AN EFFICIENT AND AUTOMATIC DEBRIS DETECTION FRAMEWORK BASED ON GPU TECHNOLOGY

Francesco Diprima[(1) (2)], Fabio Santoni[(1)], Fabrizio Piergentili[(1)], Vito Fortunato[(2)],
Cristoforo Abbattista[(2)], Leonardo Amoruso[(2)]
[(1)] La Sapienza, [(2)] Planetek Italia
diprima@planetek.it, fabio.santoni@uniroma1.it, fabrizio.piergentili@uniroma1.it, fortunato@planetek.it,
abbattista@planetek.it, amoruso@planetek.it

1. ABSTRACT

This paper presents the implementation of an optimized and performance-oriented pipeline for sources extraction intended to the automatic detection of space debris in optical images. Algorithm reliability has been demonstrated in a prototypal environment, while the overall process automation and efficiency have been the main drivers in its final implementation. The performance advantages obtained with the huge degree of processing parallelism provided by General Purpose computing on Graphics Processing Units (GPGPU) are analyzed and demonstrated here: splitting data analysis over thousands of threads allows for big datasets processing with a limited computational time. The implementation has been tested on a large and heterogeneous images data set, containing both imaging satellites from different orbit ranges (low, medium and high orbits) and multiple observation modes (i.e. sidereal and object tracking).

2. INTRODUCTION

The higher awareness of the space debris threat has triggered the need of a distributed monitoring system for the prevention of possible space collisions (as discussed in Reference [1]). The increasing number of dedicated sensors allows for a wide and continuous monitoring of the space environment and then for an accurate knowledge of debris and their orbit determination [2-4]. Alongside with this trend, the need of automatic data analysis has being enhancing its importance in order to manage the increased images volume and to provide a quick and reliable toolbox able to identify candidate space debris and support in their analysis. Each possible debris characterization needs in fact an identification phase; orbit and attitude determination and finally the collision risks estimation are the next steps [5-8].

We present an optimized and performance-oriented pipeline for sources extraction intended to the automatic detection of space debris in optical images. In our work the object detection does not need auxiliary information, neither about the image acquisition (i.e. observed zone and orbital regime of the observed object), neither the star catalogue to perform stars subtraction before detecting streaks. Furthermore it is based on the analysis of a single image, thus the acquisition of consecutive frames of the same field (for stars field subtraction) is neither needed. The algorithm is able to detect both kinds of features can be found in the optical image (streaks and point-like objects), so allowing its adoption in both the observation modes: sidereal tracking in which the star are point-like object and the space debris are streaks and object tracking in which features' significance is inverted. We propose the use of the GPGPU in the pipeline for sources extraction. The use of the GPU in the image processing technique allow to parallelize the operation with a considerably reduction of the detection process time. Moreover, a validation study was performed on a large and heterogeneous dataset containing satellites from low to high orbits.

Proc. 7th European Conference on Space Debris, Darmstadt, Germany, 18–21 April 2017, published by the ESA Space Debris Office
Ed. T. Flohrer & F. Schmitz, (http://spacedebris2017.sdo.esoc.esa.int, June 2017)

3. IMAGE PROCESSING ALGORITHM

The pipeline for sources extraction is composed of three main phases.

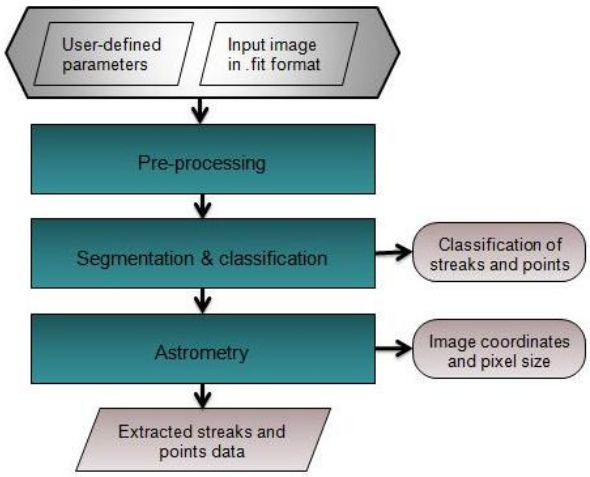

Figure 1 Pipeline for sources extraction

First one is pre-processing where a set of non-linear digital filtering techniques are applied to the greyscale image, achieving a noise reduction. The second phase consists in segmentation and classification where all the information about the connected components from the binary image are extracted and subsequently classified. Finally, the astrometry phase performs the astrometric reduction of the detected objects.

