



An Automated Approach for Segmenting Numerical Control Data With Controller Data for Machine Tools

Laetitia Monnier

Burgundy Computer Laboratory,
University of Burgundy Franche Comte,
Besancon 25000, France
e-mail: laetitia.monnier@nist.gov

William Z. Bernstein¹

Manufacturing and Industrial Technologies
Division,
Air Force Research Laboratory,
Wright-Patterson AFB, OH 45433
e-mail: william.bernstein@us.af.mil

Vincenzo J. Ferrero

System Integration Division,
National Institute of Standards and Technology,
Gaithersburg, MD 20899
e-mail: vincenzo.ferrero@nist.gov

Sebti Foufou

Computer Science, College of Computing and
Informatics,
University of Sharjah,
Sharjah 27272, United Arab Emirates
e-mail: sfoufou@sharjah.ac.ae

Developing a more automated industrial digital thread is vital to realize the smart manufacturing and industry 4.0 vision. The digital thread allows for efficient sharing across product lifecycle stages. Current techniques are not robust in relating downstream data, such as manufacturing and inspection information, back to design for better decision making. We previously presented a methodology that aligns numerical control (NC) code, a standard for representing machine tool instructions, to controller data represented in MTConnect, a standard that provides a vocabulary for generalizing execution logs from different machine tools and devices. This paper extends our previous work by automating the tool identification using a k-means clustering algorithm to refine the alignment of the data. In doing so, we compare different distance techniques to analyze the spatial-temporal registration of the two datasets, i.e., the NC code and MTConnect data. Then, we assess the efficiency of our method through an error measurement technique that expresses the quality of the alignment. Finally, we apply our methodology to a case study that includes verified process plans and real execution data, derived from the smart manufacturing systems test bed hosted at the National Institute of Standards and Technology. Our analysis shows that dynamic time warping achieves the best point registration with the least errors compared with other alignment techniques. [DOI: 10.1115/1.4064036]

Keywords: computational foundations for engineering optimization, computer-aided design, computer-aided engineering, computer-aided manufacturing, data-driven engineering, engineering informatics, knowledge engineering

1 Introduction

Reports show that on average, 60–73% of enterprise data remain unused [1]. Automated methods can help to relate disparate data in order to infer insights and knowledge from it [2]. Stark [3] defined product lifecycle (PL) management from the design stage (“as-designed”), process planning stage (“as-planned”), manufacturing execution stage (“as-executed”), and the inspection stage (“as-inspected”). Practitioners have developed tools and methods for each PL stage “in silo” to address domain-specific needs [4]. Within the *design* silo, data representations primarily store geometric dimensioning and tolerancing (GD&T) information. In the *manufacturing execution* silo, data representations capture information from device controllers. Current methods are mostly using manual techniques that do not enable automated integration of these data representations. To facilitate such integration, many SDOs, industrial consortia, and academia have embraced the *digital thread* paradigm [5]. Hedberg et al. [6] reported that a

“need exists for information standards that derive requirements to facilitate upstream and downstream flows in the product lifecycle, [and] data-format standards are not enough.” In other words, realizing the digital thread is recognized to be a grand challenge for manufacturing enterprises [7,8].

Efforts have been made to add a layer of descriptive semantics to promote interoperability through formal standards such as MTConnect [9] and through ontologies and knowledge graphs [10–12]. However, the adoption of such semantics is not a given. For example, International Organization of Standardization (ISO) 10303 application protocol (AP) 238 [13], used in the “as-designed” stage, defines a detailed data model that includes semantic tags and descriptions for CNC machine instructions used in the “as-planned” stage, such as tags for specific machining features. Still, the adoption of such a standard with enriched semantics remains an open question. Previous work [14] revealed opportunities for making informed design decisions by manually mapping data between lifecycle representations. However, aligning the PL’s different data standards in an automated way from downstream stages, e.g., inspection and production, to upstream stages, e.g., design, is not trivial. The manufacturing data remain mostly unused as its curation remains disjointed and unavailable across enterprises. Considering this issue, mapping methods must be able to operate on a small set of training data preventing the use of traditional big data

¹Corresponding author.

Manuscript received March 25, 2022; final manuscript received November 2, 2023; published online November 24, 2023. Assoc. Editor: Gaurav Ameta.

This work is in part a work of the U.S. Government. ASME disclaims all interest in the U.S. Government’s contributions.

technologies. West and Blackburn [15] concluded that achieving a “digital thread” can be cost-prohibitive considering its full implementation. Thus, emerging PL data mapping techniques must address scalability [16].

This paper proposes an automated approach for aligning data between the “as-planned” and “as-executed” stages. We assume that “as-planned” and “as-executed” information is captured as numerical control (NC) code and pipe-delimited controller data, respectively. We leverage TDPs hosted on the smart manufacturing systems (SMS) test bed² developed at the National Institute of Standards and Technology (NIST) to validate our method. Our research contributions include the following:

- (1) A methodology for aligning “as-planned” and “as-executed” data through spatial-temporal registration;
- (2) A technique for quantifying the error of aligning the two coordinate systems;
- (3) A clustering-based approach for automatically identifying tool changes from execution data;
- (4) A real-world case study comparing different approaches for spatial-temporal registration; and
- (5) Recommendations for automated post-hoc mapping between machine instructions and execution data.

2 Product Lifecycle Data Representations

2.1 Design Phase Data: As-Designed. The design stage defines the product “as-designed” before its production. At this stage, all design-specific information and knowledge become fixed. Such information includes GD&T specifications, manufacturing process requirements, and assembly considerations. Traditionally, designers leverage commercial computer-aided design (CAD) tools to develop constraint-based geometric features. ISO 10303 AP 242 [17] was developed as a technology-agnostic design representation that enables import/export of design information from a variety of CAD tools. ISO 10303 is often referred to as the standard for the exchange of product data (STEP). Abiding by an EXPRESS schema, STEP AP242 has been leveraged across other PL stages [18].

2.2 Manufacturing Phase Data. The fabrication stage is separated into two different phases which consist of the process planning of the part or assembly and of the actual execution of the processes. Those two phases focus on the operation and execution of the manufacturing resources necessary to carry out the planned processes. As shown in Fig. 1, we can refer to these two phases as as-planned and as-executed. Though the planning and execution data relate to one another, these data are often mismatched due to inconsistencies in the execution of the manufacturing resources. Causes of such inconsistencies could include overrides from operators, acceleration ramp-ups, and axis motor errors.

The as-planned stage includes detailed process plans, including machine-execution instructions required to build the part. As-planned data are derived from the as-designed stage but might vary depending on the manufacturing strategy and machine setups adopted to build the part. NC code [19] is the most widely used data format for expressing machine instructions. Recently, ISO TC184 SC4 developed STEP-NC (or STEP AP238 or ISO 10303–238), which can segment NC instructions based on individual design and/or machining features [20]. In other words, STEP-NC aims to provide computer-interpretable semantics to NC code, which is further explained in Sec. 3.1.

The as-executed stage refers to the actualization of the physical part through the execution of the manufacturing resources. Such considerations describe the tool behaviors during execution activities. MTConnect [9], an emerging standard for semantic interoperability that also offers a read-only communication protocol from shop-floor systems, can be used to capture execution data.

