IAC-25-C4.4.11-x99822

Scalable Anomaly Detection in High-Speed Combustion Imaging

Hakan Akdag[a, b*], Oliver Assenmacher[a], Claudia Comito[c], Andrija Dabanović[d], Georg Poppe[e], Alexander Rüttgers[a]

[a] *German Aerospace Center (DLR), Institute of Software Technology, High-Performance Computing Department, Linder Höhe, 51147 Cologne, Germany, hakan.akdag@dlr.de, oliver.assenmacher@dlr.de, alexander.ruettgers@dlr.de*

[b] *Cologne University of Applied Sciences (TH Köln), Institute of Information Science, Faculty of Information Science and Communication Studies, Claudiusstraße 1, 50678 Cologne, Germany, hakan.akdag@th-koeln.de*

[c] *Forschungszentrum Jülich, Jülich Supercomputing Centre, Wilhelm-Johnen-Straße, 52428 Jülich, Germany, c.comito@fz-juelich.de*

[d] *German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Spacecraft Department, Lilienthalplatz 7, 38108 Brunswick, Germany, andrija.dabanovic@dlr.de*

[e] *German Aerospace Center (DLR), Responsive Space Cluster Competence Center, Launch Segment, Eugen-Sänger-Straße 50, 29328 Faßberg, Germany, georg.poppe@dlr.de*

[*] *Corresponding Author*

Abstract

In aerospace-combustion research, advanced high-speed imaging enables the observation of dynamic processes, such as turbulence–flame interactions and instability mechanisms, that may have a significant influence on propulsion systems. To analyze such complex and data-intensive recordings, we leverage high-performance computing (HPC) for scalable anomaly detection in high-speed combustion video data using Python. Specifically, we developed a parallelized implementation of the Local Outlier Factor (LOF), a well-established density-based algorithm for detecting local (i.e., small-scale) anomalies, to support massively parallel execution across multiple GPUs. This allows for the identification of critical instabilities and localized extinction events that are otherwise difficult to detect, due to the immense volume and complexity of data, in the investigated high-speed video containing 116,487 images. Our results illustrate that memory and runtime constraints—which in the standard implementation of the LOF would require to evaluate and store the result of more than 10 billion image comparisons on a single computation node—can be significantly mitigated. By enabling scalability to substantially larger datasets, our approach shows the potential of HPC-driven anomaly detection to overcome existing computational bottlenecks in order to advance aerospace research.

Keywords: Hybrid Propulsion, Anomaly Detection, Machine Learning, Computer Vision, Combustion, High-Performance Computing

1. Introduction

High-speed imaging plays a crucial role in aerospace-combustion research, providing detailed insights into transient phenomena such as turbulence-flame interactions and instability mechanisms critical for propulsion systems. However, identifying non-obvious irregularities, such as localized extinction or transient flow instabilities, is particularly challenging. These phenomena often manifest only in subtle, short-lived deviations that are easily missed in a manual inspection of the data, especially in large datasets containing tens or even hundreds of thousands of frames. A well-established algorithm that supports the identification of such anomalies is the Local Outlier Factor (LOF) [1]. This algorithm identifies local anomalies based on density deviations with respect to neighboring data points, making it particularly suitable for detecting small-scale context-specific outliers in high-dimensional feature spaces. Its successful application to combustion-related video data has been demonstrated in prior work, c.f., for instance, Ref. [2]. However, the computational cost of applying the LOF to large-scale high-speed video data can quickly exceed the capabilities of a single computation node, both in terms of runtime and memory consumption. This is primarily due to the pairwise distance computations required between all data points, leading to a quadratic growth in complexity. There are sub-quadratic alternatives to this approach, such as the implementation in the well-known scikit-learn library [3] using tree-based algorithms, but these methods do not scale to higher feature dimensions. Since high-speed video data is characterized by both a high feature dimension, which equals the image resolution, and a high number of data points, which equals the total number of frames, the standard approach within the Python data science/machine learning software stack, such as the LOF, becomes infeasible.

Within the Python ecosystem, there is a growing need for scalable and parallelizable solutions that can handle large data volumes efficiently. In the context of high-performance computing (HPC), this translates to implementations that are capable of leveraging multi-GPU, multi-node architectures to enable massive data throughput and parallel analysis. In this work, *we present a scalable anomaly detection framework based on a parallelized implementation of the LOF algorithm*, tailored for distributed memory systems with GPU acceleration. Our approach enables the processing of high-speed combustion videos consisting of 116,487 frames, a scale at which conventional algorithms on a single workstation would break down.

We apply this algorithm to high-speed videos from slab combustion experiments with hybrid rocket fuels that were performed as part of the ATEK project [4] at the German Aerospace Center (DLR). These experiments aim at improving our understanding of the complex combustion process in hybrid rocket engines, for example by enabling optical measurements during the combustion process. The highly turbulent and unsteady combustion observed in these experiments makes the use of high-speed video cameras very desirable but the analysis of the resulting video data requires scalable algorithmic solutions. We demonstrate that our approach reliably detects both *sequence anomalies* (e.g., sustained flame detachment) and *point anomalies* (e.g., brief disruptions in flame structure) while scaling to several dozen GPUs if necessary. This highlights the capability of our HPC-based framework to extract subtle outliers from complex, noise-dominated sequences.

This work is structured as follows: In Sect. 2 we describe the experimental setup of the investigated high-speed combustion videos. In order to detect anomalies in this data set, we introduce the reader to the algorithmic approach for large-scale anomaly detection used in this work in Sect. 3. We present our results of identified anomalies in Sect. 4 and close with a concise summary and outlook in Sect. 5.

