Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Simulation of Primary Specular Reflections on Plasma Facing Components of W7-X Stellarator

Tymofii Bereznytskyi

tymofii.bereznytskyi@gmail.com

Lodz University of Technology

Dariusz Makowski

Lodz University of Technology

Joris Fellinger

Max Planck Institute for Plasma Physics - Greifswald

Bartłomiej Jabłoński

Lodz University of Technology

Zhengcheng Shen

Max Planck Institute for Plasma Physics - Greifswald

Research Article

Keywords:

Posted Date: November 24th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11797756/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Simulation of Primary Specular Reflections on Plasma Facing Components of W7-X Stellarator

Tymofii Bereznytskyi[1*], Dariusz Makowski[1*], Joris Fellinger[2], Bartlomiej Jablonski[1], Zhengcheng Shen[2], the Wendelstein 7-X Team[3]

[1*]Department of Microelectronics and Computer Science, Lodz University of Technology, Wólczańska 221, Łódź, 93-005, Poland.
[2]Max Planck Institute for Plasma Physics, Wendelsteinstraße 1, Greifswald, 17491, Germany.

*Corresponding author(s). E-mail(s): tymofii.bereznytskyi@gmail.com; dariusz.makowski@p.lodz.pl;

Abstract

Accurate temperature diagnostics of plasma-facing components (PFCs) are critical for the safe operation of fusion devices. However, the enclosed environment of stellarators such as Wendelstein 7-X (W7-X) creates challenging conditions for infrared (IR) measurements, as multiple reflections between internal components lead to temperature measurement errors.

To address this issue, a GPU-accelerated ray tracing simulator for the localization of primary specular reflections has been developed. The proposed solution demonstrates a substantial reduction in computation time compared to conventional CPU-based approaches, enabling its potential integration into the real-time diagnostic pipeline [1]. The simulator employs a simplified reflection model, focusing on achieving efficient computation through hardware and software acceleration using NVIDIA's OptiX Software Development Kit, which leverages Ray Tracing (RT) and CUDA cores of the RTX A5000 graphics processing unit.

The implementation was evaluated using data from the W7-X operation phase 1.2 (OP1.2). The ray tracing simulation achieved an execution time of 1.32 s, while subsequent reflection localization based on acquired IR images required only 0.33 s. The resulting reflection masks closely correspond to the ground truth annotations [2], although additional interreflections between targets were observed.

[3]See Grulke et al (https://doi.org/10.1088/1741-4326/ad2f4d) for the W7-X Team

The developed software is fully compatible with Linux, allowing integration into the existing W7-X temperature diagnostic system. The proposed simulator provides a practical foundation for reflection analysis in fusion diagnostics and can be further extended to improve accuracy, incorporate intensity-based reflection modeling, and be adapted for application to other fusion devices.

Keywords: Reflections, Ray Tracing, Infrared Cameras Diagnostic, Wendelstein 7-X

1 Introduction

1.1 Challenges in Infrared Diagnostics for Fusion Devices

The high temperatures necessary for plasma fusion set demanding requirements on fusion devices to withstand significant heat loads during continuous operation. Fusion experiments such as Wendelstein 7-X (W7-X) located at Max Planck Institute for Plasma Physics, Greifswald, Germany [3], Joint European Torus (JET) [4], W Environment in Steady-state Tokamak (WEST) [5] and the Mega Amp Spherical Tokamak – Upgrade (MAST-U) [6] are equipped with specifically designed diagnostic systems that are used for the machine protection. The mentioned systems aim to monitor the temperature of plasma-facing components (PFCs), modulate heating power, and alert operators. This article discusses reflections identification in thermography using the W7-X machine as a case study, where infrared (IR) cameras [7] are used to monitor the temperature across different PFCs inside the vessel.

The enclosed inner vessel creates a challenging environment for IR cameras. Numerous reflections from highly loaded targets occur on components with lower temperature limits [2]. Reflections introduce measurement errors and undesired behavior in the diagnostic system. They could trigger false-positive alarms even under low-heat configuration, making it impossible to use the system reliably. Diagnosticians can identify reflections by interpreting IR images in the context of known internal geometry and local thermal behavior. Nevertheless, deterministic identification based solely on the IR image from the camera remains challenging.

During operation phase 1.2 (OP1.2) of W7-X, recurring patterns of thermal reflections (see Figure 1) were observed. The strongest reflections were typically first-order specular reflections, originating from strike-lines on the target components and appearing on baffles and wall heat shields. While first-order reflections were relatively easy to identify, second- and third-order diffuse reflections cannot be distinguished by the human eye. However, the higher-order reflections continue to affect the overall component temperature, although the magnitude of their impact attenuates with each successive reflection.

This phenomenon can be explained by the characteristics of the materials used in the divertor's construction [8]. The base material can reasonably be described using graphite properties. An emissivity of $\epsilon_G = 0.9$ is assumed [9]. Since the graphite surface can be considered opaque (having negligible transmissivity), the reflectivity is estimated using the energy conservation equation for opaque materials $\epsilon + \rho = 1$ [10].

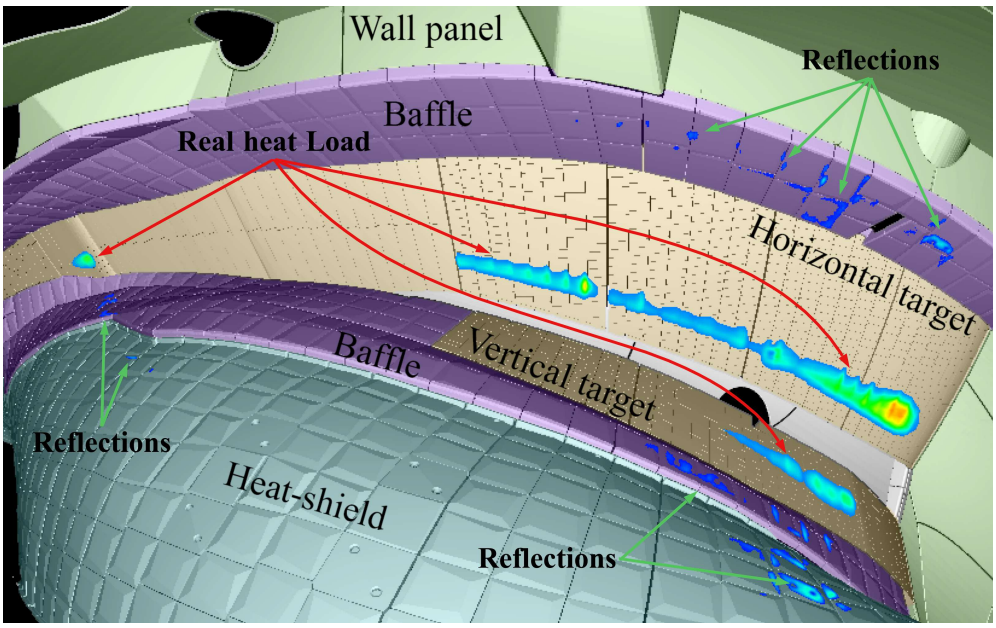

Fig. 1 Identified reflection of PFCs in the 20181017.038 dataset image from AEF10 camera. Reflections of real heat load are visible on baffles and the wall heat shield.