3.1 Pre-processing

In the preprocessing phase, we are interested to elaborate the input data in order to reduce the noise and prepare the image for the segmentation.

The first operation is the histogram stretching, which is a technique used to improve the contrast of an image by stretching the original dynamical range of intensity in a desired range [9]. To perform the stretching it is necessary to specify the upper and lower input pixel value limits over that the image has to be normalized, and select the desired output range values.

Typically, space debris image data are stored as 16-bit grayscale images in "fit" format; thus, in order to contextually reduce image dimension while preserving and highlighting image features, a remap of data in an 8-bit grayscale images, with pixel value in the range from 0 to 255, is performed.

Once obtained such stretched image, a median filter is applied with the purpose of noise removal.

The median filter is a nonlinear filter whose response is based on ordering pixels contained in the mask and then replacing the central point of the mask with the median value [10]. Median filter removes random impulse noise, it provides excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. The median image is then analyzed to estimate the image background.

To detect faintest objects in the image it is necessary to compute accurate values of background level in the image. In the context of space debris image processing, we identify as background all the pixel that have not gathered photons coming from a star or a space debris. The analog-to-digital units (ADU) value of the background pixels is the result of the sum of photons coming from the black sky, plus a delta result from the effect of dark current and read noise.

In order to take account of variation of the background level in the image, a local analysis of the image is performed. The local statistics analysis estimates the background values in each mesh of a grid covering the whole image. For each mesh the mean μ and standard deviation σ value are computed. Iteratively, all pixels with value higher than $\mu + 2.5\sigma$ are discarded and a new value of standard deviation is calculated considering the remaining pixels, until the percentage difference of the standard deviation change less than 20%. Assuming that space debris images are not crowded and that most of the pixels represent the background sky, this iterative process allows to reject all the pixel belonging to a foreground object as a star or space debris.

The obtained image is then subtracted from the median image to obtain a background-subtracted image, in

which are present only foreground pixels. To suppress possible local overestimations due to bright stars, another median filter is applied at the background-subtracted image.

3.2 Segmentation

The image segmentation includes all those operations that tend to partition an image into significant regions. The purpose of segmentation is to simplify the image information in order to make easy the features extraction. The first operation in image segmentation procedure is the image binarization.

The transformation in binary scale reduces the informative content of the image splitting the pixel in only two categories, foreground and background. Furthermore, the binary transformation decreases the storage space and increases the code performance. Two binary images are then obtained from the median background-subtracted image, one for streaks and another for points detection purpose. The difference of these two images defines the used threshold: a higher threshold enables the detection of point-like objects characterized of high ADU pixel value due at the concentration of photon on a little region of the CCD array, while a lower threshold optimizes the detection of streaks characterized of a lower ADU pixel value due at the spreading of the photons coming from a moving object. At this point, the two binary images following different segmentation procedure in order to enhance the researched features.

3.2.1 Segmentation for streaks detection purposes

In order to detect streaks, the distance transform is applied to the binary image.

The distance transform of an image is defined as a new image in which every output pixel is set to a value equal to the distance to the nearest zero pixel in the input image. The distance transform is performed by using a mask of 3-by-3 pixels, in which each point in the mask defines the distance to be associated with a point in that particular position relative to the center of the mask [11].

The distance transformation result values are then normalized and the threshold to obtain the peaks value corresponding to foreground objects is defined.

Considering this normalized distance transformed image, a morphological dilatation filter is applied.

The morphological operations are non-linear segmentation techniques exploiting the mathematical morphology to isolate or connect objects in the image. The application of a dilatation operator, actually bridges gaps and connects disjoint parts of the same object resulting from a threshold operation. In our case we have selected a square structural element of 3-by-3 pixels.

Then, aiming at measuring the streaks inclination angle, the Standard Hough transform is computed on the result image. Finally, the results of the Standard Hough transform are used to apply a morphological opening filter. The morphology opening is the combination of the two basic morphological operations, an erosion followed by a dilatation. The morphological opening effects are preserve regions with shape similar to the structural element and deletes different ones. Using a linear structural element rotated of an angle α correspondent at the peaks of the Standard Hough transform we preserve all the streak-like objects and delete all the other features. This operation is applied to the first peaks of the Standard Hough transform in order to take into account the possibility of a space debris image with different object on diverse orbit.

