THE USE OF DIFFERENT ARCHITECTURES AND STREAK OBSERVATIONS ALGORITHMS TO DETECT SPACE DEBRIS

Gerard Vives Vallduriola[(1)], Tim Helfers[(2)], Franz Biersack[(3)], Arthur Scharf[(4)], Diego Andrés Suárez Trujillo[(5)], Damien Daens[(6)], Steffen Linssen[(7)], Dr. Jens Utzmann[(8)], Jean-Noel Pittet[(9)], Alessandro Vananti[(10)]

[(1)] Airbus Defence and Space GmbH. Taufkirchen, Germany. Email: gerard.vives@airbus.com
[(2)] Airbus Defence and Space GmbH. Taufkirchen, Germany. Email: tim.helfers@airbus.com
[(3)] Airbus Defence and Space GmbH. Taufkirchen, Germany. Email: franz.biersack@airbus.com
[(4)] Airbus Defence and Space GmbH. Taufkirchen, Germany. Email: arthur.scharf@airbus.com
[(5)] Airbus Defence and Space GmbH. Taufkirchen, Germany. Email: diego.suarez@airbus.com
[(6)] Airbus Defence and Space GmbH. Taufkirchen, Germany. Email: damien.daens@airbus.com
[(7)] Airbus Defence and Space GmbH. Taufkirchen, Germany. Email: steffen.linssen@airbus.com
[(8)] Airbus Defence and Space GmbH: Immenstaad, Germany. Email: jens.utzmann@airbus.com
[(9)] Astronomical Institute of the University of Bern (AIUB), Bern, Switzerland. Email: jean-noel.pittet@esa.int
[(10)] Astronomical Institute of the University of Bern (AIUB), Bern, Switzerland. Email: alessandro.vananti@aiub.unibe.ch

ABSTRACT

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 results of the project and gives an overview of which combination of processor-algorithm yields the most promising results for our mission.

1. INTRODUCTION

Space becomes more and more congested by space debris from over 50 years of space activities [1][2]. This growing threat in orbit must be monitored in order to sustain safe access and operations of the space infrastructure [3][4][5].

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 in diameter) 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 [6], an ARM-Cortex R5F processor [7] and Microsemi’s RTG4 FPGA [8].

For image compression, the CCSDS 122.0-B-1 [9] algorithm was initially tested. For on-board object detection, both the Boundary Tensor [10][11][12] and the Differences Method [13] were tested extensively.

2. 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), Field Programmable Gate Arrays / Application Specific Integrated Circuits (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 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.

2.1. High/Performance Data Processors

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 65nm STM C65SPACE technology [15]. 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 Processing Array Element (PAE) 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.

2.2. Microsemi 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 has evaluated this device using an evaluation board from Microsemi by porting one of their Global Navigation Satellite Services (GNSS) applications onto the technology.

2.3. ARM Cortex R5F

The ARM Cortex R5F was another evaluated option and is a Reduced Instruction Set Computer (RISC) based microprocessor design featuring a Dual-Core Central Processing Unit (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).

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

2.4. NANOXPLORE NG-MEDIUM and NG-LARGE

NanoXplore is a French start-up focused on designing high-performance FPGAs, also space qualified, and since both design and production are done in Europe, their products are ITAR-free. This detail, added to a competitive price, make NanoXplore's NG-Medium and NG-Large very convenient and interesting alternatives to current ITAR-sensitive FPGAs.

The NG-MEDIUM was designed, verified and manufactured using 65nm CMOS technology from STMicroelectronics, it is radiation hard and is delivered with a set of self-developed tools used for design, synthesis, Place&Route. The NG-Medium has a capacity of 550.000 ASIC gates, 32256 registers, 112 embedded DSPs, 1 SpaceWire link (400Mbps) and 24 clocks.

The NG-LARGE was designed, verified and manufactured using 65nm CMOS technology from STMicroelectronics, it has 1900000 ASIC gates, 129024 registers, 384 embedded DSPs 1 SpaceWire link (400Mbps) and 32 clocks. Embedded in it there is a rad-hard ARM Cortex-R5x.

3. 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.

3.1. 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 [9][16].

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 Fig. 1.

