The Use of Different Architectures and Streak Observations Algorithms to Detect Space Debris

Gerard Vives Vallduriola (1), Diego Andrés Suárez Trujillo (1), Tim Helfers (1), Damien Daens (1), Arthur Scharf (1), Jens Utzmann (2), Jean-Noel Pittet (3), Alessandro Vananti (4)

(1) gerard.vives@airbus.com, diego.suarez@airbus.com, tim.helfers@airbus.com, damien.daens@airbus.com, arthur.scharf@airbus.com. *Airbus Defence and Space GmbH. Robert-Koch-Str.1, D-82024 Taufkirchen*

(2) jens.utzmann@airbus.com. *Airbus Defence and Space GmbH. Claude-Dornier-Str., D-88090 Immenstaad*

(3) jean-noel.pittet@esa.int. *Astronomical Institute of the University of Bern (AIUB) Sidlerstr. 5, CH-3012 Bern. (*)*

(4) alessandro.vananti@aiub.unibe.ch. *Astronomical Institute of the University of Bern (AIUB) Sidlerstr. 5, CH-3012 Bern.*

(*) Now at ***European Space Agency. 52 Rue Hillairet. 75012 Paris, France.***

The Use of Different Architectures and Streak Observations Algorithms to Detect Space Debris

Modern society depends heavily on satellite infrastructure. However, Space becomes more and more congested by space debris from over 50 years of space activities. This growing threat in orbit must be monitored. The aim of the ESA GSTP activity „Optical In-Situ Monitor“ is to design and test a breadboard of a space-based space debris camera and to develop and test its end-to-end processing chain.
The on-board processing functions will focus on the payload image processing in order to reduce the data volume (image segmentation for streak detection).
The suitable technologies for the processing units will be described: the HPDP, an ARM-Cortex R5F processor and Microsemi's RTG4 FPGA. For image processing, several algorithms were tested extensively: the CCSDS 122.0-B-1, the Boundary Tensor and the Differences Method.

This paper shows the current state of the project and gives an overview of what activities still need to be tackled before finalisation. The final step of the project will be to decide which combination of processor-algorithm yields the best results. Keywords: Space debris, High Performance, Reliable hardware, reprogrammable hardware and in-orbit flexibility, boundary tensor, Field-Programmable Gate Array (FPGA) RTG4, differences method, ARM-Cortex R5F.

1. Introduction

Modern society depends heavily on satellite infrastructure. However, Space becomes more and more congested by space debris from over 50 years of space activities (Klinkrad, 2006; ESA – Space Debris, 2017). This growing threat in orbit must be monitored in order to sustain safe access and operations of the space infrastructure (ESA Space Situational Awareness Group. 2012; Atkinson, 2012; Hutchinson, 2013).

The aim of the ESA GSTP activity „Optical In-Situ Monitor“ is to design and test a breadboard of a space-based space debris camera and to develop and test its end-to-end processing chain. The corresponding future flight model shall be used for the detection of small-sized (down to 1 mm) space debris in LEO as well as larger objects in GEO. It is intended to be flown on a platform in sun-synchronous orbit near the terminator plane. The breadboard system will constitute a unique facility to perform realistic tests of the end-to-end processing chain for debris observations within a controlled environment.

The on-board processing functions will focus on the payload image processing in order to reduce the data volume. The suitable technologies for the processing units are: the HPDP (PACT XPP Technologies. 2017), an ARM-Cortex R5F processor (ARM, 2011) and Microsemi's RTG4 FPGA (Microsemi, 2017).

For image compression, the CCSDS 122.0-B-1 (CCSDS, 2005) algorithm was initially tested. For on-board object detection, both the Boundary Tensor (Daëns, 2016, Köthe, 2003, Suárez Trujillo, D.A. 2015) and the Differences Method (Métrailler, 2017) were tested extensively.

2. Processing chain

The breadboard software (Utzmann, 2017) will have the structure of a processing chain, where the main parts are: Segmentation, Object recognition, Astrometry and Photometry and finally tracklet building.

3. Candidate Processors

The on-board processing functions will focus on the payload image processing in order to reduce the data volume (image segmentation for streak detection).

