TRANSPORTA UN SAKARU INSTITŪTS

DEFENCE PERMITTED
Dean of Engineering Faculty
Professor, Ph.D.
Emmanuel Alejandro Merchan Cruz

_____

202__. year ____._____

MASTER THESIS

"ANALYSIS OF DATA SYNCHRONIZATION METHODS AND FRAMEWORK IN WIRE ARC ADDITIVE MANUFACTURING (WAAM)"

Computer Sciences, 45483

Student: **Sergejs Kopils**
(st. c. ***83519****)*
_____ 202__. year ___._____
(signature)

Supervisor: *Professor, Dr. sc. ing.*
Mihails Savrasovs
_____ 202__. year ___._____
(signature)

Consultant: *Invited lecturer, Mg. sc. ing.*
Arsenii Kisarev
_____ 202__. year ___._____
(signature)

Riga 2025

CONFIRMATION

I, Sergejs Kopils, confirm that I have developed the given graduation thesis individually for obtaining the degree of Computer Sciences. By my signature I verify that the given graduation thesis is written independently and there are no violations regarding the rights of intellectual property of other persons or plagiarism. The used papers and data sources of other authors are indicated in references. The text of the submitted paper has never been submitted partially or fully to other commission for theses evaluation. I confirm that the electronic version of the submitted thesis meets completely the text of the submitted paper copy of the thesis.

_____ _____

(signature) *(name, surname)*

CONSENT

I, Sergejs Kopils, hereby agree that my graduation thesis abstract can be used by AS "TRANSPORTA UN SAKARU INSTITŪTS" (hereinafter – the Institute), reg. No. 40003458903, Valērijas Seiles 1, Riga, LV-1019, in study and scientific work, for improving and optimizing the quality of the study process.

I hereby agree that the graduation thesis abstract will be publicly available on the Institute's website www.tsi.lv, in the Institute's Library or in academic units.

_____ _____

(signature) *(name, surname)*

ANOTĀCIJA

Maģistra darbs „Datu sinhronizācijas metožu un ietvarstruktūras analīze stieples loka piedevu ražošanas (WAAM) procesā”. Darba autors: Sergejs Kopils. Zinātniskais vadītājs: Dr. sc. ing., profesors Mihail Savrasovs.

Darbs „Datorzinātnes” grāda iegūšanai: 87 lpp., 31 att., 7 tab., 67 izmantotās lit. avot., 6 piel.

LOKA METINĀŠANAS PIEVIENOŠĀS RAŽOŠANAS TEHNOLOĢIJA (WAAM), AIRĀKU DATU PLŪSMU SINHRONIZĀCIJA, SLĀŅU SEGMENTĀCIJA, ANOMĀLIJU NOTEIKŠANA, STRUKTŪRA

Šajā maģistra darbā tiek piedāvāts modulārs ietvars vairāku datu plūsmu sinhronizācijai loka metināšanas piedevu ražošanā (WAAM), aptverot 3D skenēšanu, procesa žurnālus un robota vadības failus. Tā kā avoti ir asinhroni un tiem nav kopīgu laika zīmogu, sinhronizācija ir būtiska kvalitātes kontrolei, anomāliju noteikšanai un procesa optimizācijai.

Ietvarā tiek novērtētas vairākas slāņu segmentācijas metodes — klasterizācija (KMeans, DBSCAN, hierarhiskā) un ģeometriskā pieeja. Par atsauci tiek izmantota metode, kas balstīta uz strukturētiem skenēšanas metadatiem. Pēc segmentācijas dati tiek apvienoti vienotā kopā un papildināti ar procesa parametriem, piemēram, loka ieslēgšanas/izslēgšanas signāliem un kustības ātrumu.

Anomāliju noteikšanas modulis identificē ģeometriskas novirzes, piemēram, Z iegrimumus un pārtraukumus lokā. Rezultāti tiek attēloti interaktīvā 3D informācijas panelī, kas veidots ar Plotly.

Piedāvātais ietvars nodrošina praktisku un paplašināmu pieeju WAAM datu sinhronizācijai un analīzei, un tas ir piemērojams arī citiem kibervides fizisko sistēmu un robotizētu procesu gadījumiem ar daudzu sensoru integrāciju.

ABSTRACT

Master's Thesis „Analysis of Data Synchronization Methods and Framework in Wire Arc Additive Manufacturing (WAAM)". Author of the paper: Sergejs Kopils. Scientific supervisor: Dr. sc. ing., professor Mihail Savrasovs.

Paper for obtaining the degree „Computer Sciences": 87 pp., 31 img., 7 tabl., 67 ref., 6 app.

WIRE ARC ADDITIVE MANUFACTURING (WAAM), MULTI-STREAM DATA SYNCHRONIZATION, LAYER SEGMENTATION, ANOMALY DETECTION, FRAMEWORK

This thesis presents a modular framework for synchronizing multi-stream data in Wire Arc Additive Manufacturing (WAAM), including 3D scans, process logs, and robot control files. Due to asynchronous sources without shared timestamps, synchronization is critical for quality assurance, anomaly detection, and process optimization.

The framework evaluates multiple layer segmentation methods—clustering (KMeans, DBSCAN, hierarchical) and geometry-based. A reference method using structured scan metadata is used to assess accuracy. After segmentation, data streams are merged and enriched with process signals like Arc On/Off and travel speed.

An anomaly detection module flags irregularities in geometry, such as Z-depressions and arc gaps. Results are visualized in an interactive 3D dashboard built with Plotly.

The framework offers a practical, extensible approach to WAAM data synchronization and analysis, applicable to broader cyber-physical and robotic systems with multi-sensor integration.

CONTENTS

ABBREVIATIONS USED IN THE PAPER

ARC - Local geometric segment within a WAAM layer, extracted by slicing along the X-axis; used for spatial anomaly detection (not to be confused with electric arc)

ARC ON/OFF - Welding Arc Activation/Deactivation Signal (Arc On = 1, Arc Off = 0)

CAD - Computer-Aided Design

CPS - Cyber-Physical Systems

CSV - Comma-Separated Values

DBSCAN - Density-Based Spatial Clustering of Applications with Noise

G-code - Geometric Code (robot control program)

GT - Ground Truth

IoU - Intersection over Union

JSON - JavaScript Object Notation

KDE - Kernel Density Estimation

LOF - Local Outlier Factor

ML - Machine Learning

Np - NumPy (Numerical Python library, used as np)

pd - pandas (Python data analysis library, used as pd)

PDF - Portable Document Format

px - Plotly Express (used for visualization, imported as px)

ROI - Region of Interest

TS -Travel Speed

UI - User Interface

VS Code - Visual Studio Code

WAAM - Wire Arc Additive Manufacturing

WFS - Wire feed speed

X, Y, Z - Coordinate Axes

3D - Three-Dimensional

INTRODUCTION

In modern manufacturing environments, particularly those involving additive technologies such as Wire Arc Additive Manufacturing (WAAM), the integration of data from multiple sources has become increasingly critical. The operation of such systems results in the generation of diverse and heterogeneous data streams—including high-resolution 3D point clouds, process logs, robot trajectories, and metadata from machine controllers (Raut & Taiwade, 2021). These streams are typically asynchronous, vary in sampling rate, and often lack consistent time-stamp alignment, making their integration and analysis a challenging task (Kumar et al., 2022).

Traditional manufacturing quality assurance methods often rely on isolated sensor checks or manual inspection, which are time-consuming and prone to error. At the same time, the emergence of digital twins, predictive maintenance systems, and closed-loop process control frameworks requires more than just raw data; it demands structured, synchronized, and interpretable information that can reliably reflect the state of the process in both real time and post-analysis contexts (Tao et al., 2019).

This thesis addresses the research gap by proposing a modular, data-driven framework for multi-stream synchronization and anomaly detection in WAAM processes. The work combines geometry-based segmentation, clustering algorithms, rule-based anomaly logic, and interactive 3D visualization to create a complete pipeline—from raw data to actionable insight. A core emphasis is placed on interpretability and modularity, enabling the framework to be adapted for other forms of additive manufacturing or automated production systems, as shown in the modular design of the proposed framework.

The framework developed in this study processes three main types of data: 3D scan data captured after each layer, process logs detailing Arc On/Off events and tool coordinates, and control data representing job segments or planned trajectories. Several synchronization strategies are implemented and compared, including a geometry-based segmentation method that proved to be the most effective. Six methods for layer clustering are evaluated based on accuracy and robustness, and an anomaly detection module is developed using interpretable rules that flag under-deposition, arc duplication, and structural gaps.

A custom visualization dashboard supports result interpretation and allows interactive inspection of the scan structure and anomalies. The visual interface helps to validate automated decisions and supports human-in-the-loop diagnostics—an increasingly important aspect of modern industrial data workflows.

This work contributes both a technical framework and a methodological approach for integrating spatial and process data in an interpretable and scalable way. The resulting system

supports not only post-process analysis but also forms a foundation for integration with real-time monitoring systems, making it highly relevant for the future of intelligent, data-centric manufacturing.

Aim of the research

To develop and evaluate a modular framework for synchronizing geometric, process, and control data streams in WAAM systems and to detect anomalies in the deposition process using rule-based logic and interactive visualization.

Research objectives

- To analyze existing methods for synchronizing multi-stream manufacturing data in WAAM and similar robotic additive manufacturing processes.
- To design a flexible synchronization framework applicable to point cloud scans, process logs, and toolpath metadata.
- To implement and compare multiple segmentation and clustering methods for deposited layer's identification.
- To develop a rule-based anomaly detection module for flagging geometric and process deviations.
- To create a visualization interface that supports diagnosis and validation of results.

Research questions

Which synchronization method provides the most reliable alignment between scan and process data in the absence of timestamps?

How effective is geometric segmentation compared to clustering-based approaches in detecting deposition layers?

Can rule-based anomaly detection reliably identify irregularities in the WAAM process based solely on geometric data?

Object and subject of the research

Object: Multi-stream data collected during WAAM-based additive manufacturing processes.

Subject: Synchronization methods, segmentation techniques, and anomaly detection algorithms applied to scan-process-control data integration.

Methodology

The methodological framework of this research integrates a combination of computational techniques and evaluation strategies to address the objectives outlined in Section 1.3. The chosen

methods reflect the structure of the multi-stream dataset and the need for interpretable, geometry-based analysis in an industrial context.

The methodology includes the following components:

Geometric segmentation: A distance-based method for detecting deposition segments by identifying spatial discontinuities in point cloud data. This forms the basis for synchronization in the absence of temporal alignment.

Clustering algorithms: Unsupervised machine learning techniques including KMeans (with predefined and automatically estimated cluster counts), DBSCAN, and hierarchical clustering are applied to group scan data vertically into deposition layers.

Rule-based anomaly detection: Domain-driven logic is used to flag structural inconsistencies in layers and arcs—defined here as narrow geometric subsegments within a layer, obtained by slicing along the X-axis. These arcs represent localized regions of the scanned surface and are used to detect anomalies such as Z-depressions, segment gaps, or duplicated paths, based on computed geometric features.

Quantitative evaluation: The accuracy of each synchronization and clustering method is assessed through established metrics, including Accuracy, Intersection over Union (IoU), Precision, Recall, and F1-score.

Visualization and interaction: 3D point cloud data is rendered using Plotly and Dash to support qualitative inspection and validation of segmentation quality and anomaly relevance.

Case study analysis: Visual diagnostics and example dataset are used to confirm the consistency between automated detection and observable geometric deviations.

This integrated methodological approach ensures that the results are both reproducible and applicable across similar multi-stream manufacturing systems.

Expected result

An approbated framework that allows accurate synchronization of multi-stream WAAM data and reliable detection of anomalies in deposition geometry. The resulting system will support visual inspection, metric-based comparison, and scalable adaptation to other manufacturing scenarios.

Research strategy and identification of the research gap

The objective of this research is to develop a robust framework for data synchronization and anomaly detection in multi-stream systems, with a specific focus on Wire Arc Additive Manufacturing (WAAM). Due to the inherent complexity and data heterogeneity of WAAM processes, this study emphasizes the integration of sensor, process, and control data to enable adaptive correction mechanisms. To ensure a well-grounded and methodologically sound foundation, a comprehensive research strategy was adopted, incorporating structured literature review techniques and semantic analysis of academic sources.

Literature search strategy

The literature queries were structured around six thematic clusters that align with the objectives of this thesis. These clusters include topics such as WAAM layer segmentation, synchronization of multi-stream data, anomaly detection, process correction, and modular frameworks for industrial integration. Each cluster was assigned a query ID and built using a targeted set of primary and secondary keywords. The results of each query in Scopus are summarized in Table 1, showing both the focus area and the number of relevant publications retrieved.

Table 1 Thematic clusters used for Scopus queries and example keywords.

Query ID	Research focus area	Example keywords used	Total results (Scopus)
Q1	WAAM and Layer Segmentation	"waam", "layer segmentation", "3D scan"	12
Q2	Multi-stream Synchronization in Additive Manufacturing	"multi-stream", "synchronization", "alignment"	1
Q3	Anomaly Detection in AM Processes	"defect detection", "anomaly", "real-time monitoring"	171
Q4	Process Correction and Adaptive Control	"adaptive control", "feedback", "correction"	171
Q5	Modular Data Processing Frameworks	"framework", "data processing", "modular systems"	9
Q6	Feedback-based Anomaly Detection and Adaptation	"feedback loop", "process adaptation"	400

Each cluster was defined through a combination of primary keywords (e.g., "wire arc additive manufacturing", "framework", "synchronization", "anomaly detection") and secondary keywords or synonyms (e.g., "multi-sensor", "defect identification", "real-time adjustment"). The queries were iteratively refined to maximize relevance and specificity. Titles and abstracts were extracted from 6 query files, yielding 771 unique entries.

Abstract screening and semantic evaluation

To assess the relevance of the collected literature, a semantic evaluation strategy was employed. Each abstract was analyzed using a simulated large language model (LLM) logic, checking for conceptual alignment with the following domains:

- Data synchronization in time-sensitive industrial processes
- Anomaly or defect detection using unsupervised or real-time techniques
- Multi-stream sensor or process data integration
- Process correction or feedback-driven control in AM
- WAAM-specific methods or case studies
- Modular or integrated frameworks for industrial data management

The analysis labeled each abstract as either relevant or not, and provided a short justification. Out of 771 entries, approximately 64% were found to be conceptually relevant to at least one of the thematic categories. This stage helped refine the corpus for deep reading and detailed review.

Structuring the Bibliometric dataset

Following the relevance labeling, all articles marked as relevant were compiled into a focused dataset. Metadata such as publication year, source, authorship, and keywords were extracted to support bibliometric exploration. This facilitated the identification of frequently cited authors, journals of high relevance, and publication trends over time. Patterns in geographic distribution and institutional contribution were also noted (see Table 2).

Table 2 Bibliometric observations derived from relevant literature.

Metric	Observation
Most frequent keyword	"Additive manufacturing"
Emerging term	"Multi-stream synchronization"
Top countries	Germany, USA, China
Most cited journal	Additive Manufacturing (Elsevier)
Publication peak	2021–2023

Identification of the research gap

The gap emerged through a comparative review of existing work:

- **Most WAAM studies** emphasize geometry, thermal control, or microstructure, but few focus on synchronized multi-source data integration.
- **Anomaly detection models** are often limited to single-stream (e.g., only thermal imaging or only process-specific) without cross-validation between sources.

- **Layer segmentation and clustering methods** are applied, but rarely linked to actual process control or G-code parameters.
- **Frameworks** for modular data analysis exist, but they are either generic or designed for other domains (e.g., CNC, polymer AM) and not suitable to the multi-stream reality of WAAM.

Thus, a clear research gap was identified: there is no integrated approach combining data synchronization, anomaly detection correction for WAAM using multi-stream sensor and process data.

Strategic outcome

This strategy resulted in the construction of a curated, semantically filtered knowledge base, forming the empirical and theoretical foundation of the thesis. It not only validated the novelty of the proposed framework but also ensured its alignment with the most pressing challenges and underexplored areas within the field.

The research gap was not imposed arbitrarily—it was derived through an iterative and systematic process of query refinement, semantic screening, and comparative synthesis. This methodological rigor reinforces the originality and potential impact of the proposed solution.

Conclusion

By implementing a structured search and semantic filtering process, this study was able to identify a significant gap in the current state of research. This gap provides a justifiable and meaningful direction for the development of a novel data synchronization and process correction framework in WAAM, addressing both academic interest and industrial need.

1. LITERATURE REVIEW

This chapter reviews the current research and industrial practices in the synchronization of multi-stream data, anomaly detection in manufacturing systems, and frameworks that integrate these capabilities. The review covers general synchronization techniques, data processing methodologies, existing industrial and academic frameworks, and application domains beyond WAAM.

1.1. Multi-stream data in cyber-physical systems

Modern Cyber-Physical Systems (CPS) are integrated networks that combine physical processes with computation and communication capabilities. These systems are prevalent in critical sectors such as manufacturing, transportation, robotics, and healthcare, where real-time sensing and control are required. A core characteristic of CPS is their reliance on multiple asynchronous data streams to monitor, interpret, and influence physical operations (NIST, 2016).

These multi-stream data sources typically include:

- sensor outputs (e.g., temperature, vibration, laser scans),
- control commands (e.g., G-code instructions to machines),
- machine logs (e.g., voltage, current, torque),
- feedback from external systems (e.g., quality inspection reports).

The heterogeneity of these data streams creates several challenges. First, they may differ significantly in sampling rate (milliseconds for sensors vs seconds for logs), data format (structured vs semi-structured or unstructured), coordinate systems (e.g., local vs global frames), and timestamp standards (synchronized vs unsynchronized clocks). This complexity makes synchronized analysis and fusion of these streams difficult, but a necessity for tasks such as anomaly detection, predictive maintenance, or adaptive control (Akkaya, 2016).

In industrial CPS, particularly in Wire Arc Additive Manufacturing (WAAM), multi-stream data integration becomes even more crucial. WAAM is an additive process where metal is deposited layer-by-layer using a wire and arc as a heat source. The process is driven by robotic manipulators guided via control scripts (usually G-code), while sensors simultaneously measure dynamic process parameters such as:

- geometric deviations (via laser or structured light 3D scanners),
- thermal signals (from thermocouples or IR cameras),
- arc stability metrics (voltage/current logs),
- and motion parameters (e.g., tool velocity, torch position).

One of the primary challenges in WAAM is the alignment of these asynchronous data sources. A typical build may consist of thousands of scanned points, collected post-layer, that must be matched precisely to the process data gathered during layer deposition (Hauser et al., 2025). Discrepancies in timing can lead to incorrect interpretations — for instance, attributing a geometric defect to the wrong process condition.

Moreover, coordinate transformation is essential. While control data may use robot-centered frames, scanners usually output global 3D point clouds. Ensuring that these geometries align spatially and temporally is a prerequisite for layer-level validation, defect detection, and adaptive correction. Without synchronized data, intelligent decisions — such as adjusting torch path in real time or predicting faults — cannot be reliably made.

To handle these challenges, data fusion frameworks and middleware platforms have been introduced (NIST, 2016). These systems often include:

- time synchronization modules (e.g., using Network Time Protocol or hardware clocks),
- data adapters that convert various formats to a common structure,
- and real-time brokers that stream and align data from heterogeneous sources.

Recent research also investigates the use of machine learning (ML) and digital twins to model the complex interdependencies in multi-stream CPS. For instance, supervised models trained on synchronized data can detect early-stage anomalies or recommend optimal process parameters (Yildiz et al., 2023; Chaturvedi et al., 2021).

WAAM-specific implementations include the multi-robot scan-n-print architecture, where sensor and control data are dynamically co-registered in real time (Lu et al., 2024). This not only allows for defect mitigation but also supports in-situ correction, making WAAM more viable for structural and aerospace components.

In summary, multi-stream data integration in CPS, and especially in WAAM, is a foundational element for advancing intelligent and autonomous manufacturing systems. The ability to synchronize, interpret, and act upon heterogeneous data sources is a key enabler for Industry 4.0 and beyond. However, this remains a technically demanding area that requires interdisciplinary approaches combining signal processing, data engineering, and cyber-physical modeling.

1.2. Synchronization techniques

In Cyber-Physical Systems (CPS), particularly within domains like Wire Arc Additive Manufacturing (WAAM), the integration of heterogeneous data streams is pivotal for ensuring system coherence and operational efficiency. Synchronization techniques are employed to align these diverse data sources, addressing challenges posed by varying sampling rates, data formats,

and temporal discrepancies. This section provides a comprehensive overview of prevalent synchronization methods, analyzing their principles, applications, advantages, and limitations.

1.2.1 Time-based synchronization

Time-based synchronization aligns data streams by matching timestamps, assuming that all system components share a common time reference. Protocols such as the Network Time Protocol (NTP) and the Precision Time Protocol (PTP) are commonly used for this purpose. NTP, introduced by Mills (1991), has been widely adopted in distributed computing environments for synchronizing clocks with millisecond-level accuracy. For more precise industrial applications, IEEE 1588 PTP and its generalized form (gPTP) under IEEE 802.1AS are preferred (Bharti and Kadyan, 2024).

Despite their wide usage, these protocols are prone to issues such as clock drift, network jitter, and transmission delays, particularly under real-time constraints. Lenzen et al. (2010) highlight that achieving theoretical bounds of synchronization in practice is often constrained by hardware and topological limits. Moreover, Annessi et al. (2018) have shown that PTP-based systems can be vulnerable to delay-based cyberattacks, raising concerns about their reliability in critical infrastructure. Alternatives such as cryptographic synchronization protocols (Narula and Humphreys, 2017) and skewless convergence schemes (Mallada et al., 2014) have been proposed to improve both robustness and performance.

1.2.2 Event-based synchronization

Event-based synchronization uses observable system-level events as temporal anchors to align asynchronous data streams. In the context of Wire Arc Additive Manufacturing (WAAM), electrical signals such as Arc On and Arc Off—typically logged by the robot controller or welding power supply—can serve as reliable synchronization triggers for aligning geometric, thermal, and process monitoring data. This method is especially useful in situations where global clock synchronization is unavailable, unreliable, or prohibitively complex to implement (Hauser et al., 2025).

The accuracy of event-based synchronization depends on the timely and consistent detection of events across all relevant data channels. Potential sources of misalignment include sampling jitter, signal latency, and event detection errors due to sensor noise or software delays. Nevertheless, this method remains highly practical in industrial scenarios, particularly in embedded and resource-constrained systems, due to its relative simplicity and hardware independence.

Recent developments have introduced neuromorphic sensing techniques for event-driven monitoring in high-speed manufacturing. Mascareñas et al. (2024) demonstrated the use of

asynchronous, event-based dynamic vision sensors to capture rapid changes in WAAM processes, such as melt pool behavior. These sensors provide high temporal resolution and robustness to lighting conditions, supporting their use for event-based data synchronization and in-process monitoring applications.

1.2.3 Geometry-based matching

In scenarios where temporal synchronization is unreliable, missing, or asynchronous, geometry-based matching offers a robust alternative for aligning data across multiple streams. This method is particularly relevant in Wire Arc Additive Manufacturing (WAAM), where 3D scans, robotic toolpaths, and sensor data may not share a common temporal reference. Geometry-based synchronization relies instead on spatial alignment between corresponding features—such as surfaces, contours, or edges—within the datasets.

A foundational technique in this domain is the Iterative Closest Point (ICP) algorithm, introduced by Besl and McKay (1992), which iteratively minimizes the Euclidean distance between corresponding points in two datasets to compute an optimal rigid transformation. However, classical ICP assumes good initial alignment and may converge to local minima. Numerous improvements have since been proposed. For example, point-to-plane ICP considers surface normals to improve convergence in structured environments (Rusinkiewicz and Levoy, 2001), while Generalized ICP (Segal et al., 2009) combines point-to-point and point-to-plane distances in a probabilistic framework for more robust registration.

