STREAKS DETECTION ALGORITHM IMPLEMENTED IN GPU FOR THE TOMO-E CAMERA AT KISO OBSERVATORY

Manuel Cegarra Polo[(1)], Toshifumi Yanagisawa[(1)], Hirohisa Kurosaki[(1)], Ryou Ohsawa[(2)], Shigeyuki Sako[(2)]

[(1)] Japan Aerospace Exploration Agency, 7-44-1 Jindaiji-Higashi-machi, Chofu, Tokyo 182-8522, Japan
cegarrapolo.manuel@jaxa.jp, yanagisawa.toshifumi@jaxa.jp, hirohisa.kurosaki@jaxa.jp
[(2)] Institute of Astronomy, Graduate School of Science, The University of Tokyo, 2-21-1 Osawa, Mitaka, Tokyo 181-0015, Japan, Email:ohsawa@ioa.s.u-tokyo.ac.jp, sako@ioa.s.u-tokyo.ac.jp

ABSTRACT

Images coming from the Tomo-e Gozen camera at Kiso Observatory in Japan, are composed of 2000 x 1128 pixels each, arranged in an array of 84 CMOS image sensors, that generates 13.6 GBytes of data per frame (when 18 consecutive frames are taken for each part of the sky). Typically 1.7 TBytes of data are produced per hour, and to automatically detect streaks effectively in these images, we have developed a fast algorithm implemented in a GPU system.

The algorithm is capable to detect LEO objects down to magnitude +11, and the output is a plain table where all streaks detected, photometric and astrometric parameters are listed.

A set of 21084 FITS files (approximately 2 hours of observing time) were analysed, yielding in the detection of 85 streaks, which corresponded to 35 LEO objects, and among them, 30 were catalogued in the space-track.org database. The algorithm showed an improvement in speed of 5x against its CPU counterpart.

1 INTRODUCTION

The Tomo-e camera attached to the 1.05-m Schmidt telescope at Kiso Observatory in Japan, includes 84 CMOS image sensors of 2000 x 1128 pixels each, covering a field of view of 20 sq-degrees. This telescope performs several whole-sky surveys during each night, generating approximately 1.7 TB of data per hour. When surveying the sky, the telescope is in sidereal rate motion, therefore objects orbiting the Earth will show in the camera images as streaks of different lengths and brightness, mainly depending on the slant range to the object , their size, and the exposure time in the image sensors. The aim of this work was to detect streaks in the massive amount of data generated, in a fast and effective way, and also perform astrometric and photometric measurements of them, and identify the detected streaks through matching with already catalogued objects. To process and perform pixel operations in parallel in large sets of images, GPU platforms have proved to speed up operations when compared to its CPU counterparts. For this reason, GPU was chosen as the hardware platform to develop the automatic streaks detection algorithm.

There are several strategies to detect streaks in astronomical images, among them: source detection, computer vision, machine learning or template fitting [1]. In the present work we followed the computer vision approach based in pixel and pixels-region level operations, well suited for GPU processing. It's interesting to note that currently there is not perfect method to detect streaks in astronomical images, but there are algorithms that work better than others for particular telescope-camera combinations. What matters here is to understand the limitations of the proposed method and also define the scope of human intervention on it.

2 HARDWARE

2.1 Tomo-e Gozen Camera data

Below is a list of the parameters of the images obtained with the Tomo-e Gozen Camera [2]:

- Exposure time: 0.5 seconds.
- 18 consecutive frames per FITS file pointing for a particular area of the sky.
- Sensor size: 35 mm full HD.
- Field of View: 20 sq-degrees.
- Pixel size: 19 μm/pixel.
- Pixel scale: 1.198 arcsec/pixel.
- Sensor type: front illuminate CMOS image sensor.
- Sensor readout: rolling type, with partial readout available.

The produced FITS files consist of 18 consecutive frames of 0.5 seconds exposure time each, with a size of 2000 x 1128 pixels. Each FITS file consist of 40.6 Mpixels (162 Mbytes), and each full frame (84 image sensors consists of 3.4 Gpixels (13.6 Gbytes). The data rate is around 1.7 Tbytes per hour, considering an average nudging time between different areas of the sky of 20 seconds.

