In-situ Porosity Detection in Additive Manufacturing

In-situ Porosity Detection in Additive Manufacturing

Erik Sievers
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
eriksie@chalmers.se

Marina Papatriantafilou
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
ptrianta@chalmers.se

Vincenzo Gulisano
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
vincenzo.gulisano@chalmers.se

Eduard Hryha
Chalmers University of Technology, centre for Additive Manufacture-Metal (CAM2)
Gothenburg, Sweden
hryha@chalmers.se

Lars Nyborg
Chalmers University of Technology, centre for Additive Manufacture-Metal (CAM2)
Gothenburg, Sweden
lars.nyborg@chalmers.se

Zhuoer Chen
Chalmers University of Technology, centre for Additive Manufacture-Metal (CAM2) and School of Iron and Steel, Soochow University
Gothenburg, Sweden
zchen@suda.edu.cn

Abstract

Additive Manufacturing (AM) is a rapidly growing technology with applications in aerospace, automotive, and medical industries. Scalable AM requires in-situ quality monitoring to detect defects promptly. However, in-situ monitoring introduces scalability challenges due to high data volumes, rapid acquisition rates, and strict latency requirements. We introduce Hephaestus, a continuous in-situ monitoring system for data streams from optical monitoring sensors, able to detect porosity risks promptly and to balance accuracy and timeliness by adjusting the *window* of data used for porosity detection. Using data from two builds, we study this trade-off and the method's cost-benefit towards early cancellation decisions.

CCS Concepts

• **Applied computing** → *Computer-aided manufacturing*;
• **Computing methodologies** → Machine learning approaches.

Keywords

In-situ Monitoring; Stream Processing; Machine learning

ACM Reference Format:
Erik Sievers, Marina Papatriantafilou, Vincenzo Gulisano, Eduard Hryha, Lars Nyborg, and Zhuoer Chen. 2025. In-situ Porosity Detection in Additive Manufacturing. In *The 19th ACM International Conference on Distributed and Event-based Systems (DEBS '25), June 10–13, 2025, Gothenburg, Sweden.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3701717.3734463

DEBS '25, Gothenburg, Sweden

ACM ISBN 979-8-4007-1332-3/25/06
https://doi.org/10.1145/3701717.3734463

1 Introduction

Additive Manufacturing (AM), commonly named 3D printing, refers to a family of processes in which a digital 3D model is built up layer by layer to create complex parts directly from raw materials. AM has gained widespread adoption in aerospace, medical implants, and automotive manufacturing industries due to its ability to produce lightweight, geometrically intricate components with minimal tooling. Among metal-based AM techniques, Powder Bed Fusion - Laser Beam (PBF-LB) is a common method for metal-based AM used to construct objects, from multiple thin layers stacked on top of each other, using a laser to melt powder layer by layer.

Despite the benefits it offers over traditional metal manufacturing - less waste, faster prototyping, and weight reduction - PBF-LB is still held back by a lack of process repeatability and inconsistent quality of build objects due to defects arising during the manufacturing process [8]. One such type of defect, *porosity* (empty volumes inside an object), can severely impact objects' durability and toughness [6]. Although some degree of porosity is expected due to impurities in the material used to print an object, limiting the occurrence of pores is important to ensure the reliability of printed objects. To assess quality, non-destructive evaluation of each constructed part using, e.g., X-ray Computed Tomography, is often used in industries like the aerospace and medical ones [18]. However, X-ray techniques have limited usefulness for large objects due to their limited penetration depth and the size constraints of scanning equipment. Also, when quality is assessed after manufacturing is complete, *ex-situ*, such assessment increases construction time and cost, especially when porosity occurs early in the build process for an object that is then scrapped.

Since AM builds objects in increments, this shortcoming could potentially be alleviated using *in-situ*, live monitoring

to monitor continuously the build process using different sensors such as cameras, photodiodes, microphones, and libraries of sensor *fingerprints* that are indicative of successful or failed build processes [5]. Build processes could then be stopped – entirely, if porosity is observed for all the objects being printed, or partially, only for affected objects – when data indicates an object is likely defective or that the build process deviates from the fingerprints of successful ones. Although such monitoring does not observe defects directly, but rather conditions under which defects may occur, the utility of observing said conditions has been validated once their construction was finalized [15, 26]. Hence, live monitoring could result in a *qualify-as-you-build* paradigm, validating the properties of objects being printed and scrapping defective objects early [18].

1.1 Challenges and Contributions

Although there has been previous research interest in in-situ monitoring for PBF-LB (see [8, 9] and references therein for a detailed overview), existing works propose batch-based methods in which all the data collected during the production of an object is analyzed as a whole once said object is fully built. As such, existing methods cannot provide timely results and steer a printing job based on the evolving quality of each printed object, either to halt it (when defects cannot be fixed) or adjust it (by continuing to build only non-defective objects).

To advance the state-of-the-art, we aim at designing and implementing a method able to shift from processing sensor data as a batch (suitable for ex-situ evaluation) to processing it layer by layer as a stream, capturing the temporal nature of the data and producing a monitoring result that utilizes the data as it becomes available. In order to assess the goodness of the proposed method, we focus on its accuracy and latency, which are key attributes for effectiveness. Moreover, we present and discuss results about the possible cost savings enabled by the method. We make the following contributions:

- We propose a novel encoding method based on spatial outlier detection and fingerprinting of the distribution of outlier values for in-situ monitoring of *optical tomography* (OT) images, which are high-resolution long-exposure heat images of each printed layer. The method, which we name Hephaestus, operates continuously, capitalizing on the streaming nature of the data and allowing for trading results' accuracy for timeliness based on the number of consecutive layers in which neighboring outliers are clustered and reported as a single porosity defect.
- We create a prototype streaming implementation that, after each layer is constructed, can continuously create new results based on the added data from the latest layer.
- We assess the performance of our prototype by measuring the accuracy (ranging from 0.72 to 0.99 ROC-AUC, where the Receiver Operating Characteristic - Area Under the Curve (ROC-AUC) quantifies the model's ability to identify porosity defects correctly), the latency (ranging from 0.22 to 10.42s on a commercial server), and how these change over time as more data becomes available from the build process. Combining these results, we also estimate the impact on the cost of making an early cancellation decision at different points during the build process.
- We make the datasets used in the evaluation public at [4], enabling the research community to replicate and build further on the presented methodology.

The rest of the paper is structured as follows: Section 2 provides an overview of the background and the streaming problem formulation. Section 3 explains the streaming method and its settings, followed by an analysis of the properties of the method as well as its possible impact on production costs in Section 4. We assess our proposed method, in terms of both accuracy and processing performance, in Section 5. Section 6 discusses other related work and Section 7 summarizes the outcomes and outlines possible future research.

2 Preliminaries and problem description

2.1 Optical Tomography - sensor and data

Numerous methods exist to monitor PBF-LB build processes [8]. One increasingly common setup uses a camera with a band-pass filter, mounted at a static angle relative to the powder source. In such a setup, the heat is measured across the powder bed, producing an X x Y matrix of numerical values proportional to the heat, which is commonly visualized as an image with warmer or brighter colors such as red, yellow, or white representing higher heat values and blue, purple, or black representing lower heat values. Figure 1 shows one such image, from the data used in the evaluation (see Section 5). This kind of data is commonly referred to as OT data, a designation we use for the remainder of this paper. Similar to how PBF-LB creates a three-dimensional object by layering two-dimensional slices on top of each other to form a three-dimensional representation of the build.

