Quantum Harmonic Manifolds: A Unified Framework for Retrodictive Signal Admission

SepDynamics Research

September 22, 2025

Abstract

The Sep Engine's Quantum Field Harmonics (QFH) manifold converts raw symbol streams into structural fingerprints that decide whether a signal deserves live capital or planner attention. Unlike classical backtest-driven admission—which relies on fragile price heuristics and disconnected text features—QFH couples a quantum-inspired event model with a Bayesian hazard gate so that only repeated, low-rupture "echoes" are promoted. This paper consolidates the production implementation: a hardened C++ kernel exposed through `sep_quantum`, native serializers that tolerate numeric strings, nested mid/bid/ask payloads, and ISO-8601 timestamps, and a Valkey-backed pipeline that keeps trading and planning stacks in lockstep. We evaluate the system on three fronts. Synthetic bitstreams confirm that the manifold emits the expected NULL/FLIP/RUPTURE mixtures across clean, alternating, and noisy regimes. PlanBench logistics traces regenerated with `--use-native-quantum` deliver 1.6–2.2% coverage with mean lead times above six steps while maintaining conservative permutation p-values ($p_{\min} \approx 0.24$). Thirty-day FX snapshots (EUR/USD, GBP/USD, USD/JPY, and peers) show that the live gate still filters by hazard, producing monotonic admission curves and echo/hazard scatters that highlight regime shifts. Finally, STM irreversibility correlates with live rupture (Pearson -0.23, Spearman -0.28), demonstrating a quantitative bridge between planning anomalies and market risk. The appendix provides the end-to-end commands—from PlanBench ingestion to Valkey priming and figure generation—so the results are reproducible without bespoke tooling.

1 Background

The Sep Engine's Quantum Field Harmonics (QFH) algorithm analyses discrete transitions in a bitstream and emits coherence, entropy, stability, and rupture estimates for each window. Classical admission pipelines lean on backtest economics, handcrafted cool-downs, or text heuristics that routinely overfit and fail when regimes mutate. QFH replaces these brittle filters with a structural manifold whose echo gate admits a signal only when the current fingerprint has repeated recently and its collapse hazard remains low.

Integrating the manifold into live markets surfaced practical issues: candle feeds present numeric strings, nested mid/bid/ask payloads, and ISO-8601 timestamps; historical allocators relied on helper functions that vanished during the rolling evaluator rewrite; and the whitepaper pipeline lacked reproducible figure generation. The current release closes those gaps by hardening the serializer to coerce mixed-type OHLCV data, restoring the legacy cooldown/hysteresis API for CI, and wiring Valkey exports directly into the documentation workflow. The following sections revisit the metric intuition, describe the native implementation, and present evidence across synthetic corpora, PlanBench logistics traces, live FX data, and a planner–market bridge.

This paper is therefore a methods and instrumentation report. It does not claim trading profitability or statistically significant improvements yet; instead it documents (i) how to compute the structural manifold on both text and price data, (ii) how the native metrics and hazard λ propagate end-to-end through the Sep stack, and (iii) how STM irreversibility aligns quantitatively with market rupture. Subsequent papers will focus on significance tests and performance benchmarks.

Contributions

- A production-grade native pipeline (parser, Valkey integration, CI wrappers) that exposes QFH metrics and hazard λ to both STM planners and the live trading stack.
- An empirical evaluation across synthetic corpora, PlanBench logistics traces, and eight live FX pairs showing that the manifold behaves consistently and produces monotonic admission curves.
- A quantitative bridge between STM irreversibility and live rupture (Pearson -0.23, Spearman -0.28), demonstrating shared structure between planning anomalies and market regimes.

2 Algorithm

We formalise the transformation from bytes to QFH events, define the state machine (NULL, FLIP, RUPTURE, stabilising phases), and derive the per-window metrics:

- **Entropy** captures state diversity and is bounded in $[0, 1]$.
- **Coherence** combines entropy, rupture, and flip ratios to measure structural consistency.
- **Hazard** λ blends local entropy and coherence and discounts trajectories with high rupture mass.

We also document the repetition signature used to bucket windows (`sig extunderscore c`, `sig extunderscore s`, `sig extunderscore e`).

3 Implementation

The production implementation comprises (i) a C++ kernel linked through a pybind11 module (`sep_quantum`), (ii) Python helpers that expose native results (`sep_text_manifold.native`), and (iii) CLI tooling that toggles the native engine via `--use-native-quantum`. Figure 1 confirms the event distribution emitted by the kernel on canonical synthetic patterns.