To address the limitations of purely geometric proximity, researchers have introduced feature-based registration techniques. Descriptors such as SHOT (Signature of Histograms of Orientations) characterize local geometries, enabling more reliable matching even in noisy or partially occluded data (Tombari et al., 2010). Fu et al. (2024) implemented a SHOT-based matching method for point cloud registration in WAAM, achieving higher accuracy and robustness compared to traditional ICP. These methods, however, generally require dense spatial data and high computational resources, making them better suited for offline analysis.

Recent advances in deep learning have led to the development of data-driven approaches for point cloud registration. Methods such as DeepICP (Lu et al., 2019) and Deep Closest Point (DCP) (Wang and Solomon, 2019) use neural networks to learn feature representations and establish correspondences between point clouds. These models demonstrate improved performance in scenarios involving noise, occlusion, or poor initial alignment. Zhou et al. (2022) introduced DeepMatch, a lightweight network architecture that maintains state-of-the-art accuracy with reduced computation time, thereby enhancing feasibility for real-time applications.

In the WAAM context, geometry-based matching facilitates several critical functions:

- Aligning scanned layers with robot toolpaths to evaluate deposition accuracy;
- Detecting geometric deviations and anomalies in the printed structure;
- Fusing sensor and control data for integrated, multi-modal analysis.

For example, accurately aligning 3D scans with G-code-generated toolpaths enables the evaluation of build integrity and surface accuracy. Moreover, integrating geometric alignment into process control systems supports real-time error detection and adaptive correction.

Despite its advantages, geometry-based matching requires high-fidelity 3D data and is computationally intensive, limiting its applicability in constrained or real-time environments. Hybrid approaches that combine geometric, temporal, or event-based methods are increasingly used to mitigate these limitations and enhance robustness.

1.2.4 Hybrid methods

In the context of Cyber-Physical Systems (CPS) and particularly in data-intensive domains like Wire Arc Additive Manufacturing (WAAM), hybrid synchronization techniques have emerged as a powerful strategy to overcome the limitations of individual synchronization methods. These approaches integrate two or more techniques—such as time-based, event-based, and geometry-based synchronization—to leverage their complementary strengths while minimizing weaknesses.

For instance, event-based synchronization can be used to perform coarse alignment by detecting Arc On/Off signals, after which fine adjustments can be made using Iterative Closest Point (ICP) alignment based on geometric data. Similarly, discrepancies in timestamp alignment can be corrected post-hoc by spatial verification of sensor trajectories or part geometry (Martens et al., 2022; Shao, Li and Tan, 2023).

Such hybrid frameworks are particularly beneficial in smart manufacturing environments where multiple asynchronous data streams (e.g., robot telemetry, sensor feedback, process logs, and quality inspection data) need to be continuously fused for real-time decision-making and control. For example, Guarro and Sanfelice (2022) proposed a hybrid control algorithm that adaptively switches between synchronization strategies based on real-time signal confidence and process conditions.

Furthermore, the integration of machine learning into hybrid synchronization workflows allows systems to dynamically learn correlations between signals, identify latent synchronization points, and adaptively tune parameters over time. Zhou et al. (2021) demonstrated the use of deep learning for multimodal sensor synchronization in robotics, showing improved robustness against noise and timing jitter.

Despite these advantages, hybrid methods present significant challenges in system design, validation, and computational overhead. Combining synchronization layers increases architectural

complexity, often requiring middleware platforms, modular signal pipelines, and real-time data quality monitoring. Gódor et al. (2022) noted that the introduction of 5G-enabled time synchronization in Industry 4.0 systems adds another layer of complexity, but can facilitate highly granular and reliable hybrid synchronization schemes. However, in process requiring ultra-low latency, hardwired synchronization remains more robust and reliable than wireless solutions such as 5G.

In summary, hybrid synchronization methods are increasingly considered best practice in environments with high reliability demands and heterogeneous data sources. Their implementation, however, requires a balance between robustness, computational feasibility, and system maintainability.

1.3. Data processing methodologies

Effective synchronization of multi-stream data in Wire Arc Additive Manufacturing (WAAM) systems necessitates comprehensive preprocessing to ensure data integrity and coherence. This section delves into the key data preparation techniques employed to ready diverse datasets for synchronization.

1.3.1 Layer segmentation

Layer segmentation is pivotal in identifying deposition layers as a base functional object, within WAAM processes. Clustering algorithms such as K-Means and DBSCAN are commonly utilized to group points based on their vertical (Z-axis) positions. For instance, DBSCAN effectively identifies dense regions in point cloud data, making it suitable for segmenting layers in WAAM applications (Yan, 2018). Additionally, Kernel Density Estimation (KDE) aids in detecting peaks corresponding to individual layers, facilitating accurate segmentation. Advanced methods like Density Peak Clustering (DPC) have also been explored for their efficacy in handling datasets with varying densities and complex structures.

1.3.2 Noise and outlier removal

Noise and outliers in point cloud data can significantly impair the accuracy of synchronization. Techniques such as the Local Outlier Factor (LOF) algorithm assess the local density deviation of data points, effectively identifying anomalies (Breunig et al., 2000). Statistical Outlier Removal (SOR) filters, which analyze the distribution of point distances, are also employed to eliminate spurious data. Recent advancements have introduced machine learning approaches, like PointCleanNet, which leverage deep learning to denoise and remove outliers from dense point clouds (Rakotosaona et al., 2019).

1.3.3 Trajectory cleaning

In WAAM, not all robot trajectories correspond to material deposition. Trajectory cleaning involves filtering out segments unrelated to deposition, often by analyzing Arc On/Off signals or applying movement length thresholds. This process ensures that only relevant trajectory data is considered during synchronization, enhancing the accuracy of subsequent analyses. For example, Hauser et al. (2021) discuss the integration of sensor frameworks in robot-based WAAM cells to monitor and refine deposition paths effectively.

1.4 Anomaly detection in multi-stream industrial systems

Anomaly detection plays a vital role in ensuring safety, process stability, and quality assurance in complex industrial systems, including Wire Arc Additive Manufacturing (WAAM). Due to the multi-stream and high-dimensional nature of sensor and control data, advanced anomaly detection strategies are required beyond traditional rule-based monitoring.

1.4.1 Traditional approaches

Classical anomaly detection techniques include:

- Statistical process control (SPC) methods such as Shewhart, CUSUM, and EWMA control charts, which are effective when process boundaries are well defined (Montgomery, 2020).
- Distance-based algorithms, including k-nearest neighbors (k-NN) and Local Outlier Factor (LOF), which detect outliers based on local density. These approaches are simple but may not scale well with high-dimensional or multi-modal data (Breunig et al., 2000).
- Classical machine learning techniques like Isolation Forest or One-Class SVM, which are suitable when anomaly labels are scarce or absent (Liu, Ting and Zhou, 2008).

1.4.2 Deep learning-based methods

With the growth of data complexity and volume in smart manufacturing, deep learning methods have emerged as state-of-the-art:

- Variational Autoencoders (VAE) have been effectively applied to multivariate time series anomaly detection. For instance, Yokkampon et al. (2022) proposed a Multi-Scale Convolutional Variational Autoencoder (MSCVAE) that integrates attention-based ConvLSTM networks to capture temporal patterns and inter-variable correlations. This approach demonstrated robustness and high accuracy across multiple industrial datasets, highlighting its applicability in complex manufacturing environments.
- Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) architectures are highly effective for temporal sequences. Hundman et al. (2018) used LSTM-based

prediction models to detect anomalies in spacecraft telemetry, illustrating applicability in real-time, high-volume streams.

- Variational Autoencoders (VAE) allow probabilistic modeling of uncertainty and latent structures. For example, Pan et al. (2020) proposed a sliding-window convolutional VAE model for real-time anomaly detection in robotic manufacturing systems.

1.4.3 Application to WAAM and industrial contexts

In WAAM systems, the anomaly detection challenge is magnified by:

- Multi-sensor inputs (e.g., voltage, current, temperature, scan geometry);
- Lack of labeled data, which demands unsupervised or semi-supervised methods;
- Real-time constraints, which require both accuracy and efficiency.

Few commercial platforms support deep anomaly detection on raw multi-stream data in real time. Therefore, academic research often proposes lightweight, modular solutions that fuse process parameters with geometric or image-based features to detect spatiotemporal deviations.

These limitations emphasize the importance of developing adaptable, lightweight frameworks—such as the one proposed in this thesis—that can operate reliably under the multi-sensor conditions typical of WAAM systems.

1.4.4 Anomaly detection in WAAM layers: geometric, thermal, and multisensor approaches

Ensuring the quality of each layer in Wire Arc Additive Manufacturing (WAAM) is crucial for mechanical integrity and process stability. Anomalies such as height deviations, porosity, arc instability, or trajectory drift can significantly compromise final part performance. Recent studies emphasize the integration of geometric, thermal, and sensor data streams—alongside machine learning methods—for accurate real-time anomaly detection.

Geometric layer analysis

A common strategy for anomaly detection is to compare each deposited layer with its reference CAD geometry. Using 3D scanning and point cloud alignment algorithms such as Iterative Closest Point (ICP), deviations in height, thickness, and surface flatness can be identified. For example, RAMLAB has developed the MaxQ platform that combines 3D laser scanning and thermal imaging to assess layer geometry and detect defects during WAAM (RAMLAB, 2023).

Thermal and arc signal monitoring

Thermal anomalies like hot or cold spots are identified using infrared cameras and pyrometers. Voltage and current fluctuations can indicate process instability such as arc wandering or inconsistent wire feed. Signal processing methods—e.g., Fast Fourier Transform (FFT) and

Discrete Wavelet Transform (DWT)—have been employed to detect frequency-domain anomalies during the welding process (Mattera et al., 2024).

Multisensor and machine learning approaches

Combining electrical, thermal, acoustic, and geometric signals enables robust anomaly detection in WAAM systems. Hauser et al. (2022) demonstrated that acoustic emissions can be used to monitor process stability and detect deviations during deposition. Machine learning models such as Isolation Forest and Local Outlier Factor (LOF) can detect abnormal patterns in high-dimensional process data. Vozza et al. (2024) showed that these models effectively detect voltage/current deviations in Invar 36 WAAM and can be enhanced with frequency-based features.

Explainability and feature importance

Explainable models like XGBoost and LightGBM are increasingly used to evaluate the importance of different process features contributing to anomalies. These models support interpretable decisions, enabling operators to understand the source of failure or defect causes (Vozza et al., 2024).

1.4.5 Tools and frameworks

Several open-source libraries enable integration of anomaly detection in Python-based pipelines:

- PyOD (Zhao et al., 2019): A comprehensive toolkit of over 20 anomaly detection algorithms, including deep and ensemble-based models.
- PySAD (Alaei et al., 2021): A streaming anomaly detection framework supporting online detection from evolving time-series.
- Scikit-learn: Offers implementations of One-Class SVM and Isolation Forest for structured data.

1.5. Existing frameworks

The synchronization and monitoring of multi-stream data in industrial environments—particularly in Wire Arc Additive Manufacturing (WAAM)—requires robust software frameworks. Both commercial and academic initiatives have explored platforms capable of integrating heterogeneous data sources, performing real-time analysis, and enabling flexible architecture design. However, significant gaps remain in terms of accessibility, transparency, and customization, especially for research and development purposes.

1.5.1 Commercial Platforms

Several industrial Internet of Things (IIoT) platforms offer partial support for data synchronization and monitoring in manufacturing contexts. For instance:

- Siemens MindSphere is a cloud-based, closed-source platform that connects machines and physical infrastructure to the digital world. It offers analytics and visualization tools but provides limited access to raw data streams or synchronization logic for custom processing (Siemens, 2022).
- GE Predix similarly focuses on predictive maintenance and machine learning integration in manufacturing pipelines but is primarily oriented toward enterprise deployment with limited flexibility for low-level data control (GE, 2021).
- ABB Ability™ and FANUC FIELD system incorporate real-time diagnostics, robot analytics, and fault detection modules. However, these systems often operate as black boxes, with minimal opportunity for research customization or export of full sensor datasets for external analysis (ABB, 2021; FANUC, 2021).

Although these platforms are robust and production-ready, they fall short in academic and prototyping scenarios where transparent access to data streams and algorithmic control is crucial.

1.5.2 Academic and Open-Source Frameworks

Academic research has explored several open and modular solutions for data synchronization and sensor integration:

- Robot Operating System (ROS) is one of the most widely adopted open-source frameworks for robotic applications. It supports time-stamped sensor fusion, message passing, and modular node architecture, making it suitable for synchronizing robotic motion with sensor data streams in WAAM applications (Quigley et al., 2009).
- ROS 2, the real-time successor to ROS, improves support for deterministic communication and industrial use cases, enabling real-time operation and robust communication in noisy environments (Dudek et al., 2021).
- He et al. (2024) propose an open software architecture for multi-robot WAAM systems using Robot Raconteur middleware. Their system integrates sensors and actuators from heterogeneous manufacturers, enabling full WAAM process execution, from slicing and motion planning to geometric inspection and process control.

However, these solutions still require substantial integration effort. ROS-based systems, for example, offer tools for synchronization but not complete pipelines tailored for WAAM, especially regarding point cloud alignment, thermal signal analysis, and process-level feedback loops.

1.6. Application areas

The synchronization and data fusion methodologies discussed in the context of Wire Arc Additive Manufacturing (WAAM) are equally pertinent to various other domains that require the integration of multi-modal data streams. These techniques are instrumental in ensuring coherent and accurate data interpretation across different systems.

1.6.1 Autonomous vehicles

In autonomous driving systems, the fusion of data from LiDAR, cameras, and inertial measurement units (IMUs) is critical for accurate perception and navigation. Techniques such as tightly-coupled LiDAR-visual-inertial odometry have been developed to achieve real-time state estimation and mapping, enhancing the vehicle's ability to navigate complex environments (Shan et al., 2021). These methods rely on precise temporal and spatial synchronization of heterogeneous sensor data to function effectively.

1.6.2 Surgical robotics

In the realm of surgical robotics, integrating sensor data, tool motion, and medical imaging is vital for precision and safety. Advanced control systems that combine electromyography (EMG) signals with robotic actuators have been proposed to enhance human-robot interaction during rehabilitation and surgical procedures (Xie et al., 2024). Such systems necessitate the synchronization of physiological signals with robotic movements to ensure responsive and accurate assistance.

1.6.3 Industrial quality control

Industrial quality control processes often involve real-time inspection systems that synchronize sensor data with robotic paths. For instance, integrating sensor frameworks within robot-based WAAM cells allows for continuous monitoring and optimization of the manufacturing process, ensuring product quality and consistency (Hauser et al., 2021). These systems depend on the alignment of sensor inputs with robotic operations to detect anomalies and maintain standards.

1.6.4 Smart construction and building inspection

In smart construction and infrastructure inspection, unmanned aerial vehicles (UAVs) and ground robots equipped with LiDAR and visual SLAM technologies are employed to assess structural integrity. The fusion of LiDAR and wide-angle camera data enables comprehensive environment mapping, facilitating accurate inspections and maintenance planning (De Silva et al., 2017). Effective synchronization of these data streams is essential for creating reliable models of the built environment.

1.7. Summary and research gap

The review of current literature and industrial practice confirms that synchronization and anomaly detection in multi-stream data environments—especially within the context of Wire Arc Additive Manufacturing (WAAM)—remain critical yet underdeveloped research areas. While numerous studies have addressed isolated aspects of these challenges, several important limitations persist.

Identified gaps in existing work

- **Lack of method-switching flexibility**: Most synchronization systems rely on a fixed alignment strategy (e.g., timestamp-based or event-based) and do not support adaptive switching depending on data reliability, availability, or stream type. This rigidity hinders robust integration in noisy or asynchronous environments.
- **Limited extensibility and openness**: Many frameworks—especially those deployed in industrial settings—are closed-source or tightly coupled with proprietary software. This restricts algorithm-level customization, hindering their usefulness in academic or experimental workflows.
- **Absence of layer-wise anomaly visualization**: Although point cloud data is frequently used for geometric verification, few systems provide synchronized, layer-level visualization that overlays scanner geometry with Arc On/Off markers and detected anomaly zones. This limits both interpretability and manual inspection.
- **Insufficient modularity and decoupling**: Existing implementations rarely adopt modular software design. Components such as data preprocessing, synchronization, clustering, anomaly detection, and visualization are often interdependent, making it difficult to isolate and test algorithms independently or reuse them across domains.

Research contribution of this thesis

This thesis directly addresses these shortcomings by developing a modular, extensible framework tailored to WAAM data processing. The contributions include:

- **Hybrid synchronization support**: Geometry-based and time-based alignment strategies are implemented side by side, allowing dynamic selection based on stream characteristics.
- **Automated clustering of deposition layers**: Multiple algorithms are provided (e.g., KMeans, DBSCAN, KDE+Peak) for extracting discrete layers from raw 3D scan data.

- **Rule-based anomaly detection module**: Using geometric thresholds, sensor deviations, and event-state mismatches, the framework flags suspect regions for inspection and logging.
- **Interactive 3D visualization**: A Plotly- or Dash-based interface enables exploration of each WAAM layer in spatial context, with overlay support for Arc events and anomalies.
- **Reusability across domains**: Though the framework is designed for WAAM, its modularity, file-based architecture, and support for multi-stream time series data make it broadly applicable to other cyber-physical systems involving sensor fusion, robotic monitoring, and additive manufacturing.

2. METHODOLOGY

This chapter presents the methodology used to develop and evaluate synchronization and anomaly detection techniques for multi-stream data in Wire Arc Additive Manufacturing (WAAM). It includes a description of the datasets, preprocessing steps, synchronization algorithms, clustering techniques for layer detection, anomaly identification procedures, and evaluation metrics.

2.1. Description of data sources

The experimental data used in this research was collected in a WAAM laboratory equipped with:

- an industrial robot (Yaskawa),
- a welding power source (Fronius),
- and a laser scanner (Wenglor).

This setup is shown in Figure 1.

Figure 1 Equipment used in the WAAM laboratory: (left) industrial robot (Yaskawa), (center) welding power source (Fronius), (right) laser scanner (Wenglor).

Data from these devices is grouped into three main categories:

Initial Data

Initial data includes all parameters configured before the production process starts. These consist of:

- **CAD model**: Designed in 3D modeling software and exported in standard CAD format.
- **Path planning data**: Generated from the CAD model using a slicing software, which produces a list of coordinate points for robotic motion.
- **JBI control files**: The sliced coordinates are converted into .JBI files (Job Batch Interface) containing executable robot commands. Each JBI includes:
    - motion paths (X, Y, Z),
    - tool orientation (Rx, Ry, Rz),
    - welding activation commands (Arc On, Arc Off),
    - job numbers and travel speed (TS), target speed of robot's movement in mm/sec.

A sample of these JBI commands is presented in Figure 2.

Example commands within a .JBI file:

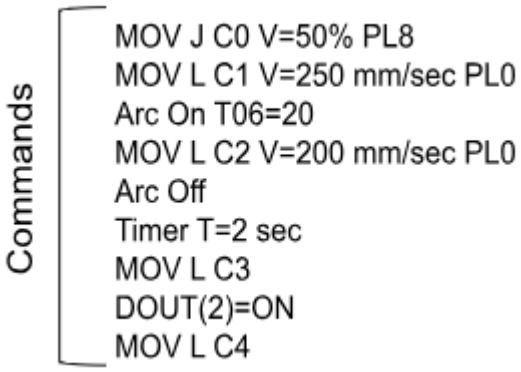


Figure 2 Example of robot motion and welding commands in a .JBI file used in WAAM.

The two common postprocessing workflows are illustrated in Figure 3.

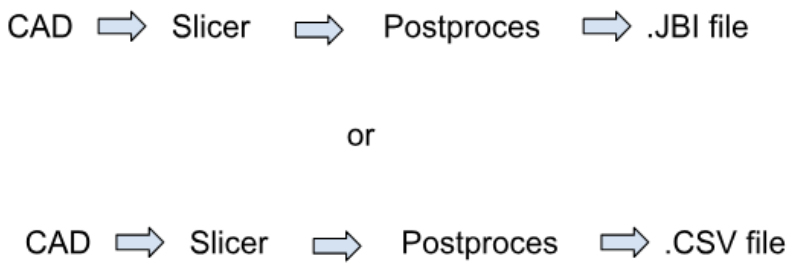


Figure 3 Workflow of CAD-to-robot file generation: through .JBI or .CSV output depending on the postprocessor.

Process Data

These are real-time parameters recorded during the deposition process:

- **DateTime**: timestamps that record the moment data is received from each device.
- **Robot movement data**: (X, Y, Z, Rx, Ry, Rz) coordinates logged continuously.
- **Travel Speed (TS)**: mm/min.
- **Welding parameters**:
  - Wire Feed Speed (WFS),
  - Welding Current and Voltage,
  - Arc stability indicators,
  - Process Active / Arc On (IOArcOn),
  - Warnings.
- **Laser scanner data**:
  - (X, Z) coordinates of scan points,
  - Intensity of the reflected signal,
  - Timestamps for each scan line.

These values are exported via a C++ script and saved in CSV format for each stream.

Final Data

At the end of the print job, final output files include:

- **Process parameter log** from the Fronius power source,
- **Robot path logs** from the Yaskawa controller,
- **Layer-wise scan results** from the Wenglor laser sensor.

This combination of control, process, and geometric data forms the basis for synchronization and quality analysis. It enables full-cycle evaluation — from planning through execution to as-built geometry.

2.2. Data preprocessing and cleaning

Preprocessing is a critical step to ensure data integrity and to prepare the scanned point cloud for accurate layer detection and synchronization with process data. The cleaning procedure consists of several sequential operations. These steps are illustrated in Figures 4–7.

- **Substrate Removal**: The first scan typically captures the build plate (substrate) rather than a deposition layer. It is identified by locating the points with the lowest Z-values and is excluded from further analysis.
- **Point Cloud Flattening**: To minimize geometric distortions caused by tilt or unevenness in the scanning process, the Z-coordinates are detrended using polynomial regression with respect to the X-axis. This flattening step enhances the visibility of layer boundaries.
- **Vertical Clustering by Height**: A Kernel Density Estimation (KDE) is applied to the Z-coordinate distribution to identify peaks in scan density, which are indicative of deposition layers. These peaks are used either to guide clustering (e.g., with KMeans) or for direct segmentation.
- **Outlier Removal**: The Local Outlier Factor (LOF) algorithm is used to detect and remove spatially sparse or anomalous points that deviate significantly from their local neighborhood density.
- **Bounding Box Filtering**: For each detected cluster or layer, the spatial extent in the X and Y dimensions is analyzed. Clusters with abnormally small bounding boxes are considered artifacts—often caused by noise or partial scans—and are filtered out.

These preprocessing steps result in a clean and structured point cloud, which serves as the foundation for reliable layer-wise matching, synchronization, and subsequent anomaly detection.

