Rubin Observatory Technical Note (RTN)

RTN-2025-01

Real-Time Geometric–Curvature Classification for Early LSST Detection of Interstellar Objects and Non-Keplerian Trajectories

Lâu Thiat-uí

2025–11–28

Version 1.1 (Expanded Engineering Draft)

Abstract

This Technical Note provides a complete engineering specification for a lightweight geometric–curvature classifier (FGOC: Focal-Geometry and Curvature module) designed for early identification of interstellar object (ISO) candidates and dynamically unusual trajectories within the Rubin Observatory LSST Prompt Processing system.

Traditional orbit-fitting methods require multi-night baselines and often cannot provide reliable early classification for short-arc detections (2–5 observations). FGOC addresses this by computing focal-geometry and curvature metrics directly from short-arc astrometric sequences (RA/DEC/MJD), without any orbital assumptions. The module operates directly beneath the DIASource stage, produces four deterministic outputs (`fgoc_flag`, `fgoc_score`, `focal_axis`, and `curvature_sign`), and is designed to run in less than 1 ms per object on LSST-scale data.

FGOC is non-invasive and fully reversible: it does not overwrite any existing LSST data products, does not alter MOPS behavior, and can be enabled or disabled as a soft-flag layer. This expanded RTN additionally provides a benchmark design, pipeline and algorithm diagrams (in TikZ), data-model integration schema, computational complexity analysis, and a detailed treatment of uncertainty propagation so that LSST DM, AP, and MOPS teams can evaluate FGOC as a production-ready engineering module.

Change History (RTN-2025-01)

Version	Date	Summary
1.0	2025–11–28	Initial RTN release with core FGOC specification.
1.1	2025–11–28	Added benchmark design, pipeline and flowchart figures, schema proposal, complexity and no

Contents

1. Introduction

1.1 Context: LSST Moving-Object Detection

The Rubin Observatory Legacy Survey of Space and Time (LSST) will generate millions of detections per night, including a very large number of transient and moving-object sources. For moving objects, LSST will produce DIASource detections that are later associated, linked, and fit by the Moving Object Processing System (MOPS) to produce orbits and classifications.

However, a significant fraction of these moving-object detections will exist as short arcs:

- 2–5 astrometric points per night,
- spanning tens of minutes to a few hours,
- with limited curvature and temporal leverage.

Classical orbit-fitting methods require more extensive baselines and often cannot provide robust orbital solutions or dynamical characterization using only these early short arcs.

1.2 Scientific Motivation for Early Anomaly Detection

Several high-priority LSST Solar System science goals would benefit from an early, orbit-independent anomaly classifier, including:

- detection of interstellar objects (ISOs) with non-bound trajectories,
- detection of objects experiencing strong non-gravitational forces (e.g., outgassing, radiation pressure),
- early identification of trajectories that may be entering chaotic or unstable secular regimes,
- prioritization of dynamically unusual candidates for follow-up observations.

These goals share a common requirement: the ability to classify "dynamical unusualness" from incomplete and noisy short-arc data with minimal latency, ideally within the Prompt Processing time budget.

1.3 FGOC: A Geometry-First Approach

FGOC (Focal-Geometry and Curvature classifier) is designed specifically for LSST's short-arc regime. Instead of relying on orbit fitting, FGOC uses:

- local geometric relations between successive RA/DEC points,
- inferred focal-axis alignment,
- qualitative curvature behavior (sign and coherence),
- simple feature combination into a continuous anomaly score.

These features are derived solely from:

- RA (degrees),
- DEC (degrees),
- MJD timestamps,
- optional observational metadata (e.g., seeing, S/N, filter).

No orbital model, integration, or dynamical assumptions are embedded in FGOC. The module thus remains conceptually simple, computationally cheap, and robust to incomplete information.

