's unlikely any will be found.
5. **Run:** Click **Run Simulation**.

Expected Result: The simulator will calculate the current date and time to convert the Alt/Az coordinates to RA/Dec and generate an image of the stars in that patch of sky. This is useful for quickly simulating what the telescope would see at a specific local direction.

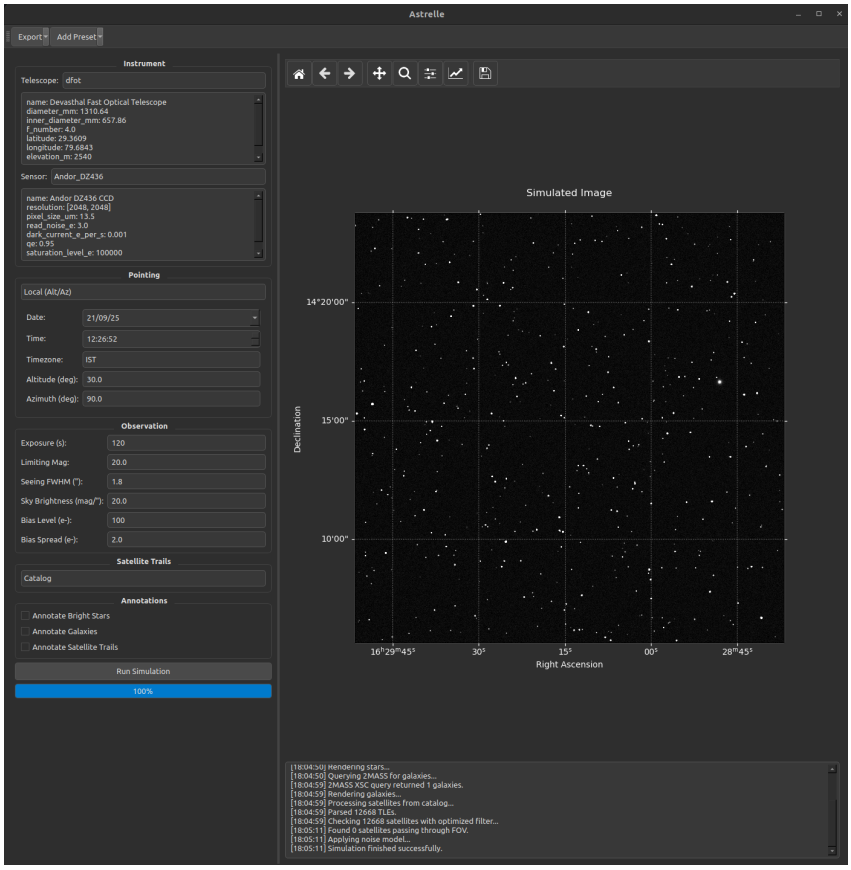

Figure 4.1: Expected output for Example 1: A star field corresponding to the specified local coordinates.

4.2 Example 2: Creating a Synthetic Satellite Trail

This example simulates a field and adds a user-defined, artificial satellite trail. This is ideal for testing trail detection algorithms.

1. **Instrument:** Select the default `DFOT` preset with any sensor you prefer.
2. **Pointing:** Select the **RA/Dec** coordinate system. Set RA to `270` and Dec to `29`.
3. **Observation:** Set Exposure Time to `300` s, Seeing to `1.2` arcsec, and Sky Brightness to `22.0` mag/arcsec^2 (for dark sky conditions).
4. **Satellites:** Change the Simulation Mode to **Synthetic Mode**. New input fields for the trail will appear.
5. **Configure Synthetic Trail:** Choose Random mode for generation.
6. **Run:** Click **Run Simulation**.

Expected Result: The image will show a randomly generated trail inside the field.