2.2 Stream Processing

Stream processing refers to the real-time analysis of continuous data streams, a methodology that sees an increasing amount of applications (cf. [12, 22, 31] and references therein). Its continuous processing is carried out by queries expressed as graphs of operators, each performing either stateless operations such as filtering, or stateful ones such as aggregation (e.g. to calculate statistics such as average, median, and quantiles), and joining to match elements from pairs of streams (e.g., to support the calculation of the correlations between streams' elements). Stateful computations using stream processing are done over sequences of data defined as *windows* over the data stream(s), characterized by two key parameters: *window size*

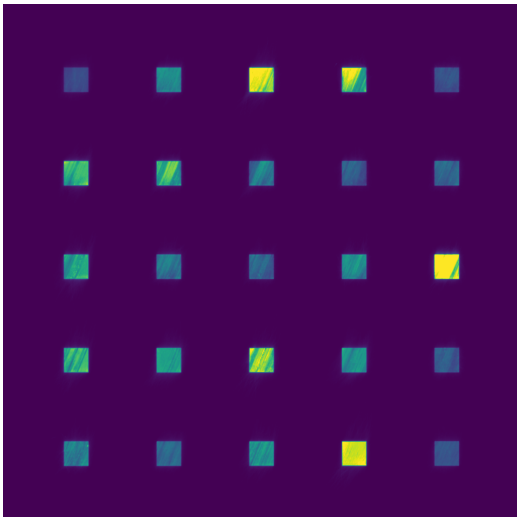


Figure 1: Single layer of optical tomography data (see Section 5). Yellow pixels indicate a higher heat, while purple pixels indicate a lower heat.

and *window advance* (denoted W_s and W_a, respectively). The window size defines the amount of past data considered for analysis. For instance, if, in the context of the problem at hand, the window size is set to 10 layers, each stateful operation is performed over the data of 10 layers. The window advance dictates the shift between two successive windows. Using a concrete example, if the window size is set to 10 layers and the window advance is 5, the analysis is run over data points from layers [1-10], [6-15], [11-20], [16-25], etc.

2.3 Problem formulation

An OT setup reports the relative temperature of the powder bed, generating one OT image per layer. These images form a bounded data stream, $S = s_1, s_2, \ldots, s_n$, with s_1 corresponding to the first layer of the constructed object and s_n the last layer. Upon completion of each window of data in the stream, we want to determine whether the layers included in the window have a porosity higher or lower than a given level, e.g., 0.5%. That is, whether these layers – and thus the object – are *conforming* (*non-porous*) to standards or *defective* (*porous*).

The following properties are of interest in our work:

- The *rate of false positives*, *false negatives*, and *true positives*, i.e., the rate at which a method misclassifies objects as defective, fails to identify defective objects, and correctly classifies defective objects, respectively, relative to ground truth (e.g., from a destructive evaluation method, see Section 5).
- The *latency*, the time it takes to receive a result from the monitoring method about a window after the data from the last layer of that window is made available.
- The resulting *cost reduction* of using the monitoring method, enabling early detection in the presence of defects, which is directly affected by the other three properties.

As we show in the paper (see Section 4), there are trade-offs associated with these properties and Hephaestus provides ways to navigate such trade-offs by tuning its parameters.

For the assessment of Hephaestus's detection accuracy based on the rate of false positives, false negatives, and true positives, we note that the three depend on the data under study and the chosen classification method, and that many classifiers can reduce the rate of false positives by accepting a higher rate of false negatives, or vice versa. To jointly study these rates and assess their interdependency, we rely on the *Receiver Operating Characteristics* (ROC) [29] curve and *Associated area Under the Curve* (ROC-AUC) [6]. The ROC curve of a detection method, showing the tradeoff between true positive rate and false positive rate (i.e., the rate of false alarms), has been used in related work on AM [14, 15]. Note that an ROC-AUC of 0.5 can be obtained by guessing, whereas 1.0 indicates perfect classification. Figure 2 shows a sample ROC curve, taken from one of the experiments later discussed in Section 5.

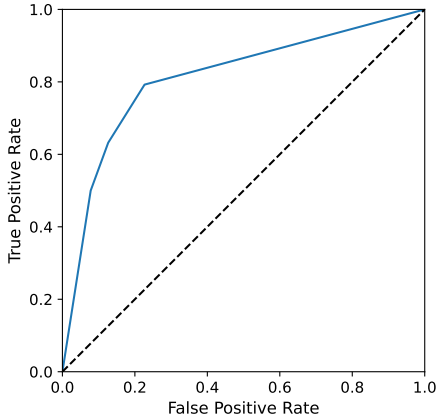
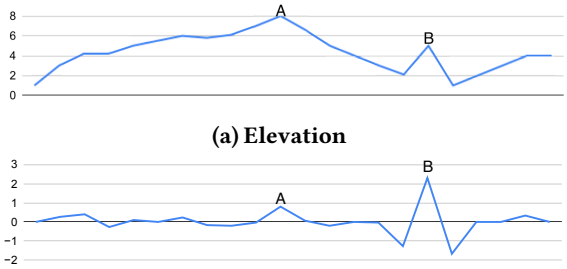


Figure 2: Sample ROC curve from an experiment in Section 5. The dashed line is the true/false positive rate dependency of a random classifier. The area under the diagonal and curve (Hephaestus) is 0.5 and 0.81, respectively.

To assess whether the latency incurred by Hephaestus can support timely detection of defective objects, we note that there exists a desirable upper limit given by the time that passes from the moment the OT sensor produces an output for the monitoring method to process until the construction of the next layer begins, which is commonly in the realm of seconds, for instance up to 3 seconds [11]. If the classifier succeeds in marking an object as conforming or defective within this time frame, the build plan could be changed to cancel defective objects before the next layer begins construction. As in [11], in our study, such a limit is set to 3 seconds (see Section 5).

Finally, to put in context the different costs involved in a printing job, we introduce an adaptation of the recent model of AM cost by Colosimo et al. [5], suited for the context of stream-based analysis and classification, in Section 4.2.

3 The Hephaestus method



(a) Elevation

(b) Local Z-score

Figure 3: 1D spatial dataset of elevation values of a sample road with corresponding Local Z-score values.

3.1 Core idea – Spatial Outlier Detection

As observed in [16], areas with high porosity tend to radiate more heat due to worse thermal conductivity. Our core idea builds on that observation. Viewing an OT image as a map, we are interested in areas whose heat radiation deviates considerably from their surrounding area. Areas like this can be detected using a *spatial outlier detection* method.

Figure 3 exemplifies the concept of a spatial outlier with 1D data about the elevation of a road. Intuition suggests that points close to each other on a road have similar elevations. This phenomenon, known as *spatial autocorrelation*, describes the tendency for close-by points to exhibit similar values. Standard features that display this behavior include wind speed, temperature, and population density measurements.

Building on this, a spatial outlier is a point different enough (based on a given threshold) from its surroundings to appear as from a different distribution. Note that what sets a spatial outlier apart is not necessarily its value, but also its location. That is, it can be a *normal* data point but in an abnormal location. Regarding the road example, Figure 3a shows the absolute elevation of points of a sample road. Although point A is the highest point in the set, point B sticks out because of its location. This can be quantified using spatial outlier detection.