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

3.2. Boundary Tensor

The boundary tensor [11] is a symmetric and positive semi definite tensor with non-negative eigenvalues λ_1 and λ_2 and with the positive semi-definite symmetric tensor T of order 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 image 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 image: a corner is detected.

To build the boundary tensor, a set of separable polar filters is applied to the image 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 image is done by convoluting the image with each filter row-like, taking into account the intensity of each pixel and the coefficients of the filter.

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 [12].

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.

3.3. 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.

3.4. Algorithm Trade-Off

The On-Board Processing Pipeline (OBPP) was developed within this project by the Astronomical Institute of the University of Bern, Switzerland. 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.

3.4.1. 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 1 s 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 Fig. 3 to Fig. 5 (Fig. 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.

3.4.2. 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 Fig. 6). The Matlab model was launched, and its output signal was used to build the binary output of the processed picture, visible on Fig. 7. Similarly with input image in Fig. 8, which was synthetically generated, its output (Fig. 9) also produces faint streaks.
A zoom into Fig. 6 shows that some noise is detected as debris, like the single dot at the right of Fig. 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.

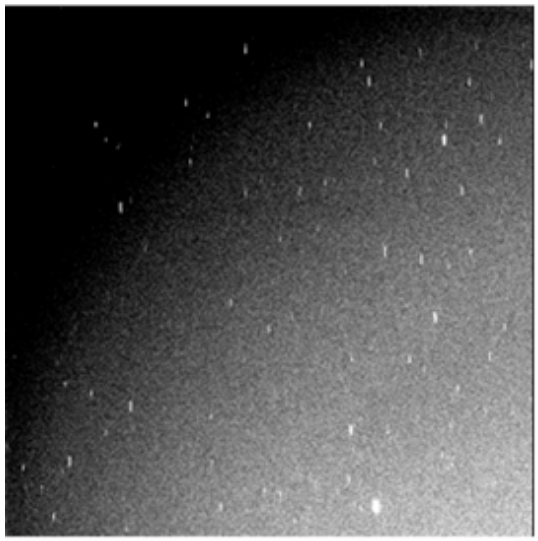

Fig. 2. Input original picture obtained by AIUB with a telescope during a night observation. This is the input image used to test the Boundary Tensor implementation with the RTG4.

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

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

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

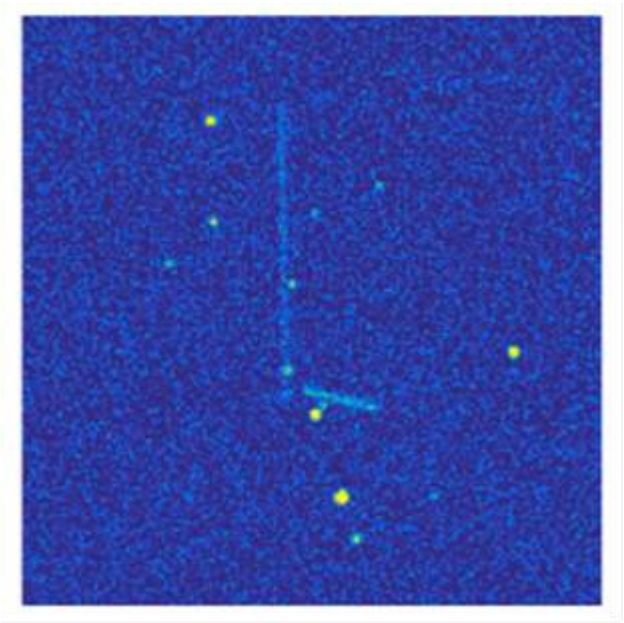

Fig. 6. Only part of the original synthetic image was taken in order to get a 2048 pixel by 2048 pixel frame. SNR of streaks are 1.5 and 4 in this frame.