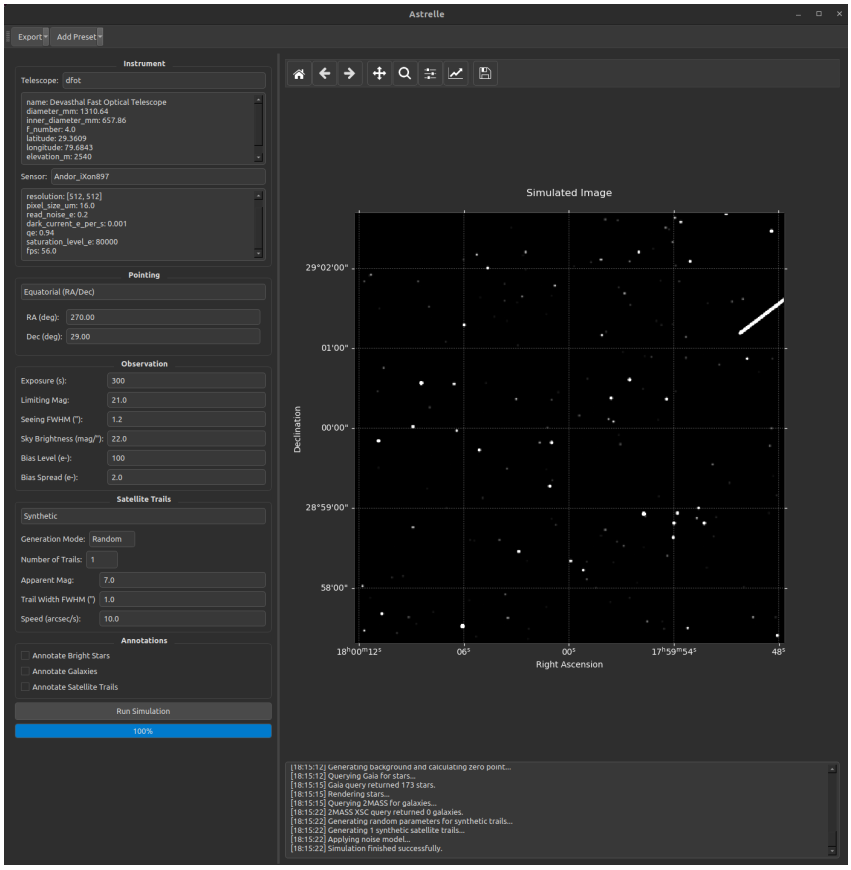

Figure 4.2: Expected output for Example 2: A deep field of galaxies with a bright, horizontal synthetic trail.

Chapter 5

Technical Reference: The Physical Model

The Astrelle simulator is built as a modular pipeline, where each stage represents a distinct physical process in the journey of light from a celestial object to a final data number in a FITS file. The entire process is orchestrated by a main control script, which calls upon specialized modules for each step. Table 5.1 lists the key instrumental parameters for the primary simulated setup.

Table 5.1: Key Parameters for the Devasthal Fast Optical Telescope (DFOT) Setup

Parameter	**Value**
Telescope Aperture	1.3 m
Focal Ratio	f/4
Observatory Latitude	29.3609° N
Observatory Longitude	79.6843° E
Observatory Elevation	2540 m
Default Sensor	Andor DZ436
Sensor Resolution	2048 x 2048 pixels
Pixel Size	13.5 μm

5.1 Radiometric Calibration: The Instrumental Zero-Point

The foundation of a physically accurate simulation is a robust radiometric calibration. We must establish a precise mathematical relationship between the standard astronomical magnitude system and the physical number of photo-electrons generated in the detector. This is achieved by calculating the instrumental **zero-point** (Z_p), defined as the count rate (in electrons per second, e^-/s) that would be measured from a star of magnitude 0.

The relationship between an object's apparent magnitude (m) and the total number of electrons (N_{e^-}) it generates during an exposure time (t_{exp}) is given by the fundamental

photometric equation:

$$N_{e^-} = t_{exp} \cdot Z_p \cdot 10^{-0.4m} \tag{5.1}$$

The validity of this equation stems from the logarithmic definition of the magnitude scale. To calculate Z_p for our simulated instrument, we must physically model the detection of a standard star. We perform a synthetic observation of a star with a known spectral flux distribution (SED), for which we use the CALSPEC spectrum of Alpha Lyrae (Vega) [8]. The total detected electron rate is the integral of Vega's flux over all wavelengths, weighted by the response of every component in the light path:

$$\begin{aligned} \text{Rate}_{e^-,Vega} = A \int_0^\infty \frac{F_{\lambda,Vega}}{E_{photon}(\lambda)} \cdot T_{atm}(\lambda) \cdot T_{tel} \\ \cdot S(\lambda) \cdot Q(\lambda)\, d\lambda \end{aligned} \tag{5.2}$$

The validity of this integral formulation stems from the principle of photon counting. A detector does not measure magnitude directly; it counts photons. The number of photons detected from a source is the integral of the source's photon spectral density (photons s^{-1} cm^{-2} $Å^{-1}$) multiplied by the total transmission probability of the entire system (atmosphere, telescope, filter) and the detection probability (quantum efficiency) over all wavelengths. Equation 5.2 is the direct mathematical expression of this process, converting the standard energy-based flux of Vega into a detected electron rate. Each term has a precise physical meaning:

- A is the telescope's unobscured collecting area in cm^2: $A = \pi((D/2)^2 - (d/2)^2)$, where D and d are the primary and secondary mirror diameters.
- $F_{\lambda,Vega}$ is the spectral flux density of Vega from the CALSPEC database, in units of erg s^{-1} cm^{-2} $Å^{-1}$. (See Figure 5.3)
- $E_{photon}(\lambda) = hc/\lambda$ is the energy of a single photon at wavelength λ. The term F_λ/E_{photon} converts the energy flux into a photon flux.
- $T_{atm}(\lambda)$ is the atmospheric transmission as a function of wavelength, modeled for the Devasthal site using the work of V. Mohan [9]. This model includes terms for Rayleigh scattering, aerosol scattering, and ozone absorption. (See Figure 5.1)
- T_{tel} is the wavelength-independent optical throughput of the telescope system.
- $S(\lambda)$ is the normalized response of the photometric filter (in this case, the Gaia G-band from [7]). (See Figure 5.2)
- $Q(\lambda)$ is the quantum efficiency of the sensor, assumed constant over the filter bandpass.