Spatial outlier treats each point as a *pivot point* for its immediate neighboring points. In Figure 3b, the value of each pivot point is the Local Z-score (Deviation from Local Mean): the distance between the average of the corresponding point and the preceding and subsequent points, and the mean value of all points, divided by all points' standard deviation. As shown, the value of B exceeds that of A, highlighting how B is a potential outlier for the given dataset.

Spatial outlier detection is not restricted to one dimension. In the example in Figure 3, the neighborhood of a point is simply the point itself as well as the points directly to its left and right. For two dimensions, such as an image with pixels, the neighborhood can be expanded to include a central pixel and all surrounding pixels. For three dimensions, such as multiple images layered on top of each other (e.g., multiple stacked OT images), the neighborhood can include a central pixel with its surrounding pixels in one image, and the same area of the contiguous images before and after the central image.

Spatial outlier detection combines nicely with the nature of AM build process data and stream processing: data points that are close in space become available shortly after each other as layers are printed. For each point in a layer, neighbors from previous layers are available right away, while more neighbors become available as the next layers are printed.

3.2 Hephaestus' overview

Hephaestus combines spatial outlier detection and stream processing in a data processing pipeline. Algorithm 1 overviews the main parameters used in the pipeline, Figure 4 overviews said pipeline, while Algorithm 2 presents its main steps as pseudocode. The first step takes each OT image produced by the camera as input. The final output classifies an object as non-porous or porous depending on the current window of the object's data (i.e., the latest reported layer and a given amount, given by the window size, of the preceding contiguous layers). The pipeline starts processing a layer as soon as it becomes available and does not need all layers to classify an object. Instead, it produces a classification for each window once all the layers belonging to such a window are received. In a nutshell, each OT image is filtered to remove *background* data (i.e., unrelated to the object being printed). Every data point is assigned a score reflecting its likelihood of being an outlier. Points that are later classified as outliers are interpreted as potential indicators of porosity based on their aggregated property (e.g., their quantity).

For ease of exposition and without lack of generality, we consider a build printing a single object. To support multiple objects in a build, Hephaestus keeps track in each step of which object the data it is operating on belongs to.

Algorithm 1 Parameters of the Hephaestus algorithm

1: W_s, W_a ▷ Window size and window advance
2: Bt ▷ Threshold used for background elimination
3: x ▷ Width and length of a neighborhood (in number of pixels)
4: h ▷ The height of a neighborhood (in number of layers)
5: min, max, bins ▷ Min, max values and nr of bins for fingerprinting
6: classifier ▷ e.g., a K-nearest neighbor ML-classifier

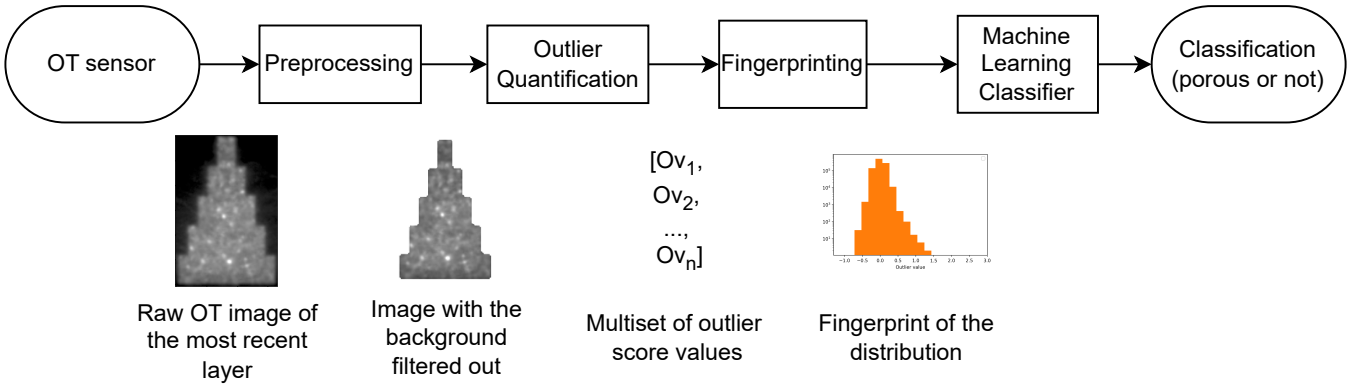


Figure 4: Overview of the HEPHAESTUS pipeline proposed in this work.

Algorithm 2 Overview of the HEPHAESTUS's steps upon reception of an OT image. For ease of exposition and without lack of generality, we assume a single object is being printed in each printing job. Also, we show the list *layers* as a growing list, while in practice, a circular buffer of size W_s suffices.

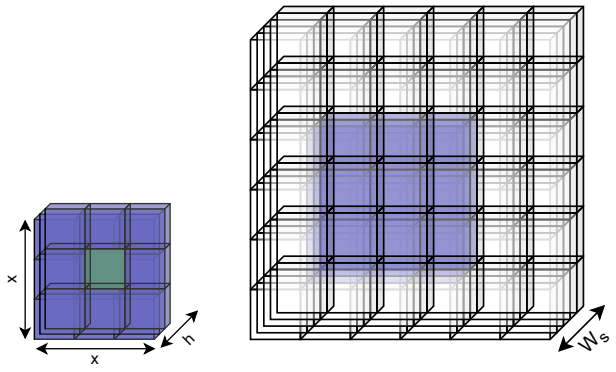
1: **list** *layers* ▷ A list of OT images
2: $i \leftarrow 0$ ▷ index of the current layer, initially 0
3: **upon** reception of *nextLayer*, the latest OT image **do**
4: $layers[i] \leftarrow preprocess(nextLayer, Bt)$ ▷ Remove background data
5: $i \leftarrow i+1$
6: **if** $i \geq W_s \wedge (i - W_s)\ \%\ W_a = 0$ **then**
7: $os \leftarrow compOutlierScores(layers[i-W_s{:}i], x, h)$ ▷ Assign outlier scores for all points in the window
8: $fp \leftarrow makeFingerprint(os, bins, min, max)$ ▷ Bin points based on their outlier score, capturing their distribution into a fingerprint
9: $class \leftarrow classifier.classify(fp)$ ▷ Classify the fingerprint, comparing it to fingerprints from a training dataset with known classes
10: Output(*class*)

3.3 Preprocessing

To capture only features related to the printed object, this step filters out background pixels from each OT image. Since the background radiates less heat than the object being printed, we remove all points below a certain heat threshold. Alternatively, filtering can rely on the printing specifications, stating which areas of each layer belong to an object.

3.4 Outlier quantification

This step relies on an outlier detection method to transform the values in the window into a multiset of values indicating how much of an outlier each data point is. Our pipeline can integrate different methods for spatial outlier detection. Since it is not within our scope to assess the quality of different methods, we only focus on the method we have chosen, following earlier studies [3], based on *Moran scatter plots*. We first normalize each data point p using the mean and standard deviation of all the points part of the streaming window of size W_s to which p belongs. Next, the outlier score assigned to p is defined as the distance from the point itself and the least squares regression line fitting all p's neighbouring points in the window.



(a) Neighbourhood **(b) Neighbourhood inside a window**

Figure 5: Illustration of a 3x3x3 neighbourhood. The neighbourhood (blue) is centered around its pivot point (green), on its own in (a) and inside a window in (b).

