FreeTure
A Free software to capTure meteors for FRIPON

Yoan Audureau[1], Chiara Marmo[1], Sylvain Bouley[1], Min-Kyung Kwon[2], François Colas[2], Jérémie Vaubaillon[2], Mirel Birlan[2], Brigitte Zanda[3], Pierre Vernazza[4], Stephane Caminade[5], Jérôme Gattecceca[6]

[1] **Université Paris-Sud, Laboratoire GEOPS, UMR8148, Orsay, F-91405, France**
yoan.audureau@u-psud.fr, chiara.marmo@u-psud.fr

[2] **IMCCE, Observatoire de Paris, France**

[3] **MNHN, Paris, France**

[4] **LAM, Université Aix Marseille, Marseille, France**

[5] **Université Paris Sud, IAS, UMR8617, Orsay, France**

[6] **Université Aix Marseille, CEREGE, Aix-en-Provence, France**

The Fireball Recovery and Interplanetary Observation Network (FRIPON) is a French project started in 2014 which will monitor the sky, using 100 all-sky cameras to detect meteors and to retrieve related meteorites on the ground. There are several detection software all around. Some of them are proprietary. Also, some of them are hardware dependent. We present here the open source software for meteor detection to be installed on the FRIPON network's stations. The software will run on Linux with gigabit Ethernet cameras and we plan to make it cross platform. This paper is focused on the meteor detection method used for the pipeline development and the present capabilities.

1 Introduction

The French FRIPON project aims to detect fireballs and to retrieve related meteorites on the ground. It also aims to detect standard meteors and to build a database of computed orbits to find related parent bodies. To do that, more than 100 stations will cover the complete surface of France. Each one will be equipped with an all-sky GigE camera and a computer. On each local computer, a software will be used to control the camera and to detect meteor events. A lot of software exist to do that, but some are proprietary (e.g. UFOCapture[1]) and the others are hardware dependent or not cross platform (Molau, 98). FRIPON needs a new meteor detection software because of its number of stations and it should have the possibility to easily modify or to develop some features. With a free, open source and a cross platform software, FRIPON can be easily extended by installations on amateur or professional stations. Thus, the public could start contributing to the project by sharing their information about a detected event or by adding new features to the software for maintaining or improving it. Anyone could for example add the support of a new camera, add new detection algorithms or build a GUI.

2 Initial features

The software is developed in *C++* using *Boost*[2] and *OpenCV*[3] libraries to easily make it cross platform on Linux and Windows operating systems. The following features were required:

- Continuous real time meteor detection, day and night.
- Input frames grabbed from a GigE camera or from a pre-recorded video.
- Output files in time sequences or stacked frames.
- Output *FITS* files (among others) without any destructive compression for scientific analysis.
- Open Source.

Table 1 – Meteor detection software.

	Open source	Platform	Output
UFO Capture[1]	no	windows	.csv, xml, avi, jpg. …
MetRec (Molau 98)	yes	Msdos, w95, w98	.bmp …
ASGARD[4]	no	Linux (debian)	.tar (.png), .txt, avi …
MeteorScan (Gural96)	no	Mac, windows	.tiff …
FRIPON (fripon.org)	yes	Linux, windows	.avi,, jpg, fits 2D, fits 3D

[1] UFOCapture, http://sonotaco.com/e_index.html.
[2] Boost: http://www.boost.org/
[3] OpenCV: http://opencv.org/

3 General software structure

We describe here the general layout of the software. Four parallel processes run at the same time: Acquisition, Stack acquisition, Detection and Recording. Their parameters may be set in a configuration file included in the package. The first thread is used to manage acquisition from a GigE camera, video or images. Under Linux, an open source library named *Aravis*[4] is used to control the camera. Under Windows, constructor's libraries are used. The acquisition thread grabs a frame and stores it in a shared buffer. Its size depends of how many frames we want to record for a detected event and it determines the memory footprint of the process. The buffer is shared with another thread used to stack frames and also with the detection thread. The last frame stored in the shared buffer and the previous one are both used by the detection process. Detected events have their own buffer shared with a recording thread to save events on the hard disk in a different file format. *Figure 1* summarizes the general structure of the program.

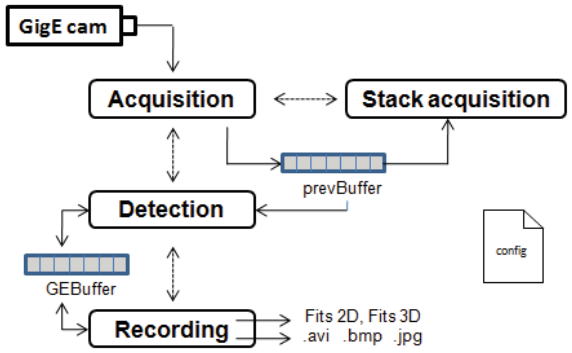

Figure 1 – Software's outline.

4 Detection method

The algorithm used for the meteor detection is quite simple and is more or less based on the detection method used by the ASGARD software (Weryk, 2013). The detection thread receives a notification to indicate that a new frame has been grabbed. The detection process starts to operate two main steps to locate probable events on the current frame and two others to try to build them in time.

The first step starts to filter the current frame to select some pixels which will be used to feed the detection process. To do that, two successive frames are subtracted for removing stationary features and a threshold is defined.