3.2.2 Segmentation for points detection purposes

The first operation to detect the point-like objects in the image is the application of a convolution filter. A square kernel of 3-by-3 pixels scans the image and replaces all image pixels under the central point of the kernel with the value 1 if the sum of the image pixel under the

kernel is higher than a threshold. An higher threshold value allows to delete single points as hot pixels or cosmic ray, while contextually fill little holes in the object and clean the object contours.

Following the application of convolution filtering, a morphological opening operation is performed. Using a circular structural element we obtain the removal of linear object as streaks or noise effect preserving the point-like objects.

Finally, to ensure the deletion of all streak-like objects from the image, a subtraction operation is performed between the morphological opening image and the binary image obtained in the segmentation for streaks detection purpose image.

With the end of the segmentation phase we presume that the obtained images allows identifying clearly the object contours and then classify them as stars or streaks.

3.3 Classification

This phase assumes that the input image is a well-segmented binary image; in which are present only the relevant features describing the geometrical characteristics of the objects. The classification phase starts with the identification of the objects contours. A contour is a list of points that represent a curve in an image. We assume that a pixel is a contour pixel if it is a white pixel and if it has at least one adjacent black pixel in his surroundings. Finally to obtain all pixels inside the contour, the Ray-casting algorithm is applied.

Once terminated this identification phase, all the detected objects are measured to distinguish if they are stars or streaks.

To compute the object barycenter and elongation we use the formulation of the image Moment [12]:

$$M_{pq} = \sum_x \sum_y x^p y^q I(x,y) \tag{1}$$

being x and y the coordinate of the pixel belonging at an object. By this definition we obtain that the Moment of zero order

$$M_{00} = \sum_x \sum_y I(x,y) \tag{2}$$

is the area of the object express in pixel and

$$\bar{x} = \frac{M_{10}}{M_{00}} \quad \bar{y} = \frac{M_{01}}{M_{00}} \tag{3}$$

are the coordinate of the object's barycenter.

Other descriptors of the object are the Central Moments

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q I(x,y) \tag{4}$$

and the centered Central Moments

$$\mu'_{pq} = \frac{\mu_{pq}}{\mu_{00}} \tag{5}$$

The centered Central Moments are used to describe the object as an elliptical shape, obtained the semi-major a and semi-minor b axis of the ellipse

$$a^2 = \frac{\mu'_{20} + \mu'_{02}}{2} + \sqrt{\left(\frac{\mu'_{20} - \mu'_{02}}{2}\right)^2 + \mu'^2_{11}} \tag{6}$$

$$b^2 = \frac{\mu'_{20} + \mu'_{02}}{2} - \sqrt{\left(\frac{\mu'_{20} - \mu'_{02}}{2}\right)^2 + \mu'^2_{11}} \tag{7}$$

With these measures explained above, we classify all the detected objects as point-like or streak-like objects.

A first selection of the objects is performed taking into account the object's dimension. A detected object is rejected if the zero order Moment is lower than a threshold σ. For the threshold σ, a value of 5 has been selected, in this manner all the false positive detection objects resulting from noise or artifact are discarded.

The remaining objects are then studied to classify them as point-like or streak-like objects. The study is based on the analysis of the object's semi-major and semi-minor axis.

An object is classified as point-like object if the following equation is satisfied:

$$\frac{a}{b} < \rho \tag{8}$$

Ideally, the ρ parameter should be equal to one; but considering the spreading of the photons, the effect of the image noise, and the artifact due to the image processing it is selected a value of $\rho = 1.6$.

On the contrary, an object is classified as streak if the following inequality is satisfied

$$\frac{a}{b} > \delta \tag{9}$$

being $\delta > \rho$. The δ parameter is function of the observed object orbit, exposure time, optical and camera features. Its value has been selected after a study of the usual dimension and shape of the space image features taking in different orbital debris, from LEO to GEO, in order to be suitable for every image type.

The δ parameter has been tested for LEO space debris image taken with long exposure time, characterized of very long streak in the image FoV, and for GEO image taken with a short exposure time (comparable with the first one) that present very short streak.

3.4 Astrometry

Finally, for each detected object, the right ascension α and declination δ angles are measured using the information contained in the USNO-B catalog. The measurement is performed with the assumption of lost in space, in which neither sky zone nor image scale are required.