2. Experimental Setup

The high-speed images analyzed in this work were recorded during a static firing test of a high-pressure laboratory engine ARIEL. The engine can operate with a variety of two-dimensional solid fuel slabs and hydrogen peroxide mass flows of up to 200 g/s. It can be run at pressures up to 2.5 MPa and permits optical measurements, including infrared and high-speed imaging, as well as pyrometer measurements. The quartz-glass windows are actively cooled with Helium. Fig. 1 presents a CAD model of the combustion chamber. The engine enables systematic characterization of different fuel compositions and additives,

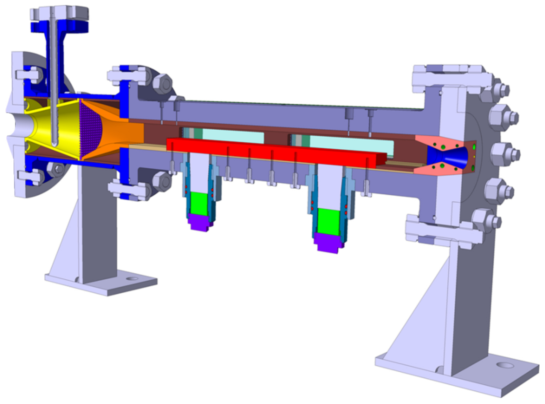


Figure 1: CAD model of the ARIEL hybrid laboratory combustion chamber.

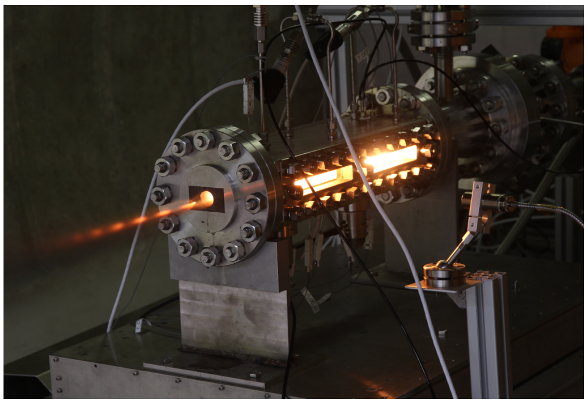


Figure 2: Image of the static firing of the ARIEL hybrid laboratory combustion chamber.

oxidizer mass fluxes and chamber pressures. Thermocouples are mounted on the upper side of the chamber and along the fuel block to record the axial temperature distribution within the solid fuel. Two pressure sensors are also installed on the upper side. The fuel block thickness is measured with a pair of ultrasonic transducers [4].

The reflected signals from the fuel-gas interface are used to estimate transient regression rates. High-speed images, with a resolution of 1024×192 pixels, were captured with a PHOTRON FASTCAM Mini AX200 through the upstream window with a time interval of 10^{-4} s between the analyzed images (10000 fps). An image of the test setup is given in Fig. 2. Due to the brightness of the chamber during operation, the minimal shutter time of 1.05 µs was used. The test burn lasted around 8.2 s, while the high-speed camera produced a total of 116,487 images with a color depth of 12 bit and generated 32 GiB of raw data. Due to the volume of data, manual analysis would require a significant amount of time. Consequently, HPC-driven anomaly detection was applied to demonstrate the feasibility of automated processing for large-scale datasets.

The resulting models can support post-processing of high-frequency measurements.

3. Algorithm for outlier detection

There are many unsupervised algorithms for anomaly detection in the literature. In Sect. 3.1, the basic principles of the Local Outlier Factor (LOF) algorithm [1] are described. Rüttgers and Petrarolo [2] compared the LOF algorithm with other potential algorithms, such as Elliptic Envelope, One-Class Support Vector Machines (One-Class SVM), and Isolation Forest, for a similar application to the one considered here: high-speed video imaging of rocket combustion. In this comparison the LOF algorithm achieved the best results. In our setting, anomalies of interest are instabilities in the combustion process that result in visible differences in the combustion structures. The algorithm in our application evaluates entire images for anomalies, so visible deviations in the image are necessary.

3.1. Local Outlier Factor

We first introduce the reader to the general formulation of the LOF. For this purpose, let $A = \{a_1, \dots, a_N\} \subset \mathbb{R}^{m\times n}$ be a time series of images with a resolution of $n \cdot m$ pixels and $d(\cdot,\cdot)$ a distance metric such as the Euclidean distance. Then, the LOF can be evaluated considering:

k-distance and neighborhood

The k-distance of an image a is the distance to its k-th nearest neighbor, defined according to

$$k\text{-dist}(a) = \min_{\epsilon\in\mathbb{R}}\Big\{\big\{b \neq a : d(a,b) \leq \epsilon\big\} \geq k\Big\} \tag{1}$$

for an image $b \in A$. The (inclusive) k-neighborhood $N_k(a)$ of a point a is the set of all points within the distance to its k-th nearest neighbor,

$$N_k(a) = \{b \neq a \,|\, d(a,b) \leq k\text{-dist}(a)\}\,. \tag{2}$$

Reachability distance

The LOF algorithm leverages the reachability distance

$$\text{reach-dist}(a,b) \;=\; \max\big\{k\text{-dist}(b), d(a,b)\big\} \tag{3}$$

between two points a and b to mitigate the effects of close neighbors, in the sense that due to this lower bound, distances cannot collapse within dense micro-clusters, allowing for more stable density estimates.