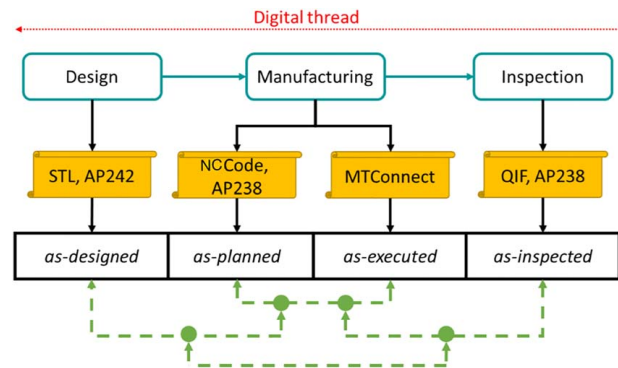


Fig. 1 Proposed mappings of the data between the different PL stages. Each dotted line represents a mapping between two stages and the circle shows the resulting mapped data shared throughout the PL. Our goal is to realize the digital thread.

2.3 Inspection Phase Data: As-Inspected. The inspection stage generates as-inspected data that relate to quality-based measurements, including the plans of how to perform the measurement and the analysis of the actual results of those measurements. One standard data representation that helps systematically store and exchange inspection data is the quality information framework (QIF) standard [21]. QIF provides the semantics to link data upstream to design and manufacturing.

Figure 1 shows each stage of the product lifecycle. The main challenge is aligning, mapping, and registering different data to inform decision-making.

3 Related Work

3.1 Semantic Extension Through Data Standards. Since each PL stage consumes/generates different data and leverages its own representations (see Sec. 2), information is often lost when converting from one data standard to another. For example, NC code generated through computer-aided manufacturing (CAM) software uses the CAD model as its basis. However, the resulting NC code is based on a discretized set of motions which does not reflect the geometry properties embedded in the part’s design.

To prevent such loss of information, previous research adds semantic descriptions to the existing data workflow. Adding specific information to the data standards will facilitate mapping data across different stages. In a similar vein, companion specifications have also been developed to provide consistency and encode relationships between different data standards. For example, the OPC UA—MTConnect companion specification [22] ensures interoperability for messaging on the shop floor. Another example is the mapping specification between STEP and QIF, which remains a work-in-progress [23]. Other directions focused on directly adding information onto those data standards. STEP AP238 has been defined as an enhanced NC code by adding machining requirements traditionally embedded in design data [13]. AP238 aims to broaden the data contained within the NC file, which allows for transferring of information from the as-designed stage toward the as-planned stage. Extending standards by adding data or semantics to them help only locally to partially realize the digital thread. In other words, solutions are developed for one standard to another but do not take into account the entire PL, as defined in Fig. 1.

3.2 Product Lifecycle Data Semantical Description Using Ontologies. Model-based methods have also been explored toward the idea of describing the different data types used in the PL stages to establish both implicit and explicit relationships. This approach incorporates an abstract layer on top of the existing data types, e.g., ontologies and reasoners. OntoSTEP [10] provides tools to translate STEP AP203 schema into an ontology that can be

²All data are available here: <https://smstestbed.nist.gov/tdp/d2mi/>

used with CAD software. Danjou et al. [24] developed an ontology describing the STEP AP238 data model which creates a semantic representation of NC code. Here, *local* ontologies are created to address individual PL stages and domain-specific needs [5]. Kwon et al. [11] tackled this problem by developing a methodology to automatically translate QIF into an ontology that can then be linked with the OntoSTEP [10] ontology of AP203.

The optimal solution would be to have a single upper ontology across the PL stages, including design, manufacturing, and inspection. However, there are different perspectives across the PL and even many more across the different manufacturing domains, making it challenging to validate a unique ontology. Those constraints are primary barriers to the development of a true ontology-based digital thread.

3.3 Virtualization Approaches. Since no robust solution to transfer data through PL stages has been demonstrated, data-driven applications have emerged that use simulations to deduce information without having to link the PL stages data standards. VERICUT³ is a commercial CAM software package that simulates the execution of machine operations from NC code. VERICUT allows users the detection of errors in the machining process by visualizing the path execution of the machine tool. The software contains data about existing machine tools and sequentially runs NC command from the as-planned stage to infer the as-executed output.

However, one should note that those simulations show significant differences between the simulated data and the actual measured data. Feng et al. [25] highlighted different cycle time between the simulated and measured build of a part. The actual build time was extended in specific areas where the operator chose to override the planned machining process in order to ensure the quality of the part. The root causes of these differences depend on multiple real-life factors [4]. Mapping techniques aiming to retrieve the measured data and relate it to the expected execution data could address such challenges.

3.4 Motivations. Past work (presented throughout Secs. 2 and 3) essentially focused on developing the “digital thread” to solve a specific manufacturing problem. To solve this challenge through semantically enriching native data structures, many changes are necessary in industry (both end users and technology providers). However, machines and software already in use may not adopt new standards. Upgrading manufacturing assets, especially legacy equipment, may not be feasible in many cases. Hence, efforts that leverage existing equipment and software already in place are still required. Another challenge lies in the expertise for manufacturing end users. Usually, an expert must be trained to acquire sufficient experience with the technology developed to follow data-intensive processes. Then, a verification stage is mandatory to check whether results derived from data capture are meaningful, appropriate, and accurate. Finally, the data need to be analyzed by human experts to infer knowledge.

Our work aims to develop a method that focuses only on transferring information from downstream PL stages, e.g., inspection and manufacturing stages, toward upstream PL stages, e.g., design stage. The end vision will be to leverage the data produced throughout the PL to provide insights and guidance to PL stakeholders, each with unique perspectives. This paper presents a “black box” methodology for mapping between as-planned and as-executed data. Moving forward this alignment will be leveraged toward the other PL stages, e.g., “as-designed” and “as-inspected” [11]. In this paper, we analyze the spatial-temporal registration issues between execution data and machine instructions. Figure 2 illustrates the goal of this paper. We propose a technique for segmenting and aligning “as-executed” data in the form of MTConnect with “as-planned” data in the form of NC code. The outcome of our

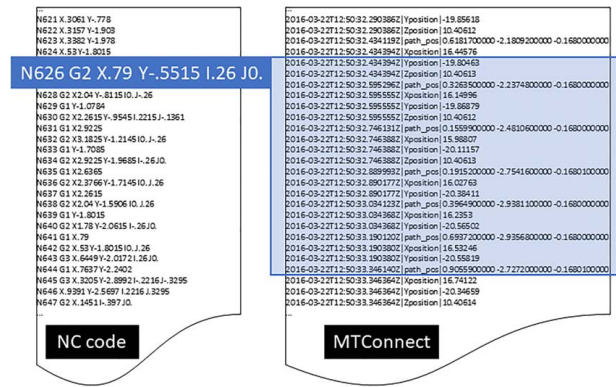


Fig. 2 Schematic representing the goal of this work. Per each NC code line, the corresponding MTConnect snippet is identified. Adopted from our previous work [26].

mapping technique is a MTConnect-based segments labeled by the NC code line that led to its creation. The spatial and temporal nature of both data formats, i.e., NC code and MTConnect, justify a close look at existing spatial-temporal registration techniques. In the next section, we review such methods.

4 Spatial-Temporal Registration

Beyond the manufacturing domain, this work broadly relates to spatial and temporal alignment techniques. Figure 3 illustrates our goal to align the NC Code and MTConnect data by registering the two dataset points based on their coordinates and following their timeline. Our method relies on using existing alignment techniques to extend this work and apply it to the PL interoperability challenge.

4.1 Triangular Inequality Measures (Euclidean Measure).

The triangular inequality measures are distance measurements that follow the “straight-line” distance measure paradigm [27], e.g., the shortest distance between the points in Euclidean space. The Euclidean distance is the main distance measurement used in geometry to compute a “regular” distance between different spatial objects. This distance measure has been often applied for registering and indexing time-series data. Lin et al. [28] focused on mapping similar time-series curves by comparing the points in each curves using an Euclidean distance measure. The “insignificant noises,” e.g., the discrepancies in the curves, are first removed to improve the comparison of similarity, by computing the Euclidean distance, of the different curve sequence patterns.