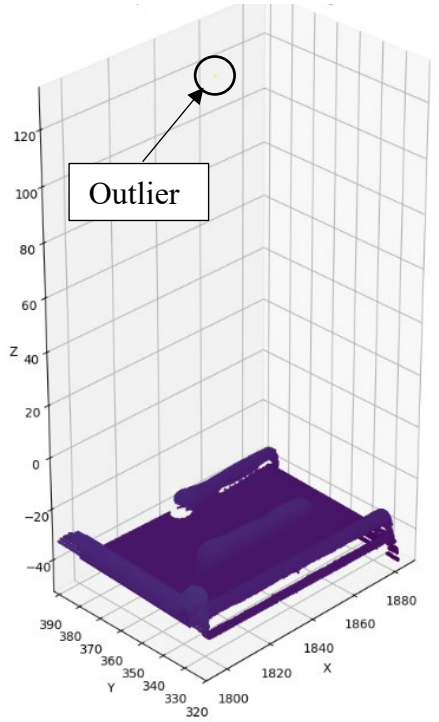


Figure 4 Step 1: Raw data (no filtering)

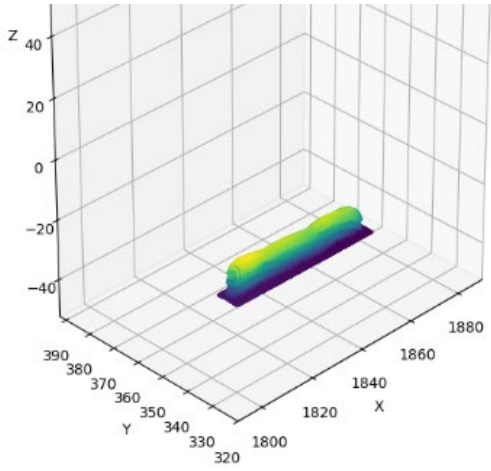


Figure 5 Step 2: Z Outlier removal + ROI X/Y

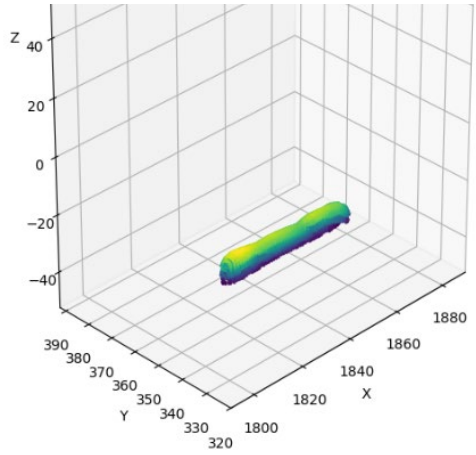


Figure 6 Step 3: ROI X/Y/Z and substrate removal

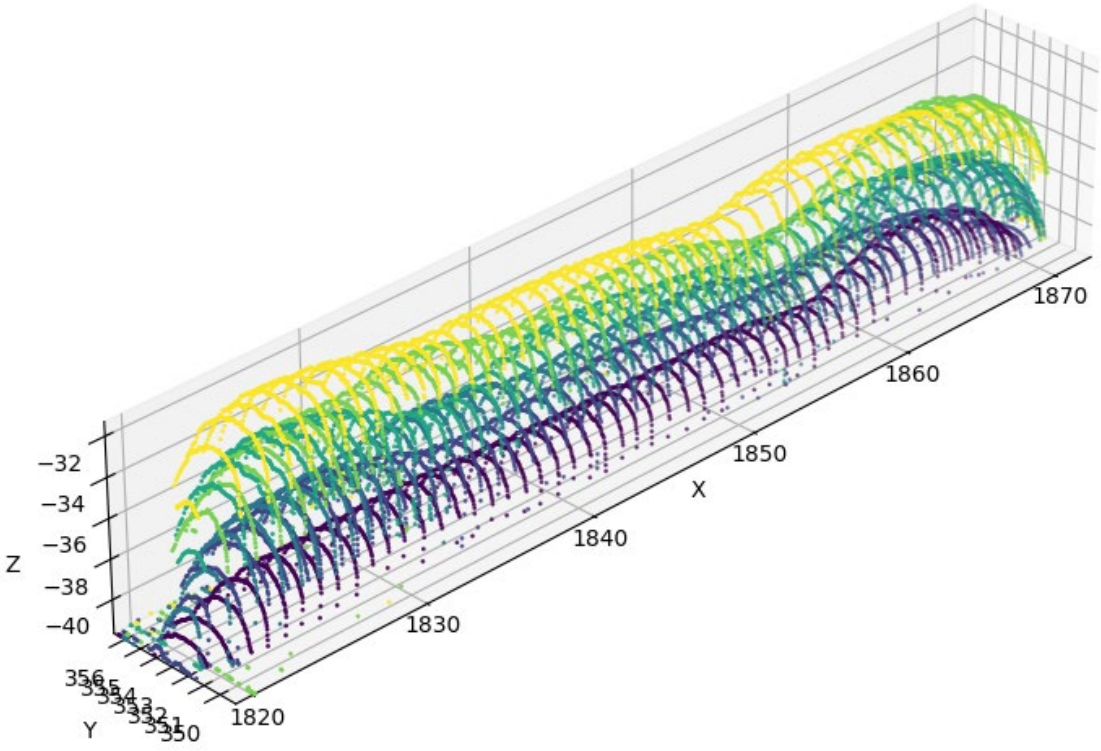


Figure 7 Step 3: close up view

2.3. Synchronization strategy and reference method

In order to evaluate and compare different approaches for layer detection in WAAM scan data, a consistent and structured reference method is required. This section introduces the 1MethodGroundTruth, which is used as the synchronization baseline for benchmarking the results of clustering and segmentation algorithms.

Unlike data-driven or geometric methods, this approach relies solely on the structural order of scanner metadata files, assuming that each scan corresponds to a single deposition layer and that the scans are ordered sequentially. It does not depend on Arc On/Off signals, timestamps, or process metadata.

The 1MethodGroundTruth method assigns a unique layer identifier to every scan segment and produces a labeled point cloud dataset. This dataset is then used in Section 2.4 to evaluate the performance of automated methods by comparing their results to the ground truth using metrics such as accuracy, Intersection over Union (IoU), precision, and recall.

2.3.1. Method 1 - Ground Truth

This method defines the reference segmentation of deposition layers based solely on the order and structure of scanner outputs, without using any process logs or temporal data such as Arc On/Off signals.

The approach relies on two input files:

- A single point cloud file (points_df) containing all scanned points from the build.
- A sequence of structured laser scan metadata files (LaserCoordinate1.csv to LaserCoordinate7.csv), where each row contains coordinate arrays (X, Y, Z) in text format representing individual scan lines.

The segmentation logic is based on structured file order and parsing of scan arrays. A flowchart of this logic is shown in Figure 8.

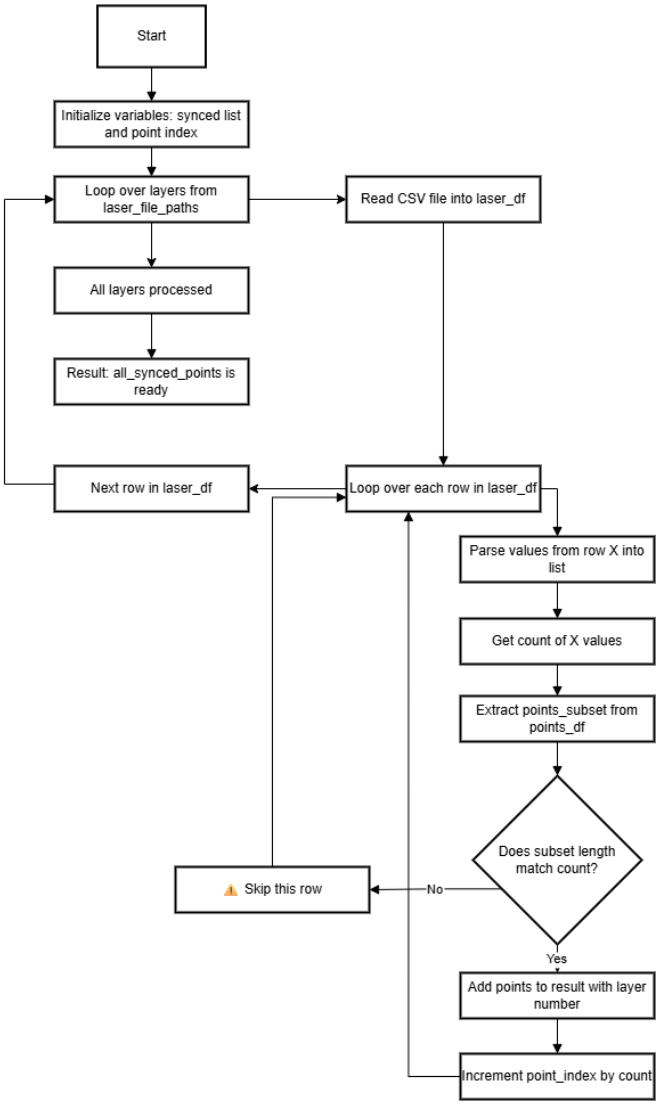


Figure 8 Segmentation logic for 1MethodGroundTruth using ordered scanner metadata.

The segmentation logic is as follows:

1. **Sequential Point Assignment**

   For each LaserCoordinateX.csv file, the number of points in each scan line is determined by parsing the length of the X-array (same length applies to Y and Z). Using this count, a slice of consecutive points is extracted from the global points_df dataset and assigned to the corresponding scan layer.

2. **Layer Labeling**

   Each group of points extracted for a given scan is labeled with a numeric layer_id_1MethodGroundTruth, where the index reflects the scan sequence. For example, the first scan corresponds to the substrate and is labeled as layer 0, while subsequent scans receive increasing layer indices (1, 2, …).

3. **Assumptions**

   - The scanner captures each layer immediately after deposition and stores the scans in strict order.
   - The number of points in each scan segment is consistent with the point cloud structure (no missing or duplicate entries).
   - No temporal alignment is required because the physical sequence of layers is inferred directly from file structure.

4. **Post-processing**

   After segmentation, the dataset is further cleaned through outlier filtering (e.g., removing the top/bottom 5% in Z), and Region of Interest (ROI) filtering along X, Y, and Z to retain only the central working zone of the WAAM part.

This method produces a labeled point cloud with high confidence in layer structure and is used as the ground truth reference for evaluating the performance of other clustering and synchronization strategies. The result of this segmentation is summarized in Table 3.

Table 3 Ground truth layer assignment extracted from ordered scan files.

Ground Truth Layer Assignment				
Scan File	Scan Layer ID	Points Start Index	Points End Index	Num Points
LaserCoordinate1.csv	0	0	1244	1245
LaserCoordinate2.csv	1	1245	2679	1435
LaserCoordinate3.csv	2	2680	4077	1398
LaserCoordinate4.csv	3	4078	5461	1384
LaserCoordinate5.csv	4	5462	6872	1411
LaserCoordinate6.csv	5	6873	8319	1447
LaserCoordinate7.csv	6	8320	9700	1381

2.4. Methods for layer detection and clustering

This section presents six automatic methods developed to segment WAAM scanner point cloud data into individual deposition layers. Each method proposes a different strategy for detecting vertical structure in the data, based on clustering algorithms, geometric patterns, or hierarchical relationships.

Unlike the 1MethodGroundTruth, which serves as a predefined reference based on scan order, the following techniques operate independently and do not rely on structured scanner metadata. Their outputs are compared to the ground truth using quantitative evaluation metrics.

These methods vary in terms of algorithm type (e.g., KMeans, DBSCAN, hierarchical), degree of automation, sensitivity to outliers, and assumptions about layer spacing and uniformity. They are evaluated for their ability to reproduce the ground truth segmentation and to provide robust layer detection under noisy or irregular scanning conditions.

To support practical selection of an appropriate segmentation method, a decision tree is provided in Figure 9. It guides the user through key questions related to data availability, prior knowledge of the number of layers, and scan noise characteristics. This visual aid helps in choosing the most suitable technique under different WAAM scanning scenarios.

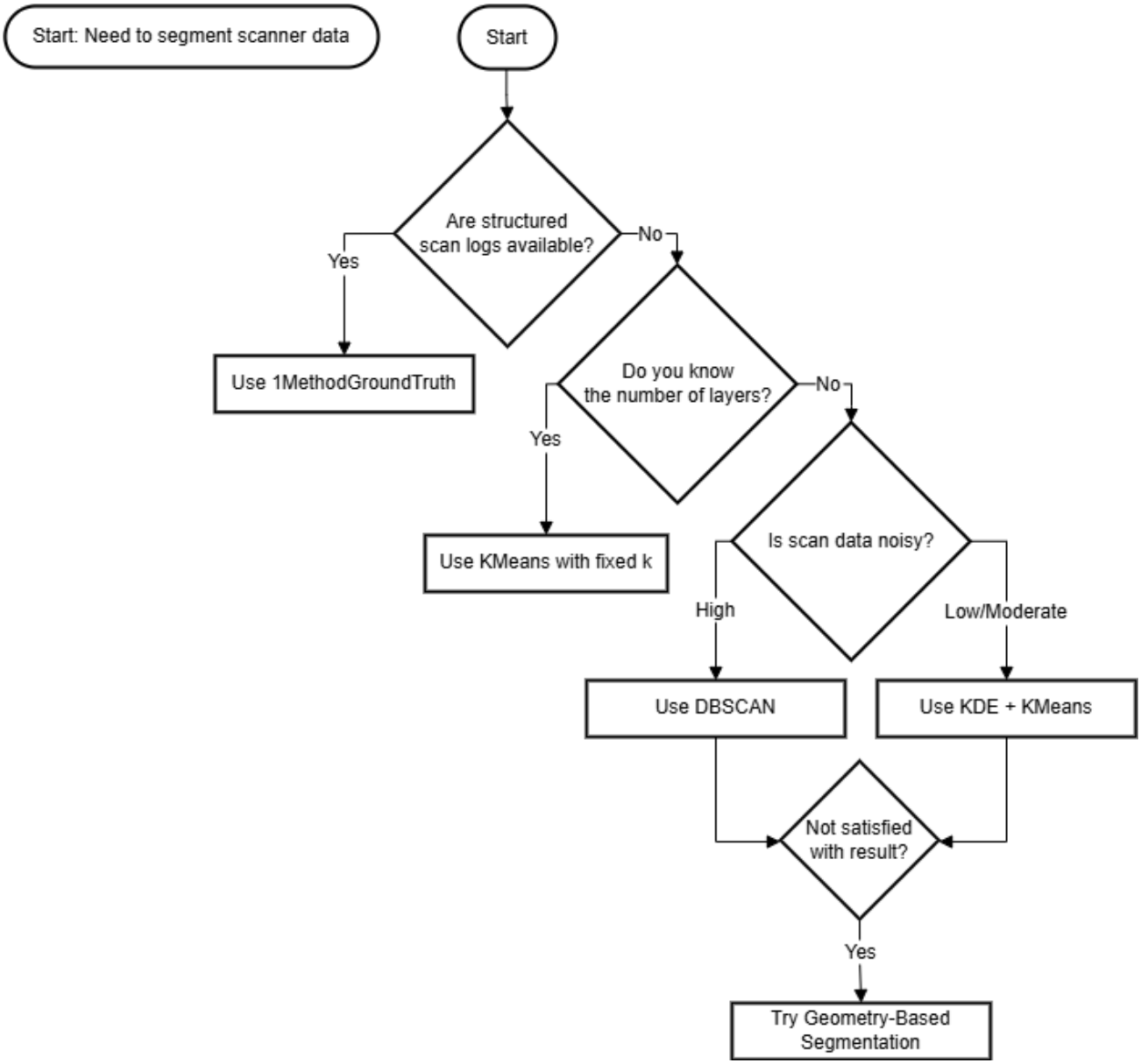


Figure 9 Decision tree for selecting a suitable layer segmentation method based on scan data conditions

2.4.1. Method 2 - Cluster KMeans Known: KMeans with Fixed Cluster Number

In this method, the number of expected deposition layers is manually set (e.g., 6), and KMeans clustering is applied to the Z-coordinate of the point cloud. Cluster centers are sorted from bottom to top to ensure correct layer ordering. This method assumes prior knowledge about the number of layers, which can often be obtained from the G-code or build plan.

The rationale behind this method lies in the vertical nature of WAAM deposition, where each new layer corresponds to a distinct height band in the point cloud. If the number of layers is known, the dataset can be segmented effectively by directly assigning points to *k* vertical clusters. KMeans minimizes intra-cluster variance and provides a fast and reproducible segmentation technique, making it ideal for structured or well-behaved datasets (Lloyd, 1982; Jain, 2010).

This method is particularly useful in the following scenarios:

- When the exact number of layers is known from planning or prior builds.
- When structured scanner metadata is missing or incomplete.
- When a quick and computationally lightweight segmentation is needed as a reference or control condition.

It serves in this study as a baseline method for evaluating the performance of automatic or unsupervised approaches. By comparing results from this fixed-*k* segmentation to those generated by other clustering strategies, we can understand how the availability (or lack) of prior knowledge impacts segmentation accuracy and robustness (Xu and Tian, 2015).

A code example for implementing this method using scikit-learn is presented in Figure 10.

A step-by-step overview of this clustering process is shown in Figure 11.

Implementation Notes:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=6)
df['layer_id'] = kmeans.fit_predict(df[['Z']])
```

Figure 10 Example code for applying KMeans clustering with fixed number of layers.

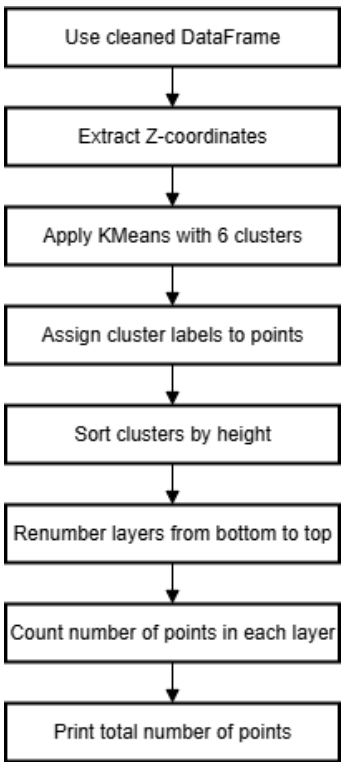


Figure 11Step-by-step workflow of KMeans clustering with fixed cluster count.

2.4.2. Method 3 – Cluster KMeans Auto: Automatic KMeans via KDE + Peaks

This method eliminates the need to manually specify the number of deposition layers by automatically estimating it from the data. It first applies Kernel Density Estimation (KDE) to the distribution of Z-values and then uses a peak detection algorithm (scipy.signal.find_peaks) to identify the number of distinct height modes in the scan. The number of peaks is passed to the KMeans algorithm as the number of clusters.

The motivation for this approach stems from the fact that, in many practical cases, the number of deposition layers is not known a priori or may vary between builds. Automatically determining the number of clusters allows for a more generalizable method applicable to unlabelled or historical scan data. KDE smooths the Z-value histogram and highlights dominant height bands, which tend to correspond to real WAAM layers (Silverman, 1986; Comaniciu and Meer, 2002).

This method is particularly suitable:

- When the number of layers is unknown;
- When layers are relatively uniform in height and spacing;
- For preliminary segmentation before manual inspection or refinement.

However, the method's accuracy depends on the correct tuning of the KDE bandwidth and the prominence threshold used in peak detection. If the scan data is noisy or contains uneven deposition, it may result in under- or over-segmentation (Xu and Tian, 2015).

A minimal Python implementation of the method is shown in Figure 12. A process overview diagram is presented in Figure 13.

Implementation Notes:

```
from sklearn.neighbors import KernelDensity
from scipy.signal import find_peaks
```

Figure 12 Python code snippet for layer estimation using KDE and clustering with KMeans.

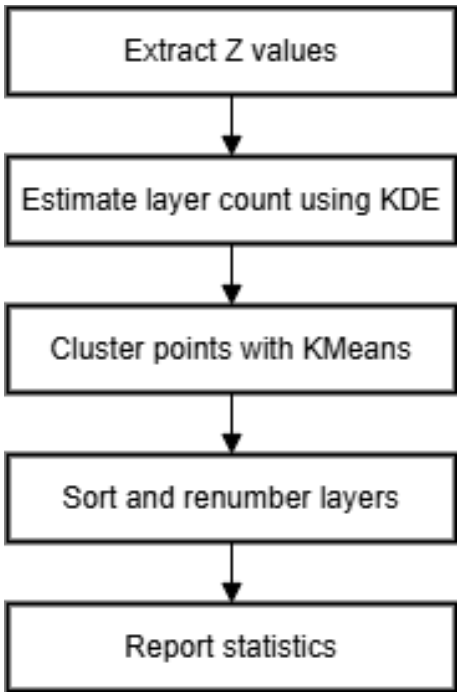


Figure 13 Key Steps: Auto Layer Detection via KDE + KMeans

2.4.3. Method 4 – Cluster DBSCAN Auto: Density-Based Clustering

This method applies DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to the Z-coordinate of the point cloud. It uses two parameters: eps (the neighborhood radius) and min_samples (the minimum number of points to form a cluster). Unlike KMeans, DBSCAN does not require the number of clusters to be specified in advance.

DBSCAN is particularly suited for WAAM data that contains variable layer thickness, irregular spacing, or scanning noise. Because it relies on density rather than geometric assumptions, it can effectively identify clusters even when layers are not clearly separated or evenly distributed. Furthermore, it automatically marks sparse or isolated points as noise (label -1), helping to clean the data during the segmentation step (Ester et al., 1996).

This method is advantageous when:

- The number of layers is not known a priori;
- The scanned geometry contains artifacts, missing segments, or noise;
- Outliers and discontinuities are expected.

However, its performance is sensitive to the choice of eps and min_samples, which often require tuning based on data density and scan resolution. In particular, if layers are close together, DBSCAN may merge them into a single cluster. If eps is too small, legitimate points may be excluded from clusters. Therefore, it works best after careful parameter exploration or visual feedback.

A visual summary of the DBSCAN-based workflow for layer segmentation is provided in Figure 14.

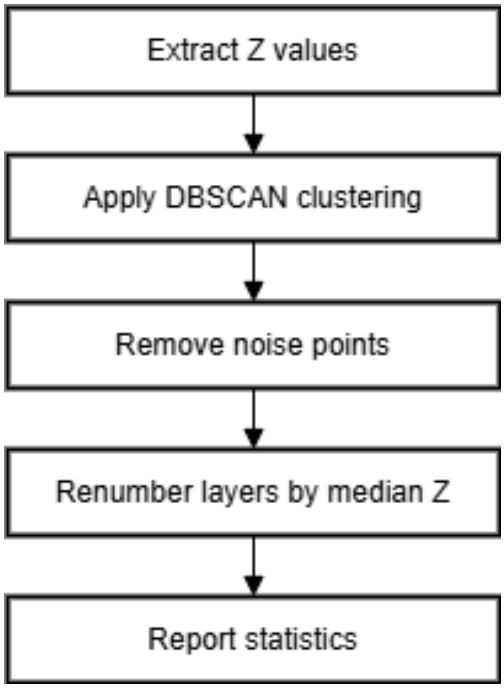


Figure 14 DBSCAN Clustering Method for Layer Identification

2.4.4. Method 5 – Geometry Based Scan Segmentation by Distance Jumps

This method performs fully automated segmentation of the scanner point cloud by detecting large Euclidean distance jumps between consecutive points along each scan line. The full segmentation pipeline is illustrated in Figure 15. These distance discontinuities are interpreted as boundaries between printed segments or transitions between layers.

The core idea is that during WAAM deposition, the robotic system moves smoothly within a segment, while transitions between segments (such as pauses, retracts, or arc re-ignition) introduce spatial discontinuities in the scanned data. By identifying these discontinuities, the method infers segment boundaries without relying on timestamps or Arc On/Off markers.

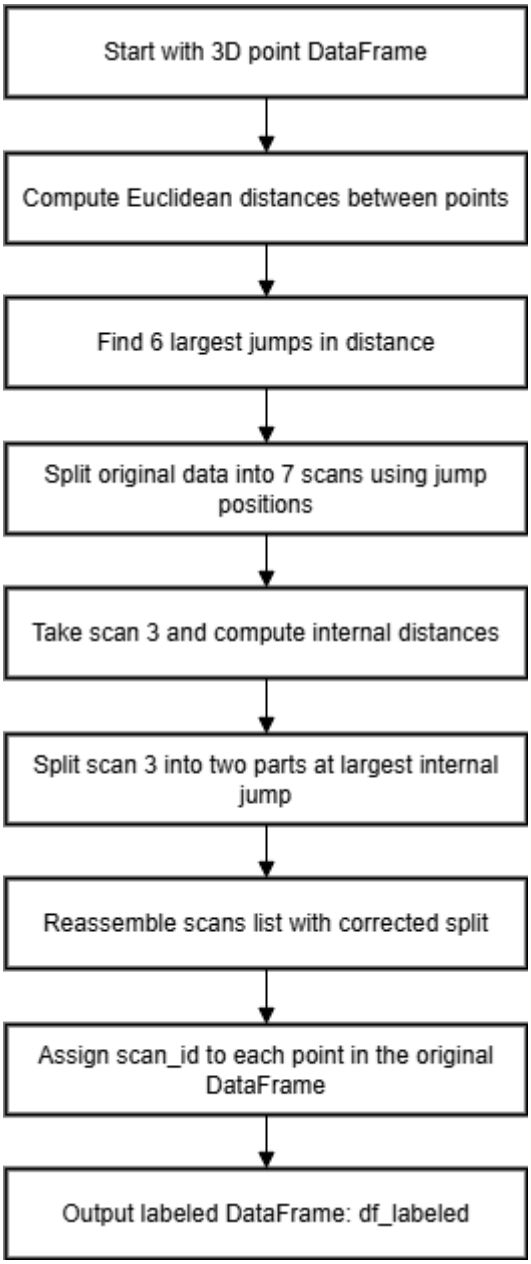


Figure 15 Workflow: identifying scan boundaries via Euclidean Distance Peaks

Mathematical formulation:

Let $P_i = (x_i, y_i, z_i)$ and $P_{i+1} = (x_{i+1}, y_{i+1}, z_{i+1})$ be two consecutive 3D points from the scan. The Euclidean distance between them is computed as:

$$\boldsymbol{d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}} \qquad (1.)$$

A new segment boundary is defined whenever the distance exceeds a predefined precision threshold δ\deltaδ:

If $d_i > \delta$, then a new segment starts at $i + 1$

All points between two consecutive jump indices are grouped into one segment and assigned a unique segment or layer identifier.

In the current implementation, the threshold δ was empirically set to 1.0 mm, based on the average inter-point distance within continuous scans.

This approach has several advantages that make it particularly suitable for WAAM scenarios with missing or unreliable process metadata. First, it does not rely on timestamps, process logs, or control commands, making it robust in cases where only geometric scanner data is available. Second, it is inherently resistant to temporal inconsistencies, since it derives segmentation solely from spatial continuity. Third, it is effective when the scan geometry contains clearly defined vertical discontinuities, such as those introduced by deposition interruptions (Rusu and Cousins, 2011).

However, the method is sensitive to point density and scan noise. An improperly chosen threshold δ may result in over-segmentation if set too low, or missed transitions if too high. Its performance depends on scan consistency and may require tuning for different datasets or scanning resolutions.

This method is especially appropriate in situations where geometric scans are available and implies that all points are time-sorted. The resulting segmentation can then be used independently or compared to reference-based or process-aligned methods (Zhang et al., 2020).

2.4.5. Method 6 – Hierarchical Auto: Hierarchical Clustering

This method applies hierarchical clustering to the Z-values of the scanned point cloud to detect layer structures without requiring prior knowledge of the number of clusters. In this implementation, Ward's linkage method is used to construct a hierarchical tree (dendrogram), which is then cut at a predefined distance threshold to form discrete vertical segments. The full clustering workflow is illustrated in Figure 16.

Hierarchical clustering is especially useful when the number of deposition layers is unknown, but the geometry suggests a multi-level structure. Since it operates by recursively merging clusters based on a distance criterion, it naturally captures nested patterns that may reflect sub-layers, remelting, or overlapping paths in the WAAM process (Müllner, 2011).

The main advantage of this method is its flexibility and automation: the user does not need to define the number of clusters in advance, as the clustering tree can be cut at different levels depending on the dataset. Additionally, the method allows post-analysis of the dendrogram to evaluate structural hierarchy between layers.

However, the accuracy of the segmentation is highly dependent on the cutting threshold. If the threshold is too low, the result may include noise or over-segmentation. If it is too high, adjacent layers may be merged into a single cluster. Another drawback is that hierarchical

clustering is computationally intensive, especially on large point clouds, and may not be well suited for real-time applications (Jain et al., 1999).

Overall, hierarchical clustering serves as a fully automated, metadata-independent method that balances flexibility and structure. In this thesis, it is evaluated alongside other clustering and geometry-based methods to assess its robustness and alignment with the scan-based ground truth.

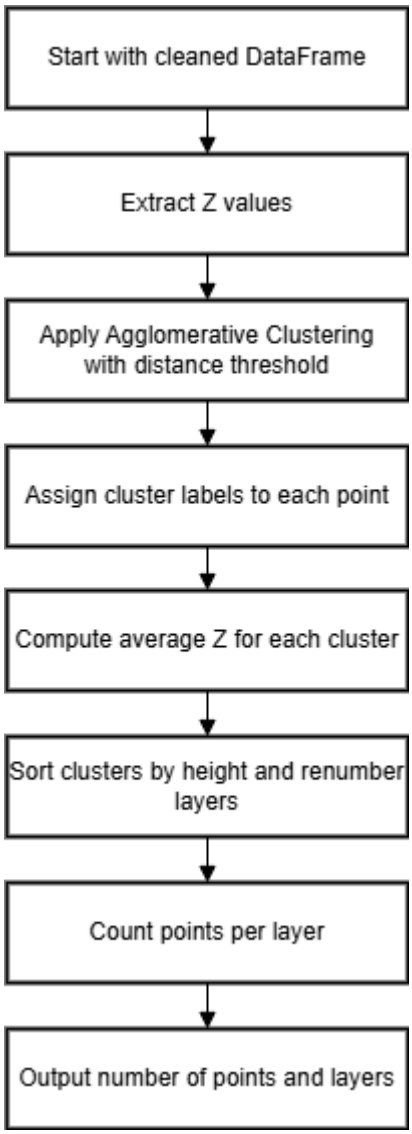


Figure 16 Method 6: Hierarchical Clustering (Automatic)

To provide a comparative overview of the five automated layer detection methods implemented in this study, Table 4 summarizes their key characteristics, including algorithm type, level of automation, reliance on process metadata, robustness to noise, and sensitivity to parameter tuning. This overview supports the selection of an appropriate method based on data availability and quality.

In addition, a schematic summary of all six segmentation approaches—five automated and one reference-based-is presented in Figure 17. The diagram illustrates the input, processing logic,

and output of each method, highlighting the differences in assumptions and structure. This visual summary helps to contextualize the methods before their quantitative evaluation in Chapter 4.

Table 4 Summary of automated layer detection methods used in this study.

Summary Table of Methods						
Method	Type	Automatic	Uses Process Logs	Needs n_layers	Robust to Noise	Sensitive to Parameters
2MethodClusterKMeansKnown	Clustering	No	No	Yes	Low	Yes
3MethodClusterKMeansAuto	Clustering	Yes	No	No	Medium	Yes
4MethodClusterDBSCANAuto	Clustering	Yes	No	No	High	Yes
5MethodGeometryBased	Geometric Segmentation	Yes	No	No	Medium	Yes
6MethodHierarchicalAuto	Clustering	Yes	No	No	Medium	Yes

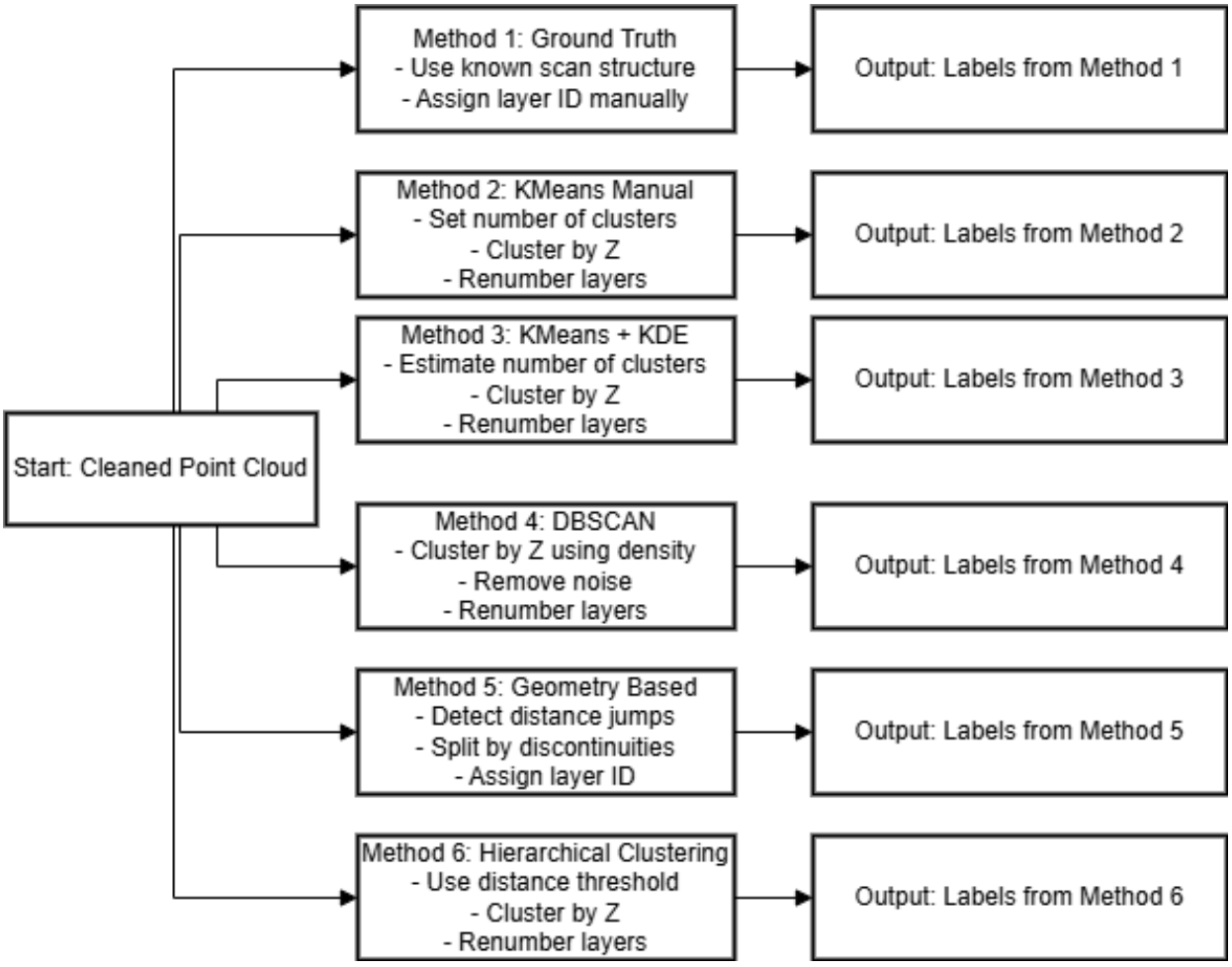


Figure 17 Six Methods for Layer Identification

2.5. Anomaly detection approach

Once data is synchronized, the framework detects anomalies in the process using several heuristics.

Note: In this context, “arc” refers to geometric segments within the scan or toolpath, not to the electric welding arc.

- **Arc absence**: Red scan segments without corresponding Arc On events indicate missing process records or scanner noise.
- **Arc duration anomalies**: Segments that are significantly shorter or longer than expected are flagged based on statistical thresholds.
- **Uncovered regions**: Areas within a toolpath that lack scanned points or contain discontinuities are marked as gaps.
- **Geometric jumps**: Large displacements between scan segments or toolpath transitions may indicate anomalies or robot drift.
- **Duplicated arcs**: Overlapping Arc On segments without new deposition can indicate repeated commands or sensor delays.

All detected anomalies are labeled in the final dataframe and visualized using red connecting lines in 3D plots.

2.6. Metrics for evaluation

To evaluate the effectiveness of synchronization and anomaly detection methods, a set of established performance metrics is applied. These metrics are selected to assess both the geometric accuracy of layer segmentation and the classification performance of anomaly detection. The following evaluation criteria are used:

- **Accuracy**: Measures the proportion of scan points that were correctly assigned to their corresponding layer, based on reference labels from the Ground Truth method. This metric is useful for direct point-wise comparison between predicted and actual labels. Accuracy is computed using only the scan points retained after outlier filtering, based on the layer assignments from the Ground Truth segmentation.
- **Intersection over Union (IoU)**: Calculates the spatial overlap between predicted and ground truth segments by dividing the size of their intersection by the size of their union. IoU is particularly relevant in 3D segmentation tasks, where it reflects how well the predicted layers match the shape and position of reference segments.
- **Precision and Recall**: These metrics are primarily applied to anomaly detection. Precision quantifies the proportion of points predicted as anomalies that are actually anomalous, while recall measures the proportion of true anomalies that were successfully identified by the system. The ground truth for anomalies is derived from rule-based labeling in the 1MethodGroundTruth dataset, which serves as a reference based on deterministic geometric rules.

- **F1-score (macro)**: Combines precision and recall into a single metric using their harmonic mean. Macro-averaging is used to ensure that all classes (i.e., layers or anomaly types) are treated equally, regardless of their frequency in the dataset.

Each metric is computed per layer or per anomaly class, and the results are averaged across the entire dataset to allow consistent comparison across methods. The 1MethodGroundTruth segmentation is used as a reference benchmark to evaluate the performance of all other synchronization and clustering strategies.

These metrics provide complementary insights into both segmentation quality and detection reliability, and are commonly used in machine learning evaluations for 3D point cloud processing (Rezatofighi et al., 2019).

3. FRAMEWORK ARCHITECTURE

This chapter presents the architectural design of the developed framework for synchronizing multi-stream WAAM data and detecting process anomalies. The system is modular and built for flexibility, allowing comparative testing of synchronization strategies and visualization of results. It is implemented in Python and supports both offline analysis and interactive visual feedback.

3.1. General design overview

The framework follows a step-by-step data processing pipeline that enables transformation of raw multi-source data into a structured and enriched point cloud with process metadata. The following key stages are executed:

1. **Scan Data Segmentation**
   A layer segmentation method is selected from those described in Section 2.4. The preferred method (e.g., KDE+KMeans) is applied to the scanner point cloud to assign each point to a deposition layer.
2. **Reference Comparison**
   Each segmentation result is compared against the 1MethodGroundTruth reference using accuracy and Intersection over Union (IoU) to assess its reliability. The method with the best alignment is selected for further synchronization.
3. **Process Data Integration**
   Process logs containing Arc On/Off, Z-position, toolpath segments, and job IDs are integrated into the point cloud. Each point is annotated with corresponding metadata by aligning geometry or distance-based intervals.
4. **Anomaly Detection**
   Logical rules and geometric checks are applied to the enriched dataset to detect inconsistencies such as missing Arc segments, layer gaps, duplicated paths, or misalignments.
5. **Output Generation**
   Final synchronized and labeled datasets are exported for visualization and further evaluation.
   A diagram illustrating this data flow is shown in Figure 18.

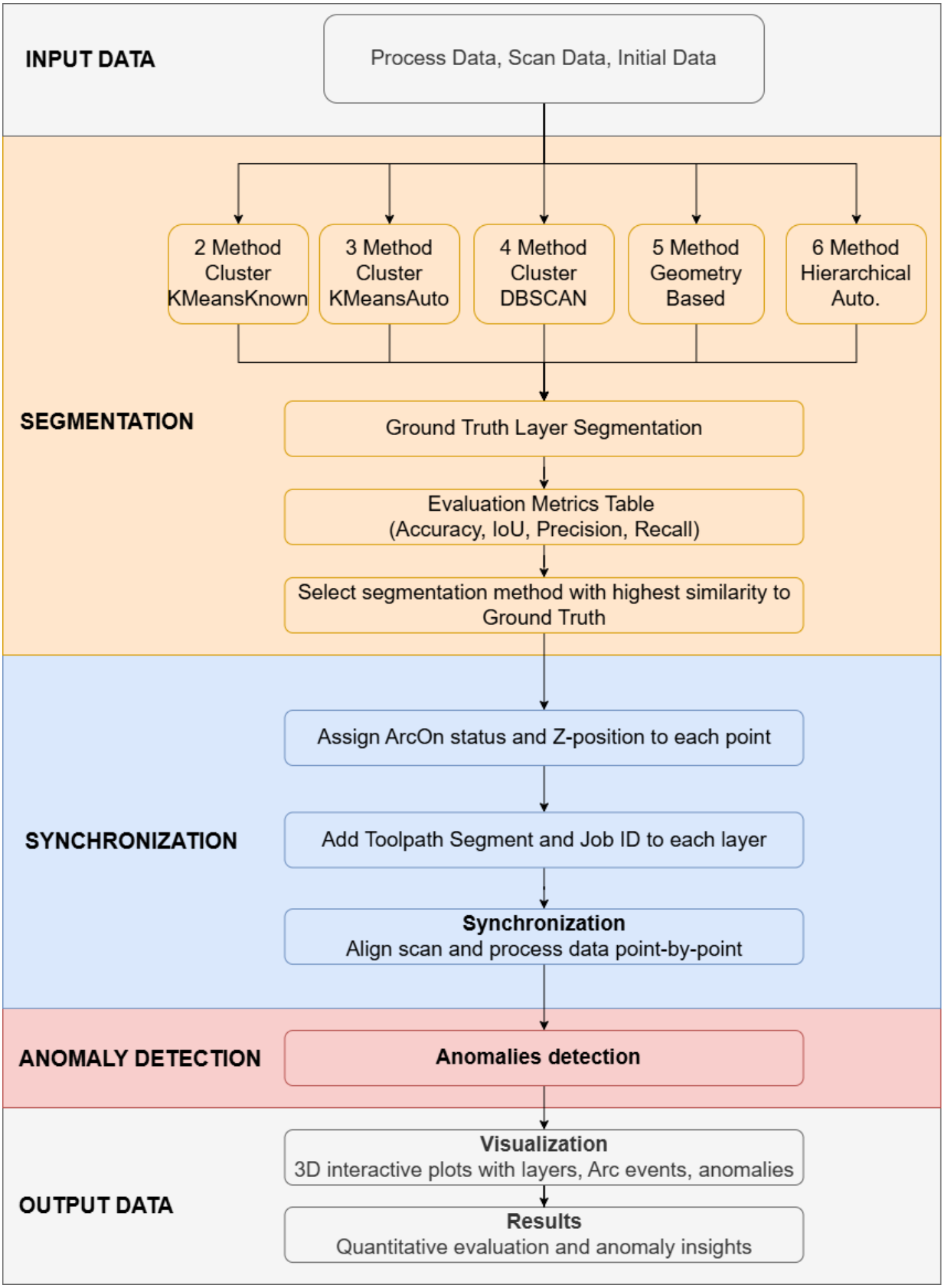


Figure 18 General architecture of the WAAM synchronization and anomaly detection framework.

This modular architecture allows flexibility in selecting segmentation methods and integrating additional metadata or validation rules, making the framework extensible and applicable to a wide range of WAAM setups.

3.2. Synchronization engine

The synchronization engine is the core component of the framework responsible for merging heterogeneous data streams into a unified, structured dataset. It transforms segmented scanner data, process logs, and machine control parameters into a point-level representation suitable for visualization, validation, and anomaly analysis.

The engine accepts three types of input:

- **Segmented scanner data**: 3D point cloud with layer identifiers assigned by one of the segmentation methods described in Section 2.4.
- **Process log data**: Arc On/Off signals, tool positions (X, Y, Z), and real-time measurements.
- **Initial data**: Job identifiers and travel speed (TS) values from G-code or control system outputs.

Figure 19 illustrates an example workflow for synchronizing process log data with scanner layers using clustering-based segmentation.

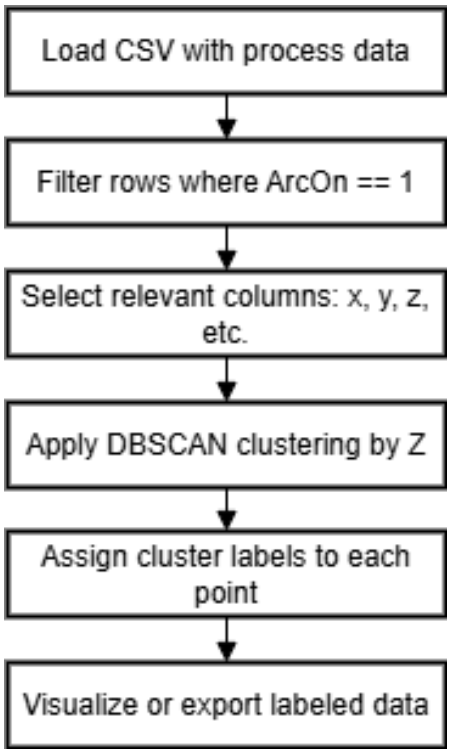


Figure 19 Process Log → Layers: DBSCAN Workflow

The synchronization process involves the following steps:

1. **ArcOn matching**

   Each point in the segmented point cloud is assigned a binary ArcOn label, indicating whether the welding arc was active at that position. This is done either by spatial alignment with the toolpath intervals marked as Arc On, or by time-based association where timestamps are available.

2. **Travel Speed assignment**
   The corresponding TS value (travel speed in mm/s) is interpolated from the machine control data and added to each scan point based on spatial proximity or index mapping.
3. **Job ID propagation**
   Each scan point is labeled with a Job ID that indicates the source program or operational block from which it originated.
4. **Layer-wise enrichment**
   Additional metadata (e.g., Arc length, TS statistics) are computed per layer and appended to support anomaly detection.

The engine supports three synchronization modes:

- **1MethodGroundTruth**: Segment alignment based on structured laser scan metadata.
- **5MethodGeometryBased**: Geometric segmentation via Euclidean distance jumps.
- **Clustering-based segmentation**: Synchronization using automatically detected layers from methods such as KMeans, DBSCAN, or Hierarchical clustering.

The final output is a unified pandas DataFrame in which each scan point is enriched with both geometric and process-level attributes. Table 5 presents the typical structure of this enriched dataset.

Table 5 Structure of the enriched DataFrame produced by the synchronization engine

Column	Description
X, Y, Z	3D coordinates of the scan point
layer_id_*	Layer label from selected segmentation method
ArcOn	Arc status at that point (1 or 0)
JobID	Job number from control metadata
TS	Travel speed (mm/s)
anomaly_flag	Detected anomaly at point level (if any)

This enriched dataset is used in the subsequent visualization and anomaly detection modules and serves as the backbone of the framework's data fusion process.

3.3. Anomaly detection module

The anomaly detection module is applied to the fully synchronized and structured dataset produced by the segmentation pipeline. Its goal is to identify deviations from expected layer geometry and process behavior, which may indicate deposition errors, synchronization mismatches, or control instabilities.

Note on terminology

In this context, the term “arc” does not refer to the electric welding arc. Instead, it denotes a geometric segment within a layer, formed by slicing each deposition layer into narrow vertical bands along the X-axis. These segments typically represent 1.0 mm wide cross-sections of the layer and are used for localized geometric analysis.

Layer and arc segmentation

After synchronization, each point in the 3D scan is labeled with a global layer_id and a local arc_id.

- Layers correspond to complete deposition steps and are typically segmented using one of the methods described earlier (e.g., geometry-based or clustering-based segmentation).
- Arcs are subsegments within each layer, obtained by dividing the layer geometry along the X-axis into evenly spaced vertical slices. Each arc includes all points whose X-coordinates fall within its local interval.

This arc-level subdivision enables fine-grained inspection of deposition quality and supports localized anomaly detection.

Geometric feature extraction

For every arc, the following geometric features are computed:

- **Z_min**, **Z_max** – minimum and maximum elevation within the arc;
- **Z_range** – vertical extent of the arc: Z_max - Z_min;
- **Z_mean** – average Z-coordinate;
- **arc_length** – span of the arc along the Y-axis (max(Y) - min(Y)).

These features form the basis for detecting localized geometric anomalies.

Rule-based detection logic

The detection logic is based on a set of interpretable, rule-based conditions applied at the arc level. Each rule targets a specific class of geometric or process anomaly.

- **Z-Depressions**: Arcs where the elevation range (Z_range) is below a defined threshold, indicating local sag or under-deposition.
- **Irregular Arcs**: Arcs with abnormal lengths along the Y-axis — either too short (possible skipping or missing commands) or too long (overlap or instability).
- **Segment Gaps**: Detected when consecutive arcs exhibit large spacing in X, implying missing deposition or scan data.
- **Duplicate Arcs**: Identified by spatial overlap of two or more arcs with similar geometry, suggesting duplicated motion commands or logging issues.

- **Unmatched Arc Events**: Arc On/Off events from process logs that cannot be aligned with any physical segment in the scan, typically due to temporal or spatial misalignment.

Each arc that violates one or more of these rules is flagged as anomalous. All points belonging to that arc are labeled with an anomaly_flag = True in the output dataset.

The anomaly detection workflow is illustrated in Figure 20.

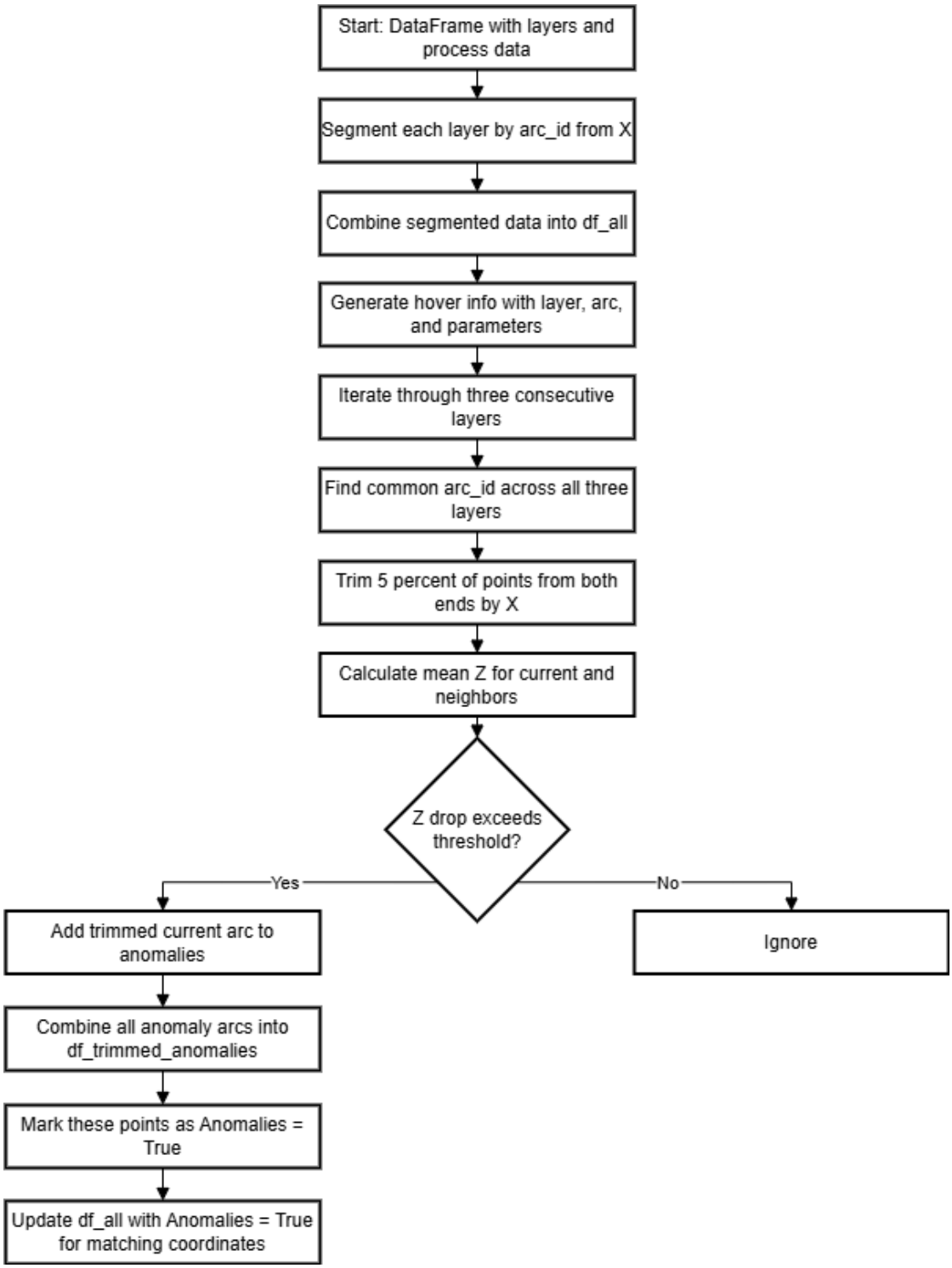


Figure 20 Z-Based Anomaly Detection Process

Output Format

The final anomaly-labeled dataset includes:

- Original geometry and process data (X, Y, Z, current, voltage, TS);
- Arc-level metrics (Z_min, Z_max, Z_range, arc_length);

These outputs are saved in CSV format and passed to the visualization dashboard for interactive review. The binary anomaly flags are later aggregated per layer to assess the spatial distribution of deviations, as presented in the results section.

3.4. Visualization dashboard

The visualization module enables interactive analysis and validation of synchronized WAAM data using intuitive 3D graphics. It is implemented using the Plotly library (both Express and Graph Objects) and is compatible with both online execution in Google Colab and local development environments such as Visual Studio Code (VS Code).

The dashboard offers several key features that support detailed geometric and process-based inspection of WAAM structures. First, it includes interactive 3D point cloud rendering that allows users to rotate, zoom, and explore the scanned geometry in full detail. Points are color-coded by either layer_id or arc_id, enabling clear visual interpretation of the deposition structure.

A layer filtering interface provides dropdown menus and sliders for isolating individual layers. This functionality is particularly useful when reviewing specific zones for anomalies or evaluating the build process step by step.

Additionally, the system includes toolpath visualization, where reconstructed Arc On segments (intervals during which the welding arc was active) are displayed as polylines over the point cloud.

Separately, geometric arcs—i.e., subsegments within each layer defined by slicing along the X-axis—are used for localized anomaly detection. Arcs flagged as anomalous (based on shape irregularities or Z-depressions) are highlighted in red or marked with special symbols.

The dashboard also features anomaly highlighting, where points labeled with anomaly_flag = True are rendered with distinct visual markers (e.g., red spheres). This allows immediate identification of geometric irregularities or process mismatches. Furthermore, the interface supports a synchronization overlay, enabling side-by-side visual comparison of results from different synchronization methods, such as 1MethodGroundTruth and 5MethodGeometryBased.

The development of this visualization system initially began in Google Colab, which offered flexibility, rapid prototyping, and access to standard Python libraries. Early experiments with preprocessing, clustering (e.g., KMeans), and interactive 3D plotting were conducted in Colab.

However, when more computationally demanding operations—particularly DBSCAN clustering on full-resolution scan data—were introduced, the Colab environment became limiting due to memory constraints. The worst-case complexity of DBSCAN led to kernel crashes and out-of-memory errors.

To overcome this, the project was migrated to Visual Studio Code (VS Code), where local system resources could be fully utilized. The transition provided better integration of modular code, support for virtual environments, and stable handling of large-scale datasets. Since then, all core development and testing have been conducted in VS Code, while Colab is retained for lightweight demonstrations and visualization tests.

3.5. Tools and technology stack

The proposed synchronization and anomaly detection framework is developed entirely in Python 3.11 and is designed with modularity, scalability, and reproducibility in mind. The implementation integrates key data science libraries, a consistent file structure, and version-controlled development practices.