These on-board image processing requirements are challenging in terms of on-board processing performance. First assessments of the on-board functionality show an input data rate of ca. 100 Megabits per second (Mbps) and a required integer processing rate of at least 5 Giga-operations per second (GOPS) with at least 16 bit accuracy. Baseline for this assessment is a frame rate of one 2000 pixels × 2000 pixels image per second and a feature detection algorithm for data reduction. A buffer memory of 128 Megabytes (MB) must be envisaged for intermediate storage of the image during the processing steps. Power Consumption for the processing part should not exceed 20 Watts.

The suitable technologies for the processing units can be summarised into three different categories: General Purpose Processors (GPP)/Digital Signal Processors (DSP), FPGAs/ASICs and Specialised Processing Units (SPU). GPPs, and also in part DSPs, provide the easiest development environment and highest developer productivity but the throughput rate can be rather low compared to an FPGA or some Specialised Processing Units and the power consumption relatively high. For IN-SITU, an ARM-Cortex R5F processor was tested. FPGAs are mass produced devices containing numerous look-up tables and other elements interlinked by configurable interconnects. This approach is less efficient than Application-Specific Integrated Circuits (ASICs) since there will inevitably be unused elements of the FPGA, however it offers greatly enhanced flexibility. It combines multiple cores with different characteristics to allow efficient mapping of algorithms with high processing demand. For IN-SITU, Microsemi's RTG4 FPGA was tested. Finally, most SPUs are essentially an array of processing elements with efficient access to memory. The increased specialization

makes them more efficient but more difficult to program. For IN-SITU, the HPDP falls into this latter category.

3.1. High-Performance Data Processor (HPDP)

The HPDP, based on the XPP-III Core by PACT XPP Technologies is a radiation hardened, reprogrammable array processor, a 16 bit architecture designed in 65 nm STM C65SPACE technology (PACT XPP, 2009). The main component of the XPP-III Core represents a dataflow array, consisting of two-dimensionally arranged Processing Array Elements (PAEs), connected by a communication infrastructure that can be reconfigured at runtime, as well as the operations performed by each PAE.

The XPP core architecture is modular in nature and consists of a number of reconfigurable PAEs connected by a reconfigurable data and event network where the data path can go from top to bottom or vice-versa. Two types of PAEs exist: an Arithmetic-Logic Unit (ALU) PAE and a Random Access Memory PAE. The vertical data in the architecture and event routing channels are always contained within a PAE, in the form of Forward Registers (FREGs) that route data vertically from top to bottom, and Backward Registers (BREGs) that route data vertically from bottom to top.

The array network is enhanced by Very Long Instruction Word (VLIW-type) processors called Function-PAEs (FNCs), which are used for controlling and configuring the network and execution of control type processing.

3.2. Microsemi's RTG4 FPGA

Microsemi's RTG4 device is the fourth generation of flash based FPGAs, designed for applications in space. The number of logic gates, registers and specialised

multiplier blocks is significantly higher than in the current generation of FPGAs. Therefore the device is announced to be suitable for signal processing tasks in satellite applications. Airbus Defence and Space GmbH in Ottobrunn evaluates this device using an evaluation board from Microsemi by porting one of their GNSS applications onto the technology.

3.3. ARM Cortex R5F

The ARM Cortex R5F was another evaluated option and is a RISC based microprocessor design featuring a Dual-Core CPU that is able to run in a Lock-Step configuration. As this allows to run the CPU in a redundant mode detecting errors during CPU operations, it is used in safety-critical applications as automotive braking systems (ABS), electric power steering (EPS), railway communications but also in aerospace and avionics. It implements the ARMv7-R architecture and is able to use the Thumb-2 instruction set (a slightly reduced ARM instruction set where both 16 and 32 bit instructions can be used) (ARM 2011).

As for processing performance, the R5F provides dynamic branch prediction using a global history buffer, which allows the CPU to guess or predict which branch will be taken in a code condition. Such a branch prediction is in particular useful with the available 8 stage pipeline. This is done by the PreFetch Unit (PFU) that can fetch instructions (also on a speculative basis) and data via the Data Processing Unit (DPU) to increase the performance of the instruction stream on the R5F.

