Multi-Sensor Track Correlation for Missile Defense: Position-Velocity Fusion and Threat Classification

FORGE Research • Technical Report

forge-track-correlator — Multi-Sensor Track Correlation Engine

April 2026

Abstract. Effective ballistic missile defense requires the fusion of heterogeneous sensor data into coherent, maintainable tracks that support timely threat assessment and engagement decisions. We present the design and algorithms of the forge-track-correlator, a multi-sensor track correlation engine developed for the FORGE framework. The system ingests detection reports from overhead infrared (OPIR) and ground-based radar sensors via an Apache Kafka pipeline, correlates them into unified tracks using a position-velocity scoring metric with a fixed gating threshold, and assigns threat levels on a 1–5 scale ranging from debris/decoy through ICBM. We describe the correlation algorithm, track lifecycle management, quality metrics, stale-track cleanup, and threat classification criteria. Performance considerations and validation approaches for the correlation engine are discussed. The architecture is designed for real-time operation in a streaming data environment with bounded computational cost per detection.

Table of Contents

1. Introduction

Ballistic missile defense (BMD) systems operate in an environment characterized by multiple simultaneous threats, heterogeneous sensor modalities, and severe time

constraints. A central challenge is track correlation: the process of associating incoming sensor detections with the correct existing tracks—or recognizing that a detection represents a new object—so that downstream consumers can reason about discrete, coherent objects rather than a stream of unassociated observations.

Without effective correlation, a single missile might appear as multiple phantom tracks, debris clouds could fragment tracking into hundreds of spurious entries, and sensor handoff events (e.g., from OPIR to radar) would create duplicate tracks that confuse engagement planning. Conversely, overly aggressive association can merge distinct objects into a single track, masking divergent threats.

1.1 The Track Correlation Problem

The multi-target multi-sensor track correlation problem is a well-studied but unsolved problem in the general case. Formally, given a set of existing tracks $T = \{t_1, \ldots, t_n\}$ and a new detection d, the correlation engine must decide whether d should be associated with some existing t_i or whether it represents a new object requiring a new track.

This is a variant of the data association problem described comprehensively by Bar-Shalom [1] and Blackman [2]. In dense threat environments with maneuvering targets, the combinatorial explosion of possible associations makes exact solutions (e.g., joint probabilistic data association, multiple hypothesis tracking) computationally expensive. Practical BMD systems typically employ gated nearest-neighbor approaches with carefully chosen metrics [3].

The FORGE track correlator adopts this pragmatic approach: a deterministic, position-velocity scoring metric with a fixed gating threshold that provides predictable, bounded-cost correlation decisions suitable for real-time streaming operation.

1.2 Scope of This Paper

This paper describes the architecture and algorithms of the `forge-track-correlator`, the multi-sensor track correlation engine within the FORGE framework. We cover:

- The sensor data sources and their characteristics (Section 2)
- The position-velocity correlation algorithm with gating (Section 3)
- Track lifecycle management, quality metrics, and cleanup (Section 4)

- Threat level classification on a 1–5 scale (Section 5)
- The Kafka-based fusion architecture and FORGE integration (Section 6)
- Performance characteristics and validation approaches (Sections 7–8)

2. Sensor Data Sources

The forge-track-correlator operates on detection reports from two primary sensor modalities: overhead persistent infrared (OPIR) satellites and ground-based radar systems. These modalities have fundamentally different observation characteristics, making their fusion both valuable and challenging.

2.1 Overhead Persistent Infrared (OPIR)

OPIR sensors detect the thermal signature of missile plumes during the boost phase from satellite platforms. Their key characteristics include:

Attribute	Characteristic
Observation phase	Boost phase (primary), early midcourse
Detection basis	Infrared signature of exhaust plume
Position accuracy	Moderate (angular measurement from orbit)
Velocity estimation	Indirect, derived from sequential position fixes
Coverage	Wide-area, persistent (geostationary or HEO)
Update rate	Periodic scans; typical frame rates of 1–10 Hz
Limitations	Signature fades post-burnout; limited altitude/range resolution

OPIR provides the earliest detection of launches, giving critical warning time. However, track accuracy degrades once the missile's motor burns out and the infrared signature diminishes.