Local reachability density (LRD)

The inverse of the average reachability distance from point a to its neighbors defines the local density estimate, i.e,

$$\text{lrd}_k(a) = \frac{\text{card}\,(N_k(a))}{\sum_{b\in N_k(a)} \text{reach-dist}(a,b)}\,, \tag{4}$$

where the cardinality $\text{card}(N_k(a))$ denotes the number of elements in $N_k(a)$.

Local Outlier Factor

Finally, the LOF compares the local density around a to the densities of its neighbors as

$$\text{LOF}_k(a) \;=\; \frac{\sum_{b\in N_k(a)} \text{lrd}_k(b)}{\text{card}\,(N_k(a)) \cdot \text{lrd}_k(a)}. \tag{5}$$

For the interpretation of Eq. (5), we consider the three different scenarios:

$$\begin{aligned} \text{LOF}_k(a) &< 1 \rightarrow a \;\text{ is in a denser region,}\\ \text{LOF}_k(a) &\approx 1 \rightarrow a \;\text{ has a similar density,}\\ \text{LOF}_k(a) &> 1 \rightarrow a \;\text{ has a lower density}\end{aligned}$$

than its neighbors. Consequently, the larger $\text{LOF}_k(a)$ the more likely a can be considered a potential candidate for an outlier. Conversely, if $\text{LOF}_k(a) \leq 1$, then the image is a potential inlier or even a cluster center.
In practice, the final selection of outliers requires choosing a threshold $\tau > 1$. This can, for instance, be done with a fixed cutoff, a data-driven rule such as flagging the top 0.1 % of scores, or by tuning τ on a validation set to optimize a target metric. Beyond hard thresholds, several methods replace the LOF-type scores by corresponding probabilities. One example are Local Outlier Probabilities (LoOP) [5] that map LOF-scores to $[0, 1]$ to improve interpretability. While such extensions are outside the scope of this work, our implementation presented in Sect. 3.3 can be adapted accordingly.
Furthermore, we note that the correct value of the hyperparameter k depends on the application at hand. The original authors of the LOF algorithm [1] suggest using a range of values for k and taking the maximum over the LOF values corresponding to this range of values for k instead of relying on a single value. This approach can be integrated in our implementation with little additional effort because the k-neighborhoods of each point only have to be computed once for the upper bound of the range of values for k. The neighborhoods for all smaller values of k by only using a subset of the larger neighborhood. In the application considered in this paper, however, we observe very similar results for a relatively large range of values for k. Therefore, we fix a single value instead using a range of values. In Sect. 4.1, the choice of k for this specific application is discussed in more detail.

3.2. The Heat library for high-performance data analytics

The Helmholtz Analytics Toolkit (Heat)[1] [6, 7] is a Python framework designed for large-scale data analysis on high-performance computing (HPC) systems. It allows researchers to process datasets that are too large to fit into the memory of a single computer by distributing the computation across multiple machines, or nodes, in a cluster. The core principle is that each node runs the same program

[1] https://github.com/helmholtz-analytics/heat

on a different piece of the overall dataset and periodically exchanges necessary information with the other nodes.

At its core, Heat uses a data structure, the distributed n-dimensional array or DNDarray, to represent the entire dataset. In practice, this large array is physically partitioned into smaller chunks, with each chunk residing in the memory of a different compute node. This partitioning happens along a single, user-specified dimension, which ensures the computational workload is balanced. Such a data-centric approach is crucial for problems where the entire dataset must be treated as a coherent whole, rather than being broken down into independent tasks. On each node, Heat uses the PyTorch [8] library as its computational engine, which enables hardware acceleration on both CPUs and GPUs transparently.

To coordinate the work across nodes, Heat manages the required data transfers using the Message Passing Interface (MPI), a standard for parallel computing, via the mpi4py [9] library. For the researcher, this complexity is hidden behind a simple programming interface that mimics NumPy and scikit-learn, widely-used Python libraries for numerical operations and machine learning. Heat automatically handles the complex, multi-dimensional data exchanges between nodes efficiently, preserving the data's structure without requiring manual reorganization. This design makes it straightforward to adapt existing Python code for large-scale, parallel execution. However, for massive datasets, a direct parallelization of the LOF algorithm still faces the critical bottleneck of the pairwise distance calculation, which remains an issue of quadratic complexity. This motivated the development of a more advanced, memory-efficient approach, as detailed in the next section.

3.3. Scalable Implementation

The bottlenecks of LOF-based anomaly detection are primarily associated with the calculation and storage of the full pairwise distance matrix, which scales *quadratically* with the number of data points n. In this work, we provide an implementation of the LOF for the Python array ecosystem with two main advancements compared to the conventional, single-node CPU implementation:

1. Parallelized implementation with GPU acceleration
2. Memory-efficient distance matrix for nearest neighbors

The parallel, GPU-accelerated version was achieved by implementing the LOF algorithm described in [1] on top of the Python library Heat (see Sect. 3.2). Optimizing for scalability, i.e., setting up a distance matrix that satisfies the memory constraints for arbitrarily large data sets, is the more intricate task.