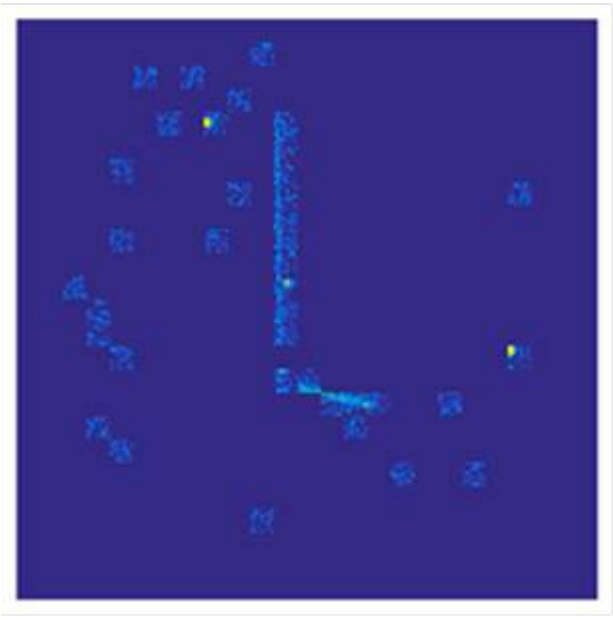

Fig. 7. Output of the differences method where input was Fig. 6. Stars in the background have been reduced or completely eliminated; only the area around streaks and objects of interest is visible.

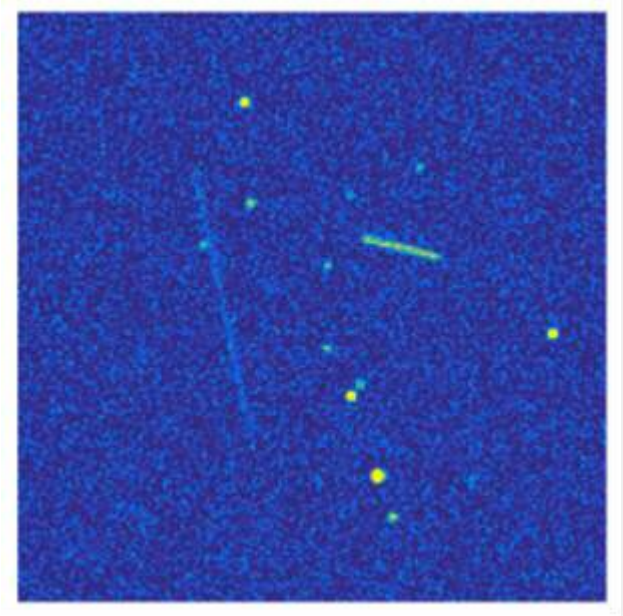

Fig. 8. Original frame synthetically generated by AIUB. SNR of streaks are 1.2 and 6 in this frame.

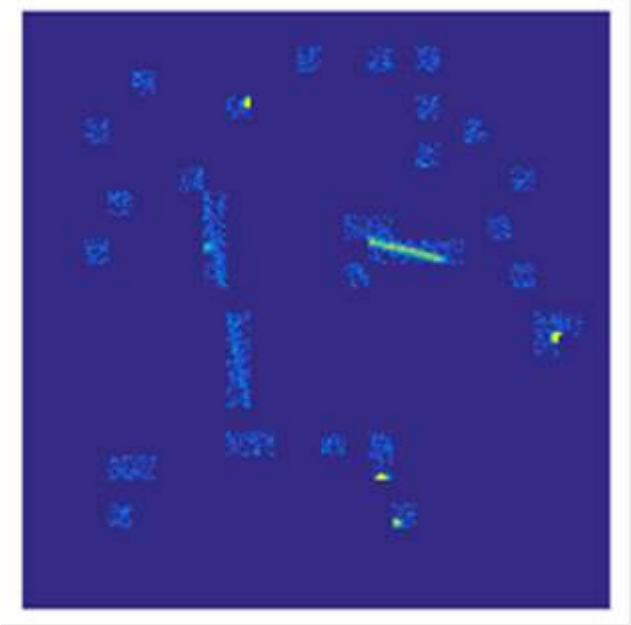

Fig. 9. Output of the differences method where input was Fig. 4. It is noteworthy that the algorithm can detect very faint streaks and objects.

3.4.3. Algorithm Trade-Off Conclusion

As can be seen from Fig. 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.

TABLE I. STREAK DETECTION ALGORITHMS AND PLATFORMS USED DURING THE PROJECT. BEST CASE RESULTS OBTAINED SO FAR.