The second step aims to build a list of local event objects. A local event (LE) refers to a group of regions of interest (ROI) which intersect each other. If a pixel exceeds the threshold value, a region of interest (ROI) of 10 x10 pixels around its location is defined and extracted. The ROI is kept if there are more than *n* pixels inside which also exceed the threshold value. In that case, the region of interest is colored in black in the frame to avoid treating the same event many times. The extracted region is compared to the element of the list of local events. If it intersects a ROI of an existing LE, it is added in the same LE. Otherwise a new LE is created and the ROI is added in it. To quickly know if a new ROI intersects an existing ROI in a LE, a colored map is produced. Each LE has its own RGB color on this map, which only exists for the current frame. At the position of the new ROI, the color is extracted. If there is at least one pixel inside with another color than black, this ROI is linked to the LE which has the same color in the LE list.

Once the list of local events is known, the detection on the current frame is done. But the local events must still be linked with global event objects. A global event is a group of local events from different frames. It always exists, contrary to the local event list which only exists during a frame analysis. It is used to link local events which intersect each other not in space but in time. That means that they belong to the same event. If a new local event intersects none of the last local events of a global event, it is added into a new global event. We also check during the link operation if the new potential local event location seems to follow the general global event moving direction. This is done to check if the construction of a global event moves in one direction in function of time.

Finally, the existence of global events stored in a list is managed. Each global event stores the number of frames passed since its creation and the number of frames without any new local event has been linked. A limit which can be defined for an event duration is also used to avoid to record too long events like planes. With these three sets of information, a detected event which is stored in the computer memory is finally saved on the hard disk or removed.

5 Running the software

The software is quite simple to run. There is just need to write a command line with the name of the program to be written together with some arguments according to the chosen mode. Currently, there are three modes.

1) List detected GigE cameras
2) Make a single capture
3) Run detection

The first one is used to list detected GigE cameras with some other information about the devices. The second allows to test a camera by making a single capture and by setting some options like the exposure time, the gain and the acquisition format. Finally, the last mode is the detection mode to start to detect meteors. For the second mode, parameters are directly given as arguments in the command line. For the third mode, parameters may be set in a configuration file.

[4] ARAVIS: https://wiki.gnome.org/Projects/Aravis

6 Results

With our detection software, some first bad and good results have been recorded, like the following plane (*Figure 2*) and meteors (*Figure 3 and 4*).

As planes are longer events than meteors, their recording can be avoided by a definable limit for an event duration.

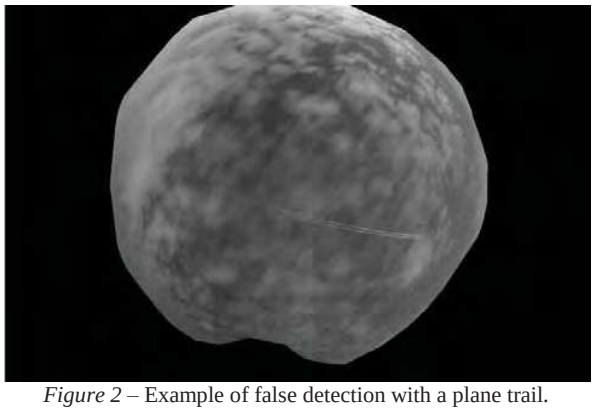

Figure 2 – Example of false detection with a plane trail.

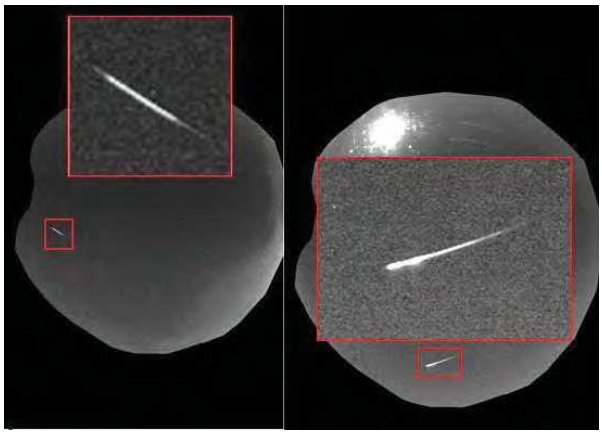

Figure 3 – Meteor examples.

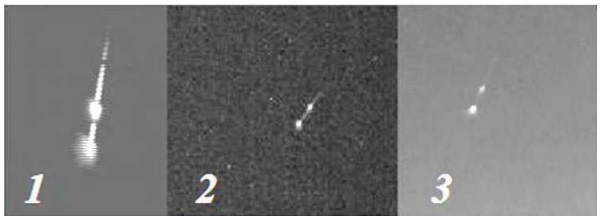

Figure 4 – The same meteor detected by three stations. 1) Stephan Jouin with UFO Capture, 2) Orsay with our software, 3) Observatoire de Paris by regular captures with long exposure.

7 Conclusion and future work

Currently, the main features required are operational and the software can run every night to start detecting meteors events. In the future, we plan to add an algorithm for the daytime detection and to make some comparisons with other meteor detection software to check the efficiency of ours. Finally the Windows version of the software still needs to be packaged.

References

Molau S. (1998). "The Meteor Detection Software MetRec". In, Baggaley W. J., Porubčan V., editors, *Proceedings of the International Conference Meteoroids 1998*, Tatranska Lomnica, Slovakia, August 17–21, 1998. Pages 131–134.

Gural. P. (1997). "An operational autonomous meteor detector". *WGN, Journal of the IMO*, **25**, 136–140.

Weryk R. J., Campbell-Brown M. D., Wiegert P. A., Brown P. G., Krzeminski Z., and Musci R. (2013). "The Canadian Automated Meteor Observatory (CAMO): System overview". *Icarus*, **225**, 614–622.