We approach this problem by considering the following: While the full distance matrix has complexity $O(n^2)$, a naive data-parallel computation can reduce it to $O(n^2/p^2)$, where p indicates the number of processes. Still, due to the quadratic scaling in n, the distance matrix can easily exceed the capacities of the any cluster with an increased amount of data points. To tackle this issue, our main idea is to dynamically reduce the size of the distance matrix during the parallelized calculation to the k-nearest neighbors, which are according to Eq. (2) and Eq. (5) the only entries in the distance matrix required to compute the LOF. Thus, we replace this quadratic scaling in n by a linear scaling of the form $O(nk/p^2)$ following the steps illustrated in Fig. 3. In this manner, the distance matrix for the k-nearest neighbors is evaluated piecewise in a parallelized fashion and, if sufficiently many processes are available, does not exceed the memory constraints.

Although this implementation is able to distribute the memory consumption of the distance matrix across multiple nodes, it still may set challenging demands on an HPC cluster, especially when compute resources are limited. To address this, we introduce an additional parameter c that allows us to choose a trade-off between memory demand and computation time. In practical terms, c defines how many chunks (or iterative steps) are used to compute the distance matrix on each process. A larger c means that each chunk is smaller and requires less memory, but more iterations are needed and thus the overall runtime may increase. Conversely, a smaller c reduces the number of iterations and speeds up execution, but requires more memory per process. In this sense, our algorithm starts with computing the first c rows of the distance matrix, i.e., it requires only memory for a matrix of complexity $O\big(nk/(p^2 \cdot c)\big)$, reduces it to the k-nearest neighbors, and stores it separately. After this procedure has been repeated for all iterations, the chunks are put together to restore the whole distance matrix.[2]

4. Results

In this section, we present the results of our scalable LOF, as described in Sect. 3, when applied to the high-speed combustion data detailed in Sect. 2. To this end, we first discuss the ability of the LOF to detect subsequence and point anomalies and then investigate how our algorithm scales with increasing data load.

4.1. Application of the LOF

In order to apply the LOF, we treat each image with 1024×192 pixels and three color channels as a single data point with $1024 \cdot 192 \cdot 3 = 589824$ features. These data points are organized in one single vector with 116,487 entries, one for each image, and each element is normalized to the range $[0, 1]$. The result of the LOF thus tells us which of these images is likely to be considered an outlier,

[2]Note that the parameter c reduces peak memory per GPU and does not replace parallel resources. For the datasets in this study, this parameter has a negligible effect and is henceforth set to 1, but it ensures scalability to problems with an increasing number of data points.

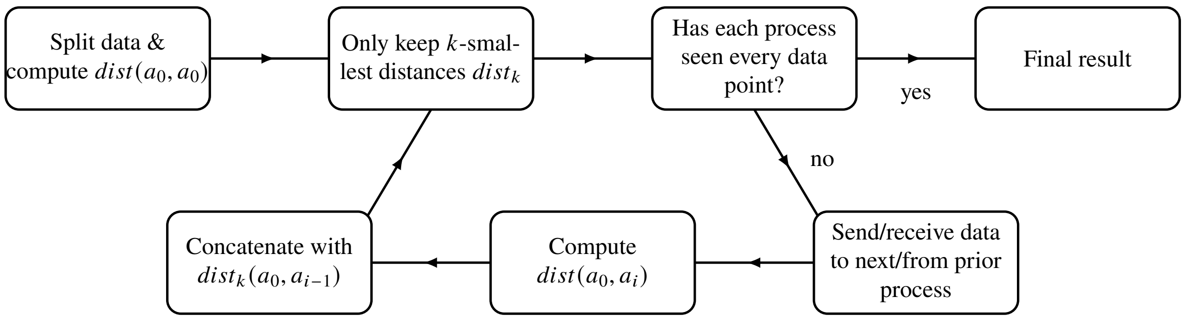

Split data & compute $dist(a_0, a_0)$
Only keep k-smallest distances $dist_k$
Has each process seen every data point?
yes
Final result
no
Send/receive data to next/from prior process
Compute $dist(a_0, a_i)$
Concatenate with $dist_k(a_0, a_{i-1})$

Figure 3: Algorithmic sketch to compute the distance matrix in a memory-safe way. Except for the splitting of the data, each operation in this figure is to be understood as an operation that runs in parallel on each single process. The quantity a_0 indicates the raw input data that lies on the respective process after splitting, while a_i indicates the newly received data from a different process at communication step i. The quantity $dist_k$ describes the distance matrix in which only the k-smallest distances for each row of the distance matrix are kept.

without immediately telling us where exactly the anomaly is located in the detected single image. We will address this point in the next subsection.

To compute the LOF scores we applied the Euclidean distance and have chosen the hyperparameter k from Eq. (5) to be 100, which deviates by roughly one order of magnitude from the empirically more common value of 20, cf. Ref. [1]. This choice is motivated by the observation that, if the value of k is too small, the LOF is too sensitive to noise in the data. This means that the algorithm is for too small k not capable of identifying local patterns, as these often include more than 20 frames due to the high frequency with which frames are captured. On the other hand, if k is chosen too large, e.g., $k = 1000$, the algorithm shifts towards a global approach and is not able to identify anomalies which are only visible in local patterns in the image. We furthermore note that other values between $k = 50$ and $k = 500$ as well as different distance measures, like the Manhattan distance,[3] may also be possible and lead to similar results. However, a sophisticated fine tuning is out of scope of this paper.