2.2 Ground-Based Radar

Radar systems provide precision tracking during the midcourse and terminal phases. Their characteristics complement OPIR:

Attribute	Characteristic
Observation phase	Midcourse and terminal
Detection basis	Radar cross-section (RCS) reflection
Position accuracy	High (range and angle precision)
Velocity estimation	Doppler-derived; high accuracy
Coverage	Finite field of regard; horizon-limited
Update rate	Typically higher than OPIR (10–50 Hz)
Limitations	Horizon constraint delays first detection; vulnerable to decoys in midcourse

Radar becomes the primary precision tracking source once the target enters the radar field of regard. The high update rate and Doppler velocity measurement make radar detections particularly valuable for refining track state estimates.

2.3 Complementarity and Challenges

The fundamental complementarity is temporal: OPIR sees first, radar tracks best later. Effective sensor fusion requires seamless handoff between these modalities. The correlation engine must handle:

- **Differing coordinate frames and accuracies:** OPIR detections have larger position covariance than radar detections. A correlation metric must accommodate both without bias.

- **Temporal gaps:** Between OPIR burnout and radar pickup, tracks may coast with no updates. The correlation algorithm must tolerate growing position uncertainty during these gaps.
- **Signature changes:** The same object appears as a bright IR point source to OPIR but as an RCS-dependent radar return. There is no direct feature matching; correlation relies entirely on kinematic state.
- **Correlation density:** During raids, multiple closely-spaced objects (penetration aids, debris, RVs) create ambiguous association neighborhoods.

3. Correlation Algorithm

The core of the forge-track-correlator is a deterministic position-velocity scoring algorithm that gates incoming detections against existing tracks and assigns them to the best-matching track (or creates a new track if no match is found).

3.1 Position-Velocity Scoring

For each incoming detection d with position (x_d, y_d, z_d) and velocity (vx_d, vy_d, vz_d), and each existing track t_i with predicted position (x_i, y_i, z_i) and velocity (vx_i, vy_i, vz_i), the algorithm computes:

$$\Delta pos = \sqrt{((x_d - x_i)^2 + (y_d - y_i)^2 + (z_d - z_i)^2)}$$

$$\Delta vel = \sqrt{((vx_d - vx_i)^2 + (vy_d - vy_i)^2 + (vz_d - vz_i)^2)}$$

$$score = \Delta pos / 100 + \Delta vel$$

The position difference is normalized by dividing by 100, reflecting an implicit assumption that a 100-unit position difference is of comparable discriminative weight as a 1-unit velocity difference. This normalization creates a combined score in a unified scale that balances positional and kinematic proximity.

> **Design note:** The 100-unit position normalization factor encodes domain knowledge about the relative scales of position and velocity uncertainties in the FORGE coordinate system. In practice, position differences between a track's predicted state and a correlated detection are typically in the hundreds of units (due to prediction error and sensor noise),

> while velocity differences are typically in single-digit units. The factor equalizes their contribution to the final score.

3.2 Gating and Match Threshold

A fixed gating threshold of **1.0** is applied:

$$\text{if score} < 1.0 \rightarrow \text{detection matches track } t_i$$

$$\text{if score} \geq 1.0 \text{ for all tracks} \rightarrow \text{create new track}$$

When multiple tracks satisfy the gating threshold, the track with the lowest score receives the detection (nearest-neighbor association). This greedy assignment avoids the computational overhead of global optimization while producing effective results in typical BMD scenarios where tracks are well-separated relative to the gating threshold.