The size of p's neighbourhood is defined by two hyperparameters: x, the size of the neighbourhood within a layer, and h, the size in terms of number of layers[1]. Figure 5a illustrates

[1]We assume x and h are odd and that p falls in the center of the neighbourhood.

how these hyperparameters affect the neighbourhood: h is bounded by W_s: a neighbourhood cannot span more layers than the window; W_s can be set to a value greater than h however, leading to a better normalization and often more precise classification (see Section 5.3). For each point p, we only consider neighbourhoods for which exactly x and h points can be found along the three dimensions, thus excluding points from neighbourhoods located at the edge of the printed objects.

3.5 Fingerprinting

To capture the distribution of the outlier score values from the previous step, we bin such values into a histogram, using bins of the same size. This summarization acts as a fingerprint of the process, which can be compared to previous builds' fingerprints to assess the likelihood of the build being defective.

As aforementioned in Section 1, state-of-the-art quality assessment is currently conducted by post-completion inspection of printed objects. Hence, there usually exist datasets with data referring to successful/unsuccessful jobs. In our case study, we had access to such data (see Section 5), containing information about objects that (1) were printed with a setup resulting in both non-porous and porous objects and (2) were also analyzed after their complete printing to label them as non-porous or porous. The minimum and maximum values we chose are determined using such data, representing the smallest and largest outlier values, respectively.

Figure 6 shows one fingerprint obtained from a porous object visualised as a histogram (orange, transparent) compared to the average of a larger set of objects (blue), out of which some are non-porous and some are porous.

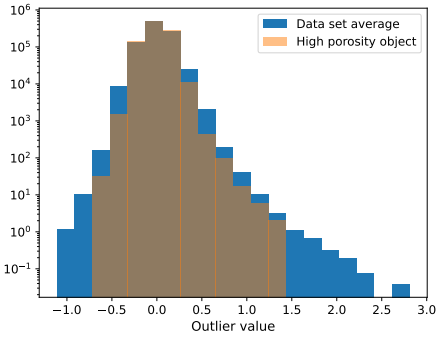


Figure 6: Histogram of outlier score values obtained using Moran scatter plot. Orange (transparent) values correspond to one highly porous object, and blue is the average of numerous objects (both non-porous and porous).

3.6 Classification

The fingerprint is then used as input to a machine learning classifier, which outputs a yes/no answer to whether the window indicates a porosity level above the threshold.

Similarly to what is said in Section 3.4, it is not within the scope of our work to assess the quality of different classifiers (k-nearest neighbours, random forest, multi-layer perceptron, or logistic regression [29]). We opt for the k-nearest neighbours method since, as also discussed in [28], it supports the explainability of the outcomes reporting objects as porous.

In Hephaestus, the k-nearest neighbour classifier is first given a training data set of fingerprinting outcomes whose class (i.e., non-porous or porous) is already known. Then, the classifier constructs a search tree, e.g., a KD-tree [2]. When queried for the classification of a new fingerprinting outcome, the classifier finds the k closest fingerprinting outcomes in the training data using the tree and lets those fingerprinting outcomes vote for which class the new fingerprinting outcome should belong to. As a result, it is possible to inspect the result to find out which fingerprinting outcomes in the training data were the closest and see why any given classification was made. This is of interest in particular to improve the classifier when the latter leads to incorrect classification.

Also in this case (see Section 3.5), we rely on data from previous builds for training purposes (see Section 5).

4 Properties of the Hephaestus method

Recalling from Section 2, key aspects we use to assess Hephaestus's performance include the latency for producing results and the impact on production costs. In the following subsections, we elaborate on both aspects.

4.1 Latency and time complexity

The latency metric quantifies the time required to obtain a result for a complete streaming window once data from the last layer is available. As discussed in Section 2.3, a key consideration is whether this latency remains within the time going from the moment the OT sensor produces an image to that at which the construction of the next layer starts, typically a few seconds [11]. Hephaestus's time complexity influences the latency metric according to the individual steps of the processing pipeline. Based on the outlier quantification (see Section 3.4), fingerprinting (see Section 3.5), and classification (see Section 3.6) techniques we chose, we next estimate the time complexity of the work performed by Hephaestus to process all the data points within a window noting that:

- The first stage (preprocessing), which uses a static threshold to remove background data points, requires checking which points in each layer have a value above said threshold before adding such points to the window itself, implying an $O(W_s n)$ time complexity, n being the number of data points in a layer.

- The subsequent stage, outlier quantification, involves a constant number of steps per point per window (depending only on the local neighbourhood, which is defined by the x and h constants), hence also implying an $O(W_s n)$ time complexity.
- The fingerprinting stage incurs a number of steps linear in the window size and number of points in a layer, leading also to an $O(W_s n)$ time complexity.
- Finally, the k-nearest neighbor classifier used by Hephaestus with a K-D tree for neighbourhood search leads to an $O(log(a))$ time complexity per window in expectation, where a is the number of examples used for training.

Accounting for all these costs, we note that:

Lemma 4.1. *The time complexity of* Hephaestus *for any window of size* W_s *is* $O(W_s n + log(a))$.

As discussed in Section 2.2 and shown in Algorithm 2, for a given W_a, a result is produced by Hephaestus every W_a layers. Since W_a affects how often the processing of a window's data is triggered, we can estimate the per-layer average complexity:

Lemma 4.2. *Given Lemma 4.1, the average complexity of* Hephaestus *per layer is* $O(\frac{W_s n + log(a)}{W_a})$.

Lemma 4.2 implies that Hephaestus provides two knobs that can be turned to tweak performance trade-offs, according to e.g. quality of service requirements and the hardware available. In particular, using a smaller window size W_s will improve latency. Also, using a smaller window advance W_a will create more frequent results at the expense of needing more computations per layer. If the hardware doing the processing cannot keep up with the rate of new layers (i.e., by the time a new layer has finished, previous layers are still being processed), W_a can be increased to reduce the amount of work.

4.2 Impact on production cost

The production cost metric estimates the influence of making timely cancellation decisions on the construction cost, based on e.g., material, energy, and time. Such a metric has been proposed by e.g., Colosimo et al. [5]. In state-of-the-art, though, such a metric refers to assessments made at the end of a printing process, while we aim at porting it to the streaming context, as discussed next.

From [5], the cost for producing a single object is:

$$C_{net} = C_{pre} + C_{proc} + C_{post} + C_{material} \tag{1}$$

where C_{pre}, C_{proc}, C_{post}, and $C_{material}$ are the costs for preprocessing, processing, postprocessing, and materials, respectively. Preprocessing includes setting up the machine by preparing the powder, loading the files containing the information about what is to be printed, etc. Processing refers to the task of physically constructing the object in the machine. Postprocessing includes removing the build objects from the chamber, cutting them from the base, and investigating the object for defects. We refer to the defect investigation cost, part of C_{post}, as $C_{inspection}$. Finally, the material cost is mainly referring to the cost of the metal powder used for the process.