This yields $\rho_G = 0.1$, indicating that only a small fraction of IR radiation is reflected. However, both emissivity and reflectivity are highly dependent on the material surface condition, composition and incident angle.

During W7-X OP2.1, several carbon tiles were replaced with tungsten tiles [11]. Tungsten exhibits a significantly higher reflectivity, leading to more intense thermal reflections. In future W7-X operation campaigns, reflections from tungsten tiles will have a bigger impact on thermography, making proper evaluation of temperature even more important. Future devices, such as International Thermonuclear Experimental Reactor (ITER), will use tungsten-based components [12], [13]. Consequently, even stronger and more complex reflection patterns are expected in new machines.

1.2 State of the Art

An early simulation [14] of IR reflections was conducted for Tore Supra (France) using a Monte Carlo ray tracing algorithm implemented in SPEOS CAA (OPTIS), integrated within CATIA V5. A simplified bidirectional reflectance distribution function (BRDF) [15] was applied to the surface. The method aimed to approximate temperature measurement error by comparing synthetic clean heat loads with simulated reflections, including the effects of surface deposits [16]. While results were promising [17], the method was limited by synthetic data and dependence on commercial software, making it unsuitable for iterative or operational use. The papers presenting the method were not oriented towards time efficiency and used a central processing unit (CPU) for calculations.

As an evolution of the previously described method, a recent approach at the WEST tokamak employs ray tracing to generate synthetic data [18] used to train an AI model [19–21] for correcting temperature measurement errors caused by these reflections. Ray tracing is not applied during operation but serves as a data-generation tool in the pre-operation phase, allowing the AI model to perform fast evaluation, potentially in real time, during operation. The simulations utilize a BRDF measured [22, 23] from tungsten surfaces at WEST to ensure realistic reflection modeling. Initially, simulations were performed entirely on the CPU, requiring approximately 10 hours per image. Subsequently, intersection tests remained on the CPU, while scene illuminance calculations were parallelized on the GPU using CUDA [24], reducing execution time to several minutes [19] and confirming the effectiveness of GPU acceleration. Although this approach reduces computational demands during operation, it is limited by the need for large volumes of synthetic training data and the lengthy training process. Additionally, the AI model produces non-deterministic results, which can occasionally lead to even greater measurement errors.

Another approach to improving efficiency and enabling real-time implementation involves the precalculation of a transfer matrix [25] that describes the contribution of all scene elements to the observed signal. Once computed, the matrix enables fast, deterministic evaluation of measurement errors and remains valid as long as the scene geometry and surface materials remain unchanged. It is constructed using backward ray tracing [26] with surface BRDFs to account for multiple reflections. Originally proposed to mitigate stray light in spectroscopic measurements, the method is equally applicable to IR thermography. Its primary limitation lies in computational complexity, which scales with the number of matrix elements, corresponding to the number of source surfaces in the scene. In the referenced study, this number was reduced to 60 to limit execution time, introducing approximation error. Calculations were performed on the CPU using LightTools (Synopsys, Inc.), though the approach could significantly benefit from GPU acceleration.

While the reviewed studies focus primarily on accuracy, they place less emphasis on computational efficiency. Moreover, most implementations rely on commercial software which does not run on Linux, making integration into the main control systems challenging. As fusion devices move toward more intensive and continuous operation, real-time diagnostic capabilities become crucial. Therefore, developing cost-effective and time-efficient implementations is essential for practical application. It can be achieved only with GPU implementation.

1.3 Research Objective

The objective of this study is to develop and adapt a flexible and extensible simulation framework that runs on Linux and employs ray tracing as a primary and self-sufficient method, rather than a tool for the generation of synthetic data. The central focus of this work is the presentation of efficient software for GPU-based parallel ray tracing. Hence, the simplified reflection model was used. The proposed model aims to efficiently identify the strongest specular reflections on PFCs of a single divertor within a several seconds, using available input data from experimental campaigns. The framework must support modularity and future enhancements, enabling integration within diagnostic

systems running on a Linux operating system. The reflection model should be designed to allow future extensions with additional parameters of materials and environment to improve accuracy while maintaining computational efficiency. This will allow for the establishment of a solid foundation for future research and development.

2 Methods

2.1 Reflections Simulation Model

For the initial simulation phase, a simplified model was adopted, assuming only specular reflections on the target and baffle modules visible through a single IR camera. First-order specular reflections are typically the strongest and most significant. Although the current model does not include diffuse reflections, the software should be structured to facilitate the addition of new parameters and reflection behaviors in future extensions.

The ray tracing output can be preserved as an intermediate result to optimize computations. Given that the geometry of the system remains static throughout a particular operation phase, this intermediate output can be reused without recomputation. This approach has already demonstrated efficiency gains in related work involving stray light mitigation for ITER spectroscopy [25]. The saved output can later be compressed and used in other systems. One of the use cases could involve identifying strong specular reflections originating from selected hot regions on PFCs.

Following existing examples of ray tracing in fusion applications [20], [25], it is more suitable to use backward ray tracing for a configuration involving a single static camera. In backward ray tracing, rays are generated from the camera location and traced toward visible surfaces. At the point of first intersection, the ray is reflected and traced again until it either hits a light source or leaves the scene. The reflection of a ray is described by the equation $r = v - 2(v * n)n$, where r is the reflected ray, v is the incident ray, and n is the surface normal [15].

The first intersection point with the surface is treated as the reflection, while the second intersection surface point is identified as the source of that reflection. This is valid only for rays that intersect exactly two surfaces. Ray tracing is terminated after the second hit, as the model's requirement is to determine only the primary specular reflection. Rays that intersect only one surface and then exit the scene are considered missed. The pixel for which such a ray was traced is therefore expected to have no visible reflection. However, in practical applications, these rays could correspond to reflections originating from components visible in other cameras.

An example of a single traced ray is shown in Figure 2. In this case, the ray intersects two surfaces, which represent the full reflection scenario when the reflection source is identified. The image was taken from the Blender software, which was used to edit the geometry model. Point "1" indicates the position of the camera and serves as the origin of the traced rays. The point "2" marks the first intersection with the surface and corresponds to the location where the reflection occurs. Then the ray is reflected and intersects the second surface. The point "3" identifies the second intersection, which is interpreted as the source of the reflection observed at the point "2".

