TSGuard: A Real-Time System for Detecting and Imputing Missing Data in Streaming Time Series

Imane Hocine[1], Yacine Hakimi[2], Asma Abboura[3], Soror Sahri[4] and Grégoire Danoy[1,5]

[1]SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
[2]École nationale Supérieure d'Informatique (ESI), Algiers, Algeria
[3]Hassiba Benbouali University, Chlef, Algeria
[4]Université Paris Cité, Paris, France
[5]FSTM-DCS, University of Luxembourg, Esch-sur-Alzette, Luxembourg

Abstract

TSGuard is a lightweight real-time system for detecting and imputing missing values in streaming environmental time series. It combines Graph Neural Networks (GNNs) to capture spatial relationships and Long Short Term Memory (LSTM) networks for temporal modeling, while enforcing domain-specific physical constraints to ensure plausible imputations. This hybrid design allows TSGuard to handle data delays, sensor failures, and environmental disruptions with low response times. We demonstrate its effectiveness on real-world environmental data and compare it to a deep learning baseline to show its suitability for real-time monitoring.

Keywords

Time Series, Missing Values Detection, Missing Values Imputation, GNNs, LSTM, Real-Time Systems

1. Introduction

Time-series data streams are critical in domains such as remote sensing, finance, healthcare, and environmental monitoring, where timely detection of anomalies, sensor failures, or catastrophic events (e.g., extreme weather, wildfires) is essential. However, sensor malfunctions, atmospheric interference, and transmission errors frequently lead to missing values, compromising data integrity and degrading downstream analytics.

In this work, we focus on satellite-based earth observation systems as a representative and challenging use case. These systems often experience missing data because of clouds blocking sensors, hardware failures, or communication delays. This can cause significant issues for time-sensitive applications like wildfire detection or flood monitoring. For instance, a missing or delayed satellite reading during a wildfire could slow down early alerts, which put the environment and people at greater risk.

A wide range of graph-based and deep learning models have been proposed to handle missing values imputation by using spatial and temporal dependencies, including GRIN [1], PriSTI [2], INCREASE [3], and deep architectures targeting fine-grained temporal patterns [4]. While effective, most operate in offline or batch mode, limiting their suitability for real-time streaming. Online approaches such as SANNI [5] and ORBITS [6] address streaming settings but often overlook explicit spatial modeling or domain constraints.

Despite this progress, existing solutions face three main challenges: (1) capturing complex temporal dependencies and nonlinear spatial relationships in real time, (2) enforcing physical or domain-driven constraints to ensure plausible reconstructions, and (3) working effectively in fast-changing environments with missing, inconsistent, or delayed data, where quick and accurate responses are crucial and waiting for additional recordings is not always possible.

Motivating Example. To illustrate the challenges of real-time missing value detection and imputation, consider a typical Earth observation scenario involving multiple satellites streaming near-infrared

Proceedings of the Demonstration Track at International Conference on Cooperative Information Systems 2025, CoopIS 2025, Marbella, Spain, October 20–22, 2025.

0000-0002-9992-577X (I. Hocine); 0009-0005-4820-8091 (Y. Hakimi); 0009-0002-1740-088X (A. Abboura); 2222-3333-4444-5555 (S. Sahri); 0000-0001-9419-4210 (G. Danoy)

(NIR) reflectance measurements. In such settings, ensuring *timeliness*[1] is critical, as delays directly affect *completeness*[2] and compromise the reliability of downstream analysis. Delay detection relies on monitoring the *arrival interval* (Δt_i) and comparing it against a user-defined threshold k.

Suppose four satellites (S_1, S_2, S_3, S_4) measure NIR reflectance over a monitored region every 15 minutes. At timestamp $t = 3$, S_3 reports an unusually high reflectance, potentially indicating a wildfire or sensor saturation. At $t = 4$, satellite S_1 fails to report a reading. The system computes the arrival interval:

$$\Delta t_1 = \text{current time} - \text{last recorded time}.$$

If it has been 25 minutes since S_1's last reading and the user-defined threshold is $k = 20$, then:

$$\Delta t_1 = 25 > k = 20.$$

Since this exceeds the threshold, S_1 is flagged as *delayed*, and the missing value is passed to the imputation module. The earlier anomaly reported by S_3 not only provides valuable context for estimating the missing reading but also helps domain experts infer the potential cause of the missing data: such as a sensor fault triggered by a sudden environmental event, e.g., a wildfire. In this way, the system supports both accurate reconstruction and improved situational understanding.