Platform	Algorithm		
	CCSDS 122.0-B-1	***Boundary Tensor***	***Differences Method***
HPDP	2 images in 4.31s	1 image in 0.7s	N/A
RTG4 (VHDL)	N/A	2 images in 0.2s	See Tables VII - VIII
ARM CORTEX	N/A	N/A	2 images in 0.8s
Desktop PC in C	2 images in 0.8s [a]	1 image in 12s [b]	2 images in 0.8s
Desktop PC in Matlab	N/A	2 images in 0.9s	2 images in 0.8s

[a]. *Desktop tests were performed with an Intel Core i3-6100 Processor clocked at 3.7 GHz.*

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

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

4. FURTHER RESULTS WITH DIFFERENCES METHOD

During the course of the last months, since the final image processing algorithm was chosen, several further tests were performed on it to test its boundaries and also those of the hardware platforms where it ran. The major difference to the tests performed up to now was that images were no longer only 2048x2048 pixels large but also: 668x1002 pixels and 2672x4008 pixels. Two different simulations were added to the results obtained so far: an expanded C++ simulation and a VHDL version of the algorithm running on an RTG4 FPGA from Microsemi.

4.1. The C++ implementation

A C++ implementation on a Windows 7 and a Linux environment was used to simulate the on-board chain on an elegant breadboard. The results can be seen in the following table:

TABLE II. EXECUTION TIMING OF THE C++ IMPLEMENTATION

Platform	Resolution	Differences Method
Linux 3.4 GHz	668x1002	0.112
	2048x2048	0.354
	2672x4008	1.112
Windows 7, x86-64, 3.4 GHz	668x1002	0.281
	2048x2048	0.331
	2672x4008	0.893

The Difference Method running on a Desktop Computer clearly benefits from 8 GB of RAM and a significantly higher clock frequency of 3.4 GHz, a factor 11 higher than what can be used for the on-board processing platforms, the ARM Cortex R5F and the RTG4.

4.2. The VHDL implementation

For this simulation, an external Dual-Port SRAM as implemented in the VHDL design is assumed for the following calculations. In Table III, the necessary amount of clock cycles per operation when processing 668x1002 images is analysed as well as for which steps one can give an exact number of clock cycles and for which steps the number of required cycles depends on the content of the image.

TABLE III. NUMBER OF CLOCK CYCLES (4x4 PRE-BINNING)

Operation	**668x1002 pixels**	**2048x2048 pixels**	**2672x4008 pixels**
Storing	2,683,104	16,782,976	42,843,264
Binning	670,672	4,194,304	10,709,376
Star Detection	47,917	268,144	675,336
Displ. Vector Callibration	4,504	4,504	10,516
Second Binning	41,085	260,100	667,332
Difference	40,261	258,064	664,000
Region of Interest Selection	4,565+?x 324	28,900+?x324	74,148+?x324
Adding stars	3200	3,200	9,800
Image Compression	670,672	4,194,304	10,709,376
Total	4,165,980	25,994,496	66,357,136

In order to run the simulation, a clock period of 10ns was estimated and run with Synplify Pro from Synopsys. The result revealed that the Difference Method implemented as examined earlier can run on the RTG4 with an estimated clock frequency of 5.3 MHz (theoretical maximal clock frequency: 300MHz). From this result it is clear that there is a bottleneck in the design which needs to be analysed. The dedicated simulations show as follows:

TABLE IV. ESTIMATED CLOCK FREQUENCY ON THE SEPARATED ALGORITHM MODULES

Algorithm Operation	Estimated Clock Frequency [MHz]
Storing	11.8
Binning	125.2
Star Detection	5.3
Displacement Vector	97.9
Second Binning	72.8
Difference	23.7
ROI Selection	61.1
Addition of stars	139.3
Image Compression	319.7
Top Level Module	5.3

These modules are all working with clock frequencies above what is required to comply with the timing constraint. For the ones that do not, it is possible let the synthesis result display the critical path causing the longest delay, to know where the design has to be optimized. In this case, the longest path clearly is caused by the complex star detection.

