



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

UNIVERSITAT POLITÈCNICA DE CATALUNYA
ESCOLA SUPERIOR D'ENGINYERIA INDUSTRIAL, AEROESPACIAL I
AUDIOVISUAL DE TERRASSA
ENGINEERING IN AEROSPACE TECHNOLOGIES

Evaluation of the Lightning Imaging Sensor aboard the International Space Station using Lightning Mapping Array

BACHELOR THESIS
– REPORT –

Author:
Ícar FONTCUBERTA

Supervisor:
Joan MONTANYÀ

Delivered: 10 June 2019

ABSTRACT

In this work the data from a Lightning Mapping Array (LMA Ebro Delta) and the Lightning Imaging Sensor (ISS-LIS) has been used with the aim to explore which properties of the former make it more likely to have simultaneous ISS-LIS detections associated. To do so, the data has been divided in 10ms time bins; the ISS-LIS and LMA data points inside a same time bin have been associated as a sole emission. Then, the properties of each sensors' detections have been transferred to the bins, to which have been assigned the properties of both kinds of detections. The analysis has shown a clear relation between the amount of LMA data points in a bin and the likelihood of that bin to have a simultaneous LIS detection. This relation is probably due to the sum of powers of LMA data points; the study has not shown any clear relation between the maximum power recorded in a bin with its simultaneous ISS-LIS detection. Other parameters such as the height of LMA detections has been taken into account; which has shown no clear influence on generating orbit detectable emissions. In this work is also presented an introductory technical report on ISS-LIS and its current data products, as well as the way how they have been treated for acquiring the mentioned results. Plus, it has been performed a literature research with the aim to find a relation between the voltage applied to the open air in order to generate an electrical arc and its emissions on the 777.4 nm band; research that has shown unsatisfactory results.

Contents

1	Introduction	8
2	Description of Sensors	10
2.1	INTERNATIONAL SPACE STATION - LIGHTNING IMAGING SENSOR	10
2.1.1	International Space Station - LIS	11
2.1.2	LIS: instrument description	12
2.1.3	Event Grouping Process	14
2.1.4	Technical properties	16
2.2	LIGHTNING MAPPING ARRAY	17
2.2.1	Principle of Operation	17
2.2.2	Ebre Lightning Mapping Array	18
3	Sensors' Data Handling	19
3.1	ISS-LIS HIERARCHICAL DATA FORMAT FILES	19
3.1.1	LIS_ HDF_ processor.m	20
3.1.2	LIS_ vs_ LMA_ comparator.m	24
3.2	LMA DATA	25
3.2.1	LMA_ zoom7.sci	25
4	LIS data analysis using LMA as reference	26
4.1	OVERVIEW OF GATHERED DATA	26
4.1.1	Space-Time distributions of detections	26
4.1.2	Influence of excited pixels' position on the CCD	37
4.1.3	Section summary	38
4.2	INFLUENCE OF VHF SOURCES' PROPERTIES ON ITS LIS DETECTIVITY . . .	39
4.2.1	Hypothesis and analysis description	39
4.2.2	Sources' Height Influence on Detectivity	40
4.2.3	Sources' Maximum Power influence on Detectivity	45
4.2.4	Density of sources in the time bins	47
4.2.5	Section Summary	50
4.3	ANALYSIS OF THE FLASH DURATION CONCORDANCE BETWEEN LIS AND LMA SENSORS	51
4.3.1	Flash Duration Analysis	51
4.3.2	Section summary	54
5	Electric arc emissions for LIS calibration	55
5.1	Minimum Radiance Emissions	55
5.2	Relation peak voltage – radiance emitted	58
5.2.1	Direct relation	58
5.2.2	Other approaches	59
6	Conclusion and further work	62

Appendix A APPENDIX	64
A.1 Geostationary Lightning Mapper description	64
A.2 Night vs. Day distribution of LIS during 2017 period	65
A.3 Detections' properties influence on LIS detectivity from a typical value approach . . .	66
A.3.1 Sources density	66
A.3.2 Sources' height	68
A.3.3 Sources' Power	70
A.3.4 Section summary	74
A.4 Extra Figures	74
A.4.1 Mean sources' height	74
A.4.2 Histogram distributions from a data-point point of view	76
A.5 User Guide for data the processing codes	77
A.6 Codes for data processing	89
A.6.1 LIS_HDF_processor.m	89
A.6.2 LIS_vs_LMA_comparator.m	113

ACRONYMS AND ABBREVIATIONS

In alphabetical order

ASIM Atmosphere-space Interaction Monitor

CCD Charged-Coupled Device

CG Cloud to ground (flash)

EM Electromagnetic

ESEIAAT Escolta Superior d'Enginyeria Industrial, Aeroespacial i Audiovisual de Terrassa

FOV Field of View

HDF Hierarchical Data Format

IC Intra Cloud (flash)

IFOV Instantaneous Field of View

ISS International Space Station

LAT Latitude

LON Longitude

LMA Lightning Mapping Array

LINET Lightning detection Network

LIS Lightning Imaging Sensor

LRG Lightning Research Group

OTD Optical Transient Detector

oe only sources

pwr Power

RF Radio-frequency

se sources + events

SNR Signal-to-Noise Ratio

TLE Transient Luminous Event

TOA Time of Arrival
TRMM Tropical Rainfall Measuring Mission
txt Text
UTC Universal Coordinated Time
VHF Very High Frequency

List of Figures

2.1	Measuring range of OTD and TRMM imaging sensors. Source: [2]	11
2.2	ISS-LIS observable area. Source: [2]	12
2.3	Lightning Spectra Information. Source: [6]	14
2.4	Example of grouping process. Source: [8].	15
2.5	3-D models of the ISS-LIS sensor. Source: [7]	16
2.6	Lightning Mapping System antennae locations. Source: [11]	17
2.7	Position of LMA antennae in Ebre delta (white circles). Source: [13].	18
3.1	ISS-LIS HDF files organisation	20
3.2	Screenshot of a output LIS txt file	23
4.1	Space and time distribution of LIS and LMA detections during 17-10-18 10:30.	27
4.2	Height[m] vs. time [sec. from start of 10 min. period]. Zoom on time distribution of LIS and LMA detections during 17-10-18 10:30.	28
4.3	Space and time distribution of LIS and LMA detections during 17-10 18 17:00. Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.] Bottom right: height[m] vs. LAT/LON [deg.]. Sources are dots, events are circles. Sources coloured by time, events by group.	28
4.4	Space and time distribution of LIS and LMA detections during 17-10 18 17:00. Focus on LIS view time.	29
4.5	Sources and Events detections printed on height and time	30
4.6	Space and time distribution of LIS and LMA detections during 18-04-27 13:10.	31
4.7	Space and time distribution of LIS and LMA detections during 18-04-29 11:30 view time. Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.] The smaller picture displays the whole 10 min period.	32
4.8	Space and time distribution of LIS and LMA detections during 18-05-25 01:40 view time.	33
4.9	Space and time distribution of LIS and LMA detections during 18-06-05 15:10 view time.	33
4.10	Space and time distribution of LIS and LMA detections during 18-06-06 14:20 view time.	34
4.11	Space and time distribution of LIS and LMA detections during 18-06-13 17:50 view time.	35
4.12	Height and time distribution of LIS and LMA detections during 2018-08-09 18:50 view time.	35
4.13	Height and time distribution of LIS and LMA detections during 2018-08-09 04:40 view time.	36
4.14	Space and time distribution of LIS and LMA detections during 2018-09-18 03:30 view time.	36
4.15	Space and time distribution of LIS and LMA detections during 2018-10-18 15:10 view time.	37
4.16	Excited pixels distribution on the CCD for bad data cases.	37
4.17	Excited pixels distribution on the CCD for two cases with good data: 17-10-18	38

4.18	Mean height histograms for the cases of the 2017 fall	41
4.19	Median of Sources Heights Histograms for all periods of data	42
4.20	Example for symmetrical distribution of power	43
4.21	Power-weighted average sources' height	44
4.22	Maximum power histograms for all periods of data	46
4.23	Distribution of Density of sources in a bin for all periods	48
4.24	Distributions of the sum of power for all periods	49
4.25	Sources (+) and events (x) detected during the 171018 1030 time period printed over time, coloured by flash.	52
4.26	Zoom on a flash of the 171018 1030 time period. Sources (+) and events (x) printed over time.	52
5.1	MODTRAN output Flux and Radiance	57
5.2	Atmospheric transmittance depending on the wavelength by MODTRAN	57
5.3	Emission properties of controlled electric arc discharges. Extracted from [19]	58
5.4	Emission properties of controlled electric arc discharges (second). Extracted from [19]	59
5.5	Emission intensity of different spectral lines with the plasma temperature. Source: [21]	60
5.6	Intensity of emissions depending on the wavelength. Source: [23]	61
A.1	GEOS satellite and its FOV on the Earth surface. Source: [10]	65
A.2	GLM sensor performance properties. Source: [10]	65
A.3	Night-Day presence of lightning detected by LIS around Deltebre area from March 2017 to July 2018.	66
A.4	Typical values of sources densities in bins for each 10 min period.	67
A.5	Typical values of sources' height in each 10 min. time period, achieved from the typical height values inside each bin of the same period.	68
A.6	Space-time detections distribution of 2017-10-18 with LINET included (X and +). "x" are negative strokes and "+" are positive strokes. Circles are events (coloured by flash) and dots are events (coloured by time).	69
A.7	Sum of powers for each bin	70
A.8	Typical values for power-weighted centroid positions in each time period.	71
A.9	Maximum power for each bin during 171018 1030	72
A.10	Typical values of maximum power for each inside-FOV time period	72
A.11	Data-point POV histograms for 171018 10:30	76
A.12	Data-point POV histograms for 17-10-18 17:00	76
A.13	Data-point POV histograms for 18-08-09	76
A.14	Data-point POV histograms for 18-08-31	76
A.15	Data-point POV histograms for 18-09-18	77
A.16	Data-point POV histograms for 18-10-18	77

List of Tables

2.1	ISS-LIS sensor technical properties. The frame rate is of 2ms and the total FOV is of 600 x 600 km. Source: [9]	16
4.1	Duration of LMA and LIS flashes (seconds). In the table there are not the flashes that were not detected by LIS. The rows with zeroes represent those flashes that LIS detected but not LMA.	53

5.1 Typical radiance values of events with sources associated at 2500m, with a tolerance of
50m in $[\frac{\mu J}{sr^2 m^2 \mu m}]$ 56

Chapter 1

Introduction

Motivation

Lightning are rather puzzling phenomena: beams of light that fall from the sky, announcing the arrival of a sound blast that will frighten men and animals for equal. It is not surprising that they have sparkled mythologies, stories and the human mind an imagination from millennia. Through observation and reasoning we have mostly unraveled the physical mechanisms that generate such beautiful elements; yet their measurement and observation has been far from losing interest.

In the modern times lightning have gained a huge interest as indicators of thunderstorms: they are wonderful electromagnetic signal generators. Their print is widely used to generate thunderstorm maps by weather-forecasting institutions all over the globe. Scientists and engineers have pushed themselves along the years to design a number of systems that can map the presence of lightning. One of those are the Lightning Mapping Arrays (LMA), systems of antennae that can detect lightning in a radius of the order of 100km. Despite the LMA being extremely useful to generate local lightning maps, since the 90s some space agencies such as NASA have put their focus on orbit-based systems, which have an extremely wider range of detection. These systems, although not having the same precision or being as effective as the LMAs, have proven their utility by allowing the scientific institutions study thunderstorm populations where other sensors could not "dream" to reach. Monitorin the oceanic storms, for instance, is not only mandatory for understanding the weather in situ, which has a gigantic economic impact in maritime trade, but is also important for being able to prescience the weather else were.

That being said, the success and utility of orbit-based lightning detectors is currently circumscribed by their relatively short time of experience. Validation and calibration are major operations that each one of these sensors has to face and can be done, for instance, from a cross-sensor approach. The LMA systems gain here a new role: they can provide very accurate local data to less accurate global detectors. This approach is relevant since the systems are already installed and the technology is well known. The only perk, nonetheless, is that orbit-based systems, being the Lightning Imaging System at the International Space Station (ISS-LIS) the most promising at the moment, don't read the same signal: while LMA captures VHF RF pulses (at the band of 60MHz approx.), the ISS-LIS detects the light pulses of much greater frequency (wavelength of about 777 nm).

In order to be able to compare the data, for example with the aim of calibrating a LIS sensor, it is mandatory to understand if both detections can be associated and in which way. Assuming that a lightning emits both types of signals, the first step could be to explore which properties of the LMA detections have an impact on the likelihood of generating a simultaneous detection from LIS.

Finally, as commented above, the newness of orbit-based systems, and in particular ISS-LIS, leads to a lack of information about the systems' properties and data products. Moreover, there is little knowledge and analysis on data crossing from LIS-LMA systems.

Aim and Scope

Accordingly at what has been said above, this work aims to provide:

1. A COMPREHENSIVE REPORT ABOUT THE TECHNICAL PROPERTIES AND THE FUNDAMENTAL STRUCTURE OF THE LIS AND LMA SYSTEMS
2. A DESCRIPTION ABOUT THE ISS-LIS DATA PRODUCTS AND HOW THEY HAVE BEEN PROCESSED IN THIS WORK
3. A REPORT ON THE EXPLORATION OF THE INFLUENCE OF THE LMA SOURCES' PROPERTIES ON ITS LIKELIHOOD TO BE DETECTED WITH ISS-LIS AND OTHER CROSS-SENSOR ASSESSMENTS

Out of the scope will be left:

- An exhaustive technical description of the systems
- Data analysis with sensors other than ISS-LIS and LMA
- Production of statistical models for prediction of detection by ISS-LIS of LMA sources

In summary, the main objective of this work is to evaluate the detections of lightning from space by means of high-resolution Lightning Mapping Array systems (LMA); and as partial objectives will be done reviews of lightning and lightning imagers, definition of the evaluation, develop codes for reading data and develop codes for the evaluation.

Chapter 2

Description of Sensors

Contents

2.1	INTERNATIONAL SPACE STATION - LIGHTNING IMAGING SENSOR	10
2.1.1	International Space Station - LIS	11
2.1.2	LIS: instrument description	12
2.1.3	Event Grouping Process	14
2.1.4	Technical properties	16
2.2	LIGHTNING MAPPING ARRAY	17
2.2.1	Principle of Operation	17
2.2.2	Ebre Lightning Mapping Array	18

In this chapter the reader will find non-exhaustive descriptions of the sensors involved in the current lightning analysis. A complete technical description of the sensor is left out of scope of this chapter, and merely generic system descriptions and technical properties will be provided. The aim of this chapter is to give comprehensive compendium of the information recollected from more complete, technical sources, that will be indicated; this compendium should allow the reader to gain basic technical knowledge about the systems to which this work refers.

2.1 INTERNATIONAL SPACE STATION - LIGHTNING IMAGING SENSOR

The Lightning Imaging Sensor (LIS) is a sensor designed by NASA, the University of Alabama in Huntsville and their partners in the early nineties with Hugh Christian as principal investigator. The LIS is an orbital sensor that monitors Earth low-atmosphere and detects lightning via the observation of high luminosity events (measuring its radiance) during thunderstorms.

Historical Background¹

The first orbital sensor that detected lightning was the Optical Transient Detector (OTD), launched by NASA in 1995, with the objective of provide a better understanding of thunderstorms distribution through a wide tropical strip. The OTD could detect intra-cloud (IC) and cloud-to-ground (CG) lightning via imaging process, with a spatial resolution of 10 km and a detection efficiency² of 50%. From a 740km high and 70 °orbit, it had a footprint (projected area where it could measure) of 1300 x 1300 km.

In order to gain spatial resolution and to be able to compare the imaging detection method with other types of measurements (X-ray, gamma ray, RADAR...) the LIS sensor was designed. LIS would be a sensor that would scan for luminous events during thunderstorms from orbit and its imaging method

¹Most information about the historical background has been extracted from [1].

²Understood as the percentile of lightning that sensor detects in a given are to the total amount, estimated with other measurements.

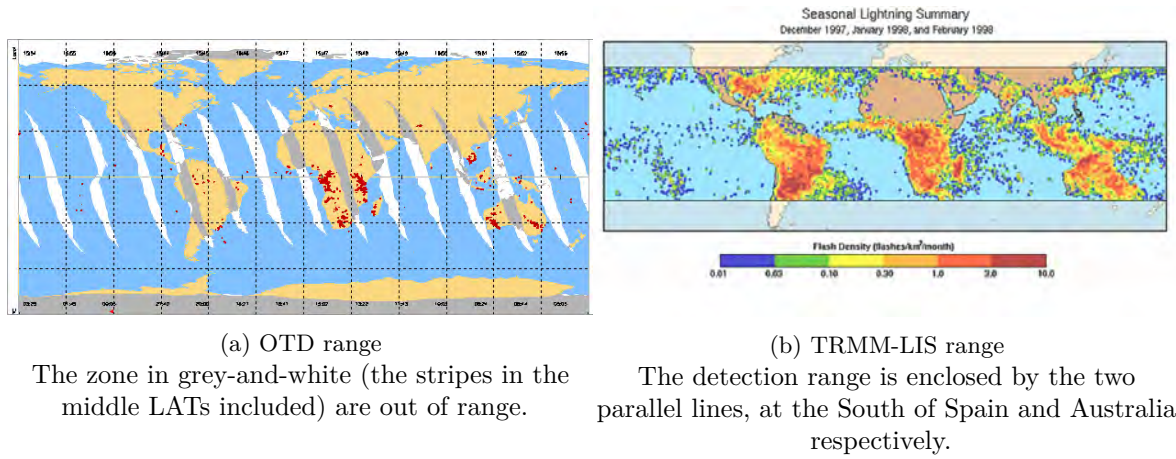


Figure 2.1: Measuring range of OTD and TRMM imaging sensors. Source: [2]

would be compared to data gathered from other instruments that would scan the same phenomena. In 1997 the Tropical Rainfall Measuring Mission (TRMM) was launched. Its satellite orbited the Earth at 350 km and with 35 °of inclination, and contained a LIS sensor alongside a RADAR, a Infra-red scanner, a microwave sensor and other measuring devices. Despite that a lower orbit implied a reduction of the wideness (LAT direction) of the scanned region, it was estimated that it detected approximately the 90% of lightning that occur through out the Earth each year. This high efficiency detection while latitude range reduction was achieved because the vast majority of lightning happen near the tropics, so detection reduction near the poles will not affect the overall efficiency. The lower orbit also led to a better spatial resolution (up to 4 km). Nonetheless, also due to the same parameter the footprint was reduced to 600 x 600 km.

A visualization of both OTD and TRMM imaging sensors is displayed in the figure 2.1. It is possible to see how the higher orbit (OTD) and inclination satellite had a wider observable zone: the maximum latitudes of observation are on the north of Greenland, whereas the TRMM-LIS only could observe up to the south Mediterranean. In the recent years, more advanced satellite-based sensors have been developed, such as LIS or GLM. Below the reader will find a description of the former. Information about the latter is available in appendix A.1.

2.1.1 International Space Station - LIS

From March 2017 a new LIS sensor is operative. Integrated on the International Space Station (ISS), the scope of the ISS-LIS is to monitor the global lightning distribution and other parameters associated with thunderstorms and lightning, such as total precipitation volume; lightning and precipitation relation; relation between lightning and the type of cloud etc.

Some of the advantages of the ISS-LIS over the TRMM-LIS are:

- Higher latitude coverage. Represented in fig. 2.2.
- Substantial improvement in data availability and telemetry velocity (real-time data possible).
- Possibility of observation comparison with other sensors integrated on the ISS, such as: the Atmosphere-Space Interaction Monitor (ASIM) – which detects high energy emissions (X-ray and gamma ray, TLEs...); or the Global Lightning and Sprite Measurement Sensor, which detects Very High Frequency (VHF) emissions. Furthermore, the availability of other sensors makes possible a cross-sensor validation of the LIS.

Also, as the ISS orbits at roughly 400km (higher than the TRMM satellite) the observable area of the ISS-LIS sensor is wider than the one from TRMM, covering now the USA and most of Europe.

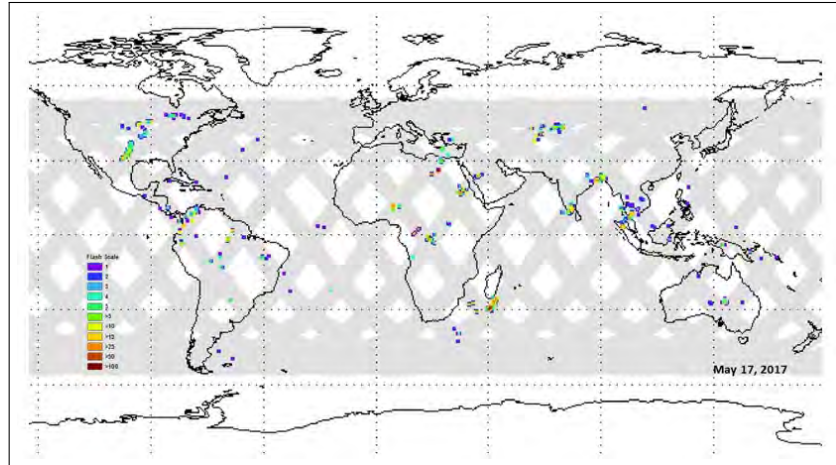


Figure 2.2: ISS-LIS observable area. Source: [2]

2.1.2 LIS: instrument description

The scope of this section is not to provide a technical description of the sensor; its detailed structure or system configuration. On the contrary, its mission is to give a general description on what are the important sub-systems that configure the LIS sensors, in order to gain knowledge that may be useful for future allusions when analysing its data. Most information of this section is extracted from [3].

A variety of characteristics of the LIS are described below. As they are not of the same nature (system point of view or requirement-to-description point of view) they will be split in both approaches.

System Approach

The LIS is a sensor that in many aspects resembles a digital camera, although modified to take shots of radiation outside the visible spectre. I.e. an input of radiation is set on a Charged- Coupled Device (CCD) that generates a map of triggered photo-diodes. Below are summarized the most important subsystems that configure the sensor.

- **Imaging System** The LIS has an Imaging System that provides a correct (in terms of spectrum, direction...) radiation beam to the CCD. It consists of a telescope with a FOV of $80^\circ \times 80^\circ$ that converges the entering light to a 5° wide beam and sends it to a filter. The latter is a band-pass filter centred approximately at 777.4 nm (*). Afterwards, a re-imager is required to resend the radiation to the CCD in a proper way.

(*): The centre of the band-pass is actually a bit shifted to low wavelengths, as the incident beam has non-perpendicular waves, and the band-pass filters shift their centring for non-perpendicular incising waves.

- **Focal Plane Assembly** In order to configure an image (also a map that shows a x-y distribution of radiation) a CCD is used. This device is a wide matrix of photo-diodes (the Focal Plane Assembly) alongside some electronic components that testify the triggering of the photo-diodes to a processor. Each element of the matrix (the photo-diodes) will generate an electrical impulse under some specific conditions. This conditions, in our case, is a threshold of incident radiation of a particular wavelength. As there is a matrix of photo-diodes, for each time that the processor records which photo-diodes are activated an x-y map of electric impulses is generated, which will be processed as an image.
- **Real-time Signal Processor + Background Remover** The scope of this subsystem is to provide a true/false response to a pulse, deciding if it is from a lightning or from background. To do so, it averages the output coming from the CCD and generates an estimation for the

background value. Then, it subtracts the background estimation to the current signal, leaving a signal with less offset (near 0) and some peaks (that correspond to lightning pulses). This signal is then processed and, if a peak surpasses a threshold, it is considered a lightning. This threshold can be modified in order to match different requirements. For instance, during daytime the threshold must be higher as the amount of noise (peaks of incoming light that are not lightning) is bigger and more frequent.

- **Event Processor and Formatter** When a peak in the signal is detected and accepted as a lightning pulse, the system must process this raw measurement to provide a formatted set of data. To do so it is used an algorithm that groups the peaks ("lightning pulses" or events), and applies some criteria to provide data files that contain more complex information. This criteria will be described further in the document.
- **Power Supply**
- **Interface Electronics**

3-D model representations of the system are displayed in figure 2.5 at the end of this section.

Measurement approach

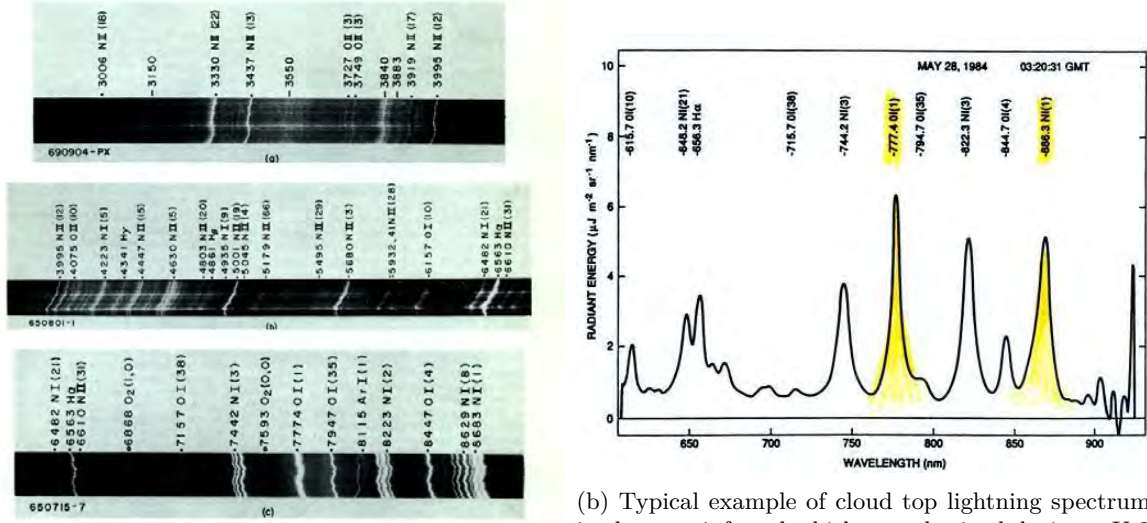
It is important to mention that lots of efforts were made to make sure that LIS could also detect lightning during daytime, against an also-very-bright background. This requirement implied some particular sensor characteristics, basically to increase the lightning signal in front of the one from the background.

To do so, filtering in intensity, space and time is applied, using electronic components. Four methods are used:

1. Spatial filtering: matches the instantaneous field of view (IFOV)³ of each excited element on the CCD to the typical value for the cloud-top area illuminated by a lightning.
2. Spectrum filtering: the device has a band-pass filter on 777.4 nm radiation, emitted by OI(I); which is one of the wavelengths where lightning emit the most, as is shown in the figure 2.3.
3. Taking advantage of the difference between the pulse duration of a lightning (400 μ s) and the background (constant).
4. Observing if a radiation input is temporal(**) or constant. The former should be associated with a lightning pulse and the latter to the background.
5. Subtraction of successive frames, which allows the background signal, which remains roughly constant, to be eliminated.

(**): To properly capture the lightning pulse in front of the background signal a high signal-to-noise (SNR) ratio is required. This ratio improves if the integration time, which is the time during which the pixel accumulates charge between two excitations, is approximately this same as the pulse duration. For a 400 μ s pulse a 1 ms integration time would be appropriate, but technical difficulties imposed an integration time of 2 ms. The measurement time that is associated to the whole integration time corresponds to the latter's center, i.e. 1 ms before the end of the integration time [4].

³The IFOV is the area on the surface that a sole pixel of the CCD is observing.



(a) Slit-less lightning spectra of lightning in the (a) near-ultraviolet, (b) visible, and (c) near-infrared spectral regions. Wavelengths shown in Å.

(b) Typical example of cloud top lightning spectrum in the near infrared which was obtained during a U-2 aircraft study on thunderstorms.

Figure 2.3: Lightning Spectra Information. Source: [6]

2.1.3 Event Grouping Process

As previously has been mentioned, a lightning pulse, a peak in the signal, is called an event. As one lightning can generate various events, there has to be some criteria to correctly group these events in flashes, which is the name given to the whole discharge. Also, inside of a flash there can be groups of events that take place closely in time and space. They will be called groups. A simple criteria⁴ of space and time separation between events is applied to group the them in *groups* and *flashes*. This criteria can be summarized in the three points below.

- Event: excited pixel on the CCD.
- Group: One or more events that take place in adjacent CCD pixels and occur at the same integration time.
- Flash: One or more groups that occur within less than 330ms and 16.5km of separation.

With these simple definitions it is possible to group the phenomena, as displayed in figure 2.4 below.

The data provided by LIS has the events labeled with numbers (1,2,3...n), where those are its group and flashes. Inside each flash, groups go from 1 to n; in each LIS viewtime, flashes also go from 1 to n. Worth to mention, several filters and improvements have been introduced in this algorithm in order to generate data as precise as possible. Details about the algorithm and its quality control can be found on [8].

⁴More detailed information about the algorithm available at [8].

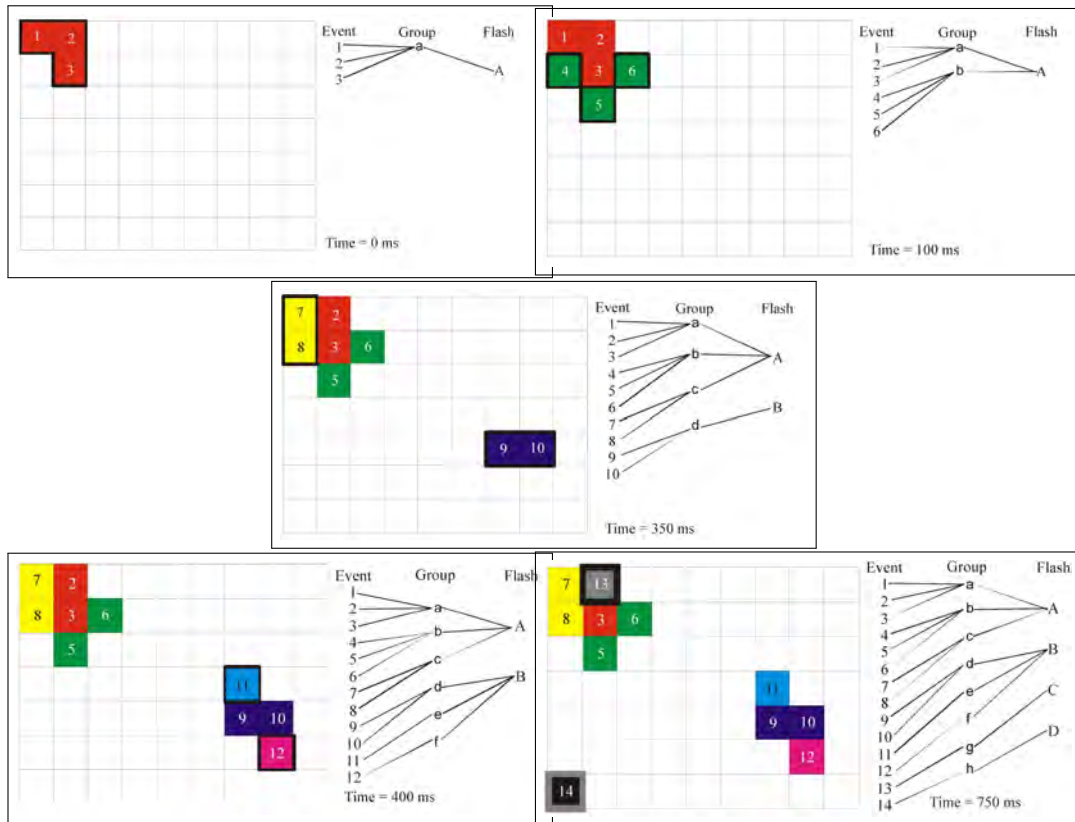


Figure 2.4: Example of grouping process. Source: [8].