Construction processes have two potential outcomes: they can result in a successfully constructed object, a *part*, or a defective one that needs to be discarded, a *scrap*. The ratio of scrap to all constructed objects is called the *scrap ratio* and is denoted γ. When accounting for the scrap ratio we obtain the base case, C_{base}, i.e., the cost of producing a single part successfully when not using any monitoring to influence the build:

$$C_{base} = \frac{C_{net}}{1-\gamma} \tag{2}$$

However, if an object is identified as a scrap before construction has finished, there is no need to continue constructing it, resulting in a reduced processing cost if the classification is correct. The reduction in processing cost is proportional to the *completion ratio*, CR, i.e., the ratio of the object completed. For the sake of simplicity, we assume this is the same as the fraction of finished layers to the total number of layers of an object. Since scraps are not usually inspected, we note the cost of inspection $C_{inspection}$ can be deducted from C_{post}[2]. With this in mind, Equation 3 breaks down the cost of producing a scrap.

$$C_{scrap} = C_{pre} + (C_{proc} + C_{material}) * CR + C_{post} - C_{inspection} \tag{3}$$

Assuming that the construction of individual objects can be canceled based on the result of an in situ monitoring method, we need to know the rate of false positives α and the rate of false negatives β of said method to calculate its cost. a fraction of $(1-\gamma)(1-\alpha)$ of all objects that begin construction also finish construction successfully, whereas $\gamma\beta$ of all objects are finished constructing despite being defective. Meanwhile, $(1-\gamma)\alpha$ of all objects are incorrectly cancelled ahead of time, and a $(\gamma)(1-\alpha)$ fraction of them are correctly cancelled ahead of time. Putting all of this together we obtain the cost of stopping once a defect is detected, C_{SOD} (stop-on-defect):

$$C_{finished} = C_{net}((1-\gamma)(1-\alpha) + \gamma\beta) \tag{4}$$

$$C_{cancelled} = C_{scrap}((1-\gamma)\alpha + \gamma(1-\beta)) \tag{5}$$

$$C_{SOD} = \frac{C_{finished} + C_{cancelled}}{(1-\gamma)(1-\alpha)} \tag{6}$$

Note that the cost of producing a part whose production can be halted based on a monitoring method can be lower than, equal to, or greater than that of producing said part without

[2]As mentioned in Sections 3.5 and 3.6, scrap could be inspected to create data later used to train defect-assessment methods such as Hephaestus. This would apply only to a finite set of selected scrap parts. We thus do not account for it as part of the regular production costs. Our metric can be easily modified to account for that.

monitoring. Intuitively, if the part is scrap and correctly classified as such, the monitoring method will lead to reduced costs. Conversely, if the part is not scrap but incorrectly classified as such, the costs incurred in building a part later classified as scrap will be wasted. An interesting observation is that a monitoring method that always flags all objects as conforming leads to the same costs as a system without monitoring:

Lemma 4.3. *$C_{SOD} = C_{base}$ if the false positives α and false negatives β rate of a monitoring method are* 0 *and* 1*, respectively.*

This level of performance can serve as a baseline: any method that has a performance resulting in $C_{SOD} > C_{base}$ (that is, the cost of not using it is lower than actually using it) is not worth using. This outcome is not unrealistic, particularly for processes with very low scrap ratios, as shown in [5].

4.3 Trade-offs and relation between metrics

To minimize the cost in Equations 3 and 6, it is desirable to have a low false positive rate α, a low false negative rate β, and a low completion ratio CR for defective objects. Machine learning models can be tweaked to trade α for β (or vice versa). For Hephaestus, α and β are mostly influenced by the window size, W_s, creating though a dependency with CR and the latency too. On the one hand, a greater W_s leads to a larger time in producing results, both because of requiring a whole window to be complete before producing the very first result (see Algorithm 2), and also because of the increased processing time (see Section 4.1), directly increasing CR and latency. On the other hand, a larger W_s results in a larger number of outlier scores within each window, and thus a larger fingerprint before a classification is made. As we show later in the evaluation (Section 5.3), this often results in a more precise classification, i.e., a reduction in α and/or β. Effectively, W_s – together with W_a, see Lemma 4.2 – act as knobs that can be turned to find a balance between timeliness and precision, influencing in turn the impact in cost savings (depending on time, material, energy) by timely and accurate detections of defects.

5 Evaluation

5.1 Data Set Description

The evaluation uses one build for training and one for testing. The training set consists of 56 objects with two geometries: 28 of them are a five-tiered pyramid shape printed lying down (Figure 7, left), and the remaining 28 are a stack of three rectangular blocks printed standing upright (Figure 7, right).

The testing set contains 25 objects, all using the stacked block shape. The model is thus trained on a diverse set of shapes but evaluated solely on one geometry to assess generalization. Figure 8 shows the layout of both builds. The material, for both, is IN718, a nickel-based Inconel superalloy.

The five-tiered lying down pyramid objects consist of 187 layers, while the stacked block objects used for testing consist of 225 layers. In the stacked block objects, the bottom, middle, and top blocks consists of 150, 37, and 38 layers, respectively.

Both shapes have been designed by system experts, accounting for shape changes in consecutive layers and sharp corners in the printed objects that are sensitive in real-world printing jobs. Also, both builds use a range of process parameters, including scan speed, laser power, and hatch distance, both within and outside of recommended ranges. This variability ensures the presence of porosity-related defects, making the dataset representative of real-world conditions. The bottom 13 layers are excluded from training/testing as the thermal conductivity differs due to reduced surrounding material.

For ground truth labeling, porosity analysis is performed using an optical microscope. Before being inspected, the objects are cut across the middle, and etching is applied. The porosity can then be calculated as the fraction of the dark, empty area over the total area. The threshold used to classify an object as non-porous or porous is set to 0.5%, similar to [6] and [15]. The classification results are:

- Training set: 42 non-porous objects, 14 porous objects.
- Testing set: 18 non-porous objects, 7 porous objects.

This dataset design allows evaluating how well the model, trained on a mix of geometries, performs when tested on a single shape under varying process conditions.

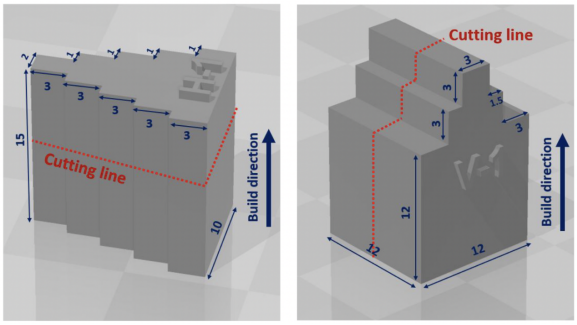
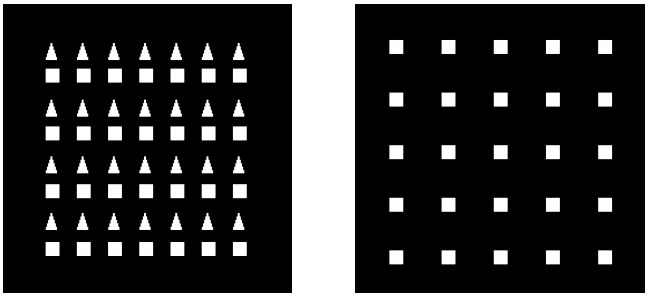


Figure 7: The two shapes used in the evaluation (picture taken from [4]). The cutting lines indicate where objects are cut/etched before being inspected for porosity using an optical microscope.

5.2 Hardware and Software Setup

The OT data has been collected from an EOS M290 machine, recording the near-infrared radiation generated by the laser beam. The recording is done layer by layer, integrating across the whole build, using a 5 MP CMOS camera on top of the printing chamber (Filter: 900 nm ± 12.5 nm, FoV: 2000 × 2000 px - pixels per layer, 125 µm/px).



