Sensors and Materials, Vol. 38, No. 6 (2026) 3623–3636
MYU Tokyo

S & M 4524

From Static Thresholds to Smart Baselines: Incremental AI Enhancement for Industrial Time Series Data Monitoring

Ruming Tang,[1*] Hua Lou,[1] Xingzhi Chang,[1] Xidao Wen,[2] and Kanglin Yin[3]

[1]Changzhou College of Information Technology, Wujin District, Changzhou 213164, China
[2]BizSeer Technologies Co., Ltd., Haidian District, Beijing 100083, China
[3]Key Laboratory for Satellite Digitalization Technology, Chinese Academy of Sciences
Pudong New District, Shanghai 200120, China

(Received February 9, 2026; accepted June 23, 2026)

Keywords: AIOps, anomaly detection optimization, time series anomaly detection

Industries increasingly demand intelligent monitoring for data anomalies in industrial sensing systems. However, traditional threshold-based sensor data anomaly detection remains prevalent; a rapid shift to fully AI-enhanced systems is often impractical. Instead, incrementally optimizing traditional methods is more feasible. In this study, we enhance traditional dynamic threshold techniques for sensor time series data using large model-driven recommendations and parameter optimization. Our method first relabels and corrects samples from existing threshold outputs to recalculate appropriate values. Through a time-segmented split mechanism, it dynamically adjusts threshold ratios within each segment to better reflect sensor data variations. Additionally, we incorporate a large language model to recommend optimal parameter configurations, improving usability and flexibility. Real-world deployment demonstrates that our approach significantly outperforms traditional dynamic thresholds in detection accuracy and adaptability, offering a scalable solution for transitioning toward intelligent sensor monitoring. While effectively addressing the rigidity and high false-alarm rates of traditional sensor baselines, this study's limitation is its reliance on human validation for complex sensor drift. The remaining challenge is achieving the fully autonomous handling of multi-sensor correlations.

1. Introduction

Operations and maintenance (O&M) are essential tasks that underpin the stability and efficiency of industrial sensing systems.[1,2] Among various data formats, time series data continuously generated by sensor networks serve as the cornerstone of numerous real-world operations, such as sensor monitoring, anomaly detection, and performance forecasting.[3,4] The paradigm for managing these tasks has evolved significantly, progressing from basic manual operations and simple automated scripts to DevOps (co-operation between development and O&M),[5] and now to the era of Machine Learning Operations[6] and AI for IT Operations (AIOps).[7] AI technologies are transforming sensor data operations through intelligent data

*Corresponding author: e-mail: tangruming@czcit.edu.cn
https://doi.org/10.18494/SAM6282

ISSN 0914-4935
https://myukk.org/

analysis, enabling proactive and predictive capabilities.[7–9] Recently, this potential has been expanded by large language models (LLMs), which are being applied to automate log analysis, generate reports, and provide support for operations teams.[10,11]

However, a significant chasm persists between the aspirational vision of AIOps and its practical implementation in sensing applications. Despite extensive research, widespread adoption in sectors such as transportation, energy, and manufacturing remains limited.[9] This slow progress stems from a convergence of challenges in sensor systems, including poor data quality, technical integration difficulties, and insufficient AI literacy within the workforce.[6,8,12]

Given these real-world limitations, the impracticality of a rapid, wholesale shift to AI-driven O&M necessitates an incremental, evolutionary approach that augments existing sensing systems.[10] Specifically, we propose a roadmap that begins with traditional optimization and advances through stages of increasing AI-driven autonomous O&M capabilities. Rather than jumping directly to a full AI model, we advocate for a transition path: starting from full manual operation, evolving to a "Manual Target + AI Parameters" model where humans define goals and AI tunes settings, and ultimately aiming for fully autonomous management. Grounded in this philosophy, we present a case study modernizing the traditional dynamic thresholds method, which is a widely used technique for sensor data anomaly detection. We first refine its output to better capture complex sensor patterns and then modularize its adjustment logic. This modularization creates a natural integration point for LLMs, which we leverage for parameter recommendation. The result is a dual improvement: enhanced detection precision (effectively reducing sensor false alarms) and simplified operator interaction, demonstrating a tangible pathway to transform a legacy tool into an intelligent AIOps component for sensing applications. On the basis of the evolutionary philosophy of AIOps adoption, we make the following contributions:

- We advocate for an evolutionary AIOps pathway centered on system augmentation, providing a clear roadmap from traditional O&M to AIOps.
- We introduce a novel segmentation and transformation technique to improve the traditional dynamic thresholds method for sensor data, significantly boosting its detection accuracy.
- We enhance usability by incorporating an LLM for intelligent parameter tuning, simplifying user interaction without compromising the system's reliability.

The rest of this paper is organized as follows. In Sect. 2, we present a case study in dynamic thresholds method and our optimization idea. In Sect. 3, we describe in detail our proposed methodology, including the target curve setting, time segmentation, and curve transformation. In Sect. 4, we present the experimental results with comparative analysis and discuss potential directions for future research. Finally, in Sect. 5, we conclude the paper.

