energies

MDPI

Article

Implementation of Thermal Event Image Processing Algorithms on NVIDIA Tegra Jetson TX2 Embedded System-on-a-Chip

Bartłomiej Jabłoński *, Dariusz Makowski and Piotr Perek

Department of Microelectronics and Computer Science, Lodz University of Technology, 90-924 Łódź, Poland; dmakow@dmcs.pl (D.M.); pperek@dmcs.pl (P.P.)
* Correspondence: jablonskiba@dmcs.pl

Abstract: Advances in Infrared (IR) cameras, as well as hardware computational capabilities, contributed towards qualifying vision systems as reliable plasma diagnostics for nuclear fusion experiments. Robust autonomous machine protection and plasma control during operation require real-time processing that might be facilitated by Graphics Processing Units (GPUs). One of the current aims of image plasma diagnostics involves thermal events detection and analysis with thermal imaging. The paper investigates the suitability of the NVIDIA Jetson TX2 Tegra-based embedded platform for real-time thermal events detection. Development of real-time processing algorithms on an embedded System-on-a-Chip (SoC) requires additional effort due to the constrained resources, yet low-power consumption enables embedded GPUs to be applied in MicroTCA.4 computing architecture that is prevalent in nuclear fusion projects. For this purpose, the authors have proposed, developed and optimised GPU-accelerated algorithms with the use of available software tools for NVIDIA Tegra systems. Furthermore, the implemented algorithms are evaluated and benchmarked on Wendelstein 7-X (W7-X) stellarator experimental data against the corresponding alternative Central Processing Unit (CPU) implementations. Considerable improvement is observed for the accelerated algorithms that enable real-time detection on the embedded SoC platform, yet some encountered limitations when developing parallel image processing routines are described and signified.

Keywords: embedded image processing; graphics processing unit; tegra; thermal event; real-time algorithm; plasma diagnostics

Citation: Jablonski, B.; Makowski , D.; Perek, P. Implementation of Thermal Event Image Processing Algorithms on NVIDIA Tegra Jetson TX2 Embedded System-on-a-Chip. *Energies* **2021**, *14*, 4416. https://doi.org/10.3390/en14154416

Academic Editors: Wiseman Yair and Donghwa Shin

Received: 31 May 2021
Accepted: 19 July 2021
Published: 22 July 2021

1. Introduction

Infrared (IR) thermal imaging nowadays is a non-destructive monitoring and measurement technique widely applied in both the industry and research. Thermography enables one to observe the radiating surface of an object with a temperature greater than 0 K [1]. Under real conditions, the image captured by an IR detector has to be pre-processed to take into account the observed body emissivity as well as device nonuniformities. With the advances in thermographic cameras and computational capabilities, it has become feasible to capture and process high-resolution images in real-time. Currently, dedicated vision systems are one of the major diagnostics in large-scale physics experiments, especially in nuclear fusion projects [2–5].

1.1. Image Plasma Diagnostics

Current objectives for image diagnostics systems mainly concern machine protection, plasma control and physics exploitation. The first exemplary use case, utilising images in nuclear fusion, covers algorithms for detecting and tracking thermal events occurring throughout a plasma discharge (see Figure 1). In the Wendelstein 7-X (W7-X), the algorithms identify events such as hotspots, leading edges, strike-lines, surface layers that exceed the operational limits of the stellarator [6–10]. Based on the detected thermal events, the risk is estimated, and if it exceeds the threshold, the interlock system is triggered [2]. Similar

challenges are faced in the W Environment in Steady-state Tokamak (WEST) with persistent and transient hotspots detection and categorisation [11,12]. Moreover, strike-line images together with toroidal current are inputs for a Convolutional Neural Network (CNN) that outputs lower and upper divertor coils controls. The instrumentation is realised in order to modify a strike-line position and shape to protect Plasma Facing Components (PFCs) [13].

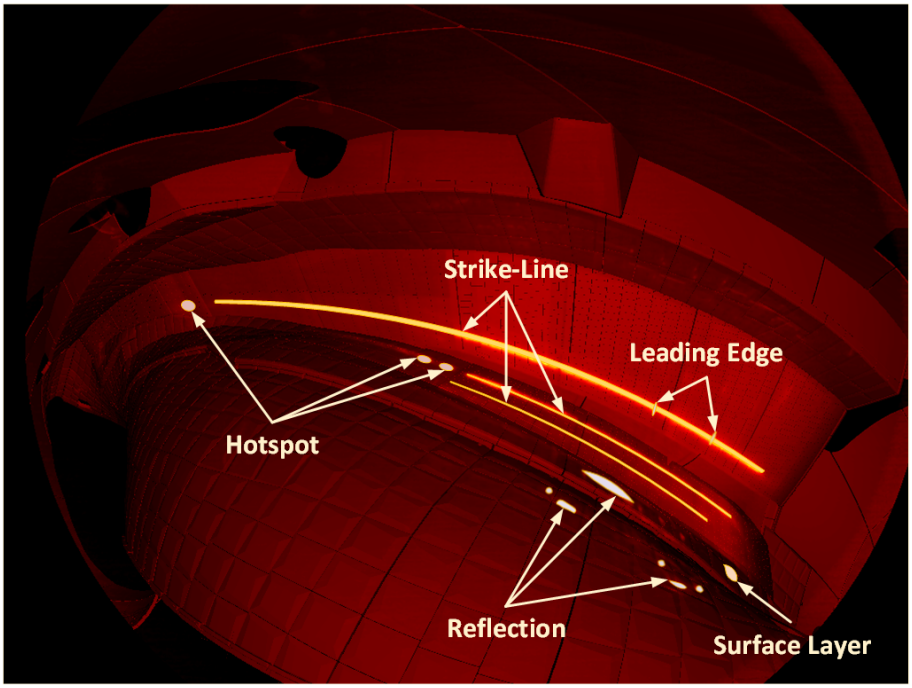


Figure 1. Exemplary thermal events in the Wendelstein 7-X (W7-X). The authors overlaid thermal events onto the stellarator model and coloured surface temperature from the available dataset for visualisation purposes, similarly to the figure in [2].

Physical quantities such as heat flux are computed using surface temperature and THermal Energy Onto DivertOR (THEODOR) code [14,15]. The estimated heat flux together with Gaussian spatial filtering and moving average filtering allows one to characterise strike-lines in the W7-X [6].

Visible light images processing is applied in areas where IR systems have issues with emissivity, absorption and dynamic range. Visible spectrum images are utilised to determine plasma discharge location, shape and timing in the W7-X [16,17] as well as to reconstruct a plasma boundary in the Experimental Advanced Superconducting Tokamak (EAST) [4]. Hotspots are also analysed by detecting clusters of bright pixels and tracking them across consecutive frames in the Joint European Torus (JET) [3]. Furthermore, images aid in the analysis and detection of anomalies such as falling debris, Multifaceted Asymmetric Radiation From the Edges (MARFEs) and Edge Localised Modes (ELMs) [11].

Besides, archived data is further utilised for offline analysis by physicists and diagnosticians. Isolation of thermal events and determining their ontology allow one to automate the plasma control strategies [6]. Reliable, automated and real-time systems are the basis for steady-state operation in future fusion experiments.

1.2. General-Purpose Computing on Graphics Processing Unit

According to the requirements for the future viewing systems in the ITER project, data volume for processing from a single camera was estimated to [18]:

- 600 MB/s for a 16-bit IR image with resolution 2000 × 1500 at 100 Hz
- 12 GB/s for an 8-bit visible image with resolution 4000 × 3000 at 1 kHz

As a consequence, for robust autonomous machine protection and plasma control during operation, significant computational power is necessary. Acceleration of compu-

tations might be provided by the general-purpose computing on Graphics Processing Units (GPUs) [19]. GPUs are flexible hardware accelerators for intensive computations that offer the best price-performance ratio compared to Field-Programmable Gate Arrays (FPGAs) [20] and Digital Signal Processors (DSPs) [21]. The highest performance increase is observed in applications with massive parallelism and minimal data transfers between a device and a host, i.e., Central Processing Unit (CPU). Although FPGAs might provide better power consumption-performance ratio, the increased implementation time and complexity is inconvenient in the rapidly changing thermonuclear fusion systems. Moreover, GPUs are common in plasma diagnostics setups [4,7,10], i.e., current configurations typically include GPUs as image processing hardware accelerators. As a consequence, it is significantly easier to integrate described algorithms and techniques. However, it is advisable to incorporate FPGAs for simpler and established pre-processing algorithms, e.g., correction and calibration steps.