Keogh and Pazzani [29] demonstrated that coupling the notion of weighting features to the regular Euclidean distance measurement can increase the time-series classification accuracy. They pushed this work further by applying this upgraded distance, weighted Euclidean metrics, to identify the most similar curve sequences in

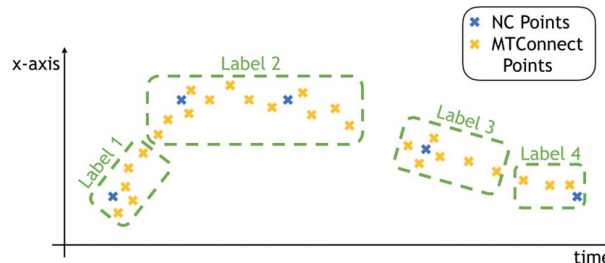


Fig. 3 Representation of our goal in aligning the NC Code data and MTConnect data based on spatial and temporal comparison

³<http://www.cgtech.com/>

a dataset [30]. Manually applied weights are attributed to the features forming the curves in the dataset. The “significant” features, i.e., “picks” on the curves, have higher weight than the more common features such as “lines.” The Euclidean distance is then balanced by these weights which gives a more flexible measure that allows a better registration of the similarities between the curve patterns.

4.2 Iterative Techniques. Using a Euclidean-based measure is a common approach to analyzing similar spatial datasets in alignment, registration, and indexing. To improve its efficacy, Besl and McKay [31] coupled the Euclidean distance measure with an iterative algorithm. The implementation created a new way of correlating similar points in a spatial dataset. The iterative closest point (ICP) algorithm uses a “triangular” measure (Sec. 4.1), such as the Euclidean distance measurement, to identify the closest point of the shape of the resulting object. Each iteration adds more points considered “closest to the object shape” until a stable shape is found. Bearee et al. [32] applied this method to an Aircraft deburring problem. The ICP allowed the convergence of the sensor point alignment to identify and match the burring contour on an aircraft nose. Chen and Medioni [33] pushed this method further. Instead of using a squared distance measure, a tangent plane is created from the point to match. The goal is to minimize the distance between this tangent plane and the other *closest* point.

The issue with ICP is that it does not handle temporal data well. The algorithm focuses on aligning multiple datasets to identify a contour shape, but the time sequences are lost in the process. Note that we deploy the tangent plane idea from Chen and Medioni [33] as a model to derive an error measurement in order to validate the alignment of the “as-planned” and “as-executed” datasets.

4.3 Sliding Windows Comparison. Time-series datasets have very specific characteristics. In a time series, data are correlated to the sequence in which the data appears. Faloutsos et al. [34] took this notion into account by introducing a sliding window that allows to detect patterns based on sequence. The window scans by comparing the average Euclidean distance between the points in the sliding window and the pattern found in the dataset. Then, this matching result is stored for each iteration in a R-Tree allowing the best results to be selected given the final similarity indexing. This method does not scale well with larger datasets. Since the R-Tree is used for storage, there exists a time limitation. Rafiei and Mendelzon [35] managed to lighten the previous algorithm by using linear transformations. The patterns are compared following the sliding window method and for each iteration a linear transformation is applied. A Euclidean distance is used to compare the similarities of the transformed curves and the searched patterns. This method shows a faster performance than the regular scanning but is an approximate matching and cannot guarantee an exact matching.

The issue that we found here is that it focuses on matching similar patterns that have been identified prior to the execution. This method could miss identifying important features that could help the alignment of the data. Moreover, we assume that we do not have technical insights (a priori) on the datasets. Thus, our goal is to have a naive approach on the data so that the algorithm can identify and align potential patterns in an automated way.

4.4 Dynamic Time Warping. Dynamic time warping (DTW) is a method aiming to serve as a distance measurement to align similar datasets following an underlying time sequence. Sakoe and Chiba [36] introduced DTW for use in speech recognition. The goal is to merge the sliding window idea (defined in Sec. 4.3) with an iterative algorithm (defined in Sec. 4.2) that will measure a standard distance, e.g., Euclidean distance (defined in Sec. 4.1), between the points in the dataset. Following the timeline,

the distance is computed between the points and based on the results the corresponding points are added to the sliding window attributed to a specific portion of the respective dataset. Each iteration refines the different windows identified until a stable alignment is found. Itakura [37] validated this method by applying it again with speech recognition.

The advantage of using DTW compared to regular distance measures is that it can identify the time distortion that may occur between the compared datasets. Chu et al. [38] explains that distance measures such as Euclidean approaches struggle to match time series when time distortions come into play. The alignment method should take into account the “elastic shifting of the X-axis” (see Fig. 2) which is not taken into account with exact matching methods. Thus, DTW is a good alternative to distance measure for time-series datasets. DTW has been vastly used to analyze time-series data. In the medical domain, Aach and Church [39] applied this method to identify RNA expression data. Gavrila et al. [40] deployed DTW to align biometric data (GAIT). Image recognition has also been interested in DTW. Munich and Perona [41] used DTW to solve the problem of signature verification, whereas Rath and Manmatha [42] applied it to index repositories of handwritten historical documents. Attempts have also been made in manufacturing, Helu et al. [4] used DTW to relate simulation of execution data to the actual real-life execution data. Such examples of past successes with DTW motivate our approach, detailed in Sec. 5.

5 Mapping Algorithm

In this section, we present the methodology for aligning three-axis execution data back to NC code, see Fig. 2. The intervals of MTConnect code (partially shown on the right side of the figure) are mapped to the corresponding line of NC code. Our approach consisted of three different steps: data processing, applying our algorithm, and controlling the error discrepancies. In our previous work [26], we presented preliminary results of this mapping algorithm. Updates presented here are focused on automating the tool change analysis (Sec. 5.2) and implementing improved distance measures (Sec. 5.4) to have a better alignment of the datasets.

5.1 Data Pre-Processing. First, the dataset must be verified and cleaned to initialize the alignment algorithm. MTConnect provides time-stamped execution data based on the machine tool’s activities whereas the NC code captures the low-level instructions fed into that machine to cut the part. Note that the MTConnect data might contain information beyond the scope of the NC code and it is common to find multiple part builds within the same MTConnect data file. We assume that once the machine tool begins following instructions per the NC code, the machine tool follows the subsequent NC code lines in their provided sequence. In other words, there is an implicit timeline associated with each command, which can be simulated based on instructed feed rate and distance traveled. This data segment must be identified in the MTConnect stream. Then, all data not related to spatial-temporal registration must be removed from the dataset to facilitate the alignment. At the end of this step, the MTConnect dataset contains only execution data related to the NC code instructions.

The second step of pre-processing is to retrieve and relate the coordinate systems of the two datasets. Both coordinate systems, the NC code and execution data, are defined in R^3 in xyz -space. We retrieve and store the first triple from R^3 as (x_0, y_0, z_0) . NC code reports the overall position with respect to a coordinate system or it reports the relative change to the previous point. As a result, for each new instruction, one or more point coordinates must be inferred. For example, if only the x dimension is changed then the resulting point would keep the previous y and z dimensions. Hence, the new triple is stored as (x_1, y_0, z_0) . Similarly, controller data (in the form of MTConnect) report both axial position and the direction of the tool. Often, axial positions are reported in

isolation. To store a set of triples in \mathbb{R}^3 , we assume no change in unreported positions. Applying this process will provide two array sets of \mathbb{R}^3 triples that relate to the corresponding MTConnect, \mathbf{P}_{mt} , and NC code, \mathbf{P}_{nc} . The two standards contain other information besides coordinate data which must also be stored. For example, MTConnect files might report several lines corresponding to feeds and speeds before sending new coordinate data.