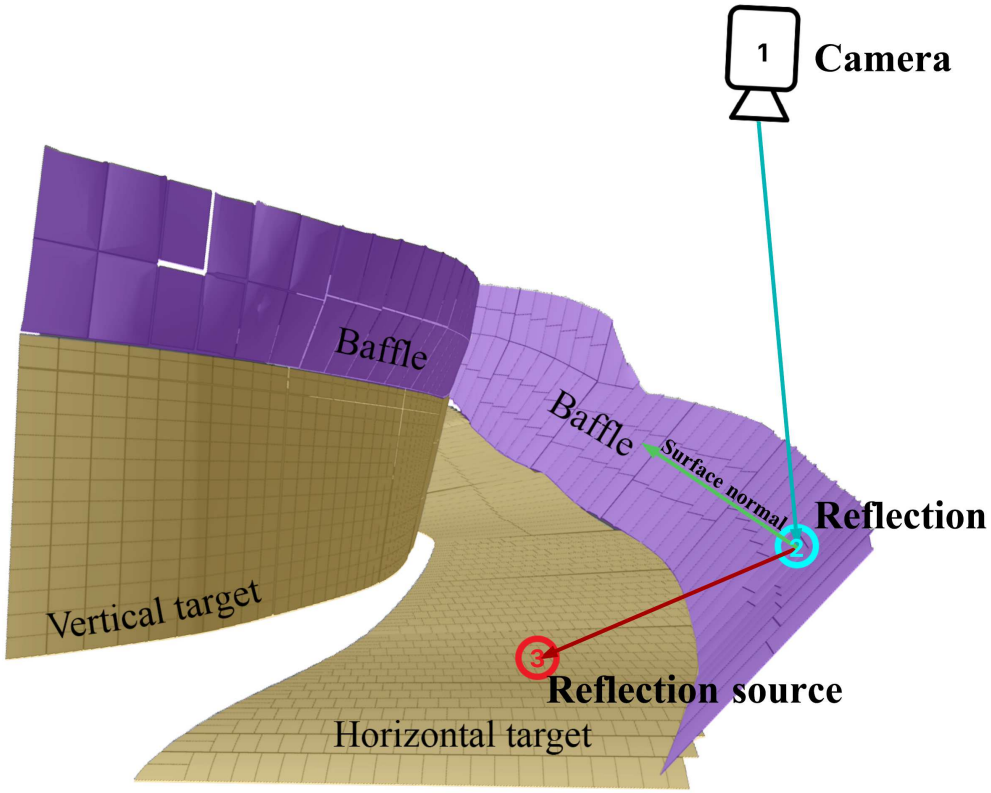

Fig. 2 The full reflection scenario of a backward-traced ray originating from the camera. 1 - camera position, 2 - reflection on the baffle, 3 - reflection source on the horizontal target.

After a ray is traced, the 3D coordinates of the reflection source are obtained, if the source exists. However, the required output is expected in 2D pixel coordinates. Therefore, each 3D coordinate must be mapped to its corresponding pixel in the image. A straightforward 3D-to-2D mapping approach is to assign the point to the closest pixel within a specified distance threshold. However, this method leads to visible artifacts due to missing samples. A more robust solution is to map the point to all pixels located within the threshold distance. In the conducted simulations, a threshold of 1 cm was employed. This approach can be regarded as a fast, artificial approximation of a minuscule diffuse component, which significantly reduces simulation artifacts.

Based on the proposed model, the required input data can be defined. The simulation relies on three inputs: a 3D mesh of the plasma-facing components visible within a single camera's field of view, 3D coordinates of the camera, and a mapping between each IR image pixel and a 3D point on the surface it observes. All input data corresponds to the configuration during the OP1.2 experiments and was taken from W7-X. The geometry model provided in STL format is simplified to include only the baffle

and target modules. Pixel-to-surface mapping is available as part of a scene model [7], [27].

Table 1 Simulation input

#	Input name
1	PFCs geometry
2	Camera location
3	2D-3D pixel-to-coordinates map

Since the final result is intended to be interpreted by people in the 2D image space captured by the camera, the ray tracing output is stored as an intermediate matrix that encodes the reflection relationships between individual image pixels. This reflection map is expressed as a 2D matrix $M_r[p_i, p_s]$, where the first dimension p_i corresponds to the index of an image pixel, and the second dimension p_s denotes the index of its associated reflection source in 2D coordinates. Hence, the matrix defines a dependency between each visible pixel and another location in the image, which is considered to be its specular reflection source.

2.2 Simulation Visualization

As the results of the simulations are intended for use and interpretation by operators, they must be presented in a form that is easy to visualize. While the reflection matrix obtained from ray tracing is suitable for storage and further computational processing, it is not directly interpretable by a human observer. The map for a 1024×768 pixel camera is visualized in Figure 3. In this representation, red denotes the pixels where the first reflected ray does not intersect the defined scene, indicating the absence of a reflection. Blue marks pixels where reflections are observed, green denotes their sources, and cyan represents pixels that act both as reflection points and as sources.

However, the matrix-based map can be further adapted for enhanced visualization. One possible use case is to identify which reflections originate from a known high-temperature source. For example, an operator may need to determine where reflections of a hot strike-line are observed in the camera image. In such a case, the matrix needs to be remapped to obtain $M_r[p_s, p_i]$, where p_s denotes a source set and p_i the corresponding reflection set. Thus, each source element p_s is associated with multiple reflection points p_i. Using the map as a function $f(p_s) = p_i$, the set p_i of pixels is obtained, indicating the locations of the strike-line reflections in the image.

The set p_i of input pixels can be selected using various methods, such as manual selection by an operator, AI-based pattern identification, threshold filtering, adaptive filtering, or other techniques. In the example discussed in this paper, threshold filtering was chosen due to its deterministic nature and ease of implementation. The resulting reflection mask is shown in Figure 4, where a temperature threshold of 750 K was applied to frame 20181017.038 (38 pulse in 17th October 2018) from the AEF10 camera installed in W7-X inside the immersion tube [3]. In this visualization, red represents the IR intensity, blue the selected set of hot pixels above the threshold, and white the simulated reflections.