In this paper, two thermal event detection algorithms for nuclear fusion were implemented to investigate the suitability of low-power embedded NVIDIA Tegra System-on-a-Chip (SoC) platforms for real-time processing. Conclusions are derived based on the performed benchmarks and the comparison between embedded GPU and CPU implementations.

2. Development and Evaluation System Setup

2.1. NVIDIA Tegra System-on-a-Chip

NVIDIA Tegra SoC integrates an embedded CPU and GPU into a single integrated circuit. NVIDIA Tegra devices are low-cost, small and energy-efficient, yet powerful embedded platforms. NVIDIA Jetson TX2 [22] is an embedded system equipped with an NVIDIA Tegra destined for edge computing (see Table 1). It runs Linux for Tegra (L4T) Operating System (OS) based on Ubuntu 18.04 that is a part of dedicated JetPack SDK (https://developer.nvidia.com/embedded/jetpack/, accessed on 21 July 2021) 4.5.1 incorporating Compute Unified Device Architecture (CUDA) 10.2 toolkit for building accelerated solutions.

Table 1. NVIDIA Jetson TX2 technical specification.

Feature	Description
GPU	256-core Pascal @ 1300 MHz (memory shared with RAM)
CPU	Arm Cortex-A57 (4 cores) @ 2 GHz + NVIDIA Denver2 (2 cores) @ 2 GHz
RAM	8 GB 128-bit LPDDR4 @ 1866 MHz | 59.7 GB/s
PCIe	Gen 2
Power	7.5 W/15 W

The developed and evaluated algorithms are optimised for the NVIDIA Jetson TX2 developer board [23] shown in Figure 2.

MicroTCA.4 is a strongly adapted architecture in large-scale scientific experiments, including nuclear fusion projects such as ITER and W7-X. The MicroTCA.4 form factor enables a modular design that supports acquisition from several fast high-resolution cameras and frame grabbers equipped with a Camera Link interface in a single chassis. In addition, MicroTCA offers embedded system management and health monitoring capabilities [5,24,25]. Due to the low-power consumption, NVIDIA Tegra embedded GPUs are interfaced through additional modules [26,27] with MicroTCA.4 architecture in order to deliver a complete processing chain. MicroTCA.4 slots are limited to 80 W, and the average power consumption for a Tesla series Peripheral Component Interconnect Express (PCIe) GPU is 250 W. As a consequence, many discrete GPUs are inapplicable.

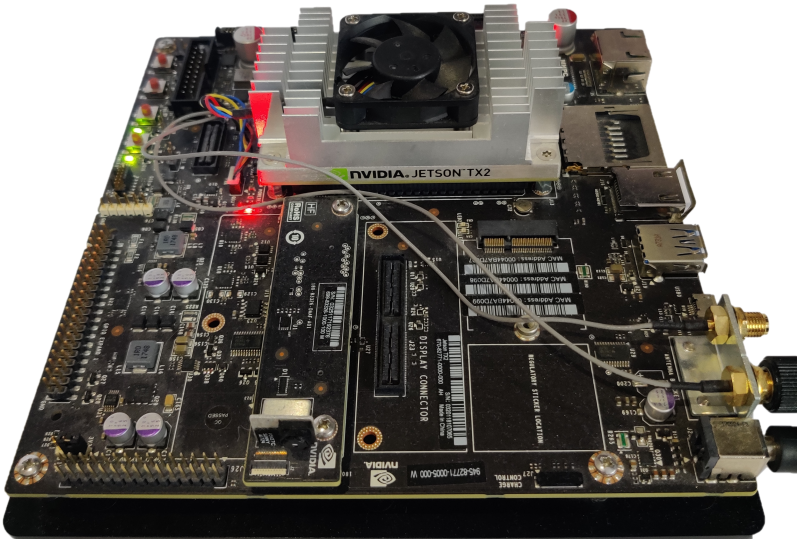


Figure 2. NVIDIA Jetson TX2 developer board.

2.2. Computer Vision Software

OpenCV (https://opencv.org/platforms/cuda/, accessed on 21 July 2021) 4.4.0 was mainly used to implement image processing algorithms for both CPU and GPU. It is a leading open-source, optimised and cross-platform library for Computer Vision (CV) tasks. The library was compiled with CUDA support to enable Application Programming Interface (API) for graphics accelerators. Moreover, other NVIDIA libraries such as VisionWorks (https://developer.nvidia.com/embedded/visionworks/, accessed on 21 July 2021) 1.6, Nvidia Performance Primitives (NPP) (https://docs.nvidia.com/cuda/npp/, accessed on 21 July 2021) 10.2.89 and Thrust (https://developer.nvidia.com/thrust/, accessed on 21 July 2021) 1.9.7 were evaluated to be feasible to perform additional GPU operations that complement missing or suboptimal OpenCV functionalities.

The programming language used for development was C++17 compiled with GNU Compiler Collection (GCC) 7.5.0 as it supports the abovementioned libraries; besides, it allows one to provide custom CUDA kernels as well as manually manage the GPU's memory through the CUDA runtime API.

2.3. Wendelstein 7-X Experimental Data

The experimental data used for implementing and evaluating algorithms comes from W7-X stellarator experiments (Max Planck Institute for Plasma Physics, Greifswald, Germany). The program identifier of the archived calibrated pulse is 20171114.053, and the camera port is AEF20. Each image, captured by IRCam Caleo 768kL IR camera, has a 14-bit depth with a resolution of 1024×768. In addition, frames have assigned epoch timestamps as well as dynamic attributes such as Region of Interest (RoI), sensor temperature and frame rate. The data is stored in Hierarchical Data Format version 5 (HDF5) and contains supplementary pixel-wise scene model information such as Field of View (FoV) of the camera, Computer-Aided Design (CAD) model of the stellarator, hierarchical PFC identifiers. The HDF5 supports n-dimensional datasets, bundles together metadata and features high I/O performance; therefore, it is widely applied throughout scientific fields.

Data flow in the evaluated NVIDIA Jetson TX2 configuration is visualised in Figure 3. Since the GPU and CPU share the same physical memory, a pinned buffer is accessible from both devices. However, page-locked memory is uncached on NVIDIA Jetson TX2; therefore, a frame is efficiently copied with Direct Memory Access (DMA) to a cacheable device memory buffer for further computations [28].

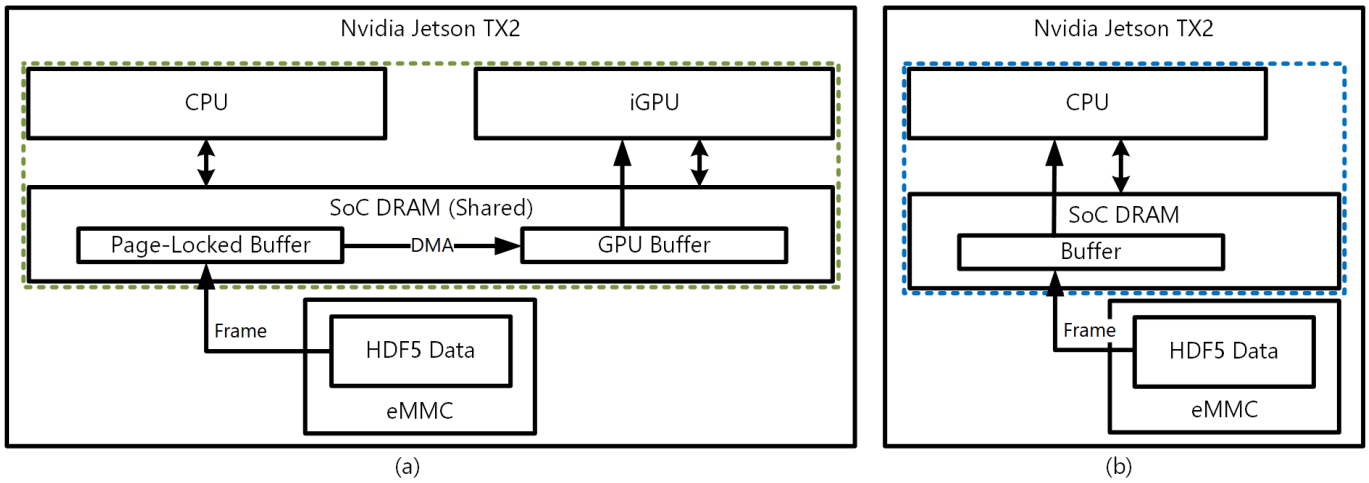


Figure 3. (**a**) Data flow diagram in the configuration involving the embedded Graphics Processing Unit (GPU). (**b**) Data flow diagram in the embedded Central Processing Unit (CPU) only configuration. Dotted lines mark the sections included in the evaluation measurements.