2.2 GPU system

The GPU system consists of four Quadro RTX8000 NVIDIA cards connected through PCI Express interface with an Intel Core i9-10940X 3.30 GHz CPU, and 128 GB DDR4 RAM memory. Each GPU card includes 72 multiprocessors and 48 GB of video memory.

3 PROCESS

The algorithm consists in two stages that run in parallel: detection and analysis. In the detection stage we identify the different streaks in the images and in the analysis stage we perform several astrometric and photometric measurements over these detected streaks.

3.1 Detection

In Fig. 1 are shown the steps of the detection stage. Each of 3D (2000 x 1128 x 18 pixels) FITS files are loaded in memory as 3-dimensional arrays. These arrays are conveniently flatten to 2D arrays in order to be processed in parallel by the OpenCV CUDA image processing libraries.

Previously a preprocessing algorithm consisting in the removal of stars is performed [3]. This way, diffraction spikes created by very bright stars won't create false positives in the algorithm.

The detection is based in the probabilistic Hough transform, which is used typically in computer vision to detect lines in images. The Hough Transform [4], developed and patented in 1962, is probably the more common mathematical method to detect morphological features in images through computer vision techniques. It has been the basis of many publications about the topic, where consecutive improvements over the original method were proposed. Currently also this method is reliably implemented in standard software packages of computer vision, such as OpenCV [5] or Scikit-image [6], which facilitates the use of this algorithm implementation in the image processing chain of any detection of features algorithm.

In the Hough transform parameters, we select high resolution and minimum threshold, and this yields in an excess of detected lines, therefore several lines will be detected per streak. To identify the actual streak for each group of lines, we join these lines in clusters through a series of consecutive clustering algorithms: by frame, by slope and finally by collinearity. The clustering by frame consists on associating each detected line to a particular subframe of the 18 frame fits file, based on its coordinates. The clustering by inclination basically discerns between the groups of detected lines based on the calculated inclination obtained through its coordinates. Finally, for the collinearity clustering, we obtain the areas formed by 3 coordinates: each of the ends of a detected line and the 2