The code numerically integrates Equation 5.2. As Vega's apparent magnitude is defined as $m_{Vega} \approx 0.03$, the final zero-point is found by scaling the result: $Z_p = \text{Rate}_{e^-,Vega}/10^{-0.4 \cdot m_{Vega}}$.

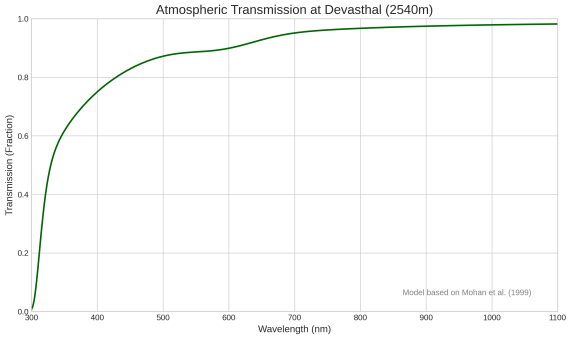

Figure 5.1: Atmospheric transmission curve for Devasthal, based on the model by Mohan et al. [9], showing the combined effects of Rayleigh scattering, aerosol scattering, and ozone absorption.

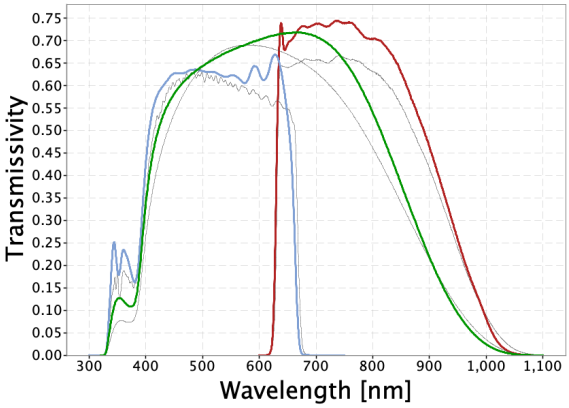

Figure 5.2: The Gaia DR3 G-band filter passband.

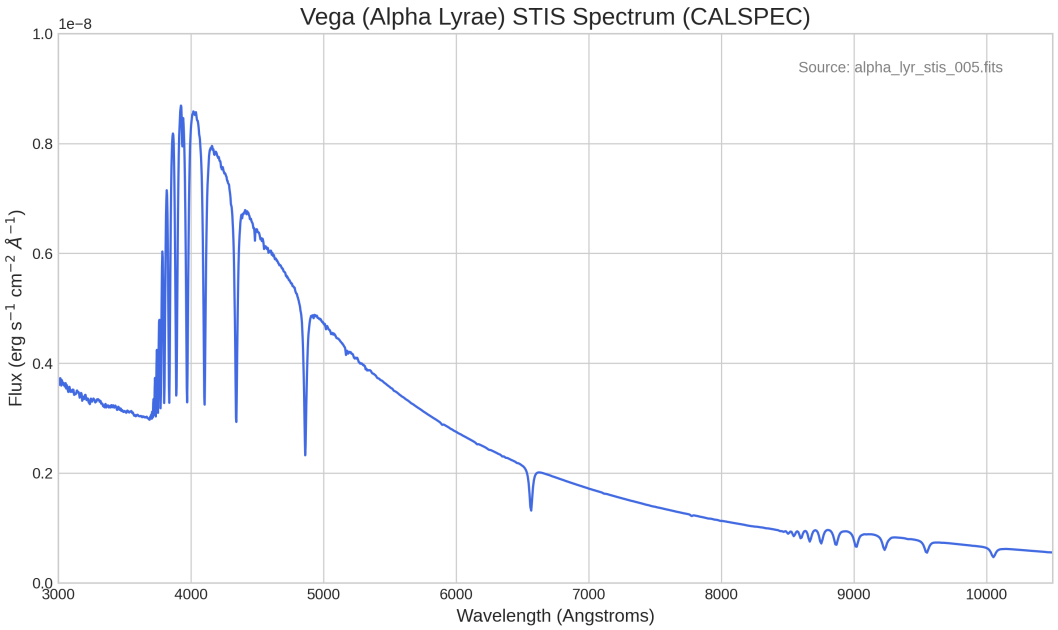

Figure 5.3: The CALSPEC spectral flux distribution of Vega.

5.2 Sky Background and Stellar Population

The sky itself is a source of light. The sky brightness, specified by the user in mag/arcsec^2, is converted to an electron rate per pixel using the zero-point and the pixel scale of the instrument. This forms the base flux level of the image.

The stellar population is sourced from the **Gaia DR3** catalog [19]. Gaia was chosen for its unparalleled astrometric precision, all-sky coverage, and deep, homogeneous photometry, making it the definitive source for stellar positions and magnitudes. The query sent to the Gaia archive is not a simple cone search, but rather a polygonal query on the celestial sphere. This is a critical detail, as it correctly accounts for the projection distortion of a rectangular field of view onto the curved sky, ensuring that stars at the edge of the field are accurately selected.