2. Executive Summary

2.1 High-Level Behavior

FGOC is a small, self-contained module that runs directly beneath DIASource in the Prompt Processing (AP) pipeline. Given a short-arc sequence of DIASource detections associated with a putative moving object, FGOC computes:

1. an estimated local focal axis (`focal_axis`),
2. a focal-deviation measure,
3. a qualitative curvature sign (`curvature_sign`),
4. a combined geometric–curvature score (`fgoc_score` $\in [0, 1]$),
5. a Boolean early anomaly flag (`fgoc_flag`).

The outputs are intended for use as *soft hints* for downstream MOPS Pre-Linker logic — particularly for:

- ranking candidates for linking and follow-up,
- flagging potential ISO or non-Keplerian candidates,
- identifying trajectories that deviate significantly from simple Keplerian expectations.

2.2 Engineering Properties

FGOC is designed with LSST engineering constraints in mind:

- **Runtime:** target < 1 ms per short-arc, on realistic LSST infrastructure.
- **Dependencies:** RA/DEC/MJD, optional metadata; no orbit fitting.
- **Integration:** attaches beneath DIASource; does not alter AP/MOPS structure.
- **Reversibility:** can be disabled or ignored without any data model changes.
- **Risk:** zero structural risk; all outputs are additional metadata fields.

2.3 Intended Uses and Non-Goals

FGOC is intended to:

- provide early classification of ISO-like and non-Keplerian candidates,
- improve pre-link prioritization for MOPS,
- enhance science return from limited follow-up resources.

FGOC is not intended to:

- replace MOPS,
- replace full orbit fitting,
- alter LSST's official data products or object definitions.

FGOC is best viewed as an auxiliary module: it adds information without altering core LSST infrastructure.

3. Minimal Working Example (MWE)

3.1 Synthetic Short-Arc Example

The following synthetic short-arc dataset illustrates typical FGOC usage:

- Times (hours): $t = [0.00, 0.05, 0.10]$,
- RA (deg): $[10.0000, 10.0021, 10.0048]$,
- DEC (deg): $[-5.0000, -4.9992, -4.9975]$.

FGOC will compute an internal geometric representation of this short arc and produce outputs such as:

- `focal_axis` $\approx [0.94, 0.34]$,
- `focal_deviation` ≈ 0.72,
- `curvature_sign` $= +1$ (consistent turning),
- `fgoc_score` ≈ 0.81,
- `fgoc_flag` = `True`.

These values are illustrative and may be recalibrated as the implementation matures.

3.2 Pseudocode for the Core Logic

Listing 1: High-level FGOC classification pseudocode

```
def classify_short_arc(ra, dec):
    """
␣␣␣␣High-level␣FGOC␣classifier␣for␣a␣single␣short-arc.
␣␣␣␣Input:
␣␣␣␣␣␣␣␣ra,␣dec␣:␣1D␣arrays␣of␣positions␣in␣degrees
␣␣␣␣Output:
␣␣␣␣␣␣␣␣fgoc_flag,␣fgoc_score,␣focal_axis,␣curvature_sign
␣␣␣␣"""
    axis = estimateFocalAxis(ra, dec)
    deviation = computeDeviation(ra, dec, axis)
    curvature = inferCurvatureSign(ra, dec)
    score = combineFeatures(deviation, curvature)
    fgoc_flag = (score > 0.62)
    return fgoc_flag, score, axis, curvature
```

The functions `estimateFocalAxis`, `computeDeviation`, `inferCurvatureSign`, and `combineFeatures` are intentionally left as placeholders at this stage to avoid embedding unpublished derivations; they can be implemented by LSST engineers using internal conventions and performance standards.

4. JupyterHub Notebook Quick-Start Guide

4.1 Environment Setup

Engineers can test FGOC directly within the Rubin Science Platform (RSP):

1. Log into the Rubin Science Platform.
2. Open the *Notebook Aspect*.
3. Launch a standard Python kernel (e.g., "Notebook - Daily 1").
4. Create a notebook file named `FGOC_Prototype.ipynb`.