The processor also has an L1 (level 1) memory system - essentially a cache very close to the CPU - and can handle up to 64KB for both instruction and data (Harvard

architecture) and is interconnected with a Memory Protection Unit (MPU) and Tightly Coupled Memory (TCM) areas. Featuring up to 12 distinct memory regions, the MPU can configure each one with separate attributes, as e.g. shared memory, cached (write-through and write-back) and non-cached regions. Also access permissions can be set separately. Each cache can have an enabled TCM (dedicated fast RAM) with an optional error detection and correction mechanism to protect the stored data.

If the TCM and L1 cannot satisfy a data or instruction fetch, the request is forwarded to the L2 (level 2) system via the AXI master interface. It can then handle the request via an AXI slave interface, three peripheral interfaces (AMBA AXI and optional AHB access) to access peripherals or the Accelerator Coherency Port (ACP) interface, in case the multi-core configuration is used.

The above described R5F design is implemented in the Texas Instruments TMS570LC4357 microprocessor, which is later used for the actual implementation of the Differences Method. This CPU has various dedicated peripherals that are connected to the AHB/AXI interfaces, amongst others an external memory interface (EMIF), a Peripheral Interconnect Subsystem featuring various modules as for example DMA, Ethernet Mac (EMAC) and multiple communication modules as SPI, I2C, CAN, etc.

The TMS570LC4357 is built with 4MB on-chip flash memory with error-correcting code (ECC), 512KB of internal static random access memory (SRAM) with optional ECC and is running at 300 MHz, providing up to 1.66 Dhrystone MIPS per MHz when used with an 8-stage pipeline.

4. Candidate Algorithms

For image compression purposes, the CCSDS 122.0-B-1 algorithm was tested while for on-board object detection, both the Boundary Tensor and the Differences Method were tested extensively.

4.1. CCSDS 122.0-B-1

The CCSDS 122.0-B-1 is an image compression standard published in 2005 by the Consultative Committee for Space Data Systems (CCSDS), which also released other compression standards for arbitrary and hyperspectral data. It is a recommendation for compression of two-dimensional grayscale image data and was specifically designed for on-board processing of payload data on spacecrafts. The aim of the recommendation is to provide an image compression standard that can be implemented despite the limited computational power and memory (CCSDS 2005, 2007).

Two different modes for lossless and lossy compression are supported. Lossless compression is achieved by quantization and entropy coding, for lossy compression in addition image information is removed, depending on compression factors and beginning with the least important information.

The compression is based on a Discrete Wavelet Transform (DWT). The resulting sub-bands of the original image signal are then compressed by a Bit Plane Encoder (BPE), as seen in figure 1 (CCSDS, 2005, 2007).

4.2. Boundary Tensor

The boundary tensor (Köthe, 2003) is a symmetric and positive semi definite tensor. Its non-negative eigenvalues λ_1 and λ_2 are defined by:

$$\lambda_{1,2} = ½ \times (t_{11} + t_{22} \pm \sqrt{((t_{11} - t_{22})^2 + 4 \times t_{12})}) \quad (1)$$

With the positive semi-definite symmetric tensor of order 2 being

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} \\ t_{12} & t_{22} \end{bmatrix} \quad (2)$$

These eigenvalues represent the variations in the pixel intensity in the direction of their orthogonal eigenvectors. In other words, the boundary tensor analyses the area around the processed pixel, and provides a local base showing the direction of the intensity variation, and the strength of the variation. As such, if λ_1 and λ_2 are both null it means that the area of the picture has pixels of constant intensity. If λ_1 is strictly positive and λ_2 is null ($\lambda_1 \geq \lambda_2$) by definition) then the pixel intensity is only changing in the direction given by the eigenvector associated to λ_1: an edge is detected. If λ_1 and λ_2 are both not null, it means that pixel intensity changes in all the directions in the area of the picture: a corner is detected.