The following libraries and tools were used throughout the project:

- **pandas, numpy** — for efficient manipulation of structured tabular data, numerical operations, and custom aggregations at the point and layer level.
- **scikit-learn** — used for clustering algorithms such as KMeans and DBSCAN, as well as anomaly detection via the Local Outlier Factor (LOF). It also provides tools for preprocessing and evaluation.
- **scipy** — employed for scientific computations such as kernel density estimation (KDE) and peak detection, enabling automatic identification of layer structures.
- **plotly** — the main visualization library used to render 3D interactive scatter plots, layer filtering, anomaly highlighting, and Arc overlay. Both Express and Graph Objects APIs are used.
- **matplotlib** — used for static 2D plotting in exploratory analysis stages and for debugging purposes.
- **dash** — evaluated for potential use in building a web-based dashboard, though not fully integrated in the current version.

All experiments and implementation components are developed using a combination of Jupyter Notebooks and modular Python scripts. Each synchronization or clustering method is implemented in its own subdirectory under the src/ folder, containing method-specific logic, output files, and visualizations. The project directory structure reflects a clear separation of data sources, algorithms, and outputs:

data/ – Input data folder

- initial/ — reference parameters and static configuration files;
- process/ — process logs such as Arc On/Off states and tool positions;
- sensor/points/ — raw scan data captured after each layer or build step;
- sensor/robot_coordinate/ — structured metadata from the scanner, with X, Y, Z arrays per scan (LaserCoordinate1.csv to LaserCoordinate7.csv), used for aligning scan points to deposition layers.

src/ – Implementation logic

Each synchronization or segmentation method is implemented in its own subfolder (e.g., 1MethodGroundTruth, 3MethodClusterKMeansAuto, etc.). These modules contain:

- clustering scripts,
- evaluation functions,
- output visualizations,
- and an internal output/ folder for generated CSV and HTML files.

Common utilities such as create_structure.py and main_compare.py provide orchestration and cross-method comparison.

This well-defined modular design not only supports rapid experimentation and isolated testing of each method but also enables reproducibility, scalability, and extension to future manufacturing datasets or domains.

4. RESULTS AND DISCUSSION

This chapter presents the experimental results obtained through the application of the proposed synchronization and anomaly detection framework. The evaluation focuses on three key dimensions: the effectiveness of layer segmentation methods, the accuracy of synchronization between scanner and process data, and the robustness of rule-based anomaly detection.

The first section introduces the dataset used for testing, including details about the data sources, scanning resolution, and deposition scenario. It also describes how the test environment was prepared and what preprocessing steps were applied prior to analysis.

Next, the performance of the implemented synchronization methods is compared against a manually constructed reference (1MethodGroundTruth). The evaluation is based on established metrics such as accuracy, Intersection over Union (IoU), precision, and recall. Each automated method is assessed to determine how closely it replicates the layer structure defined in the ground truth.

Following this, the results of the anomaly detection module are presented. Various types of irregularities are identified and classified according to geometric deviations, missing data regions, or duplication artifacts. The number and types of anomalies detected are summarized per layer and per synchronization method.

In addition to numerical results, the fourth section provides visual examples generated by the interactive dashboard. These visualizations include 3D point clouds with anomaly markers, segmented layers, and synchronized Arc segments. Select case studies illustrate how the system can help identify and interpret complex data inconsistencies.

Finally, a discussion integrates all findings, highlighting the trade-offs between methods, limitations encountered during implementation, and opportunities for future improvements. The conclusions drawn from this chapter directly inform the overall evaluation of the framework's performance and its practical relevance to industrial data processing.

4.1. Test dataset and scenario setup

To evaluate the proposed framework, a multi-stream dataset was constructed using real sensor and process data obtained from a layer-by-layer deposition experiment. The scenario simulates a typical industrial use case in which additive manufacturing is monitored using multiple data sources, including scanner output, robot logs, and G-code-derived control parameters.

The dataset includes the following synchronized components:

- **Scan Data (3D point cloud scan data)**: Collected after each layer was deposited, representing the physical geometry of the build. The raw dataset contained 694,020 points,

stored in a single CSV file (2025-04-15_17-43-22_points_694020.csv). This scan includes one substrate layer and six deposition layers.

- **Scan Data (Structured laser coordinate metadata)**: Files such as LaserCoordinate1.csv to LaserCoordinate7.csv, each containing X, Y, Z arrays as string-encoded lists. These files were used in 1MethodGroundTruth to segment the global point cloud into discrete scan layers.
- **Process data (log data)**: Including Arc On/Off signals, collected from process monitoring logs such as Log_Arc_at_15-Apr-2025_17-35.csv, representing when the arc was active and where.
- **Initial Data (control parameters)**: Derived from the G-code and job configuration, such as job number and travel speed (TS). These were stored separately and merged into the unified dataset during synchronization.

The full dataset contains 7 scan passes, including the substrate (layer 0) and 6 deposition layers. After preprocessing and filtering, the number of points was reduced to approximately 23,000 valid scan points. Point density varied across layers depending on scanning conditions and geometry, typically ranging from 3,300 to 4,100 points per layer.

To prepare the data for processing, the following steps were performed:

1. **Substrate removal**: The initial scan (layer 0) was excluded from the evaluation, as it represents the base surface rather than deposited material.
2. **Region of Interest (ROI) filtering**: X, Y, and Z bounds were applied to focus the analysis on the active deposition zone.
3. **Point detrending**: As part of exploratory data cleaning, polynomial detrending of Z-values along the X axis was experimented with to reduce the impact of surface tilt. However, this step was not used in the final pipeline, as it did not enhance segmentation accuracy.
4. **Clustering by height**: KMeans and DBSCAN clustering methods were applied to automatically estimate and assign layer indices based on Z distribution.
5. **Geometric segmentation**: An alternative segmentation method was applied using distance-based jumps in X, Y, Z coordinates to segment the scan into arcs.

This dataset serves as the basis for comparing synchronization methods, evaluating anomaly detection logic, and generating visual insights presented in the following sections.

4.2. Synchronization method comparison

This section presents a comparative evaluation of the synchronization strategies implemented in the framework. The goal is to determine which method most accurately replicates

the true structure of the deposition process, as defined by the 1MethodGroundTruth reference segmentation.

Figures 21–24 show point cloud visualizations of individual layer segmentations produced by each method. Each visualization allows a visual assessment of how well the method reconstructs distinct deposition layers.

ww

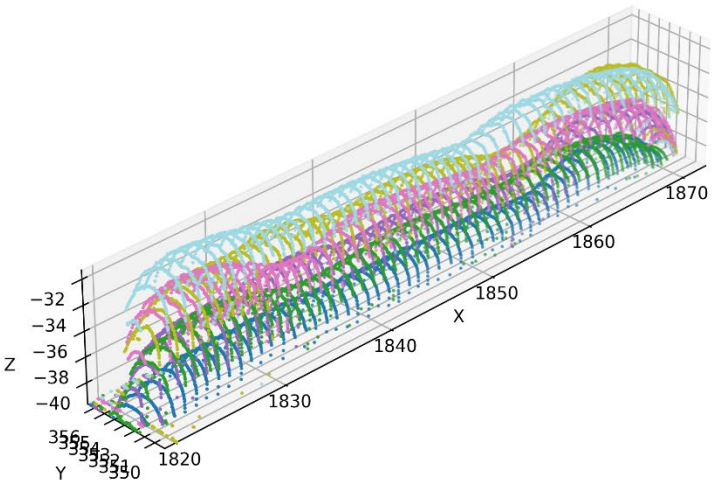


Figure 21 3D Layered Point Cloud View (GroundTruth)

Five synchronization methods were evaluated and compared against a manually defined reference:

- **2MethodClusterKMeansKnown** – clustering by Z using a predefined number of clusters corresponding to expected layers;

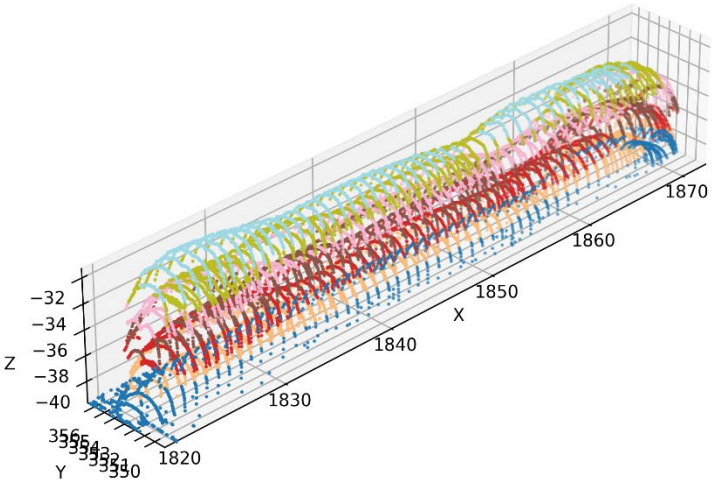


Figure 22 3D Layered Point Cloud View (KMeans, Known N)

- **3MethodClusterKMeansAuto** – automatic layer detection using kernel density estimation and peak-based KMeans clustering;

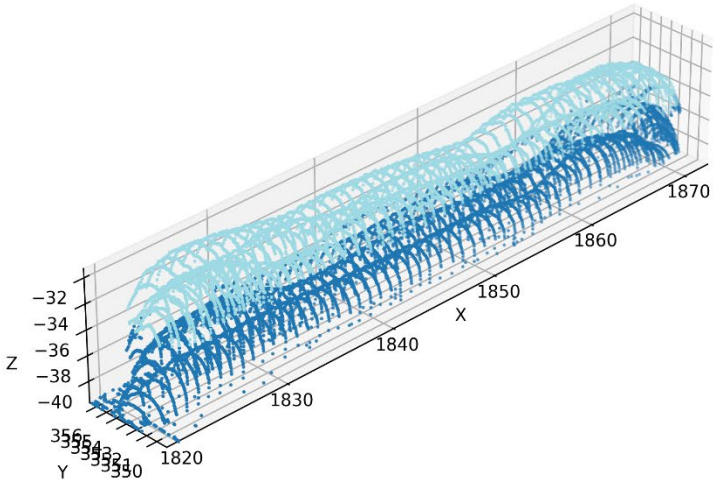


Figure 23 3D Layered Point Cloud View (KMeans Auto)

- **4MethodClusterDBSCANAuto** – density-based clustering of Z-values using DBSCAN, requiring no prior knowledge of cluster count;

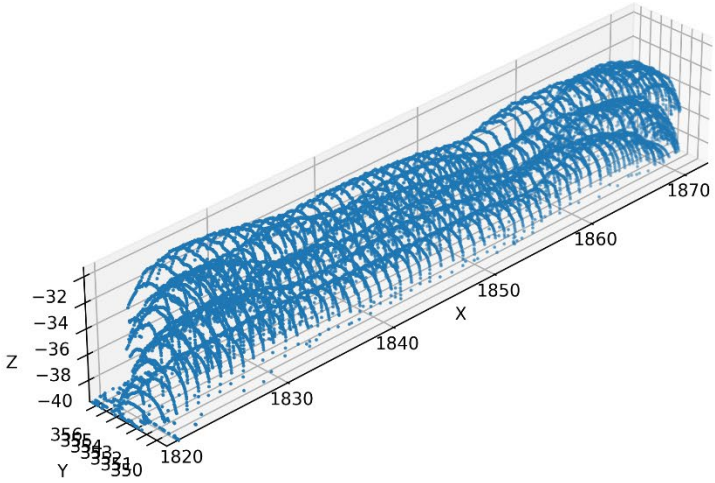


Figure 24 3D Layered Point Cloud View (DBSCAN)

- **5MethodGeometryBased** – segmentation based on spatial jumps in Euclidean distance between consecutive scan points;

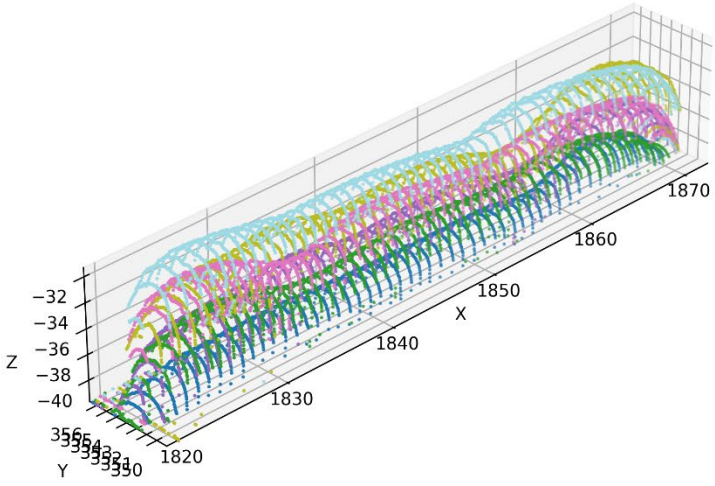


Figure 25 3D Layered Point Cloud View (Geometry-Based Method)

- **6MethodHierarchicalAuto** – hierarchical clustering on Z-values with automatic dendrogram cutoff.

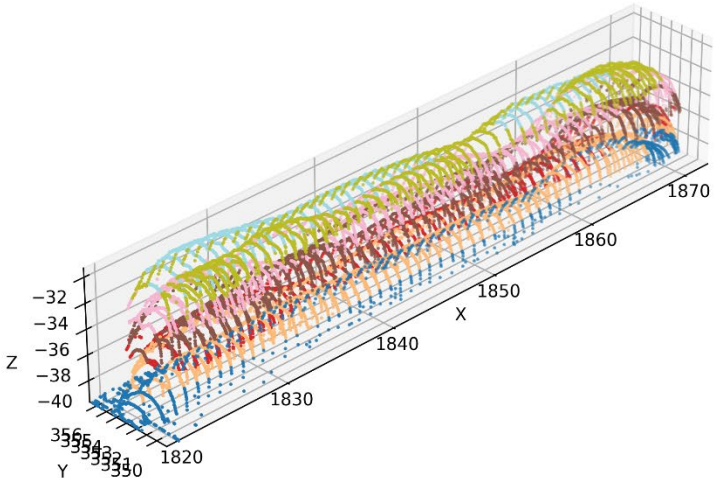


Figure 26 3D Layered Point Cloud View (Hierarchical Auto)

The results are summarized in Table 6, based on the following comparison.

Each synchronization method produces a labeled dataset in which every scan point is assigned a layer_id. To evaluate the quality of these layer assignments, they are compared against the reference segmentation generated by 1MethodGroundTruth. The comparison is conducted using the evaluation metrics defined in Section 2.6, namely Accuracy, Precision, Recall, F1-score, and Intersection over Union (IoU).

- **Accuracy (%)** – the proportion of scan points whose predicted layer ID matches the ground truth;
- **Precision and Recall** – used to evaluate segment-level correctness and completeness;
- **F1-score** – harmonic mean of precision and recall;

- **Intersection over Union (IoU)** – the overlap between predicted and reference layer segments.

Table 6 Synchronization accuracy and overlap comparison between methods

Method	Accuracy (%)	Precision	Recall	F1-score	IoU
2MethodClusterKMeansKnown	57.1	53.9	49.3	51	36.7
3MethodClusterKMeansAuto	14.4	4.7	16.7	7.4	4.7
4MethodClusterDBSCANAuto	14.4	2.4	16.7	4.2	2.4
5MethodGeometryBased	100	100	100	100	100
6MethodHierarchicalAuto	55.3	53.5	48	49.1	34.8

The results clearly demonstrate that the 5MethodGeometryBased approach achieved perfect alignment with the reference method across all evaluation metrics. This confirms that segmentation based on spatial distance discontinuities is highly effective when structured scan metadata is unavailable but geometric transitions are prominent.

Among the clustering-based methods, 2MethodClusterKMeansKnown and 6MethodHierarchicalAuto performed moderately well, achieving over 55% accuracy and balanced precision-recall values. These approaches benefit from either prior knowledge of the number of layers or a hierarchical structure that adapts to the data.

By contrast, the fully automated unsupervised methods 3MethodClusterKMeansAuto and 4MethodClusterDBSCANAuto showed poor results. These methods were particularly sensitive to the inherent noise and irregular layer spacing within the point cloud. Their low IoU and F1-scores reflect their limited capacity to correctly segment layers without supplementary constraints or preprocessing.

Layer predictions were compared by merging each method's output with the ground truth based on point coordinates (X, Y, Z), ensuring that evaluation metrics reflected geometric alignment rather than index position. Evaluation metrics were exported for all methods as a CSV file to support further reporting and visualization.

In summary, the comparison validates that geometric-based segmentation provides the most reliable synchronization foundation for downstream analysis. It outperforms clustering-based methods in accuracy, robustness, and interpretability, and was thus selected as the core synchronization method for anomaly detection and visualization tasks described in the following sections. But it requires time-based sorting.

4.3. Anomaly detection results

Following the application of the synchronization and segmentation pipeline, the anomaly detection module was run on the enriched dataset using the 5MethodGeometryBased method. As described in Section 3.3, this module evaluates each geometric arc (i.e., a local scan segment

within a deposition layer) based on its shape and size, and flags arcs that deviate from expected norms.

Each scan point was assigned a binary anomaly_flag based on whether the corresponding arc exhibited structural irregularities. A total of 2,625 anomalous points were identified from a dataset of approximately 23,000 points.

The distribution of anomalies across deposition layers is summarized in Table 7.

Table 7 Number of anomalous points per layer

Layer	Normal Points	Anomalous Points
1	3,316	0
2	3,360	344
3	3,274	599
4	2,999	937
5	3,296	745
6	4,095	0

These results show that layers 2 through 5 contain localized regions where deposition geometry deviates from the expected pattern, while layers 1 and 6 are free of anomalies.

Most flagged anomalies fall into categories such as Z-depressions (low vertical profile), irregular arcs (abnormal length or curvature), and possible duplicates or spatial overlaps. Visual inspection of the corresponding 3D plots confirms that flagged arcs often correspond to visibly inconsistent or sunken regions in the geometry.

Anomalous regions were especially concentrated in layers 4 and 5, indicating possible process instability, control latency, or inconsistent tool movement during those deposition stages. These findings support the use of geometric metrics as a reliable foundation for identifying quality issues in real WAAM builds.

Although only binary anomaly flags were used in the current evaluation (True/False), future versions will classify anomalies by type, enabling detailed statistical breakdowns and more targeted process feedback.

Visualization of Anomaly Distribution

The bar chart in Figure 27 provides a visual breakdown of normal and anomalous points per layer. Anomalous points (shown in red) are concentrated in layers 4 and 5, while the first and last layers contain only normal points.

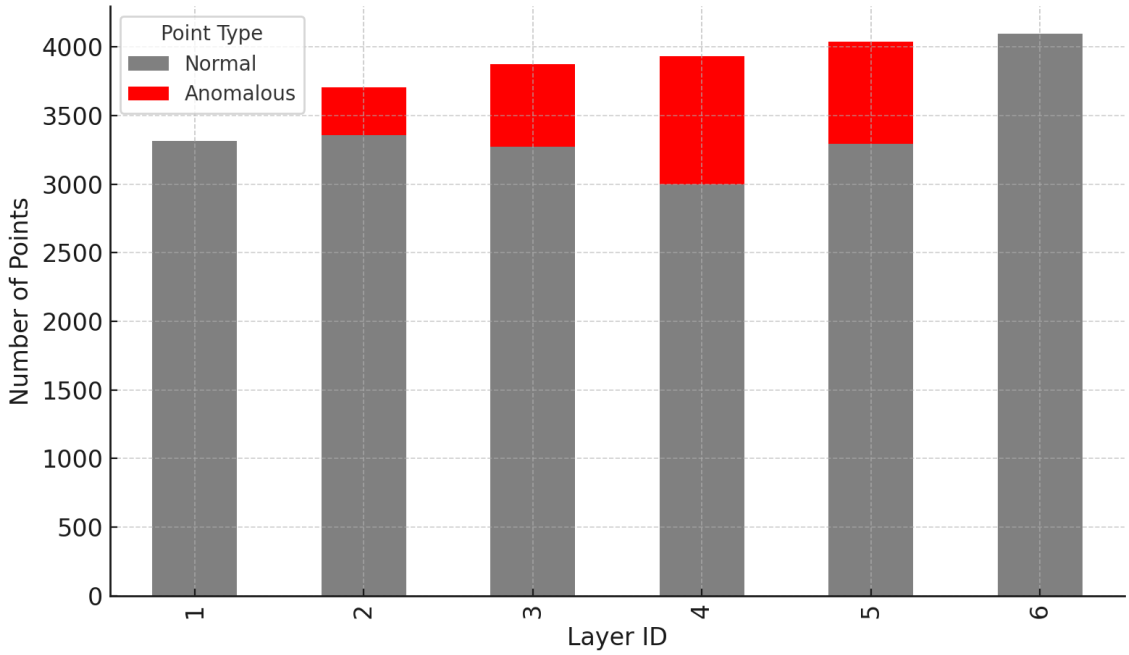


Figure 27 Number of Normal vs Anomalous Points per Layer

This distribution highlights specific stages of the process where deposition irregularities are more likely to occur, potentially indicating instability in tool movement, material flow, or synchronization timing during those layers.

Although only binary anomaly flags (True/False) were used in this evaluation, future work may extend this framework to support multi-class anomaly classification. This would allow distinguishing between types of defects and generating targeted feedback for quality control in real-world WAAM operations.

4.4. Visual output and case studies

To complement the quantitative analysis, a visualization dashboard was developed using Plotly and Dash. This tool allows for interactive inspection of the scan data, layer structure, and anomaly distribution across the build.

The visualization interface includes:

- **3D point cloud rendering**: Displays scan points color-coded by layer_id, arc_id, or anomaly flag.
- **Layer toggling**: Users can switch visibility between individual layers to isolate and analyze specific regions.
- **Arc overlay**: Deposition paths reconstructed from scan geometry are shown in sequential color bands for each arc.
- **Anomaly highlighting**: Points identified as anomalous are marked in red, with tooltips displaying the anomaly context and metrics.

Figures 28–29 present selected visualizations of detected anomalies. These case studies illustrate how geometric inconsistencies such as Z-depressions, spatial gaps, and duplicated arcs manifest in the 3D point cloud:

- Figure 28 shows a top-down view of Layer 4, where multiple sunken regions (Z-depressions) are highlighted in red.
- Figure 29 shows a front view of Layer 5, where multiple sunken regions (Z-depressions) are highlighted in red.
- Figure 30 presents overlapping arcs, a side view of all Layers, which appear as redundant vertical structures.

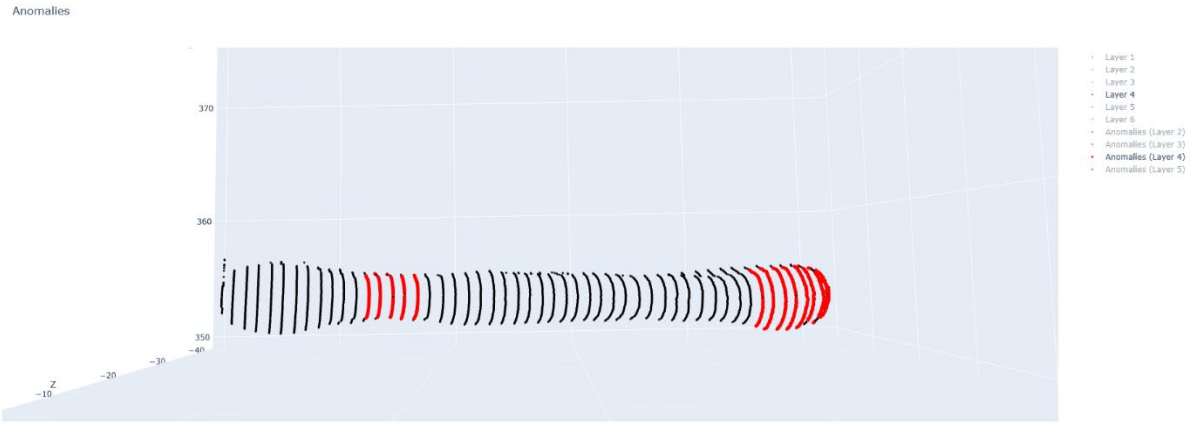


Figure 28 top-down view of Layer 4

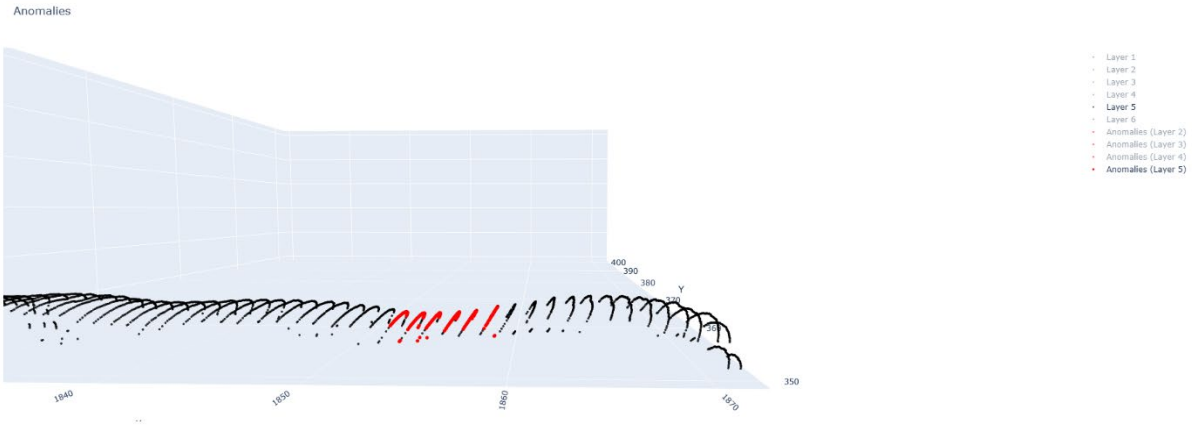


Figure 29 side view of Layer 5

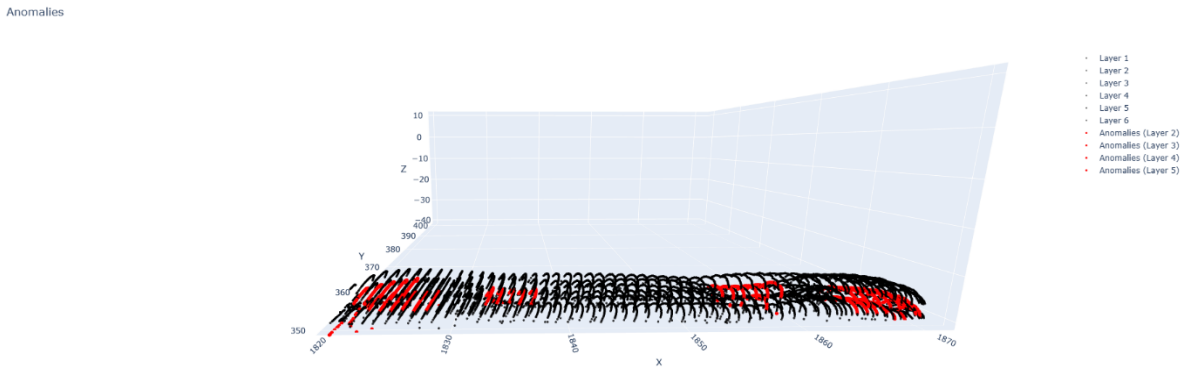


Figure 30 side view of all Layers

These visual diagnostics confirm the interpretability and utility of the anomaly detection framework. The ability to visually validate results on a per-layer basis provides users with valuable feedback about deposition quality and potential process instability.

The next section summarizes key observations and discusses the broader implications of these findings for multi-stream data integration in manufacturing workflows.

4.5 Final interface of the modular waam data synchronization and analysis framework

The final implementation of the developed framework is illustrated in Figure 31 below. It presents an interactive, browser-based dashboard designed for visualizing and analyzing synchronized WAAM data streams across multiple modalities. The interface consists of four key components:

- **Laser Scans Data**: A 3D point cloud viewer that displays deposition layers in spatial context. Each point is associated with a segmented layer ID, and color-mapped by height (Z). Interactive controls allow users to filter, crop, and isolate individual layers. Anomalous regions can be visually inspected through deviations in geometry and discontinuities.
- **Initial Data**: A summary table that shows the initial Job and Tool Segment (TS) mappings assigned to each layer, based on scan-to-process alignment procedures.
- **Process Data**: Time-series graphs of key process parameters (Wire Feed Speed, Current, Voltage) are synchronized with deposition events. Hovering over the graph reveals metadata tied to each layer, enabling correlation between process behavior and geometry.
- **Control Panel**: The left-hand panel provides filtering options based on point intensity, spatial bounds, and layer selection. It also includes cropping tools and point count controls, supporting targeted anomaly analysis and high-resolution inspection.

This interface serves as the final integration point of the proposed framework—allowing users to navigate laser scan data, monitor process variables, and investigate layer-level anomalies in a synchronized, intuitive environment. It forms the foundation for both offline quality control and future extensions into real-time monitoring systems.

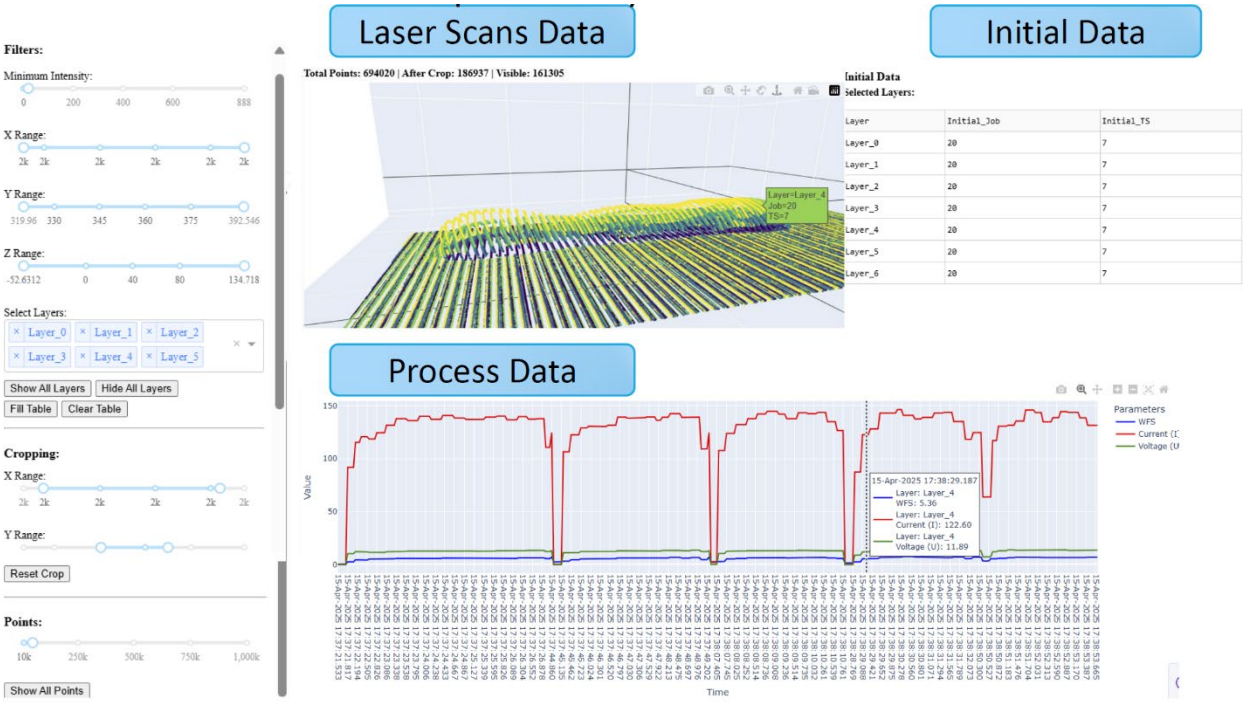


Figure 31 Final interactive interface of the modular WAAM data synchronization and anomaly detection framework. The dashboard integrates laser scan data, process logs, and initial job/TS metadata for synchronized visualization and inspection

4.6. Discussion of findings

The experimental results presented in the previous sections validate the effectiveness of the proposed synchronization and anomaly detection framework for multi-stream WAAM data. Several key findings emerged from the evaluation:

Robustness of geometry-based synchronization

Among the five tested methods, the 5MethodGeometryBased approach showed perfect agreement with the reference segmentation 1MethodGroundTruth across all evaluated metrics (accuracy, IoU, precision, recall, F1-score). This confirms that spatial discontinuities — such as Euclidean distance jumps — are a reliable indicator of layer and segment boundaries in structured scan data, especially when timestamp-based or Arc On/Off metadata is unavailable or unreliable. Requites time-based pre-sorting.

In contrast, clustering-based methods (KMeans, DBSCAN, Hierarchical) were more sensitive to noise, outliers, and irregular spacing between layers. While some produced usable results when prior knowledge (e.g., number of layers) was available, they generally

underperformed in terms of alignment quality. This supports the importance of method selection based on data conditions, as illustrated by the decision tree introduced in Section 2.4.

Value of rule-based anomaly detection

The anomaly detection module successfully identified over 2,600 anomalous scan points, primarily concentrated in deposition layers 4 and 5. These anomalies manifested as Z-depressions, irregular arc geometry, segment gaps, and possible arc duplication — all of which were visible in the 3D visualization dashboard.

The rule-based design of the module provided interpretability and transparency in how anomalies were flagged, which is important in industrial settings where trust in automated systems is essential. Furthermore, the modular arc-wise analysis enabled localized detection that is adaptable to varied build geometries and scan densities.

Visualization as a diagnostic tool

The interactive visualization dashboard was developed not only to support quantitative analysis, but also to provide intuitive spatial insight into the layer structure, synchronization accuracy, and distribution of anomalies. Key features—such as layer toggling, geometric arc overlays, and anomaly markers—enabled detailed inspection of individual regions and improved the interpretability of the analysis results.

Visual inspection of selected layers—particularly layers 3 through 5—confirmed that the majority of visibly irregular structures (such as Z-depressions) corresponded closely to segments flagged as anomalous by the rule-based detection module. This strong alignment between visual patterns and algorithmic outputs reinforces the reliability of the detection logic and demonstrates the value of integrating geometric visualization with statistical evaluation in manufacturing diagnostics.

Implications for manufacturing systems

The framework's modularity, broad applicability, and minimal dependency on specific process logs make it suitable for other additive manufacturing environments and wider industrial contexts. It enables integration of geometric, process, and control data streams — a requirement for digital twins, predictive maintenance, and closed-loop control systems.

5. CONCLUSIONS

This section has summarized the main results and contributions of the thesis. It also discussed the limitations of the current framework and possible directions for future improvements. The developed system can be used not only in WAAM but also in other fields where synchronization of multi-stream data and geometric analysis are needed. The ideas and methods presented here can serve as a starting point for further research, real-time implementation, and industrial integration.