2. Preliminary Methodology: A Case Study in Sensor Data Thresholds

Time series data anomaly detection is a cornerstone of modern sensing systems, where establishing a "normal" behavior pattern is crucial for detecting anomalies.[4] Besides basic, traditional static thresholds, a prevalent and intuitive approach to this is dynamic thresholds.[13,14] The fundamental principle is to model the expected behavior of a sensor metric by statistically

analyzing its historical data[4,15] and generate alert boundaries (sometimes also called alert baselines). In practice, sensor data such as cooling fan speed readings or machine inlet temperature measurements typically exhibit periodic patterns driven by regular operational schedules. Because we focus exclusively on such periodic data in this study, the dynamic thresholds method operates by aggregating measurements from corresponding temporal windows across multiple historical cycles. Statistical techniques are then applied to define the distributional boundaries. Typically, these involve calculating statistical measures, such as the mean (μ) and standard deviation (σ) to define a normal distribution range. An alert is then triggered when the current metric value deviates beyond a predefined threshold, e.g., $\mu \pm 3\sigma$.

Although this method provides an improvement over static thresholds by accounting for temporal variations in the data, it is fundamentally constrained by its reliance on a single statistical model. Such a model is adept at establishing a broad and generalized range of expected behavior, but is not suitable for different systems with unique data patterns (e.g., varying noise profiles or drift characteristics) that demand greater precision.

Another critical shortcoming of this method is its detachment from the underlying physical or operational semantics of the sensing environment. The configuration of its parameters is predominantly an exercise in manual tuning, guided by operator experience rather than a data-driven understanding of sensor behavior. Once deployed, the static statistical framework offers limited avenues for granular adjustment, rendering it ineffective for heterogeneous sensing environments where different sensor types demand distinct sensitivity requirements. As a result, the feedback loop for refining the model is coarse-grained at best. Operators can only apply global modifications to mitigate false positives or missed alerts, mainly based on their own experiences, lacking the capability to incorporate targeted feedback to optimize detection for specific sensor scenarios or equipment.

To address these limitations in sensor data anomaly detection, we propose a series of enhancements to the existing methodology. Acknowledging the technical complexities inherent in a complete paradigm shift,[9,10] we advocate for a phased, incremental integration of AI capabilities. Consequently, our initial strategy is not to replace the foundational method, but to refine its output. We will focus on postprocessing the initially generated thresholds, with the core objective of aligning the adjusted threshold curve more closely with the actual sensor data, thereby reducing false alarms triggered by noise.

3. Sensor Threshold Optimization Method Design

In this section, we approach the adjustment of legacy dynamic thresholds for sensor data from a fundamental perspective: how to transform the generated alert boundary curves to better fit sensor readings. Our methodology is guided by three core research questions that define a comprehensive transformation framework.

- RQ1: What is the optimal granularity within a sensor data anomaly detection cycle?
- RQ2: What are the target values for the transformed curves?
- RQ3: How is the original curve transformed to align with the target curve?

3.1 Time segmentation mechanism

Before transforming the original sensor thresholds, the first question is the granularity at which to apply this process. In conventional sensor data anomaly detection, a single threshold curve is typically applied and adjusted uniformly across an entire cycle. However, to achieve higher precision in capturing subtle sensor anomalies, we can perform finer-granular adjustments. To achieve this, we partition a cycle into distinct time segments (called adjust windows) and then independently adjust the curve within each segment.

A simple approach is to split sensor data into fixed-length segments or based on operational schedules, such as active manufacturing periods versus machine idle periods. To improve these heuristic methods, we propose a dynamic, data-driven segmentation approach based on the deviation between sensor readings and the original threshold curves. We first calculate the deviations (e.g., mean absolute error, *MAE*; mean percentage error, *MPE*)[16] over time to generate deviation curves. We then identify significant shifts in these curves as potential split points. To prevent over-segmentation caused by minor sensor fluctuations or transient environmental noise, we implement a hysteresis mechanism. This ensures that a new segment is only created when the deviation change is substantial and sustained, thereby filtering out sensor noise and creating more robust, meaningful data partitions for subsequent threshold adjustments.

3.2 Target curve setting

To achieve a higher degree of precision in sensor thresholds, we introduce a feedback mechanism that incorporates historical labeled sensor data. Instead of solely relying on statistical properties, our approach now explicitly considers both positive (normal sensor operations) and negative (anomalous sensor readings such as drift or spikes) samples from historical data. By leveraging these ground-truth instances, we can iteratively refine the thresholds. The core objective is to position the new threshold (target curve) in a way that maximizes the inclusion of known normal sensor data points while minimizing the encroachment on confirmed anomalous points, thereby directly aligning the dynamic thresholds with actual sensing and operating experience.

For concreteness, we begin with the simplest case of an upper threshold for a single time point, monitoring a specific sensor metric. Ideally, the anomalous sensor readings should be greater than the normal ones, and a clear separation should exist between the minimum value among all historical anomalous points and the maximum value among all historical normal points, as shown in Fig. 1(a). Theoretically, positioning the target threshold within this gap would effectively satisfy our detection requirements, correctly distinguishing between normal readings and anomalies. Extending this to a time window of length L, we can construct two envelope curves in Fig. 1(b). The first is derived from the minimum values of the anomalous sensor data points at each time step, while the second is constructed from the maximum values of the normal sensor data points. Consequently, the desired threshold curve should be placed within the region bounded by these two curves to accurately capture sensor deviations.