To build the boundary tensor, a set of separable polar filters is applied to the picture which is to be analysed. These filters are defined as the product of an angular and a radial function in order to optimize its frequency behaviour. It will also help getting the invariance to rotations. The first step in applying the filters to the picture is done by convoluting the picture with each filter row-like, according to the following equation, where $R_{row}(x,y)$ is the result of the convolution. $l(x,y)$ is the intensity of the pixel at the coordinate (x,y) and K_i are the coefficients of the filter:

$$R_{row}(x,y) = \sum_{i=-\infty}^{+\infty} K_i \times l \times (x + i,y) \quad (3)$$

In practice, the filter coefficients are taken equal to zero outside a radius *r*. *r = 4* will be the used value, as it represents a good compromise between complexity and precision.

The result of the row-like convolution is then again convoluted with the filters, but this time column-like. For the algorithm, the kernels were chosen equal to those taken by Diego Andrés Suárez Trujillo in his implementation of the algorithm on a HPDP (Suárez Trujillo, 2015).

After having simulated several algorithms on the HPDP (compression, boundary tensor and several communications algorithms) Airbus can point out which algorithms are most appropriate for the chosen architecture. This architecture, commonly used in space applications, is especially performant with loops, as these are processed in parallel. However, sequential programs are executed slowly. In contrast, a typical PC-architecture (i.e. programmed in C) is slow for loops, as they cannot be executed in parallel, but very fast for sequential execution.

There is a hardware limitation that for the HPDP data types should be preferably 16 bit fixed point arithmetic, which can be interpreted as using “short integers” instead of “reals” in C. For the on-board image processing S/W module, this fact must be considered when choosing the corresponding algorithms for feature detection and filtering. Floating point could be emulated on on-board H/W, but with high degradation of performance.

The boundary tensor can be split in the odd energy which accumulates in the step edges and in the even energy which accumulates in the roof edges.

The final step of the algorithm is to determine if a pixel corresponds to a resident space object (RSO) or not. In order to do so, it is necessary to extract from the boundary

tensor a measure of the probability of the pixel being an edge. The tensor trace is actually the energy contained in the edges: it is the sum of the eigenvalues of the tensor.

4.3. Differences Method

The basic idea of this final algorithm is to compare two frames to detect the changing features. The registration (alignment and scale transformations) between the two frames is critical to perform a comparison well enough. Usual image treatment tools are used in post-processing manner and allow transformation at a sub pixel level. The following assumptions are made:

(1) Alignment at the pixel level (integer displacement vectors) with an afterwards binning.

(2) Rotation and scale transformations can be approximated as a number of displacement vectors on an equal number of sub-frames.

(3) Selection of the features and their surrounding can be made on the base of the binned pixels.

For the complete on-board segmentation the following steps are suggested.

(1) Remove the saturated stars.

(2) Find the brightest non-saturated stars.

(3) Divide the whole frame into 5 x 5 pixel sub-frames and estimate the displacement vector for each of them.

(4) Execute a binning on two frames taking into account the displacement vectors.

(5) Subtract one frame from the other.

(6) Pixels with values over (under) a threshold are selected as potential streak of the frames being processed.

4.4. Algorithm Trade-Off

The On-Board Processing Pipeline (OBPP) was developed within this project by the Astronomical Institute of the University of Bern. The main objectives of the OBPP are the autonomous on-board data reduction, preliminary image segmentation and object detection. These steps are critical and lead to an effective on-board processing pipeline optimizing the downlink bandwidth usage.

4.4.1. Properties of the Boundary Tensor

In terms of performances, Matlab simulations performed by AIUB show that the SNR obtained in resulting images processed by the Boundary Tensor which was implemented with the RTG4 yields 3.8.

Concerning the time needed to process one single image using the Boundary Tensor, the HPDP implementation needs over 0.734 s, while the RTG4 only needs 0.06 s, which is much faster than the requirement of 1s for processing 2 images for the IN-SITU project. The difference in timing between the two implementations can be explained by the fact the FPGA can process all the convolutions at the same time, as well as calculating the output, without needing to write any data in an external memory, unlike the HPDP. The resulting images, as seen in figures 3 to 5 (figure 2 was the input, an original image obtained with a telescope from AIUB), show that the boundary tensor algorithm can also detect streaks of different intensity.