(a) Layout of the first build. 28 objects of each shape for a total of 56 objects.

(b) Layout of the second build. 25 objects with a stack of three rectangular blocks.

Figure 8: Layout of the two builds on the powder bed. The OT images are from layer 10, the squares being the base of the stack shape, shown in Figure 7, right.

On average, the recoating process taking place in between the printing of each layer lasts three seconds. As such, we assume the latency threshold for Hephaestus to produce a timely classification is of three seconds.

The evaluation runs on an Intel Xeon E5-2637 v4 @ 3.50GHz (4 cores, 8 threads) server with 64 GB of RAM, with Ubuntu 18.04. Hephaestus is implemented in Python, using Numpy, OpenCV, and scikit-learn [23]. The timing is measured using the time module from the Python standard library.

For Hephaestus's parameters (see Section 3), various window sizes W_s are used, specifically 5, 25, 75, 105, and 212, with the window advance W_a set to half of W_s (i.e., 2, 12, 37, 52, and 106, respectively). The chosen W_s values range from small windows that provide early results starting from the 5th layer to very large windows that produce essentially a single post-completion result. The window advance W_a is fixed at half of W_s to ensure that each new result is generated only after at least half of the data in the window has been updated. Note that the largest W_s is set to 212, accounting for the aforementioned bottom 13 layers excluded from the analysis. The values for x, h, the number of bins, and k are 1, 5, 20, and 3, respectively. These were determined through five-fold cross-validation on the training set.

5.3 Quality of the Classification

As each window is processed, the ROC-AUC is calculated for that window. A specific ROC-AUC curve from the W_s=25 setting has been previously shown in Figure 2. Figure 9a shows a box plot of the ROC-AUC for the different W_s and W_a combinations. Generally, the median score grows as the window size increases, and a similar trend is observed for quartile values. Results do not further improve from a window size of 75 to 105, but reach score 1 once all the data from the build is processed together when $W_s = 212$ layers. As mentioned in Section 2.3, the rates of false positives, false negatives, and true positives, upon which the ROC-AUC values are computed, depend on both the classification method as well as the data under study. We note that, given the geometry of the object and the aforementioned number of layers of each block, a window size larger than 75 layers implies that layers from all three blocks will be present in at least one window, in this case resulting in a slight decrease in the classification accuracy.

5.4 Latency

Figure 9b shows how the latency changes for the different W_s and W_a combinations. The latency scales proportionally to window size, which is expected given Hephaestus's time complexity (see Section 4.1). The median is close to the 75th percentile for all window sizes: this is due to the geometry of the object. The time needed for Hephaestus to process a layer is proportional to the size of the cross-section of the object at the layer, and the base of the object is the widest and spans more layers than the rest of the object.

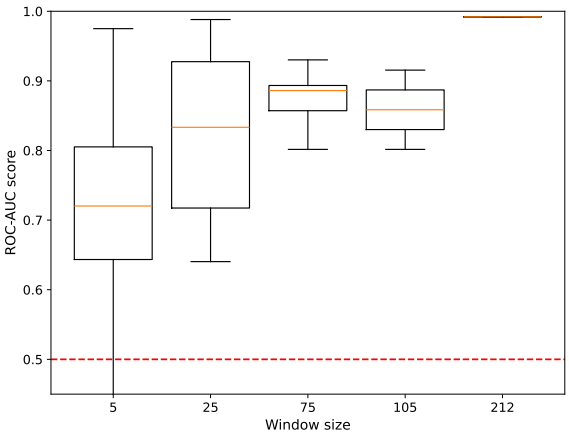
For $W_s < 75$, Hephaestus meets the 3-second deadline imposed by the recoating process. For some of the windows when $W_s = 75$ and all windows when $W_s > 75$, this is not the case, however. This means that the result is only available once the next layer has started processing. In addition, reducing W_s means the first result is available earlier, which is useful when making cancellation decisions to reduce cost.

5.5 Estimated impact on production cost

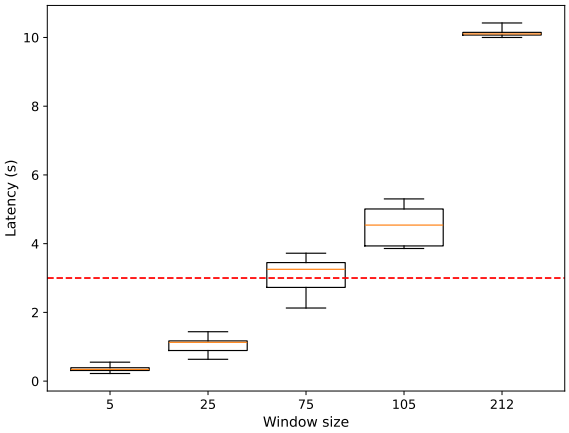
We assess the potential cost reduction by simulating a cancellation at a given stage, using the formulas in Section 4.2. For W_s values that meet the three-second deadline (i.e., $W_s = 5$ and $W_s = 25$), we assume that classification occurs immediately after receiving the final layer of the window and before the next layer begins printing. Accordingly, we set $CR = \frac{W_s+13}{225}$, where 13 represents the bottom layers excluded from the analysis. For the remaining W_s values, we assume classification occurs after the next layer has already started printing. In this case, we set $CR = \frac{W_s+14}{225}$. The cost parameters are set to the same values as those used for the machinery in the case study presented in [5]. To account for parameter stochasticity, we run the evaluation 10000 times and examine scrap rates from 10% to 40%.

As discussed in Section 4.2, a difference compared to [5] is the separation of the inspection and postprocessing costs, which we make since there is no need to inspect objects that do not finish construction, although other postprocessing costs (e.g., removal of objects from the build chamber) remain.

The result of this estimation is shown in Figure 10. It shows the fraction of the cost using early cancelation, C_{SOD}, and always finishing builds, C_{base}. Values greater than 1 indicate an increase in cost, while values lower than 1 indicate a decrease. Larger windows yield more accurate classification (Figure 9a)



(a) ROC-AUC for different window sizes. The red line represents guessing; higher scores are better.



(b) Latency for different window sizes. Less is better. The dashed red line is at 3s, the deadline.

Figure 9: Results for different window sizes.

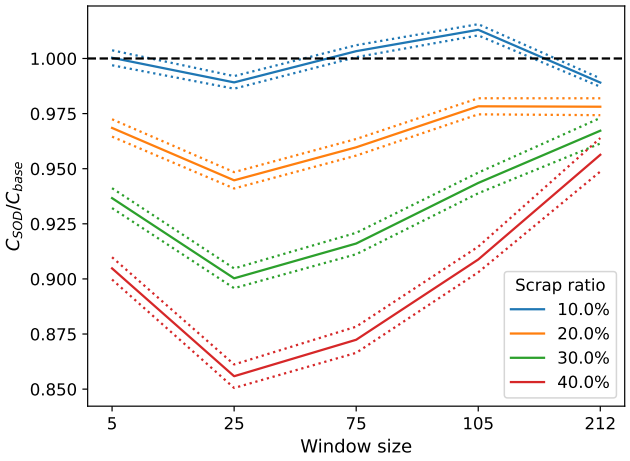


Figure 10: Estimated reduction in cost for different window sizes. The reduction is measured as the estimated cost of using Hephaestus compared to not using any monitoring method. Less is better, the dashed black line is 1.0 (i.e., the same as not using any monitoring method). The solid lines are the means, dotted lines are two standard deviations away from the mean.

but require larger parts of the object to be constructed. As a consequence, a trade-off exists between the classification's accuracy and the extent of object construction when selecting the optimal window size for cancellation decisions.