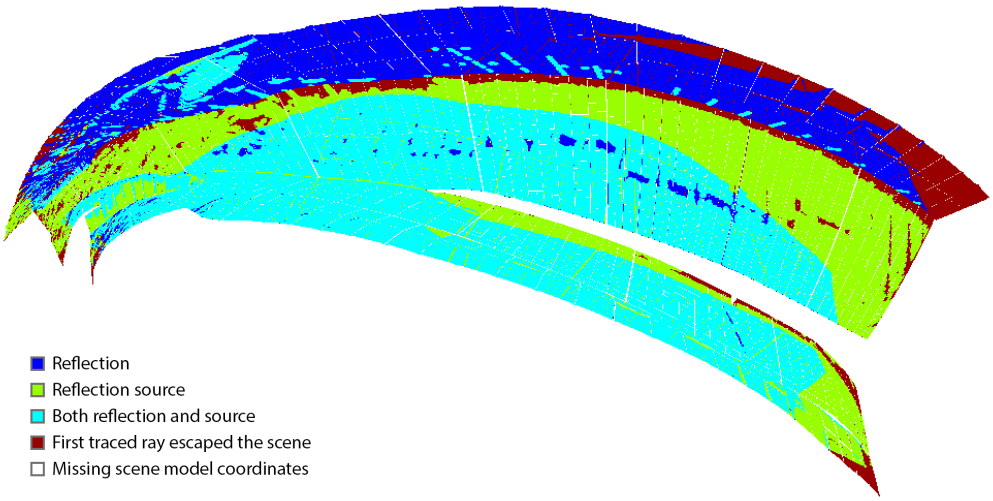

Fig. 3 Visualization of reflection map as ray tracing result of reflections simulation on the target and baffle components visible through AEF10 camera with resolution 1024×768 during 20181017.038 pulse. Red - missed ray pixels, blue - reflection, green - reflection source, cyan - both reflection and source.

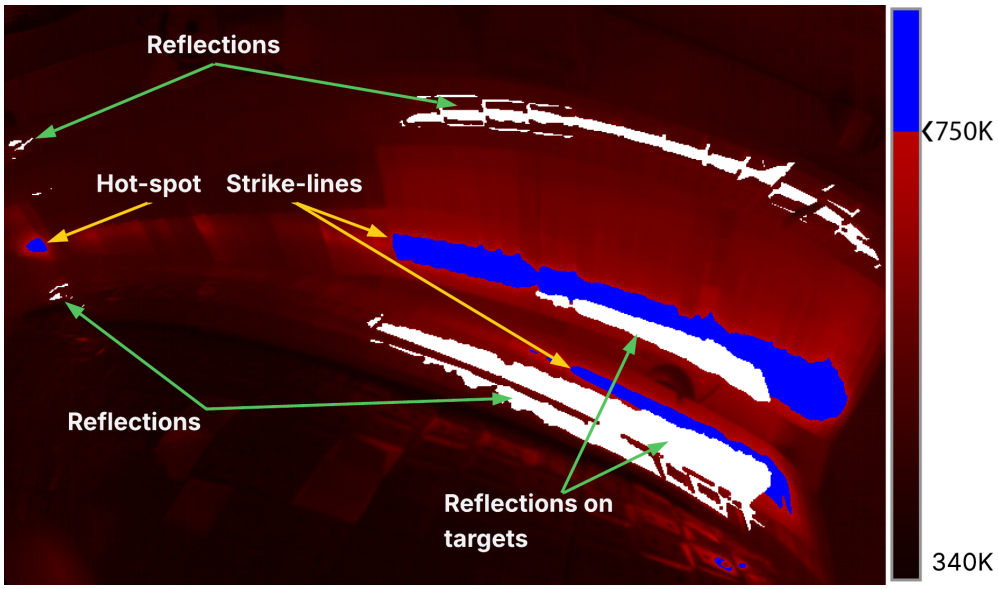

Fig. 4 Simulated strike-lines reflections for the pulse 20181017.038 observed from the AEF10 camera in W7-X. Temperature threshold for hot region is 750 K. Red - IR intensity, blue - filtered hot regions (p_s set), white - their simulated reflections (p_i set). Reflections are morphologically closed with 3×3 kernel.

As the ray tracing was performed on a low-resolution image, many discontinuities appear in the output. They can be reduced in the post-processing phase using methods such as morphological closing or blurring. In the output presented in Figure 4, morphological closing was applied with a kernel 3×3.

Several similarities can be identified in comparison with Figure 1, particularly the simulated reflection in the baffle region. Reflections are also anticipated on the target; however, their intensity was not evaluated. The impact is nevertheless expected to be less significant, as reflections on the divertor result in lower apparent temperatures than the sources, so they do not generate false alarms.

2.3 Simulator Software Implementation

The theoretical concept of the proposed ray tracing model was initially validated using a Python-based prototype. This prototype employed a brute-force ray tracing approach executed on a single CPU thread. The computational complexity of this solution was $O(sp + p^2)$, where s denotes the number of surfaces and p represents the number of pixels. The simplified divertor model used for the simulations contained $s = 31{,}413$ surfaces, and an image of 1024×768 pixels resulted in approximately $p \approx 7 \times 10^5$ pixels to be processed. While the number of surfaces in a more detailed model would increase to at most 3×10^5 for a single camera, this remains manageable due to the linear complexity. In contrast, the number of pixels grows much more rapidly with higher-resolution cameras due to quadratic complexity, reaching approximately 3.6×10^6 pixels for a highly detailed QHD image. Although brute-force ray tracing itself has a complexity of $O(sp)$, finding 3D-to-2D mapping for each hit geometry coordinate requires nested iteration over all pixels, leading to the quadratic factor in p. Nevertheless, ray tracing has a substantially higher computational cost compared to mapping due to the complex arithmetic operations. The prototype confirmed that such an approach is computationally impractical, thereby underscoring the necessity of a specialized ray tracing framework.

The use of a Bounding Volume Hierarchy (BVH) [28], configured with a single node per leaf, reduces the scene traversal time complexity to $O(p \log s)$. Parallel computation on modern GPUs enables simultaneous processing of batches of pixels [24]. In this case, the computational cost of ray tracing alone is $O(b \log s)$, where b is the number of batches depending on the batch size $b = \frac{p}{b_{\text{size}}}$. The computational gain arises from the fact that the number of batches is substantially smaller than the total number of pixels, typically on the order of 10^2. Since 3D-to-2D mapping is required for each pixel with $O(p^2)$ in total, it can also be executed in parallel within the same pipeline, leading to the overall desired time complexity of $O(b \log s + bp)$. This performance can be achieved only through a sophisticated ray tracing framework that leverages GPU hardware acceleration. Such a framework needs to support BVH-based traversal and large-scale parallelization. Finally, the framework should provide C++ support to enable seamless integration with Linux systems.

Common game engines, such as Unreal Engine and Unity, are not suitable for the proposed approach, as they are primarily designed for the development of standalone applications, which complicates their integration into purely software-based systems.

Software Development Kits (SDKs) with lower-level Application Programming Interfaces (APIs) are therefore preferable for integration. SDKs such as OptiX, Vulkan, and DirectX Ray Tracing (DXR) provide support for essential GPU ray tracing techniques, including intersection testing, scene traversal, and hardware-accelerated arithmetic. A comparative overview of their high-level differences is presented in Table 2.

