Streak Detection for Space Debris observation

Gerard Vives Vallduriola [(1)], Diego Andrés Suárez Trujillo [(1)], Tim Helfers [(1)], Damien Daens [(1)], Dr. Jens Utzmann (2) Jean-Noel Pittet (3), Nicolas Lièvre (4)

[(1)] gerard.vives@airbus.com, diego.suarez.external@airbus.com, tim.helfers@airbus.com, damien.daens@airbus.com. *Airbus DS GmbH. Robert-Koch-Str.1, D-82024 Taufkirchen*

[(2)] jens.utzmann@airbus.com. *Airbus DS GmbH. Claude-Dornier-Str., D-88090 Immenstaad*

[(3)] jean-noel.pittet@aiub.unibe.ch. *Astronomical Institute of the University of Bern (AIUB) Sidlerstr. 5, CH-3012 Bern*

[(4)] Nicolas.lievre@micos.ch. *Micos Engineering GmbH. Überlandstr. 129, CH-8600 Dübendorf*

ABSTRACT

Man-made space debris orbit around the Earth at many altitudes in different orbits and many launches add further debris to already congested orbits. The ESA GSTP-funded Optical In-Situ Monitor project started in 2016 and aims to analyze the End-to-End processing pipeline from image acquisition to the extracted positions of space debris. A camera will take images and an on-board data processor will filter and compress the images obtained, which – in case of a future Flight Model of the instrument - shall be downlinked to the ground station. For the image filtering and compression, several algorithms are considered and also two different kinds of processors are considered. The difficulty of the images is to differentiate between stars in the background (which have to be filtered out) and the debris. We present a short overview of the project, the current status and the algorithm trade-off.

Keywords

Space debris, High Performance, Reliable hardware, reprogrammable hardware and in-orbit flexibility, boundary tensor, FPGA RTG4, differences method.

1. INTRODUCTION

The space environment presents a level of radiation which is significantly higher than the one encountered on ground or even avionic levels, mainly because of galactic cosmic rays and different solar phenomenon like the solar flares, coronal mass ejection and solar. Some of these particles can get through spacecraft shielding and irradiate electronic components. This is especially true in the Van Allen belts, where some protons and electrons are trapped and can induce errors in the behaviour of electronic systems and change their behaviour. Single-Event Transient (SET), Single-Event Upset (SEU) and Single-Event Functional Interrupt (SEFI) are perturbations that can be recovered, but the equipment can also receive permanent damage under the form of a Single-Event Latch-up (SEL) or a Single-Event Gate Rupture (SEGR). [1].

Human activity in space has brought advances in communications, security, earth observation, space exploration and other fields of science, but at the same time has contributed to populate earth orbit with human-made defunct particles, such as residues from rocket firings, de-commissioned satellites and its fragments from deterioration [2]. Man-made space debris orbit around the Earth at many altitudes in different orbits. Collisions and some launches add further debris to the already congested debris packs. As an example, the ISS has to perform several manoeuvers every year to avoid catastrophic collisions with space debris flying by.

The following figure [4] shows how space debris has increased in the past decades.

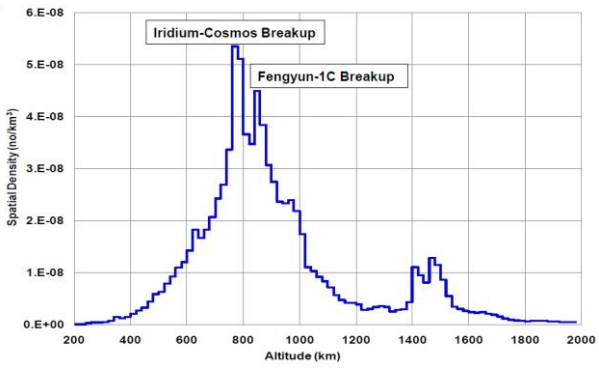

Figure 1 Spatial density of LEO space debris by altitude, according to 2011 a NASA report to the United Nations Office for Outer Space Affairs