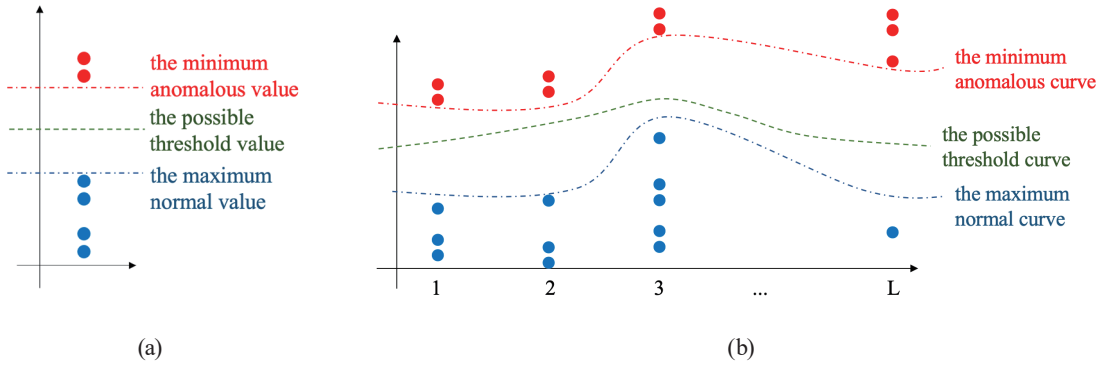


Fig. 1. (Color online) (a) Single time point and (b) time window of target sensor thresholds and labeled samples.

This scenario is a classic binary classification problem. A straightforward method is to use the midpoints between the two boundary curves as the threshold curve. Alternatively, this threshold curve can be dynamically adjusted with a specific parameter to position it closer to either boundary, allowing operators to control the sensitivity towards sensor noise versus subtle anomalies. Beyond this, a wide range of other machine learning methods are also applicable for separating sensor states, from traditional statistical methods,[17,18] to more advanced deep learning techniques.[19–21] The choice of the method and its corresponding parameters are one set of parameter configurations of the overall adjustment process.

3.3 Transformation method

Once the time segments and target curve for the sensor data are established, the next step is to transform the original curve into the new, adjusted one. For ease of implementation, we employ a linear transformation, which is formalized by the equation below, where $b(t)$ and $b'(t)$ denote the original and adjusted threshold curves, respectively; c_{per} denotes the proportional tolerance coefficient, scaling $b(t)$ by a relative factor (effectively accommodating sensor drift that scales with the measurement magnitude); c_{abs} denotes the absolute tolerance coefficient, adding a fixed offset to $b(t)$ (accounting for constant sensor offsets or persistent background noise).

$$b'(t) = b(t) \times (1 \pm c_{per}) \pm c_{abs} = b(t) \pm (b(t) \times c_{per} + c_{abs}) \tag{1}$$

The objective of this transformation is to identify the optimal parameters (c_{per} and c_{abs}) that align the transformed curve as closely as possible with the target curve, ensuring that the baseline accurately reflects the expected sensor behavior. This task can be formalized as an optimization problem, which is solvable through methods such as the least squares approach, as shown below, where $f(t)$ denotes the target threshold curve derived from historical sensor readings and L denotes the adjust window length.

$$\arg\min_{c_{per},c_{abs}} \frac{1}{L}\sum_{i=1}^{L}\left(b(t)\times(1\pm c_{per})\pm c_{abs} - f(t)\right)^2 \tag{2}$$

The optimal parameters, combined with those from time segmentation and target curve setting, constitute a complete configuration set. This parameter configuration set can then be delivered to sensor system operators as a configuration profile for manual deployment, or integrated into an automated monitoring script to directly update the threshold curve, enabling a fully autonomous and responsive sensing adjustment cycle that adapts to dynamic operational conditions.

3.4 Method workflow

In summary, our method is operated as a sequential, three-stage process for sensor data: the time segmentation of one monitoring cycle, target curve setting for each individual segment, and the transformation phase where the original sensor threshold is adjusted accordingly. The process is governed by a set of configurable parameters described in Table 1. These encompass both the choice of methods and their specific parameter values.

The overall workflow for sensor threshold optimization is shown in Fig. 2. The different parameter configurations are distilled into a structured knowledge base and serve as a critical resource for LLMs, enabling them to guide sensor system operators in adjusting different parameter combinations to suit specific sensing scenarios based on the given sensor data type.

Specifically, the LLM determines the optimum parameter configuration through the following structured process:

(1) Input Context: The LLM is prompted with the statistical features of the target sensor signal (e.g., noise variance, baseline drift rate, and periodicity) and the operational context (e.g., "temperature sensor in a stable environment" versus "fan speed with high-frequency noise").
(2) Reasoning & Recommendation: On the basis of its embedded prior knowledge, the LLM infers the appropriate smoothing aggressiveness and segmentation sensitivity. For instance, it recommends a wider smoothing window and stricter hysteresis for noisy signals to avoid false alarms, and a narrower window for stable signals to maintain sensitivity.

