Approach, Design Decisions & Code Walkthrough

Problem Understanding

The goal of this assignment is to identify **sustained photometric deviations** in satellite observations.
Rather than flagging single noisy points, the objective is to detect meaningful changes in behavior by comparing a **stable historical reference period** with a **current observation window**.

Overall Design

I intentionally avoided machine learning techniques.

This decision was made because:

- The problem is comparative, not predictive
- Explainability is more important than model complexity
- Statistical envelopes are sufficient and more transparent

High-Level Pipeline Flow

1. Load all CSV files belonging to one era
2. Clean invalid or non-physical measurements
3. Convert timestamps and sort data
4. Split data into reference and current windows
5. Build a phase-dependent reference envelope
6. Compare current data against this envelope
7. Group sustained deviations into anomaly events
8. Output results as structured JSON

Note : All eras are processed independently to avoid cross-era bias.

Function-by-Function Explanation

run()

This is the main entry point of the pipeline.

It reads the input configuration, loops over each era, processes them independently, and finally writes all detected anomalies into a single JSON output file.

This structure makes the code easy to run, test, and review. It also reflects how a real monitoring pipeline would be executed.

_process_single_era()

This function contains the full processing logic for one era.

Each era is treated independently because:

- observational conditions may change over time
- reference behavior should not leak across eras
- it prevents artificial normalization

Inside this function:

- raw data is loaded
- cleaned
- split into reference and current windows
- anomalies are detected per NORAD object

_load_era_csvs()

Each era consists of multiple CSV files.

This function:

- reads all CSVs from the era folder
- concatenates them into a single DataFrame

This keeps file-handling logic isolated and prevents clutter in the analytical code.

_clean_data()

This step ensures data quality before analysis.

The following rows are removed:

- missing NORAD ID
- missing timestamps
- missing equatorial phase
- missing magnitude
- negative uncertainty values (if present)

These checks are necessary because invalid values can silently distort statistical behavior downstream.

_prepare_dataframe()

This function converts timestamps to UTC and sorts the data chronologically

Temporal ordering is critical because anomaly detection relies on identifying **consecutive deviations**, not isolated points.

_split_reference_current()

This function splits the cleaned data into:

- a **reference window** (historical baseline)
- a **current window** (period under evaluation)

The split is based strictly on timestamps provided in the input configuration and mirrors how anomaly detection is done in real monitoring systems.

_summarize()

This function collects step-wise data statistics such as:

- number of rows
- null counts per column
- validate data flow
- support reporting
- explain how data volume changes across stages

This is what feeds the “Data Volume Across Processing Stages” plot.

_handle_outliers()

This function removes extreme magnitude outliers from the **reference data only** using an IQR-based filter.

- Reference behavior should represent stable, typical observations
- Extreme values can distort the envelope
- Outliers are not automatically anomalies

_build_reference_profile()

This function builds the expected photometric behavior as a function of equatorial phase.

Key design choices:

- Phase is binned using a fixed bin size
- Median is used as the central tendency
- IQR (Q1–Q3) defines acceptable variation
- Glint-prone phase regions are excluded

This results in a **phase-dependent reference envelope** that is:

- interpretable
- robust
- physically meaningful

_detect_anomalies()

This is the core anomaly detection logic.

Steps involved:

1. Current observations are aligned with reference phase bins
2. Deviation is computed relative to the reference envelope
3. Points exceeding a threshold are flagged
4. Consecutive anomalous points are grouped
5. Only sustained groups are reported as events

A smooth, bounded deviation score is computed to represent anomaly severity. And this avoids binary decisions and allows severity comparison across events.

Visualization Usage

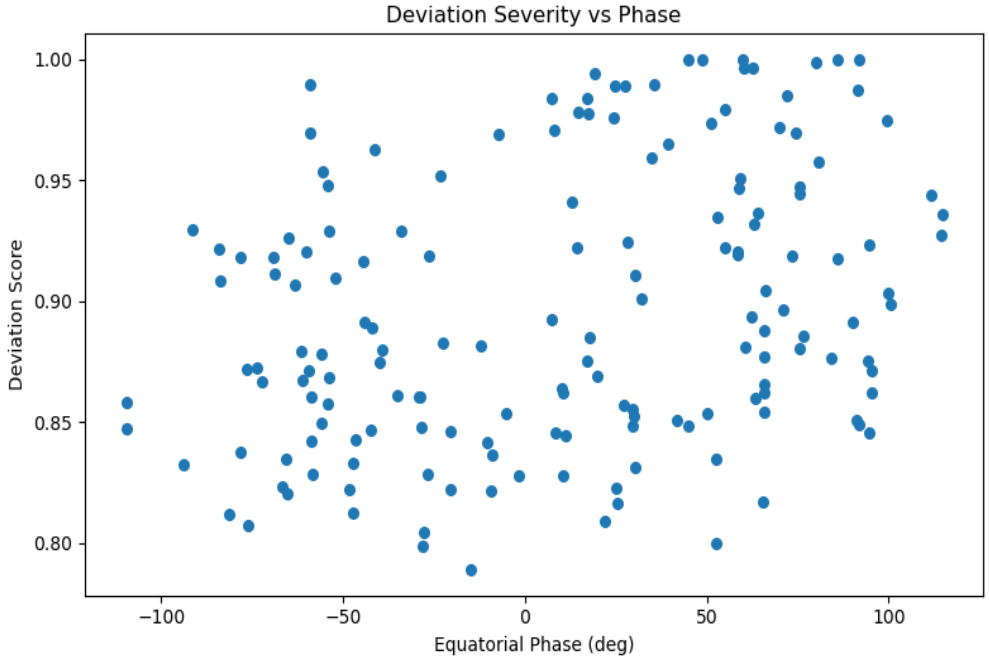
To use this function we have to enable the plot by True (plot_enabled=True)

1. Deviation Severity vs Phase

This plot shows how anomaly severity varies across equatorial phase.

Purpose:

- Validate that deviations are not randomly distributed
- Help reviewers visually connect geometry with severity
- No thresholds or decisions are derived from this plot



2. Data Volume Across Processing Stages

This plot shows how the dataset evolves through the pipeline.

Purpose:

- Demonstrate transparency
- Confirm that cleaning and filtering behave as expected
- Provide confidence in data handling