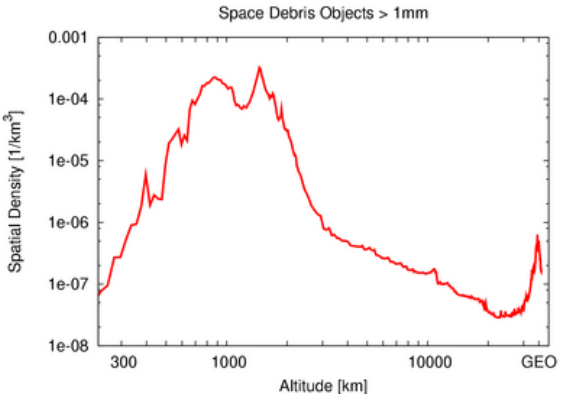

Figure 2 Spatial density [5] of space debris by altitude according to ESA MASTER-2001, without debris from the Chinese ASAT and 2009 collision events

The ESA-funded IN-SITU project started in 2016 and aims at observing and taking pictures from the space debris. This study uses previous GSP results and actually aims at test-bed and bread-boarding to reduce the risk of a future mission. The next step of this study would be an engineering model and a mission.

In this mission, a camera will take images and an on-board data processor will filter and compress the images obtained, which – in case of a future Flight Model of the instrument - shall be downlinked to the ground station. For the image filtering and compression, several algorithms are considered and also two different kinds of processors are considered: an XPP-based high-performance data processor (HPDP) and Microsemi's RTG4 FPGA. The difficulty of the images is to differentiate between stars in the background (which have to be filtered out) and the debris. We present a short overview of the project, the current status and the algorithm trade-off.

This article is organized in the following way: Chapter 2 gives an overall view of the IN-SITU project. Chapter 3 describes the candidate processors for the image processing. Chapter 4 describes the algorithms currently being considered for streak detection with some simulation results. Chapter 4 presents the conclusions reached so far.

2. IN-SITU Mission

ESA started in 2013 the study "Assessment Study for Space Based Space Surveillance Demonstration Mission (Phase A)" with the goal to analyse the feasibility of an optical space-based space surveillance (SBSS) demonstration mission and to consolidate the design approach. The SBSS Demonstrator instrument was originally designed to be compatible with existing microsatellite platforms for a dedicated mission. The next step was IN-SITU.

The breadboard software will have the structure of a processing chain, where the main parts are: Segmentation, Object recognition, Astrometry and Photometry, Tracklet building.

In the segmentation part all image objects (stars, moving objects and cosmic rays) are identified by defining a threshold value depending on the background, and the pixels values over the threshold are selected. In the space-based scenario, since the observed objects have high angular velocities the streaks are very faint and long on the frames, with a small signal-to-noise ratio (SNR ~ 1) at the single pixel level. The difficulty here is to find an algorithm able to distinguish the faint streaks from the background. Different algorithms have been proposed for the segmentation purpose, based on various image analysis methods, such as spatial and temporal filtering and convolution, local or regional averaging, gradient-based methods, region-based methods, contour methods, and different image transforms (e.g. FFT).

In the object recognition part, the moving objects are identified and separated from the stars or other artefacts. The identification phase implies finding all pixels belonging to the same object, which is difficult when the streak is non-uniform, or overlaps stars and other streaks. The discrimination of moving objects from stars can be performed either on single frames on the basis of the different morphology of the object and the star images (trails vs. round images; known lengths and orientation of star images) or by comparing their positions on consecutive frames (i.e. during the tracklet linking step). In case that the object's angular velocity on the image is similar to the stars angular velocities only the second method is applicable. None of the object discrimination algorithms currently used has been optimized for on-board processing applications. After the identification, the exact position of the object needs to be determined (centroiding). This takes the identified objects pixel into account and makes use of brightness and shape of the object structure.