Table 2 Comparison of Ray Tracing SDKs

Characteristic	**OptiX**	**Vulkan**	**DRX**
GPU Vendors	NVIDIA	NVIDIA, AMD, Intel	NVIDIA, AMD, Intel
Hardware Acceleration	CUDA, RT Cores	RT Cores (NVIDIA), Ray Accelerators (AMD)	RT Cores (NVIDIA), Ray Accelerators (AMD)
Proprietorship	Proprietary (NVIDIA)	Open-source	Proprietary (Windows)
Cross-Platform	No	Yes	No
Primary OS Support	Linux, Windows	Linux, Windows	Windows only
Integration	Standalone or SDK	Graphics/Compute API	Direct3D
Support	Strong NVIDIA support, large community. Many example projects are available	Mainly community support	Mainly Windows support
Ray Tracing Pipeline	Ray-based approach with flexible control of ray generation and possibility of behavior definition	Depth testing, blending, anti-aliasing, and rasterization. Dedicated ray tracing pipeline	User-defined shaders within a dedicated pipeline
Notes	Optimized for NVIDIA GPUs, full RT cores exploitation	Cross-platform, supports hardware acceleration on multiple vendors	Integrated into Direct3D, leverages hardware acceleration on multiple vendors

The OptiX SDK was selected due to its strong support from both NVIDIA and the wide developer community. It follows a framework-oriented design that enables defining ray behavior based on interactions with the scene and parameters inherited from previous intersections, thereby fulfilling the requirements of modularity and functional extensibility. Various light effects can be implemented in C++ functions that follow the SDK's interfaces, allowing fast modification, extension, and on-demand substitution of model components.

OptiX leverages Ray Tracing (RT) cores [29], which have been available since the NVIDIA Turing GPU architecture. These RT cores provide hardware-accelerated intersection testing, resulting in ray tracing that is up to seven times faster compared to software-based implementations [29]. For parallelization, OptiX relies on CUDA cores, which enable the scheduling of multithreaded computations. In this approach, the ray tracing for each pixel is executed in an independent thread, where the threads are organized into blocks for efficient execution. Studies on Synthetic Aperture Radar (SAR) ray tracing have demonstrated that OptiX fully utilizes the parallel processing capabilities of GPUs, achieving speedups of up to five orders of magnitude compared

to single-threaded CPU implementations [30]. Furthermore, the use of dedicated hardware acceleration units (RT cores) for SAR simulations has been shown to provide an additional 10 times increase in performance compared to GPU execution without such hardware support [30]. Therefore, the OptiX framework in combination with NVIDIA RTX GPU was selected for the reflection simulations in the W7-X stellarator.

The simulations were performed on an NVIDIA RTX A5000 GPU. This device employs the Single Instruction, Multiple Threads (SIMT) execution model [29], allowing up to 8,192 CUDA cores to execute the same instruction in parallel across its streaming multiprocessors in 1 clock cycle. In this case, the approximate number of batches is $b = \frac{p}{b_{\text{size}}} = \frac{1024 \times 768}{8{,}192} \approx 96$, yielding a time complexity of $O(b \log s + bp)$ with $p = 1024 \times 768$, $b = 96$, and $s = 31{,}413$. The chosen GPU has 24 GB of global memory, which is enough to store the result of the ray tracing simulation - a 1.5 GB reflection map. The GPU specifications and relevant ray tracing parameters used in this work are summarized in Table 3.

Table 3 Key characteristics of the NVIDIA RTX A5000 GPU and essential simulation parameters.

Characteristic	**Value**
GPU Architecture	Ampere
CUDA Capabilities	8.6
CUDA Cores	8,192
RT Cores	64 (2nd Generation)
RT Core performance	54.2 TFLOPS
Base Clock	1.41 GHz
System Interface	PCIe 4.0 x16
Memory Size	24 GB GDDR6
Memory Interface	384-bit
Memory Bandwidth	768 GB/s
Image Resolution	$p = 1024 \times 768$ pixels
Rays per Simulation	865,765
Surfaces in Simulation	$s = 31{,}413$
Sources in the resulting $M_r[p_i, p_s]$	$p_s = 512$
Reflection Map $M_r[p_i, p_s]$ Size	1.5 GB

The results of the ray tracing process are written during execution into the global GPU memory in the form of a matrix $M_r[p_i, p_s]$, which stores the reflection sources for each evaluated pixel. If no sources are detected, zero values are assigned to preserve data alignment. The 2D coordinates are stored in the matrix as two unsigned 16-bit integers, where $p_i = p = 1024 \times 768$ corresponds to the number of image pixels and $p_s = 512$ denotes the number of sources. The number of sources p_s is limited because the model is designed for specular reflections. The method used to impose this limitation is described in Section 2.1.

Since memory transfers to and from the GPU are relatively slow compared to on-device computation, efficient memory management is crucial. Multiple transactions of unaligned data into global memory can significantly increase execution time [31]. Consequently, aligned ray tracing data makes the resulting matrix large. This matrix can be compressed using Run-Length Encoding (RLE) to reduce storage requirements

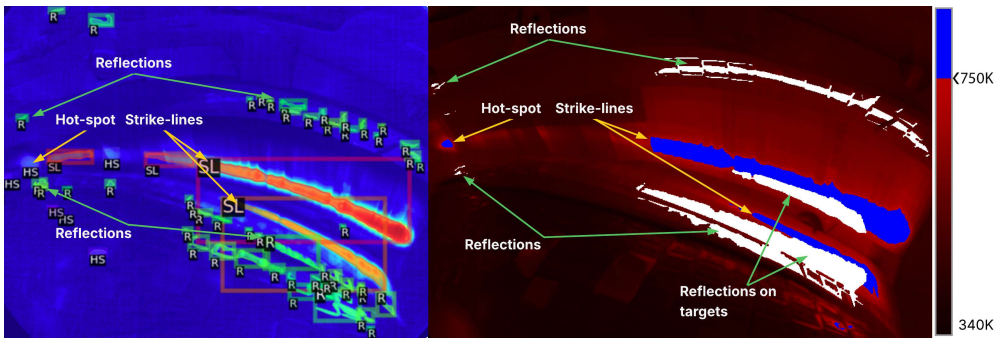

Fig. 5 Simulated (on the right) reflections for the pulse 20181017.038 observed from the AEF10 camera in W7-X compared to the ground truth annotations (on the left) [2].

[32]. RLE compression should be sufficient as more than half of the matrix is expected to be filled with zero values that represent missed rays. However, compression is not the focus of this work. Finally, the matrix may be remapped to the representation $M_r[p_s, p_i]$, which was employed for visualization purposes. The remapping was performed on the CPU.