The final step for our data pre-processing phase is to retrieve the path of the tool from the MTConnect dataset and identify the tool changes in the NC code dataset. The live stream data from the machine should directly contain the vector for which the tool follows to execute cutting operations. If not available, we can derive the vectors between new MTConnect points from the previously retrieved coordinates of the MTConnect dataset. These vectors represent the tool path executed by the machine. The NC code contains explicit commands for tools changes, so the method should identify and store the different parts of the files where the tool must be changed. These two truncated datasets are then stored to serve as input parameters for helping the alignment algorithm perform on the MTConnect and NC Code datasets.

5.2 Tool Changes Analysis. Identifying the tool changes in the NC code and MTConnect datasets is critical. Each tool change causes the tool offsets to differ which will alter the alignment of the \mathbf{P}_{nc} and \mathbf{P}_{mt} points. The MTConnect standard facilitates the sending of information when tool changes occur per each NC code line. However, not all machines report tool changes. To this end, we must deduce when tool changes occur based on the tool paths retrieved from the MTConnect dataset and instructed tool changes from the NC Code. Large deviations of tool trajectory relative to the workpiece can be attributed to tool operation such as tool changes, e.g., as opposed to machining operation. Here, we used a clustering algorithm, e.g., k-means [43], to automatically identify deviations in the tool paths. These deviations represent the command from the NC code to change the tool. The k-means algorithm partitions the tool paths into several clusters that each represents an abnormal behavior of the tool compared to operations realized on the workpiece. Once tool changes have been identified in the MTConnect tool paths, the MTConnect points dataset, \mathbf{P}_{mt} , is segmented by the different operations for each tool and associated to the corresponding NC code points dataset, \mathbf{P}_{nc} , of the same operation.

5.3 Mapping Alignment. Figure 4 (left) presents an automata state diagram of our mapping algorithm. First, we use Eq. (1) to refine the triple lists, i.e., to cluster similar points that have been recorded by the machine controller. This step allows to improve the efficiency of the algorithm by (1) reducing the number of points in the dataset and thus quickening the algorithm's execution and (2) facilitating the correct identification of the points without distortion based on the data frequencies afforded by a given controller.

$$\sum_{i=0}^{n-1} d(\mathbf{P}_i, \mathbf{P}_{i+1}) \leq c_r(k) \quad (1)$$

where $n = |\mathbf{P}|$, d is the distance between two points, c_r is the refining coefficient, and k is the coefficient value identified for the case study based on its operations and parameters. Defined per each use case and depending on the machining operations and parameters, the refining coefficient, c_r , is a threshold for assigning adjacent points to the same NC point. Multiple refining coefficients can be identified, based on the machining operations, parameters, and tools deployed. Since execution data are logged as time-series data, it is possible that two points are recorded at the same location (e.g., during machine idling or during an adjustment from the operator). Moreover, since the NC code and the execution data are generated independently, we cannot guarantee that if $\mathbf{P}_{nc}(i)$ is very close to $\mathbf{P}_{nc}(i+1)$ a relative $\mathbf{P}_{mt}(j)$ will be found. Thus, the coefficient c_r is

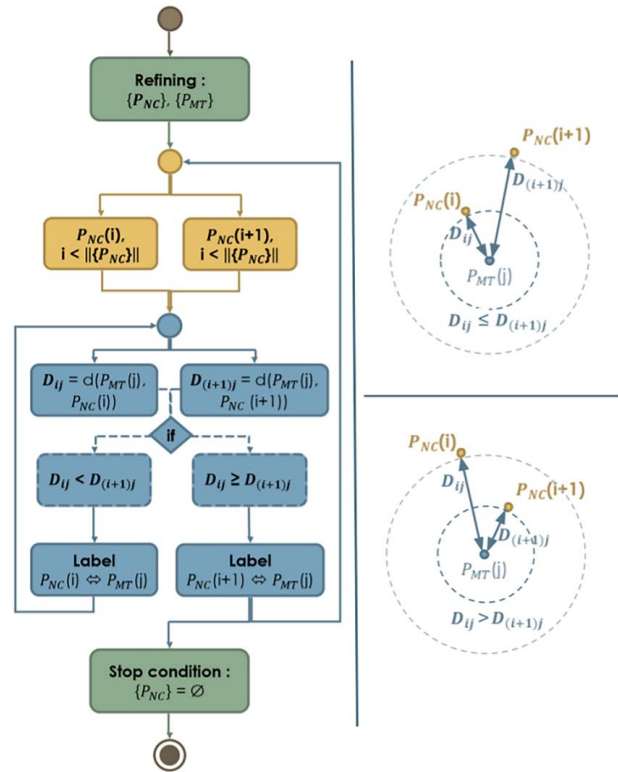


Fig. 4 (Left) Diagram representing the mapping algorithm and (Right) schema showing the coordinates alignment attribution, where $i < |\mathbf{P}_{nc}|$, $j < |\mathbf{P}_{mt}|$, \mathbf{P}_{mt} the MTConnect points dataset, \mathbf{P}_{nc} the NC Code points dataset, $|\mathbf{P}_{nc}|$ the number of points in the \mathbf{P}_{nc} dataset, $|\mathbf{P}_{mt}|$ the number of points in the \mathbf{P}_{mt} dataset, and d the distance between two points [26].

important to identify those cases. The value of the coefficient should be chosen depending on the machining operation (e.g., drilling, turning, milling) and the data feed rate. For example, in finer operations c_r should be low in order better to capture the tool movement.

After refinement, the algorithm then indexes through \mathbf{P}_{nc} and tests the proximity of each $\mathbf{P}_{mt}(j)$ to relate the appropriate points of each dataset to one another. Since the goal is to align the NC code to the execution data, the $\mathbf{P}_{nc}(i)$ and $\mathbf{P}_{nc}(i+1)$ points are first readout and then their distance from $\mathbf{P}_{mt}(j)$ is compared. Depending on the result of this comparison, either the algorithm continues through the \mathbf{P}_{mt} points or the window of \mathbf{P}_{nc} points is moved. In the latter case, we assume the next execution points relate to the following NC code instruction. Figure 4 (right) demonstrates these clustering conditions, the dotted blue areas represent the range of D_{ij} that indicates whether $\mathbf{P}_{mt}(j)$ is attributed to the neighboring $\mathbf{P}_{nc}(i)$ or $\mathbf{P}_{nc}(i+1)$ based on the distance measurement chosen.

5.4 Distance Techniques. To relate the coordinates of $\mathbf{P}_{nc}(i)$ to $\mathbf{P}_{mt}(j)$, the distance between the points must be computed. As reviewed in Sec. 4, there exist several techniques to link spatial entities. We compute the geometric distance (direct, radial), compare the similarity between the points, and iterate through the points to find an optimal alignment. Computing the geometric distance can be done using a regular spacial direct distance measure, e.g., Euclidean distance, or with a radial distance measure, e.g., cosine distance. The Euclidean distance measures the shortest distance between the points depending on the spatial environment. The cosine distance measures the smallest angle between the points. Comparing the similarity between the points, e.g., the correlation of the points, measures the standard deviation between these points. Focusing on the deviation instead of the distance between the points facilitates

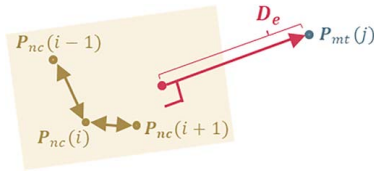


Fig. 5 Schema of the error measurement in the data alignment

the abstraction of the spatial environment and the differences across the datasets. As described in Sec. 4.4, DTW compares different time-series datasets taking into account the time distortion of each dataset’s timeline. The method does not guarantee an exact matching between the points but focuses on giving the “most likely” matched. The presented distance techniques are a fraction of all the available distance measurements. We choose these techniques to give an example of the possible result when used with our mapping algorithm (see Sec. 5.3). The choice of the distance technique should be based on the use case and the datasets.