Table 1
Parameter configuration examples.

Stage	Parameter	Example
Time segmentation	segmentation_methods	Fixed-length or dynamic
	segmentation_parameters	Length = 8 h
Target curve setting	curve_setting_methods	Mid-point, SVM
	curve_setting_parameters	SVM parameters
Transformation	optimization_methods	Least squares

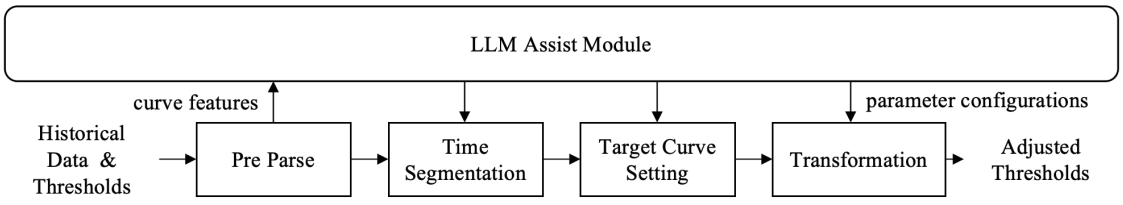


Fig. 2. Workflow of adjustment method.

(3) Output: The LLM outputs a structured configuration (e.g., smoothing polynomial degree, window size, hysteresis parameter), which is then directly applied to the algorithm.

4. Experiments and Discussion

In this section, we present the experimental results from a real-world deployment and discuss future work for this framework.

4.1 Real-world deployment

To evaluate the efficacy of our enhanced method, we performed a six-month deployment test in collaboration with an industry partner. The dataset used in this study was collected from a commercial bank's datacenter. The data acquisition was conducted internally by the bank's infrastructure team, and we accessed the data via their internal database categorized as "environmental control sensors," rather than reading directly from the physical sensor hardware. Owing to the bank's strict confidentiality and data privacy policies, the data is desensitized, and specific hardware details, such as sensor brands or exact models, cannot be disclosed. However, the physical measurement types of the metrics are explicitly defined. The evaluated sensor signals primarily consist of the following critical operational metrics collected from the datacenter infrastructure:

- temperature monitored by the cooling system: the ambient or coolant temperature readings that reflect the thermal state of the facility;
- fan speed of the cooling system: the rotational speeds (typically in RPM) of the HVAC/cooling unit fans;
- fan speed of the servers: the rotational speeds of the internal fans within the server nodes, which indicate the local computing load and thermal dissipation status;
- power consumption data monitored by the power distribution units: the real-time electrical power consumption metrics of the IT equipment.

Note that because these metrics inherently possess considerably different physical units and scales, and in strict compliance with the data desensitization policies, all raw sensor readings have been normalized into a dimensionless scale prior to being fed into the proposed algorithm. This preprocessing not only ensures data privacy but also guarantees that the threshold calculation and parameter optimization processes are universally applicable across different types of sensor signal without being biased by their original magnitude differences.

The original sensor data anomaly detection used a model based on normal distribution and five-point cubic polynomial smoothing. In contrast, our proposed framework introduces an LLM-assisted parameter configuration for sensor thresholds, leveraging optimized statistical parameters, a lower envelope to better capture the normal physical operating bounds, along with the Savitzky–Golay filter smoothing method,[22] and dynamic time segmentation. The transformation is then executed as a linear model solved by the least squares method to fit the sensor data variations. The results demonstrate the superior precision and reliability of our enhanced approach: the average *MAE* and *MPE* are reduced by 34.2 and 10.6%, respectively,

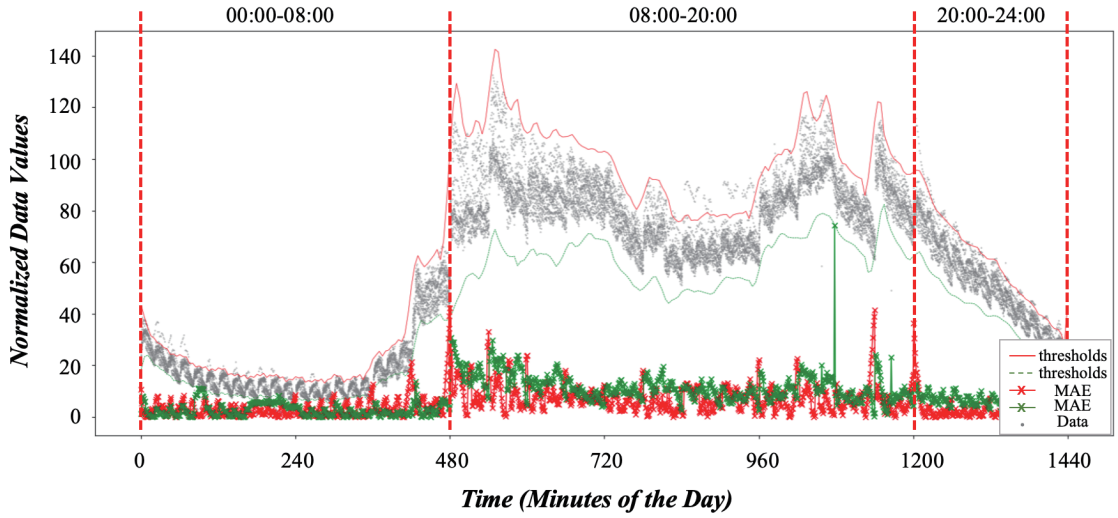
without generating any additional sensor false alarms. The reduction in *MAE* indicates that the optimized dynamic threshold fits the normal sensor signal much more closely. This improvement directly enhances detection accuracy in two ways: first, a threshold that closely tracks the normal signal prevents normal operational fluctuations from crossing the alert boundary, thereby improving precision; second, any genuine anomaly causes a distinct deviation from the threshold, making subtle anomalies easier to detect without being masked by an already loose baseline. An additional benefit is the reduction in operational overhead, as the LLM provides an accessible configuration guide that empowers the facility operators to manage the physical sensing infrastructure more effectively.