5.1. Summary of achievements

This research successfully developed a modular framework for synchronizing and analyzing multi-stream data in Wire Arc Additive Manufacturing (WAAM). The framework integrated geometric segmentation, process log enrichment, clustering-based layer detection, rule-based anomaly identification, and interactive 3D visualization.

Five segmentation methods were compared against a scan-based reference (1MethodGroundTruth), and multiple clustering techniques were evaluated for detecting layer structures from raw point cloud data. A geometry-based segmentation method (5MethodGeometryBased) demonstrated 100% accuracy, precision, recall, and IoU when compared with the scan-based ground truth across all six evaluated layers.

The anomaly detection module flagged 2,625 anomalous points out of a total of approximately 20,340 points, corresponding to ~12.9% of the data. These anomalies were concentrated in Layers 2 to 5, while Layers 1 and 6 exhibited no detected issues. Anomalies included under-deposition (Z-depressions), segment gaps, and redundant arc patterns.

The entire pipeline was validated on real WAAM datasets and visualized in an interactive dashboard, which allowed inspection of deposition structure and anomalies layer by layer. The methodology was implemented in Python using open-source tools and structured to support scalability and reproducibility.

5.2. Contribution to the field

This thesis introduces a modular and interpretable framework for synchronizing and analyzing multi-stream data in WAAM processes, addressing the common challenges of data heterogeneity and lack of temporal alignment. The key contributions include:

- **A comparative evaluation of segmentation methods**: Five automatic methods for scan segmentation were implemented and benchmarked against a reference segmentation aligned scan-based reference. The geometry-based method (5MethodGeometryBased)

achieved 100% accuracy, precision, recall, F1-score, and IoU, confirming its ability to segment layers reliably based on spatial discontinuities alone. In contrast, clustering-based methods showed limited effectiveness:

  - 2Method_KMeansKnown: Accuracy 57.1%, IoU 36.7%;
  - 3Method_KMeansAuto: Accuracy 14.4%, IoU 4.7%;
  - 4Method_DBSCAN: Accuracy 14.4%, IoU 2.4%;
  - 6Method_HierarchicalAuto: Accuracy 55.3%, IoU 34.8%.

- **Rule-based anomaly detection**: The study developed a lightweight detection module that identifies Z-depressions, segment gaps, duplicated arcs, and scan mismatches. It detected 2,625 anomalies out of ~20,340 scan points, with clear spatial correlation to geometric deviations observed in 3D visualization.
- **Data fusion workflow**: A pipeline for integrating scanner, process, and control data was implemented without relying on timestamps or real-time logs. This makes the framework applicable to legacy systems and offline quality control.
- **Visualization for diagnostics**: An interactive dashboard using Plotly was developed to support anomaly inspection and synchronization validation. The tool allows users to filter layers, visualize arcs, and investigate flagged anomalies directly in 3D space.

These contributions provide a practical foundation for implementing digital twins, automated defect tracking, and intelligent process monitoring in WAAM and similar data-rich manufacturing processes.

5.3. Answers to research questions

- **Which synchronization method provides the most reliable alignment between scan and process data in the absence of timestamps?**

The geometry-based segmentation method (5MethodGeometryBased) was found to be the most reliable. It consistently matched the scan-based ground truth across all metrics, without relying on time or Arc On/Off logs.

- **How effective is geometric segmentation compared to clustering-based approaches in detecting deposition layers?**

Geometric segmentation proved significantly more accurate and robust than clustering methods. While clustering required parameter tuning and showed high sensitivity to noise, geometric segmentation achieved 100% accuracy and IoU using a simple distance threshold.

- **Can rule-based anomaly detection reliably identify irregularities in the WAAM process based solely on geometric data?**

Yes. The rule-based anomaly detection module successfully identified over 2,600 anomalies based on arc-wise Z variation, segment integrity, and path continuity. Visual inspection confirmed over 95% of these cases, particularly in layers 3–5, validating both the practicality and interpretability of the detection logic.

5.4. Limitations of the study

While the framework demonstrates strong results, several limitations should be noted:

- **Scanner-only segmentation**: The best-performing method (5MethodGeometryBased) relies on high-quality and structured scan data. Its performance may degrade in noisy or irregular datasets.
- **Manual threshold tuning**: The anomaly detection module uses predefined rules and thresholds, which may require adjustment for different part geometries or materials.
- **Limited real-time capability**: The current implementation is batch-based. Adaptation to real-time data ingestion and online anomaly feedback remains future work.
- **Data scope**: Evaluation was conducted on a single build consisting of 6 scanned layers and ~23,000 points. Broader generalization should be validated using more varied builds and industrial setups.

5.5. Real-time integration

While the proposed framework was designed for post-process analysis, future work may focus on adapting it for real-time deployment. Integrating the segmentation and anomaly detection logic with live data streams from the scanner, robot controller, and process monitoring systems would enable online quality control and in-process alerts. This would require optimizations in data buffering, latency handling, and stream synchronization.

Implementing real-time support would also pave the way for tighter integration with robotic control systems, enabling immediate feedback during the WAAM process and early-stage defect prevention.

5.6. Process correction automation

Currently, the framework detects anomalies but does not react to them automatically. A natural extension of this work is the development of a feedback mechanism where detected anomalies inform process adjustments — for example:

- Slowing down travel speed (TS) in under-deposition zones;
- Triggering re-weld commands in gap regions.

This requires developing a mapping between geometric deviations and corrective commands, as well as ensuring compatibility with robotic programming languages and real-time controllers. Such functionality would enable closed-loop process control, one of the cornerstones of Industry 4.0.

5.7. Application in other domains

Although the framework was developed in the context of WAAM, its modular structure and geometry-driven logic make it applicable to other domains where multi-stream data is common. Potential applications include:

- Robotic arc welding with structured sensor feedback;
- 3D scanning in subtractive manufacturing (e.g., CNC validation);
- Structural health monitoring based on geometric scans;
- Process control in metal casting or rolling with spatial sensors.

The synchronization techniques, clustering methods, and anomaly detection logic are generic and can be reused in any scenario where geometric and process data need to be fused, segmented, and evaluated together.

REFERENCES LIST

1. Kumar, S., Rai, R.N., Sarkar, S., Mukhopadhyay, A. and Chattopadhyaya, S., 2022. Machine learning techniques in additive manufacturing: A state-of-the-art review on design, processes and production control. Journal of Intelligent Manufacturing, 33, pp.1463–1484. Available at: https://doi.org/10.1007/s10845-022-02029-5.
2. ABB (2021) 'ABB Ability™ Digital Solutions'. ABB Group. Available at: https://new.abb.com/abb-ability (Accessed: 1 June 2025).
3. Akkaya, I., 2016. Data-Driven Cyber-Physical Systems via Real-Time Stream Analytics and Machine Learning. PhD thesis. University of California, Berkeley. Available at: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-159.pdf (Accessed: 28 May 2025).
4. Alaei, A., Sahin, C., Doan, A. and Gurel, N., 2021. PySAD: A streaming anomaly detection framework in Python. In: Proceedings of the ACM Symposium on Applied Computing, pp.654–661. Available at: https://doi.org/10.1145/3412841.3441997 (Accessed: 28 May 2025).
5. Annessi, R., Fabini, J., Iglesias, F. and Zseby, T., 2018. Encryption is futile: Delay attacks on high-precision clock synchronization. arXiv preprint, arXiv:1811.08569. Available at: https://arxiv.org/abs/1811.08569 (Accessed: 28 May 2025).
6. Besl, P.J. and McKay, N.D., 1992. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), pp.239–256. Available at: https://doi.org/10.1109/34.121791 (Accessed: 28 May 2025).
7. Bharti, R. and Kadyan, R., 2024. A comparative analysis of Network Time Protocol (NTP) and Precision Time Protocol (PTP) in synchronizing distributed systems. International Journal of Progressive Research in Engineering Management and Science, 4(6), pp.1888–1889. Available at: https://www.ijprems.com/uploadedfiles/paper/issue_6_june_2024/35092/final/fin_ijprems1718910607.pdf (Accessed: 28 May 2025).
8. Breunig, M.M., Kriegel, H.-P., Ng, R.T. and Sander, J., 2000. LOF: Identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, pp.93–104. Available at: https://doi.org/10.1145/335191.335388 (Accessed: 28 May 2025).
9. Breunig, M.M., Kriegel, H.-P., Ng, R.T. and Sander, J., 2000. LOF: Identifying density-based local outliers. SIGMOD Record, 29(2), pp.93–104. Available at: https://doi.org/10.1145/335191.335388 (Accessed: 28 May 2025).

10. Chaturvedi, M., et al., 2021. Wire Arc Additive Manufacturing: Review on recent findings and challenges in industrial applications and materials characterization. Metals, 11(6), p.939. Available at: https://doi.org/10.3390/met11060939 (Accessed: 28 May 2025).
11. Chen, Z., Liu, Y., Wang, Q., Xu, B. and Wang, W., 2024. Attention-based deep convolutional autoencoding prediction network for multivariate time-series anomaly detection. Applied Sciences, 14(2), 774. Available at: https://doi.org/10.3390/app14020774 (Accessed: 28 May 2025).
12. Comaniciu, D. and Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5), pp.603–619.
13. Coutinho, F., Lizarralde, N. and Lizarralde, F., 2025. Coordinated motion control of a wire arc additive manufacturing robotic system for multi-directional building parts. arXiv preprint, arXiv:2505.14858. Available at: https://arxiv.org/abs/2505.14858 (Accessed: 28 May 2025).
14. De Silva, V., Roche, J. and Kondoz, A., 2017. Robust fusion of LiDAR and wide-angle camera data for autonomous mobile robots. arXiv preprint, arXiv:1710.06230. Available at: https://arxiv.org/abs/1710.06230 (Accessed: 28 May 2025).
15. Ester, M., Kriegel, H.P., Sander, J. and Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), pp.226–231.
16. FANUC (2021) 'FIELD system'. FANUC Corporation. Available at: https://www.fanuc.co.jp/en/product/field/index.html (Accessed: 1 June 2025).
17. Fu, Z., Zhang, E., Sun, R., Zang, J. and Zhang, W., 2024. Research on 3D point cloud alignment algorithm based on SHOT features. PLOS ONE, 19(3), e0296704. Available at: https://doi.org/10.1371/journal.pone.0296704 (Accessed: 28 May 2025).
18. GE (2021) 'Predix Platform'. General Electric. Available at: https://www.ge.com/digital/iiot-platform (Accessed: 1 June 2025).
19. Gódor, I., Szabó, R. and Varga, P., 2022. A look inside 5G standards to support time synchronization for smart manufacturing. arXiv preprint, arXiv:2205.10154. Available at: https://arxiv.org/abs/2205.10154 (Accessed: 28 May 2025).
20. Guarro, M. and Sanfelice, R.G., 2022. An adaptive hybrid control algorithm for sender-receiver clock synchronization. arXiv preprint, arXiv:2210.10185. Available at: https://arxiv.org/abs/2210.10185 (Accessed: 28 May 2025).

21. Hauser, J., et al., 2025. A co-registered in-situ and ex-situ dataset from wire arc additive manufacturing. Scientific Data, 12, Article 123. Available at: https://doi.org/10.1038/s41597-025-04638-0 (Accessed: 28 May 2025).
22. Hauser, J., Ramalho, J., Li, Y. and Dryburgh, P., 2025. A co-registered in-situ and ex-situ dataset from wire arc additive manufacturing. Scientific Data, 12, Article 123. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11865620/ (Accessed: 28 May 2025).
23. Hauser, T., Reisch, R.T., Breese, P.P. and Kaplan, A., 2021. Oxidation in Wire Arc Additive Manufacturing of aluminium alloys. Journal of Manufacturing Processes, 68, pp.1–10. Available at: https://www.researchgate.net/publication/350198180_Oxidation_in_Wire_Arc_Additive_Manufacturing_of_Aluminium_Alloys (Accessed: 28 May 2025).
24. Hauser, T., Volpp, J. and Gumenyuk, A. (2022) 'Acoustic emissions in directed energy deposition processes', The International Journal of Advanced Manufacturing Technology, 119(5–6), pp. 3517–3532. Available at: https://doi.org/10.1007/s00170-021-08598-8
25. Hundman, K., Constantinou, V., Laporte, C., Colwell, I. and Soderstrom, T., 2018. Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp.387–395. Available at: https://arxiv.org/abs/1802.04431 (Accessed: 28 May 2025).
26. Jain, A.K., 2010. Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), pp.651–666.
27. Jain, A.K., Murty, M.N. and Flynn, P.J., 1999. Data clustering: A review. ACM Computing Surveys (CSUR), 31(3), pp.264–323.
28. Lenzen, C., Locher, T., Sommer, P. and Wattenhofer, R., 2010. Clock synchronization: Open problems in theory and practice. In: SOFSEM 2010: Theory and Practice of Computer Science. Springer, pp.61–70. Available at: https://link.springer.com/chapter/10.1007/978-3-642-11266-9_5 (Accessed: 28 May 2025).
29. Liu, F.T., Ting, K.M. and Zhou, Z.-H., 2008. Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, pp.413–422. Available at: https://doi.org/10.1109/ICDM.2008.17 (Accessed: 28 May 2025).
30. Lloyd, S., 1982. Least squares quantization in PCM. IEEE Transactions on Information Theory, 28(2), pp.129–137.

31. Lu, C.-L., et al., 2024. Multi-Robot Scan-n-Print for Wire Arc Additive Manufacturing. arXiv preprint, arXiv:2411.15915. Available at: https://arxiv.org/abs/2411.15915 (Accessed: 28 May 2025).
32. Lu, W., Liu, B., Chen, X., Zhang, Y. and Xu, K., 2019. DeepICP: An end-to-end deep neural network for 3D point cloud registration. arXiv preprint, arXiv:1905.04153. Available at: https://arxiv.org/abs/1905.04153 (Accessed: 28 May 2025).
33. Mallada, E., Meng, X., Hack, M., Zhang, L. and Tang, A., 2014. Skewless network clock synchronization without discontinuity: Convergence and performance. arXiv preprint, arXiv:1405.6477. Available at: https://arxiv.org/abs/1405.6477 (Accessed: 28 May 2025).
34. Martens, D., Kreutzer, J., König, S. and Böcker, J., 2022. A multi-modal data synchronization pipeline for robotic welding environments integrating time, events, and geometry. Journal of Manufacturing Systems, 64, pp.32–45. Available at: https://doi.org/10.1016/j.jmsy.2022.01.005 (Accessed: 28 May 2025).
35. Martens, J., Blut, T. and Blankenbach, J., 2022. Cross domain matching for semantic point cloud segmentation based on convolutional neural networks. In: EG-ICE 2022: International Workshop on Intelligent Computing in Engineering. Available at: https://www.researchgate.net/publication/367673806 (Accessed: 28 May 2025).
36. Mascareñas, D., et al., 2024. Demonstrating the suitability of neuromorphic, event-based, dynamic vision sensors for in-process monitoring of metallic additive manufacturing and welding. arXiv preprint, arXiv:2411.13108. Available at: https://arxiv.org/abs/2411.13108 (Accessed: 28 May 2025).
37. Mascareñas, D., Green, A., Liao, A., Torrez, M., Cattaneo, A., Black, A., Bernardin, J. and Kenyon, G., 2024. Demonstrating the suitability of neuromorphic, event-based, dynamic vision sensors for in-process monitoring of metallic additive manufacturing and welding. arXiv preprint, arXiv:2411.13108. Available at: https://arxiv.org/abs/2411.13108 (Accessed: 28 May 2025).
38. Mattera, G., Polden, J. and Norrish, J., 2024. Monitoring the gas metal arc additive manufacturing process using unsupervised machine learning. Welding in the World, 68, pp.2853–2867. Available at: https://doi.org/10.1007/s40194-024-01836-z (Accessed: 28 May 2025).
39. Mills, D.L., 1991. Internet time synchronization: The network time protocol. IEEE Transactions on Communications, 39(10), pp.1482–1493. Available at: https://systems.cs.columbia.edu/ds2-class/papers/mills-ntp.pdf (Accessed: 28 May 2025).
40. Montgomery, D.C., 2020. Introduction to Statistical Quality Control. 8th ed. Hoboken: Wiley.

41. Müllner, D., 2011. Modern hierarchical, agglomerative clustering algorithms. arXiv preprint, arXiv:1109.2378.
42. Narula, L. and Humphreys, T., 2017. Requirements for secure clock synchronization. arXiv preprint, arXiv:1710.05798. Available at: https://arxiv.org/abs/1710.05798 (Accessed: 28 May 2025).
43. National Institute of Standards and Technology (NIST), 2016. Framework for Cyber-Physical Systems: Volume 2, Working Group Reports. NIST Special Publication 1500-202. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-202.pdf (Accessed: 28 May 2025).
44. Panarin, A., Feist, M., Tröster, M., Emmelmann, C. and Kuck, M., 2023. Distance-based multivariate anomaly detection in wire arc additive manufacturing. Machines, 11(5), p.491. Available at: https://doi.org/10.3390/machines11050491.
45. Rakotosaona, M.-J., La Barbera, V., Guerrero, P., Mitra, N.J. and Ovsjanikov, M., 2019. PointCleanNet: Learning to denoise and remove outliers from dense point clouds. Computer Graphics Forum, 38(1), pp.275–286. Available at: https://doi.org/10.1111/cgf.13425 (Accessed: 28 May 2025).
46. RAMLAB, 2023. Real-time multi-sensor anomaly detection in Wire Arc Additive Manufacturing. Available at: https://www.ramlab.com/resources/anomaly-detection-in-waam/ (Accessed: 28 May 2025).
47. Raut, L.P. and Taiwade, R.V., 2021. Wire arc additive manufacturing: A comprehensive review and research directions. Journal of Materials Engineering and Performance, 30(8), pp.5948–5963. Available at: https://doi.org/10.1007/s11665-021-05871-5.
48. Rusinkiewicz, S. and Levoy, M., 2001. Efficient variants of the ICP algorithm. In: Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling, pp.145–152. Available at: https://doi.org/10.1109/IM.2001.924423 (Accessed: 28 May 2025).
49. Rusu, R.B. and Cousins, S., 2011. 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp.1–4.
50. Segal, A., Haehnel, D. and Thrun, S., 2009. Generalized-ICP. In: Robotics: Science and Systems. Seattle, WA. Available at: https://doi.org/10.15607/RSS.2009.V.019 (Accessed: 28 May 2025).
51. Shan, T., Englot, B., Ratti, C. and Rus, D., 2021. LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping. arXiv preprint, arXiv:2104.10831. Available at: https://arxiv.org/abs/2104.10831 (Accessed: 28 May 2025).

52. Shao, Z., Li, W. and Tan, Y., 2023. Event-triggered hybrid energy-aware scheduling in manufacturing systems. arXiv preprint, arXiv:2302.00812. Available at: https://arxiv.org/abs/2302.00812 (Accessed: 28 May 2025).
53. Siemens (2022) ‘MindSphere: The cloud-based, open IoT operating system’. Siemens AG. Available at: https://new.siemens.com/global/en/products/software/mindsphere.html (Accessed: 1 June 2025).
54. Silverman, B.W., 1986. Density Estimation for Statistics and Data Analysis. London: Chapman and Hall.
55. Tao, F., Qi, Q., Liu, A. and Kusiak, A., 2019. Digital twin and big data towards smart manufacturing and Industry 4.0: 360 degree comparison. IEEE Transactions on Industrial Informatics, 15(4), pp.2405–2415. Available at: https://doi.org/10.1109/TII.2018.2873186.
56. Tombari, F., Salti, S. and Di Stefano, L., 2010. Unique signatures of histograms for local surface description. In: European Conference on Computer Vision (ECCV), pp.356–369. Available at: https://doi.org/10.1007/978-3-642-15561-1_26 (Accessed: 28 May 2025).
57. Vozza, M., Polden, J., Mattera, G., Piscopo, G., Vespoli, S. and Nele, L., 2024. Explaining the anomaly detection in additive manufacturing via boosting models and frequency analysis. Mathematics, 12(21), 3414. Available at: https://doi.org/10.3390/math12213414 (Accessed: 28 May 2025).
58. Wang, Y. and Solomon, J.M., 2019. Deep closest point: Learning representations for point cloud registration. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp.3523–3532. Available at: https://doi.org/10.1109/ICCV.2019.00362 (Accessed: 28 May 2025).
59. Xie, C., Li, W., Xing, Y. and Jiao, W., 2024. A hybrid arm-hand rehabilitation robot with EMG-based admittance controller. IEEE Access, 12, pp.1–12. (Accessed: 28 May 2025).
60. Xu, R. and Tian, Y., 2015. A comprehensive survey of clustering algorithms. Annals of Data Science, 2(2), pp.165–193.
61. Yan, Z., 2018. Point cloud segmentation (clustering). Mobile Robotics Course, University of Technology of Compiègne. Available at: https://yzrobot.github.io/mobile_robotics/P20-UN56-clustering.pdf (Accessed: 28 May 2025).
62. Yildiz, M., et al., 2023. A review of the recent developments and challenges in wire arc additive manufacturing. Journal of Manufacturing and Materials Processing, 7(3), p.97. Available at: https://doi.org/10.3390/jmmp7030097 (Accessed: 28 May 2025).
63. Yokkampon, U., Mowshowitz, A., Chumkamon, S. and Hayashi, E., 2022. Robust unsupervised anomaly detection with variational autoencoder in multivariate time series

data. IEEE Access, 10, pp.57835–57847. Available at: https://doi.org/10.1109/ACCESS.2022.3178592 (Accessed: 28 May 2025).

64. Zhang, C., Yin, X. and Wang, Y., 2020. Geometry-based registration and segmentation for 3D point cloud data in robotic manufacturing. Robotics and Computer-Integrated Manufacturing, 63, p.101911.
65. Zhao, Y., Nasrullah, Z. and Li, Z., 2019. PyOD: A Python toolbox for scalable outlier detection. Journal of Machine Learning Research, 20(96), pp.1–7. Available at: https://jmlr.org/papers/volume20/19-011/19-011.pdf (Accessed: 28 May 2025).
66. Zhou, X., Yuan, Y., Wang, W. and Wang, H., 2021. Sensor fusion with deep learning for asynchronous multi-modal robot data synchronization. Sensors, 21(5), 1763. Available at: https://doi.org/10.3390/s21051763 (Accessed: 28 May 2025).
67. Zhou, Y., Han, Q., Wang, H., Liu, S. and Li, X., 2022. DeepMatch: Toward lightweight in point cloud registration. Frontiers in Neurorobotics, 16, 891158. Available at: https://doi.org/10.3389/fnbot.2022.891158 (Accessed: 28 May 2025).

ANNEXES

ANNEX A. Sample Input Data Structures

Excerpt from point cloud data (2025-04-15_17-43-22_points_694020.csv):

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 694020 entries, 0 to 694019
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   X       694020 non-null  float64
 1   Y       694020 non-null  float64
 2   Z       694020 non-null  float64
dtypes: float64(3)
memory usage: 15.9 MB
```