Each star's celestial coordinates (RA, Dec) are projected onto the 2D detector grid using a standard gnomonic projection (RA—TAN, DEC—TAN), which forms the basis of the image's WCS. An ideal star, as viewed from outside the atmosphere, is a perfect point source (mathematically, a Dirac delta function). The turbulence in the Earth's atmosphere blurs this point source into a fuzzy spot. This blurring process is known as **convolution**, and the resulting blurred shape is the **Point Spread Function (PSF)**.

At the star's calculated pixel coordinates (x_0, y_0), a PSF is rendered. This PSF is the result of convolving the ideal point source with the atmospheric "seeing" profile, which is well-approximated by a 2D Gaussian [10]. In polar coordinates (r, θ), this is:

$$I(r) = A \exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{5.3}$$

The profile is circularly symmetric (independent of θ). The standard deviation σ is directly related to the user-provided seeing FWHM by $\sigma = \text{FWHM}/(2\sqrt{2\ln 2})$. The amplitude A is normalized such that the total volume under the 2D Gaussian is equal to the total electron count (N_{e^-}) for that star, calculated from Equation 1.

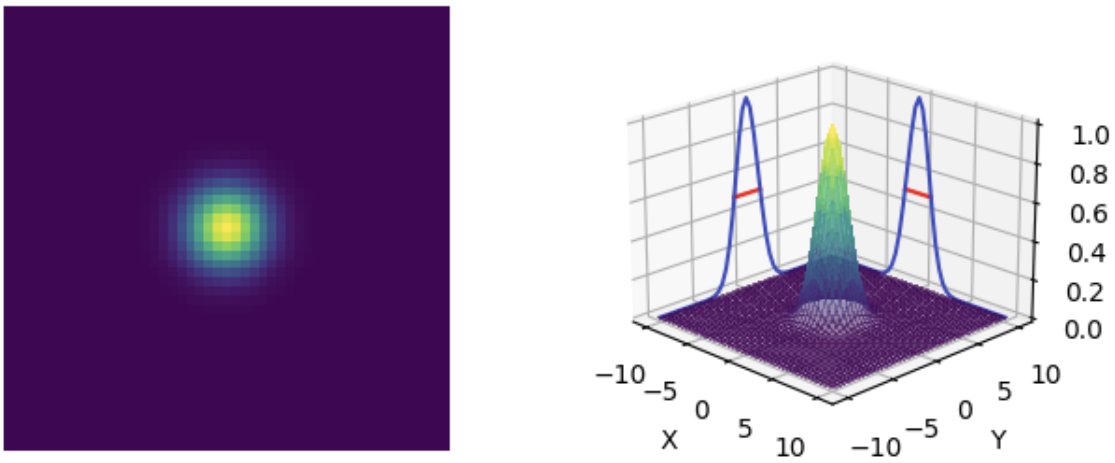

Figure 5.4: A 3D visualization of the Gaussian Point Spread Function (PSF) used to model stars. This represents the "seeing" applied to every point source.

5.3 Galaxy Population

Galaxies are sourced from the **2MASS Extended Source Catalog (XSC)** [20], chosen because it provides the necessary structural parameters (effective radius, ellipticity, position angle) for rendering. Unlike stars, galaxies are "extended sources"; they already have an intrinsic shape and size. This ideal, "space-view" shape is modeled using the Sersic profile [1, 2], a general function:

$$I(R) = I_e \exp\left(-b_n \left[\left(\frac{R}{R_e}\right)^{1/n} - 1\right]\right) \tag{5.4}$$

where R_e is the effective radius, I_e is the surface brightness at R_e, and n is the Sersic index describing the profile's shape.

To simulate the final on-sky appearance, this ideal Sersic model must also be blurred by the atmosphere, just like a star. This is achieved by **convolving** the 2D ideal Sersic profile of the galaxy with the 2D Gaussian PSF that represents the atmospheric seeing. The result is a realistic, "fuzzy" galaxy image where the sharp features of the ideal model have been softened by the atmosphere. To render an elliptical galaxy, the radius R is calculated for each pixel (x, y) based on its elliptical distance from the center, accounting for the galaxy's ellipticity and position angle provided by the 2MASS catalog.