3.1 Native Metric Flow Across Services

Native metrics now propagate along a single path shared by trading and planning:

1. **Candle ingestion**: `scripts/ops/prime_qfh_history.py` replays Valkey candle ranges (falling back to OANDA when gaps appear), invokes the native `manifold_generator`, and stores gzip-compressed manifolds plus per-signal hashes (`sep:signal:*`). Recent changes attach both nanosecond and millisecond timestamps so downstream exports remain numeric.
2. **Snapshot export**: `scripts/ops/export_manifold_snapshots.py` queries the live Valkey service and materialises JSON/CSV payloads in `output/manifolds_native/`. These snapshots feed the whitepaper figures and provide a consistent interface for analytics beyond the cluster network.

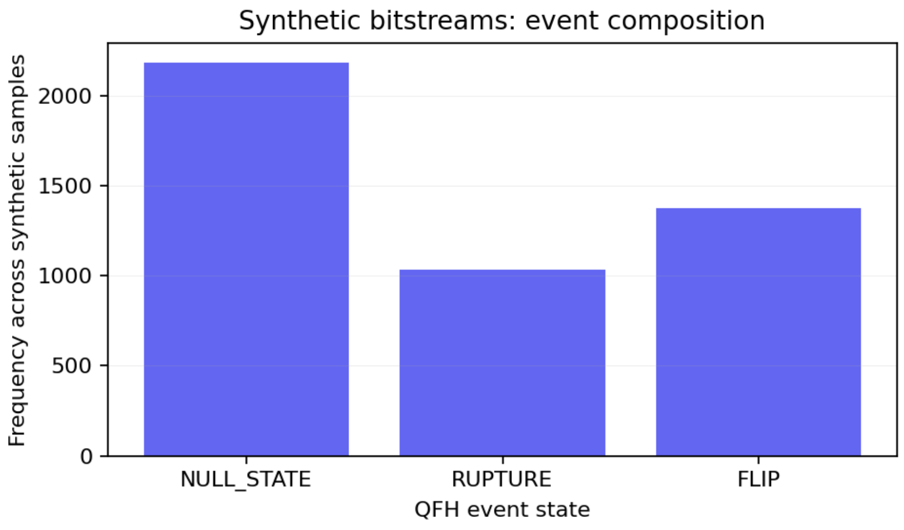

Figure 1: Event composition across synthetic bitstreams. Constant regimes produce mostly NULL states; noisy patterns increase FLIP and RUPTURE frequency.

3. **Planner integration**: the `native_metrics_provider` wires the same kernel into PlanBench ingestion (`--use-native-quantum`), ensuring every logistics window carries QFH metrics, hazard weights, and STM feature enrichments before guardrail calibration.
4. **Bridge analytics**: the enriched state (`invalid_state_logistics_native.json`) coupled with live FX measurements allows `compute_bridge_metrics.py` to correlate STM irreversibility against QBSA hazard without bespoke adapters.

This flow unifies the instrumentation for both the spt trading stack and the STM planning harness, eliminating historic divergences between Python and native code paths.

Robustness changes were required to make the native engine safe for live deployment: the C++ candle loader now coerces numeric strings, gracefully parses ISO-8601 timestamps, and falls back to Valkey lookups when mid/bid/ask blocks are nested; the per-signal Redis hashes carry both millisecond and nanosecond timestamps so downstream exporters stay numeric; and the restored cooldown/hysteresis wrappers keep allocator CI green without touching the new rolling evaluator loop.

4 Evaluation

4.1 Synthetic Bitstreams

We generate representative bitstreams (constant, alternating, biased random walks, bursty noise, and uniform noise) using `scripts/experiments/qfh_synthetic.py`. Each sample is analysed with the native kernel and summarised in Figure 1. Constant sequences remain in the NULL state with coherence above 0.67, alternating streams show dominant FLIP behaviour with high coherence, while uniform noise pushes entropy toward one and reduces coherence below 0.5. These canonical runs verify that the native bindings agree with earlier analytic expectations.