5.5 Error Measurement. Leveraging Eq. (2), we measure alignment errors by evaluating distances described in Secs. 5.3 and 5.4 of $\mathbf{P}_{mt}(j)$ to a plane derived from NC code.

$$D_e = \hat{\mathbf{n}} \cdot \mathbf{p}_0 + \mathbf{d}_o \quad (2)$$

where $\hat{\mathbf{n}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$ the unit normal vector, \mathbf{p}_0 the evaluated point, and \mathbf{d}_o the distance of the plane from the origin.

Through this step, we compute the distance between the idealized path based on the NC code instructions, defined as a plane between vectors defined by $\mathbf{P}_{nc}(i-1)$, $\mathbf{P}_{nc}(i)$, and $\mathbf{P}_{nc}(i+1)$. We then measure the average distance between the orthogonal vector from $\mathbf{P}_{mt}(j)$ to the defined plane, as shown in Fig. 5. This process not only presents an overall error estimation between the execution data and planning data but also provide a means for more efficiently aligning data in the future. Note that there is no guarantee that a plane can be defined by every three points in an NC code. For example, when two adjacent vectors are co-linear, no plane can be formed. When such a situation occurs, we measure the error by comparing $\mathbf{P}_{mt}(j)$ to the nearest definable plane. We anticipate that this error

measurement will observe sources of discrepancies between as-planned and as-executed data.

6 Use Case Application

To validate our method, we leverage design and manufacturing information hosted on the SMS Test Bed repository at NIST. As shown in Fig. 6, the test part is a machined aluminum heat sink from an avionics control box. The information includes the full setup sheets, CAM programs, machine instructions in the form of NC code (ISO 6983 [44]), and design data hosted as an STEP AP242 file [45]. The execution data were logged via the MTCConnect standard [9], and included controller-reported data from a GF Agie Mikron HPM 1850U.

6.1 Data Pre-Processing. After plotting the MTCConnect data, we observed that the file contained another operation executed before the start of the NC code instructions. The machine controller reported more information than was required to follow our mapping method. Thus, to construct \mathbf{P}_{mt} , we registered new points after each dimension change (as in Sec. 5.1) and stored all accompanying data that did not relate to coordinates, e.g., feeds and speeds, for use later. Similarly, we constructed \mathbf{P}_{nc} by setting the first operation of the machine as (x_0, y_0, z_0) and indexing through each dimension change. We then plotted \mathbf{P}_{mt} and \mathbf{P}_{nc} and manually identified when the NC code is first realized in the execution data so that the \mathbf{P}_{mt} data not related to the identified machining processes can be removed.

6.2 Tool Changes Analysis. Once the coordinates of the two datasets, \mathbf{P}_{mt} and \mathbf{P}_{nc} , have been retrieved, the next step is to translate those set of points into the same coordinate system to facilitate comparison. Thus, we apply a translation vector to all the \mathbf{P}_{mt} points that depends on the tool offsets defined for each tool used. The \mathbf{P}_{mt} dataset is segmented to identify the tool changes during the machining process so that the translation vector corresponding to this specific tool offset can be applied. Note that some newer machine tools report tool changes through the controller and the MTCConnect streams would capture such information explicitly. Figure 7 compares the automated and manual identification [26] of tool changes in the execution data. As shown in the top barplot M, the

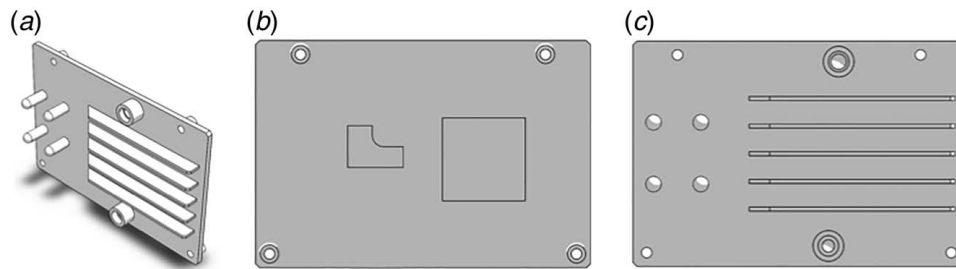


Fig. 6 (a) Isometric view of the test part, (b) view of “Operation 1” bottom of the part, and (c) view of “Operation 2” top of the part

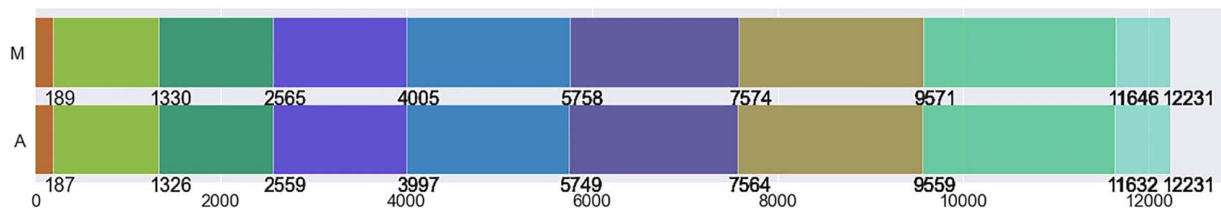


Fig. 7 Bar plot representing the tool segmentation of the \mathbf{P}_{mt} done by identifying the tool changes: M refers to manual segmentation based on visually identifying the tool changes pattern and A refers to automated segmentation based on a k-means classifier

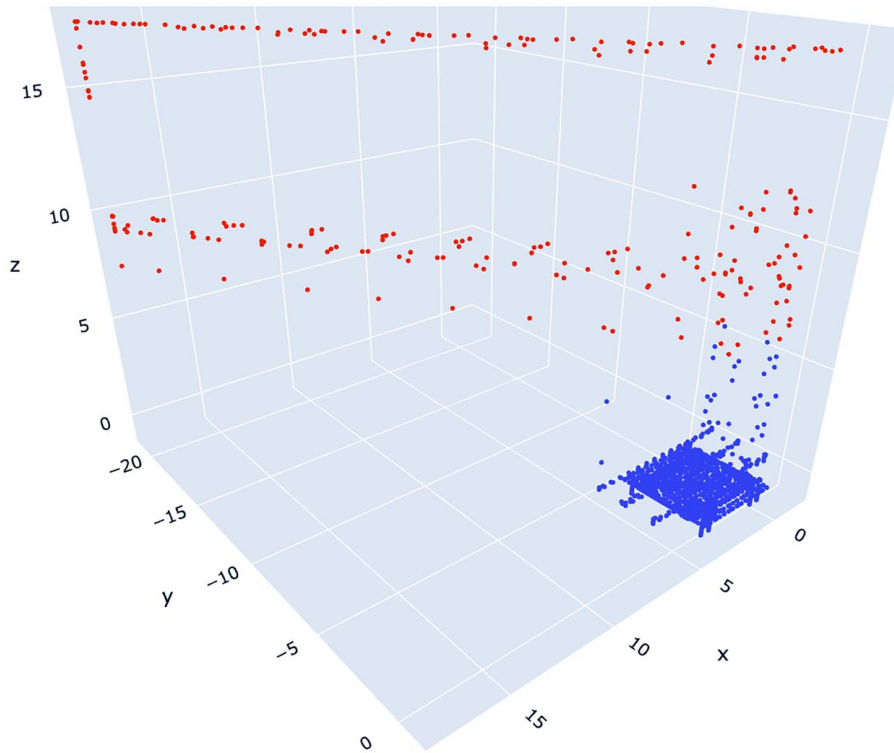


Fig. 8 Plot representing the tool paths identified as tool changes (in red) and tool operations (in blue)