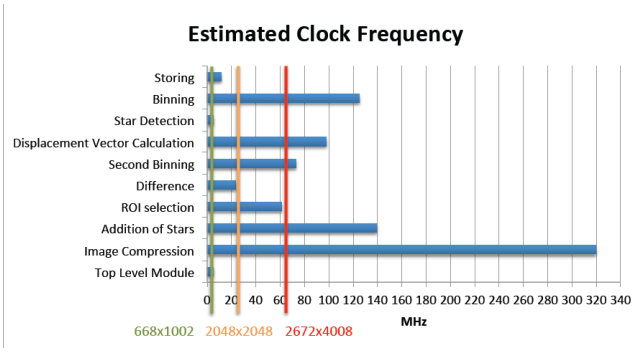

Fig. 10. Bar graph comparing the clock frequency distribution to the required frequencies for processing a certain resolution in time.

For different image sizes it can be seen that the maximum cock frequency is always limited by some blocks of the algorithm. The consequence is that these blocks will have to be optimised or ported outside the FPGA.

TABLE V. RESOURCE USAGE OF THE RTG4 PROVIDED BY SYNPLIFY PRO.

	Carry cells	DSP Blocks	LUTs
Storing	1553	8	1888
Binning	472	3	740
Star Detection	10082	43	12228
Displacement Vector	251	0	459
Second Binning	2851	8	3103
Difference	1436	6	1906
ROI Selection	934	17	1194
Addition of stars	324	13	407
Image Compression	127	0	176
Top Level Module	13852	66	18523

Since the Differences Method only takes up a small percentage of the available resources of the RTG4, more commercial FGPAs were taken as a comparison: NanoXplore's NG-Medium and NG-Large. These two FPGAs are a very convenient and affordable alternative to other FPGAs, even if they are smaller than the RTG4 (as an example, NG-MEDIUM is comparable to a RTAX-2000).

TABLE VI. RESOURCE USAGE OF THE NANOXPLORE NG-MEDIUM AND NG-LARGE PROVIDED BY NANOXPLORE'S NXP TOOL COMPARED TO THE RTG4 RESULTS. RESULTS FOR NG-LARGE STATIC TIMING ANALYSIS NOT SUPPORTED YET.

	Carry cells	DSP Blocks	LUTs	Req. Freq. [MHz]	Max. Freq. [MHz]
RTG4	13852	66	18523	300	5.3
NanoXplore NG-Medium (After Synthesis)	7140/8064 (89%)	42/112 (38%)	3361/32256 (11%)	50	16.16
NanoXplore NG-Medium (After Place & Route)	7154/8064 (89%)	42/112 (38%)	3304/32256 (11%)	50	16.16
NanoXplore NG-Large (after Synthesis)	7140/8064 (23%)	42/384 (11%)	3304/129024 (3%)	n/a	n/a
NanoXplore NG-Large (after Place & Route)	7154/8064 (23%)	42/384 (11%)	3304/129024 (3%)	n/a	n/a

The following image simulates a single debris object moving across the scene slightly below the centre of the image. The exposure time of the BBI camera is set to 0.5 s. The debris particle is revealed after performing the difference and appears as a bright spot which is highlighted by a red circle. It is found during the subsequent ROI selection by a simple thresholding operation.

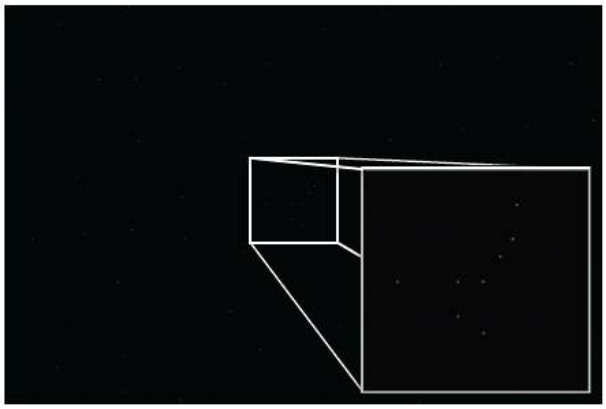

Fig. 11. VHDL result image, with zoom-in for detail view.

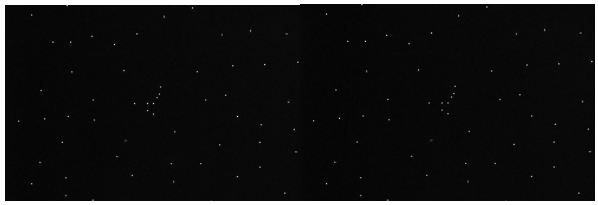

Fig.-12. Binned images 1 and 2.