An important step in the reduction of the observations of any object is the determination of its position (e.g., α and δ) in a celestial reference frame. This reduction consists in essence of the determination of parameters of a transformation between the reference frame on the celestial sphere and the coordinate system in the focal plane (mapping model). Apart from the projection of the sphere onto a plane this transformation contains all kinds of distortions generated by the telescope optics, as well as characteristics of the 'measurement system' of the detector. The astrometric reduction uses the celestial coordinates of astrometric reference stars together with the measured positions on the CCDs of the same stars to determine the transformation parameters. In the photometry the magnitude of the observed object is determined according to photometric reference stars.

In tracklet building, the observations of the same moving object on different frames are grouped together forming a so-called tracklet. The usual algorithms try to associate the observations based on the velocity and direction of the object in the frame coordinates. In the case of three observations, for example, the velocity and direction extracted from two observations can be used to associate a third observation, assuming that they are approximately constant over the short interval between three consecutive frames. This step indirectly also acts as a filter for the cosmics still present after the object recognition phase, since the cosmic events are randomly distributed and in general not aligned.

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. 100Mbps and a required integer processing rate of at least 5GOPS with at least 16bit accuracy. Baseline for this assessment is a frame rate of one per second 2Kx2K image and a feature detection algorithm for data reduction. A buffer memory of 128Mbyte must be envisaged for intermediate storage of the image during the processing steps. Power Consumption for the processing part should not exceed 20W.

The suitable technologies for the processing units can be summarised into three different categories: General Purpose Processors/DSPs, FPGAs/ASICs and Specialised Processing Units. GPPs, and also in part Digital Signal Processors (DSP), provide the easiest development environment and highest developer productivity but the throughput rate is rather low compared to FPGA or Specialised Processing Units and the power consumption relatively high. 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. combine multiple cores with different characteristics to allow efficient mapping of algorithms with high processing demand. Most of these 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. The proposed HPDP falls into this latter category.

3.1 HPDP [6]

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. 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 Elements (PAEs) connected by a reconfigurable data and event network. Two types of PAEs exist: a RAM-PAE and an ALU-PAE. The vertical data and event routing channels are always contained within a PAE, in the form of Forward Registers (FREGs) that route vertically from top to bottom, and Backward Registers (BREGs) that route vertically from bottom to top.

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

The following figure shows the block diagram of this core architecture.

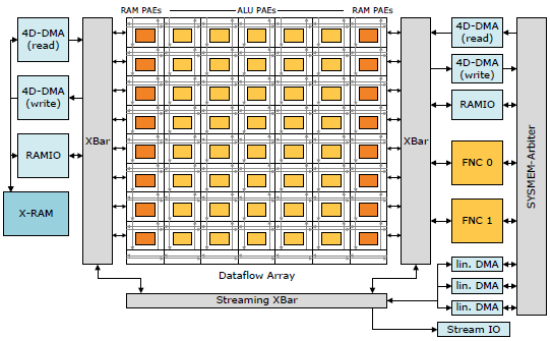

Figure 3 XPP core architecture, used for HPDP

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 as in the current generation of FPGAs. Therefore the device is announced to be suitable for signal processing tasks in satellite applications. Airbus DS in Ottobrunn evaluates this device using an evaluation board from Microsemi by porting one of their GNSS applications onto the technology. The GNSS application contains the signal processing functions suitable to assess the technology for this kind of purposes.

A block diagram of the device is depicted in the following figure:

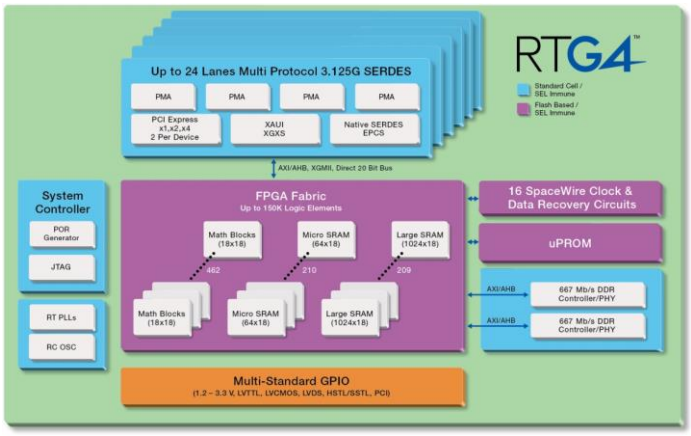

Figure 4 RTG4 device block diagram [7]