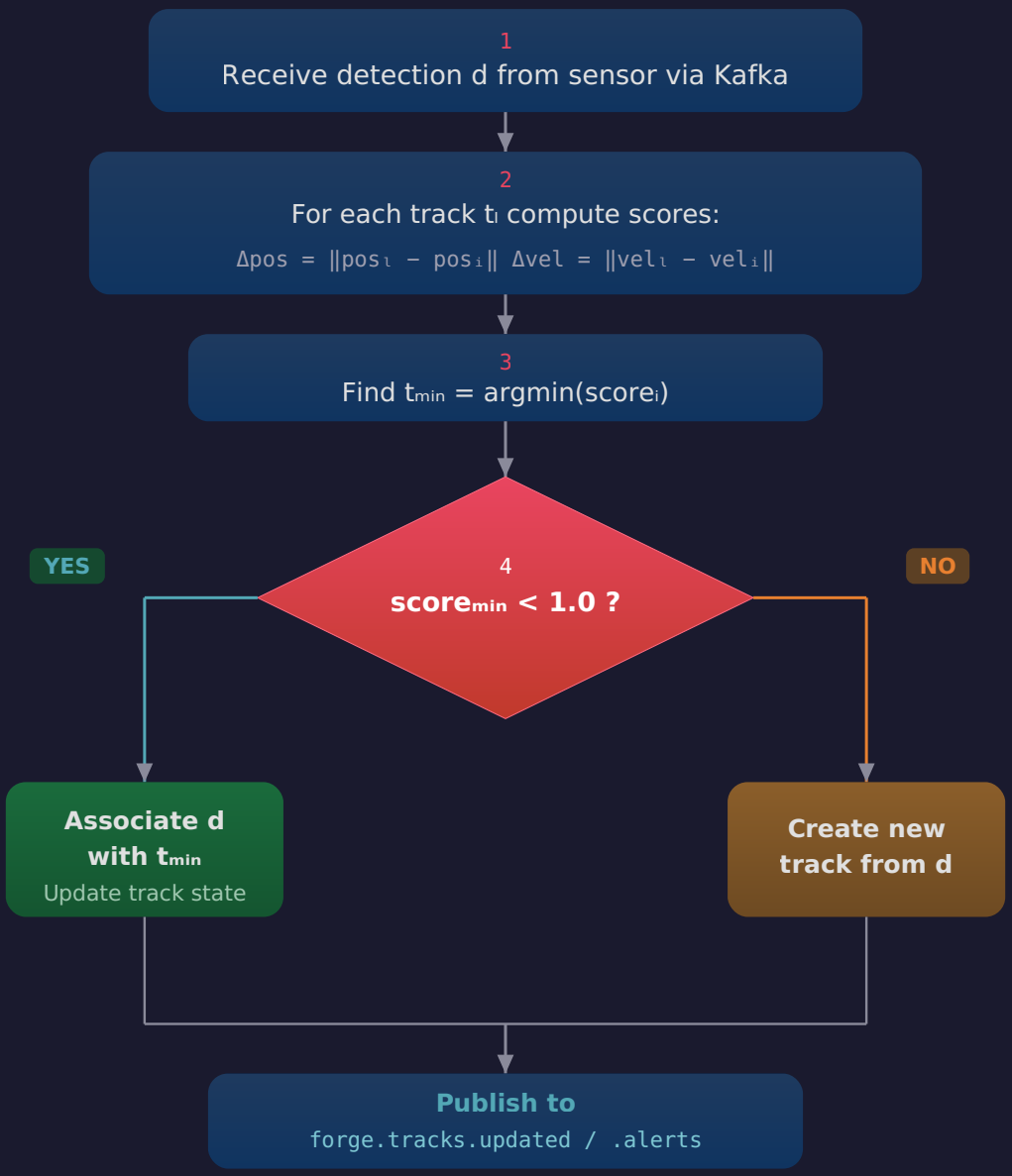
The threshold of 1.0 represents a design trade-off:

- **Lower threshold** → fewer false associations but increased risk of track fragmentation (splitting one object into multiple tracks)
- **Higher threshold** → fewer fragmentations but increased risk of track merging (combining distinct objects)

In the FORGE operating environment, track fragmentation is generally more harmful than occasional merging because downstream threat assessment operates on track counts. A threshold of 1.0 provides a balanced operating point that can be tuned for specific mission profiles.

3.3 Algorithm Walkthrough

The complete correlation procedure for each incoming detection is:



The algorithm operates in O(n) time per detection where n is the number of active tracks. In typical BMD scenarios, n is bounded by the number of objects in the battlespace (tens to low hundreds), making this linear scan feasible without spatial indexing structures. For extremely dense environments, a spatial index (e.g., kd-tree or grid) could be added to reduce the average case, though the worst case remains linear.

4. Track Management

4.1 Track Lifecycle

Each track in the system follows a defined lifecycle:

Figure 1: Track lifecycle state machine

Initiated: A track is created when a detection fails to correlate with any existing track (score ≥ 1.0 for all tracks). The initial track state is populated directly from the detection's position, velocity, and sensor metadata. A single-detection track is tentative; it requires at least one subsequent correlated detection to be promoted to Active status.