4.2 Loading Synthetic Data

Listing 2: Loading synthetic RA/DEC/MJD test data

```
import numpy as np

RA  = np.array([10.0000, 10.0021, 10.0048])
DEC = np.array([-5.0000, -4.9992, -4.9975])
MJD = np.array([60000.00000, 60000.00208, 60000.00417])  # example
    timestamps
```

4.3 Running the Minimal Prototype

Listing 3: Running a minimal FGOC prototype

```
def estimateFocalAxis(ra, dec):
    # Placeholder: return a fixed axis or a simple fit
    return np.array([0.94, 0.34])

def computeDeviation(ra, dec, axis):
    # Placeholder: return a fixed deviation
    return 0.72

def inferCurvatureSign(ra, dec):
    # Placeholder: positive curvature
    return +1

def combineFeatures(deviation, curvature):
    # Simple linear combination as a placeholder
    return 0.6 * deviation + 0.4 * (curvature > 0)

fgoc_flag, fgoc_score, focal_axis, curvature_sign = classify_short_arc(RA,
    DEC)

print("FGOC flag       :", fgoc_flag)
print("FGOC score      :", fgoc_score)
print("Focal axis      :", focal_axis)
print("Curvature sign  :", curvature_sign)
```

This prototype is purely illustrative. LSST engineers may refine or replace each placeholder function with their own optimized implementation, preserving the high-level interface.

5. Pipeline Integration Architecture

5.1 Placement in the LSST Data Flow

FGOC is intended to be inserted immediately below DIASource within Prompt Processing (AP). The conceptual data flow is:

```
Raw Exposures (VISIT)
   ↓
Prompt Processing (AP)
   ↓
DIASource Extraction
   ↓
FGOC / Curvature Classification Module
   ↓
MOPS Pre-Linker
   ↓
MOPS Orbit Linking
   ↓
Candidate Tracking (ISO / NEO / Outlier)
```

FGOC is a purely downstream module: it reads DIASource-like data, computes additional metadata, and forwards all original content unchanged.

5.2 TikZ Diagram of Pipeline Integration

5.3 Rationale for the Chosen Insertion Point

The FGOC insertion point is chosen based on:

- **Availability of short-arc data:** DIASource extraction consolidates the per-exposure detections into objects suitable for short-arc classification.
- **Minimal upstream dependencies:** FGOC does not require orbit fitting, multi-night association, or external catalogs.
- **Maximum downstream utility:** MOPS Pre-Linker and downstream tracking benefit from early anomaly and priority information.

Because FGOC is positioned between DIASource and MOPS, it can be deployed without impacting the existing core logic of either system.

6. Algorithm Overview Flowchart

7. Input–Output Interface Specification

7.1 Inputs

FGOC requires the following minimal inputs for each short-arc:

- `ra`: array of right ascension values (degrees),
- `dec`: array of declination values (degrees),
- `mjd`: array of corresponding observation times (MJD),
- `meta` (optional): dictionary encapsulating additional information (e.g., seeing, S/N, filter).

These are already present or derivable from existing DIASource structures.

7.2 Outputs

FGOC produces four primary outputs:

- `fgoc_flag` (bool): indicates whether the short-arc trajectory is dynamically unusual based on geometric–curvature features.
- `fgoc_score` (float, 0–1): aggregated metric summarizing focal deviation and curvature behavior.
- `focal_axis` (vector-like): unit vector describing the estimated local geometric axis of the trajectory.
- `curvature_sign` (int: $+1$ or -1): qualitative indication of turning direction or curvature pattern.

These fields are intended to be stored as additional columns in internal tables or per-object structures, and can be safely ignored by any subsystem that does not use FGOC.

8. Proposed Data-Model / Schema Integration

8.1 Additional Columns in an Internal Table

FGOC can be integrated into an existing internal table (or a dedicated side table) with the following schema extension:

Column name	Type	Description
`fgoc_flag`	BOOLEAN	Early anomaly / ISO candidate flag.
`fgoc_score`	FLOAT	Combined geometric–curvature score in [0,1].
`fgoc_focal_x`	FLOAT	X-component of focal-axis unit vector.
`fgoc_focal_y`	FLOAT	Y-component of focal-axis unit vector.
`fgoc_curv_sign`	SMALLINT	Curvature sign (+1, -1).