4.2. Anomaly detection

We follow the nomenclature in Refs. [10, 11] to categorize the detected anomalies according their *dimensionality*, *locality*, and *length*. As the LOF score depends on a large number of image features (i.e., pixel values), anomalies cannot be described by a single parameter. Thus, we refer to the dimensionality as *multivariate*. Moreover, due to the nature of the algorithm and our choice of number of nearest neighbors k, which is small in comparison to the number of images in the full dataset, the detected anomalies are also *local* by construction. Hence, all identified anomalies in our case are both multivariate and local. We illustrate the distribution of LOF scores in Fig. 4. From the set of detected candidates, we select two representative examples for further analysis: one dominant *subsequence anomaly* and one *point-like anomaly*. Their respective LOF scores, along with those of their surrounding data points, are shown in Fig. 5. The subsequence anomaly (left) is clearly distinguishable from the point-like anomaly (right) by the broader distribution of elevated outlier scores. We use the term *point-like*, since only a few consecutive time steps exhibit elevated scores.[4] In our case, only three LOF scores in the point-like anomaly lie above 1.1, compared to roughly 50 points for the subsequence anomaly.

While the LOF scores provide useful indications of potential anomalies, their interpretation still requires manual inspection. As part of a hybrid approach combining algorithmic detection with human evaluation, we selected two points before, during, and after the detected outliers, as highlighted in Fig. 5, and show the corresponding images in Fig. 6a and 6b, respectively. To gain further insight into which local patterns contributed to the LOF-based anomaly detection, we provide the heat maps in Fig. 7a and 7b. The latter exemplarily illustrates the contribution of each pixel to the distance matrix that determines the LOF score. Each pixel value $p_{ij}(t)$ in the heat maps at time step t represents the mean of the distance to all k-nearest neighbors, i.e.,

$$p_{ij}(t) = \frac{1}{k}\sum_n d_{ij}(n,t)\,, \qquad (6)$$

where the sum runs over all k-nearest neighbors of n and $d(n,t)$ is the Euclidean distance matrix

$$d_{ij}(n,t) = \sqrt{\sum_{l=0}^{2}\left(a_{ijl}(t) - a_{ijl}(n)\right)^2} \qquad (7)$$

[3]In principle, any distance measure can be applied in the LOF. In Ref. [2], for example, a structural dissimilarity measure (DSSIM) is applied. But due to the high computational cost of the latter, we rely on the standard Euclidean distance.

[4]The interpretation of a point anomaly depends, of course, strongly on the corresponding temporal resolution.

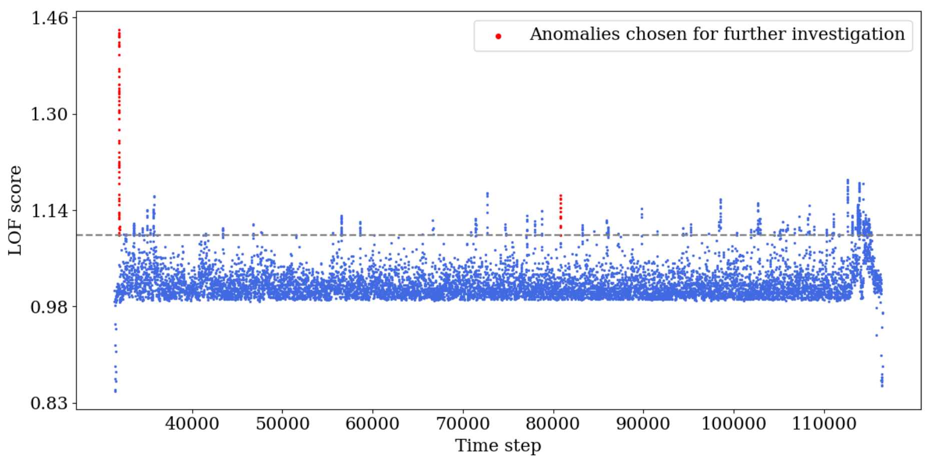
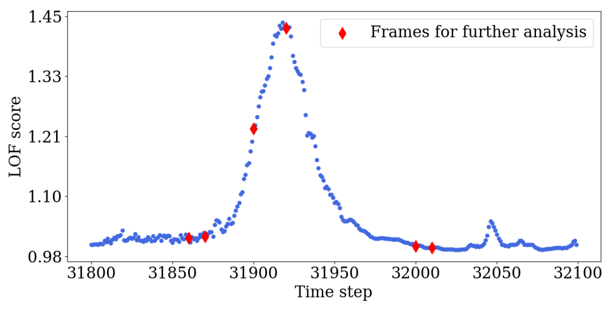
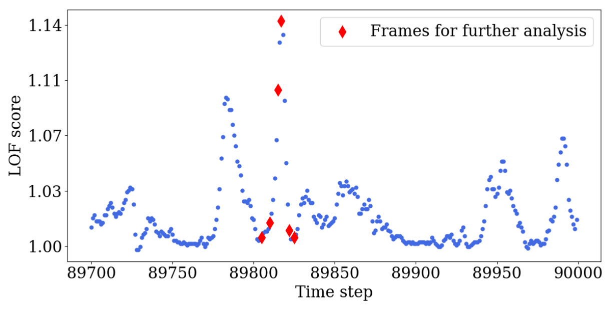


Figure 4: LOF scores for each frame in the high-speed combustion video. Frames in the very beginning of the ignition phase are left out, as they contain mainly black pixel. The dashed gray line may serve as an empirical threshold to select outlier-candidates. To simplify the illustration, we depict only every tenth point below this threshold. The anomalies that are investigated further in the main text and shown in more detail in Fig. 5 are indicated as red dots.



(a) Subsequence anomaly



(b) Point anomaly

Figure 5: LOF scores for a chosen subsequence (left) and point anomaly (right). The red diamonds indicate frames that are analyzed further in Fig. 6a and 6b, respectively.

between the images at time step n and t.[5] The matrix a_{ijl} holds all pixel values at coordinates i, j with color channels l.