The presented data demonstrate that employing the OptiX framework on the GPU with dedicated RT cores and massive multithreading allows for a performance improvement of up to 100,000 compared to a single-threaded CPU implementation. This highlights the massive efficiency gains achievable through hardware-accelerated ray tracing on modern GPUs.

3 Results

3.1 Accuracy

The accuracy of the simulated reflections was not the primary focus of this study. Nevertheless, it is essential to identify and localize the strongest and most impactful reflections. A comparison between simulated and annotated [2] reflections of 20181017.038 pulse recorded on the AEF10 IR camera is presented in Figure 5.

The simulated and annotated reflections are not identical. However, they occur in corresponding regions of the same components. Reflections in the simulations display a wider spatial distribution than those in the annotations. For example, annotated reflections in the upper-right corner of the image are primarily confined to the edges of the baffle tiles. In contrast, the simulated image shows reflections along similar edges but covering a larger number of full tiles. Annotated reflections on the inner wall are absent in the simulated image, as the inner wall geometry was not included in the simulation model.

The most notable discrepancy appears in the target region. While the simulations predict reflections between targets, such patterns are absent in the annotated image and cannot be observed directly. The reflections in this region may not be sufficiently strong to be distinguished from the actual heat load observed in the thermal image.

The simulation framework can be further refined by adjusting its parameters to achieve more accurate results. Nevertheless, the current model provides sufficient precision for addressing the primary objective of this paper, which is the evaluation of execution times for the simulation of first-order specular reflections.

3.2 Execution Time

The execution times of two reflection simulation implementations are summarized in Table 4. The initial simulations were performed using a custom Python-based ray tracing implementation without scene traversal acceleration, executed on a single CPU thread. For a relatively simple model including only the baffles and targets visible through a single camera, the computation required 71 hours (almost three days), making it impractical for experimental applications. Such an extended runtime also prevents efficient development of the reflection model, as the feedback loop becomes excessively long. A possible strategy to shorten the development cycle is to simulate only a subset of the image and use it for evaluation. Nevertheless, this approach remains impractical for the operational phase, where simulations must be performed for ten cameras.

Table 4 Simulators implementation comparison.

Characteristic	**Python prototype**	**OptiX simulator**
Ray Tracing Unit	CPU	GPU with CUDA and RT cores
Unit Model	Intel Xeon Silver 4410Y	NVIDIA RTX A5000
Traversing Acceleration	None	BVH structure
Threads	1 (single core)	8,192
CPU RAM Size	128 GB	128 GB
Utilized RAM Size	1.7 GB CPU RAM	1.7 GB VRAM
Image Resolution	$p = 1024 \times 768$ pixels	
Rays per Simulation	865,765	
Surfaces in Simulation	$s = 31{,}413$	
Reflections Matrix Size	1.5 GB	
Ray Tracing Time	71 h	1.32 s
Visualization Time	1.8 s	0.33 s

Hence, the simulation time must be significantly reduced, with the ideal target being results obtainable within a few seconds. To achieve this, the NVIDIA OptiX SDK was selected for software acceleration. For the final simulations, the NVIDIA RTX A5000 GPU was used. This GPU supports the simultaneous execution of up to 8192 threads, enabling highly parallelized computations. The effective execution time for the reflection simulation of the same model was measured at 1.32 s, which is an even better outcome than initially anticipated. This performance gain can be attributed to the architecture of the GPU, particularly its RT cores dedicated to accelerating surface intersection tests [29] in ray tracing.

Nevertheless, the current performance remains insufficient for real-time operation. The presented execution time also includes 3D-2D mapping, which appears to have higher time complexity $O(bp)$ than ray tracing itself with $O(b \log s)$. The

mapping is much simpler than ray tracing, yet it will require more operations with higher-resolution images. Changing the algorithm to a geometrical transformation of coordinates would help to reduce the execution time [33].

Nevertheless, current performance remains insufficient for real-time operation. The presented execution time includes the 3D–2D mapping stage, whose computational complexity $O(bp)$ exceeds the ray-tracing $O(b \log s)$. Although mapping is algorithmically simpler than ray tracing, it becomes increasingly demanding for higher-resolution images. The replacement of the current procedure with a geometric coordinate-transformation approach would reduce execution time [33]. Further reductions could be achieved by employing more powerful GPUs or clusters of GPUs. In addition, optimization of the ray tracing process through improved BVH construction would decrease the time required for scene traversal.

However, since the ray tracing model remains unchanged throughout the campaign, the simulation only needs to be executed once at its beginning, making the 1.32 s runtime fully acceptable. The resulting data can be used to mask reflections based on temperature, a process that is considerably faster than ray tracing, requiring only 0.33 s for multithreaded GPU computations. Using direct GPU memory transfer and more powerful hardware, reflection masking for images with 1024×768 resolution could be brought below 10 ms, which would allow a 100 Hz frame rate obtained from the W7-X cameras during the operation.

4 Conclusions

This study addressed the challenge of localizing reflections in infrared diagnostic images obtained from fusion devices. Reflections within the enclosed inner vessel significantly contribute to measurement errors in surface temperature evaluation, not only in tungsten-based machines, but also in carbon-based devices such as W7-X, where their impact is also considerable. The proposed simulation framework aims to efficiently identify primary specular reflections on PFCs using ray tracing, providing deterministic results based on machine geometry and surface characteristics. Since these reflections are expected to have the greatest influence on the final measurement results, their accurate localization is crucial for enhancing diagnostic reliability.

An initial implementation of the simulator, developed in Python and executed on a single CPU thread, was found to be computationally infeasible due to the long execution time of 71 h. To overcome this limitation, a new simulator based on the NVIDIA OptiX SDK was developed. By exploiting massively parallelized GPU computation and hardware-accelerated ray tracing, the execution time was reduced to 1.32 s for a single camera, fulfilling the requirement for simulation within several seconds. Furthermore, the code is fully compatible with Linux, ensuring possible integration into the existing diagnostic system.

The achieved execution time remains outside the scope of real-time operation. Nevertheless, the geometry of the model and the material properties remain fixed during the experimental campaign, which means that ray tracing simulations are required only at the campaign initialization phase. Consequently, the workflow was divided into two steps: the initial ray tracing simulation and a subsequent reflection localization

step, which masks reflections by combining temperature distribution images of the PFCs with the simulation results. Reflection localization requires only 0.33 s and, with further optimization, could be integrated into the real-time diagnostic pipeline [1].