Some small differences can be observed between the results of the implementation on both architectures (HPDP and RTG4). After an analysis of the dataflow of both architectures, it has been noted that the one on the HPDP had to scale up the kernel coefficients with a multiplying factor in order to make them bigger than 1.

The RTG4 architecture uses fractional length to deal with this problem. The kernels used by the HPDP simulation must make the architecture a more sensitive one, as it seems to detect more debris, but also gets more noise. As a conclusion, both hardware architectures are comparable in terms of output results; the only difference is that the HPDP version changes the kernel coefficients and the threshold value while scaling them up, which takes processing time.

4.4.2. Properties of the Differences Method

The sensitivity of the Differences Method was tested by AIUB. Matlab simulations show that the SNR obtained in resulting images processed by the Differences Method is 2.5 and final results show that this algorithm can detect faint debris streaks down to a peak SNR of 1.2.

Only a part of an image was selected in order to get an input 2048x2048-pixel frame, as required (this image is visible in figure 6). The Matlab model was launched, and its output signal was used to build the binary output of the processed picture, visible on figure 7. Similarly with input image in figure 8, which was synthetically generated, its output (figure 9) also produces faint streaks.

A zoom into image 6 shows that some noise is detected as debris, like the single dot at the right of figure 5, which does not seem to be a streak, but noise.

The implementation with the ARM Cortex processor shows that timing constraints can be met: 2 images can be processed in less than 0.8 seconds. However, some optimisation is still needed on this platform as not all images yield this result: too many stars or streaks may increase the processing time.

4.5. Algorithm trade-off conclusion

As can be seen from figures 2 through 9 and Table 1, the Differences Method algorithm offered valid results within the timing requirements of 2 images / second. In terms of sensitivity, Matlab simulations performed by AIUB show that the SNR obtained in resulting images processed by the Differences Method is 2.5 and final results went as low as 1.2, whereas the Boundary Tensor ran with the RTG4 only obtained 3.8.

The fact that the Differences Method is more sensitive to faint streaks made the project decide for this latter algorithm.

5. Conclusion

In this article, both the growing problematic of space debris and the aim of the ESA GSTP activity "Optical In-Situ Monitor" have been presented. Several image processing algorithms (CCSDS 122.0-B-1, Boundary Tensor and Differences Method) were described, implemented and results were presented.

In terms of performances, Matlab simulations performed by AIUB show that the SNR obtained in resulting images processed by the Differences Method is more sensitive than the Boundary Tensor; hence, the Differences Method was selected due to its sensitiveness to faint streaks with very low SNR. All efforts were focussed on this algorithm and the need to test it with several hardware architectures was identified.

Also, both architectures currently considered for the image processing (ARM Cortex R5F and RTG4 FPGA) and the algorithms considered for data reduction (Boundary Tensor and the Differences Method) are being tested. The RTG4 FPGA implementation of the Boundary Tensor has results which match those attained with the HPDP; and the ARM Cortex also offers interesting results with the differences method.

The current state of the project shows that the ARM Cortex R5F is a promising candidate to implement the Differences Method algorithm for a real mission. However, the implementation of the Differences Method on a Microsemi RTG4 FPGA is still open to complete Table 1. Up to now, all architectures were able to run both the Boundary Tensor and the Differences Method within the timing constraints of 2 images per second. The expectation is that Microsemi's RTG4 FPGA will also perform well within the requirements with the Differences Method. Hence, the final decision on what platform shall be selected is still open.

The final step will be to decide which combination of processor-algorithm yields the best results.

6. Acknowledgements

The authors acknowledge and thank ESA for their support in the IN-SITU project. We also want to thank AIUB for allowing us to publish the original images used during simulations.

Project IN-SITU is funded by ESA Contract No. 4000116518/16/D/SR.

7. References

- Daëns, D. 2016. "Feature Detection on new Space FPGA Technology". *Master Thesis*. Airbus DS GmbH.
- Microsemi Corporation, 2017. "RTG4 FPGA Datasheet". doi: https://www.microsemi.com/document-portal/doc_view/135193-ds0131-rtg4-fpga-datasheet
- PACT XPP Technologies. 2017. "Processor Licensing". doi: http://www.pactxpp.com
- Klinkrad, H. 2006. "Space Debris: Models and Risk Analysis". Springer-Verlag Berlin, 1st Ed.
- Köthe, U. 2003. "Integrated Edge and Junction Detection with the Boundary Tensor". Proceedings of the 9th IEEE International Conference on Computer Vision.