4. CANDIDATE ALGORITHM

4.1 Boundary Tensor [1]

The boundary tensor [8] is a symmetric and positive semi definite tensor. Its non-negative eigenvalues are defined by:

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

With the tensor being

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

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 polar 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. I(x,y) is the intensity of the pixel at the coordinate (x,y) and K_i are the coefficients of the filter:

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

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.

After having simulated several algorithms on the HPDP (compression, boundary tensor, and communications) Airbus can point out which algorithms are most appropriate for its 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 boundary tensor is finally calculated by adding the odd and even energy, which is equivalent to adding T_{odd} and T_{even}.

The final step of the algorithm is to determine if a pixel corresponds to a 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).

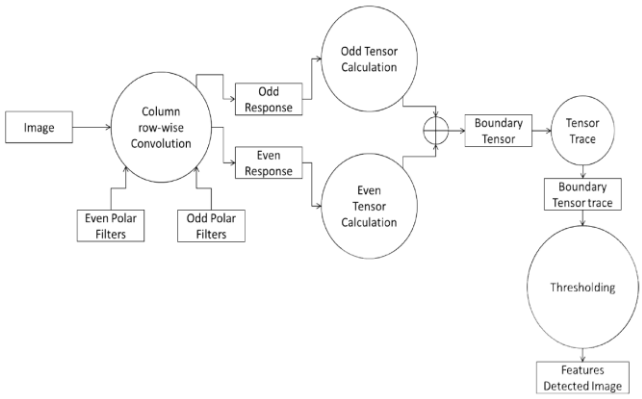

Figure 5 Description of the architecture of the boundary tensor algorithm

The final results are combined to calculate the diagonal coefficients of the boundary tensor, then its trace. The final step is applying a thresholding, to get a binary picture in which the pixels set to 1 are debris.

4.2 Differences method [9]

The basic idea is to compare two frames to detect the changing features. The registration (alignment, rotation 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 subframes (called cells in the following paragraphs).

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 in (5 by 5) cells and estimate the displacement vector for each of them.