Active: A track that has been updated by at least one additional detection since initiation. Active tracks are the primary output of the correlation engine and feed downstream threat assessment and engagement planning.

Stale: A track that has not received a correlated detection within a configurable time threshold. Stale tracks are retained in the track store but flagged for downstream consumers. Stale status allows temporary sensor gaps (e.g., during OPIR-to-radar handoff) without premature track deletion.

Deleted: A stale track that exceeds a second, longer time threshold is permanently removed from the track store. Deletion reclaims resources and prevents stale tracks from degrading correlation performance for new detections.

4.2 Quality Metrics

Each track maintains quality metrics that inform downstream consumers about the reliability of the track state estimate:

Metric	Description
Update count	Total number of detections correlated to this track. Higher counts indicate more persistent observations.
Time since last update	Elapsed time since the most recent correlated detection. Growing values indicate degrading track currency.

Contributing sensors	Set of sensor types (OPIR, Radar) that have contributed detections. Multi-sensor tracks are generally higher quality than single-sensor tracks.
Mean score	Average correlation score across all updates. Low mean scores indicate consistent, well-predicted tracks; high scores suggest marginal associations.
Track age	Total time since track initiation.

These metrics support downstream decision-making. For example, an engagement planner might require a minimum update count and multi-sensor confirmation before committing an interceptor, while a warning system might alert on any track with a high threat level regardless of update count.

4.3 Stale Track Cleanup

The automatic stale track cleanup mechanism prevents unbounded growth of the track store:

```
For each track t_i:
  If (current_time - t_i.last_update_time) > STALE_THRESHO
    Mark t_i as stale
  If (current_time - t_i.last_update_time) > DELETE_THRESH
    Remove t_i from track store
```

The stale threshold and delete threshold are configurable parameters that should be set based on the expected sensor revisit rates and threat timeline. Typical values might be:

- **Stale threshold:** 30–60 seconds (allows for sensor handoff gaps)
- **Delete threshold:** 5–15 minutes (allows for temporary radar shadow zones)

The cleanup runs periodically rather than on every detection, reducing per-detection processing overhead. A reasonable cadence is once per second or once per batch of detections.

5. Threat Classification

Once a detection is correlated to a track (or a new track is created), the system assigns a threat level that drives downstream alerting and engagement prioritization. The forge-track-correlator uses a 1–5 ordinal scale:

5.1 Threat Level Scale

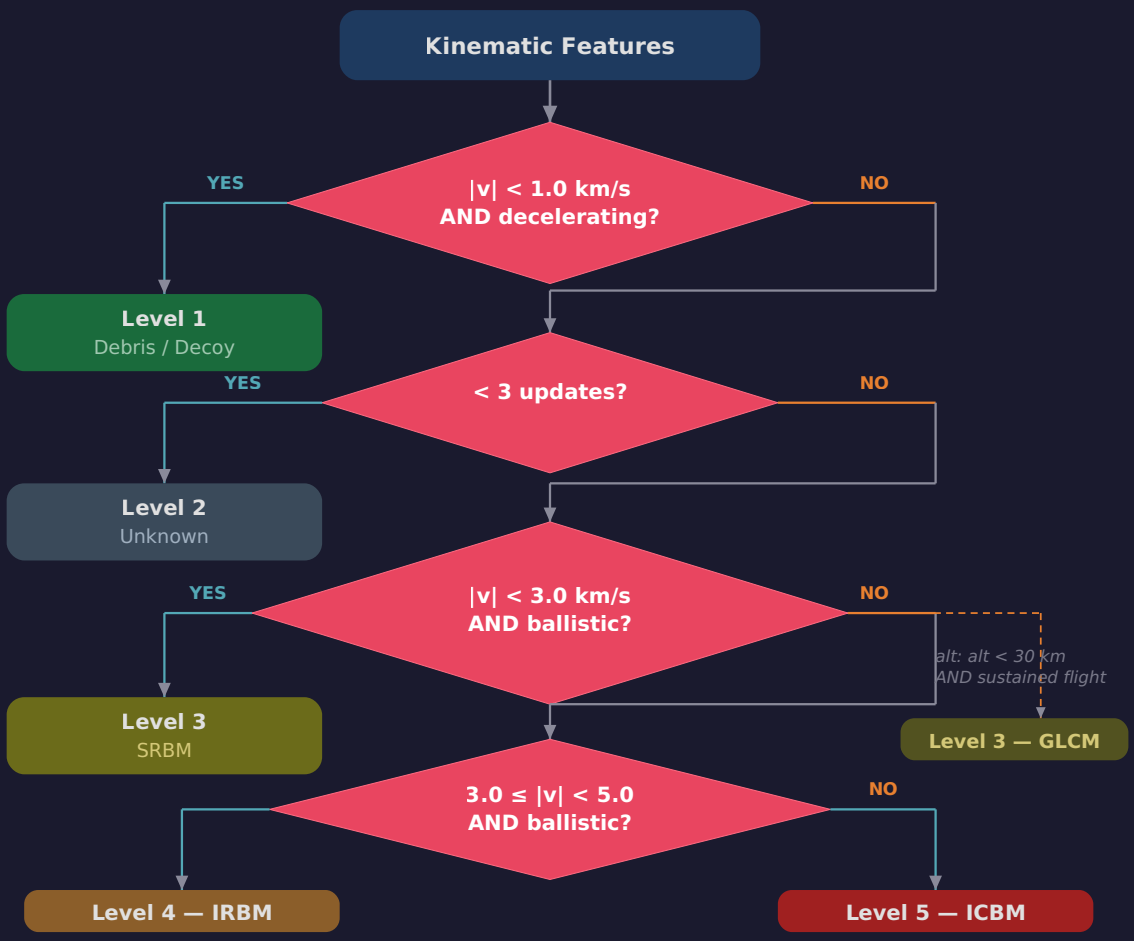
Level	Classification	Description
1	Debris / Decoy	Objects exhibiting non-threatening kinematics: low velocity, decelerating, ballistic debris, or objects with signatures inconsistent with a powered flight path.
2	Unknown	Insufficient data to classify. Track exists but lacks sufficient observations or kinematic discrimination to assign a higher level. Default for new single-detection tracks.
3	SRBM / GLCM	Short-Range Ballistic Missile or Ground-Launched Cruise Missile. Characterized by lower apogee, shorter range trajectories, and moderate velocities consistent with theater-level threats.
4	IRBM	Intermediate-Range Ballistic Missile. Higher apogee and velocity than SRBM; trajectories consistent with ranges of 3,000–5,500 km.
5	ICBM	Intercontinental Ballistic Missile. Highest apogee, highest velocity, longest range trajectories. Characterized by boost-phase kinematics consistent with ranges exceeding 5,500 km.

5.2 Classification Criteria

Threat classification is based primarily on kinematic features extracted from the track state:

- **Velocity magnitude:** The total velocity $|v| = \sqrt{(vx^2 + vy^2 + vz^2)}$ provides the primary discriminant. ICBM-class objects have significantly higher burnout velocities (typically >6 km/s) than SRBMs (<3 km/s) or cruise missiles (subsonic to low supersonic).
- **Apogee estimate:** Projected peak altitude of the trajectory, derived from current position and velocity. Higher apogees correlate with longer-range missiles.
- **Flight path angle:** The elevation angle of the velocity vector relative to the local horizontal. Steep angles during boost are characteristic of ballistic trajectories; shallow, sustained angles suggest cruise missile profiles.
- **Acceleration profile:** If available, the magnitude and direction of acceleration distinguish powered flight (thrust) from ballistic coast and from decelerating debris.
- **Sensor signature:** OPIR intensity profiles and RCS characteristics from radar can support decoy discrimination (Level 1).

The classification decision follows a rule-based approach: a series of thresholds on the above features maps to a threat level. For example:



> **Note:** The specific velocity and altitude thresholds above are illustrative. Operational thresholds are classified and domain-specific. The forge-track-correlator exposes these thresholds as configurable parameters to support different threat libraries and operational contexts.

5.3 Dynamic Reassessment

Threat levels are not static. As a track accumulates more observations, its classification is reassessed:

- A newly created track defaults to Level 2 (Unknown) until sufficient kinematic data is available.
- As velocity estimates stabilize with additional detections, the threat level is promoted to 3, 4, or 5 as appropriate.
- If a track's kinematic profile changes (e.g., detected deceleration or signature change consistent with a decoy), the threat level can be demoted to 1.

- Downstream consumers are notified of threat level changes through the track update messages published to the Kafka output topic.