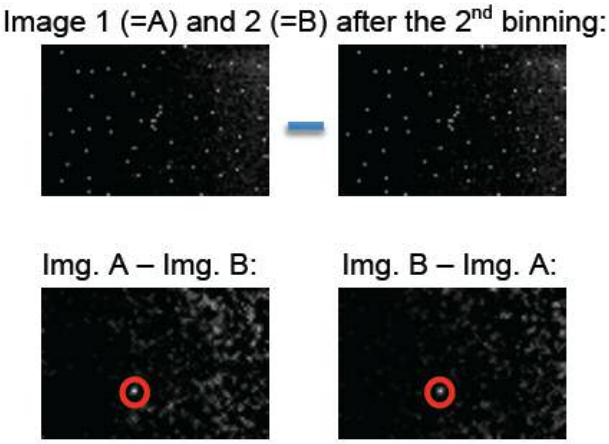

Fig.-13. First pair of the G1 TSU 668x1002 test images and their intermediate results.

4.3. Direct comparison of all implementations

Even though a direct comparison of the three different implementations (C, C++ and VHDL) with respect to their execution time is not immediately possible, a summarising table has been put together to give the reader an idea of

TABLE VII. EXECUTION VERSION OF THE VHDL AND C VERSION FOR THE CONTINUOUS MODE

Algorithm Operation	VHDL [s]	C [s]
Pre-Processing	1.583	0.010
Binning with Dual-Port RAM	0.791	10.842
Binning with Single-Port RAM	0.840	
Median Calculation	n/a	0.349
Star Detection with Dual-Port RAM	0.051	0.169
Star Detection with Single-Port RAM	0.052	
Displacement Vector	0.001	0.002
Second Binning	0.0049	0.470
Difference	0.0049	1.002

ROI Selection	0.005	0.003
Addition of stars	0.001	
Image compression	0.791	1.268
TOTAL with Single-Port RAMS	3.321	14.115
TOTAL with Dual-Port RAMS	3.371	

The Difference Method at first glance performs noticeably better in VHDL than it does in C for both modes. However, not all of the above listed steps of the algorithm should be looked at as results which cannot be improved. Especially the performance of the very first operations and the last ones is tightly coupled to the specific memory interface since during these stages the images all have to be fetched from and written back to external storage elements as they are too large to be stored internally. Thus, care should be taken when directly comparing the timing data for the image pre-processing, the binning and the image compression in particular since the SD card would certainly be replaced by a faster memory type if the ARM processor is chosen for the on-board processing platform.

TABLE VIII. EXECUTION VERSION OF THE VHDL, C AND C++ VERSION FOR THE ONE-SHOT MODE

Algorithm Operation	VHDL [s]	C [s]	C++ [s]
Pre-Processing	3.167	0.0019	n/a
Binning with Dual-Port RAM	0.791	21.681	n/a
Binning with Single-Port RAM	0.840		
Median Calculation	n/a	0.698	n/a
Star Detection with Dual-Port RAM	0.051	0.338	n/a
Star Detection with Single-Port RAM	0.052		
Displacement Vector	0.001	0.002	n/a
Second Binning	0.049	0.470	n/a
Difference	0.049	1.002	n/a
ROI Selection	0.005	0.005	n/a
Addition of stars	0.001		
Image compression	0.791	2.627	n/a
TOTAL with Single-Port RAMS	4,905	26.842	0.354
TOTAL with Dual-Port RAMS	4,905		

The DM in VHDL replaces the median calculation with an average calculation and had to deal with several issues during the design of the star detection despite finally being faster than the C implementation. During the displacement vector calculation, both images then do not rely on the external memory anymore and the evaluation becomes a lot clearer. The Difference Method in VHDL is optimized not to waste a single clock cycle when processing the images during the second binning and the difference. Both steps of the Difference Method now turn out to perform better on the FPGA. Once it is about composing the result image out of regions of interest and reference stars, the ARM takes the lead.

It is faster for both operations since in C an index vector for each row of the image is calculated instead of explicitly copying image content from one RAM to another one as it is done in VHDL. Then finally, for the image compression, both versions need to communicate with the external storage again. Here, the VHDL version one more time benefits from using fast SRAM components.