2.1.4 Technical properties

In the table 2.1 below are displayed the most relevant technical properties of the ISS-LIS sensor.

PARAMETER		VALUE	PARAMETER		VALUE
FOV		80° x 80°	B	Location	1 pixel
IFOV (nadir)		4 km		intensity	10%
A	wavelength	777.4 nm		time	tag at frame rate
	bandwidth	1 nm	C	sensor head assembly	20 x 37 cm
Detection Threshold		4.7 \micro J m		electronics box	31 x 22 x 27 cm
SNR		6	Weight		20 kg
CCD Array Size		128 x 128 pixels	Power		30 Watts
Dynamic Range		>100	D	data rate	8 kb/s
Detection Efficiency		~90%		format	PCM
False Event Rate		<5%	E	Integration time (I.T.)	2 ms
				Convention for time assignation	1 ms before end of I.T.

Table 2.1: ISS-LIS sensor technical properties. The frame rate is of 2ms and the total FOV is of 600 x 600 km. Source: [9]

Area **A** is "Interference Filter", **B** is "Measurement Accuracy", **C** is "Dimensions", **D** is "Telemetry" and **E** is "Time labeling at the CCD".

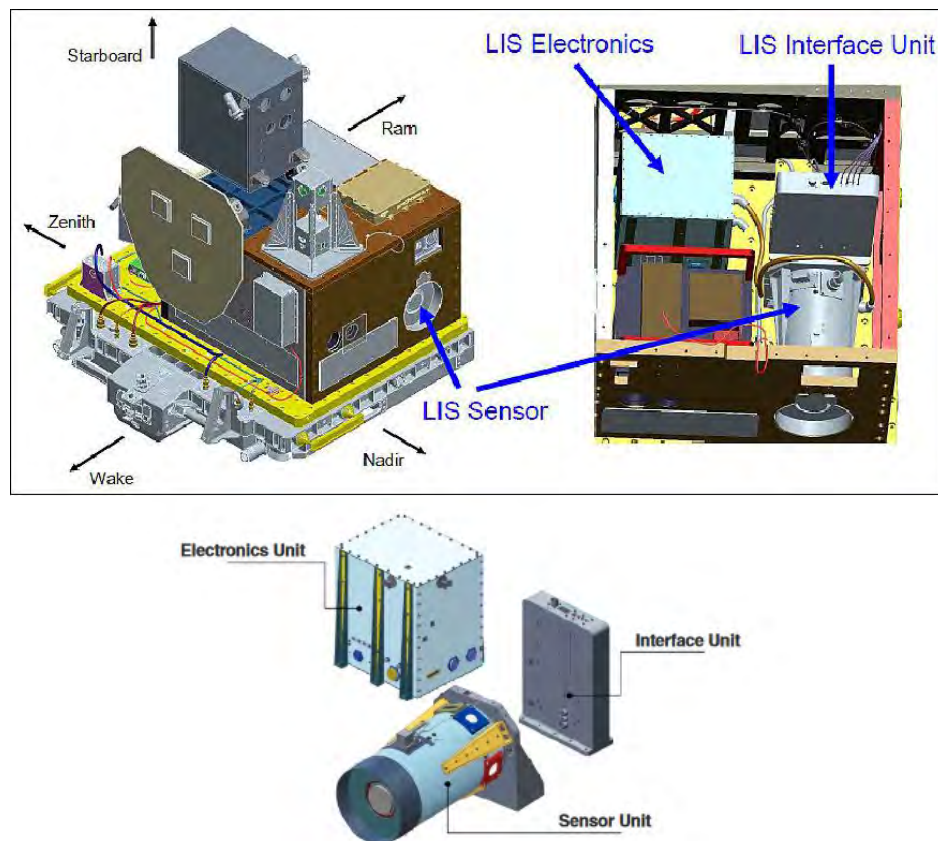


Figure 2.5: 3-D models of the ISS-LIS sensor. Source: [7]

2.2 LIGHTNING MAPPING ARRAY

The Lightning Mapping Array or LMA is a system that locates in 3-D and time very high frequency (VHF) emissions from lightning. The system consists of a set of sensors (radio frequency antennas), located at a distance of kilometres from each other. The term "LMA" is used widely to refer all the systems that follow the same design in the world. Indeed, there are LMA systems installed in different places, by different institutions. For instance, the Lightning Research Group (LRG) at Polytechnic University of Catalonia (UPC) has control over three of those: one in the south of Catalonia, called "Ebro"; and the others in Colombia, in Santa Marta and Barrancabermeja.

Despite the introduction of satellite-based lightning measurement systems such as LIS, which provide a global coverage, LMA systems still remain relevant. Since they have a much higher detection efficiency, their measurements can be used as a reference for satellite-based sensors calibrations.

2.2.1 Principle of Operation

It is known that lightning emit VHF electromagnetic pulses. These pulses can be measured by radio antennas that, when set up in a network (fig. 2.6), can track the position of the VHF source. The principle of operation relies on computing the time of arrival (TOA) of the each signal and then computing the origin by modeling the light velocity in the medium.

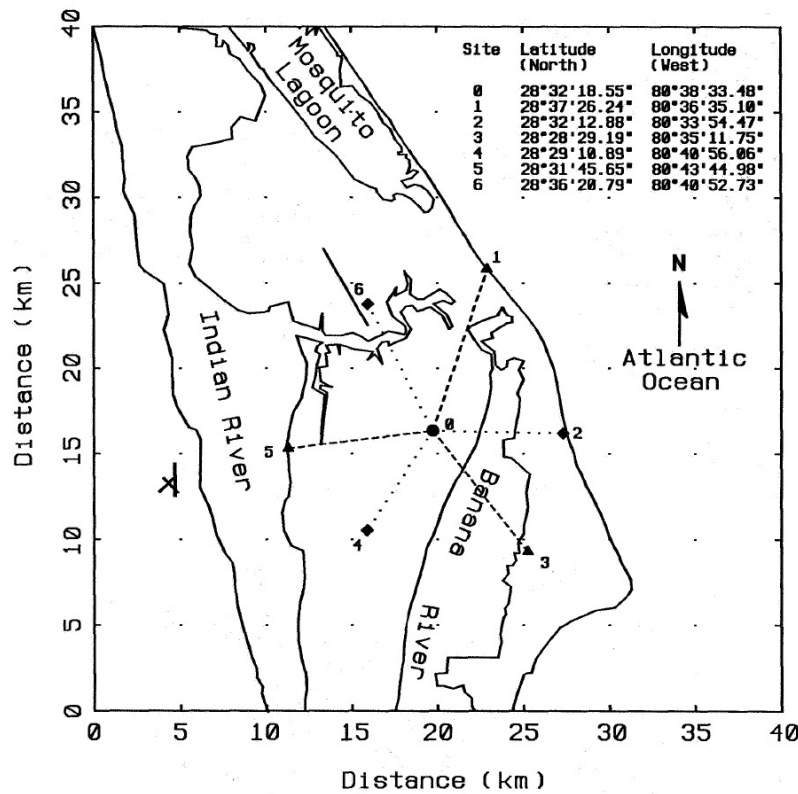


Figure 2.6: Lightning Mapping System antennae locations. Source: [11]

Each antenna computes the TOA of its received pulse, and then – knowing the propagation velocity of the EM wave – the distance of the source from the antenna. Afterwards, with the correlation of all distances computed by all the antennas and their position, the source coordinates can be obtained. The source position, its time of detection and power will be saved. Each source is registered for the antennas as an electromagnetic pulse (EMP) with its corresponding energy, or power. The detected sources' power may vary widely: using the LMA system they have been registered to range from 1W up to 10-30kW [12].

2.2.2 Ebre Lightning Mapping Array

The LRG of UPC has installed over the Ebre delta a set of sensors, forming the Ebre LMA. It is the LMA system from which the data used in this work is generated. In the west Mediterranean sea, few storms -compared to tropics in South America- take place. Furthermore, during the cold season they occur over the sea and during the hot season over the mainland. Then, placing the LMA system throughout the Ebre delta has revealed a great solution for capturing thunderstorms from both seasons, as its measuring range reaches both the sea and the mainland. Moreover, the presence of the LINET network in the area provides the possibility of complementary observations. In the figure 2.7 below the position of LMA antennae is displayed. Currently 7 of them are operative.

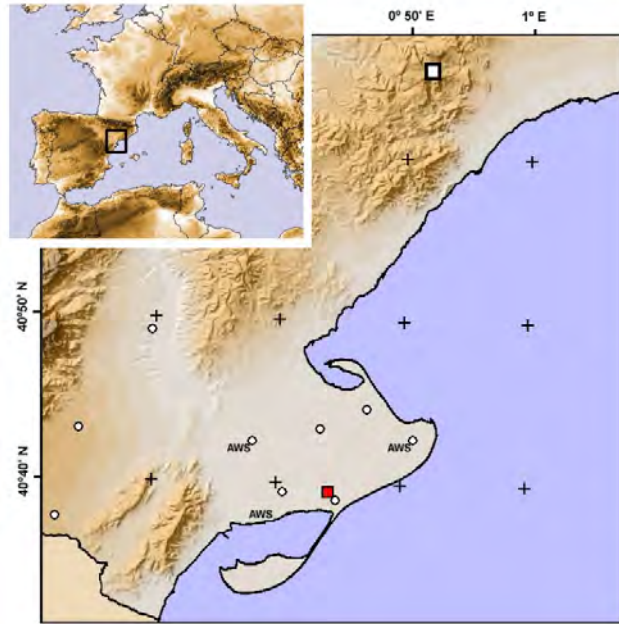


Figure 2.7: Position of LMA antennae in Ebre delta (white circles). Source: [13].

There is a town in the delta's centre named Deltebre, so the area may be called Deltebre Area as well.

Chapter 3

Sensors' Data Handling

Contents

3.1	ISS-LIS HIERARCHICAL DATA FORMAT FILES	19
3.1.1	LIS_ HDF_ processor.m	20
3.1.2	LIS_ vs_ LMA_ comparator.m	24
3.2	LMA DATA	25
3.2.1	LMA_ zoom7.sci	25

The objective of this chapter is to give an introduction to the reader regarding what data formats and programs have been used to get the results displayed further in this document. Here will not be presented a technical description of the codes nor the data, but it should be regarded more as a comprehensive description that will allow the reader to understand the outputs of the codes and how the data from the sensors is treated.

3.1 ISS-LIS HIERARCHICAL DATA FORMAT FILES

The Hierarchical Data Format (HDF4) files are items formatted in a self-describing way (i.e. they contain metadata that describes the data that is stored within the file), and they are designed to store large amount of very different scientific data¹. Basically, these files can contain the following types of data, also represented in the figure 3.1a below:

- Raster Images: Graphical representation of data; in this case, the orbit.
- Palette: Information about graphical display of the data in the raster images (e.g. colour distribution).
- Scientific Data Set: N-Dimension matrix of data (includes the attributes of the data points, which are the axis of the matrix).
- Annotation: Text data that provides info about the file
- Vdata: data stored in table format.
- Vgroup: group of data items. A Vgroup can contain one or more of the previous data items.

This way of information organisation allows the storage, in one file, of lots of data that is indeed related but not necessarily has the same format, nor refers to the same physical phenomena. For instance, in ISS-LIS case each file contains information of a whole ISS orbit. Nonetheless, each file contains information about the orbit itself (start time, start coordinates in Geo. Coordinates System...); information about the LIS sensor in every moment along the orbit; information about lightning phenomena recorded by LIS etc.

All these different types of data will be stored under different Vgroups, which will contain various HDF items (Sci. data sets, Vdata tables...), depending on the respective requirements. In the figure 3.1b

¹Most information in this section has been extracted from [14].

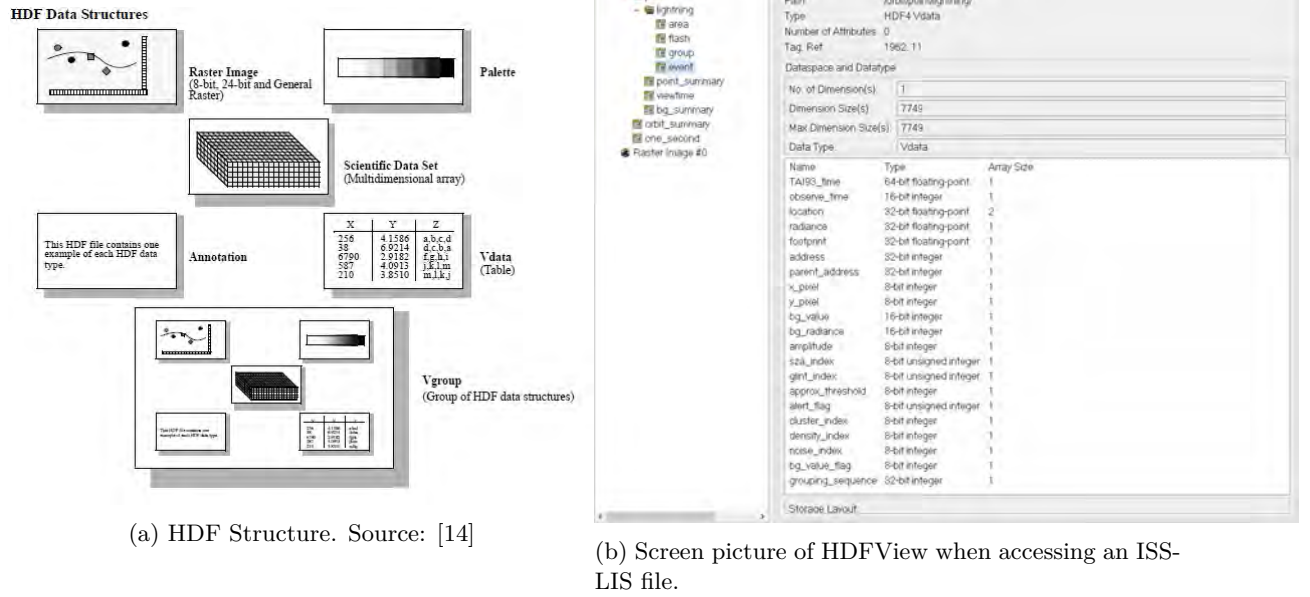


Figure 3.1: ISS-LIS HDF files organisation

this is clearly displayed. Each "folder" icon represents a Vgroup item, than contains other items. In this case, all information except the Raster Imager is stored in Vdata tables. Notice also how for the selected Vdata set, "event", some metadata is displayed on the menu on the right: this Vdata table has 21 fields that contain data of different sizes, from 7749 events or "rows"; the class of the data arrays etc.

More precise information about this file format can be extracted from creators' website ([14]). Although the HDF4 files can be accessed through a number of programs (e.g. HDFView), they do are not useful for massive data extraction. Thus, in this project a custom MALTLAB code will be used, named "LIS_HDF_processor.m". MATLAB uses its own commands for accessing the HDF4 files, so a particular description for this case is required. The MATLAB codes developed for this project are available at the appendix A.6 and at <https://github.com/icarfontcu/LIS-LMA-data-reading-codes>.

3.1.1 LIS_HDF_processor.m

The program was made to satisfy the need of creating files of LIS data that can be an input of the program "LMA zoom7.sci", a Scilab code made by van der Velde, O. that plots data from LMA, LINET, LIS and other sensors in the space-time domain. As it was made, "LMA zoom7.sci" receives LIS data from txt files. Considering that the LIS full data is stored in HDF4 files, the need of a program that converts HDF4 to txt files arises. Furthermore, some relevant issues appeared, regarding the bulk data downloading and selection processes. They will be discussed later.

All the data is available online and can be download through Earthdata website client. When this program was developed, the data was not yet downloaded on any UPC server so some assumptions, mainly the organization of said data, had to be taken.

The program has a total of 7 functionalities, to be selected trough a "switch" function when the program starts. Some of them more relevant than others. Following is a description of those functions.

READ AND STORE INFORMATION FROM HDF FILES

This case will write txt files from HDF4 files. To fulfil this task, some other functions will be previously executed.

"Check_for_folders"

This program is written bearing in mind the files organization of Earth Data HTLM server². This is:

1. Year folders
 - (a) Orbit ID folders
 - i. Bulk data

Inside the latter are all the .hdf files that contain the data of the events detected during the orbits. Supposing that the data, once downloaded, would be organized through:

1. General folder
 - (a) Subgeneral folder
 - (b) Subgeneral folder
 - i. subsubgeneral folder
 - ii. subsubgeneral folder

and so on until the last folders, where there would only be .hdf data items.

The first task that the code shall bear is to find the data inside whatever server it is. This task is given to the function *"check_for_folders"*. What this function does is, given a general directory on the computer (e.g. *C : /Users/Name/Desktop/Data*), go through all the subfolders of the directory until finding those folders with only data files (or not-folders, as it is coded).

Precisely, this folders use the MATLAB function *"folderinfo"* to get the information about the items that are inside the folder where the function currently is set. Using *folderinfo.isdir*, the user gets a logical comma-separated-list of which items inside the folder are directories. With that information we can do some conditional commands. In our case, as the info. is organized, the code will set the current file as a final reading directory only if no item in the folder is a folder. To do so, we can transform the comma separated list to a logical array *"cell2mat()"* and check for 1 (directory) or 0 (not directory).

If the folder is a final reading directory its address is saved in an size-increasing string array, *"dir_list"*, which will return to the main program as an output of the function. If the folder is not a final reading directory (it does contain other folders) the function recalls itself and explore the subfolders that the current folder has.

"Select_interestingfiles"

Once we have the list directory for the data that we want to acquire, we shall extract that data. It is supposed that the user will be searching data with some space and time boundary conditions (they must be specified in the first lines of the codes). Supposing also that the UPC server will have a lot of bulk data that may not be interesting, the program will search in the folders of the directory file for the HDF4 files containing data that is interesting (events within the space-time conditions). Those files will be called *"interestingfiles"*, and those events *"interestingevents"*. The utility of this method is that a list of *"interestingfiles"* can be stored for later use: in case the user would want to operate again with those files, the program wouldn't need to search through all the data base

To check if the files contain any *"interestingevents"*, the MATLAB functions *"intersect(a,b)"* and *"find(x>y)"* have been used, in order to select the array indexes where the data that is within the space-time boundaries is. Previously another method was used (line-by-line and logical true/false analysis) was made, but the use of *"intersect"* and *"find"* turned up to be much faster.

²<https://ghrc.nsstc.nasa.gov/pub/lis/iss/data/science/nqc/hdf/>

ISS-LIS HDF4 files data organisation

In order to get the data arrays where MATLAB can search for interesting data it is previously required to import that data. Then, the knowledge about the LIS HDF4 data becomes important. IN the ISS-LIS case, each HDF contains:

1. Orbit Information (groupdata)
 - (a) Point information (groupdata)
 - i. Lightning data (groupdata)
 - A. area (vdata)
 - B. flash (vdata)
 - C. group (vdata)
 - D. event (vdata)
 - ii. point summary (vdata)
 - iii. viewtime (vdata)
 - iv. background summary (vdata)
 - (b) Orbit summary (vdata)
 - (c) one second (vdata)

2. A raster Image of the Orbit through the world map

MATLAB can access the structure and meta-data of these files with the command "hdfinfo", which will give the names, properties (size, class...) etc. of the data.

Once you read a groupdata address MATLAB will show a structure containing different field with labels, and data characteristics of that data group. The "sub_ folders" (vdata or groupdata) will be labelled as 2 different fields. Accessing it as a structure field will lead to an opening of a new structure with fields or directly data, respectively. The vdata files within will be regarded as a field, that will contain at the same time data in the structure form. E.g., we can access the lightning data groupdata through:

$$vdata_address(i) = fileinfo.Vgroup.Vgroup.Vgroup.Vdata(i)$$

We can read the vdata that we want with the command `hdfread(vdata_ address)`. With *i* (1,2,3 or 4) we can select the area, flash, group or event vdata, cell arrays where each cell are cell arrays containing columns of same-class and size data (i.e. `event_ vdata=hdfread(vdata_ address(4))` will give a group of group of cell arrays. We can store them in a new structure element called "event": `event.coordinates=event_ vdata(3)`. (The third column of this `vdata_ address(i)` contains 1x2 elements which represent the coordinates of the event)).

Now, with this data organization, more clear and manageable, we can introduce the space-time boundaries and know if the hdf file we are reading is an interesting file. This last information will be stored in a vector that will remain in the global workspace during the rest of the program.

PRINT txt FILES WITH EVENTS' INFORMATION**"w_ txt_ files"**

This function will read all the interestingfiles and plot its interesting events to a txt file with the similar name: "ISS_ LIS_ 20171018_ 1032_ 1034_ events", where the 20171018 indicates the date time through the format YYYYMMDD and the 1032 and 1034 indicate the time (HHMM) of the first and last event in the file.

To do so, we will introduce again the space-time boundaries and with the indexes of the interesting events apply the MATLAB function "fprintf()" to print row-by-row the relevant information. The data will be separated with a tab and the file will contain a header with the titles of each column (e.g. TAI93³ Latitude, Longitude...).

³Time of the measurement in seconds from 1 Jan 1993. To convert from TAI93 to UTC the former has to be summed to the date (1 Jan 1993) as follows: (7.8248e+08)[s] + (1 Jan 1993) = (18 Oct 2017 10:31:56).

At this stage, it was required to make a decision on which data from the HDF4 file will be stored in the txt files (remember that we can access area data, flash data, group data, event data, orbit data...). Regarding that the interesting parameters to measure are radiance, space-time coordinates of the events and its grouping with other events, the stored parameters are:

- TAI93 time: TAI93 time in seconds resolution of the event
- event LAT and LON: space coordinates of the event
- radiance: event's radiance
- group address: The group to which the event belongs
- group LAT and LON: the group's radiance-weighted centroid
- flash address: The flash to which the event belongs
- flash LAT and LON: the flash's radiance-weighted centroid
- area address: The area to which the event belongs
- area LAT and LON: the area's radiance-weighted centroid
- exited pixels' position on the CCD
- observe-time: the time during which the LIS sensor was focused on the are

The reader should note that the addresses of each group, flash and area are renewed for different interesting -txt- files, so events from two different files may have the same group address, which doesn't mean they do belong to the same group. This is an aspect to improve in the near future, but at the moment it doesn't represent a problem, as for this documents the txt files have been processed (and displayed) separately.

The output txt files will have a column for each saved parameter, and a header containing the title of each column. A screen shot of a txt file is displayed in the figure 3.2.

TAI93_time	e_lat	e_lon	e_radiance	group	g_lat	g_lon	flash	f_lat	f_lon	area	a_lat	a_lon	a_observe_time	x_pixel	y_pixel	bg_radiance
7.98988647127281321e+08	48.824	0.812	12.653	48.769	0.001	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647127281321e+08	48.768	-0.024	22.653	48.769	0.001	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647127281321e+08	48.727	0.031	15.653	48.769	0.001	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647299861061e+08	48.817	0.022	21.654	48.767	0.024	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647299861061e+08	48.761	-0.014	37.654	48.767	0.024	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647299861061e+08	48.776	0.077	20.654	48.767	0.024	114	48.767	0.025	25	48.767	0.025	57	1	14	286	
7.98988647299861061e+08	48.727	0.041	24.654	48.767	0.024	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647310123681e+08	48.817	0.023	12.655	48.766	0.006	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647310123681e+08	48.768	-0.013	31.655	48.766	0.006	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647310123681e+08	48.726	0.041	11.655	48.766	0.006	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647313633201e+08	48.768	-0.013	11.656	48.768	-0.013	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647317142721e+08	48.817	0.024	23.657	48.763	0.011	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647317142721e+08	48.768	-0.012	48.657	48.763	0.011	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647317142721e+08	48.726	0.042	19.657	48.763	0.011	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647317142721e+08	48.776	0.079	14.657	48.763	0.011	114	48.767	0.025	25	48.767	0.025	57	1	14	286	
7.98988647319126371e+08	48.768	-0.012	11.658	48.768	-0.012	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647333210211e+08	48.759	-0.011	11.659	48.759	-0.011	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647410084011e+08	48.813	0.029	38.660	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647410084011e+08	48.756	-0.007	43.660	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647410084011e+08	48.772	0.084	26.660	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	1	14	286	
7.98988647410084011e+08	48.722	0.040	21.660	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647447773221e+08	48.811	0.032	54.661	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647447773221e+08	48.754	-0.004	72.661	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647447773221e+08	48.704	-0.041	12.661	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	3	13	302	
7.98988647447773221e+08	48.720	0.050	34.661	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647447773221e+08	48.770	0.087	47.661	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	1	14	286	
7.98988647447773221e+08	48.820	0.124	12.661	48.768	0.031	114	48.767	0.025	25	48.767	0.025	57	0	14	270	
7.98988647476413971e+08	48.810	0.033	16.662	48.766	0.038	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647476413971e+08	48.753	-0.003	21.662	48.766	0.038	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647476413971e+08	48.769	0.088	15.662	48.766	0.038	114	48.767	0.025	25	48.767	0.025	57	1	14	286	
7.98988647476413971e+08	48.719	0.052	11.662	48.766	0.038	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647501545191e+08	48.809	0.035	23.663	48.766	0.031	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647501545191e+08	48.752	-0.001	34.663	48.766	0.031	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647501545191e+08	48.768	0.090	13.663	48.766	0.031	114	48.767	0.025	25	48.767	0.025	57	1	14	286	
7.98988647501545191e+08	48.718	0.053	12.663	48.766	0.031	114	48.767	0.025	25	48.767	0.025	57	2	14	286	
7.98988647501544091e+08	48.809	0.035	44.664	48.768	0.035	114	48.767	0.025	25	48.767	0.025	57	1	13	292	
7.98988647501544091e+08	48.752	-0.001	37.664	48.768	0.035	114	48.767	0.025	25	48.767	0.025	57	2	13	291	
7.98988647501544091e+08	48.768	0.090	32.664	48.768	0.035	114	48.767	0.025	25	48.767	0.025	57	1	14	286	
7.98988647501544091e+08	48.718	0.053	19.664	48.768	0.035	114	48.767	0.025	25	48.767	0.025	57	2	14	286	

Figure 3.2: Screenshot of an output LIS txt file

EVENT PLOTTING

This task is optional and the writing of general events txt files may be done without plotting the data. Nonetheless, if the plotting is activated, the function **plot-events** is activated.

It simply plots on a gridded map the events, sized by radiance and coloured by group, of each file. It also draws the area where the data is selected, and plots a low resolution coastline loaded from the MALTAB mapping toolbox.

HDF4 FILE NAMES TO URLs

To download LIS bulk data you need a txt files with the URLs of the HDF4 files that you want to download from Earthdata server. These URLs have the following shape:

https://ghrc.nsstc.nasa.gov/pub/lis/iss/data/science/nqc/hdf/2018/0613/ISS_LIS_SC_P0.2_20180613_NQC_08218.hdf

On the LIS data website you can download a txt file with the URLs from all the HDF4 files whose orbit is within a period of time specified by the user. Nonetheless, this generates a problem: the selected files will correspond to orbits in the time period, but that orbit may not pass (actually will be the common case) through the area that we want to analyse. Therefore, the amount of data downloaded would be much higher than the required and the process of selecting interesting files way much longer.

A workaround to this problem was found by using the space-time domain search engine from the LIS website. This tool lets you specify the time period as well as the area were you want to look for, and displays a list with the names of the HDF4 files that contain data in this space-time domain. Then, it is required to select "by hand" this list and copy it to a txt file. This txt file will be later processed by the function `webfilenames2urls()`, that will construct URLs from the files names taken from the website list.

Then, these URLs are written on a new txt file, and finally we are able to download the interesting HDF4 files with the `curl` command:

`wget -user username -ask-password -auth-no-challenge -no-check-certificate -i URLsfile.txt`

This, provided by instructions from the LIS website will download all the HDF files listed in the `URLsfile.txt`.

For an optimal usage of the LIS data with the program *LIS_vs_LMA_comparator.m*, the best praxis is to join all txt files for the interesting period of time with the aim to have a large sole file containing all the events. This can be done manually, since the number of txt files that the studies presented in section 4.2 require is of the order of 5.

Also, a posteriori Prof

3.1.2 LIS_vs_LMA_comparator.m

In order to compare the data from LMA and LIS a MATLAB program has been made. Such code compares data from both sensors (fundamentally events and sources) and associates them by time proximity. The output of the program are statistical result (typical value distributions, histograms...) that contain information about such associations. For instance, a source and an event that are separated in time less than 10 ms are considered to be associated. Since events and sources have different properties, via this association we can compare events properties (e.g. radiance) with sources properties (e.g. height) and produce distributions that allow a study of, say, the radiance evolution with height. A more exhaustive explanation of this approach is done in the section 4.2 and in this section only the structure and behaviour is to be discussed.

To use the program, the user should edit the code (first lines, indicated in MATLAB) to introduce the proper variables:

LIS_total_filename The txt file containing all the LIS data

LMA_filename The txt file containing the LMA data for of the interesting time period

timestep Maximum time between an event and a source to associate them

Plotting Options Histograms/plotting/sources_ and_ events. Set variable as 1 or 0 to plot histograms, typical values of each time bin or display all sources and events printed over time.

comparing length section This option enables the execution of a section that is used to compare the length of lightning computed from LIS or LMA data; results described in section 4.3.

A more detailed explanation of the user-introduced variables can be found in the user guide, at A.5. As the reader might infer, the fundamental output of this program is graphical displays of the evolution of some properties with others (e.g. radiance vs counts, height vs counts, radiance vs height etc). Its results can be seen in section 4.2.

Another section of the code is dedicated to assess which is the typical radiance of events that had sources associated at a given height. For this functionality, the user should set the variables DISCHARGE-ALT [m] and tolerance [m] accordingly. The results of this study are displayed in the Chapter 4.

3.2 LMA DATA

The LMA data is generated by each LMA antenna, by dumping ".dat" files each 10 min. Each antenna dumps its files respectively, containing information about the distance and timing of the sources detected by the antenna. In order to get the 3-D position of each source, a post-processing is required, made by an UPC-LRG⁴ algorithm. This algorithm crosses the information about various antennas regarding the same time, and dumps ".dat" files for each 10 min period, containing the position; timing; power and other values of the sources detected during the said period.