Demo Relevance and Readiness

TSGuard integrates spatial and temporal models with domain constraints to handle missing data in real-time, particularly in dynamic environmental monitoring contexts. It integrates three key components: (1) **graph-based spatial modeling** via GNNs to capture correlations among geographically related sensors, (2) **recurrent temporal modeling** via LSTM networks to learn both short and long-term trends, and (3) **constraint-guided reconstruction** using domain expert–provided physical ranges and environmental correlations, ensuring physically plausible and context-aware imputations. This combination enables TSGuard to effectively handle data delays, sensor failures, and transient disruptions while maintaining domain-consistent reconstructions. We also compare TSGuard's imputation results against Pristi [2], a new deep learning-based system, demonstrating its competitive performance.

The system is implemented in Python, using the Streamlit framework for an interactive user interface, and structured as a streaming pipeline compatible with low-power edge devices. Unlike conceptual prototypes, TSGuard is fully operational and has been validated on a real satellite sensor dataset. The source code is publicly available at Github. The demo interface allows attendees to visualize and detect missing values through the following features:

1. Real-time Visualization of GNN-learned spatial graphs and LSTM-predicted trends.
2. Enable/disable domain constraints to compare unconstrained with constraint-guided imputation.
3. Replay historical AQI-36 air quality datasets [3] or observe live data streams with simulated sensor faults.

Through the demo, attendees will observe real-time detection, fast reconstruction, and immediate alerts when values fall outside expert-defined ranges. It shows how AI methods like GNNs and LSTMs can be combined with expert knowledge to make information systems smarter and more responsive. By combining data-driven learning with domain-specific constraints, TSGuard remains both flexible and interpretable.

2. TSGuard Overview

This section presents the main components of TSGuard, as illustrated in Figure 1.

[1] Timeliness refers to producing results quickly enough to be useful in the intended application, often within strict deadlines or real-time constraints.

[2] Data completeness refers to the extent to which all required data is present.

[3] https://github.com/LMZZML/PriSTI/tree/main/data/pm25/SampleData

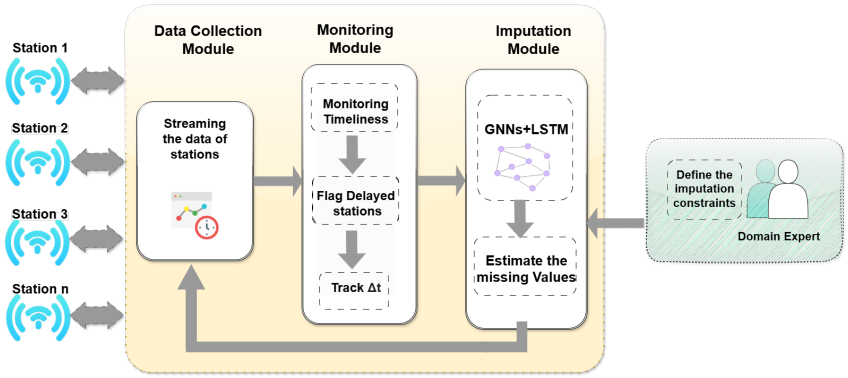


Figure 1: TSGuard system overview.

Data Collection Module. TSGuard collects heterogeneous environmental or spatial readings, such as temperature, atmospheric pressure, and radiation levels, continuously transmitted from satellite-mounted sensors to the base station. Transmission can be disturbed by factors such as signal loss, hardware failure, or environmental interference. This module establishes a real-time data pipeline from multiple sensors (stations), synchronizes readings based on arrival timestamps, and forwards them to the monitoring module for integrity checks.

Monitoring Module or Timeliness-Aware Module. This module ensures the integrity of collected data by detecting missing values in real time. If a sensor fails to deliver a reading within an acceptable time window, the system flags it as delayed or missing and triggers an alert to the imputation module. This module can be considered as the Timeliness-Aware Module.