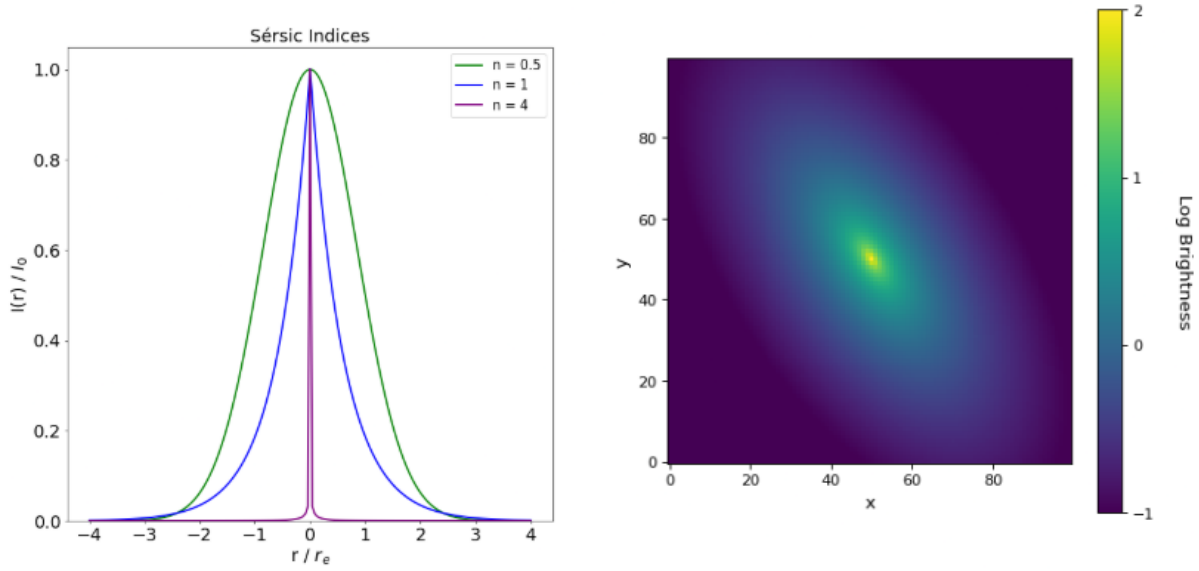

Figure 5.5: Comparison of 1D Sersic profiles for different Sersic indices (n). $n = 1$ represents an exponential disk typical of spiral galaxies, while $n = 4$ represents a de Vaucouleurs profile typical of elliptical galaxies.

5.4 Satellite Dynamics and Trail Rendering

The simulation of satellite motion uses the **Skyfield** library, which provides a robust implementation of the SGP4/SDP4 orbital propagators. These are the standard models used by NORAD to predict satellite positions from their Two-Line Element (TLE) sets. The TLE data is sourced in near real-time from CelesTrak [21].

A satellite trail is rendered not as a single line, but as the temporal integral of the satellite's PSF as it traverses the sensor. Astrelle simulates this by discretizing the exposure into a number of frames based on the sensor's specified Frames Per Second (FPS). For each frame, the satellite's position is calculated, and a PSF stamp is added to the image. The flux in each stamp is the total satellite flux (from its apparent magnitude) divided by the total number of frames. This correctly simulates the fact that a fast-moving satellite will deposit fewer photons per pixel than a slow-moving one.

5.5 Detector Noise Model

The final stage is to convert the ideal, noiseless floating-point image into a realistic integer-valued FITS image. This is achieved by simulating the stochastic noise sources inherent in CCD/CMOS detectors [16]:

$$\sigma_{pixel} = \sqrt{N_{signal} + N_{dark} + \sigma_{read}^2} \tag{5.5}$$

Where N_{signal} is the sum of electrons from stars, galaxies, and sky background, and N_{dark} is from thermal dark current. The simulation implements this by:

1. Treating the signal in each pixel ($N_{signal} + N_{dark}$) as the mean of a Poisson process and drawing a random value to simulate **shot noise**.
2. Adding random noise from a Gaussian distribution with standard deviation σ_{read} to simulate **read noise**.
3. Adding a constant **bias offset** with a small Gaussian spread.

The final pixel values are converted to integers (Analog-to-Digital Units or ADU) and stored in the FITS file. Table 5.2 summarizes these noise sources.

Table 5.2: Summary of Simulated Detector Noise Sources

Noise Source	**Origin**	**Statistical Model**
Shot Noise	Photon arrival statistics	Poisson
Read Noise	Detector electronics	Gaussian
Dark Current Noise	Thermal electrons	Poisson
Bias Noise	Electronic offset variation	Gaussian

Chapter 6

Results and Discussion

6.1 The Astrelle Simulator: Software and Functionality

The primary result of this project is the Astrelle software itself, a fully functional simulation tool capable of producing high-fidelity synthetic astronomical images. It is controlled via a comprehensive Graphical User Interface (GUI) as shown in Figure 3.1, which provides an intuitive workflow for setting up and running complex simulations.

6.2 Qualitative Analysis of Simulated Images

Visual inspection of the simulated FITS files reveals a high degree of realism. Figure 6.1 shows a 120-second exposure of a dense star field. Several key features that mimic real observations are immediately apparent:

- **PSF Variation:** Stars are not rendered as single bright pixels but as realistic, seeing-convolved PSFs. The brightness of stars correctly scales with their catalog magnitude, from bright, saturated stars with large halos to faint stars that are barely distinguishable from the background noise.
- **Background Texture:** The sky background is not a flat, uniform color. It exhibits a subtle, grainy texture. This is the result of the Poissonian shot noise from the sky flux itself, combined with the Gaussian read noise of the detector, which accurately reproduces the characteristic "salt-and-pepper" noise of a real CCD image.
- **Dynamic Range:** The image correctly displays a wide dynamic range, with pixel values spanning from the background level (around the bias level) to the saturation limit of the sensor for the brightest stars.