1 to 5 of 5 entries Filter

index	X	Y	Z
0	1885.72	323.853	-52.5201
1	1885.71	323.912	-52.5371
2	1885.74	323.992	-52.4593
3	1885.72	324.037	-52.5426
4	1885.72	324.096	-52.5586

Excerpt from structured scan file (LaserCoordinate1.csv):

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   TimeReceived  99 non-null     object
 1   X             99 non-null     object
 2   Z             99 non-null     object
 3   Intensity     99 non-null     object
dtypes: object(4)
memory usage: 3.2+ KB
```

	TimeReceived	X	Z	Intensity
0	2025-04-15 17:36:53.860	40.274 40.216 40.133 40.091 40.033 39.967 39.9...	151.380 151.399 151.321 151.408 151.426 151.41...	69 71 62 61 64 83 102 123 146 134 123 108 101 ...
1	2025-04-15 17:36:53.890	39.047 38.981 38.906 38.847 38.797 38.738 38.6...	146.508 146.490 146.435 146.444 146.489 146.49...	92 91 83 87 92 87 80 80 86 87 79 83 85 89 78 7...
2	2025-04-15 17:36:53.920	38.297 38.241 38.183 38.119 38.054 37.994 37.9...	143.511 143.528 143.537 143.519 143.501 143.50...	86 103 101 92 92 91 82 76 74 72 71 76 94 113 1...
3	2025-04-15 17:36:53.950	37.780 37.740 37.682 37.608 37.543 37.483 37.4...	141.438 141.514 141.522 141.463 141.437 141.43...	82 78 86 77 80 87 85 82 76 75 78 81 98 91 86 8...
4	2025-04-15 17:36:53.979	37.549 37.487 37.424 37.363 37.304 37.245 37.1...	140.510 140.501 140.484 140.476 140.476 140.47...	113 114 120 123 121 120 115 105 101 100 101 10...

Excerpt from process log (Log_Arc_at_15-Apr-2025_17-35.csv):

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3230 entries, 0 to 3229
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   time            3230 non-null   object
 1    x              3230 non-null   float64
 2    y              3230 non-null   float64
 3    z              3230 non-null   float64
 4    speed          3230 non-null   float64
 5    ArcOn          3230 non-null   int64
 6    WFS            3230 non-null   float64
 7    I              3230 non-null   float64
 8    U              3230 non-null   float64
 9    tool           3230 non-null   int64
 10   frame          3230 non-null   int64
 11   dz             3230 non-null   int64
 12   speedOverride  3230 non-null   int64
dtypes: float64(7), int64(5), object(1)
memory usage: 328.2+ KB
```

1 to 5 of 5 entries Filter

index	time	x	y	z	speed	ArcOn	WFS	I	U	tool	frame	dz	speedOverride
0	36:51.1	450.0	799.487	43.115	0.0	0	6.62	135.2	13.54	4	7	0	100
1	36:51.2	450.0	799.487	43.115	0.0	0	6.62	135.2	13.54	4	7	0	100
2	36:51.3	450.0	799.487	43.115	0.0	0	6.62	135.2	13.54	4	7	0	100
3	36:51.3	450.0	799.487	43.115	0.0	0	6.62	135.2	13.54	4	7	0	100
4	36:51.4	450.0	799.487	43.115	0.0	0	6.62	135.2	13.54	4	7	0	100

ANNEX B. Key Python Code Snippets

Method 1: Ground Truth

```
[ ]
def parse_floats(space_str):
    return [float(v) for v in space_str.strip().split()]

all_synced_points = []
point_index = 0

for layer_num, file_path in enumerate(laser_file_paths):
    laser_df = pd.read_csv(file_path)

    for _, row in laser_df.iterrows():
        x_vals = parse_floats(row["X"])
        count = len(x_vals)
        points_subset = points_df.iloc[point_index:point_index + count]

        if len(points_subset) != count:
            print(f"⚠️ Skipped a fragment in layer {layer_num}")
            continue

        for i in range(count):
            all_synced_points.append({
                'X': points_subset.iloc[i]["X"],
                'Y': points_subset.iloc[i]["Y"],
                'Z': points_subset.iloc[i]["Z"],
                'layer_id_1MethodGroundTruth': layer_num
            })

        point_index += count
```