Interpreting these results, we observe in Fig. 5 that the LOF identified the ignition phase as a subsequence anomaly, a result that is expected and thus underlines the feasibility of the algorithm.[6] As can be judged from the corresponding heat map in Fig. 7a, the anomaly is due to the fact that the whole upper surface of the fuel block starts to burn with a small flame. At all previous time steps, this area contains mainly black pixels, while at later time steps this area contains flames with a significantly higher volume.

On the other hand, the point-like anomaly in Fig. 6b clearly shows a transient flame flash at the tip of the block, which leads to the increased LOF score for the corresponding few frames, as additionally underlined by Fig. 7b.

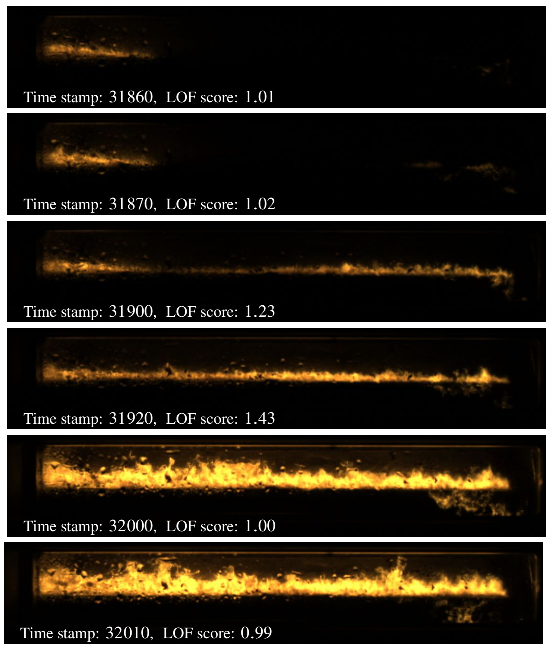
4.3. Scalability tests

To demonstrate the scalability of the parallel LOF algorithm, we conduct tests in which both the dataset size and the number of processes are increased stepwise. In the HPC context, such tests are known as *weak scaling experiments*. Weak scaling is of particular interest for our use case, since the volume of input data increases naturally with longer recordings or higher-resolution combustion videos. In these scenarios, it is essential that the runtime of the anomaly detection pipeline remains approximately constant when we use more resources to analyze the increased amount of data. Thus, weak scaling experiments allow us to assess the algorithm's ability to handle ever-larger datasets.

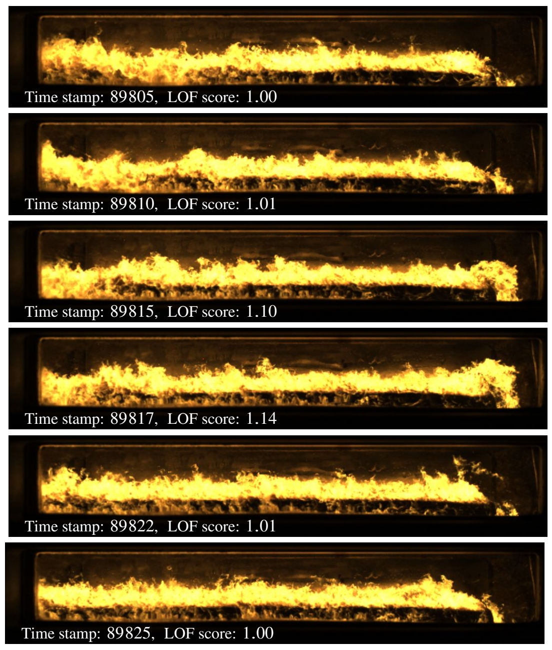
We conduct our weak scaling experiments on the *terrabyte*

[5]Note that this computation does not need HPC and can be carried out with standard numerical analysis tools, e.g. NumPy [12].

[6]All frames captured before the time step of roughly 30,000 belong to the very beginning of the ignition phase and are effectively all black.

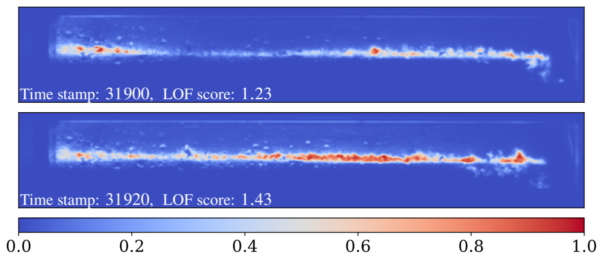

Time stamp: 31860, LOF score: 1.01
Time stamp: 31870, LOF score: 1.02
Time stamp: 31900, LOF score: 1.23
Time stamp: 31920, LOF score: 1.43
Time stamp: 32000, LOF score: 1.00
Time stamp: 32010, LOF score: 0.99

(a) Subsequence anomaly

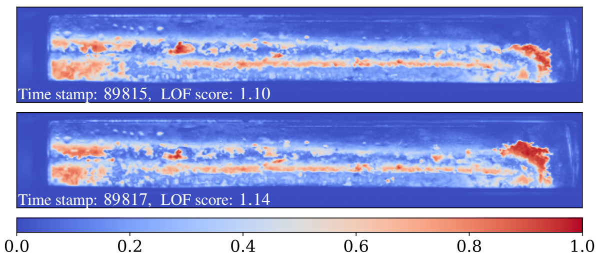

Time stamp: 89805, LOF score: 1.00
Time stamp: 89810, LOF score: 1.01
Time stamp: 89815, LOF score: 1.10
Time stamp: 89817, LOF score: 1.14
Time stamp: 89822, LOF score: 1.01
Time stamp: 89825, LOF score: 1.00

(b) Point anomaly

Figure 6: Identified anomalies according to their LOF scores. From top to bottom, the sequences show two frames captured shortly before the onset of the anomaly, during the anomaly, and immediately after its occurrence.