Figure 6.2 demonstrates the core functionality of the simulator. A bright satellite trail is superimposed on the star field. The simulation accurately captures key features of a real satellite trail:

- **Linearity and Width:** The trail is a sharp, linear feature, consistent with the near-constant angular velocity of a satellite during a short exposure. Its width is determined by the convolution of the satellite's intrinsic PSF with its motion during each discrete "frame" of the exposure.
- **Flux Deposition:** The surface brightness of the trail is consistent with the satellite's apparent magnitude. The flux is spread along the trail's path, correctly reducing the peak pixel value compared to a stationary object of the same brightness.

Figure 6.3 shows a simulated deep field populated with galaxies. The rendering of these extended sources using elliptical Sersic profiles adds another layer of realism. The galaxies exhibit a range of sizes, ellipticities, and surface brightness profiles, from the centrally concentrated profiles typical of elliptical galaxies ($n = 4$) to the flatter, exponential disks of spiral galaxies ($n = 1$), all correctly convolved with the atmospheric seeing. The combination of these elements results in a synthetic image that is challenging to distinguish from a real astronomical observation at first glance, fulfilling the primary goal of the project.

Figure 6.1: Example simulation of a dense star field in the galactic plane. Note the variation in stellar brightness and the noisy texture of the background.

Figure 6.2: A simulation showing a satellite trail passing through a moderately dense star field. The trail is correctly rendered as a streak with a finite width defined by the PSF.

Figure 6.3: A simulated deep field containing several galaxies rendered as Sersic profiles, convolved with the atmospheric PSF.

6.3 Quantitative Validation

Beyond visual inspection, the simulator's output must be quantitatively correct. This validation is twofold: **photometric** (are the brightnesses correct?) and **astrometric** (are the positions correct?).

6.3.1 Photometric Validation

The scientific validity of the simulation rests on the correctness of its physical models. The radiometric calibration pipeline ensures that the number of electrons generated for any object of a given magnitude is physically correct for the specified instrument and conditions. This means that synthetic photometry performed on the simulated images using standard tools like DAOPHOT [10] would recover the input catalog magnitudes, after accounting for the simulated noise. This ground-truth nature is what makes the generated data invaluable for algorithm testing.

6.3.2 Astrometric Validation

A critical requirement for any scientific image simulator is astrometric accuracy: the pixel coordinates in the FITS file must map precisely to the correct celestial coordinates (RA, Dec) via the World Coordinate System (WCS) header.

To confirm this, a simulated FITS file generated by Astrelle was submitted to the "blind" astrometric solver, **Astrometry.net**. This service analyzes the pattern of stars in an image and, without any prior information, determines the image's true pointing, orientation, and pixel scale.

The simulated image (whose input parameters are shown in Figure 6.4) was successfully "solved." As shown in Figure 6.5, the solution provided by Astrometry.net perfectly matched the input RA, Dec, and pixel scale that were specified in the Astrelle GUI. This test confirms that the simulator's gnomonic projection (RA—TAN, DEC—TAN) and WCS header generation are implemented correctly, ensuring that the positions of all simulated objects are scientifically valid.

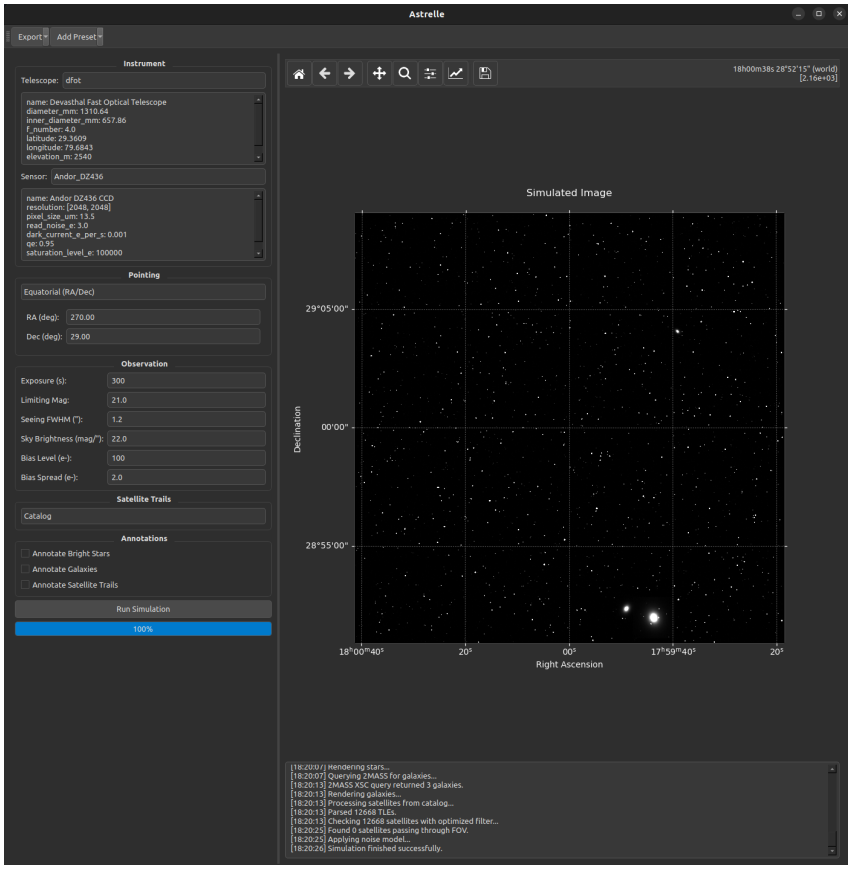

Figure 6.4: Input parameters for the astrometric validation test.