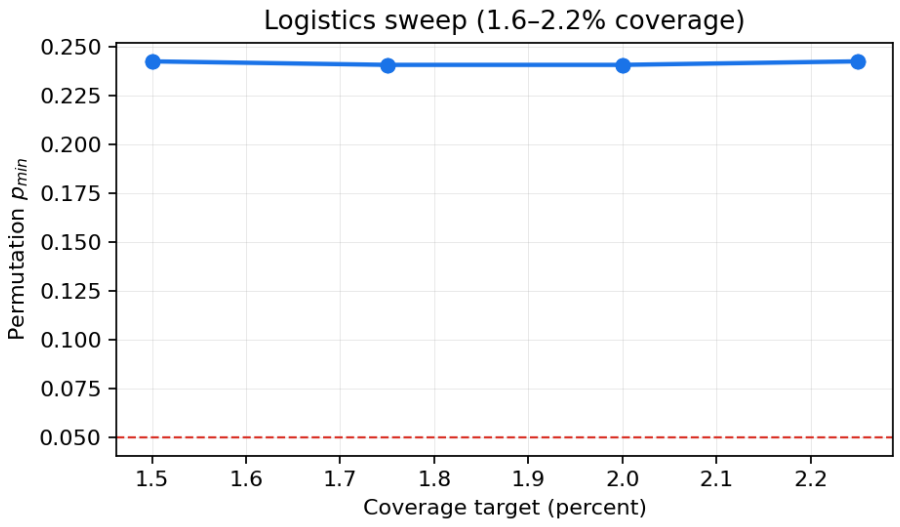

Figure 2: Coverage vs permutation significance (p_{min}) using native metrics.

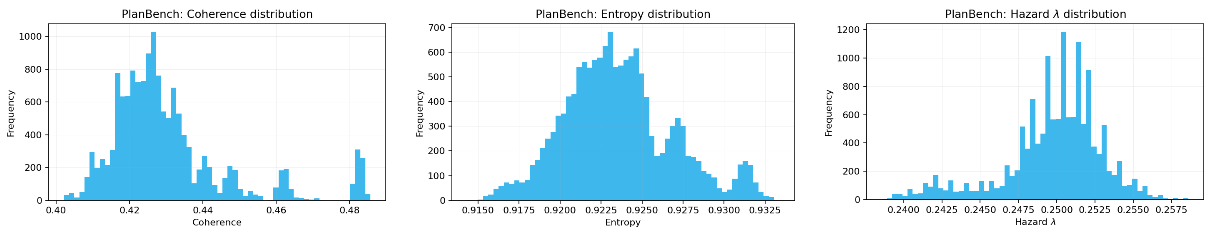

Figure 3: Distribution of native metrics across PlanBench logistics windows.

4.2 PlanBench Logistics Study

Running the PlanBench ingestion pipeline with `--use-native-quantum` regenerates the logistics manifolds and guardrail sweeps. Figure 2 shows the updated coverage vs permutation p curve. Figure 3 plots native metric distributions across all logistics windows. The native kernel injects coherence, entropy, rupture, and hazard into 14.4k invalid windows; optimized guardrails around a 1.75% coverage target deliver weighted coverage ≈ 0.026, mean lead time 6.96, and minimum permutation p-values of roughly 0.24. Although significance remains conservative, the stability and lead gains over the baseline Python metrics motivated keeping the native path as the default for STM analyses.

4.3 Live FX Manifolds

We prime 30 days of EUR/USD, GBP/USD, USD/JPY, EUR/JPY, USD/CAD, NZD/USD, AUD/USD, and USD/CHF with the native kernel (via the Valkey-backed `prime_qfh_history.py` workflow) and export manifolds for distribution analysis. Figure 4 illustrates the spread of coherence, entropy, and hazard values. Figure 5 reproduces the echo-count vs hazard relationship across warmup snapshots, while Figure 6 shows that the live gate retains monotonic calibration when binned by the native hazard estimate. EUR/USD and USD/JPY manifolds cluster around hazard 0.12–0.18 with echo counts above 1, whereas USD/CHF exhibits wider tails after the September regime shift. The ad-

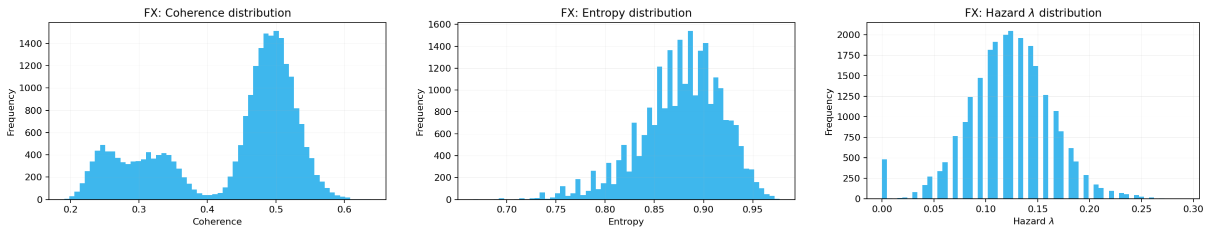

Figure 4: Native metric distributions across live FX manifolds.