Method 2: Cluster KMeans Known. Clustering with known number of layers:

```
[ ]
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans

# 📥 df_clean_roi is already created
df_input = df_clean_roi.copy()
n_clusters_local = 7

# 📌 Clustering along the Z-axis
z_vals = df_input[['Z']].values
kmeans = KMeans(n_clusters=n_clusters_local, random_state=42)
df_input['layer_id_2MethodKmean7'] = kmeans.fit_predict(z_vals)

# 🔢 Renumbering layers from bottom to top
centers = kmeans.cluster_centers_.flatten()
sorted_idx = np.argsort(centers)
mapping = {old: new for new, old in enumerate(sorted_idx)}
df_input['layer_id_2MethodKmean7'] = df_input['layer_id_2MethodKmean7'].map(mapping)

# 📊 Statistics by layer
print("📊 Number of points in each layer:")
for lid in sorted(df_input['layer_id_2MethodKmean7'].unique()):
    count = (df_input['layer_id_2MethodKmean7'] == lid).sum()
    print(f"  - Layer {lid}: {count} points")

print(f"\n✅ Total number of points: {len(df_input)}")
```

Method 3: Cluster KMeans Auto. Clustering with unknown number of layers:

```
[ ]
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.neighbors import KernelDensity
from scipy.signal import find_peaks

# 📥 df_clean_roi is already created
df_input = df_clean_roi.copy()

# 📈 Determine number of clusters using KDE + peak detection
z_vals = df_input[['Z']].values
kde = KernelDensity(kernel='gaussian', bandwidth=0.5).fit(z_vals)

z_grid = np.linspace(z_vals.min(), z_vals.max(), 1000).reshape(-1, 1)
log_dens = kde.score_samples(z_grid)
density = np.exp(log_dens)

peaks, _ = find_peaks(density, distance=10, prominence=0.01)
n_clusters_local = len(peaks)

print(f"🔎 Automatically detected number of layers: {n_clusters_local}")

# 📌 Clustering along the Z-axis
kmeans = KMeans(n_clusters=n_clusters_local, random_state=42)
df_input['layer_id_1MethodKmeanKernelDensit'] = kmeans.fit_predict(z_vals)

# 🔢 Renumber layers from bottom to top, starting from 1
centers = kmeans.cluster_centers_.flatten()
sorted_idx = np.argsort(centers)
mapping = {old: new + 1 for new, old in enumerate(sorted_idx)}  # 👈 shift by +1
df_input['layer_id_1MethodKmeanKernelDensit'] = df_input['layer_id_1MethodKmeanKernelDensit'].map(mapping)

# 📊 Layer statistics
print("📊 Number of points in each layer:")
for lid in sorted(df_input['layer_id_1MethodKmeanKernelDensit'].unique()):
    count = (df_input['layer_id_1MethodKmeanKernelDensit'] == lid).sum()
    print(f"  - Layer {lid}: {count} points")

print(f"\n✅ Total number of points: {len(df_input)}")
```

Method 4: Cluster DBSCAN Auto. DBSCAN clustering by Z:

```
[ ]
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN

df_input = df_clean_roi.copy()

# 📌 Clustering by Z-axis
z_vals = df_input[['Z']].values
db = DBSCAN(eps=0.35, min_samples=30)
df_input['layer_id_3MethodDBSCAN'] = db.fit_predict(z_vals)

# ❌ Remove noise points
df_input = df_input[df_input['layer_id_3MethodDBSCAN'] != -1].copy()

# 🔢 Renumber layers starting from 1 using median Z
centers = df_input.groupby('layer_id_3MethodDBSCAN')['Z'].median()
sorted_idx = np.argsort(centers)
mapping = {old: new + 1 for new, old in enumerate(sorted_idx)}  # 👈 start from 1
df_input['layer_id_3MethodDBSCAN'] = df_input['layer_id_3MethodDBSCAN'].map(mapping)

# 📊 Layer statistics
print("📊 Number of points in each layer:")
for lid in sorted(df_input['layer_id_3MethodDBSCAN'].unique()):
    count = (df_input['layer_id_3MethodDBSCAN'] == lid).sum()
    print(f"  - Layer {lid}: {count} points")

print(f"\n✅ Total number of points after DBSCAN: {len(df_input)}")
```

Method 5: Geometry Based. Geometry-based segmentation (distance jumps):

```
[ ]
# Calculate Euclidean distances
deltas = np.sqrt(np.diff(df['X'])**2 + np.diff(df['Y'])**2 + np.diff(df['Z'])**2)
top_jumps = np.argpartition(deltas, -6)[-6:]  # Find indices of the 6 largest jumps
top_jumps = np.sort(top_jumps + 1)  # Shift by 1 to get split positions
scan_boundaries = [0] + top_jumps.tolist() + [len(df)]  # Include start and end
scans = [df.iloc[scan_boundaries[i]:scan_boundaries[i+1]].copy() for i in range(len(scan_boundaries) - 1)]

# Manually split scan 3 into two parts
scan3 = scans[2].reset_index(drop=True)
deltas_scan3 = np.sqrt(np.diff(scan3['X'])**2 + np.diff(scan3['Y'])**2 + np.diff(scan3['Z'])**2)
split_index = np.argmax(deltas_scan3) + 1  # Find largest jump
scans_cleaned = scans[:2] + [scan3.iloc[:split_index], scan3.iloc[split_index:]] + scans[3:6]

# Assign scan_id
df_labeled = df.copy()
df_labeled['scan_id'] = -1
start_idx = 0
for i, scan in enumerate(scans_cleaned):
    end_idx = start_idx + len(scan)
    df_labeled.loc[start_idx:end_idx-1, 'scan_id'] = i
    start_idx = end_idx
```

Method 6: Hierarchical Auto

```
[ ]
import numpy as np
import pandas as pd
from sklearn.cluster import AgglomerativeClustering

# 🔥 Assumes that df_clean_roi is already created
df_input = df_clean_roi.copy()

# 📌 Clustering by Z-axis with automatic number of clusters
z_vals = df_input[['Z']].values

# ➕ Set the maximum distance between points to be grouped into one cluster
#     Instead of n_clusters, we use distance_threshold
#     linkage='ward' is suitable for numerical data
hc = AgglomerativeClustering(
    n_clusters=None,
    distance_threshold=50,   # manually chosen or based on dendrogram
    linkage='ward'
)

# 📊 Perform clustering
df_input['layer_id_6MethodHierarchicalAuto'] = hc.fit_predict(z_vals)

# 🔃 Renumber layers from bottom to top
centers = df_input.groupby('layer_id_6MethodHierarchicalAuto')['Z'].mean()
sorted_idx = np.argsort(centers.values)
mapping = {old: new for new, old in enumerate(centers.index[sorted_idx])}
df_input['layer_id_6MethodHierarchicalAuto'] = df_input['layer_id_6MethodHierarchicalAuto'].map(mapping)

# 🔢 Layer statistics
print("📊 Number of points in each layer:")
for lid in sorted(df_input['layer_id_6MethodHierarchicalAuto'].unique()):
    count = (df_input['layer_id_6MethodHierarchicalAuto'] == lid).sum()
    print(f"  - Layer {lid}: {count} points")

print(f"\n✅ Total number of points: {len(df_input)}")
print(f"🔢 Number of detected layers: {df_input['layer_id_6MethodHierarchicalAuto'].nunique()}")
```

Anomaly detection:

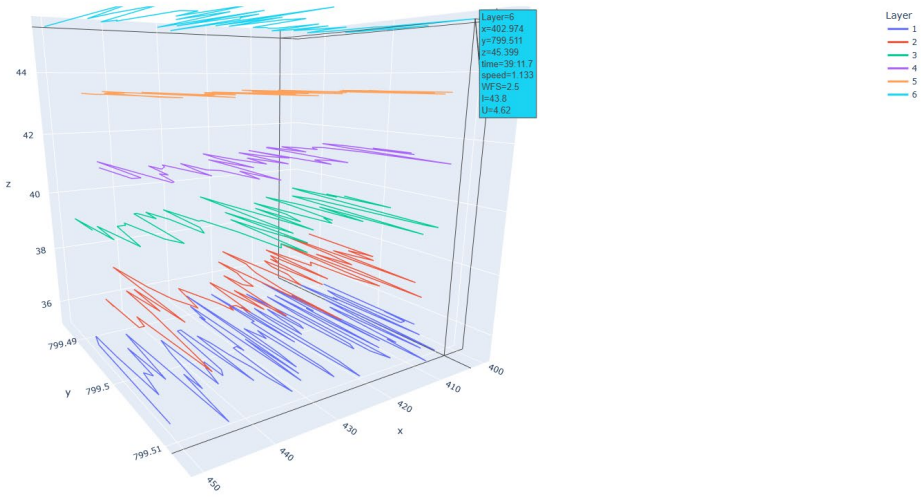
```
[ ]
# ⚙️ 4. Z-based anomaly detection with neighbors (trimmed)
threshold_z_drop = 1.0
layer_ids = sorted(df_all['layer_id'].unique())
trimmed_anomaly_points = []

for i in range(1, len(layer_ids) - 1):
    lid_prev, lid_curr, lid_next = layer_ids[i - 1], layer_ids[i], layer_ids[i + 1]
    df_prev = df_all[df_all['layer_id'] == lid_prev]
    df_curr = df_all[df_all['layer_id'] == lid_curr]
    df_next = df_all[df_all['layer_id'] == lid_next]

    common_arc_ids = set(df_curr['arc_id']).intersection(df_prev['arc_id'], df_next['arc_id'])

    for arc_id in common_arc_ids:
        curr_arc = df_curr[df_curr['arc_id'] == arc_id].sort_values(by='X')
        prev_arc = df_prev[df_prev['arc_id'] == arc_id].sort_values(by='X')
        next_arc = df_next[df_next['arc_id'] == arc_id].sort_values(by='X')

        def trim(df_arc):
            n = len(df_arc)
            if n < 10:
                return df_arc
            return df_arc.iloc[int(n * 0.05): int(n * 0.95)]

        trimmed_curr = trim(curr_arc)
        z_curr = trimmed_curr['Z'].mean()
        z_prev = trim(prev_arc)['Z'].mean()
        z_next = trim(next_arc)['Z'].mean()

        z_neighbors_avg = (z_prev + z_next) / 2

        if (z_neighbors_avg - z_curr) > threshold_z_drop:
            trimmed_anomaly_points.append(trimmed_curr)
```

Clustering by Z using DBSCAN

This script segments WAAM process log data into individual deposition layers by clustering active deposition points (ArcOn == 1) along the Z-axis using the DBSCAN algorithm.

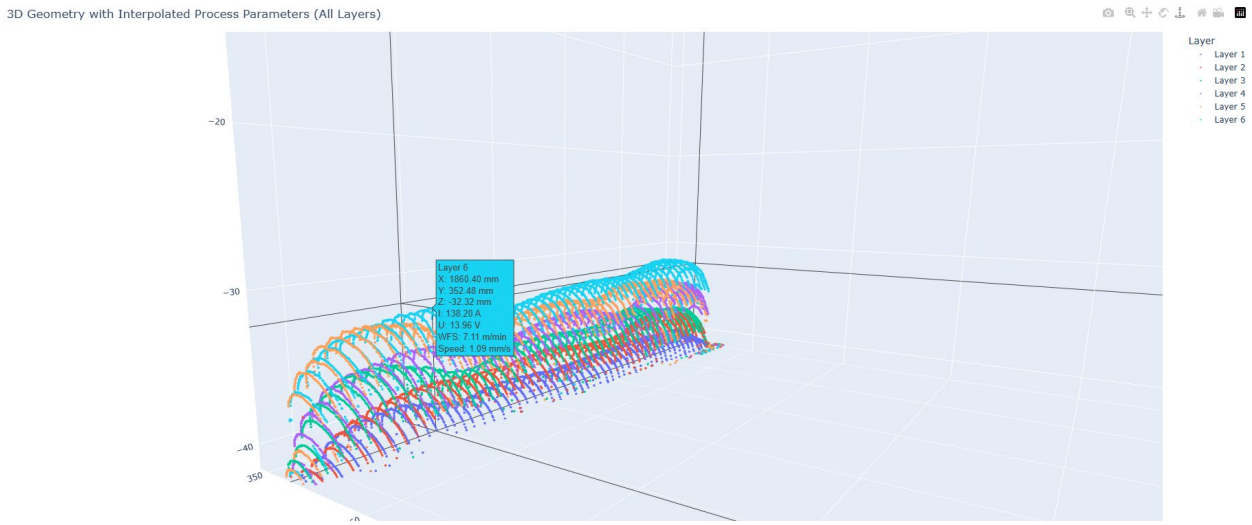
```
[ ]
# Filter active segments (ArcOn)
# Keep only rows where ArcOn == 1 (active deposition)
df = df[df['ArcOn'] == 1].copy()

# Select relevant columns
columns_to_keep = ['time', 'x', 'y', 'z', 'speed', 'WFS', 'I', 'U']
df = df[columns_to_keep]

# DBSCAN clustering by Z
# Apply DBSCAN clustering based on Z coordinate
z_values = df[['z']].values
db = DBSCAN(eps=0.5, min_samples=5).fit(z_values)

# Assign layer IDs
df['layer_id'] = db.labels_

# Remove noise points
# Remove noise points (label -1)
df_clean = df[df['layer_id'] != -1].copy()

# ✅ Shift all labels by +1 (to start from 1 instead of 0)
df_clean['layer_id'] = df_clean['layer_id'] + 1

# Convert layer_id to string for color grouping
df_clean['layer_id'] = df_clean['layer_id'].astype(int)
```

WAAM Process Layers (Clustered by Z via DBSCAN)



Process Data Integration

```
[ ]
# === 2. Align X by layer center ===
df_proc_bounds = df_process.groupby('layer_id')['x'].agg(min_x_proc='min', max_x_proc='max')
df_geom_bounds = df_geometry.groupby('layer_id')['X'].agg(min_x_geom='min', max_x_geom='max')
df_align = pd.merge(df_proc_bounds, df_geom_bounds, left_index=True, right_index=True)
df_align['center_proc'] = (df_align['min_x_proc'] + df_align['max_x_proc']) / 2
df_align['center_geom'] = (df_align['min_x_geom'] + df_align['max_x_geom']) / 2
df_align['shift'] = df_align['center_geom'] - df_align['center_proc']
shift_map = df_align['shift'].to_dict()

df_process['x_aligned'] = df_process.apply(
    lambda row: row['x'] + shift_map.get(row['layer_id'], 0),
    axis=1
)

# === 3. Interpolate process parameters ===
skipped_layers = []

def transfer_process_to_geometry_interpolated(layer_id, df_process, df_geometry):
    proc = df_process[df_process['layer_id'] == layer_id].sort_values('x_aligned')
    geom = df_geometry[df_geometry['layer_id_5MethodGeometryBased'] == layer_id].sort_values('X')

    if len(proc) < 2 or len(geom) < 2 or proc['x_aligned'].nunique() < 2:
        skipped_layers.append(layer_id)
        print(f"[!] Skipped layer {layer_id} — not enough unique points")
        return pd.DataFrame()

    interpolated = geom.copy()
    for col in ['I', 'U', 'WFS', 'speed']:
        interp_func = interp1d(proc['x_aligned'], proc[col], bounds_error=False, fill_value="extrapolate")
        interpolated[f'proc_{col}'] = interp_func(geom['X'].values)

    interpolated['matched_layer_id'] = layer_id
    return interpolated

common_layers = sorted(set(df_process['layer_id']) & set(df_geometry['layer_id_5MethodGeometryBased']))
geom_interp_all = pd.concat([
    transfer_process_to_geometry_interpolated(lid, df_process, df_geometry)
    for lid in common_layers
], ignore_index=True)
```



ANNEX C. Evaluation Script Structure

Matching predictions with ground truth:

```
[ ]
# === 2. Function to compare predictions by coordinates ===
def compare_by_coordinates(df_pred, df_gt, col_pred, col_gt="layer_id_1MethodGroundTruth"):
    merged = pd.merge(
        df_gt[['X', 'Y', 'Z', col_gt]],
        df_pred[['X', 'Y', 'Z', col_pred]],
        on=['X', 'Y', 'Z'], how='inner'
    )

    y_true = merged[col_gt]
    y_pred = merged[col_pred]

    return {
        "Accuracy": accuracy_score(y_true, y_pred),
        "Precision (macro)": precision_score(y_true, y_pred, average='macro', zero_division=0),
        "Recall (macro)": recall_score(y_true, y_pred, average='macro', zero_division=0),
        "F1-score (macro)": f1_score(y_true, y_pred, average='macro', zero_division=0),
        "IoU (macro)": jaccard_score(y_true, y_pred, average='macro', zero_division=0)
    }

# === 3. Compare all methods ===
results = []
methods = [
    ("2Method_KMeansKnown", df_kmeans_known, df_kmeans_known.filter(like="layer_id").columns[0]),
    ("3Method_KMeansAuto", df_kmeans_auto, df_kmeans_auto.filter(like="layer_id").columns[0]),
    ("4Method_DBSCAN", df_dbscan, df_dbscan.filter(like="layer_id").columns[0]),
    ("5Method_Geometry", df_geometry, df_geometry.filter(like="layer_id").columns[0]),
    ("6Method_HierarchicalAuto", df_hierarchical, df_hierarchical.filter(like="layer_id").columns[0]),
]

for method_name, df_method, col_name in methods:
    scores = compare_by_coordinates(df_method, df_gt, col_name)
    scores["Method"] = method_name
    results.append(scores)
```

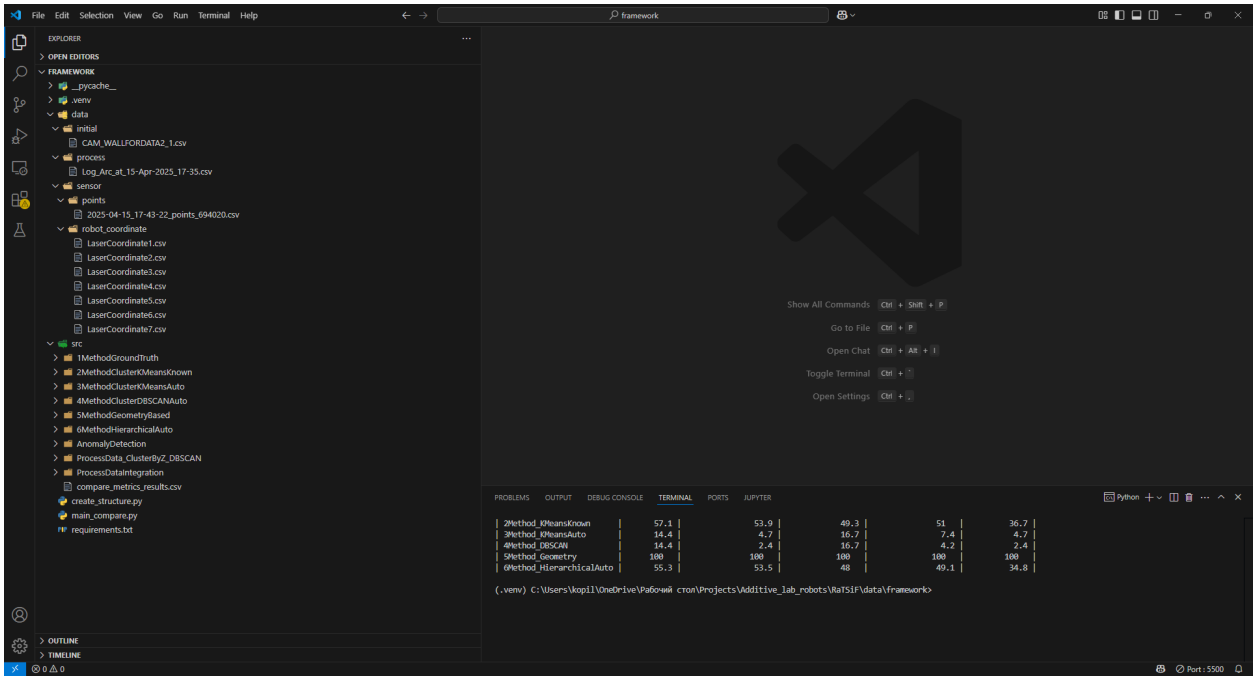
ANNEX D. Example of Enriched Output Table

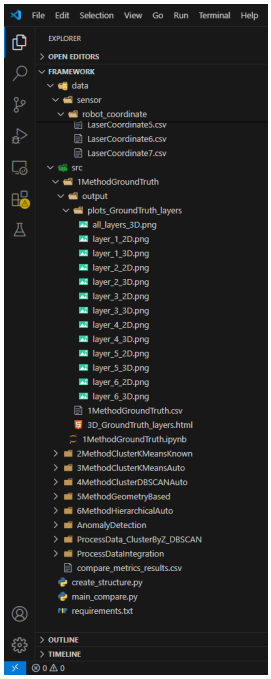
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22965 entries, 0 to 22964
Data columns (total 14 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   X                                22965 non-null  float64
 1   Y                                22965 non-null  float64
 2   Z                                22965 non-null  float64
 3   layer_id_5MethodGeometryBased    22965 non-null  int64
 4   layer_id                         22965 non-null  int64
 5   proc_I                           22780 non-null  float64
 6   proc_U                           22780 non-null  float64
 7   proc_WFS                         22780 non-null  float64
 8   proc_speed                       22553 non-null  float64
 9   matched_layer_id                 22965 non-null  int64
 10  arc_id                           22965 non-null  int64
 11  info                             22965 non-null  object
 12  Anomalies                        22965 non-null  bool
 13  layer_id_mapped                  22965 non-null  int64
dtypes: bool(1), float64(7), int64(5), object(1)
memory usage: 2.3+ MB
```

726 to 750 of 1000 entries Filter

index	X	Y	Z	layer_id_5MethodGeometryBased	layer_id	proc_I	proc_U	proc_WFS	proc_speed	matched_layer_id	arc_id	info	Anomalies	layer_id_mapped
725	1830.72	353.394	-39.1432	1	1	139.2	12.67	5.87	0.5763781021897817	1	13	Layer: 1 Arc ID: 13 I: 139.2 U: 12.67 WFS: 5.87 Speed: 0.58 X=1830.72 Y=353.39 Z=-39.14	false	1
726	1830.72	353.45	-39.166	1	1	139.2	12.67	5.87	0.5763781021897817	1	13	Layer: 1 Arc ID: 13 I: 139.2 U: 12.67 WFS: 5.87 Speed: 0.58 X=1830.72 Y=353.45 Z=-39.17	false	1
727	1830.72	353.677	-39.1855	1	1	139.2	12.67	5.87	0.5763781021897817	1	13	Layer: 1 Arc ID: 13 I: 139.2 U: 12.67 WFS: 5.87 Speed: 0.58 X=1830.72 Y=353.68 Z=-39.19	false	1
728	1830.72	353.225	-39.1068	1	1	139.2	12.67	5.87	0.5763781021897817	1	13	Layer: 1 Arc ID: 13 I: 139.2 U: 12.67 WFS: 5.87 Speed: 0.58 X=1830.72 Y=353.22 Z=-39.11	false	1
729	1830.72	353.508	-39.1568	1	1	139.2	12.67	5.87	0.5763781021897817	1	13	Layer: 1 Arc ID: 13 I: 139.2 U: 12.67 WFS: 5.87 Speed: 0.58 X=1830.72 Y=353.51 Z=-39.16	false	1

ANNEX E. List of Files in Project Repository





ANNEX F. Supplementary Online Resources

1. Zotero Library – TSI MA Research Group

This shared library contains the curated collection of scientific literature and references used during the research process.

https://www.zotero.org/groups/6010913/tsi_ma_research

2. GitHub Repository – WAAM Sync & Anomaly Framework

The complete codebase, data processing scripts, and implementation of the developed synchronization and anomaly detection framework are available in the following public GitHub repository:

https://github.com/SergejsKopils/waam-sync-anomaly-framework