4.2 Time segmentation test

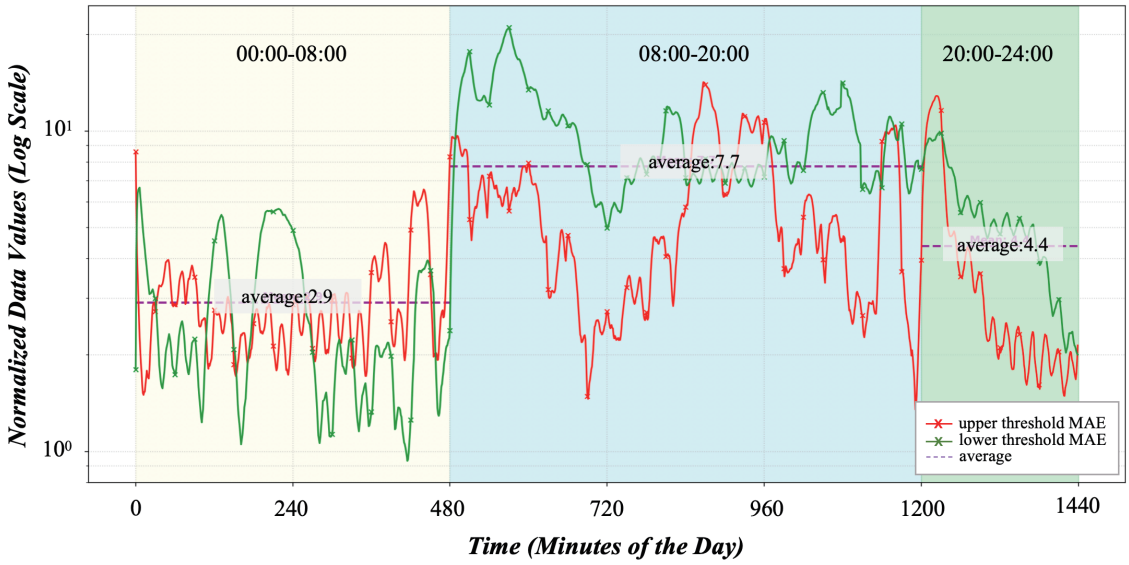
Figure 3 presents the time segmentation of the normalized total power consumption of a deployment unit based on the *MAE*. Figure 3(a) shows the original sensor data and threshold curves. The *x*-axis represents time (minutes of the day) and the *y*-axis represents the normalized data value. As illustrated in Fig. 3(a), the gray dots represent the normalized historical sensor data over a day (in min). The red solid and green dashed lines denote the calculated upper and lower thresholds, respectively, while the curve with X markers near the *x*-axis indicates the corresponding *MAE* values of the threshold fitting. The figure demonstrates how a time period is partitioned into three distinct segments, with the middle segment exhibiting a significantly greater *MAE* fluctuation than the others, indicating a shift in the operational state or environmental conditions affecting the sensor. Specifically, this middle segment corresponds to peak daytime hours, where increased computational workloads elevate the physical load on servers, consequently causing more pronounced fluctuations in physical sensing metrics such as equipment temperature and cooling fan speed.

Figure 3(b) provides a more detailed view of the *MAE* values across the three segments. The *x*-axis represents time (minutes of the day) and the *y*-axis uses a logarithmic scale to clearly display the *MAE* values. The red and green curves represent the *MAE* value of the upper and lower thresholds, respectively, with smoothing applied for better visualization. The average *MAE* values for the three segments are 2.9 (00:00-08:00), 7.7 (08:00-20:00), and 4.4 (20:00-24:00), highlighting the distinct fluctuation patterns: the middle segment has the highest average *MAE* and greater volatility, while the other two segments show relatively stable *MAE* values. Notably, even within the same time segment, the *MAE* curves still exhibit considerable fluctuations (especially in the middle segment). In practice, we can segment the time period into more fine-grained intervals, but here, we only divide it into three segments. This decision is made for two reasons: one is to align with the actual business (work schedule) and the other is due to system overhead (which is discussed later in this paper). This visualization emphasizes the varying *MAE* behavior across different operational periods, supporting the segmentation logic.