(a) Subsequence anomaly



(b) Point anomaly

Figure 7: Illustration of the contribution of each pixel to the respective LOF score, calculated with Eq. (6) and normalized to the range between 0 (pixel without deviation compared to k-nearest neighbors) and 1 (pixel with highest deviation) in each image.

cluster at DLR.[7] For this purpose, we consider datasets with the same number of features as the high-speed combustion video (589,824), while varying the number of data points per GPU. The amount of nearest neighbors to compute the LOF is set to the same value (100) as in Sect. 4. During each run we record the maximum GPU memory allocation, averaged over all GPUs, using the performance analysis framework *perun* [13]. For the number of data points that are evaluated with each GPU, we consider two scenarios: a scaling with the problem complexity and a linear scaling. When increasing the amount of data, the actual workload per GPU scales with the complexity of the problem. As the latter is mainly driven by the distance matrix, which scales quadratically with the number of data points, the dependence of the total number data points n on the number of GPUs p is given by

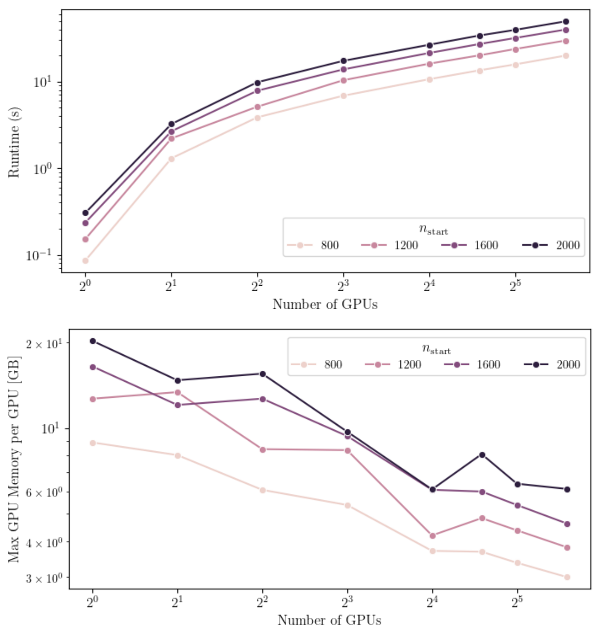
$$n = n_{\text{start}} \cdot \sqrt{p}\,, \tag{8}$$

where the parameter n_{start} is varied during our tests. We choose $n_{\text{start}} \in \{800, 1200, 1600, 2000\}$ and $p \in \{1, 2, 4, 8, 16, 24, 32, 48\}$. In addition, we consider a linear scaling, i.e.,

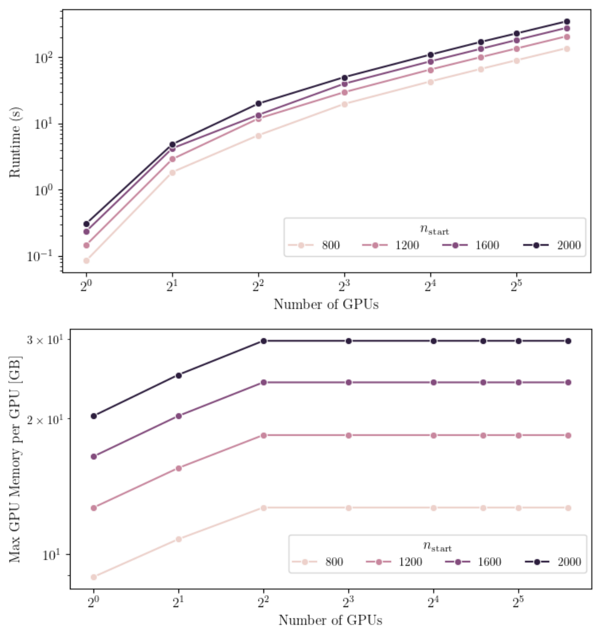
$$n = n_{\text{start}} \cdot p\,, \tag{9}$$

which allows us to investigate the same setup for a larger amount of total data points.

[7]Each GPU-node of the terrabyte cluster contains 2 Intel Xeon Gold 6336Y 24 cores 185 W 2.4 GHz, 1 TiB RAM, and 4 Nvidia HGX A100 80 GiB 500 W GPUs.



(a) Scaling with problem complexity



(b) Linear scaling

Figure 8: Results of the weak scaling experiments as described in the main text of Sect. 4.3. The left (right) side illustrates how the scales with problem complexity (linearly), while the upper (lower) panel depicts the runtime (peak memory consumption per GPU) of the scalable LOF algortihm.

The results, as shown in Fig. 8, demonstrate that the peak memory consumption per GPU remains constant in the linear scenario, while complexity-based scaling leads to a further reduction, thereby confirming that our parallel LOF implementation exhibits a very convenient weak-scaling behavior with respect to memory.

Regarding the runtime we observe a slight increase, which is caused by the additional communication overhead when transferring the data between the GPUs. The runtime of the LOF algorithm for the largest data set in Fig. 8 with 96,000 data points is approximately 350s. In comparison, an evaluation with the scikit-learn implementation on a single CPU-only node takes—of course depending on the specific hardware—several hours.