The object's celestial coordinates on CCD image are calculated by the plate reduction [13]. This technique exploits the geometric gnomonic transformation to transform coordinates of the tangential CCD plane to the celestial coordinates.

The plate reduction is computed using at least four stars, and results of this operation are the image reference point $P_{I,ref}$ in image coordinates and in celestial coordinates $P_{C,ref}$ and the transformation matrix M that take into account the image rotation, skew and scaling.

The conversion from image coordinates to celestial coordinates is obtained by the following formula

$$\begin{bmatrix}\alpha\\ \delta\end{bmatrix} = \begin{bmatrix}m_{11} & m_{12}\\ m_{21} & m_{22}\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix} + \begin{bmatrix}P\alpha_{C,ref}\\ P\delta_{C,ref}\end{bmatrix} \tag{10}$$

in which u and v are the relative pixel coordinates with origin in the image reference point $P_{I,ref}$

$$\begin{aligned} u &= x - Px_{I,ref}\\ v &= y - Py_{I,ref}\end{aligned} \tag{11}$$

The matrixes together define a unique transformation from pixel coordinates to the plane-of-projection.

4. GPGPU

The use of General Purpose computing on Graphics Processing Units (GPGPU) is a technological choice aimed at increasing the computational performance of scientific and engineering applications for large scale parallel processing applications [14]. In 2006, the NVIDIA Company with the Compute Unified Device Architecture (CUDA) has released the Application Programming Interface (API) and Software Development Kit (SDK) with the intent to simplify the accessibility and the use of the GPU. The CUDA-C language is totally integrable with the C++ code and its APIs are quite similar to those of the C language making it easy to understand.

Now let us examine how the GPU works. In general the hardware of a computer is divided in host, which is a traditional CPU architecture, and device, which is massively parallel processor as GPU.

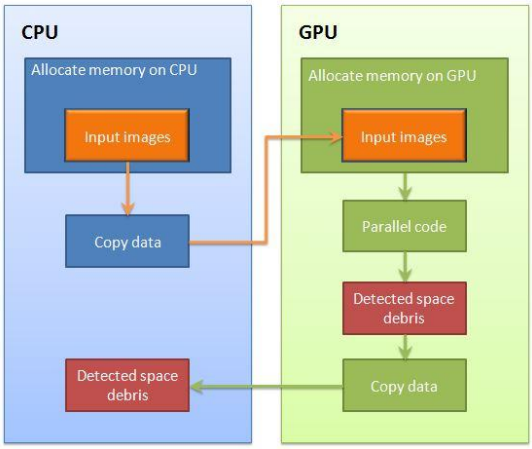

Figure 2 Heterogeneous programming

The host drives the computational process by a CUDA program. The CUDA program is a heterogeneous code consisting of many parts having phases that can execute both on the host and device, thus having a unique source code that contain both host and device code. The host code is written in C++, while the device code is written in CUDA-C code, which is an extended version of the C language with special keywords for labeling data-parallel kernels and their associated data structures. Usually a CUDA program is composed at least of these phases: read input data, copy input data from host memory to device memory, process the data on the GPU by parallel kernel and finally copy result data from device memory to host memory

A kernel is a function written in CUDA-C language that executes parallel code, it can run only on NVIDIAs GPU. The kernel is executed by each GPU's thread; threads are identified by a unique ID, enabling the programmer to address different parts of GPU memory relative to the thread ID.

CUDA organizes threads in a Scalable Programming Model: the GPU's threads are grouped in block (mono-/bi-/tri-dimensional) and identified by means of a thread index called $threadIdx$; in turn blocks are grouped in a grid (mono-/bi-dimensional) and identified with a block index called $blockIdx$. The GPU has a finite number of threads per block and a finite number of blocks per grid, and therefore has a limit to the amount of parallel execution. If the number of blocks exceeds the max limit, then the GPU sequentially processes the maximum possible number of blocks, therefore a kernel can be considered as executed in parallel manner in function of the hardware limit.

Although the use of the GPU can achieve improvement in the code performance, the advantage over CPU computations is only gained if the mathematical problem is parallelizable on a large scale. Fortunately the image processing is one of the perfect fields to the adoption of the GPU: the image elaboration enables separation of the input data to process and large-scale parallelization due to the not dependence of the computation. For these reasons all the aforementioned image processing techniques have been implemented as GPU kernels and used to produce results for this paper.