However, the efficacy of this segmentation is highly sensitive to parameter selection. Figure 4 illustrates the effect of the hysteresis parameter on the segmentation. The small subplot on the left displays the variation amplitude of the normalized power consumption, while the three plots



(a)



(b)

Fig. 3. (Color online) Segmentation of power consumption data based on *MAE* curve. (a) Data samples, thresholds, and *MAE* curves. (b) Smoothed *MAE* variation across different time segments.

to the right (share the same data and coordinate axes as Fig. 3) show the segmentation results as the hysteresis parameter value is progressively increased from left to right. In practice, creating an excessive number of time segments incurs a substantial computational overhead for the algorithm. To balance performance with practical feasibility, we therefore constrain the segmentation to a maximum of six segments in our deployment.

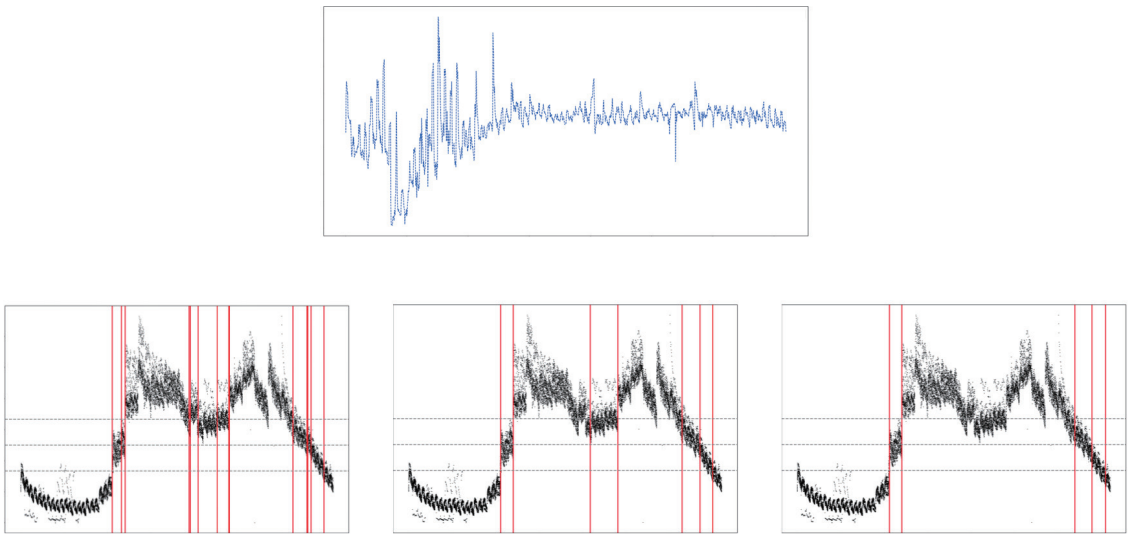


Fig. 4. (Color online) Different hysteresis parameters and segmentation results.

4.3 Target curve setting test

The target sensor baseline configuration depends on the generation algorithm (e.g., SVM, midpoint, and normal distribution) and the smoothing technique (e.g., polynomial fitting and moving average), both of which impact computational overhead.

Figure 5 shows a comparisons threshold fitting results obtained using different smoothing methods. This figure uses the same data and coordinate axes as Fig. 3 (x-axis: time in minutes of the day; y-axis: normalized data value). The gray dots represent the historical sensor data. For clarity of presentation, only the lower threshold curve (red solid line) and its *MAE* curve (red dashed line) are highlighted, while the upper threshold is omitted. The composite method (lower envelope + Savitzky–Golay filter) aligns best with the underlying sensor data distribution, effectively capturing both local sensor noise and global operational trends. However, this accuracy increases computational overhead compared with simpler techniques, highlighting the accuracy–efficiency trade-off. Furthermore, facility operators typically prioritize minimizing sensor false alarms to reduce manual workload, often preferring more lenient smoothing parameters.

4.4 Future work

For future work, we believe that our case study in dynamic thresholds can be generalized to other types of sensor data anomaly detection. By leveraging the continuously evolving capabilities of LLMs, this approach has the potential to provide progressively advanced AI solutions for a wider range of industrial scenarios, such as predictive maintenance for manufacturing robotics, structural health monitoring via vibration sensors, and environmental