This crossover of information may be done more or less exhaustively. For instance, the user can select to cross information from 5 to all 7 antennas. The number of antennas is the number of simultaneous, separated detections that the algorithm requires to declare a source in the respective location. More antennas will give results with less noise but the user might lose information, as some VHF emissions might be only detected by less antennas than the specified. In the other hand, too few antennas will produce a very noisy result, where a lot of the declared sources may be errors or detections of other phenomena.

The LMA .dat files used in this document have been computed using a 5 (out of 7) antenna restriction, a parameter that has been found to give a nice equilibrium between quality and quantity. In order to process the said .dat files a program has been made with Scilab, LMA_zoom7.sci (Van der Velde. 2017). This program displays the sources in space and time domain, as well as other information such as their power, spread velocity etc. Alongside the sources the program can also display the LIS events, LINET detections, Meteorage detections and other data coming from other sensors. For this document, the LMA_zoom7.sci will be used mainly to compare LIS vs. LMA detections, and only a part of all the program applications will be executed. Below are described the capabilities of the Scilab program that are of use in this document.

3.2.1 LMA_zoom7.sci

The main goal of this program is to display the information about position and timing of detections in a comprehensive way. To do so, the user must select a 10 min LMA file and the program will display a panel with 4 graphs. In this panel there is information regarding

1. Time-height position: top panel
2. LAT-LON position: bottom left panel
3. LAT/LON vs height position: bottom right panels

The program allows the user to colour the detections under a number of parameters (power, time, flash...). An example of the output figures is fig. 4.1 below. Note that the sources have been coloured by time. As the reader may notice, the LIS detections are also displayed as a circles, coloured by group. To do so the user must select a corresponding LIS txt file. Symmetrically, the same can be done with LINET, if data is available. The usefulness of this display is that allows the user to have a generic view on the data gathered at a given time. It allows a search for space or time offsets and other particularities of the data set that with a less graphic approach could go unnoticed.

⁴Lightning Research Group at Universitat Politècnica de Catalunya

Chapter 4

LIS data analysis using LMA as reference

Contents

4.1	OVERVIEW OF GATHERED DATA	26
4.1.1	Space-Time distributions of detections	26
4.1.2	Influence of excited pixels' position on the CCD	37
4.1.3	Section summary	38
4.2	INFLUENCE OF VHF SOURCES' PROPERTIES ON ITS LIS DE- TECTIVITY	39
4.2.1	Hypothesis and analysis description	39
4.2.2	Sources' Height Influence on Detectivity	40
4.2.3	Sources' Maximum Power influence on Detectivity	45
4.2.4	Density of sources in the time bins	47
4.2.5	Section Summary	50
4.3	ANALYSIS OF THE FLASH DURATION CONCORDANCE BETWEEN LIS AND LMA SENSORS	51
4.3.1	Flash Duration Analysis	51
4.3.2	Section summary	54

In this chapter is mainly reported an analysis of the influence of the LMA sources' properties on its likelihood to be detected from orbit; along some other assessments. It has been done using data coming from LMA and LIS and associating it by a time criteria, in order to extract statistical distributions. For clarity purposes, in the section 4.1 is presented the data that will be used. Also, some comments regarding the availability of data can be found in appendix A.2. Finally, in the section 4.3 is presented a first-approach analysis on which is the assessed duration of a flash depending on the used sensor.

4.1 OVERVIEW OF GATHERED DATA

4.1.1 Space-Time distributions of detections

The time periods during which LIS detected events around Deltebre from March 2017 until October 2018 are listed below. Because the LMA files are provided each 10 minutes, the listing indicates only the start time of the 10 min period where the events that LIS detected occurred (i.e. event detected from 2017-10-18 10:31 to 10:36 are listed as 2017-10-18 10:30). Nonetheless, LIS usually has a view-time on Deltebre area of around 2 min of duration, so within this 10 min periods a lot of LMA data won't be detected by LIS, since it was outside LIS FOV. Further down this last perk of the problem is discussed.

- | | | | |
|---------------------|---------------------|---------------------|----------------|
| 1. 2017-10-08 10:30 | 4. 2018-04-29 11:30 | 7. 2018-06-06 14:20 | 10. 2018-08-31 |
| 2. 2017-10-18 17:00 | 5. 2018-05-25 01:40 | 8. 2018-06-13 17:50 | 11. 2018-09-18 |
| 3. 2018-04-27 13:10 | 6. 2018-06-05 15:10 | 9. 2018-08-09 | 12. 2018-10-18 |

Notation: YYYY-MM-DD hh:mm.

Also the notation YYMMDD hh:mm will be used in the future.

Fall of 2017 to Summer 2018

2017-10-08 10:30

This is the first and one of the most interesting case. In the figure 4.1 it is shown the time and space distributions of detected LMA sources (dots) and LIS events (circles). In the time distribution panel (height [km] vs. time [sec.] from the starting of the 10 min. time period), it is possible to observe a lot of vertical sources clusters. Those are flashes. Even if they seem almost only vertical, the reader must bear in mind that for an electrical discharge process the current time resolution is relatively low, and later on we will zoom in the panel to better see time distribution of a flash.

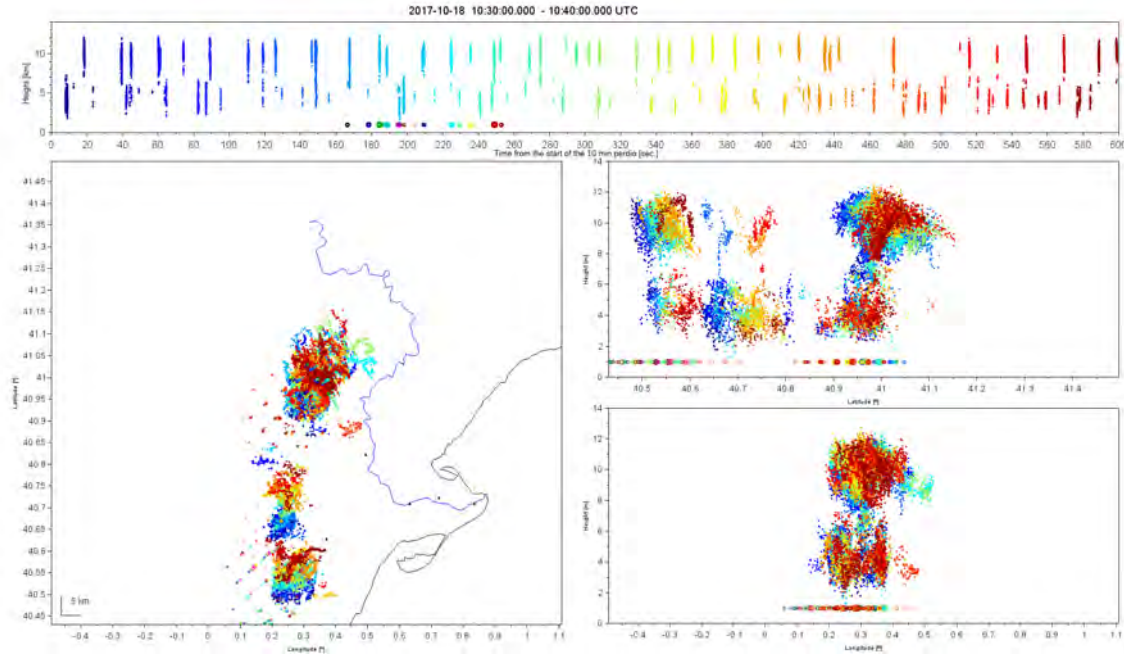


Figure 4.1: Space and time distribution of LIS and LMA detections during 17-10-18 10:30. Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.] Bottom right: height[m] vs. LAT/LON [deg.]. Sources are dots, events are circles. Sources coloured by time, events by group.

In this case, we observe how in a period of roughly 90 sec. there are multiple events, that concur with flashes on top of them. The events are printed at a height of near 0 km because LIS detects only luminosity on the surface of the clouds taking "pictures" from above. This means that by itself LIS only has LAT/LON distribution, not height. The location (in height) of the detected event could be achieved by crossing the LIS data with data of other sensors, but in this hasn't been done.

Even if as shown in 4.2 the LIS and LMA detections are near simultaneous, the LAT/LON panel from fig. 4.1 clearly show a kind of offset of the LIS detections from the LMA ones, to the S.W. This phenomena will be further observed in the remaining cases and it is a matter to study. This could result in some interesting conclusions about comparison of both sensors; but it has to be said that the LIS data provided to users is calibrated in order to correct these kind of offsets, and in the past the

4.1. OVERVIEW OF GATHERED DATA

LIS team has re-uploaded some data with offset corrections

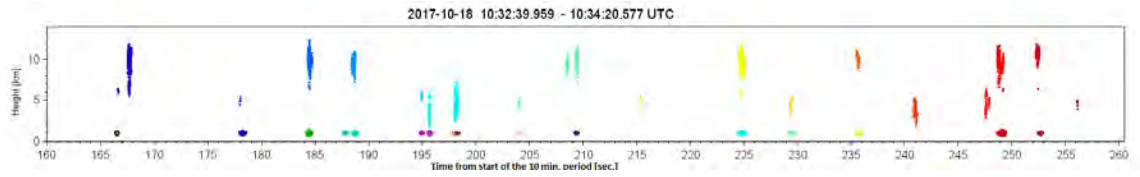


Figure 4.2: Height[m] vs. time [sec. from start of 10 min. period]. Zoom on time distribution of LIS and LMA detections during 17-10-18 10:30.

2017-10-08 17:00

In the fig. 4.3 printed below, it can be seen a pair of thunderstorms around Deltebre. The N.E. cluster and a more scattered thunderstorm in the shores of the Ebre's delta. Through the height/time panel it is clear how LIS detected at 70 sec. aprox. We can see in the LAT/LON panel how this flash was part of the scattered thunderstorm, and the height/LAT and height/LON panels show how the detection correspond to the lowest discharges.

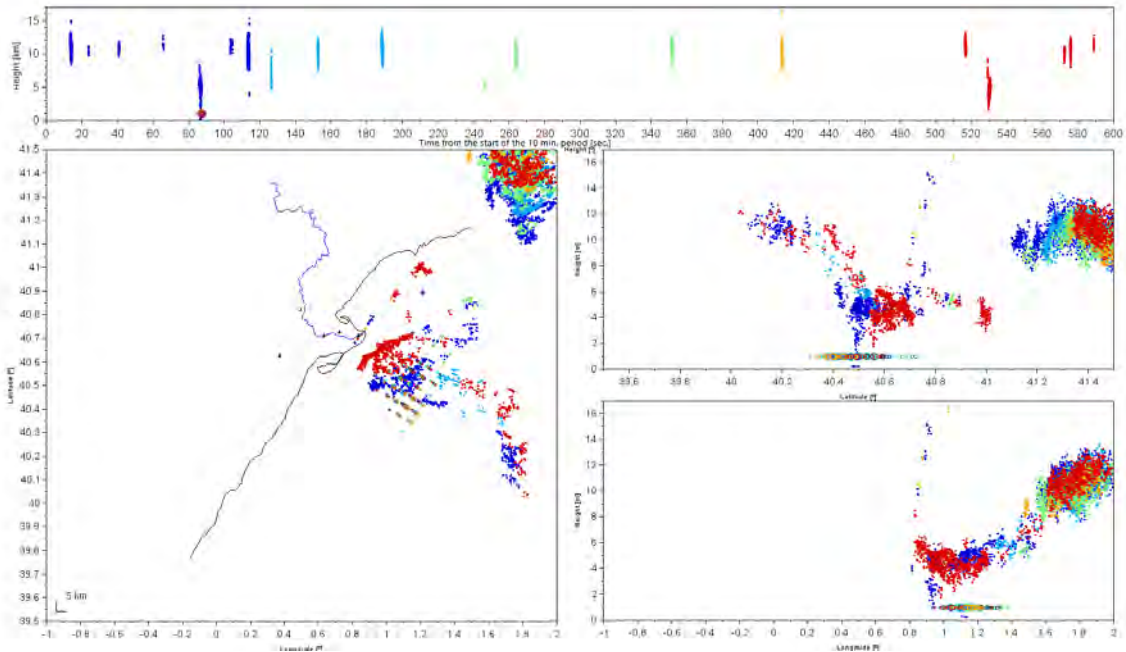


Figure 4.3: Space and time distribution of LIS and LMA detections during 17-10-18 17:00. Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.]. Bottom right: height[m] vs. LAT/LON [deg.]. Sources are dots, events are circles. Sources coloured by time, events by group.

The approximate entry and exit time of the LIS FOV throughout the area was 17:01:31 17:03:01, which is from second 91 to 181 (displayed in fig. 4.4). This means that LIS actually had the opportunity to detect some following flashes to the one that it detected. Notice how the LIS detection displayed in this figure is previous to the second 91. This is because the computation for the entry/exit time takes some assumptions that generate these kind of errors, and therefore its information is only useful for preliminary evaluation, at least in this stage of development.

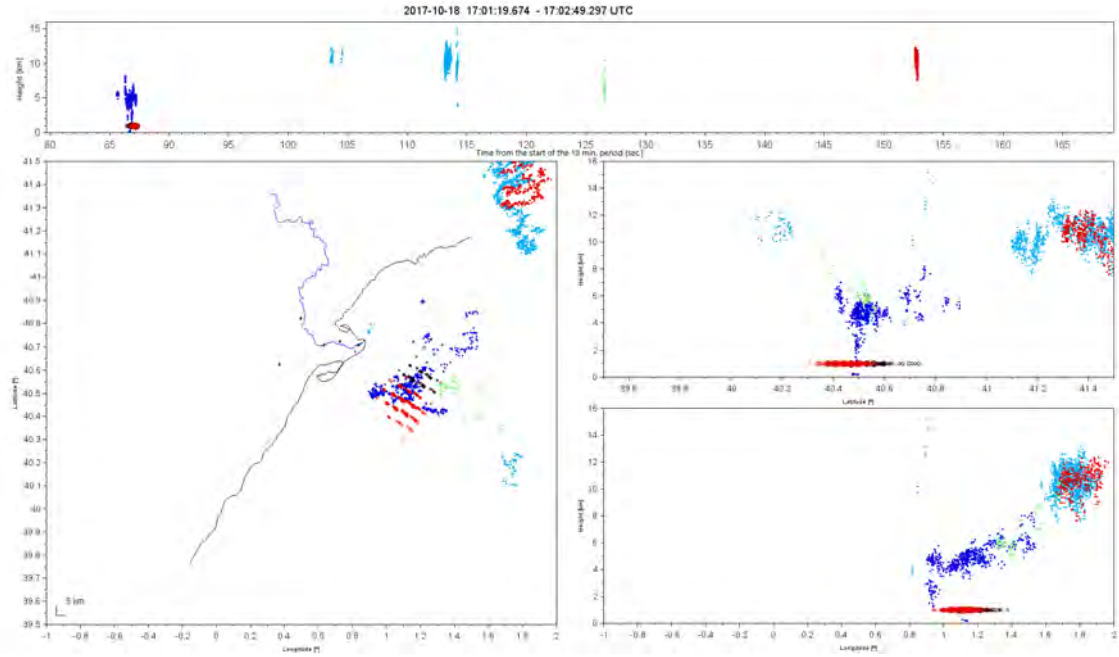
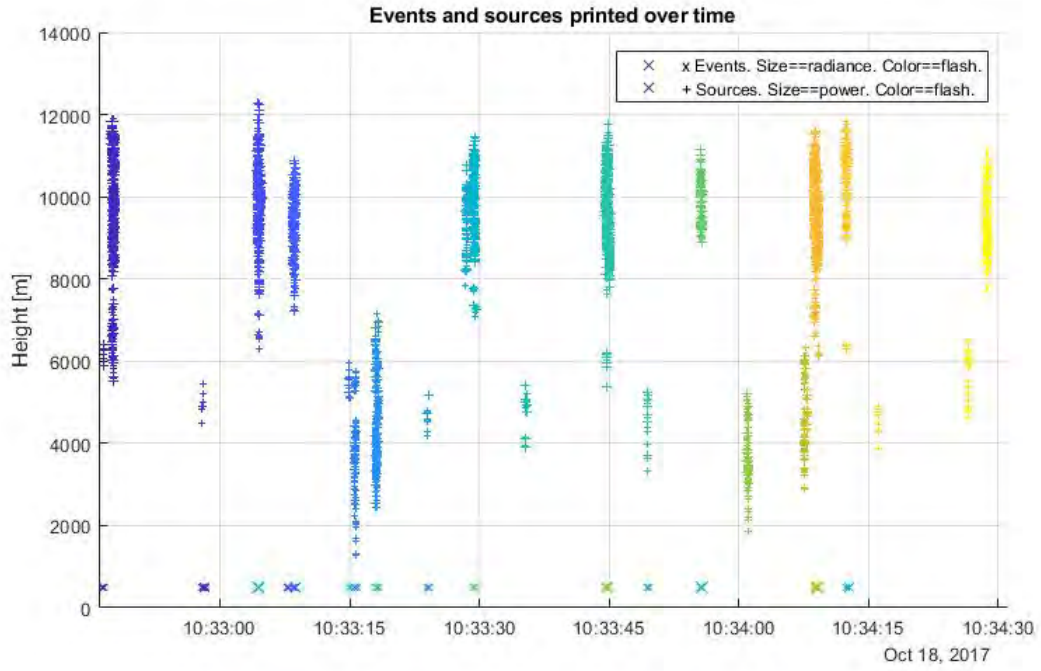


Figure 4.4: Space and time distribution of LIS and LMA detections during 2017-10-18 17:00. **Focus on LIS view time.**

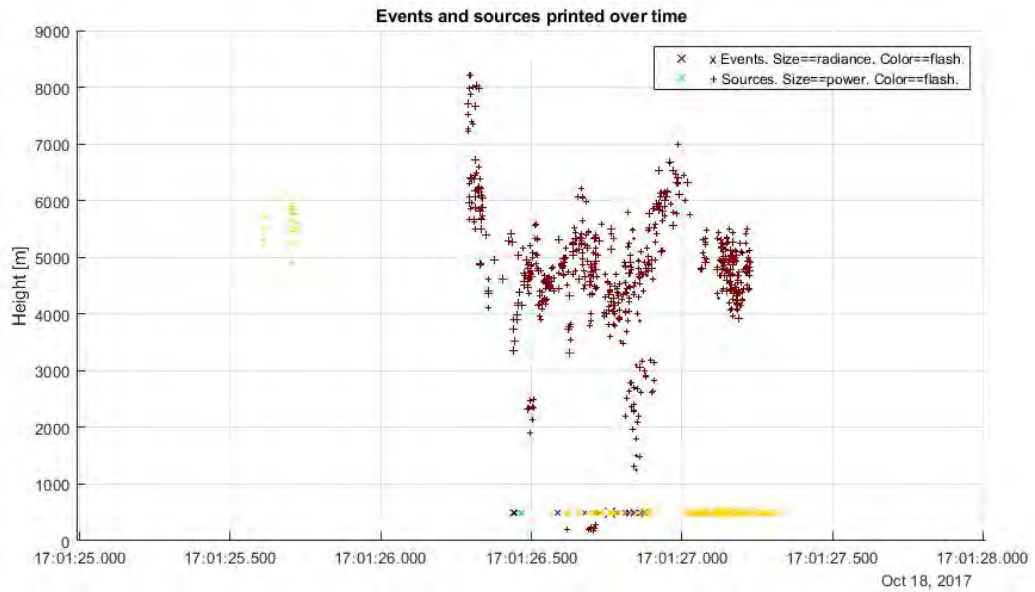
Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.] Bottom right: height[m] vs. LAT/LON [deg.]. Sources are dots, events are circles. Sources coloured by time, events by group.

With the inspection of the fig. 4.4 the reader may already infer how height should not have a strong, relevant role in LIS detection, as the only lightning detected in this case is the one with lower sources' altitudes (more info. on section A.3.2). Actually, it is possible that the detected lightning would have stroked the ground and caused a return stroke.

Equivalent results have been obtained with the program *LMA_ vs-LIS_ comparator.m*. They are displayed in the figures 4.5a and 4.5b below. There, it can be appreciated in more detail how the 10:30 case consists of a set of lightning and the 17:00 case a sole, intense discharge to the ground.



(a) 171018 10:30 period



(b) 171018 17:00 period

Figure 4.5: Sources and Events detections printed on height and time

2018-04-27 13:10

In the space-time representation of the events and sources detected during this time period (fig.4.6) it is possible to see that LIS only detected one of the lightnings during the current 10 min. period. The approximate entry and exit time of the LIS FOV throughout the area for this date was 2018-04-27 to 13:17:16; this is from second 391 to 463, so LIS detected the only lightning that occurred during its pass.

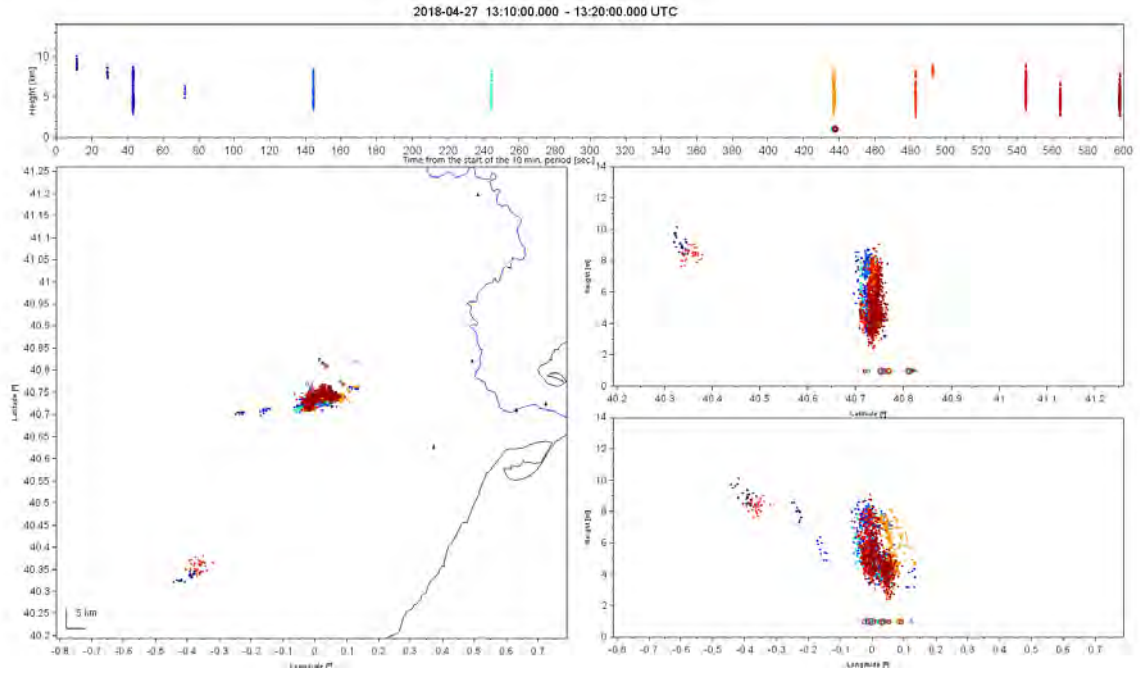


Figure 4.6: Space and time distribution of LIS and LMA detections during 18-04-27 13:10. Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.] Bottom right: height[m] vs. LAT/LON [deg.]. Sources are dots, events are circles. Sources coloured by time, events by group.

2018-04-29 11:30

This file does not seem very useful: the figure 4.7 shows how there was a single flash detection from LIS, which does not coincide in time or space to sources detected by LMA. This detection is particularly unsettling as it has a very high offset in time and space, so LIS really detected something that LMA did not, which is not usual.

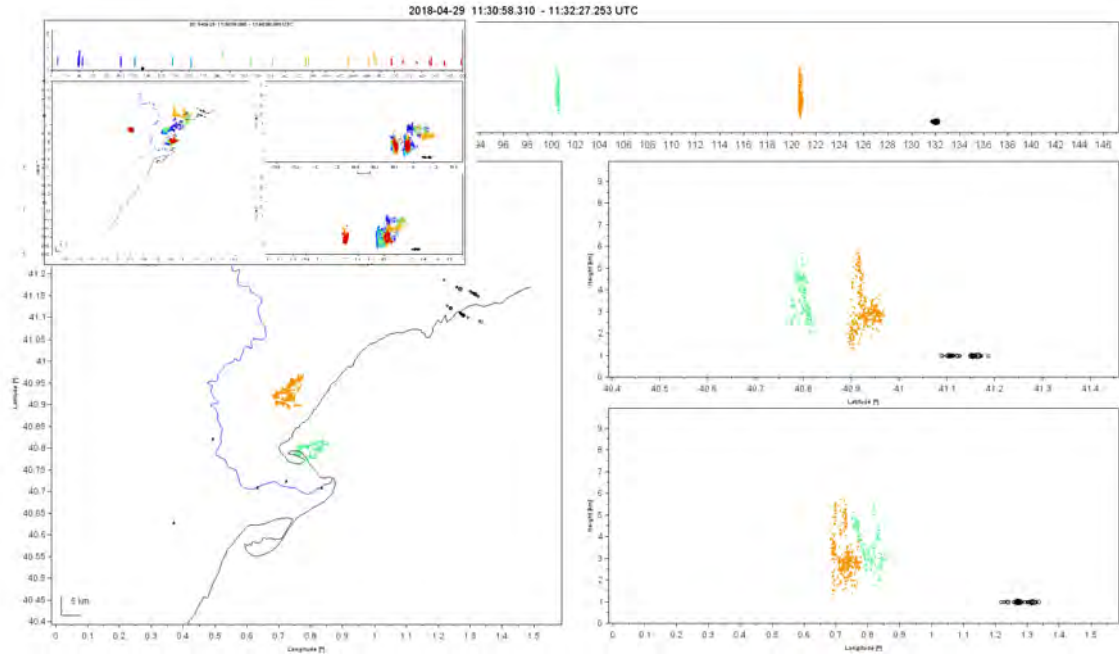


Figure 4.7: Space and time distribution of LIS and LMA detections during 18-04-29 11:30 view time. Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.] The smaller picture displays the whole 10 min period.

2018-05-25 01:40

During this time period, as seen in the figure 4.7, LIS detected some lightning in front of the coast, at S.W. from the delta. In the time-space domain window it is possible to observe that this LIS detection is not simultaneous with any other LMA detections. As a very good signal from LIS was registered (the dots correspond to the same flash but come from a number of groups), LINET data was checked and a simultaneous stroke was found in the same place. Therefore, a discharge did happen but was not detected by LMA.

4.1. OVERVIEW OF GATHERED DATA

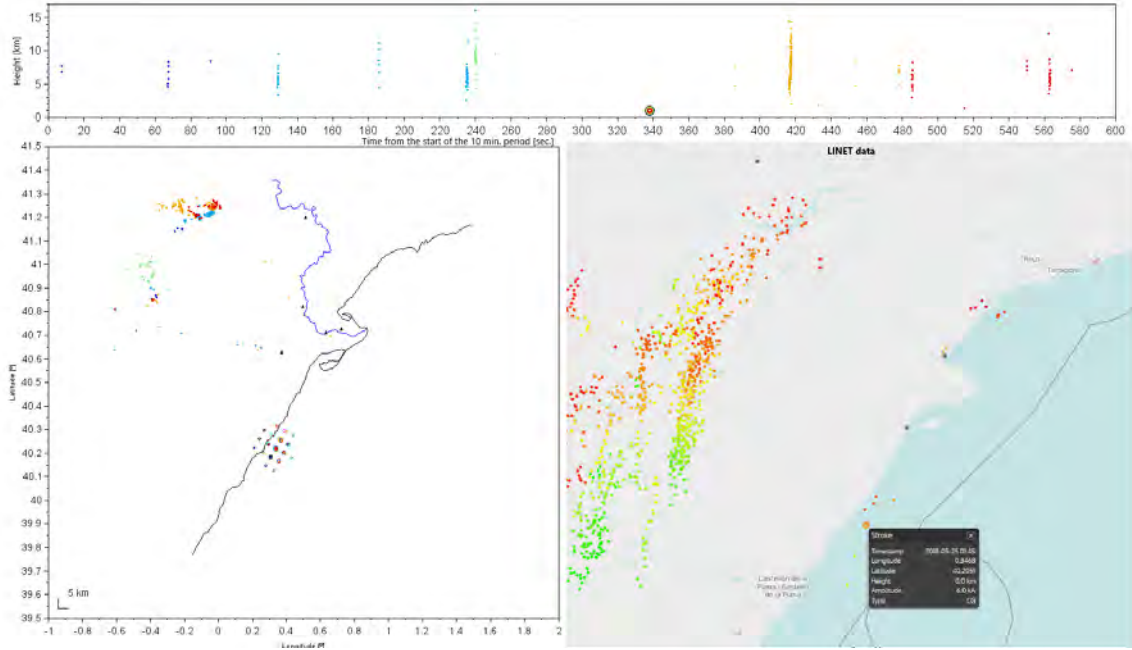


Figure 4.8: Space and time distribution of LIS and LMA detections during 18-05-25 01:40 view time. The colourful inside-figure displays the strokes detected by LINET. height/LAT and height/LAT distributions are not displayed because the non-validity of the data.

2018-06-05 15:10

In this case, displayed in the figure 4.9, only one lightning was detected. The time window of the LIS FOV was, approximately, from second 360 to 430, so probably the detected flash was the only one that LIS could had seen.

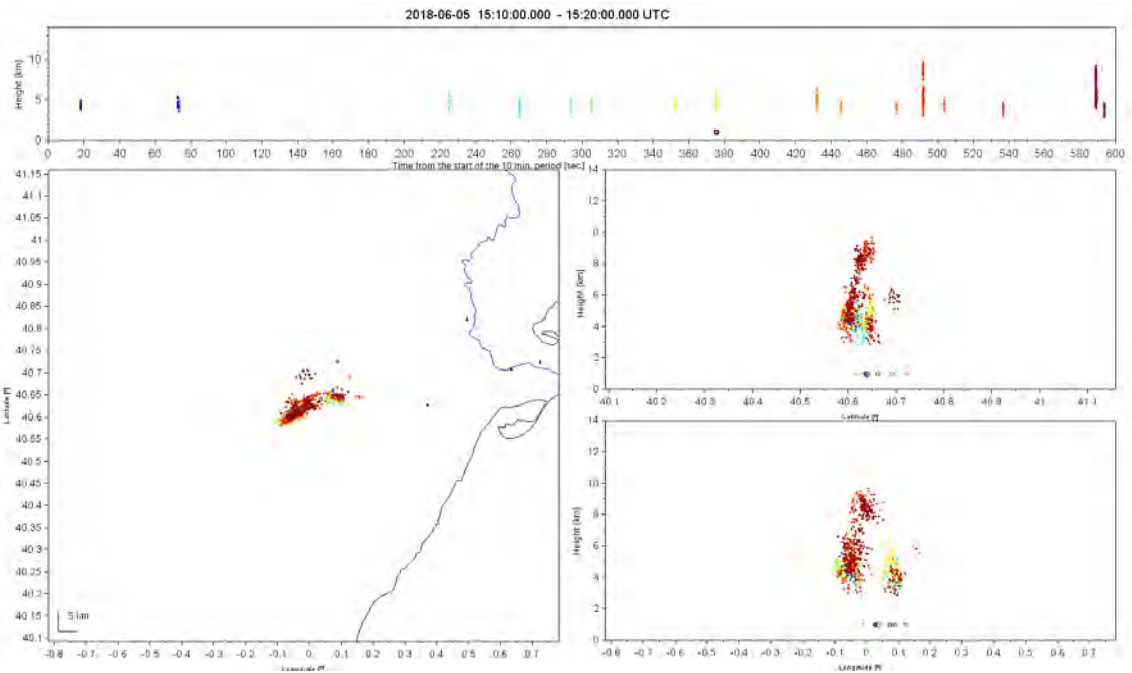


Figure 4.9: Space and time distribution of LIS and LMA detections during 18-06-05 15:10 view time. Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.] Bottom right: height vs. LAT / height vs. LON

2018-06-06 14:20

During this period few data was recorded by LMA: only what it seems to be a portion of a full-scale lightning, far S.W. from Deltebre (fig. 4.10). Furthermore, it was detected at the edge of the interesting Deltebre area, so information about the possible development of discharges outside the space boundaries lacks. Moreover, LIS didn't detect that phenomena either: it only detected two lone events far to the North. This data has no interest for methodical analysis.