4.1 Plans for the Future

The implemented simulator for first-order reflections is relatively simple, yet it provides a computationally efficient foundation suitable for further development. To achieve higher accuracy, the reflection model can be extended to include reflections of second, third, and higher orders. One step further, the IR light intensity from each surface can be decomposed into emission and reflection. Both contributions can be back-traced using ray tracing from all surfaces into each pixel on the receiving sensor.

Monte Carlo ray tracing could be employed to simulate diffuse reflections inside the machine. By incorporating emissivity parameters and incident angles, this approach can generate a detailed intensity profile of the image, facilitating the identification of reflections that have the greatest impact on the final IR measurements. Additionally, the impact of material erosion and degradation on the final results must be assessed. The introduction of a correction mechanism might be needed.

However, the application of the Monte Carlo technique would substantially increase the number of rays that must be traced. Higher image resolution and the inclusion of additional machine components in the scene model would further expand computational demands and may exceed the available memory required to store intermediate results. To mitigate these challenges, various strategies for subdivided image processing should be considered, including serial processing of smaller image regions on a single GPU or parallel processing across multiple GPUs.

5 Acknowledgments

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Program (Grant Agreement No 101052200—EUROfusion) and as a part of co-financed by the Polish Ministry of Science and Higher Education within the framework of the international project called 'PMW' International Projects Co-financed. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] Jablonski, B., Sitjes, A.P., Makowski, D., Jakubowski, M., Gao, Y., Fischer, S., Winter, A.: Implementation and performance evaluation of the real-time algorithms for Wendelstein 7-X divertor protection system for OP2.1. Fusion Engineering and Design **190** (2023) https://doi.org/10.1016/j.fusengdes.2023.113524

[2] Jablonski, B., Makowski, D., Sitjes, A.P., Jakubowski, M.: Enabling Instance Segmentation: A Semi-Automatic Method for Thermal Event Annotation. IEEE Transactions on Plasma Science (2024) https://doi.org/10.1109/TPS.2024.3382844

[3] Jakubowski, M., Drewelow, P., Fellinger, J., Sitjes, A.P., Wurden, G., Ali, A., Biedermann, C., Cannas, B., Chauvin, D., Gamradt, M., Greve, H., Gao, Y., Hathiramani, D., König, R., Lorenz, A., Moncada, V., Niemann, H., Ngo, T.T., Pisano, F., Pedersen, T.S.: Infrared imaging systems for wall protection in the W7-X stellarator. Review of Scientific Instruments **89** (2018) https://doi.org/10.1063/1.5038634/596177

[4] Valcárcel, D.F., Alves, D., Card, P., Carvalho, B.B., Devaux, S., Felton, R., Goodyear, A., Lomas, P.J., Maviglia, F., McCullen, P., Reux, C., Rimini, F., Stephen, A., Zabeo, L., Zastrow, K.D.: The JET real-time plasma-wall load monitoring system. Fusion Engineering and Design **89**, 243–258 (2014) https://doi.org/10.1016/J.FUSENGDES.2013.10.010

[5] Mitteau, R., Belafdil, C., Balorin, C., Courtois, X., Moncada, V., Nouailletas, R., Santraine, B.: WEST operation with real time feed back control based on wall component temperature toward machine protection in a steady state tungsten environment. Fusion Engineering and Design **165**, 112223 (2021) https://doi.org/10.1016/J.FUSENGDES.2020.112223

[6] Stobbs, J.M., Tookey, A., Thornton, A.J., Farley, T.: The infrared thermography system on the MAST-U tokamak. The Review of scientific instruments **95**, 113509 (2024) https://doi.org/10.1063/5.0219544

[7] Sitjes, A.P., Jakubowski, M., Naujoks, D., Gao, Y., Drewelow, P., Niemann, H., Fellinger, J., Moncada, V., Pisano, F., Belafdil, C., Mitteau, R., Aumeunier, M.H., Cannas, B., Casas, J.R., Salembier, P., Clemente, R., Fischer, S., Winter, A., Laqua, H., Bluhm, T., Brandt, K.: Real-time detection of overloads on the plasma-facing components of Wendelstein 7-X. Applied Sciences (Switzerland) **11** (2021) https://doi.org/10.3390/app112411969

[8] Qian, X., Peng, X., Fellinger, J., Boscary, J., Bykov, V., Wang, Z., Ye, M., Song, Y.: Assessment of the W7-X high heat flux divertor with thermo-mechanical analysis. Fusion Engineering and Design **109-111**, 565–568 (2016) https://doi.org/10.1016/j.fusengdes.2016.02.045

[9] Carls, A., Köppen, M., Fellinger, J., Schauer, F.: Protection of W7-X diagnostics from radiation heat loads. Fusion Engineering and Design **88**, 1433–1436 (2013) https://doi.org/10.1016/J.FUSENGDES.2012.12.034

[10] Howell, J.R., Mengüç, M.P., Daun, K., Siegel, R.: Thermal Radiation Heat Transfer. Thermal Radiation Heat Transfer (2020) https://doi.org/10.1201/9780429327308

[11] Naujoks, D., Dhard, C.P., Feng, Y., Gao, Y., Stange, T., Buttenschön, B., Bozhenkov, S.A., Brezinsek, S., Brunner, K.J., Cseh, G., Dinklage, A., Ennis, D., Fellinger, J., Flom, E., Gradic, D., Grigore, E., Hartmann, D., Henke, F., Jakubowski, M., Kharwandikar, A., Khokhlov, M., Knauer, J., Kocsis, G., Kornejew, P., Krychowiak, M., Mayer, M., McNeely, P., Medina, D., Neu, R., Rahbarnia, K., Ruset, C., Rust, N., Scholz, P., Sieber, T., Stepanov, I., Tamura, N., Wang, E., Wegner, T., Zhang, D.: Performance of tungsten plasma facing components in the stellarator experiment W7-X: Recent results from the first OP2 campaign. Nuclear Materials and Energy **37**, 101514 (2023) https://doi.org/10.1016/J.NME.2023.101514

[12] Corre, Y., Aumeunier, M.H., Durif, A., Gaspar, J., Krieger, K., Loewenhoff, T., Richou, M., Ratynskaia, S., Tichit, Q., Anquetin, Y., Dejarnac, R., Diez, M., Dubus, L., Firdaouss, M., Gerardin, J., Grosjean, A., Gunn, J.P., Loarer, T., Maget, P., Martin, C., Paschalidis, K., Tsitrone, E., Wirtz, M.: Testing of ITER-grade plasma facing units in the WEST tokamak: Progress in understanding heat loading and damage mechanisms. Nuclear Materials and Energy **37**, 101546 (2023) https://doi.org/10.1016/J.NME.2023.101546