3. Algorithms Implementation

This section describes the main ideas behind the detection of thermal events and GPU-accelerated algorithm implementation details. The source code of the implemented algorithms and benchmarks is available on GitLab (https://gitlab.dmcs.pl/jablonskiba/mdpi-energies-2021-thermal-events-detection-tegra, accessed on 20 May 2021).

3.1. Scene Model Pre-Processing

The scene model data is transformed to the format utilised in the detection algorithms prior to any computations. Hierarchical PFC identifiers are encoded in 32-bits in the scene model. Thus, they are decoded to extract top-level components (see Figure 4) according to the predefined components coding [29]. In order to obtain a continuous mask, without any undefined pixels, the morphological closing with a 3×1 kernel and median blur with a 3×3 kernel operations are applied (see Figure 4). The undefined pixels inside PFCs might be observed in Figure 5 in the form of empty spaces between tiles. The components segmentation allows one to identify components affected by thermal events and utilise the masks to execute certain operations on individual parts.

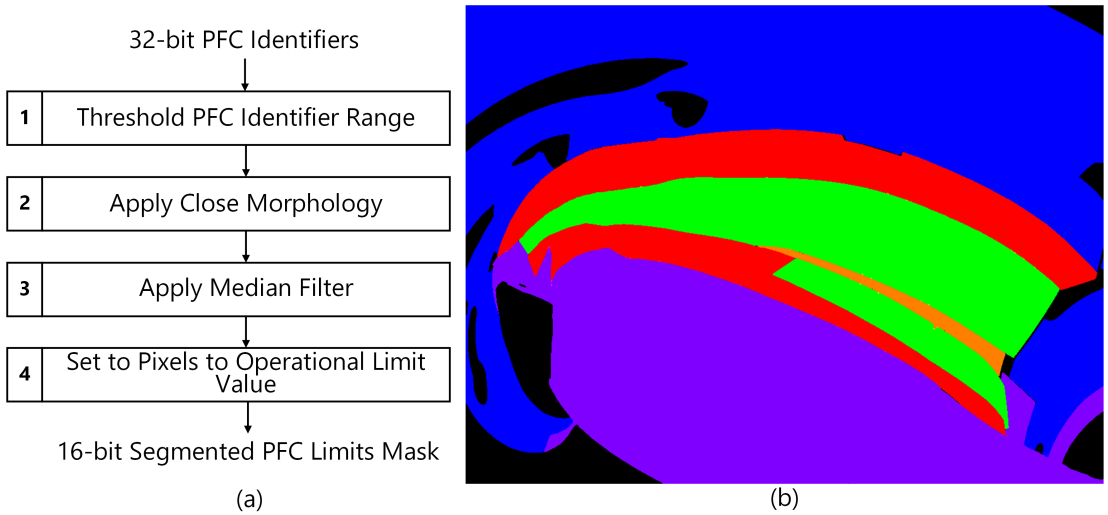


Figure 4. (**a**) Plasma Facing Component (PFC) segmentation pre-processing pipeline. (**b**) Pixel-wise PFC segmentation mask.

Furthermore, 2D pixel coordinates are transformed to 3D Cartesian coordinates based on the Lookup Table (LUT) generated from the scene model's x, y, z mappings.

3.2. Overload Hotspots Detection Procedure

The first algorithm detects and tracks overload hotspots, which exceed the operational limits of the PFCs [7] shown in Figure 5.

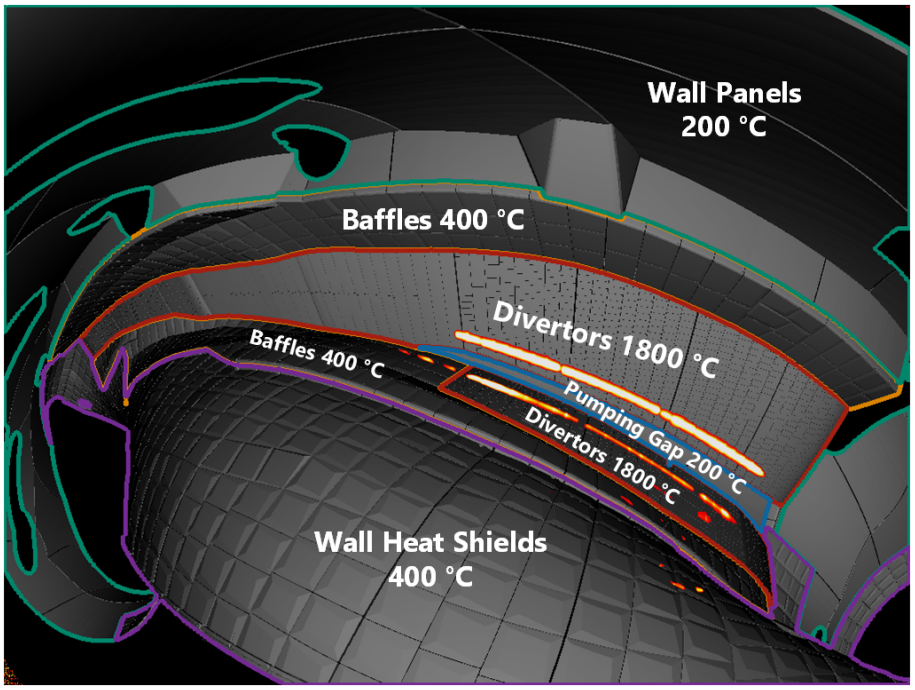


Figure 5. PFCs operational limits during Operational Phase (OP) 1.2.

The processing pipeline for the hotspot detection algorithm is indicated in Figure 6. Topological structural analysis extracts contours of overheating blobs [30]. The initial frame upload to a GPU is performed concurrently with the execution of steps one and two. A frame is divided row-wise into four chunks, and each chunk is processed asynchronously on a different CUDA stream. Copy and execution overlap is possible in this case as the host memory is pinned and executed computations are embarrassingly parallel. The transfer between steps four and five utilises unified memory with prefetching in order to reduce the migration overhead by enabling cached access to the same buffer for both the CPU and GPU. The entire host buffer is prefetched with *cudaStreamAttachMemAsync (. . . , cudaMemAttachHost)* before its use. As it is the asynchronous call to CUDA API, synchronisation is necessary if the host immediately processes the buffer. For GPU prefetching, *cudaStreamAttachMemAsync (. . . , cudaMemAttachGlobal)* is invoked. It is noteworthy that the *cudaStreamAttachMemAsync* function shall be used as the *cudaMemPrefetchAsync*, advised for discrete GPUs, is unsupported on Tegra devices [28].