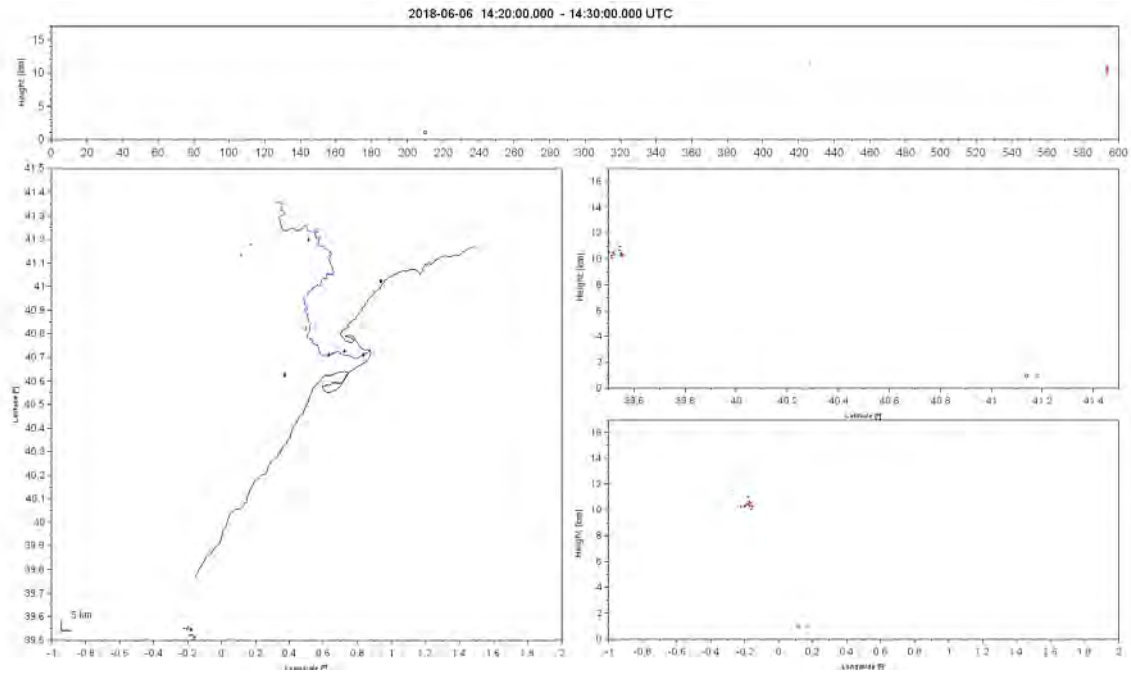


Figure 4.10: Space and time distribution of LIS and LMA detections during 18-06-06 14:20 view time.
Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.]

2018-06-13 17:50

In this date LIS detected a lightning simultaneously with LMA, as displayed on the figure 4.11 below. LIS seems to have a relatively large offset in S.W. direction, but the time domain windows shows how LIS detections matches perfectly with a lightning detected by LMA.

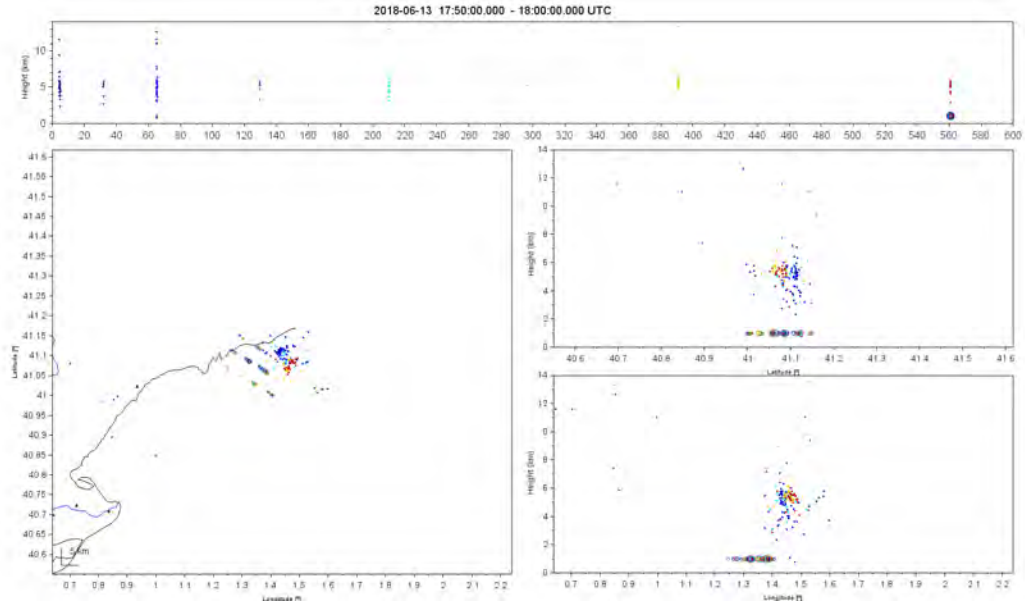


Figure 4.11: Space and time distribution of LIS and LMA detections during 18-06-13 17:50 view time.
Top: height[m] vs. time[sec.]. Bottom left: LAT/LON [deg.]

Fall of 2018

During this period 4 valuable episodes were recorded, that provided a huge number of detections. The following data only contains sources that were under the LIS FOV at the given time. Since they were observed to be concurrent in space and show a very high activity, only the outputs from **LMA_** vs **LIS_** **comparator.m** will be displayed.

2018-08-09 18:50

In this case, as in the following, a relevant quantity of synchronized sources and events were recorded. Below are printed over time. In the figure 4.12 it can be seen how the activity, despite only occupying 2 min in time, is intense. A minimum of 10 separated flashes were detected by the LMA and virtually all of them had synchronised LIS detections.

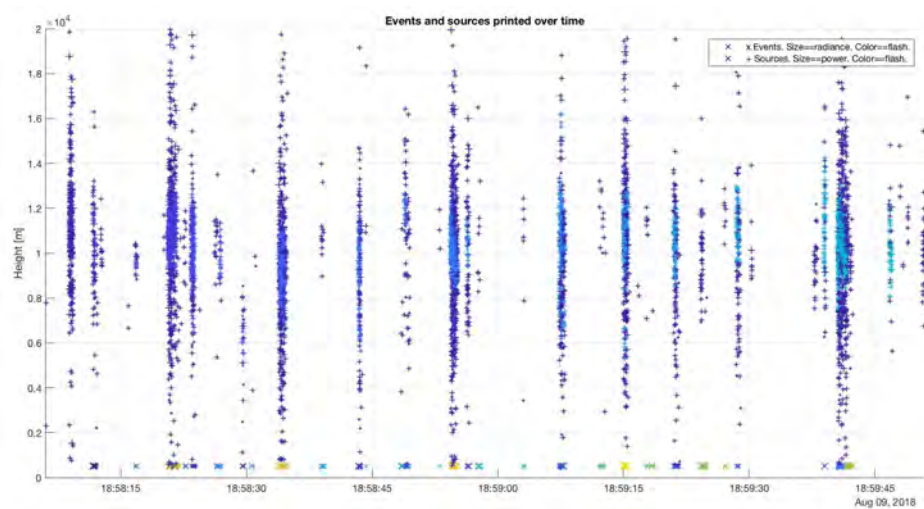


Figure 4.12: Height and time distribution of LIS and LMA detections during 2018-08-09 18:50 view time.

2018-08-09 04:40

In the figure 4.13 it can be seen how an intense activity was detected from both sensors during the LIS pass. Although more homogeneously distributed over time, here discrete lightning discharges can also be appreciated during the whole pass.

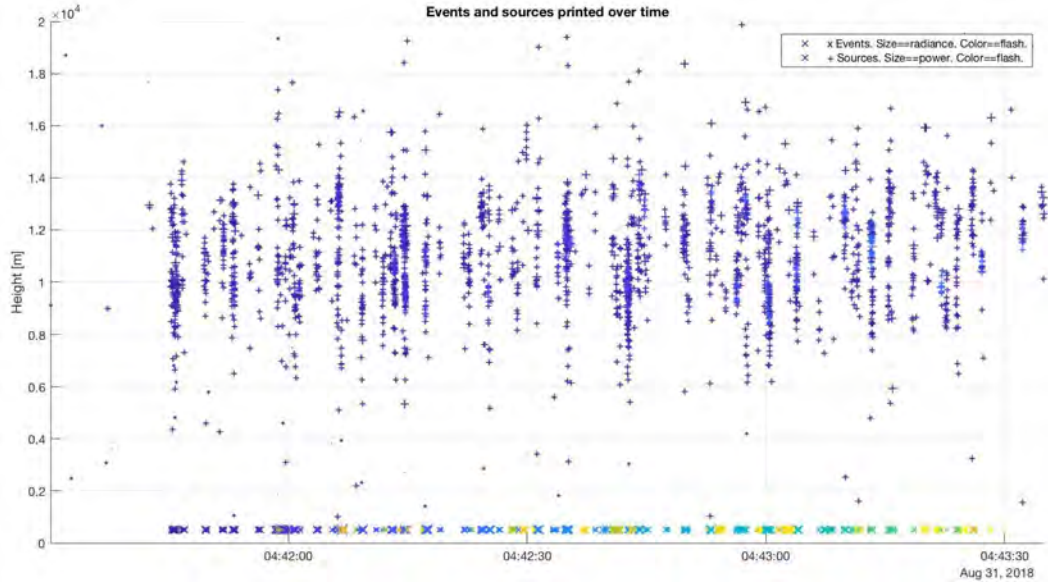


Figure 4.13: Height and time distribution of LIS and LMA detections during 2018-08-09 04:40 view time.

2018-09-18 03:30

In this pass (fig. 4.14) a remarkable intense activity was registered. In the figure it can be seen the extremely dense cluster of LMA detections and their LIS couple.

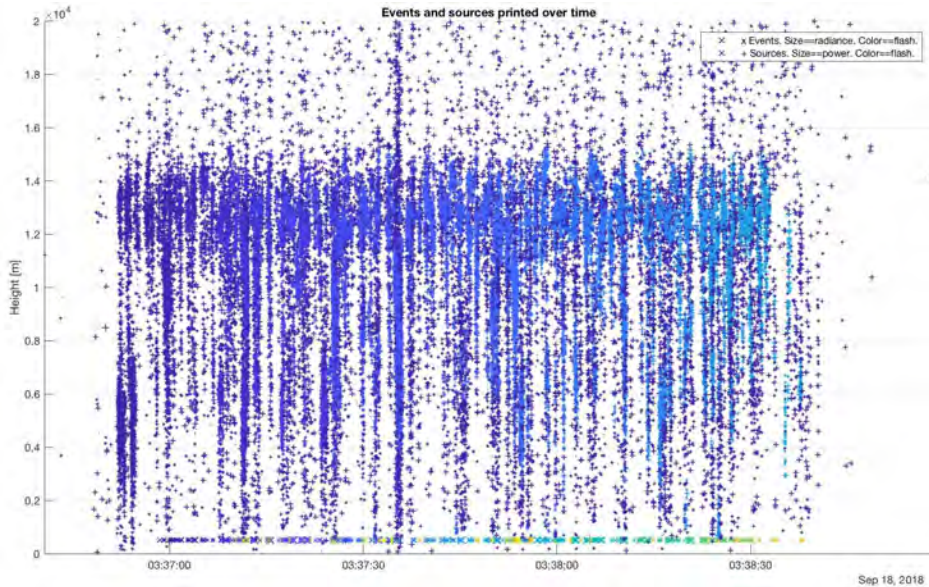


Figure 4.14: Space and time distribution of LIS and LMA detections during 2018-09-18 03:30 view time.

2018-10-18 15:10

In this pass several flashes were detected. They can be appreciated discretely in the fig. 4.15 alongside with their synchronous LIS detections. In this case, at the contrary of the 2018 09 18 one, the lightning are clearly separated in time.

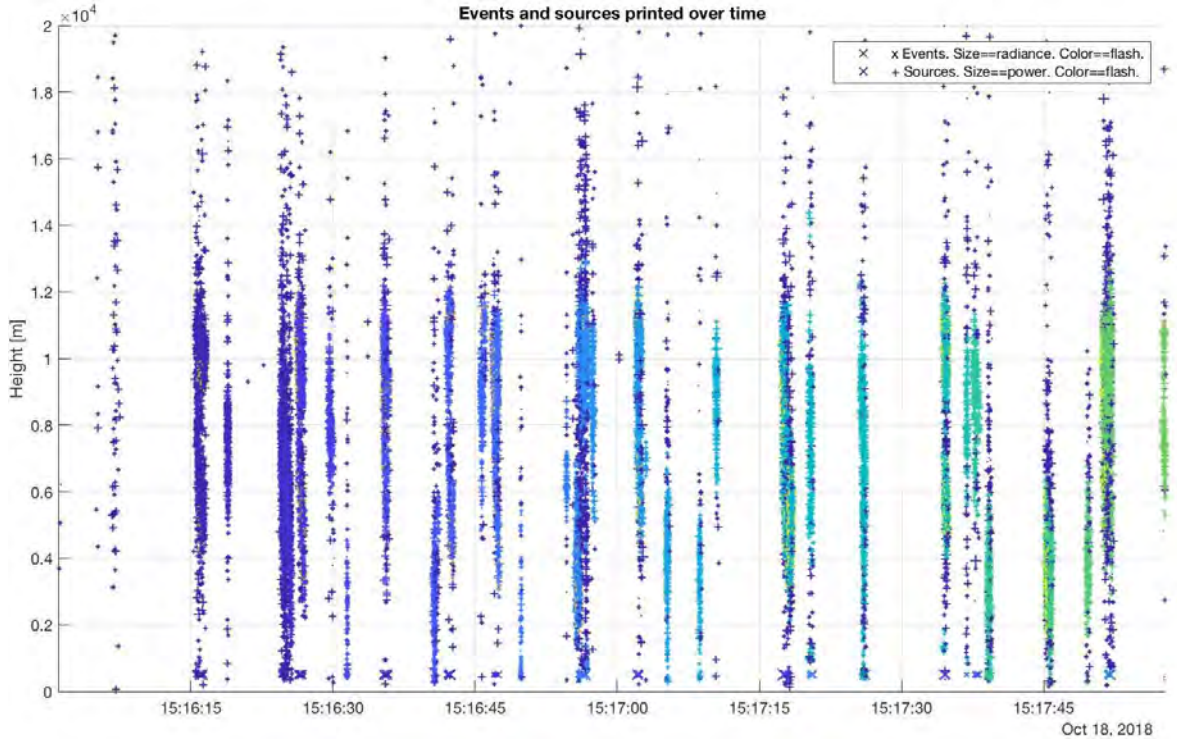


Figure 4.15: Space and time distribution of LIS and LMA detections during 2018-10-18 15:10 view time.

4.1.2 Influence of excited pixels' position on the CCD

The LIS CCD has a surface of 128x128 pixels, and it is possible to made a plot where to represent pixels, given their xy coordinates. To explore if the position of the exited pixels on the CCD has some influence on the three cases with deficient data (18-04-29, 18-05-25 and 18-06-06), the position of said pixels have been printed. They are displayed in the fig. 4.16, and there it can be seen that only few pixels were activated. In 18-04-29 and 18-06-14 cases the excited pixels were in the boundaries of the CCD whereas in the 18-05-25 they were more centred.

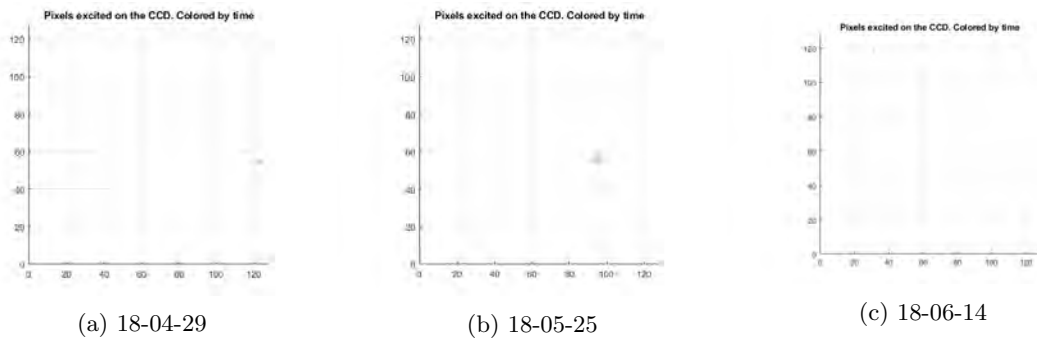


Figure 4.16: Excited pixels distribution on the CCD for bad data cases.

For comparison purposes, also the pixel excitation distribution of some good data has been displayed

(fig. 4.17). It shows the two cases of 17-10-18, from where more data is available. In fig. 4.17 it can be seen how there are excitations both in through the centre of the CCD and in the boundaries.

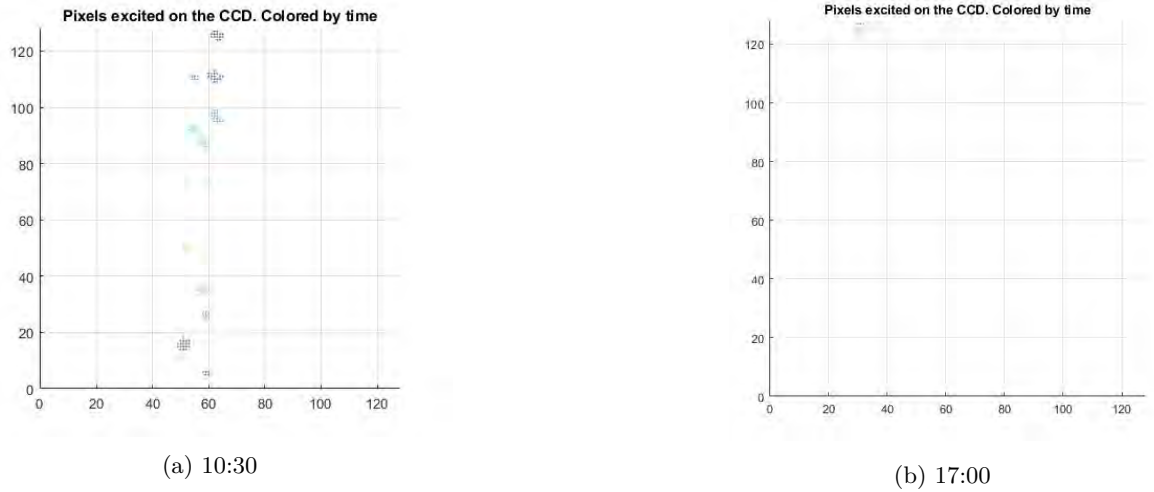


Figure 4.17: Excited pixels distribution on the CCD for two cases with good data: 17-10-18

It is therefore acceptable to sustain that there is no relation between the quality of the gathered data and the position of excited pixels on the LIS CCD.

4.1.3 Section summary

From the section 4.1.1 and 4.1.2 some conclusions arise:

- The periods of more data are the ones of the 2017-10-18 and the cases of the 2018 fall. This is concordant, since in the coast of Catalonia strong thunderstorms occur typically at the summer's end.

Other cases where data is not extremely mediocre are 180427 13:10, 180605 15:10 and 180613 17:50, where although LIS only detected one flash per case, there is a simultaneous LIS detection with low space offset from LMA data. Because they do not provide a high amount of data, compared to the more populated observations, they will not be analysed.

The deficient data cases are of two kinds:

1. In the 181525 01:40 period, LIS clearly detected a flash near the coast, as did LINET; who detected a stroke in the same position at the same time. Nonetheless, LMA did not detect anything so the detected flash (by LIS) is not useful for comparison with VHF presence.
 2. The other two cases LIS and LMA detected activity, but with a very high offset in space. This means that either they detected different phenomena or the signals that they recorded are background noise.
- The validity of the data does not seem to have a clear relation with the relative position of the sensor to the event (and therefore the position of the excited pixels on the CCD). It has been seen how in both good and bad data cases the excited pixels can be on the edge of the CCD or near the centre.

4.2 INFLUENCE OF VHF SOURCES' PROPERTIES ON ITS LIS DETECTIVITY

The scope of this section is to assess what (if any) properties of the VHF RF sources emitted by lightning discharges have an impact on the lightning detection by LIS. Since effects of the excited pixels' position on the LIS CCD have already been discussed, below this will not be discussed again. In this document a statistical analysis of data containing dozens of lightning will be done, an approach that has not been done yet with ISS-LIS. Some works have been done around the matter that concerns this section but only at a scale of studying some flashes –of the order of ten. This section is meant to be a results report, so statistical statements will only regard the used data and they are not intended to be valid about LIS detection worldwide.

4.2.1 Hypothesis and analysis description

The presented data can be analysed from different perspectives. One of these is to check the LIS efficiency to detect lightning discharges. A priori, LMA is a better detector of lightning than LIS (i.e. that it usually detects the discharges that LIS sees and some that LIS doesn't); so to measure the effectiveness of LIS in detecting lightning we can use LMA data as a reliable source of information on what LIS should detect.

Following this criteria, an hypothesis has been made:

The luminosity detected by LIS is part of the same physical process that generates the VHF emissions recorded by LMA, i.e. leader propagating through the air.

Then, with the appropriate time corrections for possible offsets between LIS instrument measuring (TAI93 time) and LMA (UTC time), detections should be roughly simultaneous. It is understood that both detectors may have offsets of microseconds and that the VHF emissions and LIS detections, although part of the same exact physical process, may not be generated at the same exact time. For instance, it is possible that in order for LIS to get the minimum luminosity to activate its threshold a number of near-simultaneous VHF sources must be closely generated. Actually, the data preview seems to support this last suggestion, since there is a much greater of LMA detections than LIS events, which might suggest that the sources must have some characteristics (e.g. minimum power, height ...) in order to be detected. The data will be analysed in order to find those characteristics.

As stated in section 3.1.2 for this analysis a MATLAB code named "LMA_vs_LIS_comparator.m"¹ was made. Basically it divides the time period in time bins, in order to assess the presence of events and/or sources in the mentioned bins. Then, every time bin may or not contain event and sources, that at the same time will have some physical properties associated, for instance the power of the source. Moreover, the program will compute some physical properties of the bins (e.g height-weighted centroid of sources' power, mean height of the sources in the bin...). To do so, it processes text files dumped from the Scilab program, which contain information about sources, flashes, power... displayed in whatever time period the user is displaying. Afterwards, "LMA_vs_LIS_comparator.m" loads events text files of the same period and compare the data.

For instance, the bins will be separated in "bins with only sources", "bins with events and sources" etc. These labels refer to the content of the bin; bins with only events will, for the moment, not been taken into account.

It is appropriate to comment that preparing the data for its analysis with the LMA_vs_LIS_comparator.m program has not been a strait forward task. The final approach has been proposed and implemented by Prof. Montanyà. To put it in few words, the LIS FOV time of a certain area has been extracted from the LIS HDF4 files, and a filter has been set in order to eliminate all the LMA data that did not match the space-time criteria of the LIS sensor. Only the sources that were under the LIS FOV have been used for the analysis.

¹The code is available at at appendix A.6 and <https://github.com/icarfontcu/LIS-LMA-data-reading-codes>.

The size (in time) of the bins that divide the time period is called "timestep". The size of the timestep should be big enough to ignore the possible micro-offset in time and small enough to be able to correctly separate different events detections in different bins. A timestep of 10 ms was selected, after observing with the Scilab program the time distribution of events and sources detections within flashes: there's rarely a gap of more of 10 ms between sources in the same flash, so inside the said flash we shouldn't get empty bins.

In this section the properties of the VHF sources that influence its detectivity with the LIS instrument will be explored. This analysis will be done from an histogram approach. A typical value approach has also been done and can be found in the appendix A.3. It is not presented in the report because it provides less complete information than the histogram approach. For an histogram point of view, only some of the observed periods have enough data to produce significant results. Below are the results of such cases.

4.2.2 Sources' Height Influence on Detectivity

The first source's parameter that has been studied is the source's height. Since LIS is an orbital imager system, one should consider some simple aspects like the atmospheric attenuation of the radiation emitted by lightning or its distortion by the clouds. Such parameters might have an impact on the final detection depending on the source's height since, the higher is the source, the less atmosphere and (probably) clouds will be between it and the detector.

Comment About the State of the Art

The most significant work on the matter can be read on the paper published by Thomas et al. [18]. In this paper are explained the results of a thunderstorm observation with a LMA system at New Mexico. In the only pass of the ISS, 128 lightning discharges were observed. With this data, a relation between height and LIS detectivity is observed: most of lightning that occurred in the higher part of the cloud were detected, not like discharges that happened in lower parts. In the said paper, this is attributed to the higher discharges being closer to the cloud boundaries since in previous work it had been found that light was more likely to escape the cloud if it was originated near the surface.

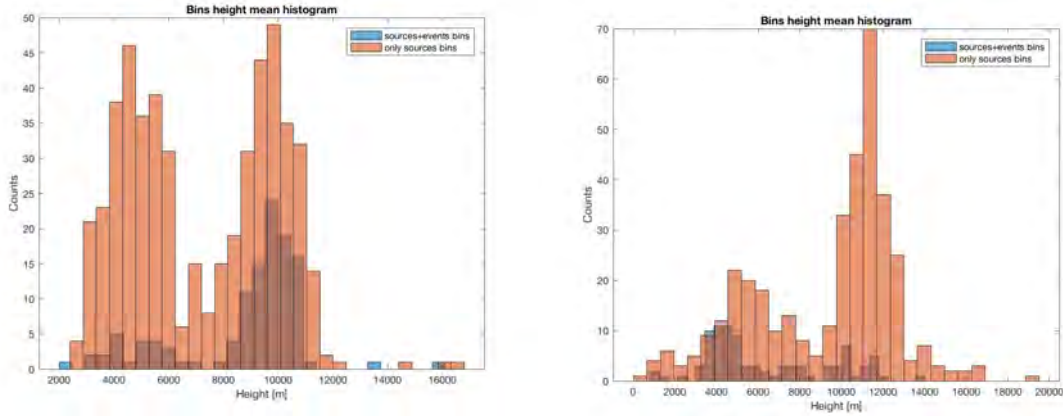
Cases of 2017 fall

The 171018 periods sources distribution are displayed in figure 4.5a. These histograms show the counts of a representative value of a time bin. I.e. the data (sources, events) has been grouped in time bins of 10ms, and a representative value of the elements in each bin has been computed. For instance, the mean heights of the sources within each bin has been assessed, and in the histogram is represented the counts of that last value.

For this case, we can observe a higher distribution of detected sources on the higher spectrum of heights, from 8 to 12km. In the figure 4.18 is represented the main height distribution of the sources of that period. There, it is clear how sources detections are distributed in two groups, the upper and the lower one. One could argue that they could correspond to sources detected in the upper and lower part of a cloud, but this is hardly defensible, as the centre of the "blocks" is separated roughly 6km. Then, it is assertive to state that LMA has detected two groups of flashes –as we have already seen in fig. 4.5a: one with lower flashes and one with higher. What is interesting here is to how the ratio of sources *also* detected by LIS was much higher in high altitude flashes than it was in lower altitude flashes. The reason of that detectivity difference can be a matter of discussion. The median histogram was also made but showed the same results.

If we take a look at the period 171018 17:00 (its detections are displayed in fig. 4.5b) we find quite a different result. In this case, where only one flash was measured, one cannot appreciate any kind of special distribution. Of course, as can be seen in the fig. 4.5b this was a low flash (it stroke the ground) so we cannot compare the detectivity with higher sources. Nonetheless, we can see that *within* a flash it is not obvious that the height matters on the detectivity. This may be because of the difference in heights within a flash is too small to have a real impact in the detection.

From another point of view, following the line of thought of Thomas et al. [18] one could extract from this observation that VHF sources' distances to the closest surface, inside the surveyed flash, were homogeneously distributed, or even more closer in the central region –around 4.6km.



(a) Mean height distribution of sources for the 171018 10:30 period (b) Mean height distribution of sources for the 171018 17:00 period

Figure 4.18: Mean height histograms for the cases of the 2017 fall
The darker red is the same as the blue (indicated in the legend)

All cases

These observations don't seem to hold when looking at the data from a more global scale, with all periods included. The histograms for the median heights of the sources detected during all the periods is displayed in the fig. 4.19 below. There, it can be seen how there is a main difference between the cases of 2017 and those of 2018.

In the first place, in the 2017 cases the cluster of detections can be clearly differentiated in two groups at different heights, while in the 2018 cases the observations approach more to a single-peak distribution. The data of the 171018 17:30 period could be disregarded, since it displays only one flash and that might interfere with the results. Nonetheless, even doing so, the 171018 10:30 period still would present an anomalous distribution of detections in the lower heights. The 181018 period is the other one to present a significant amount of detections in that spectrum of heights but in this particular case a extremely relative greater number of detections were made. On the other hand, one thing that can be observed from the data is that there seems to be a huge decrease of LIS detections of sources for heights lower than 4km, even if there have been detected sources below.

On the other hand, an important aspect can be inferred from this graphs²: all the sources that were also detected by LIS group up around 10km. Even in the cases where there was a higher number of sources detected in lower heights (i.e. 171018 10:30 and 181018) the peak of *sources+events* bins is around the said height. This could reveal that height might have an impact in limiting the LIS detection of sources, for example, it could be extracted that the efficiency in detecting sources decreases with altitude. In fact, this again is concurrent with the Thomas et al. [18] observations of some lightning in south U.S.A.

The measurements shown on figure 4.19 are the medians of the sources. The median has been chosen as the representative value of the height over the mean since it is less affected by outliers. In the lightning parameter evaluation the outliers can be particularly "dangerous", since noise can generate detections far away from where the physical phenomenon is taking place, which will deviate the data significantly. An analogous figure but with means as the representative value can be found in the appendix A.4.