This dynamic reassessment prevents premature high-confidence classifications that could misallocate defensive resources, while also ensuring that genuinely high-threat objects are rapidly promoted as evidence accumulates.

6. Fusion Architecture

6.1 Kafka Pipeline

The forge-track-correlator operates as a consumer within an Apache Kafka streaming pipeline. Kafka provides the decoupled, fault-tolerant messaging backbone that connects sensors, the correlation engine, and downstream consumers:

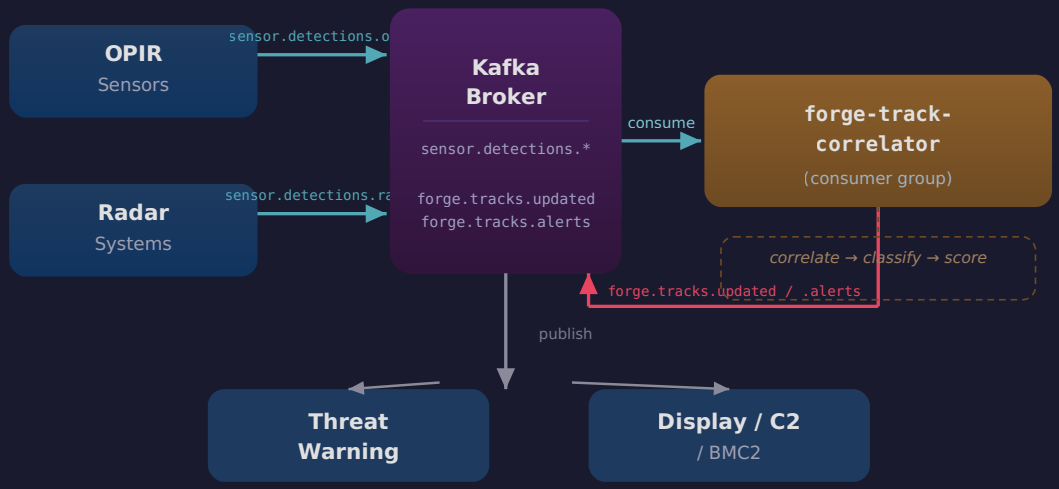


Figure 2: Kafka-based fusion pipeline architecture

Key architectural properties of the Kafka integration:

- **Topic partitioning:** Sensor detection topics are partitioned by region or sensor ID, enabling parallel consumption and horizontal scaling of the correlator.

- **Consumer group:** The correlator operates as a Kafka consumer group, allowing multiple correlator instances to share the detection load for fault tolerance and throughput scaling.
- **At-least-once delivery:** Kafka provides at-least-once delivery semantics. The correlator's idempotent track update logic (applying the same detection twice does not corrupt state) handles potential duplicate deliveries gracefully.
- **Backpressure:** If the correlator falls behind (e.g., during a raid surge), Kafka retains unprocessed messages, allowing the correlator to catch up rather than dropping detections.

6.2 FORGE Integration

The forge-track-correlator is one component within the larger FORGE framework, which provides:

- A common data model for sensor detections, track states, and threat assessments
- Shared coordinate reference systems and time standards
- Configuration management and deployment infrastructure
- Monitoring and health-check endpoints for operational awareness

The correlator consumes standardized detection messages and produces standardized track messages, enabling plug-and-play integration with other FORGE components (e.g., track predictors, engagement planners, situational display systems).

6.3 Data Flow

The end-to-end data flow for a single detection:

1. A sensor (OPIR or radar) produces a detection report containing position, velocity, timestamp, sensor ID, and signal metadata.
2. The detection is published to the appropriate Kafka topic (`sensor.detections.opir` or `sensor.detections.radar`).
3. The forge-track-correlator consumes the detection from the Kafka topic.
4. The correlation algorithm (Section 3) runs, producing either an update to an existing track or a new track creation.
5. Threat classification (Section 5) is (re)evaluated for the affected track.
6. Track quality metrics (Section 4.2) are updated.

7. The updated track state is published to the output Kafka topic (`forge.tracks.updated`), and threat level alerts are published to `forge.tracks.alerts` .
8. Downstream consumers (warning systems, C2 displays, engagement planners) consume the track updates.