- Métrailler, L., Vananti, A., Schildknecht, T., Pittet, J-N., Utzmann, J., Flohrer, T. 2017. „The Difference Method: A simple and effective on-board algorithm for space debris detection “. 68th International Astronautical Congress. Adelaide, Australia.
- Utzmann, J., Ferreira, L., Vives, G., Metrailler, L., Pittet, J-N., Silha, J., Lièvre, N., Flohrer, T. 2017. “Optical IN-SITU Monitor Breadboard System”. 7th European Conference on Space Debris. ESOC, Darmstadt, 2017.
- ESA Space Situational Awareness Group. 2012. “Space Station Manoeuvres to Avoid Space Debris”. doi: http://www.esa.int/Our_Activities/Operations/Space_Situational_Awareness/Space_Station_manoeuvres_to_avoid_space_debris
- Atkinson, N. 2012. “ISS will do maneuver Friday to avoid collision with satellite debris”. Universe Today. doi: https://www.universetoday.com/92571/iss-will-do-manuever-friday-to-avoid-collision-with-satellite-debris/
- Hutchinson, L. 2013. “How NASA steers the International Space Station around space junk”. Ars TECHNICA. doi: https://arstechnica.com/science/2013/07/how-nasa-steers-the-international-space-station-around-space-junk/
- ESA – Space Debris. 2017. “Space Debris by the numbers”. doi: http://www.esa.int/Our_Activities/Operations/Space_Debris/Space_debris_by_the_numbers
- Suárez Trujillo, D.A. 2015. “Design and Implementation of a feature detection algorithm for space debris detection on a High Performance Data Processor (HPDP)”. Master Thesis. Airbus DS GmbH.
- XPP-III reference Manual – XPP Dataflow Array. PACT XPP Technologies AG, 2009.
- ARM Cortex R5 Technical Reference Manual Revision r1p2. ARM, Sept. 2011. URL http://infocenter.arm.com/help/topic/com.arm.doc.ddi0460d/DDI0460D_cortex_r5_r1p2_trm.pd.
- The Consultative Committee for Space Data Systems CCSDS. Image Data Compression, Recommended Standard CCSDS 122.0-B-1, Blue Book, 2005.
- The Consultative Committee for Space Data Systems CCSDS. Image Data Compression, Informational Report CCSDS 120.1-G-1, Green Book, 2007.

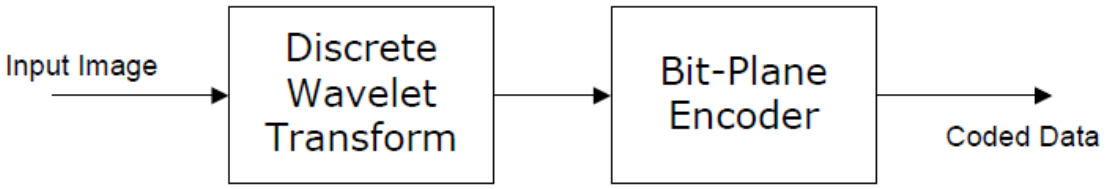

Figure 1: Functional parts of the CCSDS 122.0-B-1 recommended standard (CCSDS, 2005).