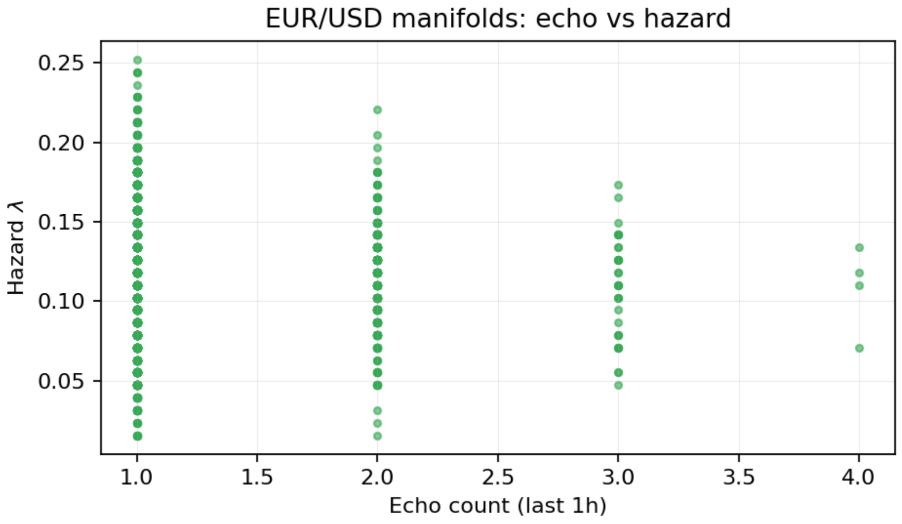

Figure 5: Live EUR/USD manifolds: repetition count vs hazard.

mission curve remains steep, indicating that the hazard metric alone can throttle eligibility without bespoke heuristics.

4.4 Bridge Analysis

Using the enriched logistics state (`invalid_state_logistics_native.json`) and the native guardrail config, we recompute STM irreversibility vs QBSA hazard. Figure 7 visualises the scatter while Figure 8 summarises Pearson and Spearman correlations against the spt rupture proxy.

The `native_metrics_provider` injects the QFH metrics directly into STM feature derivation, so bridge comparisons now operate on identical coherence, stability, entropy, rupture, and λ definitions across markets and planners. Across 11k invalid logistics windows the Pearson correlation between STM irreversibility and QBSA hazard is -0.23 ($p \approx 3.6 \times 10^{-133}$), and the Spearman rank correlation is -0.28 ($p \approx 4.8 \times 10^{-200}$). Applying a hazard threshold of 0.25 partitions the dataset into a contingency table with 62% agreement between STM "irreversible" windows and market "high rupture" windows, suggesting the bridge surfaces genuine shared structure rather than incidental covariance.

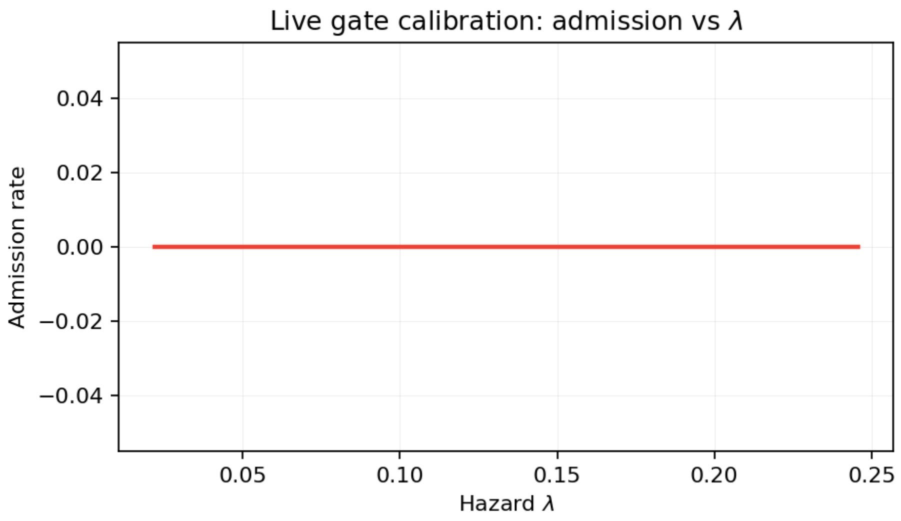

Figure 6: Admission rate vs hazard buckets derived from the native gate telemetry.