²Disregarding the single-flash data

4.2. INFLUENCE OF VHF SOURCES' PROPERTIES ON ITS LIS DETECTIVITY

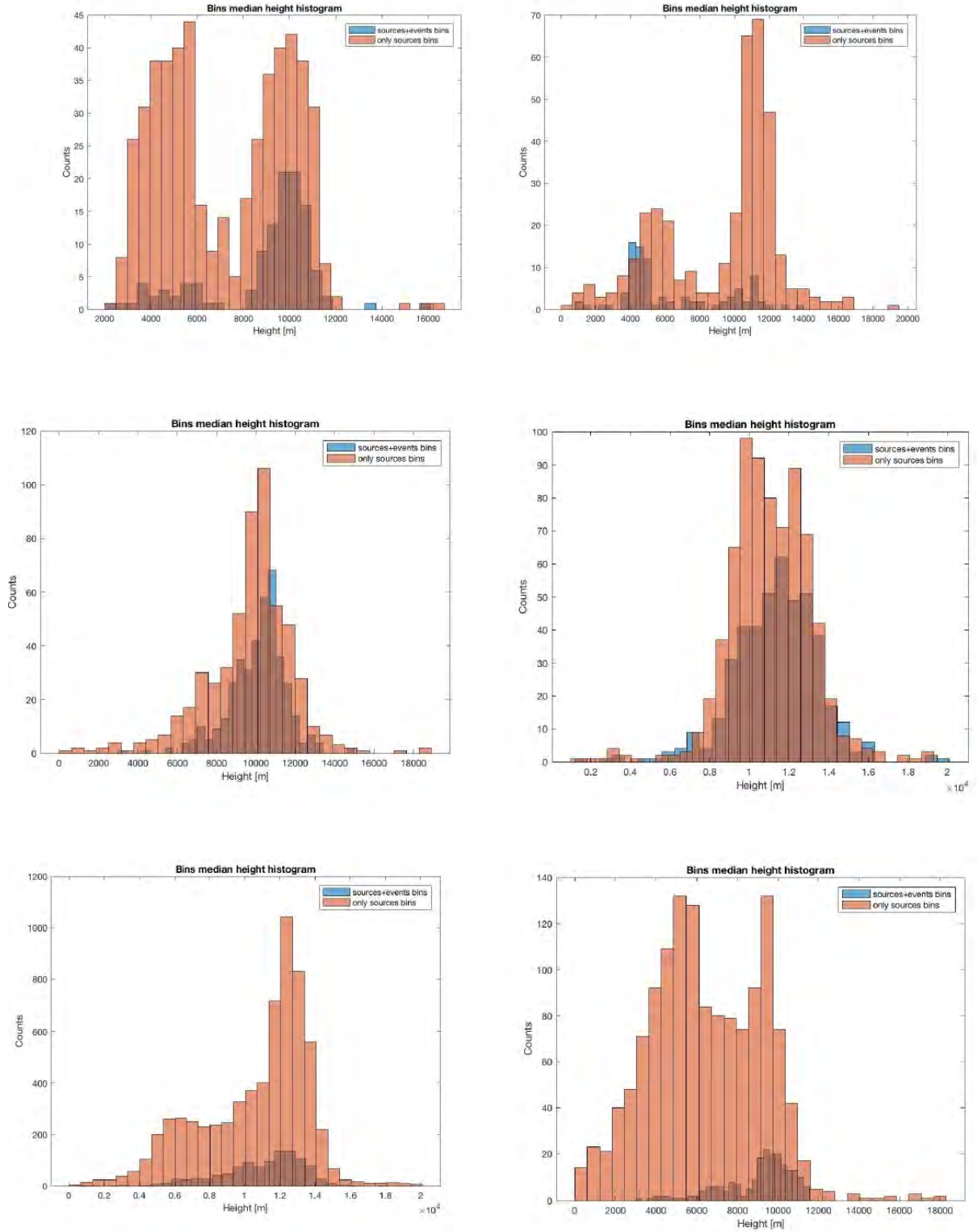


Figure 4.19: Median of Sources Heights Histograms for all periods of data
From left to right, and top to bottom:

171018 10:30
171018 17:00 (single flash)

180809 18:50
180831 04:40

180918 03:30
181018 15:10

Power-weighted average height

In this approach chosen value to represent the height property of a time bin has been the power-weighted centroid. Thus, for a given bin with sources, the average height pondered by the power of each source has been computed. The intention of this approach is to evaluate if there is a trade-off of impact between height and power for a given source. I.e. a higher sources (in terms of height) but weaker might have higher changes of being detected from the LIS.

The said power-weighted centroid height has been assessed applying the following formula in each bin:

$$\frac{\sum_{i=1}^{n_{srcs}} (pwr_i * height_i)}{\sum_{i=1}^{n_{srcs}} pwr_i}$$

Please be aware that for this operation the power has been introduced in [W] units. dbW units would produce erratic results.

The results of this approach are displayed in the figure 4.21 below.

It is remarkable the similarity of the measurements displayed in the refereed figure with the ones regarding the *median of heights* (fig. 4.19). Although some minor displacements of the distributions' peaks can be observed, they stay virtually in the same position both in absolute terms (in height and counts) and in relative, when taking a look if the *sources+events* distributions have shifted in respect to the *only sources* distributions. The absence of change when introducing a power weighting may account the possibility that each bin –then, each cluster of sources– have a symmetrical distribution of powers in respect to the median, an example shown in the figure 4.20. Thus, this approach does not provide extra information on how the altitude might affect LIS detectivity.

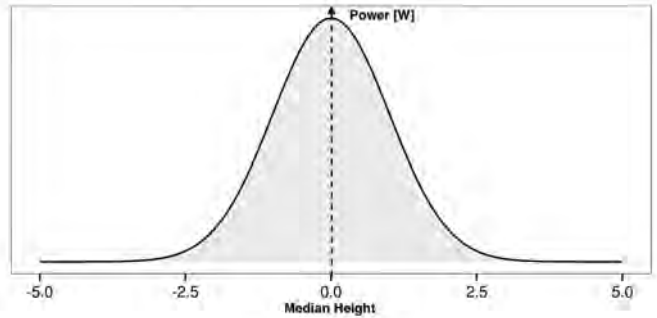


Figure 4.20: Example for symmetrical distribution of power around the heights' median inside a bin

In the height case, the median has been taken as a representative value for each bin, which has not shown any remarkable tendency. With the aim to account the possible effect of a source power to influence on the detection, also a weighing of the height median values has been done, using the power of each source as a factor. This latter approach has shown similar distributions to the without-weighting approach. And thus not providing any valuable result.

4.2. INFLUENCE OF VHF SOURCES' PROPERTIES ON ITS LIS DETECTIVITY

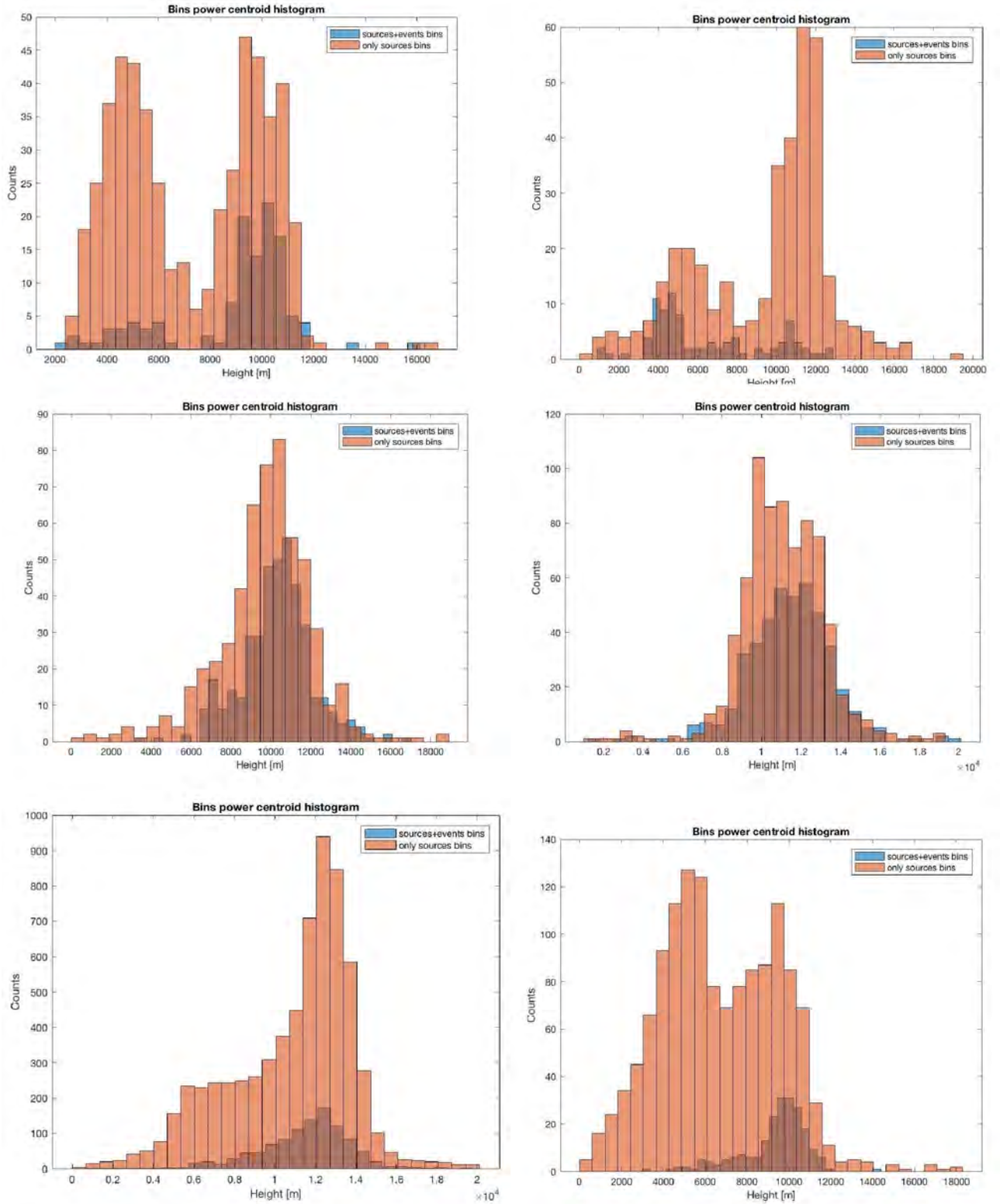


Figure 4.21: Power-weighted average sources' height
From left to right, and top to bottom:

171018 10:30
171018 17:00 (single flash)

180809 18:50
180831 04:40

180918 03:30
181018 15:10

4.2.3 Sources' Maximum Power influence on Detectivity

In the previous section it has been explored the influence of lightning height on their LIS detectivity. Nonetheless, some other parameters of the sources arise as possible influencers on the matter. In this approach, a representative value for each bin related to the sources' power has been computed, in order to explore if a correlation can be made with the LIS observations. As a representative value, the maximum power of each bin has been taken.

In the figure 4.22 are displayed the histograms for the distribution of maximum power between time bins. I.e. the data has been grouped in time bins of 10ms and then the maximum power of each bin has been assessed. The histograms show the counts for these values. Some interesting aspects of the data can be observed:³

In the first place, there is an apparent shift of the *events+sources* bin counts peak to the right of the *only sources* bin counts; only reversed for the single-flash case (171018 17:00). This characteristic could be very remarkable. A shift to the right of the *sources+events* peak may mean that LIS is more effective in detecting sources, or group of sources, with higher maximum power emissions. Thus, it seems that there could be a link between the power of a VHF emission and the luminosity of the discharge that generates it.

Secondly, it is worth observing that the bulk of *sources+events* detections are usually centred around 20-25 dBW of power. Nonetheless, it does not seem to be any relation between the position of this center and the amount of LIS detections, in absolute term or in relation to the LMA detections, for a given period. For instance, in the 180831 period the *sources+events* bin counts sum up to a value of the order of 60, and the *only sources* bins up to 100. On the other hand, for the case of and 18018 there is a higher number of *only sources detections* (maximum at around 120 counts) but a lower of *sources+events* counts (maximum at around 20 counts).

Finally, it seems to be a lower limit on the bins max. power for its LIS detection, at around 10 dBW. If luminosity is linked to the VHF power it may be relevant to determine the relation between them and what is the minimum luminosity at the origin that allows a detection from orbit.

The influence of the maximum power of a sources cluster in its LIS detectivity has also been studied. In this approach it has been seen for the first time how the power of the VHF RF sources might have an impact in its detection by LIS: the bins where also events were detected usually had a higher maximum power recorded.

³In the figures appear negative power values. Please, be aware that the units are in Decibel - Watts, so negative values have an origin at a 0.X power [W] value

4.2. INFLUENCE OF VHF SOURCES' PROPERTIES ON ITS LIS DETECTIVITY

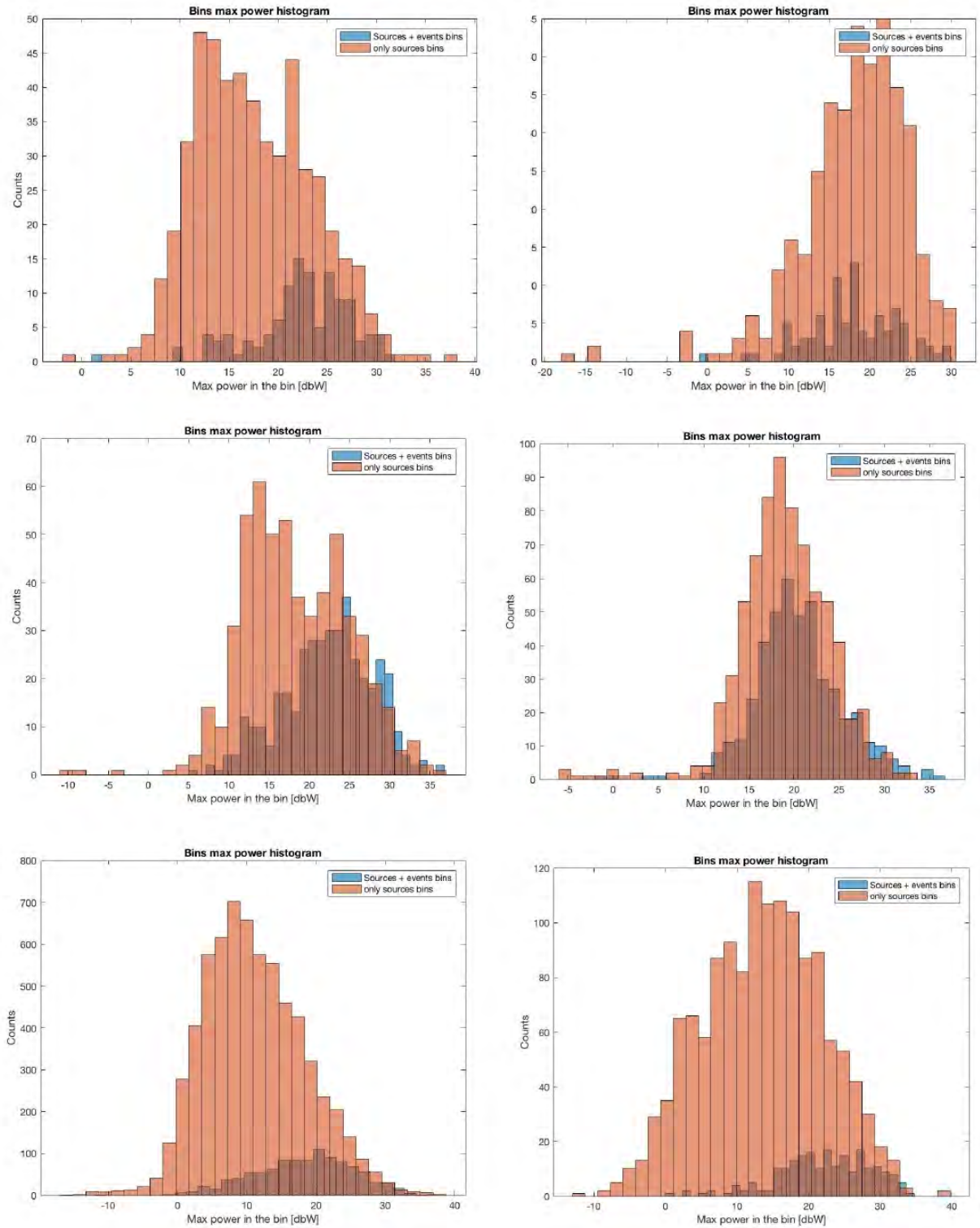


Figure 4.22: Maximum power histograms for all periods of data
From left to right, and top to bottom:

171018 10:30
171018 17:00 (single flash)

180809 18:50
180831 04:40

180918 03:30
181018 15:10

4.2.4 Density of sources in the time bins

Finally, another parameter that has been studied is the density of sources in time bins, and how it may affect the LIS detectivity. From a first approach, the relation might account for the fact that considering that a sources in the same cluster occur near simultaneously, the 777.4 nm radiation that LIS detect may sum up, thus increasing the radiation received by the CCD and therefore having a greater chance to surpass the threshold and trigger the detection.

The histograms regarding the distribution of sources densities are displayed in the figure 4.23 below. An important remark for this analysis is that the step time for each bin has been diminished to 2 ms. The reason for that is that the integration time on the LIS CCD is 2 ms and, by setting the same step, we increase the validity of affirming that a cluster –i.e. sources in a same time bin– will represent a single LIS detection and therefore the sum of radiance has a significant meaning.

In the fig. 4.23 it is clear that both *sources+events* and *only sources* low-density bins are more numerous than their counterparts. A steady increase in both cases can be observed when going from high density to low density, with the exception of the cases 180918 and 181018, where the presence of *sources+events* bins seems quite equally distributed on the lower density spectrum.

The said characteristic of the distributions becomes relevant when comparing the increase rate of the *sources+events* density counts vs. the *only sources* counts. In the figures we can see how lower-density clusters are less likely to be detected by LIS: the while in high density bins the proportion of *s+e* bins to *only s* bins is very similar, at lower densities the number of *only s* detections increase exponentially, while the *s+e* bins do so with a slower rate.

Also, there might a link with the total power expelled from a cluster: the more dense, the more power generated. To further study the density aspect but also the link with the power produced in a cluster, the distributions of the sum of the power for each bin have been produced. They are displayed in the figure 4.24. There, it can be seen how in all cases the distribution peak for the *s+e* bins is either shifted to higher power or remains parallel to the *only sources* distribution peak. This is very significant: in the studied cases, containing dozens of lightning discharges, LIS has been more likely to detect clusters of VHF RF sources that had a higher overall power, that usually has been correlated with more populated clusters. I.e. the difference in detectivity has not been produced by sources with very high power, but rather by the presence of a greater number of sources.

The density of sources in the bins has also been studied, and it has shown how both only sources and sources+events bins are more numerous in the low-density spectrum. Nonetheless, it has been observed how the increase rate is not the same, and the proportion of bins detected by LIS in relation to the not-detected diminishes in the low-density spectrum. Thus, the data shows that LIS is more likely to detect high populated source clusters. Finally, the distributions of sum of power for each bin have been studied and have shown that LIS has been more likely to detect VHF source clusters with a higher sum of power. In all the cases recorded, the peak of the sources+events bins detected is centered at the same or higher power than the only sources bins detection distribution.

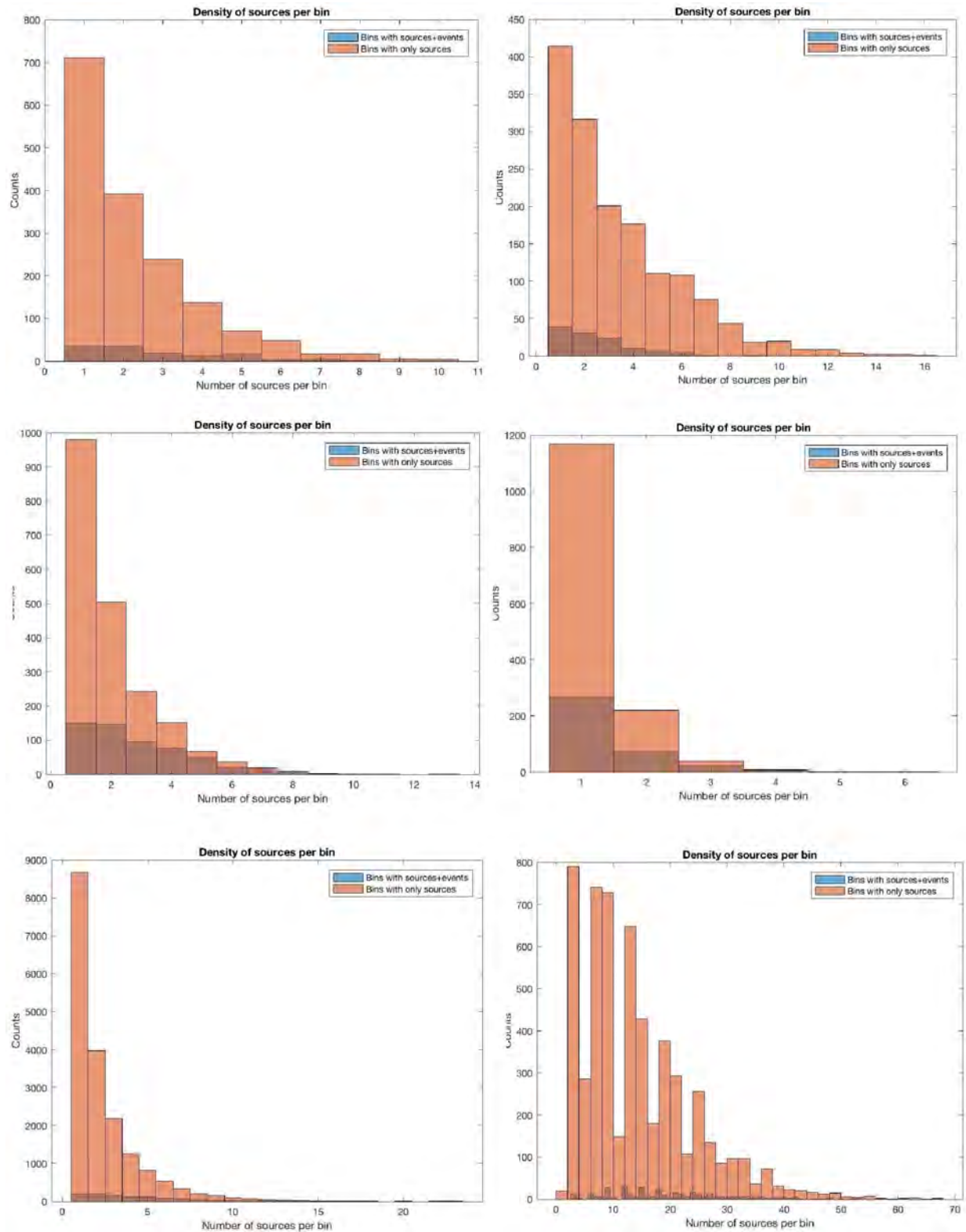


Figure 4.23: Distribution of Density of sources in a bin for all periods
From left to right, and top to bottom:

171018 10:30
171018 17:00 (single flash)

180809 18:50
180831 04:40

180918 03:30
181018 15:10

4.2. INFLUENCE OF VHF SOURCES' PROPERTIES ON ITS LIS DETECTIVITY

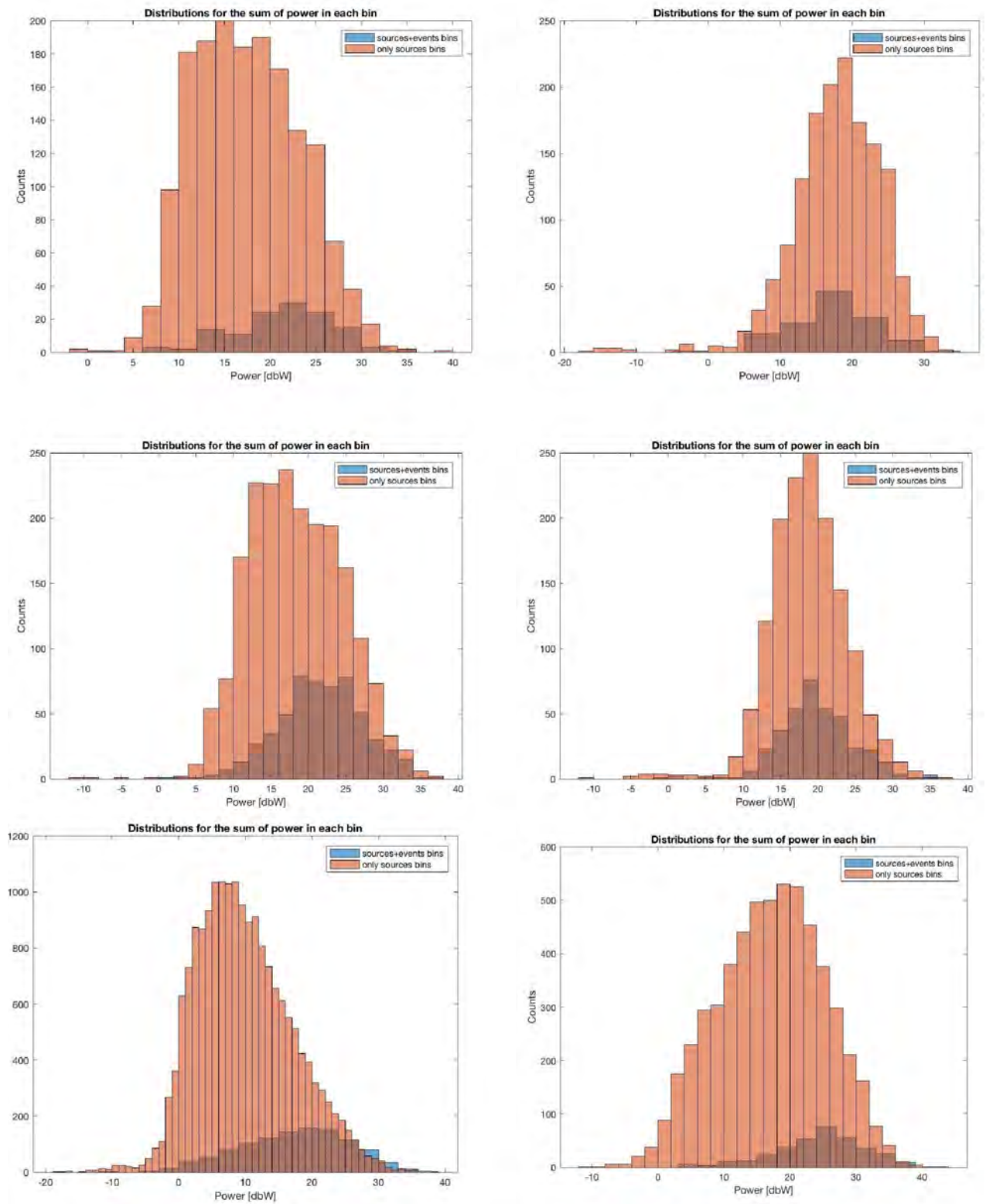


Figure 4.24: Distributions of the sum of power for all periods
From left to right, and top to bottom:

171018 10:30
171018 17:00 (single flash)

180809 18:50
180831 04:40

180918 03:30
181018 15:10

4.2.5 Section Summary

In this section the Height, Power and Density (and cross-overs) of sources in the time bins have been studied in order to see their influence on its LIS detectivity.

- In the height case, the median has been taken as a representative value for each bin, which has not shown any remarkable tendency. With the aim to account the possible effect of a source power to influence on the detection, also a weighing of the height median values has been done, using the power of each source as a factor. This latter approach has shown similar distributions to the without-weighting approach. And thus not providing any valuable result.
- The influence of the maximum power of a sources cluster in its LIS detectivity has also been studied. In this approach it has been seen for the first time how the power of the VHF RF sources might have an impact in its detection by LIS: the bins where also events were detected usually had a higher maximum power recorded.
- The density of sources in the bins has also been studied, it has shown how both *only sources* and *sources+events* bins are more numerous in the low-density spectrum. Nonetheless, it has been observed how the increase rate is not the same, and the proportion of bins detected by LIS in relation to the not-detected diminishes in the low-density spectrum. Thus, the data shows that LIS is more likely to detect high populated source clusters.
- Finally, the distributions of sum of power for each bin have been studied and have shown that LIS has been more likely to detect VHF source clusters with a higher sum of power. In all the cases recorded, the peak of the *sources+events* bins detected is centered at the same or at higher power than the *only sources* bins detection distribution.

Summing up the last two points above, an effect of the sources' power into its LIS detection has been recorded. It has been observed how source clusters more populated –and therefore more likely to have a higher total power– have been more likely detected by LIS than their counterparts.

4.3 ANALYSIS OF THE FLASH DURATION CONCORDANCE BETWEEN LIS AND LMA SENSORS

Each sensor measures electromagnetic waves coming from lightning, and more than just providing information about that phenomena, it is interesting to extract information about the whole flash. The scope of this section is to compare the information that the user would extract about the flash from both sensors. For instance, the flash duration or its size is something that a post processing of the events'/sources' time and position could provide. Nonetheless, if for example one would compute the lightning duration from LIS and LMA by saying:

$$flash_duration = \max(events.time) - \min(events.time)$$

or

$$flash_duration = \max(sources.time) - \min(sources.time)$$

Where "events.time" and "sources.time" are the vectors of events/times of each flash.

It is clear that different results could be extracted for the same flash, as LIS could detect events only in the middle of the real flash and so on.

4.3.1 Flash Duration Analysis

Comparing flash durations (with the previously proposed method) has not been trivial. Even if in the sources' txt files⁴ it is specified the "seconds-in-flash" of each source (so we have already the flash duration) and the flash number; the comparison process is not as simple as it may seem. This is because

1. in both txt files (LIS and LMA) the flashes are not numbered following the same "coordinates".
2. a simple time comparison can not be made as different start/end times in the two files could relate to the same flash

Therefore, in order to be able to compare the flash durations provided by both sensors, an extension of the LMA-vs-LIS-comparator.m has been made. This extension creates vectors with the start/end times computed from both files; and then compares them by considering that a LMA flash and a LIS file are the same if:

1. at least the start time of the LIS flash is comprehended in time in the LMA flash
2. at least the end time of the LIS flash is comprehended in time in the LMA flash
3. both ends in time of the LMA flash are comprehended in the LIS flash

Please note that the case were the LIS flash is comprehended inside the LMA flash is implemented in one of the 2 first options.

2017-10-18 10:30

In the studied data, this is the best period to check if the comparator works, as is the only one with several flashes in it. the LMA-vs-LIS-comparator.m displays the following (fig. 4.25):

In this time period, the LMA⁵ detected 23 different flashes, whereas the LIS detected 17. This can be attributed, for example, to the various flashes that LIS did not detect. In the fig. 4.25 can be seen how approximately at 10:33:37, 10:34:03, 10:34:8, 10:34:16, 10:34:25 and 10:34:29 there are LMA flashes that were not detected by LIS⁶. There are also some flashes that LIS detected but LMA did not. This

⁴As previously said, they are extracted from the Scilab program for a given 10 min period

⁵Its criteria is established in LMA-zoom7.sci and it follows a simple imposition of the maximum amount of time between two subsequent sources; if they are more separated than the established time they will be considered from two different flashes.