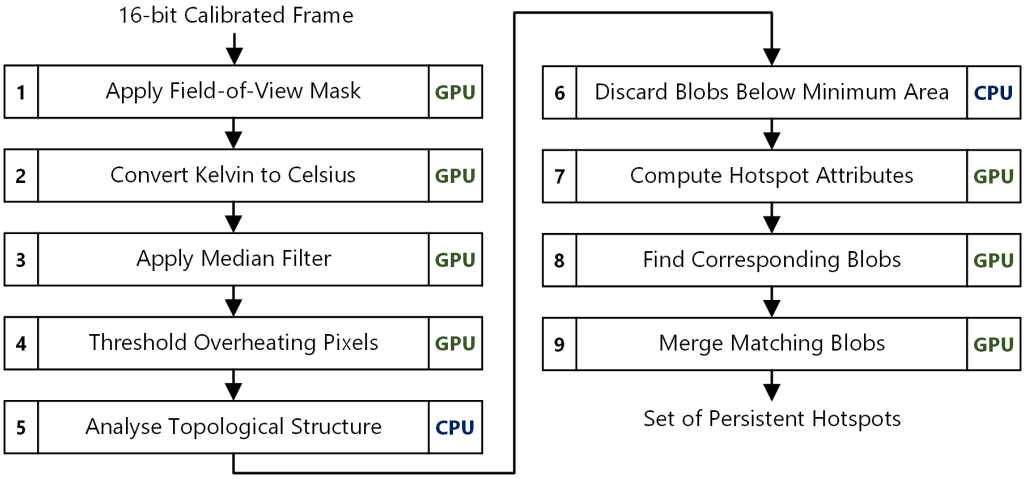
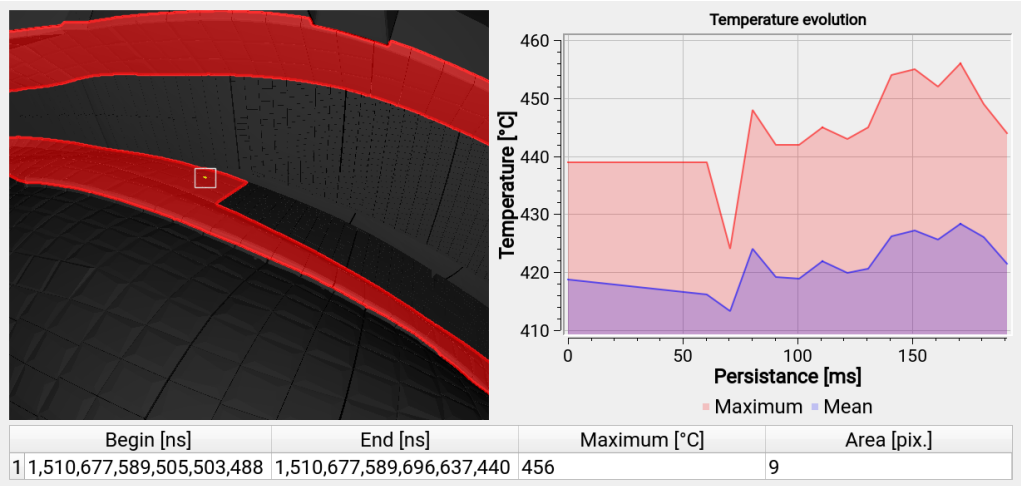


Figure 6. Overload hotspot detection pipeline.

Hotspot average temperature, maximum temperature and area attributes are computed and updated for each tracked hotspot. Moreover, pixels that contribute to the hotspot are translated to world coordinates using the LUT described in Section 3.1.

The result (see Figure 7) for the available dataset corresponds with the hotspot labelled on the baffle with a maximum temperature of 456 °C in [7]. The hotspot originated at timestamp 1,510,677,589,505,503,488 ns epoch time and persisted for around 190 ms.



	Begin [ns]	End [ns]	Maximum [°C]	Area [pix.]
1	1,510,677,589,505,503,488	1,510,677,589,696,637,440	456	9

Figure 7. The detected and tracked hotspot. The results are summarised with the application developed to manage and monitor the processing pipeline.

Tracking with Cluster Correspondence

Finally, tracking of blobs between consecutive frames is performed with correspondence criteria proposed in [3] that investigates overlapping and absolute size difference between pairs of blobs within consecutive frames.

The correspondence criteria for blobs A and B is defined with Equations (1)–(3) as:

$$overlap\_criteria(A,B) = area(A \cap B) \geq min\_area * min(area(A), area(B)), \quad (1)$$

$$oversize\_criteria(A,B) = \frac{1}{max\_oversize} \leq \frac{area(A)}{area(B)} \leq max\_oversize, \quad (2)$$

$$corresponds(A,B) = overlap\_criteria(A,B) \wedge oversize\_criteria(A,B), \quad (3)$$

where *min_area* and *max_oversize* are adjustable parameters that constrain minimum overlap and maximum absolute size difference between blobs, respectively. An example presenting correspondence criteria between two blobs is shown in Figure 8.

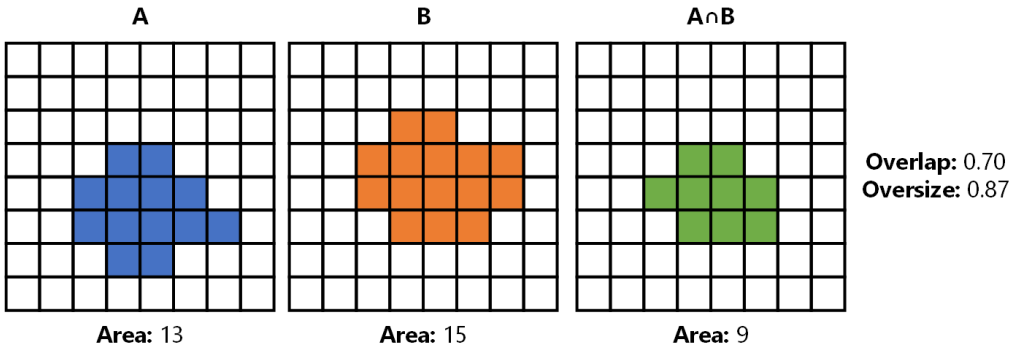


Figure 8. Exemplary cluster correspondence result between two blobs.

For the purpose of blob matching, the required minimum overlap is equal to 80% and the maximum two-fold size difference is allowed. The areas of blobs A, B, $A \cap B$ are computed concurrently on different CUDA streams to reduce the overall runtime as these operations are independent and require the same calculations.

3.3. Surface Layers Detection

The second implemented and benchmarked algorithm detects surface layers. Exposure of components to heat load (strike-line) leads to carbon erosion that is redeposited in the form of thin layers. The measured temperature is higher due to the lower thermal connection to the component surface. As a consequence, a false hotspot alarm might be triggered [9,10]. The detection scheme is based on the idea of a normalised derivative of the temperature evolution proposed by [14]. The authors state that the heating and cooling of regions affected by surface layers is more rapid compared to other regions.

This allows one to identify those regions by thresholding temperature evolution (see Figure 9). The normalised derivative is computed pixel-wise with Formula (4) as:

$$norm[t, T] = \frac{\partial_t T(t)}{T(t)} \approx \frac{1}{T(x_{t-1})} \frac{T(x_t) - T(x_{t-2})}{x_t - x_{t-2}}, \tag{4}$$

where $T(x_t)$ is the pixel temperature at time t. $T(x_t - 1)$ and $T(x_t - 2)$ correspond to the pixel temperatures in two previous frames in respect to the frame at time t. In the available experimental data, the detected surface layers are visible on the horizontal and vertical divertors (see Figure 10). Red colour designates surface layers that originated due to heating, blue due to cooling and green due to both factors. As expected, the surface layers were observed in regions that are affected by the heat load caused by the strike-lines.

The analogous procedure might be applied to detect delaminations where thermal equilibrium is reached slower and at a higher surface temperature [31]. The surface temperature in areas covered by the surface layers (pixel with coordinates x: 599, y: 403) and the normal surface (x: 748, y: 464) under the strike-line were analysed. In Figure 11, rapid heating and cooling are observed, and the reached surface temperature is higher for the area covered by the surface layers. In Figure 12, the derivative exceeds the predefined threshold, and derivative spikes are notably larger for the surface layer area in general.