Thresholds are initially user-defined. Formally, let $t_{i,\text{last}}$ be the timestamp of the most recent reading from sensor i. The elapsed time since that reading is:

$$\Delta t_i = (t_{current-time}) - t_{i,\text{last}}. \tag{1}$$

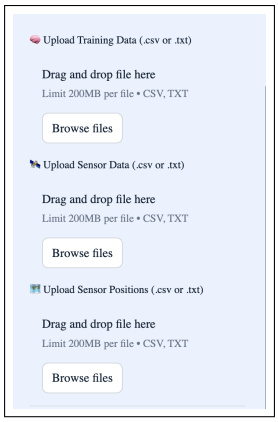
A sensor is flagged as delayed if: $\Delta t_i > k$ where k is a user-defined threshold. Any delay exceeding this value is treated as a missing observation and triggers the imputation module.

Imputation Module. Once missing data are detected, this module reconstructs them using a hybrid GNN–LSTM architecture where the GNN models spatial dependencies between sensors and the LSTM captures temporal patterns from each sensor's history. After reconstruction, domain constraints (user-defined), such as physical ranges and environmental correlations, are applied to monitor the plausibility of both the imputed values and readings from neighboring nodes. If a sensor's reconstructed value or any neighbor's reading falls outside expert-defined ranges, the system immediately issues real-time alerts to users. The module updates imputed values in real time, guaranteeing continuous data availability for downstream analytics.

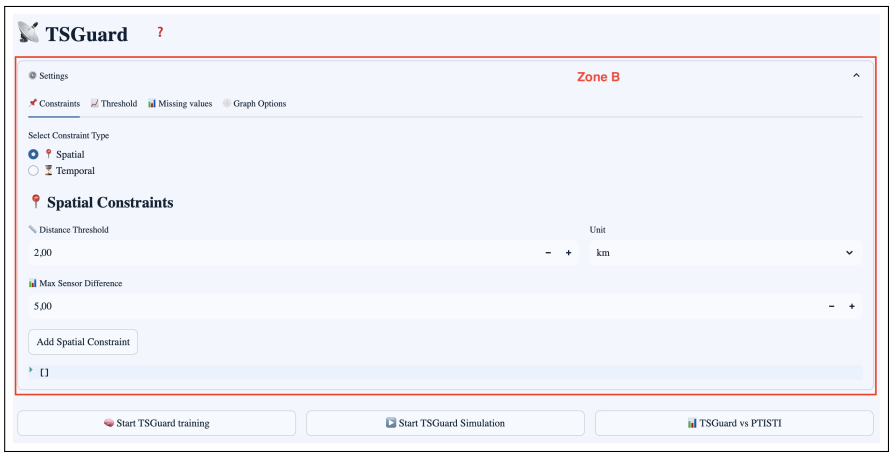
3. Demonstration Overview

3.1. TSGuard User Interface

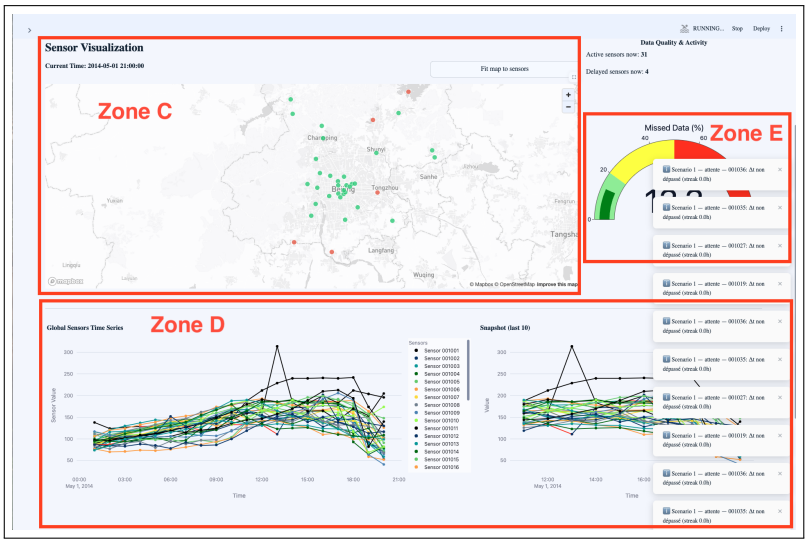
The TSGuard interface is organized into five key components. **Zone A** (Figure 2a) handles dataset uploads, sensor positioning, and training file input. **Zone B** (Figure 2b) supports domain constraint definition, station management, GNN training, and simulation control. After starting a simulation, three additional components become visible: **Zone C**, **Zone D**, and **Zone E**, shown in Figure 2c. Zone C visualizes station status on a map, Zone D displays real-time time series with imputation indicators, and Zone E reports data completeness levels classified as **low** ($\leq 20\%$), **medium** (20%–50%), or **high** ($> 50\%$).