7. Performance Considerations

The forge-track-correlator is designed for real-time operation under the computational and latency constraints of a missile defense engagement timeline. Key performance characteristics include:

Computational Complexity

The correlation algorithm scales linearly with the number of active tracks. For each detection, the system computes scores against all n tracks, performing O(n) distance calculations. Each calculation involves 6 subtractions, 6 multiplications, 2 square roots, and 1 division—all constant-time floating-point operations. The total cost per detection is O(n) with a small constant factor.

For a typical battlespace with 50–200 active tracks and a combined sensor update rate of 10–100 detections per second, the correlator must perform 500–20,000 score computations per second—well within the capability of modern server hardware.

Memory

Each track stores its state vector (6 doubles for position and velocity), metadata (sensor IDs, timestamps, quality metrics), and classification state. A typical track record is on the order of 200–500 bytes. With 200 active tracks, the total track store is approximately 40–100 KB—easily cache-resident.

Latency

The critical path for a detection is: Kafka consumption → score computation → track update → Kafka publication. The score computation and track update are microsecond-scale operations. End-to-end latency is dominated by Kafka network and serialization overhead, typically 1–10 ms in a well-configured deployment.

Scaling Considerations

For environments exceeding the single-instance capacity:

- **Spatial partitioning:** Partition tracks by geographic region, with separate correlator instances per partition. Kafka consumer groups naturally support this partitioning.
- **Spatial indexing:** Replace the linear scan with a kd-tree or uniform grid for O(log n) or O(1) average-case lookup when n grows into the thousands.
- **Batch processing:** Process detections in micro-batches rather than individually, amortizing Kafka overhead and enabling vectorized score computation.

8. Validation

Validating a track correlation system requires testing across multiple dimensions: correctness of the correlation logic, robustness under realistic operating conditions, and performance under load.

Unit and Algorithmic Testing

- **Deterministic test cases:** Known position/velocity pairs with pre-computed scores verify the scoring formula implementation.
- **Gate boundary cases:** Detections exactly at the score threshold (1.0) test the association vs. new-track decision boundary.
- **Multi-target scenarios:** Closely-spaced tracks with ambiguous associations test nearest-neighbor selection logic.
- **Track lifecycle:** Tests that verify correct transitions between Initiated, Active, Stale, and Deleted states under controlled time advancement.

Scenario-Based Integration Testing

- **Single missile track:** A notional missile trajectory from boost through midcourse, with simulated OPIR detections transitioning to radar detections, verifies sensor handoff and track continuity.
- **Raid scenario:** Multiple simultaneous launches with different threat levels test correlation under density and verify that distinct tracks are maintained.
- **Debris field:** Post-intercept or post-burst debris tests verify that fragmentation products are tracked separately and classified as Level 1 (Debris).

- **Sensor dropout:** Periods of no detections (simulating sensor outage) test stale track handling and track re-association when detections resume.

Monte Carlo Analysis

For robustness validation, Monte Carlo methods inject stochastic sensor noise (position and velocity errors drawn from sensor-specific distributions) and measure:

- **Track fragmentation rate:** Fraction of true objects that are split into two or more tracks.
- **Track swap rate:** Fraction of detections associated with the wrong track when tracks cross.
- **False track rate:** Number of tracks created that do not correspond to any real object.
- **Threat classification accuracy:** Agreement between assigned and ground-truth threat levels.

These metrics allow quantitative comparison of the current gating threshold (1.0) against alternative values, supporting threshold optimization for specific operational scenarios.

Comparison to Baseline

The fixed-threshold nearest-neighbor approach can be compared against more sophisticated association algorithms (e.g., JPDA, MHT) to characterize the performance gap. In many BMD scenarios with well-separated tracks, the gap is expected to be small; in dense, ambiguous scenarios, MHT may provide measurable improvement at the cost of significantly higher computational complexity [3]. The forge-track-correlator's design prioritizes predictability and bounded cost, which may warrant the marginal performance trade-off.

9. Conclusion

The forge-track-correlator provides a practical, real-time multi-sensor track correlation engine for the FORGE missile defense framework. Its key contributions are:

1. **A simple, deterministic correlation algorithm** based on position-velocity scoring with a fixed gating threshold. The algorithm is O(n) per detection, has a small constant factor, and produces predictable, repeatable results.