⁶Please note how the LMA criteria is considering to be the same flash the first two clusters near the 10:32:45.

4.3. ANALYSIS OF THE FLASH DURATION CONCORDANCE BETWEEN LIS AND LMA SENSORS

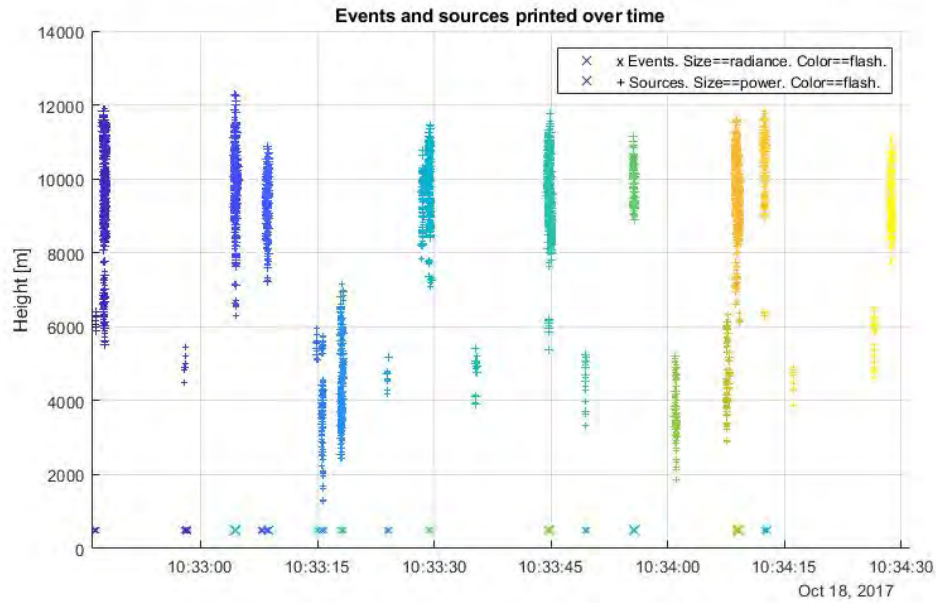


Figure 4.25: Sources (+) and events (x) detected during the 171018 1030 time period printed over time, coloured by flash.

does not mean that LIS detected more real flashes than LMA but it can be attributed to the different (probably more restricted) LIS criteria when it comes to separate various detections in different flashes.

In this time period, under a closer look to the events-sources distribution the problem about flash duration arises, as LIS and LMA detections in a flash have a very different span in time. A clear example of that is shown on the figure 4.26.

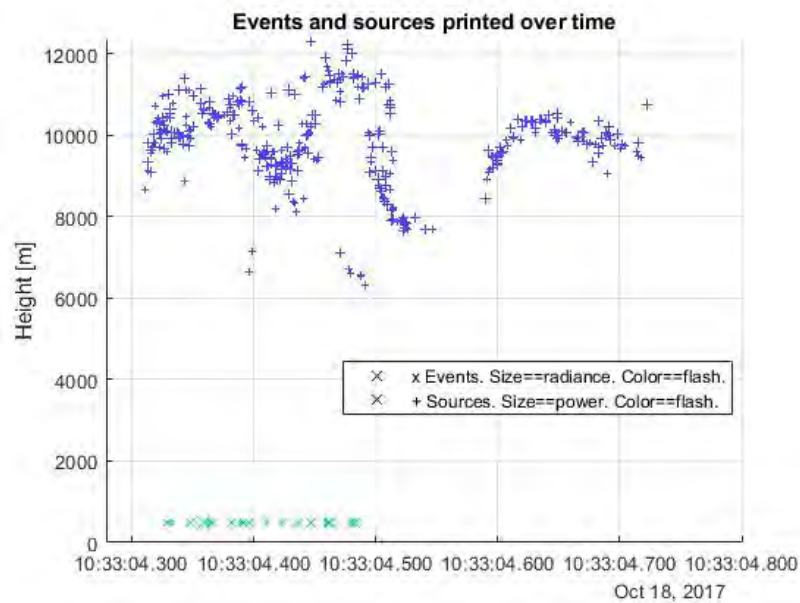


Figure 4.26: Zoom on a flash of the 171018 1030 time period. Sources (+) and events (x) printed over time.

4.3. ANALYSIS OF THE FLASH DURATION CONCORDANCE BETWEEN LIS AND LMA SENSORS

The results of computing the flash duration from data of both sensors are displayed in table 4.1 below.

LIS flash#	LMA flash#	LIS flash duration	LMA flash duration
295	0	0	0
296	31	0.4799	0.0753
297	0	0	0
298	0	0	0
299	33	0.3913	0.3516
300	33	0.0734	0.3516
301	35	0.3010	0.1796
302	37	0.0990	0.1307
303	42	0.2241	0.1971
304	43	0.2487	0.2754
305	48	0.1809	0.3528
330	32	0.1543	0.4110
331	34	0.1487	0.0736
332	36	0.3598	0.3711
333	39	0.1503	0.2809
334	41	0.3312	0.4899
335	47	0.5030	0.5481

Table 4.1: Duration of LMA and LIS flashes (seconds). In the table there are not the flashes that were not detected by LIS. The rows with zeroes represent those flashes that LIS detected but not LMA.

For the cases with complete data (2 simultaneous detections of flashes) the subtraction of LIS durations to LMA durations is shown in the following set of data:

-0.4046	0.2782	0.0317	0.0267	0.2567	0.0114	0.1587
-0.0397	-0.1214	-0.0271	0.1719	-0.0751	0.1307	0.0451

Set of data extracted from table 4.1

The typical values of this set are:

$$mean = 0.0261 \quad median = 0.014$$

Therefore, at least in this set of data, LIS tends to provide results where lightning appear to be shorter than they really are. Although as it can be seen in the data this is not an absolute statement, as there are cases where the duration values for LIS are higher than the ones from LMA.

This result is not very surprising, because it is know that LMA detects a higher number of events⁷ than LIS, so a higher probability of finding earlier and later phenomena of a flash seems reasonable.

Also, the median and mean of the flash durations are:

$$LIS : \quad mean = 0.2144 \quad median = 0.1809$$

$$LMA : \quad mean = 0.2405 \quad median = 0.2754$$

which is in the other of 10 times bigger than the median/mean value of the subtraction. This is quite surprising, because in the example (shown in fig. 4.26) the LIS flash is way shorter in time than the LMA one, at the order of a half. Despite being surprising, this is a positive result, because it shows how, at least in this set of data, typically the sensor does not have a very high influence on the typical lightning duration.

⁷Not in the sense of sources/events notation, but referring to the fact that it detects an event as as a generalisation of "EM pulse".

4.3.2 Section summary

In this section the issue for lightning duration with LIS and LMA has been addressed, by comparing simultaneous flashes between them. For the computation of the flash duration, the time of the earliest and latest event/source of a flash (coming from respective txt files) have been subtracted. Afterwards, typical values of duration and subtraction have been computed in order to see the influence of the sensor in the calculus. It has been found that

1. Typically LMA flashes last more than the LIS flashes, if the said criteria when computing the duration is used.
2. The typical value of the difference in time between LMA flashes and LIS flashes is 10 times smaller than the typical duration of respective flashes, so the influence of the sensor over the computed typical duration is rather small.

Chapter 5

Electric arc emissions for LIS calibration

Contents

5.1	Minimum Radiance Emissions	55
5.2	Relation peak voltage – radiance emitted	58
5.2.1	Direct relation	58
5.2.2	Other approaches	59

Although LMA can be very useful for calibration of orbit lightning sensors, other methods can be used. One of those can be the creation of an electric arc discharge near the ground, such that produces radiance emissions observable by LIS. This technique would clearly increase the precision of the calibration since the position of the light source would be completely known. In order to do this project, three main tackles should be overcome:

- Setting the minimum radiance emission at the source in order to make it detectable by LIS
- Establishing the relation between the peak voltage for the electric arc generation and the radiance emission at the 777.4 nm line.
- Constructing the system that enables an effective discharge

This chapter is meant to be a report on the bibliographical research about the two first stages, being the second a topic that is far from trivial in plasma physics.

5.1 Minimum Radiance Emissions

The location of the discharge generator has been supposed to be the station at El Niu de l'Àliga, situated in the mid-longitude Pyrenees of north Catalonia. It has also been supposed that the experiment would be done in a Summer night. The data regarding the weather during such periods has been extracted from [20].

The web application for MODTRAN (<http://modtran.spectral.com/>) has been used to model the atmosphere. The solver has worked with the following parameters set:

- Atmosphere Model: Mid-Latitude Summer
- Ground Temperature as average temperature of Summer months (Jun-Sep): 283.01 K
- Sensor altitude: 99 km (maximum)
- Sensor Zenith: 180° (the sensor is on top of the discharge)
- Spectral range: 0.7-0.8 μm
- Resolution: 0.00192 μm (maximum)

- Water Column (atm-cm): 3635.9
- Ozone Column (atm-cm): 0.33176
- CO2 (ppmv): 400
- CO (ppmv): 0.15
- CH4 (ppmv): 1.8
- Ground Albedo: 0
- Aerosol Model: rural
- Visibility: 23 km

In the figure 5.1 are displayed the radiance and radioactive flux registered in the conditions above. In the figure figure 5.2 below, the transmittance for each wavelength on the same conditions is displayed. It is remarkable to notice how for the wavelength of the radiation emitted by the atomic oxygen the transmittance is of approximately 0.79; where

$$T = \frac{I_{w\lambda}}{I_{0w\lambda}} \quad (5.1)$$

$$\text{where } [I] = \frac{\mu J}{sr^2 m^2 \mu m}$$

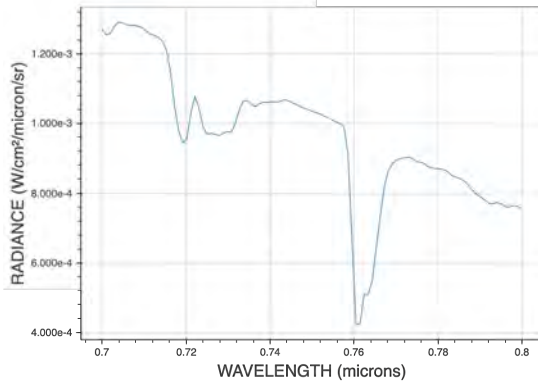
In order to set a minimum radiance emission that enables its detection from orbit, at a height of approx. 2500m a selection of the events that had sources associated at that height has been done. Then, a representative value of these events' radiance has been extracted. The values obtained are below:

Period	171018 10:30	171018 17:00	180809	180831 (t = 200m)	180918	181018
avg	18.23	14.48	18.07	10.5	15.76	16.75
m	14	12.5	14	10.5	12	13.5

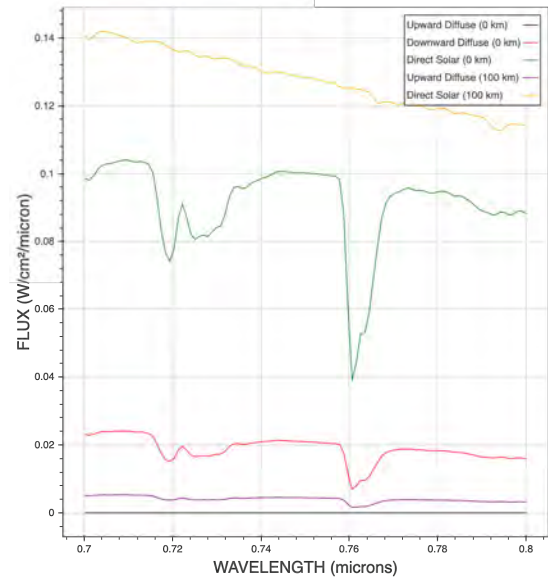
Table 5.1: Typical radiance values of events with sources associated at 2500m, with a tolerance of 50m in $[\frac{\mu J}{sr^2 m^2 \mu m}]$

For the period 181031, where no sources took place at 2500m with t =50. Results with a t=200m are displayed instead.

In the table 5.1 it can be seen how the the average and the median differ significantly, showing a high variance since the averaging is greatly influenced by outliers. Also, note that for the period of 1810831 no sources were detected withing the tolerance (2500m \pm 50m), so the latter was augmented to 200m. Still, the data shows a high deviation from the other cases so it will not be considered.



(a) Atmospheric Radiance depending on the wavelength by MODTRAN



(b) Atmospheric radiation Flux depending on the wavelength by MODTRAN

Figure 5.1: MODTRAN output Flux and Radiance

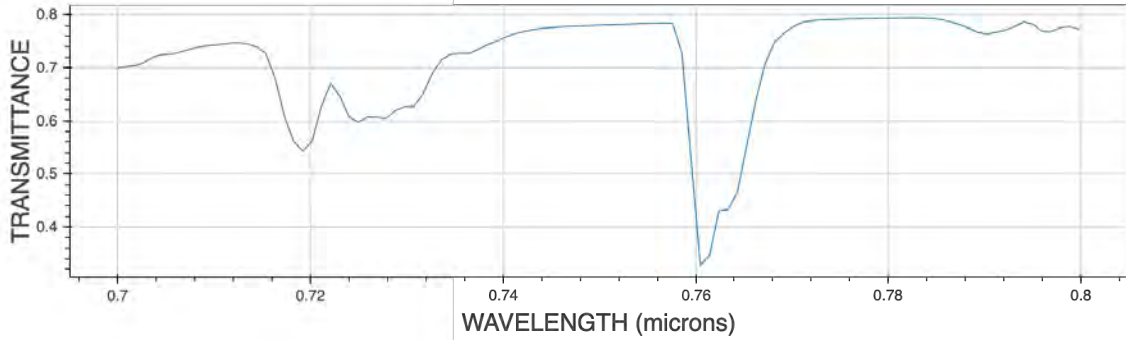


Figure 5.2: Atmospheric transmittance depending on the wavelength by MODTRAN

For the other cases, an average of the medians will be taken as a representative value, since the medians only show a standard deviation of only 0.9083 between them. Thus, an acceptable representative value is:

$$I_{w\lambda} = 13.2 \frac{\mu J}{sr^2 m^2 \mu m}$$

Thus, using the data provided by MODTRAN (see fig. 5.2) the radiance emitted at the source¹ should be, approximately:

$$I_{source} = \frac{I_{sensor}}{T_{\lambda=777.4nm}} = 16.71 \frac{\mu J}{sr^2 m^2 \mu m}$$

¹Not as "VHF source" but as "the origin of the emission"

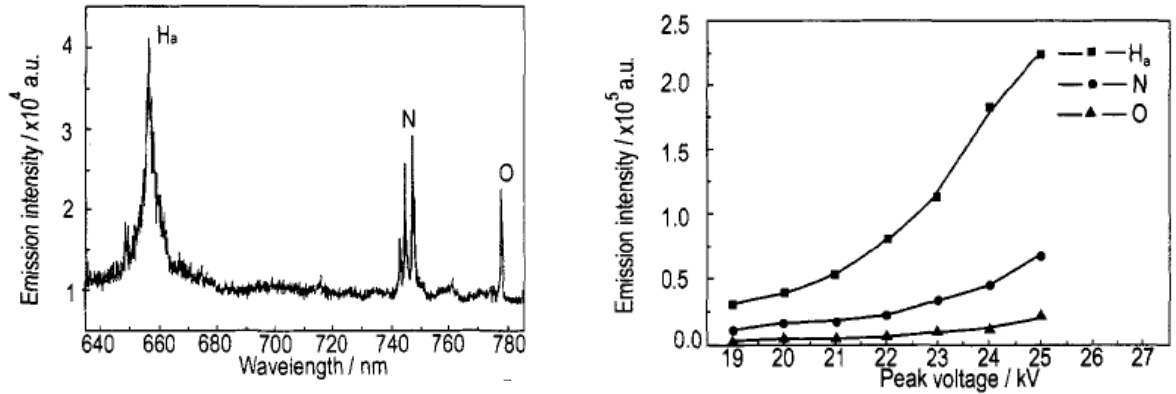
5.2 Relation peak voltage – radiance emitted

5.2.1 Direct relation

Some experiments have been done which give a relation between both parameters. A significant work is described in Liu Feng et al 2005 [19]. Below is cited the abstract of the paper:

In this study, the emission spectra of active atoms O ($3p^5P \rightarrow 3s5S_2^0$ 777.4 nm), H, ($3P + 2s$ 656.3 nm) and N ($3p^4P \rightarrow 3s^4S_2^0$ 742.3 nm, 744.2 nm, 746.8 nm) produced by the positive high-voltage pulsed corona discharge (HVPCD) of N_2 and H_2O mixture in a needle-plate reactor have successfully been recorded against a severe electromagnetic interference coming from the HVPCD at one atmosphere. The effects of the peak voltage, the repetition rate of pulsed discharge and the flow rate of oxygen on the production of those active atoms are investigated. It is found that when the peak voltage and the repetition rate of the pulsed discharge are increased, the emission intensities of those active atoms rise correspondingly. And the emission intensities of O ($3p^5P \rightarrow 3s5S_2^0$ 777.4 nm), H, ($3P \rightarrow 2s$ 656.3nm) and N ($3p^4P + 3^4S^0$ 742.3 nm, 744.2 nm, 746.8 nm) increase with the flow rate of oxygen (from 0 to 25 ml/min) and achieve a maximum value at a flow rate of 25 ml/min. When the flow rate of oxygen is increased further, the emission intensities of those atoms visibly decrease correspondingly.

For these experiments, the spacing between the electrodes was 80 mm and the discharge was done inside a stainless steel chamber with the aim to control the atmospheric conditions described in the abstract. The figures below display some of the measurements done:

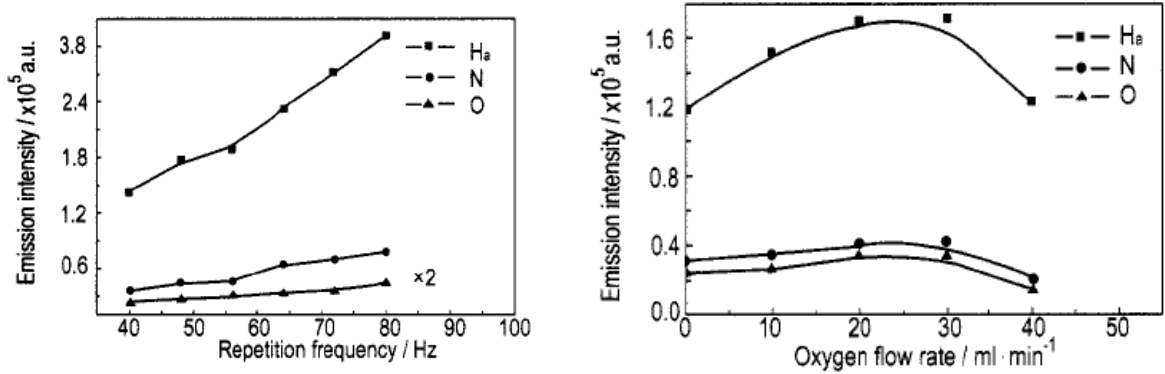


(a) Typical emission spectrum O, H_α and N generated by pulsed corona discharge in a N_2 and H_2O mixture at 24 kV peak voltage and 72 Hz repetition rate of pulsed discharge

(b) Emission intensities of O, H_α and N active atoms as a function of the peak voltage at one atmosphere and 72 Hz repetition rate of pulsed discharge

Figure 5.3: Emission properties of controlled electric arc discharges. Extracted from [19]

In the figure 5.3a it can be seen how they actually registered the well known peak for atomic oxygen radiation at 777.4 nm of wavelength. The really interesting measurements are displayed in the fig. 5.3b, where an evolution of emission intensity with the peak voltage can be observed. The figures 5.4a and 5.4b (also extracted from [19]) provide valuable information regarding the evolution emission of radiance with other parameters of the experiment: the repetition frequency for the discharge and the oxygen flow rate to the environment.



(a) Emission intensities of O, *H_α* and N active atoms as a function of the repetition rate of pulsed discharge at one atmosphere and 24 kV peak voltage

(b) Emission intensities of O, *H_α* and N active atoms as a function of the O₂ flow rate at 25 kV peak voltage and the 56 Hz repetition rate of pulsed discharge

Figure 5.4: Emission properties of controlled electric arc discharges (second). Extracted from [19]

It is understood that the discharge generator, for from-orbit observation, should generate the maximum amount of emitted atomic oxygen radiation. Figures 5.3b and 5.4a suggest the usage of maximum repetition frequency and peak voltage available. On the other hand, from figure 5.4b can be inferred that an optimisation of the emitted intensity with the oxygen flow rate should be sought.

Information regarding a link between current in an electric arc and the radiance emitted has been sought in the following search engines/databases:

- Google Scholar
- Science Direct

With the keywords: *electric arc, current, voltage, radiance, emitted, plasma, atomic oxygen, air, spectroscopy*. With, among others, the following combinations:

- electric arc current emission intensity
- atomic oxygen emission arc current open air
- electric arc open air spectroscopy
- atomic oxygen radiance emission current electric arc
- atomic oxygen radiance electric arc current
- electric arc radiation relation voltage
- atomic oxygen electrical arc open air

Also were revised the 21 citations of the article (André, P. et al. 1997, [21]) and no information regarding a link between the voltage and the radiance of an electrical arc in open air was found. Probably, it is due to the fact that the radiance of a plasma depends on its temperature and state (thermal and chemical equilibrium, ionization degree, composition...); which are parameters difficult to relate to a current without taking into consideration a prepared atmosphere etc.

5.2.2 Other approaches

Intensity of emission vs. temperature of the plasma

Other works have related not directly the current or voltage applied to the atmosphere to generate the electric arc with the emitted intensity, but rather other parameters such as the temperature of the plasma. An example of this approach is presented by André, P. et al. in [21], where a method to

compute monotonic spectral lines intensities is presented. Fundamentally, the intensity distribution by the wave longitude is computing using the Boltzmann distribution:

$$I(\lambda_{mn}) = \frac{1}{4\pi} \frac{hc}{\lambda_{mn}} A_{mn} g_m \frac{N_i}{Z_{int} T_{ex}^{at}} \exp\left(-\frac{E_m}{kT_{ex}^{at}}\right) \quad (5.2)$$

For fore information and parameters description see [21]

To solve this equation, the concentration N_i of the different species in the plasma has to be computed. The team of Prof. André does it by solving the Gibbs free energy of the plasma, written as 5.3²:

$$G = \sum_{i=1}^N n_i [\mu_i^0 + RT_{tri} \ln\left(\frac{n_i T_{tri}}{\sum_{i=1}^N n_i T_{tri}}\right)] \quad (5.3)$$

and

$$\mu_i^0 = -RT_{tri} \ln(z_{tri}) - RT_{ex}^{at} \ln(z_{int}) + e^0 \quad (5.4)$$

$$\sum_{i=1}^N n_i q_i = 0 \quad (5.5)$$

$$P - \Delta P = \sum_{i=1}^N n_i RT_{tri} \quad (5.6)$$

$$\Delta P = -\frac{1}{24\pi\epsilon_0 l_d} \sum_{i=1}^N q_i^2 n_i \quad (5.7)$$

An example of the results in [21] can be seen in the figure 5.5 below.

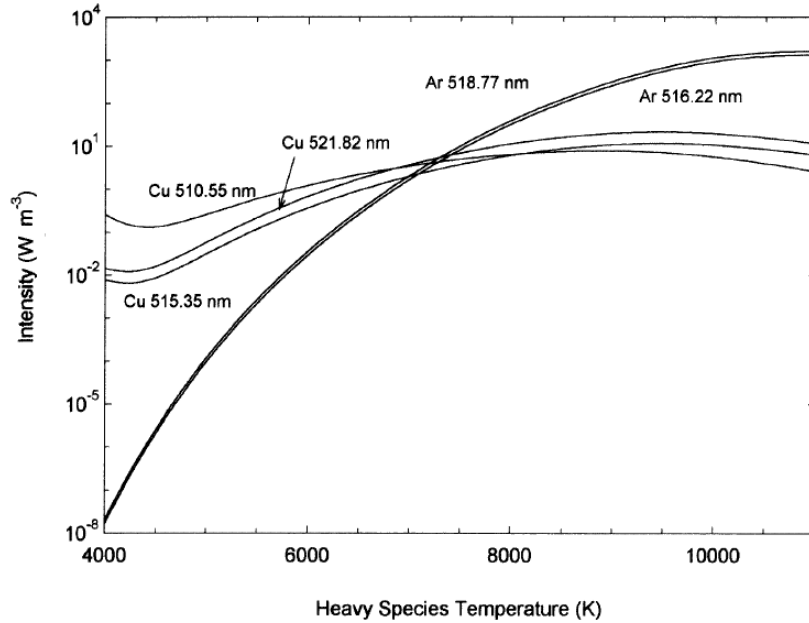


Figure 5.5: Emission intensity of different spectral lines with the plasma temperature. Source: [21]

The paper of André, P. et al. shows that a computation can be done that links the temperature of the plasma and the its emissions' intensity, but the problem remains on the link between the power provided and the temperature of the plasma's heavy species.

²The temperature is assumed to be constant in the plasma, and hence the chemical equilibrium is reached when the Gibbs energy is minimal.

Simulation of emission spectra

C. Trassy and A. Tazeemb [23] developed a simulation algorithm to compute the emission spectra of a given plasma. The input parameters are: pressure, temperature, electron density and emitting species number density. In this method some effects such as Doppler, Van der Waals, Stark and instrumental broadening, shifts and self-absorption are taken into account. The results of the simulation methodology can be seen in the fig. 5.6 below. Again, the inputs of this assessment are not what was sought about it could shed light onto the matter.

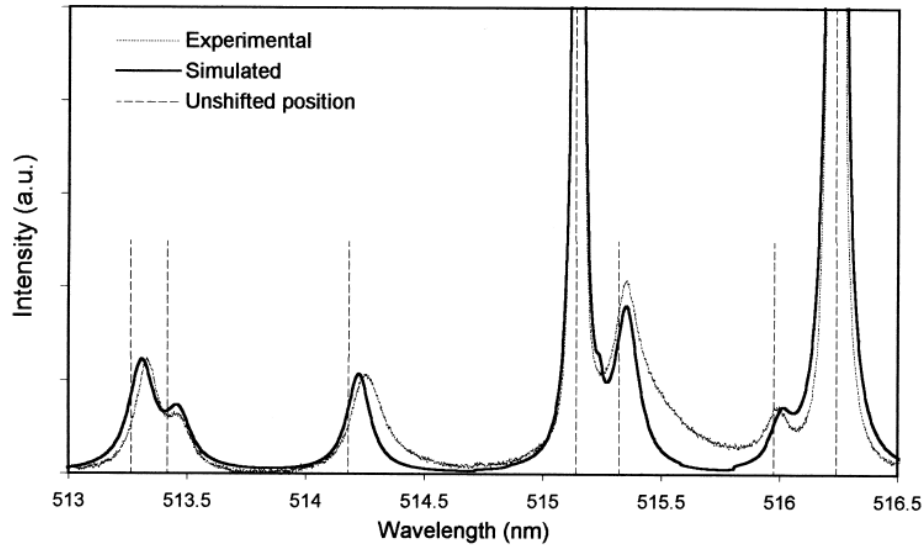


Figure 5.6: Intensity of emissions depending on the wavelength. Source: [23]

This simulation method, despite having some error with respect to experimental results has a broad area of possible applications thanks to the quantity of errors that can take into account. Nonetheless, it might be useful to use a verification of the formulation proposed in [21] since it is a numerical approach nature.

To sum up this section, only experiments in a very particular experimental environment have been found. Other experiment shave been checked, which relate properties of plasma to its radiating emissions. Although they are interesting for shed light on further research, knowing such plasma properties for an electric arc in the open-air is still a challenge.

Chapter 6

Conclusion and further work

On LIS performance

This work has constituted a first approach to explore which is the physical relation between LMA sources and LIS events, while providing a comprehensive introduction to the sensors and data products. For the analysis of which properties of LMA sources have an impact on its likelihood to be detected by LIS (through associating events with sources by time proximity), the following parameters have been taken into account:

1. Typical value the sources' height in a time bin
2. Maximum power of the sources in a time bin
3. Typical value of sources' density in a time bin

It has been found how power has a role on enhancing the detection by LIS of high populated time bins. It has been observed how this impact is not so relevant when observing the maximum power in a bin, which suggests that LIS is more likely to detect large groups of sources rather than detecting those with high power. It has also been found that the distribution of power vs height inside a time bin is most likely symmetric, since a typical height weighted by power has been found to be virtually the same as the height without power weight. More precisely:

Regarding the height:

The median has been taken as a representative value for each bin, which has not shown any remarkable tendency. With the aim to account the possible effect of a source power to influence on the detection, also a weighing of the height median values has been done, using the power of each source as a factor. This latter approach has shown similar distributions to the without-weighting approach. And thus not providing any valuable result.

Regarding the maximum power:

In this approach it has been seen for the first time how the power of the VHF RF sources might have an impact in its detection by LIS: the bins where also events were detected usually had a higher maximum power recorded. Nonetheless, the further approaches such as cluster density have shown more clear results that link power and detections.

Regarding the density of sources in the bins (and the sum of power):

It has been shown how both *only sources* and *sources+events* bins are more numerous in the low-density spectrum. Nonetheless, it has been observed how the increase rate is not the same, and the proportion of bins detected by LIS in relation to the not-detected diminishes in the low-density spectrum. This means that at lower densities LIS also has lower efficiency on detecting source clusters. Finally, this result has been enhanced by the distributions of sum of power for each bin; that show a clear pattern where bins with *sources+events* have higher sums of powers.

This is important, since it means that it is not simply the number of sources which has an impact on the detectivity by LIS (which could mean that there is another property of the sources, not studied

yet, that generates this result); but is the power that generates the difference in detectivity. Clearly, the sum of power and the density are very connected and it has been shown how is the this combination and not another (e.g. maximum power at the bin, mean height...) the ones that has an impact.

Further work regarding this study should be to explore deeper the relation between density/sum of power and the detectivity by LIS. The final objective could be the development of a model that would link sources properties and event presence/radiance, which could bring new insights on what is the link between the physical processes that generate VHF sources and light pulses. From a more pragmatic perspective, the said model could be helpful in predicting what should be the LIS detection associated to a given LMA detection,

On the other hand, it is clear how in this work the radiance –i.e. the LIS detections’ properties– has not been used; the second part of a further study could be to try to do the experiments the other way around: take the bins where events were detected and see which radiance, clustering etc. was required for it to be detected by LMA. One could argue that this approach would be not that useful, since LMA detects mostly everything that LIS detects, nonetheless it could shed some light on the matter anyway.

In this work also were produced histogram distributions from a data-point point of view, not from a bin point of view; which can be found in appendix A.4. They have not been commented since the approach is completely different: there is no association between LMA and LIS data, and as such they only offer space for an analysis within the same sensor (e.g. "LIS might be more likely to detect events with greater radiance than X).