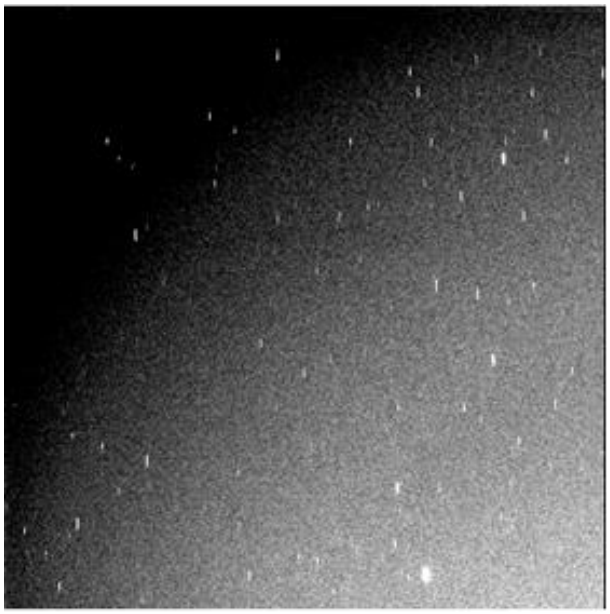

Figure 2 Input original picture obtained by AIUB with a telescope during a night observation. This is the input image to test the boundary tensor implementation with the RTG4.

Figure 3 Result obtained by the RTG4 boundary tensor implementation where the input was figure 2.

Figure 4 Zoom into the input picture shown in figure 2. Even though intensity of streaks varies, in the original image, the Boundary Tensor algorithm can still detect them.

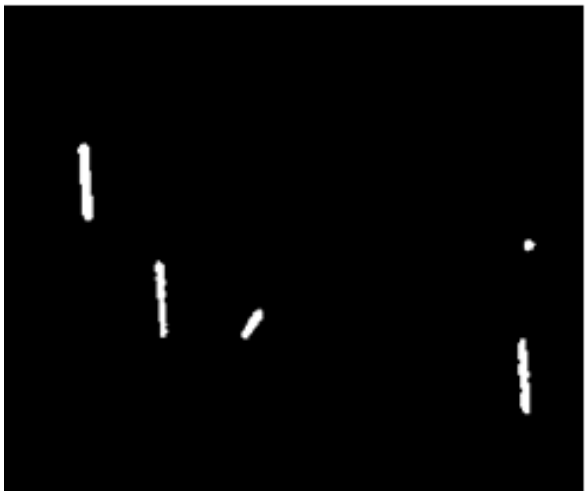

Figure 5 Zoom into figure 8, input picture is figure 2. The dot to the right of the image is noise.

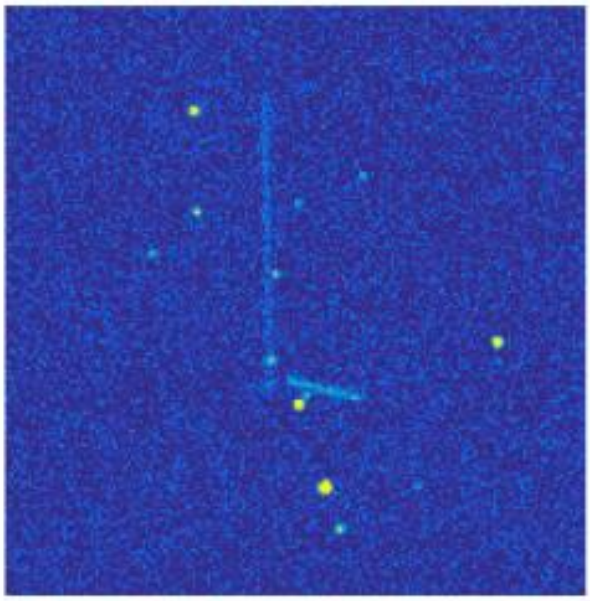

Figure 6 Only part of the original synthetic image was taken in order to get a 2048 pixel by 2048 pixel frame. SNR of streaks are 1.5 and 4 in this frame.

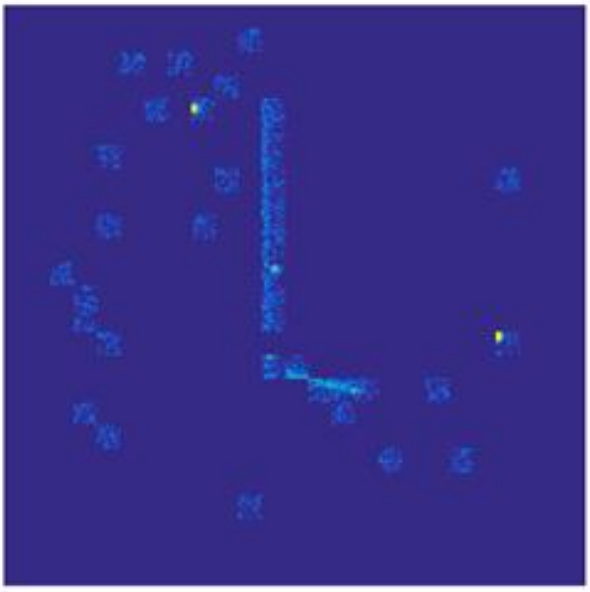

Figure 7 Output of the differences method where input was Figure 6. Stars in the background have been reduced or completely eliminated; only the area around streaks and objects of interest is visible.