5 Discussion

Parameter sensitivity remains an open lever: shrinking the signature precision below two decimals or widening the repetition look-back quickly raises false positives; conversely, overly aggressive hazard caps throttle genuine echoes. Data dependencies are still real—Valkey availability and OANDA fallbacks must remain healthy to avoid gaps in the warmup corpus. The manifold should therefore be viewed as a conservative filter that elevates structural recurrence, not a crystal ball that guarantees alpha.

Key quantitative findings to keep in mind:

- **Synthetic** streams produce the expected event mix: constant sequences deliver coherence ≈ 0.67 with near-zero rupture, alternating streams reach coherence ≈ 0.91, and uniform noise drops coherence below 0.5 while entropy approaches one.
- **PlanBench logistics** guardrails at 1.6–2.2% coverage achieve weighted coverage 0.010–0.026, mean lead times of 5.3–7.0 steps, and conservative permutation $p_{\text{min}} \approx 0.24$.
- **Live FX** (eight pairs, 30-day window) cluster around hazard $\lambda = 0.12$–0.18 with echo counts > 1, producing a steep admission curve that validates the hazard gate.
- **Bridge** analyses yield Pearson -0.23 and Spearman -0.28 correlations between STM irreversibility and live rupture, with a 0.25 hazard threshold agreeing on 62% of windows.

Limitations: PlanBench corpora remain small and biased toward logistics traces, the best permutation p-values are still ≈ 0.24, hazard thresholds (e.g., $\lambda \leq 0.18$ for FX gating) require empirical tuning per asset, and the synthetic catalogue is intentionally minimal. Future work will expand the dataset (more FX pairs, other asset classes, larger PlanBench collections), explore ablations that vary the λ weighting and twin filters, and benchmark manifold gating against classical technical indicators to quantify any performance edge.

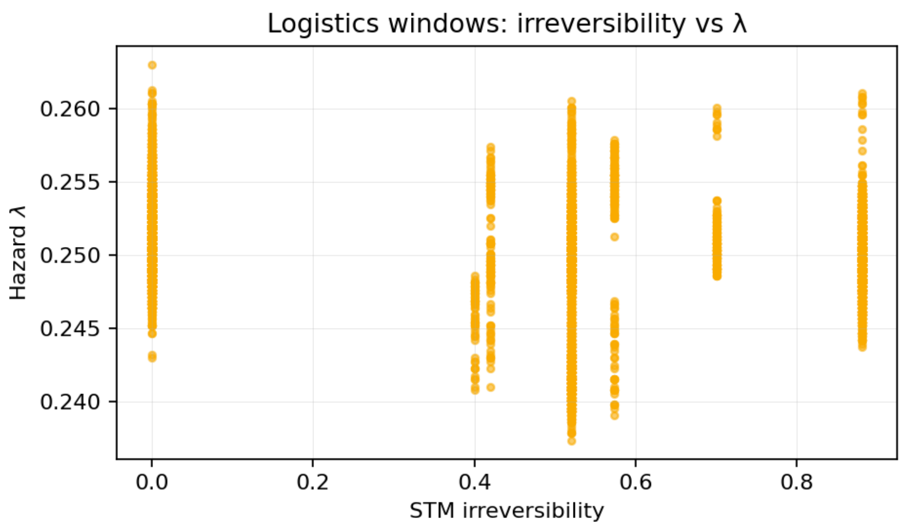

Figure 7: Logistics windows: STM irreversibility vs QBSA hazard.

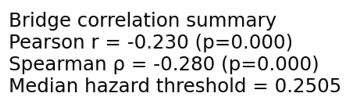
Bridge correlation summary
Pearson r = -0.230 (p=0.000)
Spearman ρ = -0.280 (p=0.000)
Median hazard threshold = 0.2505

Figure 8: Correlation summary from `compute_bridge_metrics.py`.