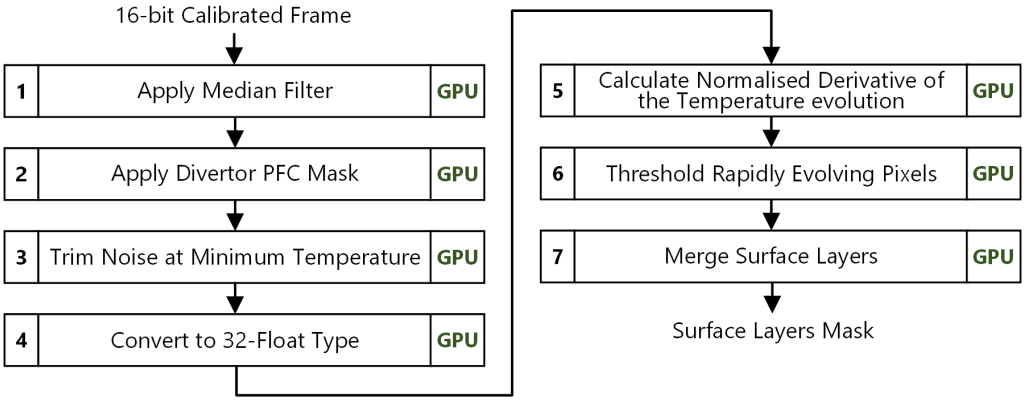


Figure 9. Surface layers detection pipeline.

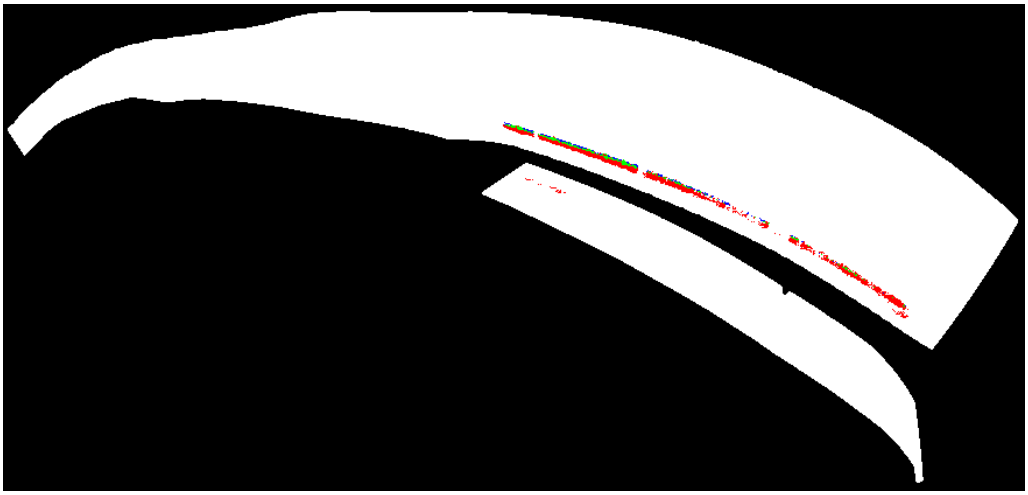


Figure 10. The detected surface layers.

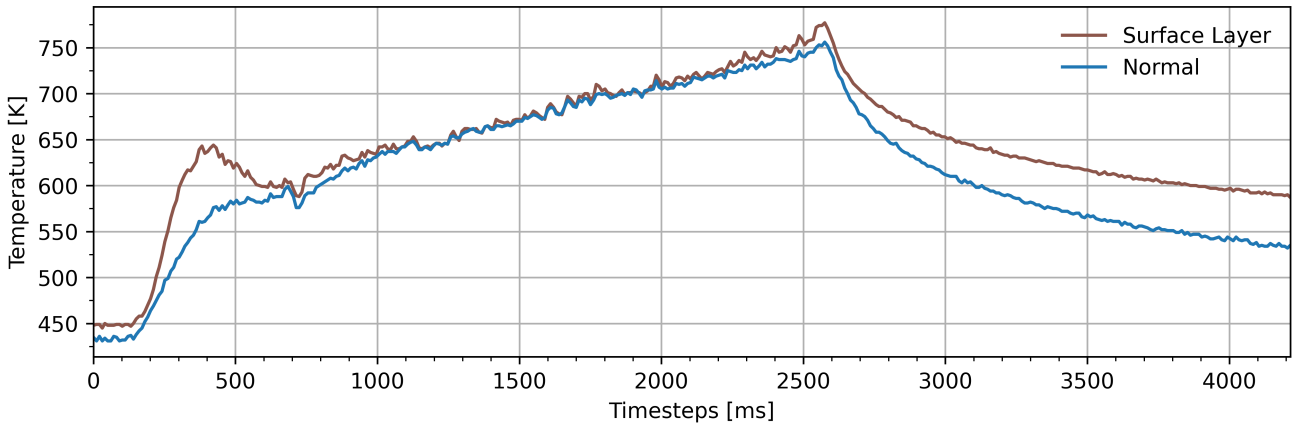


Figure 11. Surface temperature evolution of an area with detected surface layers and a normal area.

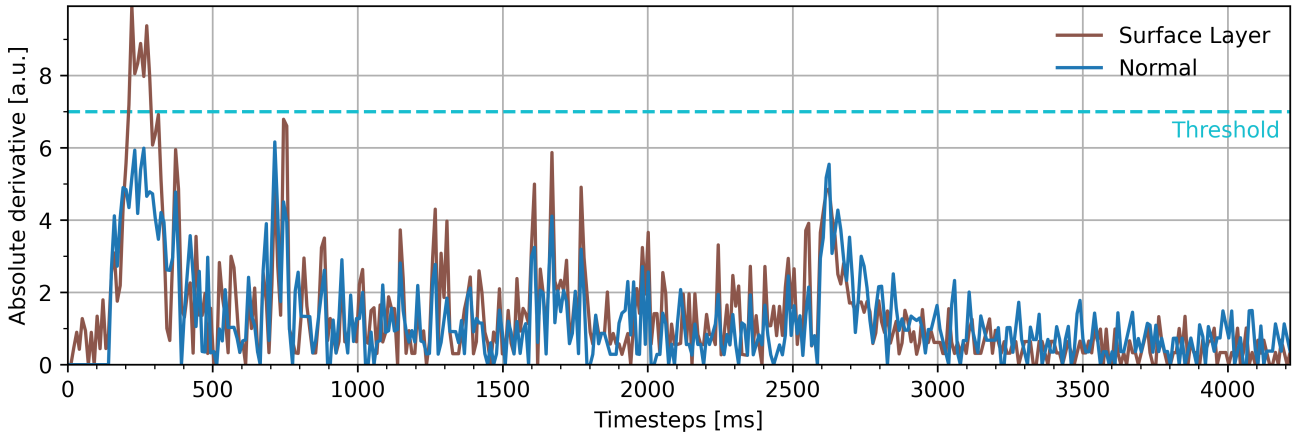


Figure 12. Absolute normalised derivative of the temperature evolution of an area with detected surface layers and a normal area.

The used derivative threshold value that indicates if the area contains surface layers is the same as in [14].

4. Results

In this section, the suitability of the embedded NVIDIA Tegra for real-time processing is investigated. The most computationally intensive pulse sections for the particular algorithms were benchmarked. The followed benchmarking method assumes that the algorithms were compiled with the highest optimisation level and are repeated in a loop. The highest performance mode for both CPU and GPU on Jetson TX2 is activated with commands *sudo nvpmodel -m 0* and *sudo jetson_clocks*. Running the algorithms on the first several frames without including them in the measurements allow one to avoid the initialisation overhead such as initial buffers allocation and CUDA context creation. Supplementary calculations for visualisation purposes, e.g., colouring surface layers types, the imposition of the CAD model, are not included in the measurements.

Benchmarks were made with analogical implementations that only utilise the CPU. Individual functions are mainly implemented in OpenCV compiled with support for the parallel framework—Threading Building Blocks (TBB) (https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onetbb.html, accessed on 21 July 2021) 2017.0 in order to benefit from multiple cores. Average statistics were calculated from the unit timings obtained for each frame across the selected pulse section. The final statistics are computed from the result set, such as average, standard deviation, minimum and maximum.