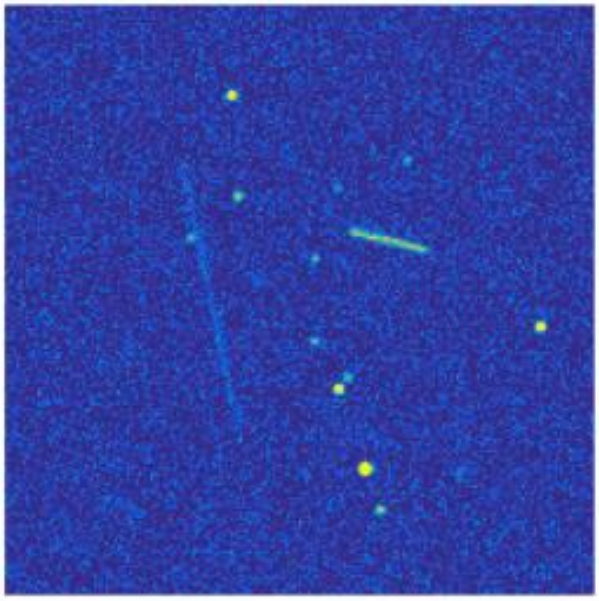

Figure 8 Original frame synthetically generated by AIUB. SNR of streaks are 1.2 and 6 in this frame.

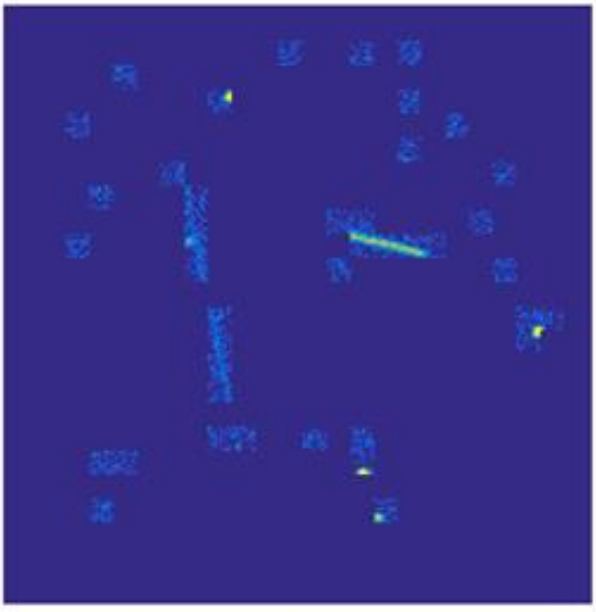

Figure 9 Output of the differences method where input was Figure 4. It is noteworthy that the algorithm can detect very faint streaks and objects.

Table 1: Streak detection algorithms and Platforms used during the project.

(*)Desktop tests were performed with an Intel Core i3-6100 Processor clocked at 3.7 GHz.

(**)Desktop tests were performed with an Intel Core i5 Processor clocked at 2.5 GHz with 4 Mbytes of L3 cache.

All results show the best possible timings obtained so far.

Platform	CCSDS	Boundary Tensor	Differences Method
HPDP	2 images in 4.31s	1 image in 0.7s	N/A
RTG4 FPGA (VHDL)	N/A	2 images in 0.2s	In progress
ARM CORTEX	N/A	N/A	2 images in 0.8s
Desktop PC in C	2 images in 0.8s (*)	1 image in 12s (**)	2 images in 0.8s
Desktop PC in Matlab	N/A	2 images in 0.9s	2 images in 0.8s