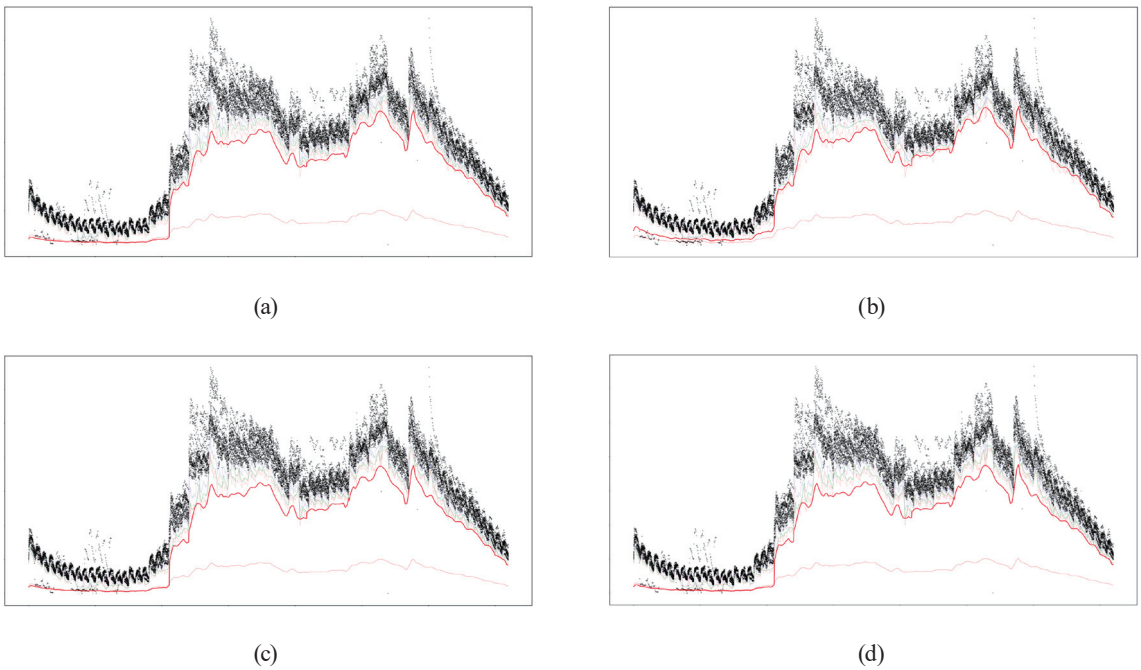


Fig. 5. (Color online) Threshold fitting results obtained using different smoothing methods. (a) 5-point cubic polynomial, (b) Savitzky–Golay filter, (c) lower envelope, and (d) lower envelope + Savitzky–Golay filter.

monitoring in smart facilities.[10–12] Concurrently, another promising research direction lies in moving beyond using LLMs solely for parameter recommendation. We aim to explore the deeper integration of LLMs with the detection algorithms themselves. For instance, in contexts involving equipment diagnostic logs and multi-modal sensor metrics,[23–25] the sophisticated text-processing capabilities of LLMs can be harnessed to correlate textual maintenance records with physical sensor readings, thereby significantly enhancing detection performance.

5. Conclusions

In this paper, we address the critical gap between traditional sensor O&M practices and the full realization of intelligent sensing systems by proposing a practical, incremental transition pathway. We advocate for a framework that augments, rather than replaces, existing sensor data anomaly detection systems, providing a clear evolutionary path from an AI-aware state to an AI-driven state. On the basis of our case study and experiments that refine traditional dynamic thresholds, we significantly enhance the detection accuracy of these legacy systems, effectively reducing false alarms caused by sensor noise and drift. Moreover, we incorporate an LLM-based recommendation system to simplify user interaction and guide operators in selecting optimization parameters for sensor baseline configuration. This integration lowers the barrier to adopting advanced techniques while preserving the system's interpretability and reliability. However, a limitation of this study is its current reliance on human validation for complex sensor drift patterns and its primary focus on univariate sensor data. The pending problem in this

research area is the handling of intricate multi-sensor correlations. While our approach addresses the rigidity of traditional methods, the remaining issue is achieving fully autonomous sensor data anomaly detection across heterogeneous sensor networks without manual oversight, which necessitates further investigation.

Acknowledgments

Our study was supported by the Industrial Big Data and Industrial Intelligence Engineering Technology Research and Development Center, Qinglan Project of Colleges and Universities in Jiangsu Province, the Scientific Research Foundation of Changzhou College of Information Technology, and Shanghai Key Technology Research and Development Program (Grant 25DZ3100700). We also sincerely appreciate the help and support provided by individuals during the research process and all the valuable comments given by the reviewers.