The selected sections correspond to certain discharge stages (see Figure 13). The overload hotspots detection algorithm is benchmarked throughout the detected hotspot persistence, from timestamp *C* to *D* (191 ms) that corresponds to 20 frames. The hotspot ceases around the time when the heat injection is finished. The surface layers detection algorithm is benchmarked throughout the initial temperature rise, from timestamp *A* to *B* (251 ms), which equals 26 frames. The initial transfer cost to a GPU is included in the measurements. Gathered measurements on the setup presented in Table 1 are depicted in Table 2.

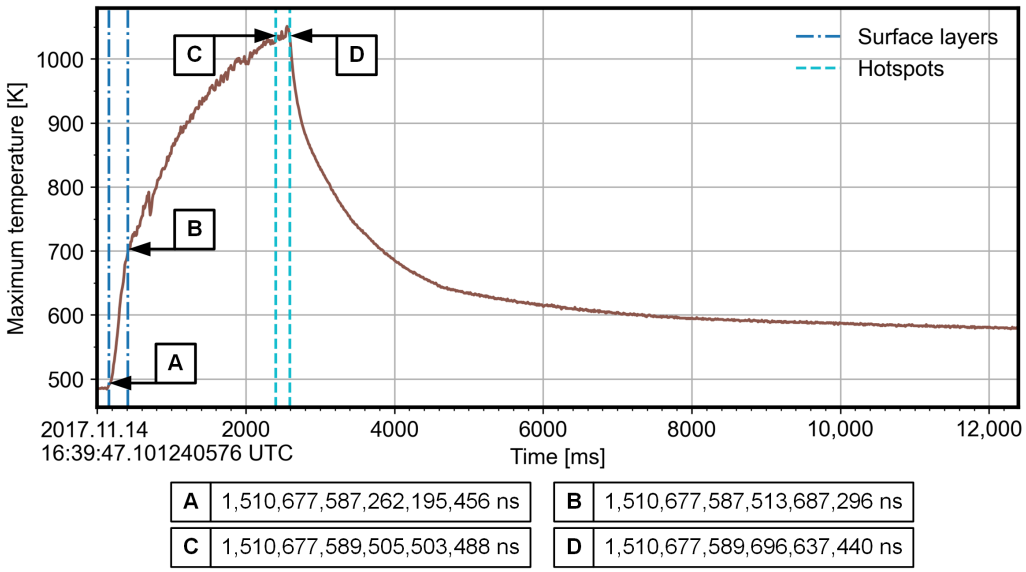


Figure 13. Maximum temperature evolution in Field of View (FoV) after applying a 3×3 median filter with the benchmarked sections marked.

The time-weighted average metrics of achieved occupancy and multiprocessor activity for all kernels, including both custom and invoked internally by the libraries, are collected with the nvprof profiling tool and summarised in Table 3.

Table 2. Performance of the thermal event detection algorithms.

Implementation	Measurements [ms]							
	Overload Hotspots Detection				Surface Layers Detection			
	Avg.	Std. Dev.	Min.	Max.	Avg.	Std. Dev.	Min.	Max.
CPU	8.04	0.067	7.86	8.12	14.04	0.149	13.88	14.42
GPU	4.52	0.046	4.44	4.63	3.67	0.037	3.63	3.78
Speed-up	×**1.78**				×**3.83**			

Table 3. Profiled GPU metrics for the implemented algorithms.

Algorithm	Achieved Occupancy	Multiprocessor Activity
Overload Hotspots Detection	0.93	98.77%
Surface Layers Detection	0.88	99.22%

In both algorithms, the utilisation of a GPU resulted in overall performance improvement.

5. Discussion

The implemented algorithms are a subset of the whole processing pipeline, and additional algorithms must be applied for each frame to provide exhaustive image plasma diagnostics. According to [7], the thermal events must be detected within 100 ms to trigger the interlock system. Nonetheless, the available time is reduced by other activities such as acquisition, correction, analysis and alarm generation. Therefore, the shorter the processing algorithm runtime is, the more time might be devoted to other processes. In the case of incorporating the embedded GPU for both algorithms running sequentially, 13.89 ms is spared on average for each frame. Moreover, there is enough time to simultaneously run both algorithms in order to assess the risk, e.g., if a detected hotspot is a result of the surface layers. As a consequence, proposed algorithms might be employed for real-time thermal events detection. A significant performance improvement is observed for the surface layers

detection on the embedded GPU as all operations are embarrassingly parallel; therefore, there is no need to migrate data to the host for other computations. For the overload hotspot detection on the embedded GPU, the acceleration is lower due to the bottleneck of a topological structural analysis performed on the embedded CPU.

It is noteworthy that to obtain at least the performance comparable to the embedded CPU overload hotspots detection implementation, it was necessary to apply several GPU optimisation techniques such as concurrent copy and execution, page-locked memory, unified memory and custom kernels in addition to incorporating optimised libraries. The crucial part was realising the discrepancies between the memory architecture of embedded NVIDIA Tegra devices and discrete GPUs. Characteristics of memory types in terms of caching are dependent on the exact compute capability of the hardware. For instance, unified memory allows one to avoid duplicate allocations and transfers between embedded GPU and CPU since the physical SoC Dynamic Random-Access Memory (RAM) is shared. Additionally, if prefetching hints [28] are applied, the coherency management is optimised by the driver. However, multiple accesses to pinned memory on both the embedded CPU and GPU result in performance degradation as the I/O coherency [28] is unsupported on the NVIDIA Jetson TX2. As a consequence, in the case of repetitive accesses, it might be advisable to copy a buffer to the device memory for optimal caching behaviour.

Algorithms with pixel-level parallelism are insensitive to image contents. Task-level parallelism requires analysis of the contents; therefore, the measured performance varies. This type of parallelism is often introduced by operations executed on a CPU. For instance, more surface layers would not affect the detection performance as all operations are pixel-wise. On the contrary, more overload hotspots would deteriorate the performance since additional blobs are analysed. Nevertheless, for artificial input frames where all pixels exceed operational limits, and if the analysed blobs are constrained to four, then the algorithm still completes in around 20 ms on the embedded GPU and in above 30 ms on the embedded CPU. Although the performance of some algorithms might vary for different input sequences, it is reasonable to assume that benchmarks performed on the real dataset correctly characterise and approximate the required computation time for future fusion experiments.

Although the image processing libraries were incorporated in the implementation of the algorithms, some available operations in the libraries were insufficient in terms of supported data types or performance. For instance, OpenCV implementation of a median filter *cv::cuda::Filter* created via *cv::cuda::createMedianFilter* is only limited to 8-bit images as well as it has low performance (56.14 ± 0.40 ms) compared to an NPP implementation *nppiFilterMedian_8u_C1R* (3.22 ± 0.04 ms) that also supports 16-bit images *nppiFilterMedian_16u_C1R*. Nonetheless, for a small median filter size, i.e., 3×3, a custom CUDA kernel was implemented and integrated with OpenCV image wrappers to enhance performance by fully utilising coalesced access to shared memory and efficient vector sorting by exchange proposed in the literature [32,33] (0.73 ± 0.02 ms). The average performance of OpenCV CPU implementation optimised for small kernels *cv::medianBlur* is (1.28 ± 0.03 ms). The runtime was measured on an 8-bit image with a resolution of 1024×768. The OpenCV GPU implementation of the median filter was the major bottleneck. The custom CUDA kernel implementation reduced the runtime by 77 times while performing nearly twice as fast as the OpenCV CPU implementation. Optimisations incorporated in the CUDA 3×3 median filter kernel are:

- Shared memory minimises global memory accesses for threads in the same block. An image is chunked into 2D thread blocks that have assigned shared memory of the block size with additional zero-padded borders. Each thread loads the corresponding pixel value from the global memory to the shared memory. Edge threads fill the zero-padding, and if the padded value belongs to the image, it is replaced with the underlying pixel value (see Figure 14) [33].