A noticeable drawback of the C implementation is that it currently only uses one of both cores of the ARM Cortex R5F. As a result, when used in One-Shot Mode, the ARM needs to apply the image pre-processing, the binning, the median calculation and the star detection to both images before it can continue with the displacement vector calculation since no previous results are available yet. Especially the binning, where large amounts of data have to be transferred over a slow memory interface, dramatically slows down the ARM's performance. Here, the VHDL version benefits from having all execution units duplicated so that every operation can be applied to both images at once, significantly improving the result of the One-Shot Mode.

The C++ version in direct comparison performs the algorithm significantly faster than both candidates for the on-board processing platform. With a clock frequency of 3.4 GHz the desktop version is clocked 11 times higher than the ARM processor and 651 times higher than the FPGA. In addition, the C++ version can make use of an optimized DRAM access benefitting from DDR protocols. With 8 GB of RAM, it also does not suffer from insufficient DRAM resources like the ARM-based implementation, forcing it to use alternative, slower technologies, and can easily store many hundred input images at once.

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) have been 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. As an alternative to the RTG4, both the NG-MEDIUM and the NG-LARGE FPGAs from NanoXplore were also successfully tested, even if the design needs some corrections.

After extensive testing, the conclusion of the project is that the ARM Cortex R5F is a promising candidate to implement the Differences Method algorithm for a real mission. The expectation was that Microsemi's RTG4 FPGA would also perform well within the requirements with the Differences Method. However, due to its low cost and robustness, the final recommendation [17] on what platform shall be selected is the ARM Cortex R5F provided with an extensive RAM memory block to ensure fast read/write access times.

6. REFERENCES

[1] Klinkrad, H. 2006. "Space Debris: Models and Risk Analysis". Springer-Verlag Berlin, 1st Ed.

[2] ESA – Space Debris. 2017. "Space Debris by the numbers". URL: http://www.esa.int/Our_Activities/Operations/Space_Debris/Space_debris_by_the_numbers

[3] ESA Space Situational Awareness Group. 2012. "Space Station Manoeuvres to Avoid Space Debris". URL: http://www.esa.int/Our_Activities/Operations/Space_Situational_Awareness/Space_Station_manoeuvres_to_avoid_space_debris

[4] Atkinson, N. 2012. "ISS will do maneuver Friday to avoid collision with satellite debris". Universe Today. URL: https://www.universetoday.com/92571/iss-will-do-manuever-friday-to-avoid-collision-with-satellite-debris/

[5] Hutchinson, L. 2013. "How NASA steers the International Space Station around space junk". Ars TECHNICA. URL: https://arstechnica.com/science/2013/07/how-nasa-steers-the-international-space-station-around-space-junk/

[6] PACT XPP Technologies. 2017. "Processor Licensing". URL: http://www.pactxpp.com

[7] 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.

[8] Microsemi Corporation, 2017. "RTG4 FPGA Datasheet". URL: https://www.microsemi.com/document-portal/doc_view/135193-ds0131-rtg4-fpga-datasheet

[9] The Consultative Committee for Space Data Systems CCSDS. Image Data Compression, Recommended Standard CCSDS 122.0-B-1, Blue Book, 2005.

[10] Daëns, D. 2016. "Feature Detection on new Space FPGA Technology". Master Thesis. Airbus DS GmbH, unpublished.

[11] Köthe, U. 2003. "Integrated Edge and Junction Detection with the Boundary Tensor". Proceedings of the 9th IEEE International Conference on Computer Vision.

[12] 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, unpublished.

[13] 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.

[14] 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.

[15] XPP-III reference Manual – XPP Dataflow Array. PACT XPP Technologies AG, 2009.

[16] The Consultative Committee for Space Data Systems CCSDS. Image Data Compression, Informational Report CCSDS 120.1-G-1, Green Book, 2007.

[17] Utzmann, J., Ferreira, L., Vives, G., Strasser, N., Probst, D., Lièvre. 2018. "Optical In-Situ Monitor - A Breadboard System to Enable Space-Based Optical Observation of Space Debris". 69th International Astronautical Congress (IAC). Bremen, Germany, 2018.