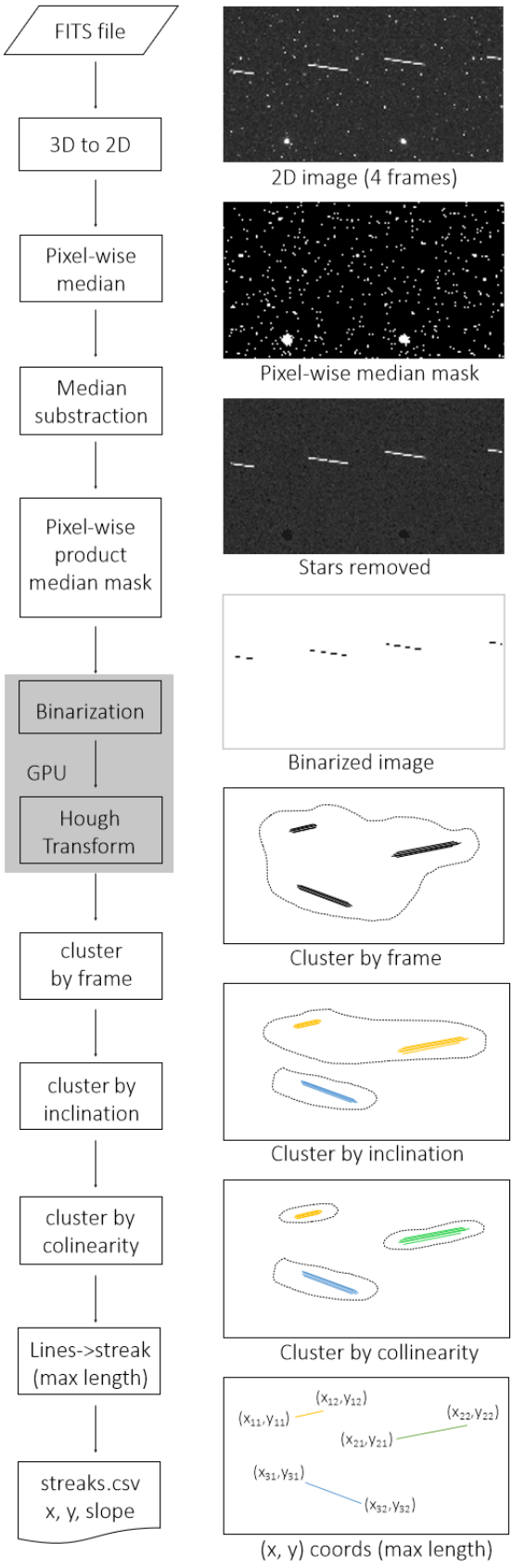

Figure 1. Steps of detection stage.

ends of the rest of each detected lines. If this area (in pixels) is below certain threshold, we state that these lines belongs to the same streak. Otherwise we assume that they belong to different streaks. After this, we select the outermost extreme coordinates of each cluster of lines to be selected as the coordinates of the final streaks.

Binarization and Hough Transform steps are implemented in GPUs, in order to increase processing speed, as they are the more pixel-level intensive operations.

3.2 Analysis

In Fig. 2 are shown the different steps of the analysis stage.

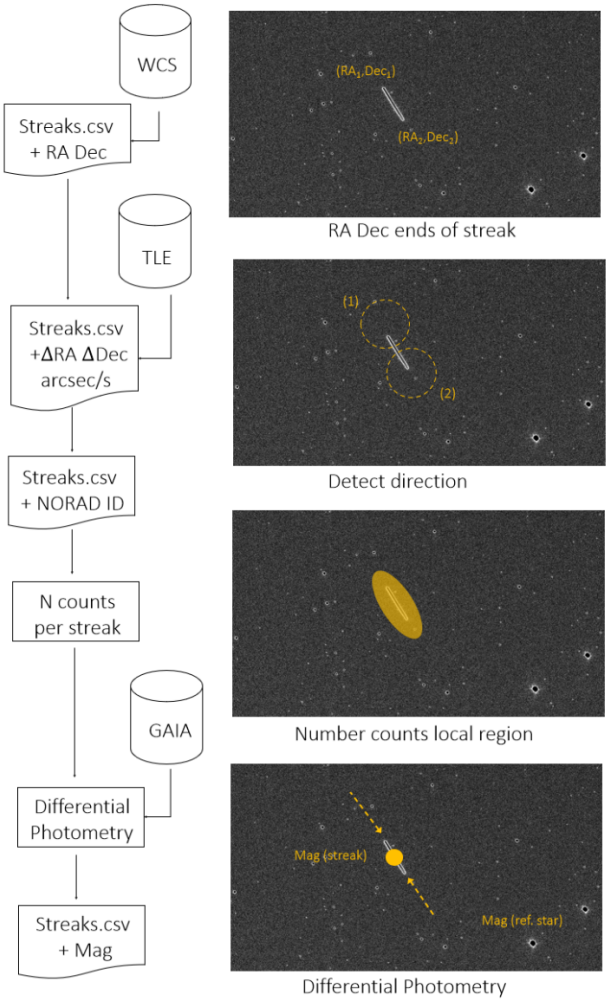

Figure 2. Steps of analysis stage.

In this stage firstly we obtain the RA-Dec coordinates of the streak ends with the help of the WCSTools library [7] and the WCS coordinates information already contained in the FITS files produced in the camera. The algorithm compare the RA and Dec coordinates with the ones of a database of orbital elements downloaded from space-track.org for the particular observation date, and we obtain the more probable candidate for this streak.