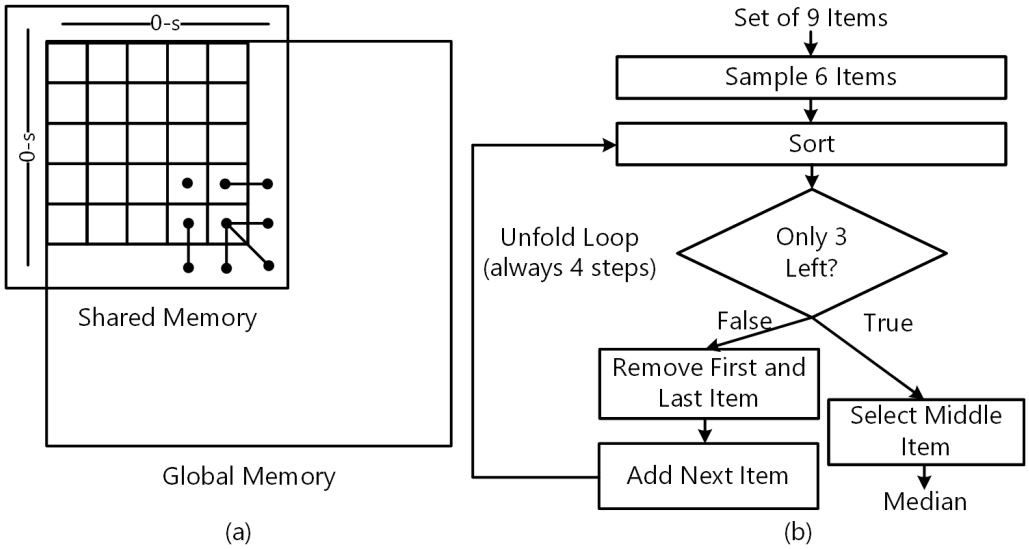


Figure 14. (**a**) Visualisation of caching values from 10×10 global memory in $(5+2) \times (5+2)$ shared memory for four bottom right threads. (**b**) Forgetful sorting for 3×3 kernel to determine the median value.

- Forgetful sorting determines the median value out of nine neighbouring values (see Figure 14). It reduces constant memory and local memory usage (register spilling) in comparison to other sorting algorithms [33].
- Branchless exchange sorts two values with a single instruction [32].

The achieved occupancy and multiprocessor activity for this CUDA kernel are 0.961 and 99.8%, respectively.

The potential solution to reduce latency in setups equipped with a fast PCIe bus and interfacing with a third-party image provider, e.g., a frame grabber, is to utilise efficient GPUDirect transfers [4] in order to directly access data on third-party devices connected to the same PCIe root complex. NVIDIA Jetson Xavier AGX supports GPUDirect technology and is incorporated into the MicroTCA.4 compliant platform [26].

In future works, the authors plan to seek further parallelisation of other thermal image processing algorithms, e.g., reflection detection based on Normalised Cross Correlations (NCC) on a Sliding Time Window (SWNCC) [12] and strike-line detection with max-tree attributes filtering [29]. Moreover, the succeeding implementations and performance benchmarks will be performed on discrete GPU series such as Quadro or Tesla with x86 processors and other NVIDIA Jetson platforms. Finally, the development of tools for managing and monitoring the thermal image processing pipeline for plasma diagnostics is already underway.

6. Conclusions

Current and future fusion devices will increasingly need computational power. GPUs might increase the system performance and provide sufficient acceleration for time-critical applications also on low-power embedded devices compatible with MicroTCA.4 architectures. The performance improvements for overload hotspots detection and surface layers detection algorithms equal 78% and 283%, respectively. The optimisation process includes the implementation of the custom CUDA median filter that outperforms the OpenCV GPU implementation by 7605%. The presented work applies to other scientific disciplines, where thermal images are processed. Described techniques and algorithms especially pertain to other fusion experiments, e.g., hotspots detecting and tracking issues occur as well in tokamaks such as ITER [18], WEST [12], JET [3]. The proposed algorithms are suitable for real-time plasma diagnostics, and the detected events align with observations described in the literature [7]. In addition, the paper proposes exact image processing pipelines for thermal event detection as well as reports expected accelerated performance in the real

nuclear fusion experiment with optimisation techniques specific to the NVIDIA Tegra embedded GPU. It might serve as a baseline for others to decide if an embedded SoC platform is sufficient for their application or to integrate the proposed solutions in systems with either discrete or embedded GPUs.

Author Contributions: Conceptualization, B.J.; methodology, D.M.; software, B.J.; validation, D.M.; formal analysis, B.J. and D.M.; investigation, B.J. and D.M.; resources, D.M.; data curation, B.J.; writing—original draft preparation, D.M. and B.J.; writing—review and editing, B.J., D.M. and P.P.; visualization, B.J.; supervision, D.M.; project administration, D.M.; funding acquisition, D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Restrictions apply to the availability of the Wendelstein 7-X experimental data. Data belongs to EUROfusion Consortium, and the authors are not entitled to disclose it.

Acknowledgments: This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014–2018 and 2019–2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. This scientific paper has been published as part of the international project called 'PMW', co-financed by the Polish Ministry of Science and Higher Education within the framework of the scientific financial resources for 2020 under the contract No 5138/H2020—Euratom/2020/2.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CV	Computer Vision
DMA	Direct Memory Access
DSP	Digital Signal Processor
EAST	Experimental Advanced Superconducting Tokamak
ELM	Edge Localised Mode
FoV	Field of View
FPGA	Field-Programmable Gate Array
GCC	GNU Compiler Collection
GPU	Graphics Processing Unit
HDF5	Hierarchical Data Format version 5
IR	Infrared
JET	Joint European Torus
L4T	Linux for Tegra
LUT	Lookup Table
MARFE	Multifaceted Asymmetric Radiation From the Edge
NPP	NVIDIA Performance Primitives
OP	Operational Phase
OS	Operating System
PCIe	Peripheral Component Interconnect Express

PFC	Plasma Facing Component
RAM	Random-Access Memory
RoI	Region of Interest
SoC	System-on-a-Chip
SWNCC	Normalised Cross Correlations (NCC) on a Sliding Time Window
TBB	Threading Building Blocks
THEODOR	THermal Energy Onto DivertOR
W7-X	Wendelstein 7-X
WEST	W Environment in Steady-state Tokamak