Calibration

Center (RA, Dec):	(270.000, 29.000)
Center (RA, hms):	$17^h\ 59^m\ 59.980^s$
Center (Dec, dms):	+29° 00' 00.275"
Size:	18.1 x 18.1 arcmin
Radius:	0.214 deg
Pixel scale:	0.531 arcsec/pixel
"Orientation", may be incorrect, use at your own risk:	Up is -0.0 degrees E of N
WCS file:	wcs.fits
New FITS image:	new-image.fits
Reference stars nearby (RA,Dec table):	rdls.fits
Stars detected in your images (x,y table):	axy.fits
Stars detected in your images, converted to RA,Dec (FITS table):	image-radec.fits
Correspondences between image and reference stars (table):	corr.fits
Legacy Surveys sky browser:	browse the sky
KMZ (Google Sky):	image.kmz
World Wide Telescope:	view in WorldWideTelescope

Figure 6.5: The corresponding Calibration from Astrometry.net. The solution perfectly matches the input parameters, validating the simulator's WCS.

Chapter 7

Limitations and Future Scope

7.1 Limitations of the Current Model

While Astrelle produces high-fidelity images, it is important to acknowledge the physical and instrumental effects that are not simulated. These represent the primary differences between the current output and real-world imagery.

- **Galaxy Morphology:** The Sersic profile is an idealized, smooth model. It does not reproduce the intricate details of real galaxies, such as spiral arms, dust lanes, HII regions, or central bars. Consequently, all simulated galaxies appear as smooth, elliptical blobs of light, lacking the complex morphology seen in deep astronomical images. State-of-the-art galaxy simulation now uses score-based generative models to produce much more realistic morphologies [4].

- **Point Spread Function (PSF) Complexity:** The simulation uses a simple, circularly symmetric Gaussian PSF. Real PSFs are significantly more complex, often exhibiting asymmetries and diffraction patterns (e.g., Airy rings, diffraction spikes from secondary mirror supports) from the telescope's optics, in addition to aberrations like coma and tracking errors that elongate the PSF [15]. The diffraction pattern of a circular aperture is described by the Airy function [6], which is neglected in the current Gaussian approximation.

- **Satellite Brightness Variability:** The brightness of a catalog satellite is modeled as constant throughout the exposure, based on a simple distance-scaling law. A real satellite's brightness can vary dramatically and rapidly due to its shape, material properties, and changing orientation, often causing bright "glints" or flashes that are not modeled.

- **Detector Artifacts:** The noise model is idealized. It does not include common instrumental artifacts such as high-energy cosmic ray hits, pixel defects (hot/cold pixels, bad columns), or charge transfer inefficiency (CTE) trails in CCDs [16].

- **Astrometric Projection:** The current simulation projects celestial coordinates onto a flat tangent plane (a gnomonic projection) using the WCS. While accurate for small fields of view, this does not account for optical distortions present in real telescope systems, which require more complex polynomial solutions for precise astrometric calibration.

7.2 Future Scope

The successful development of the Astrelle simulator is not the end goal, but rather the foundational first step. The future scope of this project involves two main thrusts: refining the simulator and developing an integrated trail detection system.

1. **Simulator Refinement:** The next iteration of Astrelle will focus on addressing the current limitations. This includes incorporating more complex PSF models that account for diffraction and optical aberrations. While the core astrometric projection has been validated (see Section 5.3.2), a future step is to simulate complex **optical distortions** (as noted in Section 6.1) that require higher-order polynomial terms in the WCS solution, moving beyond the current ideal gnomonic projection.

2. **Integrated Trail Detection Module:** The primary purpose of generating this high-quality synthetic data is to use it as a ground-truth set for a robust trail detection algorithm. This algorithm will be developed and integrated directly into the Astrelle software suite. Instead of a machine-learning approach, we will implement methods based on classical transform models, which are computationally efficient and highly effective for linear feature detection. Specifically, we will explore:

    - **The Radon Transform:** This transform is adept at identifying linear features in an image by integrating pixel values along lines of all possible angles and offsets [12]. Satellite trails will manifest as strong peaks in the Radon space, allowing for their detection.
    - **The Hough Transform:** A similar technique that maps points in the image space to curves in a parameter space. Collinear points, such as those forming a satellite trail, will produce an intersection of many curves at a single point, which can be identified with a peak-finding algorithm [13].