Although the use of the GPGPU allows to obtain the code parallelization, is not always possible to speed-up the execution time because there are a number of limitations and hurdles that must be judiciously managed to achieve an effective system.

The first limitation of the GPU is due to the time it takes to transfer information between computer memory and GPU memory. The data needed for the computation are transferred from the pc memory to the GPU memory and after the processing back to the pc memory. This means that the time saved in the parallel processing must be great enough to warrant the time taken for transferring the data. Conversely, the CPU although take a long time to processes in synchronous the data not require the data transfer because they are already stored on pc memory. Additional limitations to GPGPU are due to the absence of high-level libraries, only basic libraries have been rewritten in GPU code, then some image processing techniques have been developed for this simulation. Another important limitation is the difficulty of code debugging and of error information due to a lack of structured of error handling.

5. RESULT

The proposed method was tested and validated on a set of data containing both imaging satellites from different orbit ranges and multiple observation modes (i.e. sidereal and object tracking).

Fig. 3 shows the detection results for the GEO image in which are present two space debris with different orbit, while Fig. 4 shows a space debris in LEO orbit. Detected streaks are marked in red while in green are marked the point-like objects that in these cases, of

images take with sidereal tracking, correspond respectively with space debris and star. An example of object tracking image is shows in Fig.5, in this case, unlike the others, the detected streaks and point-like object correspond respectively with star and space debris.

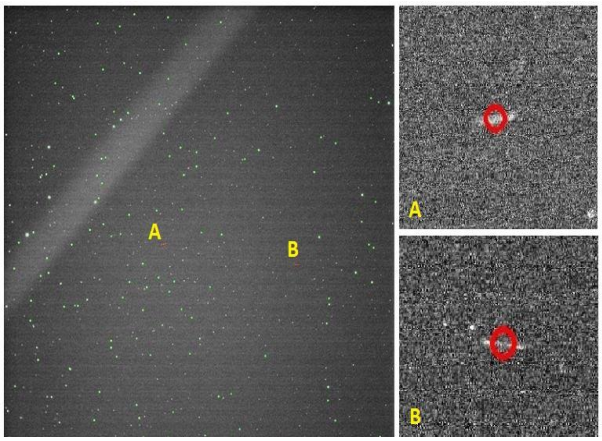

Figure 3 GEO image in sidereal tracking

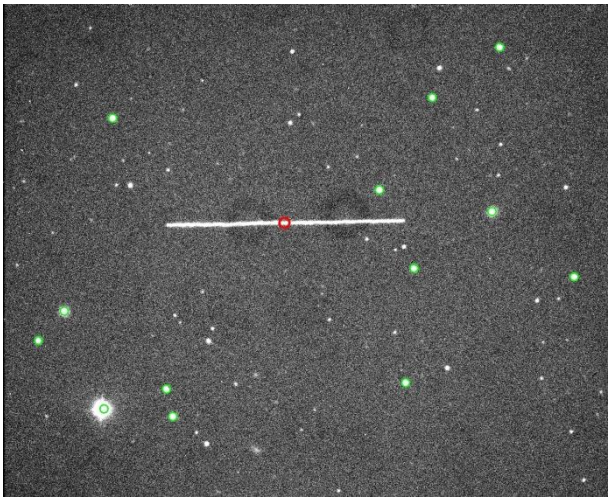

Figure 4 LEO image in sidereal tracking

During the test phase some detection errors were found, the error cases are:

- true positive, not existing objects detected;
- true negative, existing objects not detected;
- false positives, wrong objects detect as streak.

The most common cases of false positive detection are two close stars confused as possible streak or a fainter streak detected as two distinct ones (as show in the Fig. 5), while the true positive and negative errors are due to the image noise.

The hardware used in the simulation is the NVIDIA Jetson Tegra K1, a quad-core ARM Cortex-A15 CPU processor with 2GB of RAM with a NVIDIA Kepler GPU of 192 cores capable of over 300 GFLOP/s of 32-bit floating-point computation.