On other analysis of the data

It has been proved how the simple expression " $\Delta T_{Flash} = T_{LastDetection} - T_{FirstDetection}$ " produces different results for the same flash depending on which is the sensor where the detections are coming from (LIS or LMA). It has been shown that for the studied cases the typical lighting time is greater if computed with sources detections; result that is rather logical since LMA has a higher detection rate than LIS. This result can be useful as a statement about how LIS data must be used with precaution if lightning duration wants to be computed. Of course, some further studies on this matter could be focused on finding an expression, or pondering events’ parameters in a way such that the typical duration of a lightning with LIS data reassembles to the duration computed from LMA data.

On electric arc emission for LIS calibration

An attempt has been made to assess which should be the voltage applied to an open-air atmosphere at 2500 m approximately in order to generate an electric arc observable by LIS. A typical value for the radiance emitted by events that had sources associated at this height has been obtained and an attempt to model the loss of radiance throughout the atmosphere has been done. This has been done using a demo of the software MODTRAN and has not been validated. Regarding the link between the applied voltage and the radiance emission little information has been found. For a direct relation, only experiments in a very particular experimental environment have been found. Other experiments have been checked, which relate properties of plasma to its radiating emissions. Although they are interesting for shed light on further research, knowing such plasma properties for an electric arc in the open-air is still a challenge. Further work on this direction could focus on doing in situ experiments to generate data that links voltage applied with emitted radiance.

Appendix A

APPENDIX

Contents

A.1	Geostationary Lightning Mapper description	64
A.2	Night vs. Day distribution of LIS during 2017 period	65
A.3	Detections' properties influence on LIS detectivity from a typical value approach	66
A.3.1	Sources density	66
A.3.2	Sources' height	68
A.3.3	Sources' Power	70
A.3.4	Section summary	74
A.4	Extra Figures	74
A.4.1	Mean sources' height	74
A.4.2	Histogram distributions from a data-point point of view	76
A.5	User Guide for data the processing codes	77
A.6	Codes for data processing	89
A.6.1	LIS_HDF_processor.m	89
A.6.2	LIS-vs-LMA_comparator.m	113

A.1 Geostationary Lightning Mapper description

The Geostationary Lightning Mapper (GLM) is a sensor integrated on satellites that constantly monitor the Americas (fig. A.1b) for weather study [10]. A system of GLM sensors is being implemented with a new series of satellites (fig. A.1a) within the Geostationary Environmental Satellite Network (GEOS) mission; which has been developed by NASA and NOAA. This mission is launching to geostationary orbit a constellation of 4 satellites (GEOS-R,S,T,U), with a GLM instrument each, that will cover a huge part of Earth surface, centred in the Americas. The improvement in weather forecasting and monitoring with these new sensors come from the new capabilities for detecting cloud-to-ground (CG) and intra-cloud (IC) lightning; as well as from the fact that the satellites also will have a Advanced Baseline Imager that observes the atmospheric moisture with improved scale resolution from previous, similar instruments, and provides fresh data five times faster than its predecessors.

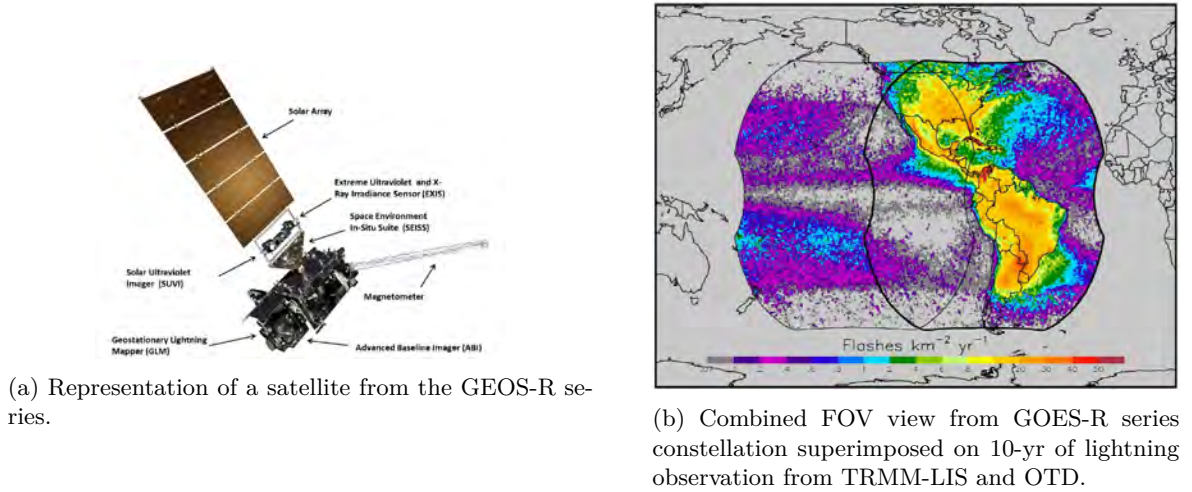


Figure A.1: GEOS satellite and its FOV on the Earth surface. Source: [10]

Sensor Description

The GLM, as the LIS, measures the radiance that lightning emit through the top of clouds with an imaging device. Actually, the sensor's principles are very similar to those from LIS¹ and will not be explained another time.

It is important to remark that it can measure CG, CI and CC lightning during day and night; the latter being crucial as the satellite is stationary and will suffer from continuous daytime cycles over its FOV.

As the GLM orbit is much higher than the one of ISS-LIS (35786km vs. 400km) it is clear that it will suffer in its spatial resolution. In fact, it goes from 8km at the NADIR of the CCD to 14km at the edge. This decrease would be much higher, but it has been lowered by the implementation of a CCD that has smaller pixels on the edges, allowing an homogeneous distribution of resolution along the image. To achieve a acceptable spatial resolution from such a high orbit, a CCD of 1372 x 1300 pixels has been implemented (for the record LIS had a 128 x 128 pixel CCD). Also, a higher telemetry speed (and therefore the amount of processable data) has had a great impact on increasing the detection efficiency, due to the possibility to establish a lower threshold to detect weaker luminous events. The overall result is a geostationary imaging sensor that has a flash detection of at least 86% and a false alarm rate of 5%.

Regarding the detections classifications the GLM uses, just as LIS, a groping structure that clusters the luminous events in groups, flashes and areas. Information about this parent-child algorithm can be found on [8].

Other technical properties are described in the table below (fig. A.2).

Figure A.2: GLM sensor performance properties. Source: [10]

A.2 Night vs. Day distribution of LIS during 2017 period

As previously mentioned, the thunderstorms in the west Mediterranean weather (and therefore over Deltebre) are not very numerous. This, added to the fact that the ISS orbit does not always pass over Catalonia; and that for a approximately 100 x 100 km area the view time at maximum of 2 min; makes the amount of lightning observations over the Ebre delta not be very high. Then, it is relevant to generate some statistics about the probability of LIS recording a thunderstorm in the place. It is

¹I.e. a wide FOV, focusing radiation beam in a CCD with an interference filter at 777.4 nm, and finally a data processing subsystem.

A.3. DETECTIONS' PROPERTIES INFLUENCE ON LIS DETECTIVITY FROM A TYPICAL VALUE APPROACH

also important to remark that LIS has been operative only since March 2017 and the data of these observations in particular reach June 2018.

While the LIS was operative (from March 2017) the ISS passed over Deltebre 1998 times. It only captured activity in 8 of those. Most of activity occurred during daytime, and the magnitude order of detected flashes has been of tenths.



Figure A.3: Night-Day presence of lightning detected by LIS around Deltebre area from March 2017 to July 2018.

In the figure A.3 is displayed the day-night distribution of LIS detected flashes during 2017 and 2018 periods. It is in this moment that the background remover and the LIS post-processing unit detailed in section 2.1.2 shows its importance, as a good managing of the background radiance is critical during daytime.

A.3 Detections' properties influence on LIS detectivity from a typical value approach

In this appendix the reader will find an analysis of the influence of some lightning properties on its detectivity using a representative value approach. This approach has been only added to the appendix because it was only after it was produced that it was found that the Histogram method provides clearer information about the matter.

A.3.1 Sources density

In this section will be analysed the effects of the number of sources in each time bin on the LIS detection. To do so, a typical value for bins with only sources and bins with sources+events has been extracted from each 10 min. time period, if the said period had simultaneous (LIS and LMA) detections to compare on.

These typical values have been made with the average and also the median, as in every set of data appear to be some heavy outliers that will very much effect the mean. They have been displayed in figure A.4; along with the number of bins with both events and sources, as the number of data available may play some role in the results.

In the figure A.4 are displayed both median and mean typical values, for the mean values there seems to be no tendency (in reference to comparison between bins with LIS detection and bins with LIS+LMA detections); but a further inspection, with the median reveals more interesting. It shows that for the cases with significant data (the first and the second in time), the bins that also had LIS detections tend to more source-populated.

The mean and the median in the 171018 17:00 period differ so much because the outliers: the standard deviation for the bins with events+sources and only sources in this period is 4.5682 and 6.9174,

A.3. DETECTIONS' PROPERTIES INFLUENCE ON LIS DETECTIVITY FROM A TYPICAL VALUE APPROACH

respectively. This is a lot, considering that the the average is situated at approximately 8 sources in each case, so the median seems indeed a better indicator of the typical value for this case.

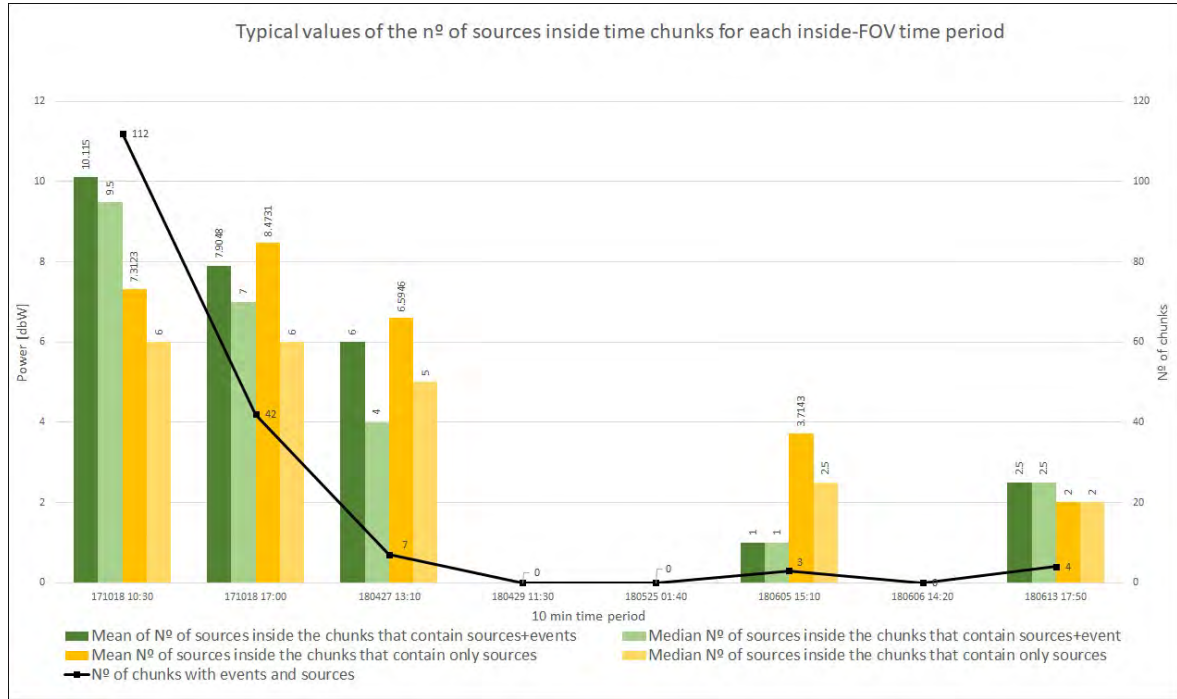


Figure A.4: Typical values of sources densities in bins for each 10 min period.

This tendency (LIS detections come from more populated bins) is not so clear in the following cases, but as the figure shows, the first case has 128 simultaneous detections and the second 42, whereas the third has only 7 and the next period with data 3.

Also, it is mandatory to bear in mind what it is known from section 4.1.1: the size of the thunderstorms recorded during 2017-10-18 was much higher than during the other dates, and this is reflected by the number of bins with simultaneous detections as well; from the first to the third time period the number of simultaneous detections is reduced a 94.53%. So if there is not a permanent tendency on all the data, it is acceptable to state that detection rate increases with source density in time, as displayed by the data that provides more information.

A.3.2 Sources' height

Typical values of height in each bin have been computed, and then averaged again to get the typical values of the typical heights for each 10 min. time period. This information is displayed on the figure A.5, where it is show the mean and median values for each case.

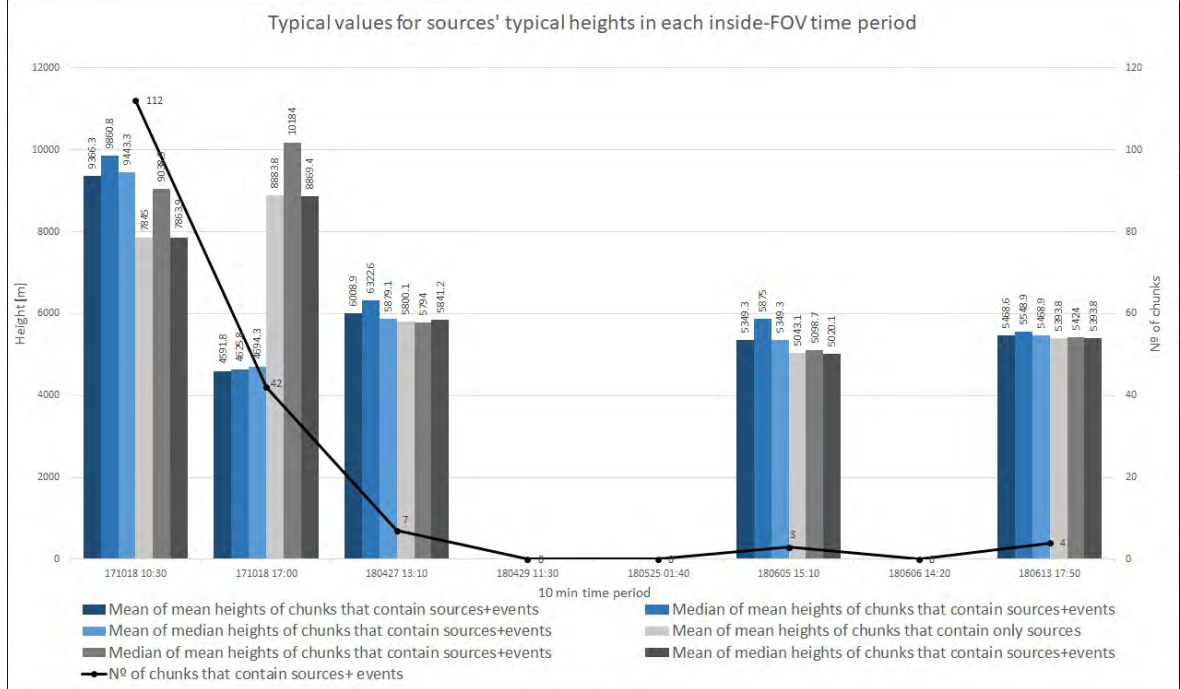


Figure A.5: Typical values of sources' height in each 10 min. time period, achieved from the typical height values inside each bin of the same period.

In the figure of this section there is 6 variables per period of time, as the following values have been computed for each type of bin:

1. Mean of the typical heights from bins with ; where the typical heights were obtained with simple mean.
2. Median of the typical heights; where the typical heights were obtained with simple mean.
3. Mean of the typical heights from bins with ; where the typical heights were obtained with the median.

The goal of this diversity of values is to check the effects of the data dispersion on the results, both of sources' heights and the variability of their typical values. As it can be seen in the figure, dispersion does not play an important role in heights, as medians and means show the same result: there is no tendency observable for the height.

Indeed, in the first case the typical height values of the events+sources bins are higher than those that only have sources, but in the second time period this relation is twisted -and very accentuated- the other way around. In fact, in this 171018 17:00 time period the detected flash was one that stoke the ground, as commented during the data overview (section 4.1.1).

LINET effect

One could argue that return stroke can be the dominant variable in this second case. Let's compare the data gathered by LINET in the two time periods of 171018.

A.3. DETECTIONS' PROPERTIES INFLUENCE ON LIS DETECTIVITY FROM A TYPICAL VALUE APPROACH

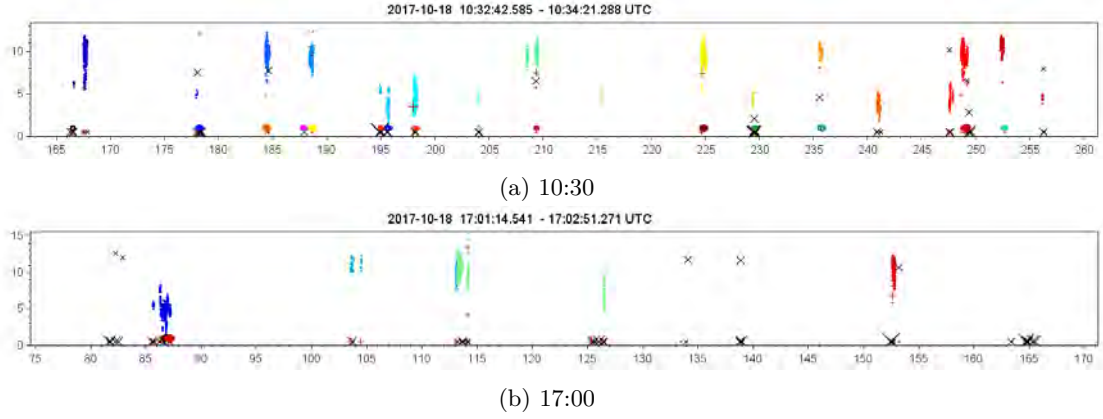


Figure A.6: Space-time detections distribution of 2017-10-18 with LINET included (X and +). "x" are negative strokes and "+" are positive strokes. Circles are events (coloured by flash) and dots are events (coloured by time).

In the figure A.6 are displayed the LIS, LMA and LINET detections for the time periods of 2017-10-18. As it can be seen, virtually all the detection clusters -flashes- have some stroke associated; both flashes that LIS did and did not detect. This means that the sole presence of strokes has not a direct implication on the LIS detection of its flash.

A.3.3 Sources' Power

One of the main sources' characteristics that may influence in LIS detection is the source's power. Moreover, it may do it in various ways. For instance, the summed up power in a bin time, the maximum power registered in that bin or the height of the power centroid may have a role in LIS detection of the said bin.

Sum of sources' powers

This case is a crossover between the influence of power in LIS detection and the density of sources in a given bin. Obviously, it will have a strong relation with the section A.3.1, but it may have some amplifying effects on some of the samples. It is displayed in the figure A.7, where the mean and median of powers' sums for all types of bins are printed.

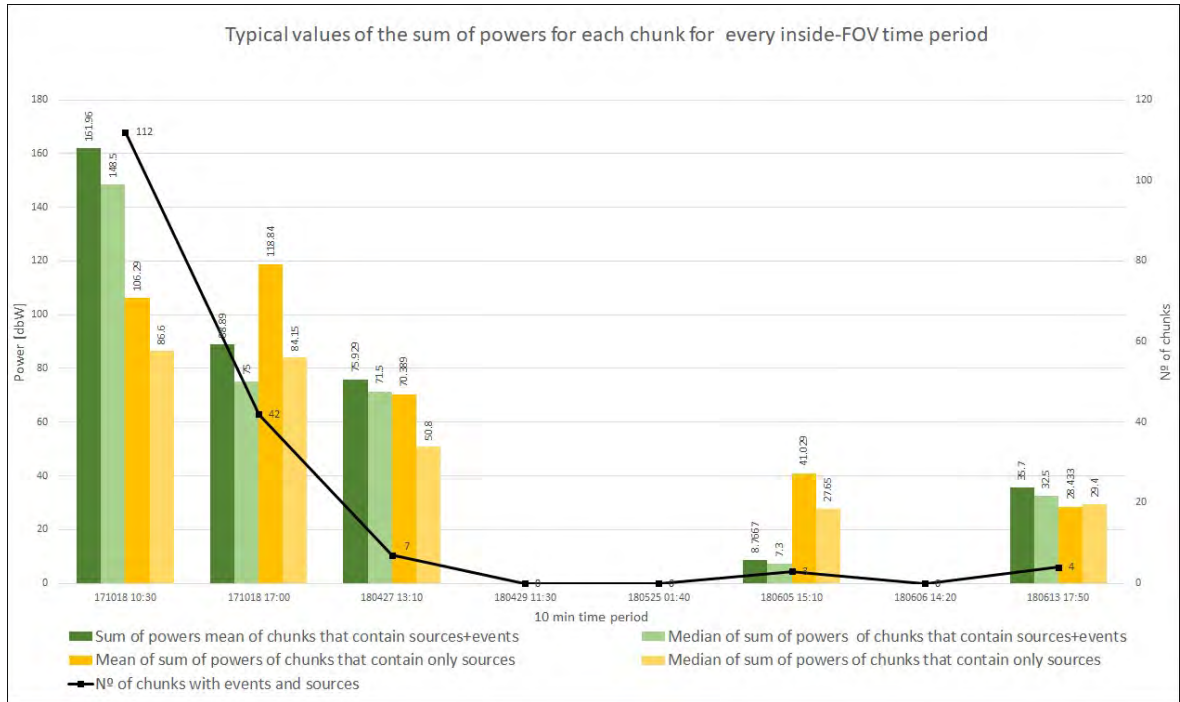


Figure A.7: Sum of powers for each bin

In the said figure, some unexpected effects arise: whereas in the first, third and last time period the relation observed in section A.3.1 seems to be enhanced, in the other cases its appearance is drastically reduced. For instance, the mean value for the only-sources bins in the period 171018 17:00 sky-rockets way above the other values for the same period; while in the first case the relation between sources+events bins and sources bins is maintained.

This actually means that, while in the 171018 10:30 period the sources of both types of bins have similar powers, in the second case the power of the sources that have no events associated is higher than those from bins where events were detected; so this may indicate that sources' power level - or at least, its sum - does not have an impact and detections.

Power-weighted sources' centroid

Even if in section A.3.2 it has been revealed that the height by itself does not have a clear role in LIS detection, its effect combined with power might do. I.e. that higher source with lower power might be more likely detected by LIS than a lower source with higher power. To reflect this effect, the power-weighted centroid (in height coordinates) for each bin has been made; and then computed

A.3. DETECTIONS' PROPERTIES INFLUENCE ON LIS DETECTIVITY FROM A TYPICAL VALUE APPROACH

its typical value for each 10 min. time period.

The said power-weighted centroid height has been computed applying the following formula in each bin:

$$\frac{\sum -i = 1^{n-srCS}(pwr - i * height - i)}{\sum -i = 1^{n-srCS}pwr - i}$$

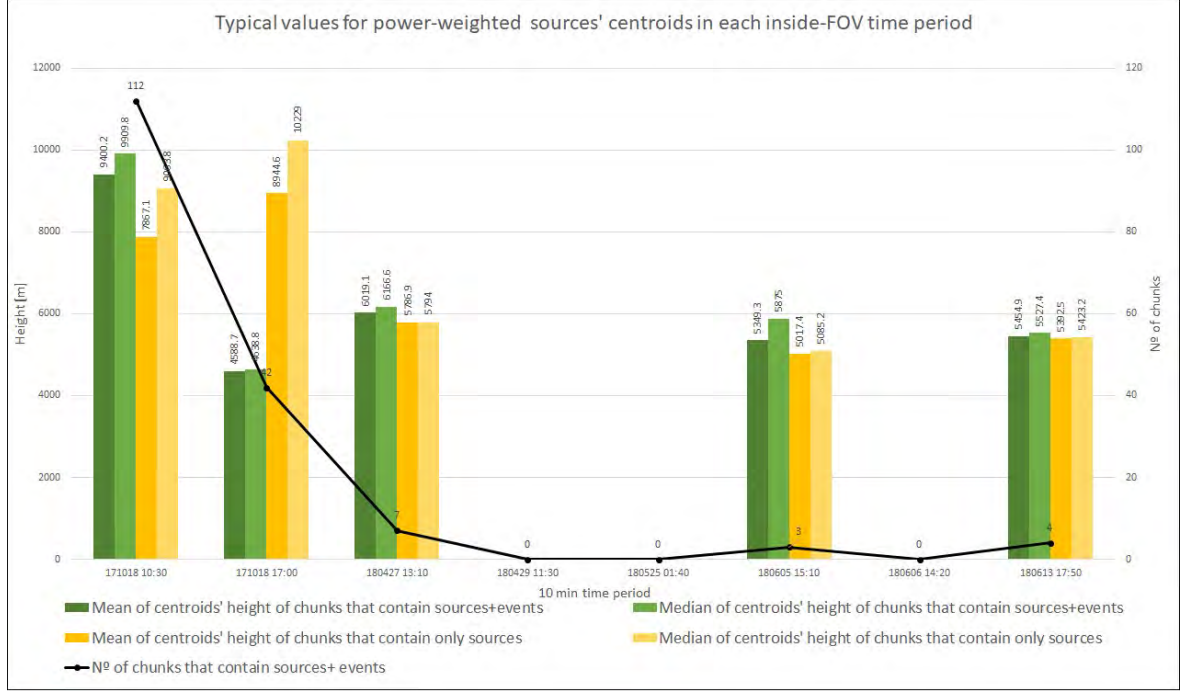


Figure A.8: Typical values for power-weighted centroid positions in each time period.

The information displayed in the fig. A.8 shows similar results to those we had from section A.3.2, figure A.5. The second case has a very strong variation between sources+events bins and sources bins: it has already been discussed how in that case the heights of detected sources were very low, as the lone detected had its sources close to ground. This, computed now with powered centroid, stays roughly equal, even if now there are big differences in mean and median; which means that in each bin there is usually no deviation in heights (a flash develops closely in space) but there is high deviation in power (a flash produces a high range of VHF emissions).

Again, in the first case (where there is information available for a number of flashes) shows some difference between sources+events and sources bins: it seems that in that period the detected sources were higher and more powerful.

Maximum power

Supposing that a number of sources take place roughly simultaneously (separated less than 10 msec.), and are detected by LIS; one can hypothesize that not all of these simultaneous sources have been detected by LIS, and in fact LIS has detected the one with maximum power. To explore this option, the maximum power values in each time bin have been recorded, and will be compared to those from bins where no events were recorded by LIS.

The figure A.9 displays a point (dot or asterisk) for each bin. Each point represents the power of the source that has the maximum power inside the bin. The blue asterisks correspond to bins where events also were detected, and black dots to bins where no events were detected. It is clear that the vertical -so almost simultaneous- sources clusters correspond to separated lightning.

A.3. DETECTIONS' PROPERTIES INFLUENCE ON LIS DETECTIVITY FROM A TYPICAL VALUE APPROACH

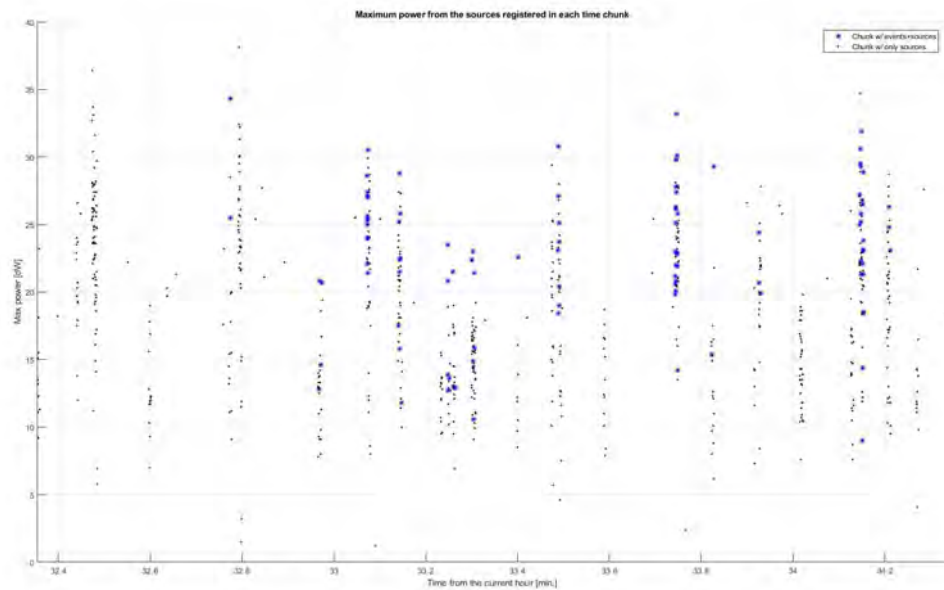


Figure A.9: Maximum power for each bin during 171018 1030

Taking in account that the area came under LIS FOV at 10:32:46, we can observe how the sources that were accompanied by events group up in higher power clusters. With the exception of the clusters at min. 32.95 and 33.3, the sources' power of clusters that LIS also detected are above 20 dBW. This tendency is well observable in the mentioned case. The figure A.10 shows results of the same approach for the other 10 min. sets.

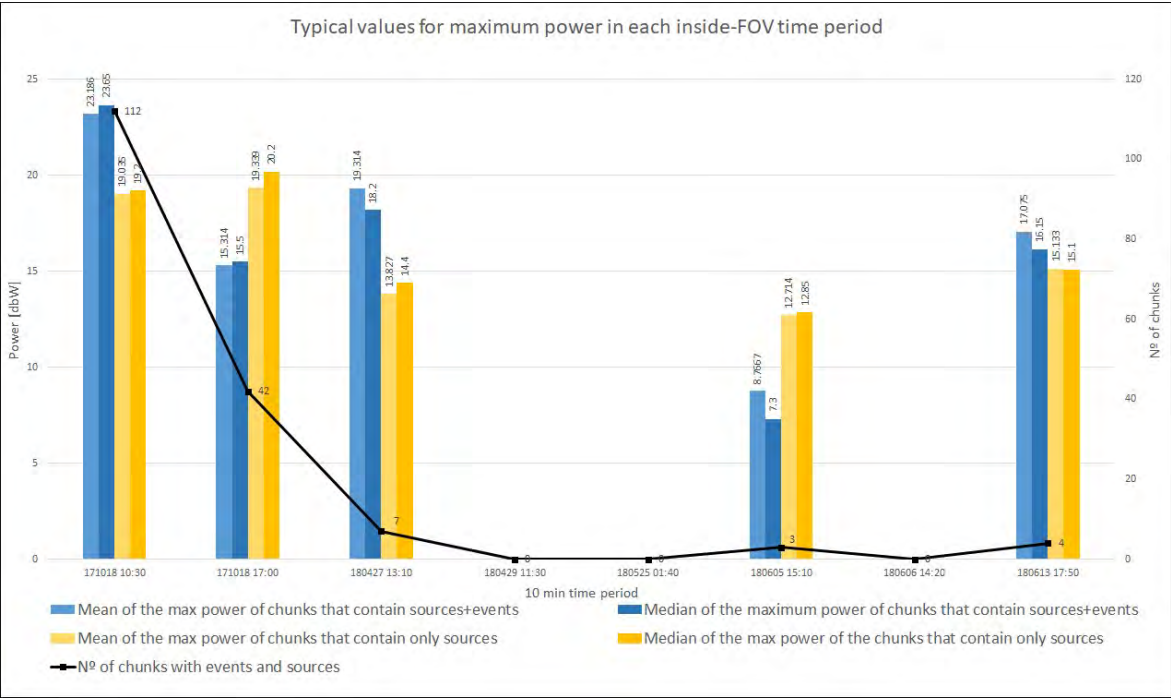


Figure A.10: Typical values of maximum power for each inside-FOV time period

There, it can be seen how this tendency is hardly maintained through the other cases. If both the 180427 and the 180613 periods have sources+events bins with higher maximum power than the other

bins, this is not the case for the second and sixth time periods, where it is the other way around.