(a) Zone A: Dataset upload and sensor configuration.



(b) Zone B: Constraint setup, station management, and model control.



(c) Interface showing time series streaming, station map, and imputation results.

Figure 2: Main components of the TSGuard user interface.

3.2. TSGuard Scenarios

We illustrate three representative scenarios:

Scenario 1: Missing value with delay below threshold Δt_i The station has not reported a value, but the delay is within the allowable range. The system waits for late data instead of triggering imputation.

Scenario 2: Missing value with delay above Δt_i and available neighbors Once the delay surpasses Δt_i, TSGuard imputes the value using spatial–temporal correlations from neighboring stations. After reconstruction, domain constraints are applied to monitor plausibility. During the demo, we illustrate three possible cases: (1) the imputed value is within range and all neighboring stations also produce values within range; (2) the imputed value is outside the acceptable range; (3) the imputed value is within range, but one or more neighboring values are out of range, indicating that naïve imputation could mask a significant environmental anomaly. In such cases, TSGuard issues real-time alerts to ensure users are immediately informed of potential hazards.

Scenario 3: Missing value with delay above Δt_i and unavailable neighbors If both the target and neighboring stations are missing values, TSGuard resorts to historical patterns for imputation. If no reliable estimation is possible, an alert is triggered to indicate a potential system or sensor fault.

3.3. TSGuard vs Pristi

We compared TSGuard's imputation results with Pristi [2], a recent deep learning model that jointly learns spatial and temporal dependencies. Pristi was chosen as a strong baseline due to its unified architecture for spatiotemporal imputation. It also operates in offline mode, unlike TSGuard's real-time, constraint-aware design. This allows us to show key trade-offs between accuracy, responsiveness, and interpretability in streaming settings.

TSGuard achieves fast, real-time imputations while maintaining domain consistency and plausibility. The demo interface shows how TSGuard can provide accurate data quickly, which makes it great for real-time environmental monitoring. The demo also encourages discussion around the challenges of deploying deep learning models like Pristi in real-time streaming environments. It shows the importance of balancing accuracy, responsiveness, and interpretability in real-time systems.

4. Conclusion and Future Work

TSGuard demonstrates how real-time AI-driven monitoring can improve the quality and reliability of sensor data streams in critical applications. By integrating timeliness-aware detection with GNN- and LSTM-based spatiotemporal modeling, TSGuard delivers timely and physically consistent reconstructions while meeting real-time constraints. Our demonstration shows its practical applicability for environmental monitoring and beyond. Future work will focus on the automatic discovery of domain constraints, the scalability analysis of TSGuard in large-scale sensor deployments, and extending the system to support multiple datasets and comparisons against diverse baseline methods.

Acknowledgments

This work is funded by the Luxembourg National Research Fund (FNR), INTER programme under the UltraBO Project (ref. INTER/ANR/22/17133848), and the Polish National Centre for Research and Development (NCBR) (ref. POLLUX-XI/15/Serenity/2023).

References

[1] A. Cini, I. Marisca, C. Alippi, Filling the g_ap_s: Multivariate time series imputation by graph neural networks, arXiv preprint arXiv:2108.00298 (2021).

[2] M. Liu, H. Huang, H. Feng, L. Sun, B. Du, Y. Fu, Pristi: A conditional diffusion framework for spatiotemporal imputation, in: 2023 IEEE 39th International Conference on Data Engineering (ICDE), IEEE, 2023, pp. 1927–1939.

[3] C. Zheng, X. Fan, C. Wang, J. Qi, C. Chen, L. Chen, Increase: Inductive graph representation learning for spatio-temporal kriging, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 673–683.

[4] P. Bansal, P. Deshpande, S. Sarawagi, Missing value imputation on multidimensional time series, arXiv preprint arXiv:2103.01600 (2021).

[5] A. Yurtin, M. Zymbler, Sanni: Online imputation of missing values in multivariate time series based on deep learning and behavioral patterns, Lobachevskii Journal of Mathematics 45 (2024) 5948–5966.

[6] M. Khayati, I. Arous, Z. Tymchenko, P. Cudré-Mauroux, Orbits, Proceedings of the VLDB Endowment 14 (2020) 294–306.