When the scrap ratio is 20% or more, the ideal observed setting is a window size of 25 layers. For the lower scrap ratio of 10% the best results are observed for a window size of 25 layers or the largest window.

Summarising, the chosen window size and advance combinations have an impact on both the ROC-AUC and latency, which behave as expected. Increasing W_s results in improved ROC-AUC and higher latency, influencing the estimated cost. While not all W_s values allow for the classification of porous objects within the strict three-second threshold, certain configurations, such as $W_s = 25$, achieve a strong balance between accuracy and efficiency. Specifically, this configuration maintains a median accuracy above 0.8, meets the required deadline, and can contribute to a reduction in production costs. These findings highlight the feasibility of real-time in-situ monitoring for timely defect detection and process optimization.

6 Related work

Interest in in-situ monitoring for PBF-LB has grown in recent years [9, 10]. Still, most research has been conducted outside of in-situ or real-time settings, not accounting for the temporal aspect of the problem (e.g., [1, 6, 14, 15, 17, 25]) nor providing solutions for timely detection of defective objects.

Although considering a different problem, some studies have explored online monitoring comparing Computer-Aided Design (CAD) models with OT images to detect deviations in object shape rather than porosity [10]. Similarly, [11] proposes an architecture for a stream processing framework for in-situ monitoring. Hephaestus contribution is complementary to that of [11], and could stimulate future work by being implemented within said framework to benefit from computational efficiency and automation provided by solutions building on fully-fledged Stream Professing Engines (SPEs), the platforms commonly used to run streaming applications.

It is also worth noting that existing research encompasses porosity detection across entire objects (cf. [6]), individual layers (cf. [18, 26, 27]), or small cuboids (cf. [7, 15]). By tweaking W_s, Hephaestus can operate on windows of layers of arbitrary

size and thus encapsulate all these different options, while also processing together several layers while incurring latencies lower than those reported in other related works focusing on intra-layer analysis only (e.g., up to 2.4 seconds in [27]).

Regarding data analysis in AM, [8, 9] comprehensively categorize monitoring problems and mention machine learning approaches as candidates, emphasizing, as also elaborated in e.g., [18, 19], the high data rates and the need for in-situ monitoring latencies to shift from seconds to milliseconds.

Cost reduction, one of the main drivers in AM monitoring methods, depends on both the accuracy and timeliness of said methods. Nonetheless, to the best of our knowledge, the only relevant study attempting to evaluate the impact monitoring methods have on cost is [5], which nonetheless does not focus on streaming monitoring (see Section 4.2). Other commonly employed classification metrics in the field, such as ROC-AUC [6, 15] and F1-score [30], are also not assessed within the context of streaming-based solutions.

7 Conclusions and Future Work

We proposed a method, named Hephaestus, to bring in-situ monitoring closer to production for AM printing processes. Namely, a streaming pipeline for conducting in-situ monitoring, one layer at a time, at the pace at which OT images are produced. As we show, the accuracy with which Hephaestus can timely detect defective objects improves as more data from the same object is processed together, a possibility that can be easily tuned in Hephaestus's streaming-based technique. Furthermore, we also show Hephaestus allows tuning the tradeoff between timely and accurate results to balance the two in the interest of cost minimization.

Future work includes the study of stream-processing methods that can further improve accuracy, in particular for small windows (which could be done by investigating the possibility for other streaming-based monitoring methods in the literature [7, 9, 13, 20, 21, 24–26] to be adapted to an AM-monitoring context). Also, there is a need to improve the timeliness (in particular for large windows), for instance, by incrementally processing each layer as it finishes. In particular, it is of interest to investigate methods that can produce results quickly and accurately with very small windows in the interest of fixing defects as they arise. Alongside improvements in accuracy and timeliness, further research is needed to transition from stream-based monitoring focused on early job cancellation to one enabling live job adjustments. This approach would allow real-time modifications, such as adjusting laser intensity to remelt the topmost layers and correct emerging defects [7], ensuring that monitoring data actively guides the object construction process. Finally, it can also be of interest to study how Hephaestus can support in-site monitoring in connection with other types of defects besides porosity, e.g., to identify spattering [26] promptly.

Acknowledgments

This work is partially supported by WASP-WISE project STRATIFIER, by Chalmers AoA frameworks Energy and Production, proj. INDEED, and WP "Scalability, Big Data and AI", respectively; by the Centre for Additive Manufacture–Metal (CAM2) supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA); by the TANDEM project within the framework of the Swedish Electricity Storage and Balancing Centre (SESBC), funded by the Swedish Energy Agency together with five academic and twenty-six non-academic partners; by the Swedish Research Council (Vetenskapsrådet) project "EPITOME" 2021-05424; by the Marie Skłodowska-Curie Doctoral Network project RELAX-DN, funded by the European Union under Horizon Europe 2021-2027 Framework Programme Grant Agreement number 101072456; by the European Union's Horizon 2020 research and Innovation Programme Additive Manufacturing using Metal Pilot Line (MANUELA) under grant agreement n. 820774 and Demonstration of Infrastructure for Digitalization enabling industrialization of Additive Manufacturing (DiDAM) research project (VINNOVA Project ID 2019-05591).

References

[1] Jan Akmal, Mihai Macarie, Roy Björkstrand, Kevin Minet, and Mika Salmi1. 2023. Defect detection in laser-based powder bed fusion process using machine learning classification methods. *IOP Conference Series: Materials Science and Engineering* 1296, 1 (Dec. 2023), 012013. doi:10.1088/1757-899X/1296/1/012013

[2] Jon Louis Bentley. 1990. K-d Trees for Semidynamic Point Sets. In *6th Symp. on Comp. Geometry*. ACM, 187–197. https://doi.org/10.1145/98524.98564

[3] Dechang Chen, Chang-Tien Lu, Yufeng Kou, and Feng Chen. 2008. On detecting spatial outliers. *Geoinformatica* 12 (2008), 455–475.

[4] Zhuoer Chen and Negar Panahi. 2024. [dataset] L-PBF Data. https://www.kaggle.com/ds/4644942. doi:10.34740/KAGGLE/DS/4644942

[5] Bianca Maria Colosimo, Simona Cavalli, and Marco Grasso. 2020. A cost model for the economic evaluation of in-situ monitoring tools in metal additive manufacturing. *International Journal of Production Economics* 223 (May 2020), 107532. doi:10.1016/j.ijpe.2019.107532

[6] Henry C. de Winton, Frederic Cegla, and Paul A. Hooper. 2021. A method for objectively evaluating the defect detection performance of in-situ monitoring systems. *Additive Manufacturing* 48 (Dec. 2021), 102431. doi:10.1016/j.addma.2021.102431

[7] Shuo Feng, Zhuoer Chen, Benjamin Bircher, Ze Ji, Lars Nyborg, and Samuel Bigot. 2022. Predicting laser powder bed fusion defects through in-process monitoring data and machine learning. *Materials & Design* 222 (Oct. 2022), 111115. doi:10.1016/j.matdes.2022.111115

[8] Marco Grasso and Bianca Maria Colosimo. 2017. Process defects and *in situ* monitoring methods in metal powder bed fusion: a review. *Measurement Science and Technology* 28, 4 (April 2017), 044005. doi:10.1088/1361-6501/aa5c4f

[9] M Grasso, A Remani, A Dickins, B M Colosimo, and R K Leach. 2021. In-situ measurement and monitoring methods for metal powder bed fusion: an updated review. *Measurement Science and Technology* 32, 11 (Nov. 2021), 112001. doi:10.1088/1361-6501/ac0b6b

[10] Maria Grazia Guerra, Marco Lafirenza, Vito Errico, and Andrea Angelastro. 2023. In-process dimensional and geometrical characterization of laser-powder bed fusion lattice structures through high-resolution optical tomography. *Optics & Laser Technology* 162 (2023), 109252.