References

1 A. Levin, S. Garion, E. K. Kolodner, D. H. Lorenz, K. Barabash, M. Kugler, and N. McShane: Proc. 2019 IEEE Int. Congress on Big Data (IEEE, 2019) 165–169.
2 P. Notaro, J. Cardoso, and M. Gerndt: ACM Trans. Intell. Syst. Technol. **12** (2021) 1. https://doi.org/10.1145/3483424
3 H. Si, C. Pei, Z. Li, Y. Zhao, J. Li, H. Zhang, Z. Diao, J. Li, G. Xie, and D. Pei: Proc. 31st ACM Joint European Software Engineering Conf. and Symp. the Foundations of Software Engineering (ACM, 2023) 1635–1645.
4 H. Si, J. Li, C. Pei, H. Cui, J. Yang, Y. Sun, S. Zhang, J. Li, H. Zhang, J. Han, D. Pei, and G. Xie: Proc. 35th Int. Symp. Software Reliability Engineering (ISSRE) (IEEE, 2024) 61–72.
5 C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano: IEEE Software **33** (2016) 94. https://doi.org/10.1109/MS.2016.68
6 D. Kreuzberger, N. Kuhl, and S. Hirschl: IEEE Access **11** (2023) 31866. https://doi.org/10.1109/ACCESS.2023.3262138
7 J. Diaz-De-Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida: ACM Comput. Surv. **56** (2023) 1. https://doi.org/10.1145/3625289
8 Y. Dang, Q. Lin, and P. Huang: Proc. 41st Int. Conf. Software Engineering: Companion Proceedings (IEEE, 2019) 4–5.
9 H. Bao, K. Yin, L. Cao, S. Li, Y. Sun, H. Yi, R. Tang, Y. Hou, S. Wang, and D. Pei: J. Software **34** (2023) 4069 (in Chinese). https://doi.org/10.13328/j.cnki.jos.006876
10 D. Pei, S. Zhang, Y. Sun, and C. Pei: ZTE Technol. J. **30** (2024) 56 (in Chinese).
11 Y. Liu, C. Pei, L. Xu, B. Chen, M. Sun, Z. Zhang, Y. Sun, S. Zhang, K. Wang, H. Zhang, J. Li, G. Xie, X. Wen, X. Nie, M. Ma, and D. Pei: Proc. 33rd ACM Int. Conf. the Foundations of Software Engineering (ACM, 2025) 503–513.
12 Z. Jiang, T. Li, Z. Zhang, J. Ge, J. You, and L. Li: Mobile Networks Appl. **26** (2021) 2353. https://doi.org/10.1007/s11036-021-01832-3
13 J. Zhang, D. Zhou, and M. Chen: IEEE Trans. Cybern. **53** (2022) 4841. https://doi.org/10.1109/TCYB.2021.3140065
14 Y. Zhao and C. Zhao: Control Eng. Pract. **124** (2022) 105180. https://doi.org/10.1016/j.conengprac.2022.105180
15 Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei: Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (ACM, 2019) 2828–2837.
16 A. D. Myttenaere, B. Golden, B. L. Grand, and F. Rossi: Neurocomputing **192** (2016) 38. https://doi.org/10.1016/j.neucom.2015.12.114
17 Y. Zhu, C. M. Yeh, Z. Zimmerman, K. Kamgar, and E. Keogh: Proc. 2018 IEEE Int. Conf. on Data Mining (ICDM) (IEEE, 2018) 837–846.
18 P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin: Proc. VLDB Endow. **14** (2021) 1717. https://doi.org/10.14778/3467861.3467863

19 P. Malhotra, L. Vig, G. Shroff, and P. Agarwal: Proc. 23rd European Symp. Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) (Proceedings of ESANN, 2015) 89–94.
20 S. Tuli, G. Casale, and N. R. Jennings: Proc. VLDB Endow. **15** (2022) 1201. https://doi.org/10.14778/3514061.3514067
21 Z. Wang, C. Pei, M. Ma, X. Wang, Z. Li, D. Pei, S. Rajmohan, D. Zhang, Q. Lin, H. Zhang, J. Li, and G. Xie: Proc. ACM Web Conf. 2024 (ACM, 2024) 3096–3105.
22 R. W. Schafer: IEEE Signal Processing Mag. **28** (2011) 111. https://doi.org/10.1109/MSP.2011.941097
23 C. Bertero, M. Roy, C. Sauvanaud, and G. Trédan: Proc. 2017 IEEE 28th Int. Symp. Software Reliability Engineering (ISSRE) (IEEE, 2017) 351–360.
24 R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang, Q. Li, Y. Sun, D. Pei, T. Wei, Y. Xu, and Y. Liu: Proc. IEEE INFOCOM 2020-IEEE Conf. Computer Communications (IEEE, 2020) 2479–2488.
25 S. Zhang, Y. Ji, J. Luan, X. Nie, Z. Chen, M. Ma, Y. Sun, and D. Pei: Proc. 39th IEEE/ACM Int. Conf. Automated Software Engineering (IEEE, 2024) 1680–1692.

About the Authors

Ruming Tang received his Ph.D. degree in computer science and technology from Tsinghua University, China, in 2020. He is currently a lecturer at Changzhou College of Information Technology. His research interests focus on AI for IT Operations (AIOps) and the engineering implementation of machine learning and deep learning in industry.

Hua Lou received his master's degree from Zhengzhou University, China, in 2004. From 2013 to 2020, he was an associate professor at Changzhou College of Information Technology. Since 2021, he has been a professor at the same institution. His research interests are in software engineering, data processing, and anomaly detection.

Xingzhi Chang received his Ph.D. degree in control theory and control engineering from Northeastern University, China, in 2008. He is currently a professor and senior engineer at Changzhou College of Information Technology. His research interests include machine learning, image processing, and natural language processing.

Xidao Wen received his Ph.D. degree in information science from the University of Pittsburgh and completed his postdoctoral research at the Department of Computer Science and Technology, Tsinghua University. He is currently a Senior Algorithm Expert in Observability at Alibaba Cloud. He has long focused on the research and engineering implementation of machine learning, deep learning, and large language models in AIOps.

Kanglin Yin received his Ph.D. degree in software engineering from Tongji University, China. He is currently an associate researcher at the Key Laboratory for Satellite Digitalization Technology, Chinese Academy of Sciences. His research interests focus on AIOps, intelligent fault diagnosis, and data-driven system health management.