first set of values represents $\mathbf{P}_{mt}(i)$ (where $i \in \{189; 1330; 2565; 4005; 5758; 7574; 9571; 11,646; 12,231\}$) that were manually identified where a tool change occurred which was based on the analysis of the Z-direction. A ceiling was fixed serving as the highest position that the tool would reach before going back to its base for a tool change. Shown in the bottom barplot A, the second set of values represents the $\mathbf{P}_{mt}(i)$ (where $i \in \{187; 1326; 2559; 3997; 5749; 7564; 9555; 11,632; 12,231\}$) that were automatically identified using k-means algorithm as tool changes. The tool path Z-direction has been fed to a k-means model to separate the tool path into two different clusters corresponding to a tool change differentiated from the one corresponding to an active operation process. The differences between the $\mathbf{P}_{mt}(i)$ points are insignificant which validates the automated tool change identification via k-means clustering. Figure 8 shows the partitioning of the \mathbf{P}_{mt} points into two categories, the points identified as tool changes (red) and those identified as operations (blue). Having the tool changes identified the tool offsets can then be applied to the corresponding \mathbf{P}_{mt} points so that the new \mathbf{P}_{mt} set of points are defined in a normalized geometric space.

6.3 Mapping Alignment. During the refinement stage, we observed that within fine movements defined by the NC code, i.e., when $\mathbf{P}_{nc}(i)$ and $\mathbf{P}_{nc}(i+1)$ were very close to each other, \mathbf{P}_{mt} failed to align as the algorithm entered an infinite loop. This problem signals the importance of properly setting a c_r to treat extremely close points as a single cluster. Additionally, there was redundant information in \mathbf{P}_{mt} as the controller, at times, was recording the same coordinates, thus, we set $c_r=0.1$ as the threshold. Since the data frequency from the controller was the same for the different machining operations, we used the same refining coefficient throughout the use case. It is possible to observe NC code lines that are linked to very few MTConnect points (i.e., less than 5). This could be due to (i) the reported frequency of sensor data from the controller, i.e., the rate at which the data are reported by the adapter on the machine is too low to log enough coordinate

Table 1 Errors measurement distribution between the alignment of the MTConnect and NC code data (in cm)

Errors distribution	25%	50%	75%	90%
Euclidean	0.0034	0.0433	1.2712	3.8999
Correlation	0.0045	0.0380	0.3636	3.8749
Cosine	0.0072	0.0421	0.7664	3.9082
DTW [46]	0.0067	0.0215	0.1101	0.6394

Note: Bold values indicate best performance per column.

points during fine NC instructions, (ii) an error in coordinate alignment, or (iii) false negatives in the alignment algorithm.

6.4 Errors Based on Distance Techniques. We implemented different distance techniques⁴ in our algorithm to compare which distance gives the best results, e.g., the distance that would allow to have the smallest overall error. Table 1 shows the distribution of errors (in cm) in the measurement based on the distance used. The errors have been segmented into four categories (e.g., 25%, 50%, 75%, 90%). The use of the Euclidean distance gives good results on identifying when the \mathbf{P}_{mt} points are the closest to the \mathbf{P}_{nc} points, e.g., distance lesser than 0.0034 cm. However, when the \mathbf{P}_{mt} points are farther from the \mathbf{P}_{nc} points, the results deteriorate. When DTW is used the error distribution shows that 90% of the \mathbf{P}_{mt} points have a distance of maximum 0.6394 cm from the \mathbf{P}_{nc} to which they are aligned.

Figure 9 represents the number of \mathbf{P}_{mt} and \mathbf{P}_{nc} clusters grouped by their errors within the cluster. One can observe that, when DTW is used, the errors within the clusters are closer to zero compared to the other distance techniques. Moreover, large deviations can be

⁴We use the scipy PYTHON package for the distance measures. <https://docs.scipy.org/doc/scipy/reference/spatial.distance.html>

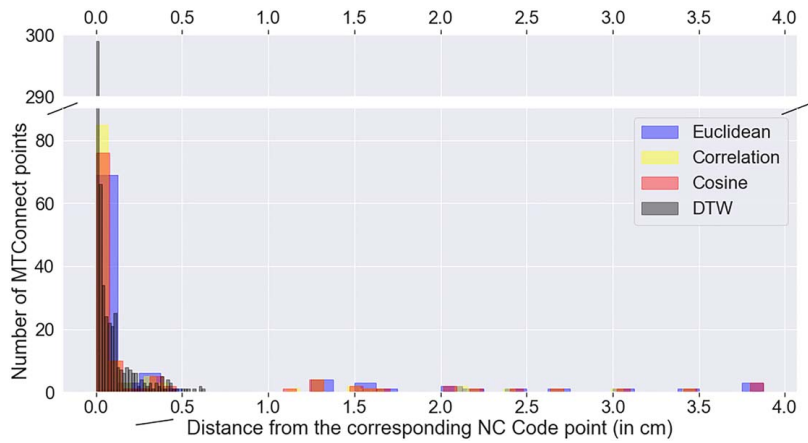


Fig. 9 Histograms of 90% errors for the distance measures

observed with the other distance techniques which highlights the fact that in those cases several P_{mt} points have been wrongly aligned to a non-corresponding P_{nc} point. Overall, we found that DTW gives the best results with this dataset. This technique does not try to link the closest points together but it tries to link the points so that the overall error is the lowest, meaning that it does not take into account the actual distance between the P_{mt} and P_{nc} points, but the overall distance between the adjacent P_{mt} points so that it reduces the noises that can be caused by machining problems, e.g., feedback time calibration. Figure 9 shows that the errors of alignment for DTW between the P_{mt} and P_{nc} points are mostly close to zero. However, we can still observe some deviations, e.g., around 0.6 cm between the P_{mt} points and their corresponding P_{nc} point, that can be attributed to external factors. We view this type of discrepancy as a possible result to exploit for automatic detection of alignment signatures. These signatures could be analyzed thoroughly in order to deduce potential tool wear or machine failures.