To detect the object direction, we compare the coordinates of the streak ends in the consecutive and previous frames. For this, we define a circular area around each end, and explore if a streak exists in the previous and consecutive frames. After this step we obtain the sign of the increments in RA and Dec coordinates, so we can compare with the proposed catalogued candidates.

In the last step we perform differential photometry with surrounding stars identified in the GAIA ED3 catalogue. For this, we first obtain the number of counts of the streak in an area around it. From there we calculate the instrumental magnitude, and then the apparent magnitude, indicating the residual error which give us an estimation of the accuracy of the obtained magnitude value.

4 RESULTS

In Fig. 3 is shown an example of a detected streak, identified as SL-8 R/B rocket body (NORAD number 16766). In this figure we can appreciate that the stars (in white) have been removed by the mask (in black), so they don't affect the detection stage of the algorithm. Also it can be noted the lines detected (in black) in the Hough Transform step, superimposed over the streak (in white).

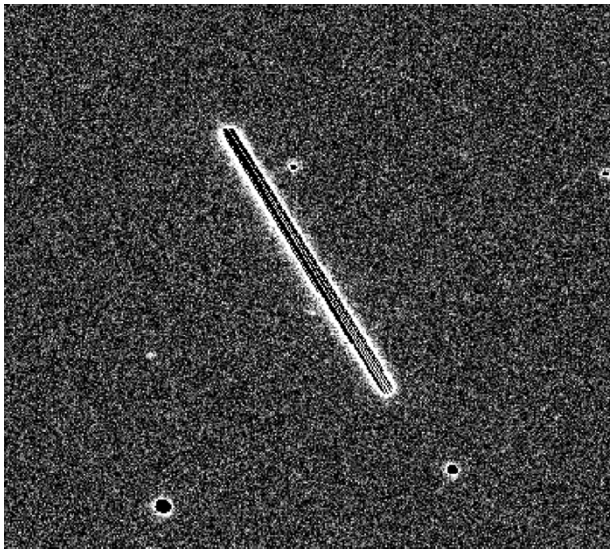

Figure 3. Streak created by SL-8 R/B (NORAD 16766).

The output of the algorithm is a table where each detected streak is listed together with the information obtained through the analysis stage. In Tab. 1 is shown an extract of this table as a reference.

21084 FITS files were analysed (2 hours observation time), and there were detected 85 files with streaks on them. These 85 streaks corresponded to 35 objects (as

Table 1. Extract of output table. "length" is in pixels. "slope" is in degrees. "sf" represents the subframe number of the same FITS file. "RA rate" and "Dec rate" are in arcsec/s. "Mag" is the apparent magnitude. "res" is the residual error in the calculation of the apparent magnitude.

file ID	length	slope(°)	sf	UTC_1	UTC_2	TLE_candidate	NORAD	RA rate	Dec rate	Mag	res.
rTMQ120201018003802472 5.fits	264	-39	1	22:55.4	22:55.9	ORBCOMM FM 36	25984	-491.2	397.7	8	0.02
rTMQ120201018003802472 5.fits	264	-39.2	0	22:54.9	22:55.4	ORBCOMM FM 36	25984	-491.2	400.1	8	0.02
rTMQ120201018003802471 6.fits	273	-39.1	9	22:59.4	22:59.9	ORBCOMM FM 36	25984	508	-412.1	8.2	0.01
rTMQ120201018003802071 1.fits	520	-62	15	07:52.2	07:52.7	STARLINK-1336	45557	-584.6	1099.8	4.7	1.29
rTMQ120201018003802071 1.fits	518	-60	14	07:51.7	07:52.2	STARLINK-1336	45557	-620.6	1073.4	4.7	1.29
rTMQ120201018003801894 4.fits	376	85	15	00:57.0	00:57.5	COSMOS 1181	11803	79.1	896.1	7	1.06
rTMQ120201018003801894 4.fits	392	81	14	00:56.5	00:57.0	COSMOS 1181	11803	146.2	927.3	6.9	1.06
rTMQ120201018003801894 3.fits	390	83.1	10	00:54.5	00:55.0	COSMOS 1181	11803	-112.6	-927.3	10.6	8.73
rTMQ120201018003801894 3.fits	393	80	11	00:55.0	00:55.5	COSMOS 1181	11803	-162.9	-927.3	10.6	8.73
rTMQ120201018003801894 2.fits	392	83	6	00:52.5	00:53.0	COSMOS 1181	11803	115	932	9.8	5.93
rTMQ120201018003801894 1.fits	394	81.1	2	00:50.5	00:51.0	COSMOS 1181	11803	146.2	932	6.3	0.87
rTMQ120201018003801894 1.fits	392	83.1	1	00:50.0	00:50.5	COSMOS 1181	11803	112.6	932	6.2	0.87
rTMQ120201018003801482 6.fits	203	8.8	9	42:23.1	42:23.6	TITAN 3A DEB	1000	-481.6	-74.3	7.5	0.32
rTMQ120201018003801482 6.fits	204	13	10	42:23.6	42:24.1	TITAN 3A DEB	1000	-476.8	-110.2	7.5	0.32