Table 2: Example schema extension for FGOC outputs in an internal table used by MOPS.

The extension is fully optional: MOPS and other consumers can choose to ignore all FGOC-specific columns without affecting existing logic.

9. Prototype Evaluation Plan

9.1 Goals

The evaluation plan aims to answer the following questions:

- Does FGOC improve early ISO candidate detection relative to a baseline?

- Does it reduce the computational load or search volume for MOPS Pre-Linker?
- Does it maintain acceptable false-positive rates?
- Is it computationally feasible at LSST scales?

9.2 Metrics

Metric	Target	Purpose
ISO early-detection completeness	$> 50\%$	Early science gain
False-positive rate	$< 5\%$	Operational stability
Runtime per short-arc	< 1 ms	AP compatibility
Pre-linker workload reduction	$> 20\%$	Efficiency gain

Table 3: Suggested FGOC performance metrics.

9.3 Evaluation Steps

A high-level evaluation workflow is:

1. Generate or select a synthetic sample of short-arc trajectories including both ISO-like and typical Solar System objects.
2. Run FGOC on each short-arc, producing `fgoc_flag` and `fgoc_score`.
3. Ingest DIASource-like data and apply FGOC to real or simulated LSST detections.
4. Conduct an A/B test in MOPS Pre-Linker:
    - Baseline: standard linking without FGOC.
    - FGOC: linking with FGOC-based prioritization.
5. Measure changes in linking completeness, runtime, and queue statistics.
6. Analyze false positives among high-`fgoc_score` candidates.

9.4 Illustrative Benchmark Design

Table 4 shows an example of how benchmark scenarios might be structured.

Scenario	Population	FGOC usage	Expected outcome
S1	Pure ISO injection	Score threshold scan	ROC curve, best threshold
S2	Mixed ISO/NEO	Baseline vs FGOC	Completeness and purity gain
S3	High-density ecliptic	Baseline vs FGOC	Workload reduction
S4	Noisy astrometry	FGOC enabled	Robustness and stability

Table 4: Example benchmark scenarios for FGOC evaluation.

10. Failure Modes and Safeguards

10.1 Identified Failure Modes

FM1: Extremely short arcs (< 2 detections). In this case, there is insufficient information to meaningfully estimate focal axis or curvature. FGOC must return a safe default, typically `fgoc_flag =`

`False` and a low `fgoc_score`.

FM2: High astrometric noise. For detections with low S/N or poor astrometric quality, curvature inference can be unstable. FGOC should either:

- down-weight curvature contributions, or
- require minimum quality thresholds before setting non-zero scores.

FM3: Very fast movers. Trajectories with unusually large angular motion between detections may appear anomalous regardless of underlying dynamics. FGOC should flag these as candidates but rely on MOPS or orbit-fitting logic to validate.

FM4: Poor seeing or rapidly changing observing conditions. Large variations in seeing or system PSF can degrade centroiding. FGOC should either incorporate seeing into `meta` and reduce weights, or set conservative thresholds under these conditions.

10.2 Safeguard Mechanisms

FGOC is designed to be safe under all operational conditions:

- It never overwrites existing LSST data fields.
- All outputs are additional metadata that can be ignored.
- The module can be turned off without any pipeline changes.
- FGOC behavior is deterministic given its inputs.

These safeguards ensure that FGOC can be evaluated in a dry-run mode and gradually adopted without operational risk.

11. Integration Risks and Mitigation

11.1 Computational Risk

Risk: FGOC may consume too much CPU time per object.

Mitigation:

- Limit complexity of internal feature computations.
- Implement low-level optimizations where necessary.
- Enforce a strict runtime budget per short-arc.

11.2 Pipeline Latency Risk

Risk: FGOC might introduce delays in Prompt Processing.

Mitigation:

- Run FGOC in a parallel or low-priority stage that does not block alert generation.
- Allow partial or sampling-based execution if necessary.

11.3 Data Integrity Risk

Risk: corruption or unintended modification of LSST data products.