Figure 10 provides a visualization as a summary of our mapping algorithm. We have represented each tool, being in use, as a specific color scale that then varies in contrast based on the different points that have been mapped by the algorithm. Differences within each color scale (or changes in shade) represent a unique NC code command linked to that observed MTConnect point. The color scales defined in Fig. 7 were recreated in Fig. 10 to help identify specific instructions. This color mapping lends insight into the complexity of instructions per tool and allows to segment the set of points that have been linked to better classify and analyze the machining operations executed by each individual tool. For example, the red-scaled points relate to a relatively small number of NC code lines, even though they appear to be prominent in the

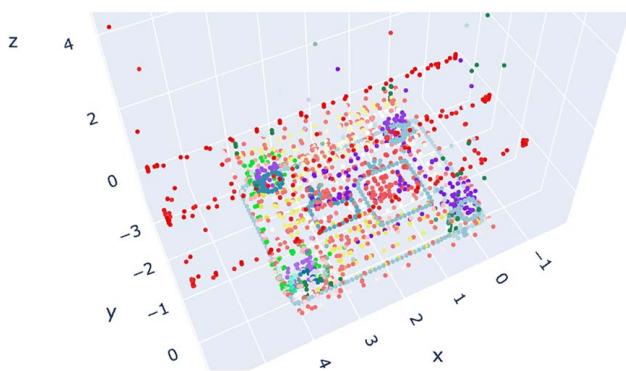


Fig. 10 Final results of mapping algorithm visualizing (1) the use of different tools through individual color scales and (2) individual NC code mappings through changes in shades within each color scale

controller logs, which indicates that the operations executed by this tool are quicker and involve a large area of the part. On the other hand, blue-scaled points have many more NC code lines affiliated with them since they represent a more complex machining feature. Note that this visualization is not meant to derive micro-level issues into what happened during the part's build but rather showcase the effectiveness of the mapping algorithm. Additional analysis and visualization effort of the dataset created by the results of our mapping algorithm will help derive knowledge from actual executed machining operations that could better inform the design of the part and improve machine maintenance.

7 Summary of Improvements of Mapping Algorithm

In our previous work [26], we suggested some possible extensions of our mapping algorithm. Here, we revisit each suggested improvement (*italicized*) and how we addressed it in this new version. *Improving coordinate representations*: In this implementation, we derive tool path vectors from the NC code and segment MTConnect data by comparing vectors rather than a point-to-point approach. *Auto-deriving a common origin & segmenting the MTConnect data per operation*: In the previous implementation, we manually identified a common origin between the MTConnect and NC code files. Also, we did not leverage tool change information to guide the mapping. In the new implementation, we used a k-means classifier to identify the tool changes that occur during machining. The classifier segments the MTConnect data into two classes which correspond to either tool cutting operation or tool change. Then, using the timeline the MTConnect data, we separate the different tool cutting operation on which our aligning algorithm can be applied. *Testing other distance measurement approaches*: In this work, we tested four different distance measures, each with their unique advantages. Dynamic time warping proved to be the most effective for aligning time-series-based spatial data. Two other extension suggestions from our previous work were not addressed in our new implementation. These include *testing error measurement in five-axis scenarios* and *dealing with external disruptions during execution*. To address these issues, we would need to develop a new dataset to test and control disruptions (e.g., operator override, emergency shut-off), while leveraging five-axis machining operations. Even if DTW (see Sec. 4.4) and the other distance measure cited Sec. 4 should work on five-axis coordinates systems, some testing will need to be done in order to better refine the data pre-processing and the tool changes analysis.

8 Conclusion

To fully realize smart manufacturing, a functioning digital thread must be demonstrated. Such a goal would require significant work.

In fact, some researchers argue that realizing a “true” digital thread would require so much work that it is innately cost-prohibitive [15]. In response, we demonstrated that even with very limited input data, using datasets from standardized systems can help overcome such challenges. We showcased an open-source approach for aligning as-executed data (as MTconnect) with as-planned data (as NC code). We believe that such data registration approach could be extended to integrate other heterogeneous data formats and standards used throughout the product lifecycle. We envision that a community-driven response to the digital thread challenge will lead to similar software releases, each representing an individual piece of the puzzle. Central to our method is the use of dynamic time warping, referred to as DTW throughout this paper, to overcome the challenge of spatial registration between the two datasets. After testing a set of registration techniques, DTW showed the best results. The resulting DTW-based method takes advantage of the temporal (or sequential) characteristics of the data. Our results further expose the importance of recording accurate timestamps per data entity in industrial processes. We recognize that a DTW-based approach will not work for in situ, ad hoc registration of as-planned and as-executed data. We implemented our approach based on the assumption that post-hoc analysis is sufficient. Adapting the method to perform in a more ad hoc manner would require a departure of the traditional use of DTW. Looking forward, it might be possible to leverage historical mappings using the DTW to predict new data as it is recorded in situ. Developing reliable tools for automating the transfer of data between the variety of PL standards is important for manufacturing. Such registration techniques represent an important step toward leveraging the full capability of artificial intelligence in manufacturing. To this end, we bridge the gap between two such standards, MTConnect and NC code. We believe that this mapping will serve as an eventual conduit to relating downstream data (e.g., from planning and execution) back to design for enhanced decision making. This backward-feeding information method would cover the need to use the existing data produced by the different PL stages that are currently unaccounted for.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

Nomenclature

P_{mt} = MTConnect points
 P_{nc} = NC code points
 CR = refining coefficient
 AP = application protocol
 CAD = computer-aided design
 CAM = computer-aided manufacturing
 CNC = computer numerical control
 DTW = dynamic time warping
 $GD\&T$ = geometric dimensioning and tolerancing
 ICP = iterative closest point
 ISO = international organization of standardization
 NC = numerical control
 $NIST$ = National Institute of Standards and Technology
 QIF = quality information framework
 $SDOs$ = standards development organizations
 SMS = smart manufacturing systems
 $STEP$ = standard for the exchange of product data
 $STEP-NC$ = STEP AP238 or ISO 10303-238