[13] Huber, A., Kinna, D., Huber, V., Arnoux, G., Sergienko, G., Balboa, I., Balorin, C., Carman, P., Carvalho, P., Collins, S., Conway, N., McCullen, P., Drenik, A., Jachmich, S., Jouve, M., Linsmeier, C., Lomanowski, B., Lomas, P.J., Lowry, C.G., Maggi, C.F., Matthews, G.F., Meigs, A., Mertens, P., Nunes, I., Price, M., Puglia, P., Riccardo, V., Rimini, F.G., Widdowson, A., Zastrow, K.D.: Real-time protection of the JET ITER-like wall based on near infrared imaging diagnostic systems. Nuclear Fusion **58**, 106021 (2018) https://doi.org/10.1088/1741-4326/AAD481

[14] Aumeunier, M.H., Travere, J.M.: Development of a realistic photonic modeling for the evaluation of infrared reflections in the metallic environment of ITER. Review of Scientific Instruments **81**, 46 (2010) https://doi.org/10.1063/1.3478676/351321

[15] Shirley, P., Ashikhmin, M., Marschner, S.: Fundamentals of Computer Graphics, 3rd edn. A K Peters/CRC Press, Boca Raton, FL, USA (2009). https://doi.org/10.1201/9781439865521

[16] Firdaouss, M., Mitteau, R., Villedieu, E., Riccardo, V., Lomas, P., Vizvary, Z., Portafaix, C., Ferrand, L., Thomas, P., Nunes, I., Vries, P., Chappuis, P., Stephan, Y.: Power deposition modelling of the ITER-like wall beryllium tiles at JET. Journal of Nuclear Materials **390-391**, 947–950 (2009) https://doi.org/10.1016/J.JNUCMAT.2009.01.243

[17] Aumeunier, M.H., Travere, J.M., Loarer, T., Benoit, F.: Simulation versus reality: A step towards a better understanding of infrared images in fusion devices. IEEE Transactions on Plasma Science **39**, 3014–3015 (2011) https://doi.org/10.1109/TPS.2011.2155678

[18] Aumeunier, M.H., Gerardin, J., Talatizi, C., Bohec, M.L., Yaala, M.B., Marot, L., Loarer, T., Mitteau, R., Gaspar, J., Rigollet, F., Courtois, X., Houry, M., Herrmann, A., Faitsch, M.: Infrared thermography in metallic environments of WEST and ASDEX Upgrade. Nuclear Materials and Energy **26**, 100879 (2021) https://doi.org/10.1016/J.NME.2020.100879

[19] Juven, A., Aumeunier, M.H., Marot, J.: U-Net for temperature estimation from simulated infrared images in tokamaks. Nuclear Materials and Energy **38**, 101562 (2024) https://doi.org/10.1016/J.NME.2023.101562

[20] Juven, A., Aumeunier, M.-H., Brunet, R., Bohec, M.L., Adel, M., Miorelli, R., Artusi, X., Reboud, C.: Temperature estimation in fusion devices using machine learning techniques on infrared specular synthetic data. Institute of Electrical and Electronics Engineers (2024) https://doi.org/10.1109/IVMSP54334.2022.9816270

[21] Aumeunier, M.H., Juven, A., Gerardin, J., Cisse, C.M.B., Pamela, S., Miorelli, R., Reboud, C., Retailleau, F., Marot, J., Rigollet, F., Marot, L.: Surface temperature measurement from infrared synthetic diagnostic in preparation for ITER operations. Nuclear Fusion **64** (2024) https://doi.org/10.1088/1741-4326/ad5a1f

[22] Yaala, M.B., Aumeunier, M.H., Steiner, R., Schönenberger, M., Martin, C., Bohec, M.L., Talatizi, C., Marot, L., Meyer, E.: Bidirectional reflectance measurement of tungsten samples to assess reflection model in WEST tokamak. Review of Scientific Instruments **92** (2021) https://doi.org/10.1063/5.0046140

[23] Natsume, H., Nojiri, K., Kajita, S., Ushiki, T., Sugie, T., Kitazawa, S.I., Kikuchi, T., Yokozuka, T., Tanaka, H., Ohno, N., Hatae, T.: Measurement of the Bidirectional Reflectance Distribution Function of Tungsten Surface Sputtered in Argon Plasma. Plasma and Fusion Research **17**, 2405041–2405041 (2022) https://doi.org/10.1585/PFR.17.2405041

[24] Nickolls, J., Buck, I., Garland, M., Skadron, K.: Scalable parallel programming with CUDA. Queue **6**, 40–53 (2008) https://doi.org/10.1145/1365490.1365500

[25] Kajita, S., Veshchev, E., Barnsley, R., Walsh, M.: Usage of Ray Tracing Transfer Matrix to Mitigate the Stray Light for ITER Spectroscopy. Contributions to Plasma Physics **56**, 837–845 (2016) https://doi.org/10.1002/CTPP.201500124

[26] Kajita, S., Veshchev, E., Lisgo, S., Barnsley, R., Morgan, P., Walsh, M., Ogawa, H., Sugie, T., Itami, K.: Influence of stray light for divertor spectroscopy in ITER. Journal of Nuclear Materials **463**, 936–939 (2015) https://doi.org/10.1016/J.JNUCMAT.2014.11.057

[27] Pisano, F., Cannas, B., Jakubowski, M.W., Niemann, H., Sitjes, A.P., Wurden, G.A.: Towards a new image processing system at Wendelstein 7-X: From spatial calibration to characterization of thermal events. Review of Scientific Instruments

89, 123503 (2018) https://doi.org/10.1063/1.5045560

[28] Shirley, P., Morley, R.K.: Realistic Ray Tracing. A K Peters/CRC Press, Natick, MA, USA (2003). https://doi.org/10.1201/9781439864449

[29] Burgess, J.: RTX on - The NVIDIA turing GPU. IEEE Micro **40**, 36–44 (2020) https://doi.org/10.1109/MM.2020.2971677

[30] Willis, A.R., Hossain, M.S., Godwin, J.: Hardware-accelerated SAR simulation with NVIDIA-RTX technology. In: Zelnio, E., Garber, F.D. (eds.) Algorithms for Synthetic Aperture Radar Imagery XXVII, p. 26. SPIE, Bellingham, WA, USA (2020). https://doi.org/10.1117/12.2558091

[31] Jia, Z., Maggioni, M., Smith, J., Scarpazza, D.P.: Dissecting the NVidia Turing T4 GPU via Microbenchmarking. Citadel (2019)

[32] Birajdar, A., Agarwal, H., Bolia, M., Gupte, V.: Image Compression using Run Length Encoding and its Optimisation. Global Conference for Advancement in Technology, GCAT 2019 (2019) https://doi.org/10.1109/GCAT47503.2019.8978464

[33] Kannala, J., Brandt, S.S.: A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**, 1335–1340 (2006) https://doi.org/10.1109/TPAMI.2006.153