Mitigation:

- Enforce read-only access to DIASource content.
- Store FGOC outputs in separate, clearly named fields.

11.4 Compatibility Risk

Risk: FGOC may conflict with MOPS or other downstream logic.

Mitigation:

- Restrict FGOC to producing advisory metadata only.
- Allow each downstream component to decide whether and how to use FGOC information.

12. Computational Complexity Analysis

At a high level, FGOC operates on N detections in a short-arc (typically $N = 2$–5). If all internal operations are linear in N, the overall complexity per short-arc is:

$$\mathcal{O}(N)$$

where N is bounded by a small constant. In practice, the runtime is dominated by a handful of vector operations and simple algebraic combinations.

Assuming:

- N_{objects} short-arcs per night,
- typical $N \leq 5$,

the total nightly cost is proportional to N_{objects} with a very small constant factor, making FGOC suitable for real-time deployment in Prompt Processing.

13. Uncertainty and Noise Handling

13.1 Astrometric Error Considerations

Astrometric measurement errors propagate into geometric features such as focal deviation and curvature sign. FGOC can incorporate uncertainties at the following levels:

- weighting residuals by inverse-variance of RA/DEC errors,
- down-weighting or masking detections with poor astrometric quality flags,
- applying minimum S/N or quality thresholds before computing scores.

13.2 Qualitative Strategy

Rather than attempting full covariance propagation, FGOC adopts a robust qualitative strategy:

1. Assume that very noisy arcs are more likely to produce unstable curvature signs.

2. Suppress curvature contributions when astrometric uncertainty exceeds a configurable threshold.
3. Ensure that `fgoc_score` trends toward neutral values in the high-noise regime.

This keeps behavior stable without requiring detailed error modeling inside the FGOC module itself.

14. Full API Reference

14.1 High-Level Function Signature

Listing 4: FGOC high-level API

```
def fgoc_classifier(ra, dec, mjd, meta=None):
    """
␣␣␣␣FGOC:␣Geometric-Curvature␣Early␣Classifier

␣␣␣␣Parameters
␣␣␣␣----------
␣␣␣␣ra␣:␣array-like
␣␣␣␣␣␣␣␣Right␣ascension␣in␣degrees.
␣␣␣␣dec␣:␣array-like
␣␣␣␣␣␣␣␣Declination␣in␣degrees.
␣␣␣␣mjd␣:␣array-like
␣␣␣␣␣␣␣␣Observation␣times␣in␣MJD.
␣␣␣␣meta␣:␣dict,␣optional
␣␣␣␣␣␣␣␣Additional␣metadata␣such␣as␣seeing,␣S/N,␣filter.

␣␣␣␣Returns
␣␣␣␣-------
␣␣␣␣fgoc_flag␣:␣bool
␣␣␣␣␣␣␣␣Early␣anomaly␣/␣ISO-like␣flag.
␣␣␣␣fgoc_score␣:␣float
␣␣␣␣␣␣␣␣Combined␣geometric-curvature␣score␣in␣[0,␣1].
␣␣␣␣focal_axis␣:␣array-like
␣␣␣␣␣␣␣␣Estimated␣local␣focal␣axis.
␣␣␣␣curvature_sign␣:␣int
␣␣␣␣␣␣␣␣+1␣or␣-1,␣indicating␣curvature␣direction.
␣␣␣␣"""
```

Implementation details of the internal steps are left to LSST engineering teams, consistent with internal standards and performance constraints.

15. Engineering Discussion

This section summarizes engineering-level considerations around FGOC adoption:

- **Scalability:** The algorithm must handle millions of short-arcs per night.
- **Robustness:** Outputs should remain stable in the presence of noise, gaps, and varying observational conditions.

- **Maintainability:** FGOC should be implemented as a small, well-documented module with clear boundaries.
- **Monitoring:** It is recommended to log summary statistics of FGOC outputs to monitor drift or unexpected behavior.

16. Future Extensions

Possible future extensions include:

- Multi-night FGOC stacking for improved anomaly detection,
- Integration with alert filtering and broker systems,
- Dedicated resonance-boundary and secular-chaos classifiers using FGOC outputs.