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

5. Subtract the two frames.

6. The big pixels over (under) a threshold are selected as potential streak of the frame A(B).

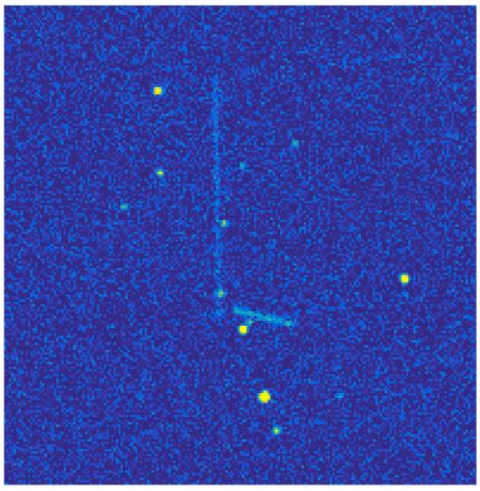

Figure 6 Original simulated frames. SNR of streaks are 1.5 and 4 in this frame A

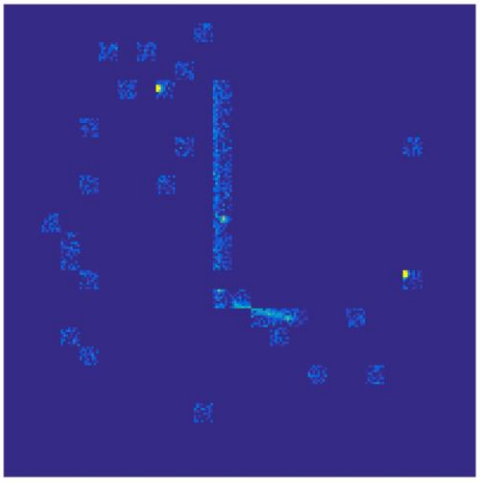

Figure 7 Output of the differences method where input was Figure 6

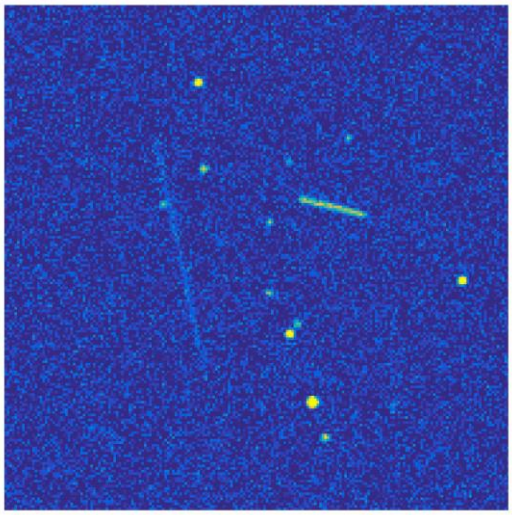

Figure 8 Original simulated frames. SNR of streaks are 1.2 and 6 in this frame B

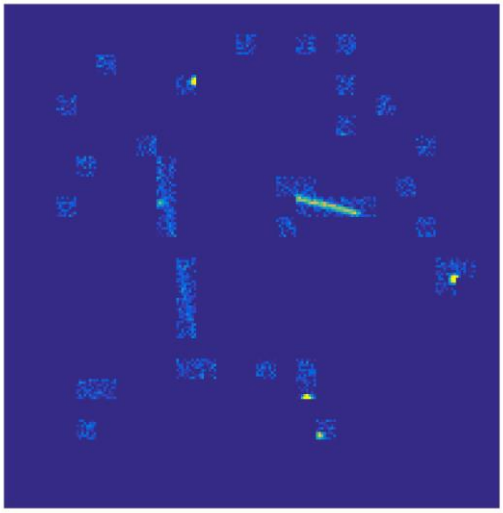

Figure 9 Output of the differences method where input was Figure 8

4.3 Algorithm trade-off

To be able to test the way the architecture works, a Simulink block has been made.

Figure 10 Input picture provided by AIUB

Only a part of this picture was selected in order to get a 2048 by 2048 frame, as required (this picture is visible in Figure 6). The Simulink model was launched, and its output signal was used to build the binary output of the processed picture, visible on the figure below.

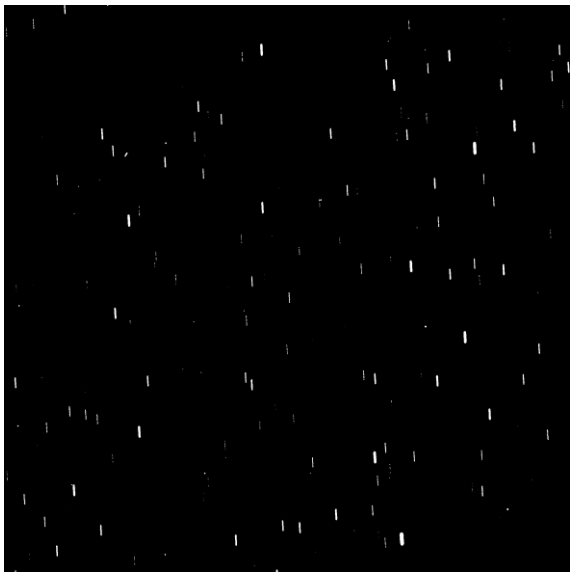

Figure 11 Result obtained by the RTG4 boundary implementation

A zoom on the picture shows that a little bit of noise passes as debris, like the dot of figure 9, which does not seem to be a RSO.