    The resulting module will not only detect trails but also characterize their properties (e.g., width, brightness, orientation) and generate a corresponding mask for use in scientific data pipelines.

Bibliography

[1] Peng, C. Y., Ho, L. C., Impey, C. D., & Rix, H.-W. 2010, AJ, 139, 2097, *Detailed Decomposition of Galaxy Images. II. Beyond Axisymmetric Models.*

[2] Peng, C. Y., Ho, L. C., Impey, C. D., & Rix, H.-W. 2002, AJ, 124, 266, *Detailed Structural Decomposition of Galaxy Images.*

[3] Geda, R., Crawford, S. M., Hunt, L., et al. 2022, JOSS, 7, 4398, *PetroFit: A Python Package for Computing Petrosian Radii and Fitting Galaxy Light Profiles.*

[4] Smith, M. J., Geach, J. E., Jackson, R. A., et al. 2023, MNRAS, 521, 1076, *Realistic galaxy image simulation via score-based generative models.*

[5] Binney, J., & Merrifield, M. 1998, *Galactic Astronomy* (Princeton: Princeton Univ. Press).

[6] Turkyilmazoglu, M. 2012, Phys. Scr., 86, 055004, *The Airy equation and its alternative analytic solution.*

[7] Evans, D. W., Riello, M., De Angeli, F., et al. 2018, A&A, 616, A4, *Gaia Data Release 2: Photometric content and validation.*

[8] Bessell, M., & Murphy, S. 2012, PASP, 124, 140, *Spectrophotometric Libraries, Revised Photonic Passbands, and Zero Points for UBVRI, Hipparcos, and Tycho Photometry.*

[9] Mohan, V., et al. (1999), BASI, 27, 335. *Atmospheric extinction measurements at Devasthal, Naini Tal.*

[10] Stetson, P. B. 1992, in Astronomical Data Analysis Software and Systems I, eds. D. M. Worrall, C. Biemesderfer, & J. Barnes, ASP Conf. Ser., 25, 297, *DAOPHOT II.*

[11] Hasan, I., Tyson, J. A., Saunders, C., & Xin, B. 2023, AJ, 165, 237, *Mitigating Satellite Trails: A Study of Residual Light After Masking.*

[12] Stark, D. V., Grogin, N., Ryon, J., & Lucas, R. 2015, ACS Instrument Science Report 2015-05, *Improved Identification of Satellite Trails in ACS/WFC Imaging Using a Modified Radon Transform.*

[13] Stoppa, F., Groot, P. J., Stuik, R., et al. 2023, A&A, 676, A11, *Automated detection of satellite trails in ground-based observations using U-Net and Hough transform.*

[14] Park, J.-H., Yim, H.-S., Choi, Y.-J., et al. 2018, AdSpR, 62, 119, *OWL-Net: A global network of robotic telescopes for satellite observation.*

[15] Karttunen, H., Kröger, P., Oja, H., et al. (eds.) 2007, *Fundamental Astronomy* (5th ed.; Berlin: Springer).

[16] Howell, S. B. 2006, *Handbook of CCD Astronomy* (2nd ed.; Cambridge: Cambridge Univ. Press).

[17] Romanishin, W. 2002, *An Introduction to Astronomical Photometry Using CCDs* (University of Oklahoma).

[18] ARIES, *3.6m Devasthal Optical Telescope*, https://www.aries.res.in/facilities/astronomical-telescopes/36m-dot. Accessed 2025.

[19] Gaia Collaboration, et al. 2023, A&A, 674, A1, *Gaia Data Release 3: Summary of the content and survey properties.*

[20] Skrutskie, M. F., et al. 2006, AJ, 131, 1163–1183, *The Two Micron All Sky Survey (2MASS).*

[21] Kelso, T.S., CelesTrak. https://celestrak.com. Accessed 2025.

[22] Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and earth satellites. Astrophysics Source Code Library, ascl:1907.024.

[23] Astropy Collaboration, et al. 2018, AJ, 156, 123, *The Astropy Project: Building an Open-science project and status of the v2.0 core package.*

[24] Ginsburg, A., et al. 2019, AJ, 157, 98, *Astroquery: An Astronomical Web-querying Package in Python.*

[25] Astropy Collaboration, et al. 2019, *PyVO: Python access to the Virtual Observatory.* Zenodo. https://doi.org/10.5281/zenodo.3549333

[26] Harris, C. R., et al. 2020, Nature, 585, 357–362, *Array programming with NumPy.*

[27] Virtanen, P., et al. 2020, Nature Methods, 17, 261–272, *SciPy 1.0: fundamental algorithms for scientific computing in Python.*

[28] Hunter, J. D. 2007, CSE, 9, 90-95, *Matplotlib: A 2D Graphics Environment.*

[29] McKinney, W. 2010, in *Proc. 9th Python in Science Conf.*, 51-56, *Data Structures for Statistical Computing in Python.*

[30] Riverbank Computing. (2023). *PyQt6.* https://www.riverbankcomputing.com/software/pyqt/