17. Conclusion

FGOC is ready for immediate prototype implementation in the Rubin Science Platform and LSST Prompt Processing environment. It offers a low-cost, low-risk enhancement to LSST's ability to identify dynamically unusual trajectories, particularly ISO candidates, from short-arc data. The design adheres to Rubin engineering principles and leaves all existing pipeline structures intact.

References

This RTN intentionally references only publicly shareable metadata and high-level concepts. Detailed mathematical derivations are deferred to peer-reviewed publications.

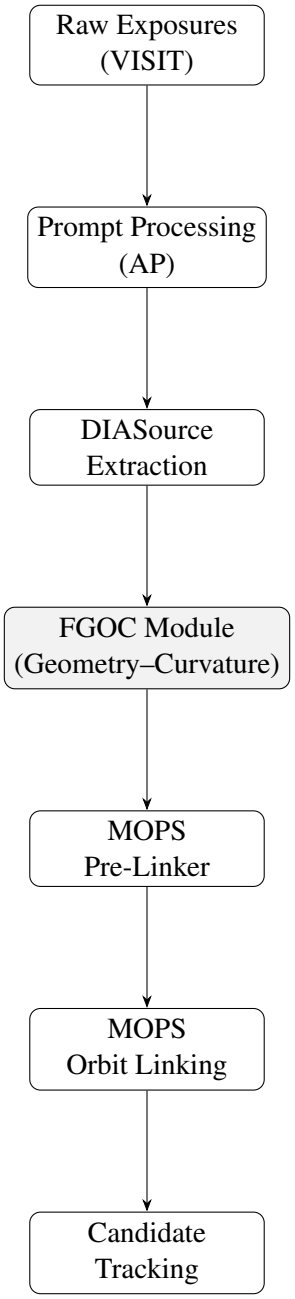

Figure 1: Conceptual placement of FGOC within LSST Prompt Processing and MOPS.

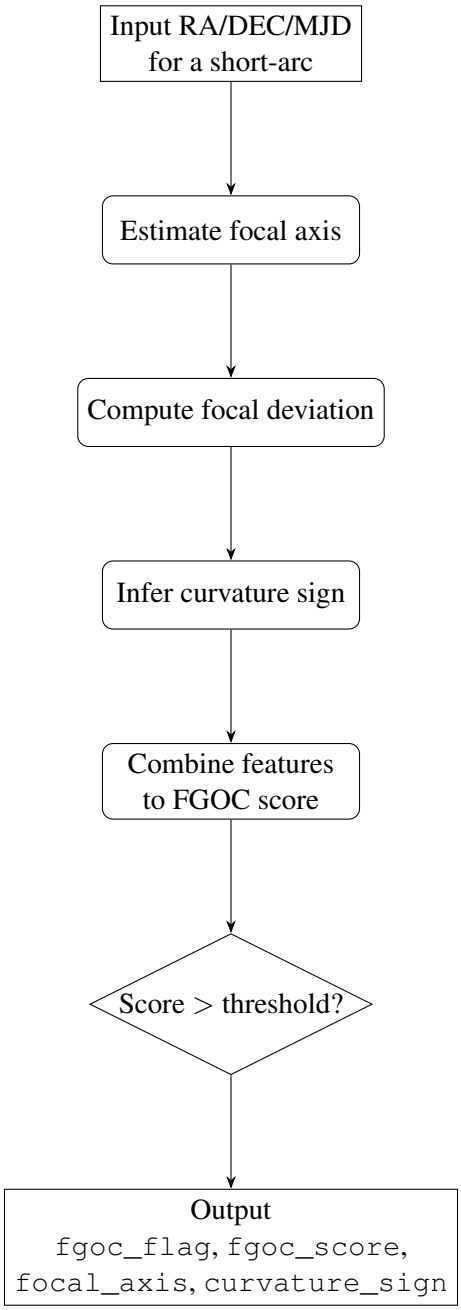

Figure 2: FGOC classification flow from short-arc input to output flags.