5. Summary & Outlook

In this work we presented a parallelized implementation of the Local Outlier Factor (LOF) algorithm applied for high-speed combustion imaging. Our approach addresses the main bottlenecks of the standard LOF, namely the quadratic complexity of the distance matrix and the associated memory requirements, by combining GPU acceleration with a memory-efficient computation of the distance matrix. By applying our framework to a dataset of 116,487 images from hybrid rocket combustion experiments at DLR, it successfully identified both sequence- and point-like anomalies that correspond to physically meaningful phenomena such as the ignition phase and transient flame irregularities. Beyond anomaly detection, we demonstrated that the algorithm exhibits favorable scalability on modern GPU clusters. In particular, weak scaling experiments on the *terrabyte* system confirmed that the peak memory consumption per GPU exhibits a stable behavior, thereby enabling the analysis of ever larger datasets without exceeding hardware limits. This highlights the potential of high-performance computing to bring established machine learning methods, such as the LOF, to application domains with extreme data volumes.

As a proof of concept, our study shows that HPC-driven anomaly detection can provide valuable insights into turbulent combustion processes and offers a promising tool for advancing aerospace research. Future work could, for instance, focus on either extending the framework towards more advanced outlier scoring methods (e.g., probabilistic variants such as LoOP) or on further applications for critical systems in aerospace, in which the identification of anomalies are vital.

Acknowledgments

This research was carried out under the project *Advanced Technologies for High Energetic Atmospheric Flight of Launcher Stages* (ATHEAt) by the German Aerospace Center (DLR). The authors gratefully acknowledge the computational and data resources provided through the joint high-performance data analytics (HPDA) project "terrabyte" of the German Aerospace Center (DLR) and the Leibniz Supercomputing Center (LRZ). The contribution of HA has been developed in the project PLan_CV. Within the funding Programme FH-Personal, the project PLan_CV (reference number 03FHP109) is funded by the German Federal Ministry of Research, Technology, and Space (BMFTR) and Joint Science Conference (GWK). This work was supported by the European Space Agency through the Open Space Innovation Platform https://ideas.esa.int as a Early Technology Development Agreement and carried out under the Discovery Program ESA Early Technology Development (Research Agreement No. 4000144045/24/NL/GLC/ov). *Disclaimer: The view expressed in this publication can in no way be taken to reflect the official opinion of the European Space Agency.* The authors gratefully acknowledge the computational and data resources provided through the joint high-performance data analytics (HPDA) project "terrabyte" of the German Aerospace Center (DLR) and the Leibniz Supercomputing Center (LRZ).

References

[1] M. Breunig, H. Kriegel, R. Ng and J. Sander, *Lof: identifying density-based local outliers*, *SIGMOD Rec.* **29** (2000) 93–104.

[2] A. Rüttgers and A. Petrarolo, *Local anomaly detection in hybrid rocket combustion tests*, *Exp. Fluids* **62** (2021) .

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12** (2011) 2825.

[4] G. Poppe, O. Božić, S. May and N.M. Bierwagen, *Characterization of regression rate and combustion process in a high-pressure 2d hybrid rocket engine with optical access*, in *Proceedings of the International Astronautical Congress, IAC*, Oktober, 2018, https://elib.dlr.de/122751/.

[5] H. Kriegel, P. Kröger, E. Schubert and A. Zimek, *Loop: local outlier probabilities*, in *Proc. 18th ACM Conf. Inf. Knowl. Manag.*, CIKM '09, (New York, NY, USA), p. 1649–1652, Association for Computing Machinery, 2009, 10.1145/1645953.1646195.

[6] M. Götz, C. Debus, D. Coquelin, K. Krajsek, C. Comito, P. Knechtges et al., *HeAT – a Distributed and GPU-accelerated Tensor Framework for Data Analytics*, in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 276–287, IEEE, 2020, 10.1109/BigData50022.2020.9378050.

[7] F. Hoppe, J.P. Gutiérrez Hermosillo Muriedas, M. Tarnawa, P. Knechtges, B. Hagemeier, K. Krajsek et al., *Engineering a large-scale data analytics and array computing library for research: Heat*, *Electronic Communications of the EASST* **83** (2025) .

[8] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky et al., *PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation*, in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, Apr., 2024, https://docs.pytorch.org/assets/pytorch2-2.pdf.

[9] L. Dalcin and Y.-L.L. Fang, *mpi4py: Status Update After 12 Years of Development*, *Computing in Science & Engineering* **23** (2021) 47.

[10] A. Blázquez-García, A. Conde, U. Mori and J.A. Lozano, *A review on outlier/anomaly detection in time series data*, 2020.

[11] K. Kotowski, C. Haskamp, J. Andrzejewski, B. Ruszczak, J. Nalepa, D. Lakey et al., *European space agency benchmark for anomaly detection in satellite telemetry*, 2024.

[12] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau et al., *Array programming with NumPy*, *Nature* **585** (2020) 357.

[13] J.P. Gutiérrez Hermosillo Muriedas, K. Flügel, C. Debus, H. Obermaier, A. Streit and M. Götz, *perun: Benchmarking energy consumption of high-performance computing applications*, in *Euro-Par 2023: Parallel Processing*, J. Cano, M.D. Dikaiakos, G.A. Papadopoulos, M. Pericàs and R. Sakellariou, eds., (Cham), pp. 17–31, Springer Nature Switzerland, 2023.