the same object was detected as several streaks in one particular FITS file). After the identification step, 30 of these 35 objects were matched with catalogued LEO objects in the space-track database.

The detection stage takes approximately 100 ms to process each FITS file, and the analysis stage takes between 8 to 15 seconds per FITS file. Detection and analysis stage are run in parallel. Streaks were detected in approximately 0.5% of the 21084 FITS images analysed. Therefore the amount of data to process by the analysis stage is 0.5% lower than the detection stage. In the worst case scenario (15 seconds to analyse a FITS file), overall the completion of the analysis stage will take less time that the detection stage on average.

The total time to process one batch of 84 CMOS image sensors frames of 2000 x 1128 x 18 pixels each, that is 1 full Tomo-e frame, is lower than the time elapsed until the next frame in another region of the sky is initiated (around 20 seconds), which means that we can detect streaks in real-time, at the same speed or better than the FITS files are produced.

The minimum detectable length of the streak is 200 pixels, which translates to objects located around 1900 km height above the Earth surface (for an elevation range of 30 degrees), which means that we can detect objects located in the upper LEO orbit.

In terms of signal-to-noise ratio of detected streaks, if we normalized the background noise to 1σ, the weaker streak detected had a value of 1.06σ. After performing the differential photometry, this faintest apparent magnitude detectable is +11 in the GAIA G-band.

5 CONCLUSIONS

We have developed a fast algorithm implemented in GPU to detect streaks in astronomical images in real time, which is particularly useful when a massive amount of data is presented, typically from survey telescopes. We have particularized this algorithm for the data coming from the Tomo-e Gozen camera of the 1M telescope at Kiso Observatory.

Our next goals towards the improvement of the current algorithm, is increase the sensitivity of the detected objects, that currently has its limiting magnitude in +11.

6 ACKNOWLEDGEMENTS

This research was funded by JSPS (Japan Society of Promotion of Science) and JAXA (Japan Aerospace Exploration Agency), and the FITS file were provided by the Tomo-e Gozen Team of University of Tokyo.

7 REFERENCES

1. Nir. G. (2018). Optimal and Efficient Streak detection in Astronomical Images. *The Astronomical Journa*l, 156:229.
2. Sako S., Ohsawa R., Hidenori T., Kojima Y., Doi M. et al (2018). The Tomo-e Gozen wide field CMOS camera for the Kiso Schmidt telescope. *SPIE Astronomical Telescopes conf. USA.*
3. Yanagisawa T., Nakajima A. (2005). Automatic Detection Algorithm for Small Moving Objects. *Publications of the Astronomical Society of Japan*, 57, 399-408.
4. Hough, P.V.C. (1962). Method and means for recognising complex patterns. U.S. Patent 3,069,654.
5. OpenCV Open Source Computer Vision. https://docs.opencv.org/3.4/index.html.
6. Scikit-image Image Procession in Python. https://scikit-image.org.
7. WCSTools: Image World Coordinate System Utilities http://tdc-www.harvard.edu/wcstools/.