Figure 12 Zoom into the input picture

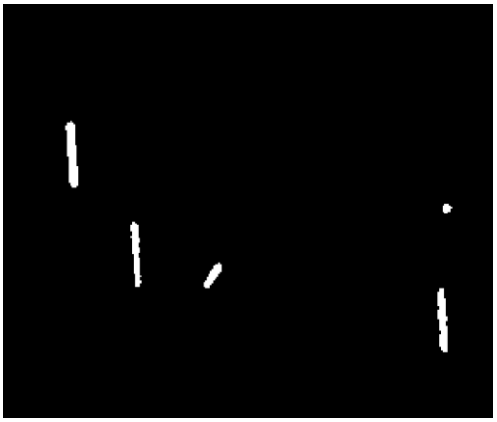

Figure 13 Zoom into the output picture

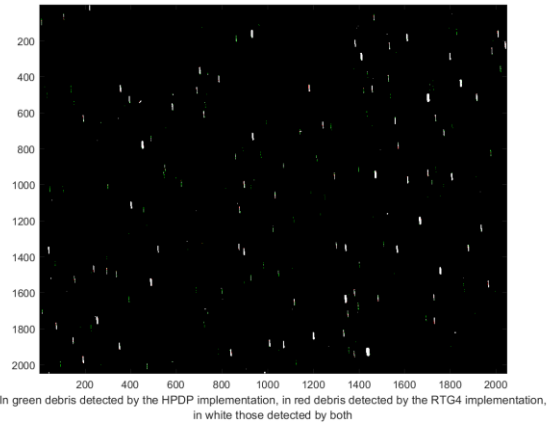

Figure 14 Debris detected by both architectures

Some small differences can be observed between the two results. After an analysis of the dataflow of the two architecture, 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. This difference means that even if the kernel coefficients sent to the two implementations were the same, the one truly used were slightly different. The kernels used by the HPDP simulation must make the architecture a little bit more sensible, as it seems to detect a little bit more debris,

but also gets a little bit more noise. As a conclusion, the two architectures are comparable in term of output results; the only difference is that the HPDP version changes the kernel coefficients and the threshold value while scaling them up.

Concerning the time needed to process a picture, the HPDP implementation needs 0.734s, while the RTG4 only needs 0.06s, which is much faster than the requirement of 1s 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 the calculus of the output, without needing to write any data in an external memory, unlike the HPDP.

5. CONCLUSION

We have presented both the space debris problem and the IN-SITU mission. Also, the two architectures currently considered for the image processing and the algorithms considered: the boundary tensor and the differences method. The RTG4 implementation of the boundary tensor has results which match those attained with the HPDP, with the small problem of the scaling of the kernel coefficients and the threshold value. Thus, the two implementations need different parameters in order to achieve the exact same results, but they both work the same intended way. The project will now test the suitability and performance of the differences method in order to decide which combination of processor-algorithm is the most suitable for the mission.

6. REFERENCES

1. Daëns, D. (2016). *Feature Detection on new Space FPGA Technology*. Master Thesis. Airbus DS GmbH.

2. Klinkrad, H. (2006). *Space Debris: Models and Risk Analysis*. Springer-Verlag Berlin, 1 Ed.

3. Utzmann, J., Ferreira, L., Pittet, J-N., Helfers, T., Vives Vallduriola, G. et al. (2016). *Breadboard Instrument & Test Set-Up Architecture Design Report*. IN-SITU_ADR iss. 1, rev. 0. Airbus DS GmbH.

4. Public Domain,

https://commons.wikimedia.org/w/index.php?curid=30598607

5. Public Domain,

https://commons.wikimedia.org/wiki/File:Spacedebris_small.png

6. http://www.pactxpp.com

7 http://www.microsemi.com

8. Köthe, U. *Integrated Edge and Junction Detection with the Boundary Tensor*.

9. Pittet, J-N. (2016). *Design & Specification Report SW*. IN-SITU DD-0007c.v1.0.AIUB.

7. ACKNOWLEDGEMENTS

We hereby acknowledge and say thank you to both the ESA and ESOC for their support in the IN-SITU project and its predecessor SBSS. 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 Nr. 4000116518/16/D/SR.