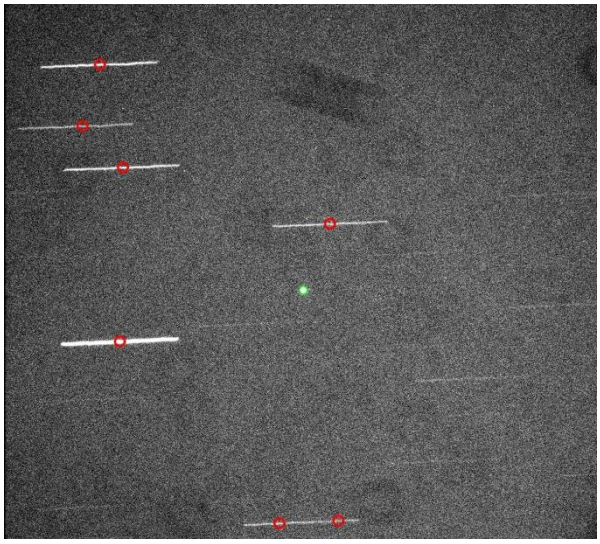

Figure 5 GEO image in object tracking

The mean execution time for image of 4096x4096 is 23.36 seconds for the CPU version of the pipeline and 3.19 seconds for the GPU version with a speedup factor of 7.3x.

The total number of processed images is 400, 547 objects were checked by visual inspection in 356 images. The pipeline detects correctly 469 objects in 308 images with a success ratio of 85%.

Considering that the analyzed image are in raw format, subtracting the bias frame and dividing by a normalized flat field it is reasonable to assume that the success ratio could be increase and that the detection errors could be decrease.

6. CONCLUSION

In the paper the speedup and the efficiency of the proposed pipeline for sources extraction is demonstrated.

The pipeline detects space debris without any a priori information and it is based on the analysis of a single image. The pipeline is able to detect space debris in both the tracking mode sidereal and object tracking.

A GPU approach was introduced in order to reach near real-time performance. The mean execution time for image of 4096x4096 is 3.19 seconds with a speedup of 7.3x respect the CPU version.

The pipeline was tested on 400 images with a success ratio of 85%.

7. REFERENCE

1. Klinkrad H. Et al, "Space Debris Activities in Europe". *Proceedings of the 4th European Conference on Space Debris*, 18-20 April 2005, ESA/ESOC, Darmstadt, Germany, p.25.
2. Piergentili F. et al. "EQUO: an Equatorial Observatory to improve the Italian space surveillance capability". *66th International Astronautical Congress*, Jerusalem, Israel, 2015, 12 – 16 October.
3. Diprima F., Cardona T., "Lessons learned in automatic operation of observatories for space debris observation". *67th International Astronautical Congress*, Guadalajara, Messico, 2016, 26 – 30 September.
4. Piergentili, F., Porfilio, M., Graziani, F., "Optical campaign for low earth orbit satellites orbit determination", (2005) *4th European Conference on Space Debris*, 8-20 April 2005, ESA SP-587, pp. 689-692.
5. Diprima F., "Automatic object tracking for space based space debris observation". *65th International Astronautical Congress*, Toronto, Canada, 2014, 29 September – 3 October.
6. Piergentili, F., Ravaglia, R., Santoni, F., "Close approach analysis in the geosynchronous region using optical measurements", *Journal of Guidance Control and Dynamics*, Vol.37, n.2, 2014, pp. 705-710. DOI: 10.2514/1.59821
7. Piergentili, F., Ceruti, A., Rizzitelli, F., Cardona, T., Battagliere, M.L., Santoni, F., "Space debris measurement using joint mid-latitude and equatorial optical observations", *IEEE Transactions on Aerospace and Electronic Systems*, 50, (1), 2014, pp. 664-675. DOI: 10.1109/TAES.2013.120272
8. Santoni, F., Cordelli, E., Piergentili, F., "Determination of disposed-upper-stage attitude motion by ground-based optical observations", (2013) *Journal of Spacecraft and Rockets*, 50 (3), pp. 701-708. DOI: 10.2514/1.A32372
9. Nixon M. S., Aguado A. S., *Feature Extraction and Image Processing*, Newnes, 2002.
10. G. Bradski e A. Kaehler, *Learning OpenCV*, O'Reilly, 2008.
11. Borgefors G., "Distance Transformations in Digital Images". *Computer vision, graphics, and image processing*, vol. 34, pp. 344-371 (1986)
12. Hu M. K., "Visual Pattern Recognition by Moment Invariants", *IRE Transactions on Information Theory*, vol. 8, pp. 179-187, 1962.
13. Kovalevsky J., Seidelmann P. K., *Fundamentals of astrometry*, Cambridge University Press, 2004.
14. NVIDIA Corporation, [Online], CUDA Toolkit Documentation v8.0.61, Available from: http://docs.nvidia.com/cuda/#, 2017