[11] Vincenzo Gulisano, Marina Papatriantafilou, Zhuoer Chen, Eduard Hryha, and Lars Nyborg. 2022. Towards data-driven additive manufacturing processes. In *Proceedings of the 23rd International Middleware Conference Industrial Track*. ACM, Quebec Quebec City Canada, 43–49. doi:10.1145/3564695.3564778

[12] Bastian Havers, Romaric Duvignau, Hannaneh Najdataei, Vincenzo Gulisano, Marina Papatriantafilou, and Ashok Chaitanya Koppisetty. 2020. DRIVEN: A framework for efficient Data Retrieval and clustering in Vehicular Networks. *Future Generation Computer Systems* 107 (2020), 1–17.

[13] Amir Keramatian, Vincenzo Gulisano, Marina Papatriantafilou, and Philippas Tsigas. 2020. PARMA-CC: Parallel Multiphase Approximate Cluster Combining. In *Proceedings of the 21st International Conference on Distributed Computing and Networking*. ACM, Kolkata India, 1–10. doi:10.1145/3369740.3369785

[14] Cody S. Lough, Tao Liu, Xin Wang, Ben Brown, Robert G. Landers, Douglas A. Bristow, James A. Drallmeier, and Edward C. Kinzel. 2022. Local prediction of Laser Powder Bed Fusion porosity by short-wave infrared imaging thermal feature porosity probability maps. *Journal of Materials Processing Technology* 302 (April 2022), 117473. doi:10.1016/j.jmatprotec.2021.117473

[15] Sina Malakpour Estalaki, Cody S. Lough, Robert G. Landers, Edward C. Kinzel, and Tengfei Luo. 2022. Predicting Defects in Laser Powder Bed Fusion Using In-Situ Thermal Imaging Data and Machine Learning. *SSRN Electronic Journal* (2022). doi:10.2139/ssrn.4073603

[16] Gunther Mohr, Simon J. Altenburg, Alexander Ulbricht, Philipp Heinrich, Daniel Baum, Christiane Maierhofer, and Kai Hilgenberg. 2020. In-Situ Defect Detection in Laser Powder Bed Fusion by Using Thermography and Optical Tomography—Comparison to Computed Tomography. *Metals* 10, 1 (Jan. 2020), 103. doi:10.3390/met10010103

[17] Gunther Mohr, Simon J Altenburg, Alexander Ulbricht, Philipp Heinrich, Daniel Baum, Christiane Maierhofer, and Kai Hilgenberg. 2020. In-situ defect detection in laser powder bed fusion by using thermography and optical tomography—comparison to computed tomography. *Metals* 10, 1 (2020), 103.

[18] Mohammad Montazeri, Abdalla R. Nassar, Alexander J. Dunbar, and Prahalada Rao. 2020. In-process monitoring of porosity in additive manufacturing using optical emission spectroscopy. *IISE Transactions* 52, 5 (May 2020), 500–515. doi:10.1080/24725854.2019.1659525

[19] William Mycroft, Mordechai Katzman, Samuel Tammas-Williams, Everth Hernandez-Nava, George Panoutsos, Iain Todd, and Visakan Kadirkamanathan. 2020. A data-driven approach for predicting printability in metal additive manufacturing processes. *Journal of Intelligent Manufacturing* 31, 7 (Oct. 2020), 1769–1781. doi:10.1007/s10845-020-01541-w

[20] Hannaneh Najdataei, Vincenzo Gulisano, Philippas Tsigas, and Marina Papatriantafilou. 2022. pi-Lisco: parallel and incremental stream-based point-cloud clustering. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. ACM, Virtual Event, 460–469. doi:10.1145/3477314.3507093

[21] Hannaneh Najdataei, Yiannis Nikolakopoulos, Vincenzo Gulisano, and Marina Papatriantafilou. 2018. Continuous and Parallel LiDAR Point-Cloud Clustering. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Vienna, 671–684. doi:10.1109/ICDCS.2018.00071

[22] Dimitrios Palyvos-Giannas, Marina Papatriantafilou, and Vincenzo Gulisano. 2022. Research Summary: Deterministic, Explainable and Efficient Stream Processing. In *Proceedings of the 2022 Workshop on Advanced Tools, Programming Languages, and PLatforms for Implementing and Evaluating Algorithms for Distributed Systems* (Salerno, Italy) *(ApPLIED '22)*. Association for Computing Machinery, New York, NY, USA, 65–69. doi:10.1145/3524053.3542750

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[24] Claudia Schwerz and Lars Nyborg. 2021. Linking In Situ Melt Pool Monitoring to Melt Pool Size Distributions and Internal Flaws in Laser Powder Bed Fusion. *Metals* 11, 11 (Nov. 2021), 1856. doi:10.3390/met11111856

[25] Claudia Schwerz and Lars Nyborg. 2022. A neural network for identification and classification of systematic internal flaws in laser powder bed fusion. *CIRP Journal of Manufacturing Science and Technology* 37 (May 2022), 312–318. doi:10.1016/j.cirpj.2022.02.010

[26] Claudia Schwerz, Ahmad Raza, Xiangyu Lei, Lars Nyborg, Eduard Hryha, and Håkan Wirdelius. 2021. In-situ detection of redeposited spatter and its influence on the formation of internal flaws in laser powder bed fusion. *Additive Manufacturing* 47 (Nov. 2021), 102370. doi:10.1016/j.addma.2021.102370

[27] Luke Scime, Derek Siddel, Seth Baird, and Vincent Paquit. 2020. Layer-wise anomaly detection and classification for powder bed additive manufacturing processes: A machine-agnostic algorithm for real-time pixel-wise semantic segmentation. *Additive Manufacturing* 36 (2020), 101453.

[28] Erik Sievers. 2023. *Data Analysis for Defect Monitoring in Additive Manufacturing - Applying Machine Learning to Predict Porosity in L-PBF*. Master's thesis. Chalmers University of Technology. http://hdl.handle.net/20.500.12380/306019

[29] Steven S. Skiena. 2017. *The Data Science Design Manual*. Springer International Publishing, Cham. doi:10.1007/978-3-319-55444-0

[30] Ziyad Smoqi, Aniruddha Gaikwad, Benjamin Bevans, Md Humaun Kobir, James Craig, Alan Abul-Haj, Alonso Peralta, and Prahalada Rao. 2022. Monitoring and prediction of porosity in laser powder bed fusion using physics-informed meltpool signatures and machine learning. *Journal of Materials Processing Technology* 304 (June 2022), 117550. doi:10.1016/j.jmatprotec.2022.117550

[31] Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. 2005. The 8 requirements of real-time stream processing. *ACM Sigmod Record* 34, 4 (2005), 42–47.