TDPs = technical data packages
 UA = unified architecture

References

- [1] Gualtieri, M., 2016, Hadoop Is Data's Darling for a Reason, Accessed April 8, 2018.
- [2] Monnier, L. V., Bernstein, W. Z., and Fofou, S., 2021, "Classifying Data Mapping Techniques to Facilitate the Digital Thread and Smart Manufacturing," IFIP International Conference on Product Lifecycle Management, Curitiba, Brazil, July 11–14, Springer, pp. 272–283.
- [3] Stark, J., 2015, "Product Lifecycle Management," *Product Lifecycle Management (Volume 1)*, Springer.
- [4] Helu, M., Joseph, A., and Hedberg Jr, T., 2018, "A Standards-Based Approach for Linking As-Planned to As-Fabricated Product Data," *CIRP Annals*, **67**(1), pp. 487–490.
- [5] Hedberg, T., Feeney, A. B., Helu, M., and Camelio, J. A., 2017, "Toward a Lifecycle Information Framework and Technology in Manufacturing," *ASME J. Comput. Inf. Sci. Eng.*, **17**(2), p. 021010.
- [6] Hedberg Jr, T. D., Hartman, N. W., Rosche, P., and Fischer, K., 2017, "Identified Research Directions for Using Manufacturing Knowledge Earlier in the Product Life Cycle," *Int. J. Prod. Res.*, **55**(3), pp. 819–827.
- [7] Helu, M., and Hedberg Jr, T., 2015, "Enabling Smart Manufacturing Research and Development Using a Product Lifecycle Test Bed," *Proc. Manuf.*, **1**(1), pp. 86–97.
- [8] Monnier, L. V., Shao, G., and Fofou, S., 2022, "A Methodology for Digital Twins of Product Lifecycle Supported by Digital Thread," ASME International Mechanical Engineering Congress and Exposition, Columbus, OH, Oct. 30–Nov. 2.
- [9] MTConnect Institute, 2014, MTConnect Standard. Accessed March 31, 2017.
- [10] Barbau, R., Krma, S., Rachuri, S., Narayanan, A., Fiorentini, X., Fofou, S., and Sriram, R. D., 2012, "OntoSTEP: Enriching Product Model Data Using Ontologies," *Comput. Aided Des.*, **44**(6), pp. 575–590.
- [11] Kwon, S., Monnier, L. V., Barbau, R., and Bernstein, W. Z., 2020, "Enriching Standards-Based Digital Thread by Fusing As-Designed and As-Inspected Data Using Knowledge Graphs," *Adv. Eng. Inf.*, **46**(1), p. 101102.
- [12] Kwon, S., Monnier, L. V., Barbau, R., and Bernstein, W. Z., 2022, "A New Implementation of OntoSTEP: Flexible Generation of Ontology and Knowledge Graphs of Express-Driven Data," *ASME J. Comput. Inf. Sci. Eng.*, **22**(2), p. 024502.
- [13] ISO 10303-238, 2007, *Industrial Automation Systems and Integration – Product Data Presentation and Exchange – Part 238: Application Protocol: Application Interpreted Model for Computerized Numerical Controllers*, International Organization for Standardization.
- [14] Bernstein, W. Z., Hedberg Jr, T. D., Helu, M., and Feeney, A. B., 2018, "Contextualising Manufacturing Data for Lifecycle Decision-Making," *Int. J. Prod. Lifecycle Manage.*, **10**(4), pp. 326–347.
- [15] West, T. D., and Blackburn, M., 2017, "Is Digital Thread/digital Twin Affordable? A Systemic Assessment of the Cost of DoD's Latest Manhattan Project," *Proc. Comput. Sci.*, **114**(1), pp. 47–56.
- [16] Helu, M., Sprock, T., Hartenstine, D., Venketesh, R., and Sobel, W., 2020, "Scalable Data Pipeline Architecture to Support the Industrial Internet of Things," *CIRP Ann.*, **69**(1), pp. 385–388.
- [17] ISO 10303-42, 2003, *Industrial Automation Systems and Integration – Product Data Presentation and Exchange – Part 42: Integrated Generic Resource: Geometric and Topological Representation*, International Organization for Standardization, Geneva, Switzerland.
- [18] Hedberg, T., Lubell, J., Fischer, L., Maggiano, L., and Barnard Feeney, A., 2016, "Testing the Digital Thread in Support of Model-Based Manufacturing and Inspection," *ASME J. Comput. Inf. Sci. Eng.*, **16**(2), p. 021001.
- [19] Electronic Industries Association, 1980, *Interchangeable Variable Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically Controlled Machines*, Electronic Industries Association.
- [20] Hardwick, M., and Loffredo, D., 2006, "Lessons Learned Implementing STEP-NC AP-238," *Int. J. Comput. Int. Manuf.*, **19**(6), pp. 523–532.
- [21] Dimensional Metrology Standards Consortium, 2014, Part 1: Overview and Fundamental Principles in Quality Information Framework (QIF) - An Integrated Model for Manufacturing Quality Information, Accessed March 31, 2017.
- [22] The Association for Manufacturing Technology, 2018, *OPC Unified Architecture for MTConnect Companion Specification, Release Candidate 2.0.5*. AMT.
- [23] Fischer, K., Rosche, P., Trainer, A., Feeney, A. B., and Hedberg, T. D., 2015, "Investigating the Impact of Standards-Based Interoperability for Design to Manufacturing and Quality in the Supply Chain," Technical Report, National Institute of Standards and Technology.
- [24] Danjou, C., Le Duigou, J., and Eynard, B., 2014, "OntoSTEP-NC for Information Feedbacks From CNC to CAD/CAM Systems," IFIP International Conference on Advances in Production Management Systems, Ajaccio, France, Sept. 20–24, Springer.
- [25] Feng, S. C., Bernstein, W. Z., Hedberg Jr, T., and Feeney, A. B., 2017, "Towards Knowledge Management for Smart Manufacturing," *ASME J. Comput. Inf. Sci. Eng.*, **17**(3), p. 031016.
- [26] Monnier, L., Bemstein, W. Z., and Fofou, S., 2019, "A Proposed Mapping Method for Aligning Machine Execution Data to Numerical Control Code," 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), Vancouver, BC, Canada, Aug. 22–26.

- [27] Fotheringham, A. S., Brunsdon, C., and Charlton, M., 2000, *Quantitative Geography: Perspectives on Spatial Data Analysis*, SAGE Publications Inc, Thousand Oaks, CA.
- [28] Lin, R., Agrawal, K.-I., and Shim, H. S. S. K., 1995, "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases," 21st International Conference on Very Large Data Bases, Zurich, Switzerland, Sept. 11–15.
- [29] Keogh, E. J., and Pazzani, M. J., 1998, "An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback."
- [30] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S., 2001, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Knowl. Inform. Syst.*, **3**(1), pp. 263–286.
- [31] Besl, P. J., and McKay, N. D., 1992, "Method for Registration of 3D Shapes," Sensor Fusion IV: Control Paradigms and Data Structures, International Society for Optics and Photonics.
- [32] Béarée, R., Dieulot, J.-Y., and Rabaté, P., 2011, "An Innovative Subdivision-ICP Registration Method for Tool-Path Correction Applied to Deformed Aircraft Parts Machining," *Inter. J. Adv. Manuf. Technol.*, **53**(1), pp. 463–471.
- [33] Chen, Y., and Medioni, G., 1992, "Object Modelling by Registration of Multiple Range Images," *Image Vision Comput.*, **10**(3), pp. 145–155.
- [34] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y., 1994, "Fast Subsequence Matching in Time-Series Databases," *SIGMOD Rec.*, **23**(2), pp. 419–429.
- [35] Rafiei, D., and Mendelzon, A., 1997, "Similarity-Based Queries for Time Series Data," 1997 ACM SIGMOD International Conference on Management of Data, Tucson, AZ, June 1.
- [36] Sakoe, H., and Chiba, S., 1978, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoust. Speech Signal Proc.*, **26**(1), pp. 43–49.
- [37] Itakura, F., 1975, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Trans. Acoust. Speech Signal Proc.*
- [38] Chu, S., Keogh, E., Hart, D., and Pazzani, M., 2002, "Iterative Deepening Dynamic Time Warping for Time Series," Proceedings of the 2002 SIAM International Conference on Data Mining, Proceedings, Society for Industrial and Applied Mathematics.
- [39] Aach, J., and Church, G. M., 2001, "Aligning Gene Expression Time Series With Time Warping Algorithms," *Bioinformatics*, **17**(6), pp. 495–508.
- [40] Gavrilu, D. M., and Davis, L. S., 1995, "Towards 3-d Model-Based Tracking and Recognition of Human Movement: A Multi-view Approach," International Workshop on Automatic Face-and Gesture-Recognition, Vol. 3, Citeseer.
- [41] Munich, M. E., and Perona, P., 1999, "Continuous Dynamic Time Warping for Translation-Invariant Curve Alignment With Applications to Signature Verification," 7th IEEE International Conference on Computer Vision, Kerkyra, Greece, Sept. 20–27.
- [42] Rath, T. M., and Manmatha, R., 2003, "Word Image Matching Using Dynamic Time Warping," 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Madison, WI, June 18–20.
- [43] MacQueen, J., 1967, "Some Methods for Classification and Analysis of Multivariate Observations," 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, Dec. 27, 1965–Jan. 7, 1966.
- [44] ISO 6983-1, 2009, *Automation Systems and Integration – Numerical Control of Machines – Program Format and Definitions of Address Words – Part 1: Data Format for Positioning, Line Motion and Contouring Control Systems*, International Organization for Standardization, Geneva, Switzerland.
- [45] ISO 10303-242, 2014, *Industrial Automation Systems and Integration – Product Data Presentation and Exchange – Part 242: Application Protocol: Managed Model Based 3d Engineering*, International Organization for Standardization, Geneva, Switzerland.
- [46] Giorgino, T., 2009, "Computing and Visualizing Dynamic Time Warping Alignments in R: The DTW Package," *J. Stat. Softw.*, **31**(1), pp. 1–24.