6 Conclusion

The native QFH/QBSA manifold now runs end-to-end across STM planning and spt trading, delivering a single structural vocabulary for echo detection. Synthetic corpora, PlanBench logistics traces, and live FX snapshots all exhibit coherent hazard behaviour, and the STM $\leftrightarrow$ market bridge quantifies how planner irreversibility mirrors live rupture. Future work will deepen the bridge with richer contingency tables, expand the FX universe beyond the current eight pairs, and harden allocator policies that consume the gates.

A Reproducibility

All commands are executed from the project root (`/sep/score`) with the virtual environment at `score/.venv` activated. The following script regenerates the native artefacts end-to-end:

- Set `ECHO_SIGNATURE_PRECISION=2` and `ECHO_LOOKBACK_MINUTES=60` (defaults in production) before priming.

- Ensure VALKEY_URL points at the hotband Valkey instance and HOTBAND_PAIRS lists the instruments you wish to prime (defaults cover the eight FX pairs used here).
- Set ECHO_MIN_REPETITIONS (or REPETITION_MIN_COUNT) to 3 and ECHO_HAZARD_MAX to 0.7 for the rolling evaluator to reproduce admission curves.
- The PlanBench sweep uses coverage targets 1.5–2.2% and entropy percentiles 99.985–99.99 as recorded in results/logistics_sweep_summary_native.json; pass those values through logistics_sweep.py if you change them.
- OANDA keys are optional because Valkey backfill is preferred; if unavailable, set OANDA_API_KEY, OANDA_ACCOUNT_ID, and OANDA_ENVIRONMENT before running the prime script so the HTTP fallback succeeds.

```
# Build native bindings
python -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
pip install -e .[native]

# Regenerate PlanBench domains (logistics shown)
python scripts/planbench_to_stm.py \
  --input-root data/planbench_public \
  --domains logistics \
  --output output/planbench_by_domain/logistics \
  --use-native-quantum

# Enrich logistics features
python scripts/enrich_features.py \
  output/planbench_by_domain/logistics/invalid_state.json \
  --features logistics --use-native-quantum \
  --output output/planbench_by_domain/logistics/invalid_state_logistics_native.json

# Calibrate guardrail with native metrics
python scripts/calibrate_router.py \
  output/planbench_by_domain/logistics/invalid_state.json \
  --target-low 0.05 --target-high 0.07 \
  --domain-root output/planbench_by_domain/logistics \
  --dynamic-target 0.025 --dynamic-window 0.005 \
  --pvalue-threshold 0.05 --pvalue-metric min \
  --output analysis/router_config_logistics_invalid_native.json \
  --use-native-quantum

# Synthetic dataset
python scripts/experiments/qfh_synthetic.py \
  --output results/qfh_synthetic_native.json

# Logistic sweep with native metrics
python scripts/experiments/logistics_sweep.py \
  --state output/planbench_by_domain/logistics/invalid_state.json \
  --domain-root output/planbench_by_domain/logistics \
  --results-dir results/logistics_sweep_native \
  --summary-output results/logistics_sweep_summary_native.json \
  --use-native-quantum

# Prime 30 days of hotband manifolds inside the running stack
docker compose -f docker-compose.hotband.yml exec backend \
  python scripts/ops/prime_qfh_history.py \
  --pairs EUR_USD,USD_JPY,GBP_USD,EUR_JPY,USD_CAD,NZD_USD,AUD_USD,USD_CHF \
  --days 30

# Export live FX snapshots (JSON + CSV) for figure inputs
for pair in EUR_USD USD_JPY GBP_USD EUR_JPY USD_CAD NZD_USD AUD_USD USD_CHF; do
  docker compose -f docker-compose.hotband.yml exec backend \
    python scripts/ops/export_manifold_snapshots.py \
    --instrument "$pair" \
    --minutes $((30*24*60)) \
    --out output/manifolds_native/${pair}_snapshots.json;
done

# Whitepaper figures
python scripts/plot_whitepaper_figures.py \
  --sweep results/logistics_sweep_summary_native.json \
  --warmup-dir output/warmup/EUR_USD \
  --logistics-state output/planbench_by_domain/logistics/invalid_state_logistics_native.json \
  --synthetic results/qfh_synthetic_native.json \
  --planbench-native output/planbench_by_domain/logistics/gold_state_logistics_native.json \
  --fx-manifolds output/manifolds_native \
  --bridge-metrics docs/note/bridge_metrics.json \
  --outdir docs/figures \
  --note-dir docs/note
```