2. **A five-level threat classification scheme** that dynamically reassesses threat level as track fidelity improves, supporting graduated response from initial warning through engagement authorization.
3. **Robust track lifecycle management** with quality metrics and automatic stale track cleanup that prevents resource leaks and maintains correlation performance over extended operations.
4. **Kafka-native streaming architecture** that provides fault tolerance, backpressure handling, and horizontal scalability through consumer group partitioning.

The system makes deliberate trade-offs. The fixed scoring metric and gating threshold sacrifice the adaptive statistical rigor of covariance-based gating (as in the Mahalanobis distance approach of Bar-Shalom [1]) in favor of simplicity and predictability. The nearest-neighbor association rule avoids the combinatorial complexity of multi-hypothesis tracking [4] at the cost of occasional suboptimal associations in dense, ambiguous scenarios.

These trade-offs are appropriate for the FORGE operating environment, where the number of simultaneous tracks is bounded, sensor revisit rates provide frequent updates, and the primary requirement is reliable, low-latency correlation rather than optimal association under extreme ambiguity.

Future work may include:

- **Covariance-aware scoring:** Incorporating sensor-specific and track-specific covariance matrices into the scoring metric to improve association accuracy when sensors have dramatically different error characteristics.
- **Adaptive gating:** Dynamically adjusting the gating threshold based on track density and time since last update (wider gate for stale tracks, tighter for recently updated tracks).
- **Multihypothesis extensions:** Maintaining multiple association hypotheses for ambiguous detections and resolving them as further evidence accumulates.
- **Machine-learning augmentation:** Training classification models on historical track data to improve threat level assignment, particularly for decoy discrimination at Level 1.

The forge-track-correlator demonstrates that a well-designed, deliberately simple correlation engine can meet the real-time demands of missile defense while providing the extensibility to incorporate more sophisticated techniques as operational requirements evolve.

References

[1] Y. Bar-Shalom, F. Daum, and J. Huang, "The Probabilistic Data Association Filter," IEEE Control Systems Magazine, vol. 29, no. 6, pp. 82–100, Dec. 2009. — Foundational treatment of probabilistic data association for multi-target tracking in clutter.

[2] S. S. Blackman and R. Popoli, Design and Analysis of Modern Tracking Systems. Norwood, MA: Artech House, 1999. — Comprehensive reference on radar and multi-sensor tracking, including gating, association, and fusion architectures.

[3] S. S. Blackman, "Multiple Hypothesis Tracking for Multiple Target Tracking," IEEE Aerospace and Electronic Systems Magazine, vol. 19, no. 1, pp. 5–18, Jan. 2004. — Survey of MHT approaches and comparison to nearest-neighbor methods for dense target environments.

[4] D. B. Reid, "An Algorithm for Tracking Multiple Targets," IEEE Transactions on Automatic Control, vol. 24, no. 6, pp. 843–854, Dec. 1979. — Seminal paper on multiple hypothesis tracking for multi-target tracking.

[5] Y. Bar-Shalom and X.-R. Li, Estimation and Tracking: Principles, Techniques, and Software. Norwood, MA: Artech House, 1993. — Detailed treatment of Kalman filtering, track state estimation, and data association.

[6] O. E. Drummond and S. S. Blackman, "Challenge of Kinematic Multiple Hypothesis Tracking," in Proc. SPIE Signal and Data Processing of Small Targets, vol. 1096, 1989, pp. 565–573. — Analysis of MHT computational complexity and practical implementation considerations.

[7] R. W. Sittler, "An Optimal Data Association Problem in Surveillance Theory," IEEE Transactions on Military Electronics, vol. 8, no. 2, pp. 125–139, Apr. 1964. — Early formulation of the multi-target data association problem.

[8] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A Distributed Messaging System for Log Processing," in Proc. NetDB, 2011. — Design and architecture of Apache Kafka for high-throughput, fault-tolerant streaming data pipelines.

[9] C. W. Glover, "Overhead Persistent Infrared (OPIR) Sensor Technology for Missile Defense," in Proc. AIAA Missile Sciences Conference, 2018. — OPIR sensor capabilities and limitations for ballistic missile early warning.

[10] L. D. Stone, T. L. Corwin, and C. A. Barlow, Bayesian Multiple Target Tracking. Norwood, MA: Artech House, 2014. — Bayesian approaches to multi-target tracking with applications to sensor fusion.