References

1. Vollmer, M.; Möllmann, K.P. *Infrared Thermal Imaging: Fundamentals, Research and Applications*; Wiley-VCH: Weinheim, Germany, 2010; ISBN 978-3527407170.
2. Puig-Sitjes, A.; Jakubowski, M.; Fellinger, J.; Drewelow, P.; Gao, Y.; Niemann, H.; Sunn-Pedersen, T.; König, R.; Naujoks, D.; Winter, A.; et al. Strategy for the real-time detection of thermal events on the plasma facing components of Wendelstein 7-X. In Proceedings of the 31st Symposium on Fusion Technology (SOFT2020), Dubrovnik, Croatia, 20–25 September 2020.
3. Drenik, A.; Brezinsek, S.; Carvalho, P.; Huber, V.; Osterman, N.; Matthews, G.; Nemec, M. Analysis of the outer divertor hot spot activity in the protection video camera recordings at JET. *Fusion Eng. Des.* **2019**, *139*, 115–123. [CrossRef]
4. Zhang, H.; Xiao, B.; Luo, Z.; Hang, Q.; Yang, J. High-Speed Visible Image Acquisition and Processing System for Plasma Shape and Position Control of EAST Tokamak. *IEEE Trans. Plasma Sci.* **2018**, *46*, 1312–1317. [CrossRef]
5. Makowski, D.; Mielczarek, A.; Perek, P.; Jabłoński, G.; Orlikowski, M.; Sakowicz, B.; Napieralski, A.; Makijarvi, P.; Simrock, S.; Martin, V. High-Performance Image Acquisition and Processing System with MTCA.4. *IEEE Trans. Nucl. Sci.* **2015**, *62*, 925–931. [CrossRef]
6. Pisano, F.; Cannas, B.; Fanni, A.; Sias, G.; Jakubowski, M.W.; Drewelow, P.; Niemann, H.; Sitjes, A.P.; Gao, Y.; Moncada, V.; et al. Tools for Image Analysis and First Wall Protection at W7-X. *Fusion Sci. Technol.* **2020**, *76*, 933–941. [CrossRef]
7. Sitjes, A.P.; Gao, Y.; Jakubowski, M.; Drewelow, P.; Niemann, H.; Ali, A.; Moncada, V.; Pisano, F.; Ngo, T.; Cannas, B.; et al. Observation of thermal events on the plasma facing components of Wendelstein 7-X. *J. Instrum.* **2019**, *14*, C11002. [CrossRef]
8. Sitjes, A.P.; Jakubowski, M.; Ali, A.; Drewelow, P.; Moncada, V.; Pisano, F.; Ngo, T.T.; Cannas, B.; Travere, J.M.; Kocsis, G.; et al. Wendelstein 7-X Near Real-Time Image Diagnostic System for Plasma-Facing Components Protection. *Fusion Sci. Technol.* **2018**, *74*, 116–124. [CrossRef]
9. Pisano, F.; Cannas, B.; Jakubowski, M.W.; Niemann, H.; Puig Sitjes, A.; Wurden, G.A. Towards a new image processing system at Wendelstein 7-X: From spatial calibration to characterization of thermal events. *Rev. Sci. Instrum.* **2018**, *89*, 123503. [CrossRef] [PubMed]
10. Ali, A.; Jakubowski, M.; Greuner, H.; Böswirth, B.; Moncada, V.; Sitjes, A.P.; Neu, R.; Pedersen, T.S. Experimental results of near real-time protection system for plasma facing components in Wendelstein 7-X at GLADIS. *Phys. Scr.* **2017**, *T170*, 014074. [CrossRef]
11. Martin, V.; Moncada, V.; Travere, J. Challenges of video monitoring for phenomenological diagnostics in Present and Future Tokamaks. In Proceedings of the 2011 IEEE/NPSS 24th Symposium on Fusion Engineering, Chicago, IL, USA, 26–30 June 2011; pp. 1–6. [CrossRef]
12. Martin, V.; Moncada, V.; Travere, J.M.; Loarer, T.; Brémond, F.; Charpiat, G.; Thonnat, M. A Cognitive Vision System for Nuclear Fusion Device Monitoring. In *Computer Vision Systems*; Crowley, J.L., Draper, B.A., Thonnat, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 163–172.
13. Pisano, F.; Cannas, B.; Fanni, A.; Sias, G.; Jakubowski, M.W.; Drewelow, P.; Niemann, H.; Sitjes, A.P.; Gao, Y.; Moncada, V.; et al. Tools for Image Analysis and First Wall Protection at W7-X. In Proceedings of the 3rd IAEA Technical Meeting on Fusion Data Processing, Validation and Analysis, Vienna, Austria, 28–31 May 2019.
14. Ali, A.; Niemann, H.; Jakubowski, M.; Pedersen, T.S.; Neu, R.; Corre, Y.; Drewelow, P.; Sitjes, A.P.; Wurden, G.; Pisano, F.; et al. Initial results from the hotspot detection scheme for protection of plasma facing components in Wendelstein 7-X. *Nucl. Mater. Energy* **2019**, *19*, 335–339. [CrossRef]
15. Herrmann, A.; Junker, W.; Gunther, K.; Bosch, S.; Kaufmann, M.; Neuhauser, J.; Pautasso, G.; Richter, T.; Schneider, R. Energy flux to the ASDEX-Upgrade diverter plates determined by thermography and calorimetry. *Plasma Phys. Control. Fusion* **1995**, *37*, 17–29. [CrossRef]
16. Zoletnik, S.; Biedermann, C.; Cseh, G.; Kocsis, G.; König, R.; Szabolics, T.; Szepesi, T. First results of the multi-purpose real-time processing video camera system on the Wendelstein 7-X stellarator and implications for future devices. *Rev. Sci. Instrum.* **2018**, *89*, 013502. [CrossRef] [PubMed]
17. Winter, A.; Bluhm, T.; Bosch, H.S.; Brandt, K.; Dumke, S.; Grahl, M.; Grün, M.; Holtz, A.; Laqua, H.; Lewerentz, M.; et al. Preparation of W7-X CoDaC for OP2. *IEEE Trans. Plasma Sci.* **2020**, *48*, 1779–1782. [CrossRef]
18. Martin, V.; Esquembri, S.; Awanzino, C.; Nieto, J.; Ruiz, M.; Reichle, R. ITER upper visible/infrared wide angle viewing system: I&C design and prototyping status. *Fusion Eng. Des.* **2019**, *146*, 2446–2449. [CrossRef]

19. Kadziela, M.; Jablonski, B.; Perek, P.; Makowski, D. Evaluation of the ITER Real-Time Framework for Data Acquisition and Processing from Pulsed Gigasample Digitizers. *J. Fusion Energy* **2020**, *39*, 261–269. [CrossRef]
20. Makowski, D.; Mielczarek, A.; Perek, P.; Napieralski, A.; Butkowski, L.; Branlard, J.; Fenner, M.; Schlarb, H.; Yang, B. High-Speed Data Processing Module for LLRF. *IEEE Trans. Nucl. Sci.* **2015**, *62*, 1083–1090. [CrossRef]
21. HajiRassouliha, A.; Taberner, A.J.; Nash, M.P.; Nielsen, P.M. Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms. *Signal Process. Image Commun.* **2018**, *68*, 101–119. [CrossRef]
22. NVIDIA Corporation. NVIDIA Jetson TX2 Series System-on-Module Data Sheet. Version 1.8. 2021. Available online: http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-modules-data-sheet (accessed on 7 July 2021).
23. NVIDIA Corporation. NVIDIA Jetson TX2 Developer Kit Carrier Board Specification. Version 20200608. 2020. Available online: https://developer.nvidia.com/embedded/dlc/jetson-tx2-developer-kit-carrier-board-spec (accessed on 7 July 2021).
24. Makowski, D. Application of PCI express interface in high-performance video systems. In Proceedings of the 2015 22nd International Conference Mixed Design of Integrated Circuits Systems (MIXDES), Torun, Poland, 25–27 June 2015; pp. 141–143. [CrossRef]
25. Mielczarek, A.; Makowski, D.; Perek, P.; Napieralski, A. Framework for High-Performance Video Acquisition and Processing in MTCA.4 Form Factor. *IEEE Trans. Nucl. Sci.* **2019**, *66*, 1144–1150. [CrossRef]
26. Xilinx Kintex UltraScale+ FPGA NVIDIA Jetson AGX Xavier SOFI Carrier, AMC. 2021. Available online: https://www.vadatech.com/media/AMC565_AMC565_Datasheet.pdf (accessed on 27 May 2021).
27. AG A1x/m1d N—Series. 2017. Available online: https://www.gocct.com/wp-content/uploads/Datasheets/AdvancedMC/aga1xm1d_0117.pdf (accessed on 27 May 2021).
28. NVIDIA Corporation. CUDA for Tegra. Version 11.3.1. 2021. Available online: https://docs.nvidia.com/cuda/cuda-for-tegra-appnote/ (accessed on 27 May 2021).
29. Clemente Bonjour, R. Detection and Classification of Thermal Events in the Wendelstein 7-X. Master's Thesis, UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament de Teoria del Senyal i Comunicacions, Barcelona, Spain, 2020.
30. Suzuki, S.; Abe, K. Topological structural analysis of digitized binary images by border following. *Comput. Vis. Graph. Image Process.* **1985**, *30*, 32–46. [CrossRef]
31. Jakubowski, M.; Drewelow, P.; Fellinger, J.; Puig Sitjes, A.; Wurden, G.; Ali, A.; Biedermann, C.; Cannas, B.; Chauvin, D.; Gamradt, M.; et al. Infrared imaging systems for wall protection in the W7-X stellarator (invited). *Rev. Sci. Instrum.* **2018**, *89*, 10E116. [CrossRef] [PubMed]
32. McGuire, M. A Fast, Small-Radius GPU Median Filter. Published in ShaderX6. 2008. ShaderX6. Available online: https://casual-effects.com/research/McGuire2008Median/index.html (accessed on 21 July 2021).
33. Malcolm, J. Median Filtering: A Case Study in CUDA Optimization. Presented at GTC Silicon Valley. 2009. Available online: https://on-demand-gtc.gputechconf.com/gtcnew/sessionview.php?sessionName=s09455-median+filtering%3a+a+case+study+in+cuda+optimization (accessed on 21 July 2021)