Even if in the first period the amount of lightning evaluated is way higher than the second, the data shows how both the mean and median of only sources bins remains roughly equal. This could mean that virtually all undetected bins on the day had an average maximum power value, but it had no effect on its detection; as in the detected cases the maximum power varies without restriction.

A.3.4 Section summary

It is important to reiterate that the amount of data in Deltebre area is low, and therefore descriptive statistical statements can't and will not be proposed. What is possible and interesting is the fact that typical values approach can at least exclude some factors of influence of LIS detection and open shed light on which might be the following way of analysis. Therefore, some conclusions can be extracted from this section, regarding the data gathered in Deltebre:

- A tendency has been observed in the time periods with more lightning detected, that associates the detection rate of a given source cluster with its density. This effect has not been recorded in two of the 3 cases that have low amount of data, but its importance in deciding if there is a tendency density-detection may be disregarded for the same reason.

In the more populated data sets (171018), this tendency has been observed in the median typical values, but not in the average, that in these cases is much higher than the mean. This suggests that in general the mentioned density-detection applies, but rather than stating that more dense clusters will be detected it should be read as low density clusters are not detected by LIS. I.e. that LIS did not detect some clusters with high density (see fig. A.4, period 171018 17:00, mean n^o sources of only-sources bins), but usually the bins with events were more source-dense than the others.

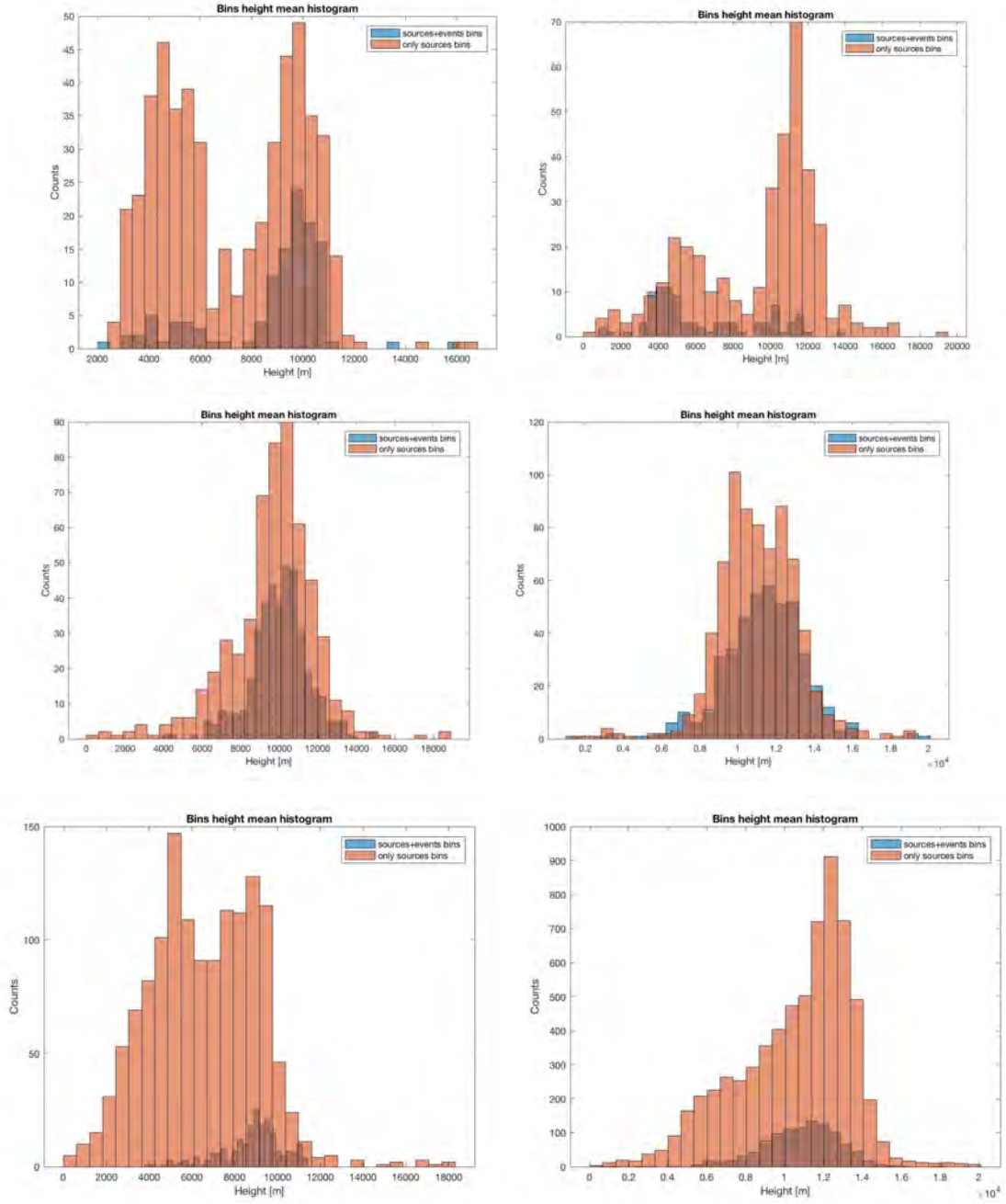
- Height by itself has not played any role in the detection of events by LIS, in the Deltebre data. In the period 171018 17:00 the only lightning detected was the one to, presumably, strike the ground. In fact, this has been checked by examining the 171018 periods LINET data. Nearly all the lightning detected by LIS and/or LMA had a stroke detection associated, so there is no relation with presence/absence of strokes detection with LIS detection. Nonetheless, it is possible that a given intensity or stokes density is required for LIS detection, or at least to enhance the possibilities of detection.
- The sum of powers in each bin does not provide any clear tendency. Obviously, it maintains a strong relation with the bins' density, but it does not the somewhat clear relation that appeared in the cases with higher amount of data. In fact, in some cases it accentuates the said tendency (171018 10:30, 180427, 180613) and in others (171018 17:00) reverses it.
- The powered-weighted centroid provides similar results to those coming from the height by itself. At this stage this might reveal itself as palpable, because now it is acknowledged how neither sum of powers or height produced any tendency on the data.
- The effect of the maximum power typical values in each bin for its detection by LIS has not show any tendency either. In fact, for both 171018 time periods the typical value for maximum power of non-detected bins stayed still, whereas for the cases that LIS detected the value varied both up and down.

A.4 Extra Figures

In this appendix the reader will find extra figures that were mentioned in the main body of the work, but were more fit to be in the appendix.

A.4.1 Mean sources' height

Below are displayed the histograms that show the mean height distribution of the sources detected during all the observation periods, once separated in time bins. Each time bin has a duration of 10msec.



Mean sources' heights of the time bins for the periods (from top to bottom, left to right):

171018 10:30	180809 18:50	180918 03:30
171018 17:00 (single flash)	180831 04:40	181018 15:10

A.4.2 Histogram distributions from a data-point point of view

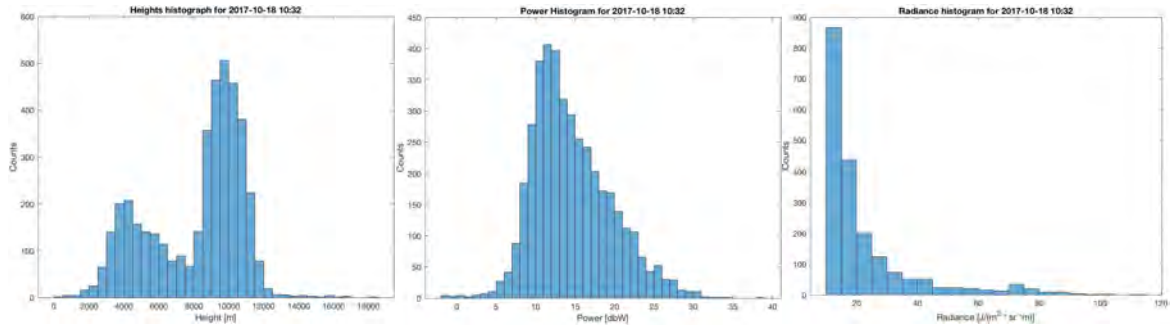


Figure A.11: Data-point POV histograms for 171018 10:30

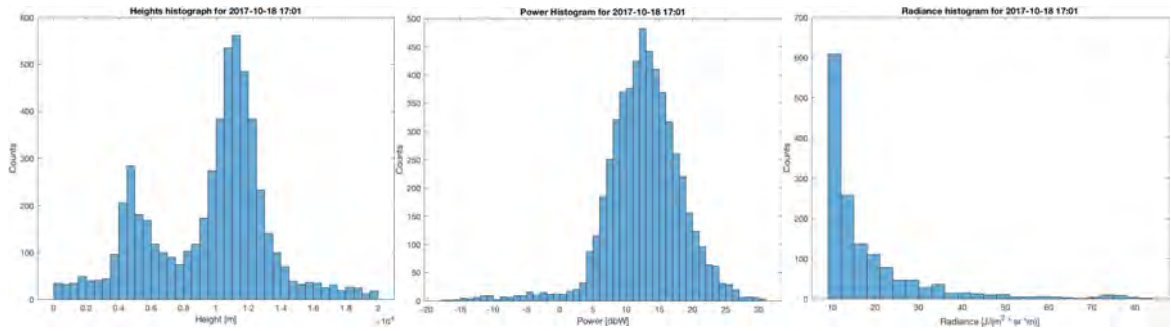


Figure A.12: Data-point POV histograms for 17-10-18 17:00

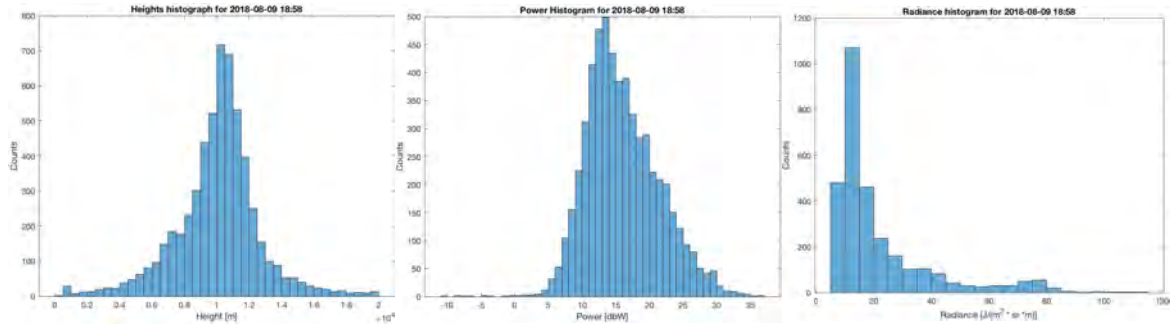


Figure A.13: Data-point POV histograms for 18-08-09

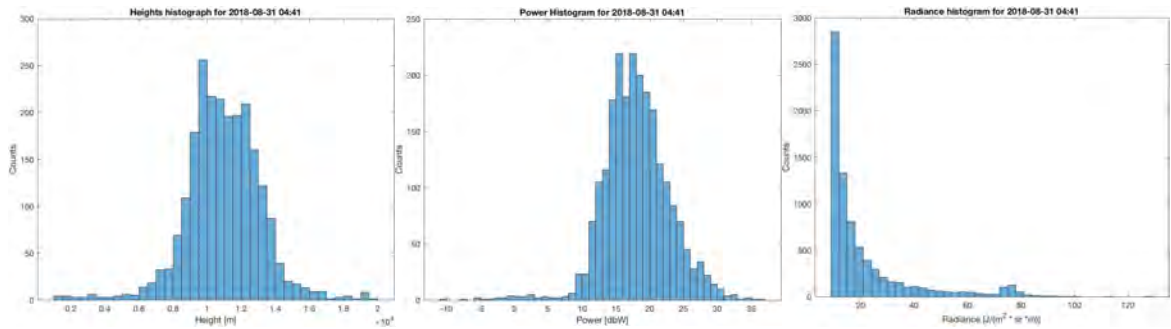


Figure A.14: Data-point POV histograms for 18-08-31

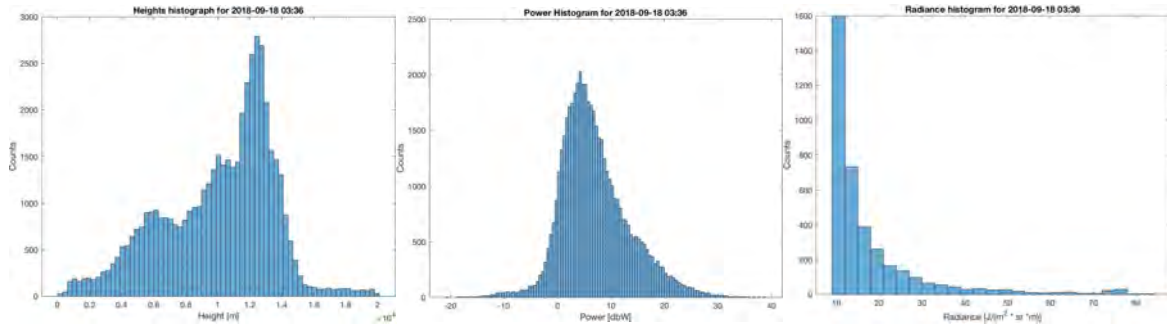


Figure A.15: Data-point POV histograms for 18-09-18

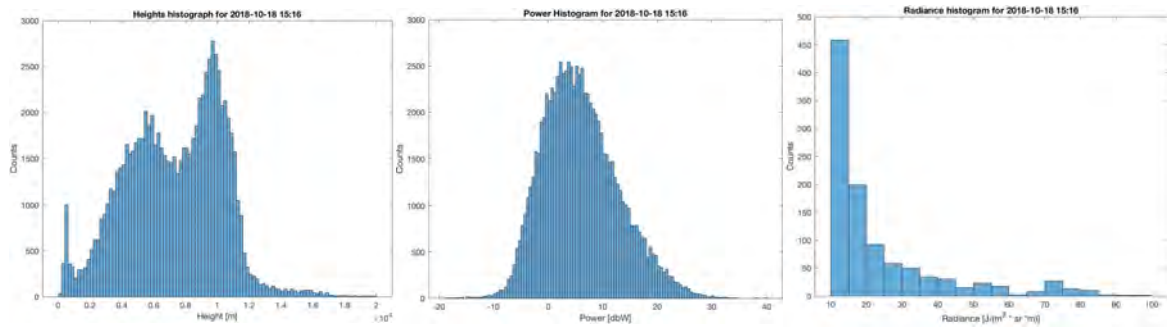


Figure A.16: Data-point POV histograms for 18-10-18

A.5 User Guide for data the processing codes

In this appendix the reader will find a a User Guide that helps in the usage of the codes used in this work.

LIS and GLM data processing software

USER GUIDE

Fontcuberta i García-Cuenca, Ícar.

May 11, 2019

The scope of this document is to make a guide for the most relevant applications of the MATLAB® codes developed by Fontcuberta, Í. for Lightning Imagin System and Lightning Mapping Array data processing. The codes can be found at: <https://github.com/icarfontcu/LIS-LMA-data-reading-codes>. Any questions or suggestions are welcomed and can be addressed to icar.fontcuberta@gmail.com.

In the Annex are attached screen-shots of the processes described in the document.

Contents

1	LIS HDF files processor	1
1.1	Download LIS HDF files	2
1.2	Read interesting information from HDF files and print it into txt files	4
2	LIS vs LMA comparator	4
3	ANNEX	7
3.1	LIS HDF processor	7
3.1.1	Download HDF files	7
3.2	Process HDF files to txt files	9
3.3	Process NC files	10
3.4	LMA vs LIS comparator	10

1 LIS HDF files processor

The software "LIS_HDF_processor.m" is a MATLAB® software that helps the user to download LIS HDF files, read them and extract its interesting information in order to print it into txt files, which is a more usual format for temporal data storage.

To do so, some the user will have to specify some variables at the first lines of code, and select some options in a prompt menu that the program displays in the command window. This guide will only describe the usage of the applications just mentioned, despite the program having some more; that

are also interesting but can be used by simple observation or by derivations of the indications in this document. Below, each section will describe the guide for a program and each sub section a particular application.

1.1 Download LIS HDF files

The LIS Bulk Data downloading tool only selects files by dates and huge amount of irrelevant data is downloaded using only that tool. Instead, a semi-automatic process has been developed, seizing the LIS data website space-time domain searcher; which allows the user to know which files contain information of its interest.

In this application the program prints a txt files containing URLs that download the mentioned interesting HDF files using a wget command (specified later on).

1. Go to <https://lightning.nsstc.nasa.gov/isslisib/isslissearch.html> and use the space-time domain searcher in order to display the names of the interesting HDF files (fig. 1b).
2. Manually select the list of HDF files and copy it to a txt file (fig. 1a). The caption of the list must also be included in the selection.
3. Introduce working directories and file names required by the code, in the first lines.
 - (a) website_filename: file and directory of the txt file containing the copy of the list extracted from LIS website.
 - (b) write_dir: directory where txt file with generated URLs will be printed.
4. Introduce in "LIS_HDF_processor.m" the space-time domain of your interest (fig. 2). This might not be the same as the one used for the LIS searcher. For instance, it might be of the user interest to select a wider domain in the website to be sure of getting all the interesting files.
5. Execute the program and introduce "7" when asked by the command window.
6. Open the windows cmd and go to the directory where you want to download the HDF files.
7. introduce the following command:

```
wget -user EarthDatauser -ask-password -auth-no-challenge -no-check-certificate -i interestingURLSfile.txt1
```

The HDF files will start to download (information displayed in the cmd) and stored in the current directory.

¹To use this command a user login has to be made into the Earth Data website

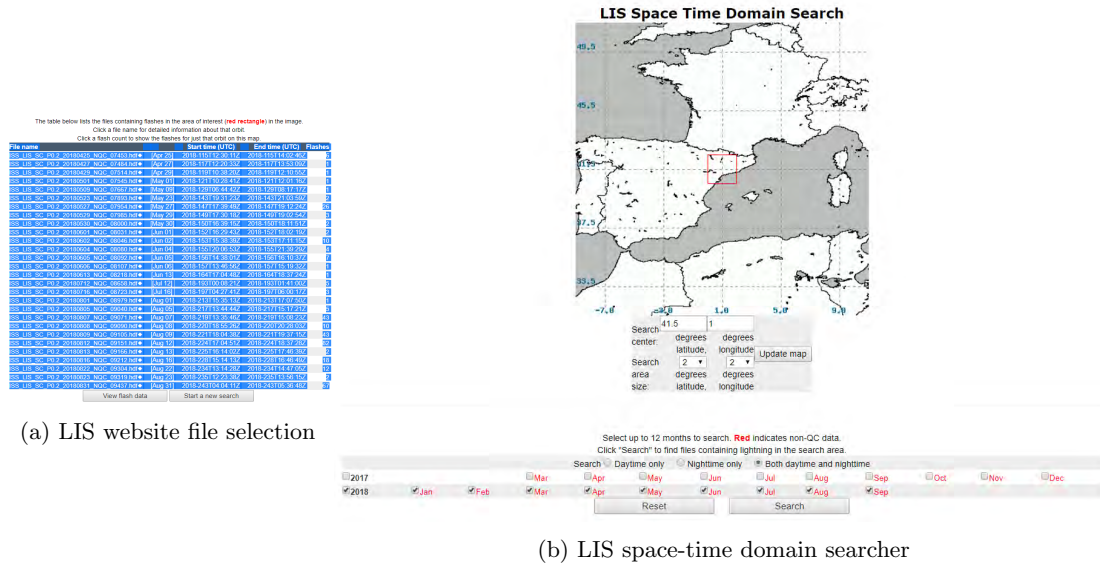


Figure 1: Screenshots of LIS website

```
%coordinates info
deltebre=[40.7212388 0.7176492];
santamarta=[11.2403547 -74.2110227];
barranca=[7.06878 -73.744418];

%scanning_area specification
centroid=barranca; %LAT/LON (remember, on the plot, this would be y and x)
range=60*sqrt(2); %range in km (the sqrt(2)) is to get for sure a squared
%time interval
starttime=datetime(2018,6,1,0,0,0);
endtime=datetime(2018,7,3,8,0,0);
timerange=[starttime endtime];
```

Figure 2: space-time domain parameters in the code

1.2 Read interesting information from HDF files and print it into txt files

This application reads the LIS HDF files that the user has stored and print its information (if relevant) to txt files with adequate names. These txt files will contain a number (it can be changed) of columns with one type of data each, e.g. LAT, LON, time...

Note that the code is prepared to access a big collection of folders with data, as it has functions implemented that register each subdirectory for other subdirectories until it finds where the HDF files are. This process is completely automatic excepting for the specification of the space-time domain selected. The steps required are described below.

1. Introduce the required directories in the first lines of the code.
 - (a) write_dir: where the txt files with information will be written
 - (b) read_dir: the directory from the program will start looking for HDF files.
2. Introduce the space-time domain as shown in the fig. 2.
3. Run the code and select the option 1 when asked by the command window.

The option 2 also plots the interesting events for validation purposes. The option 5 only plots the interesting events, without writing txt files.

Code projection and future improvements

One possible improvement is to make the downloading of HDF files fully automated. The option 6 of the code's menu is a first approach to do so: when downloading LIS bulk data, the user gets a txt files with a huge amount of HDF URLs. These can be processed to get only URLs from files that contain info from a certain period. What option 6 does is to read the mention big URLs files and generate a similar, new one with only the relevant URLs.

2 LIS vs LMA comparator

The LIS-vs-LMA-comparator.m has the objective of generating statistical distribution and moments for a study of LIS and LMA data products together. Fundamentally, it pierces the given measurement time with time bins of a given size (e.g. 10 ms). Then, it associates the LIS and LMA data points to each time bin for finally working on bin properties (e.g. mean radiance of all the LIS events inside a bin) instead of working with properties of the data points. This approach is useful since the aim is to associate LMA and LIS detections by time proximity, and in the end generate relations a la "this LMA source had this LIS radiance associated", i.e. giving a scalar value for a given detection (or bin).

The optimal way to work with this code is to provide it with two text files. First, a text file where all LIS detections are listed. Second, another text file where are listed the LMA detections that were under the LIS FOV during the measurement time. The LMA detections that were not under

the LIS field of view at a given time can be filtered out with the help of a code made by Prof. Montanyà (Electrical Engineering Dpt. UPC). Once the user has these files, to use the code:

1. Set the variable *LIS_total_filename* as the name (with the path) of the text file where all the LIS events are; line 21 of the code.
2. Set the variable *LMA_filename* as the name (with the path) of the text file where the LMA sources are, line 22 of the code.
3. Set the variable *examined_day* as the day where the measurements were done, line 23 of the code. Use the format: `datetime(yyyy,mm,dd)`.
4. Set the variable *timestep* as the time width of the time bins with which the data will be divided, line 28 of the code. A recommended value is 10ms.
5. Set plotting options by changing the following variable values (1 = on / 0 = off):
 - *histograms*: plot histograms of several bins properties (e.g. mean height of the bin, mean power of the beam...)
 - *plotting*: display plots of bin properties in their position in time (i.e. a point represents a value of a property where the coordinate is the time coordinate of the bin and the ordinate the value of the property). Also plot the excited pixels on the LIS CCD
 - *sources_and_events*: plot the sources and events detected over time. For LMA ordinate is the height, for LIS is a random value fixed at 100 m.
 - *comparing_length_section*: activate the section where assessments regarding which is the length of a given flash are done. Do not activate this if you are not particularly going to use this section.
 - *save_workspace*: save the workspace in the current LMA file directory
 - *savingcsvfile*: save a csv file on the current LMA file directory containing moments of the statistical distributions of detections' properties.

If an assessment of the power of events that had a source at a given height associated the discharge altitude has to be set at the line 576 of the code and the tolerance at the line 577.

In the figure 3 below the variables listed above can be seen. In the figure 4 are displayed the lines of code where the user should introduce the necessary variables for computing the power at a given height.

```

14 - disp('##### START OF EXECUTION #####');
15 - %{\dots}
21 - LIS_total_filename = '/Users/Icar/Google Drive/TFG/LMA_LIS_DATA/ISS_LIS.txt';
22 - LMA_filename = '/Users/Icar/Google Drive/TFG/LMA_LIS_DATA/LMA_FOV_20171018_pass1.txt';
23 - examined_day = datetime(2017,10,18); %used to get the LMA date correctly
24 -
25 -
26 - disp('##Remember to set the examined_day to the one from where the LMA is coming');
27 -
28 - timestep = 2e-3; %sec
29 -
30 - %{\dots}
48 -
49 - %Plotting Options
50 - histograms = 0;
51 - plotting = 1;
52 - sources_and_events = 0; %plot sources and events over time. COMPUTATION LOADING
53 - comparing_length_section = 0;
54 -
55 - %Other Options
56 - save_workspace = 0; %if you want to write workspace. SAFEMODE=0
57 - savingcsvfile = 0;
58 -

```

Figure 3: Introductory section of the LIS vs LMA comparator code

```

570 - %% Power of events that had a source at 2500m associated
571 -
572 - DISCHARGE_ALT = 2500; %m
573 - tolerance = 50; %m. Tolerance for the search
574 -
575 - [rad_dischargealt] = typical_radiance_at_height(DISCHARGE_ALT,tolerance,lma,lis,chunksinfo);
576 -
577 - disp(['Radiances at discharge height: ' num2str(DISCHARGE_ALT) ...
578 -      'm with tolerance: ' num2str(tolerance) 'm:']);
579 - disp(['Mean: ' num2str(mean(rad_dischargealt.averages)) '\mu J/sr/m^2/\mu m']);
580 - disp(['Median: ' num2str(median(rad_dischargealt.medians)) '\mu J/sr/m^2/\mu m']);

```

Figure 4: User introduced variables for power at a given altitude assessment

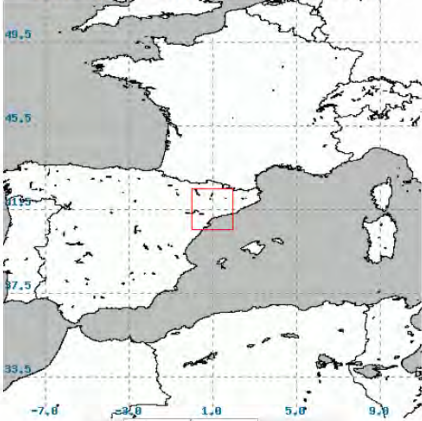
3 ANNEX

3.1 LIS HDF processor

3.1.1 Download HDF files

1. Search for interesting at iss data websiteHDF files using ISS-LIS search tool

LIS Space Time Domain Search



Search center: 41.5 degrees latitude, 1 degrees longitude

Search area: 2 degrees latitude, 2 degrees longitude

Update map

Select up to 12 months to search. **Red** indicates non-QC data.
Click "Search" to find files containing lightning in the search area.

Search ☐ Daytime only ☐ Nighttime only ☒ Both daytime and nighttime

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2017												
2018	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			

Reset Search

2. Select the list of emerging HDF files

The table below lists the files containing flashes in the area of interest (red rectangle) in the image.

Click a file name for detailed information about that orbit.

Click a flash count to show the flashes for just that orbit on this map.

File name			Start time (UTC)	End time (UTC)	Flashes
ISS LIS SC P0.2 20180425 NQC 07453.hdf	[Apr 25]	2018-115T12:30:11Z	2018-115T14:02:46Z	6	
ISS LIS SC P0.2 20180427 NQC 07484.hdf	[Apr 27]	2018-117T12:20:33Z	2018-117T13:53:09Z	1	
ISS LIS SC P0.2 20180429 NQC 07514.hdf	[Apr 29]	2018-119T10:38:20Z	2018-119T12:10:55Z	1	
ISS LIS SC P0.2 20180501 NQC 07545.hdf	[May 01]	2018-121T10:28:41Z	2018-121T12:01:16Z	1	
ISS LIS SC P0.2 20180509 NQC 07667.hdf	[May 09]	2018-129T06:44:42Z	2018-129T08:17:17Z	1	
ISS LIS SC P0.2 20180523 NQC 07893.hdf	[May 23]	2018-143T19:31:23Z	2018-143T21:03:59Z	2	
ISS LIS SC P0.2 20180527 NQC 07954.hdf	[May 27]	2018-147T17:39:49Z	2018-147T19:12:24Z	26	
ISS LIS SC P0.2 20180529 NQC 07985.hdf	[May 29]	2018-149T17:30:18Z	2018-149T19:02:54Z	3	
ISS LIS SC P0.2 20180530 NQC 08000.hdf	[May 30]	2018-150T16:39:15Z	2018-150T18:11:51Z	2	
ISS LIS SC P0.2 20180601 NQC 08031.hdf	[Jun 01]	2018-152T16:29:43Z	2018-152T18:02:19Z	2	
ISS LIS SC P0.2 20180602 NQC 08046.hdf	[Jun 02]	2018-153T15:38:39Z	2018-153T17:11:15Z	10	
ISS LIS SC P0.2 20180604 NQC 08080.hdf	[Jun 04]	2018-155T20:06:53Z	2018-155T21:39:29Z	4	
ISS LIS SC P0.2 20180605 NQC 08092.hdf	[Jun 05]	2018-156T14:38:01Z	2018-156T16:10:37Z	7	
ISS LIS SC P0.2 20180606 NQC 08107.hdf	[Jun 06]	2018-157T13:46:56Z	2018-157T15:19:32Z	1	
ISS LIS SC P0.2 20180613 NQC 08218.hdf	[Jun 13]	2018-164T17:04:48Z	2018-164T18:37:24Z	1	
ISS LIS SC P0.2 20180712 NQC 08658.hdf	[Jul 12]	2018-193T00:08:21Z	2018-193T01:41:00Z	5	
ISS LIS SC P0.2 20180716 NQC 08723.hdf	[Jul 16]	2018-197T04:27:41Z	2018-197T06:00:17Z	3	
ISS LIS SC P0.2 20180801 NQC 08979.hdf	[Aug 01]	2018-213T15:35:13Z	2018-213T17:07:50Z	1	
ISS LIS SC P0.2 20180805 NQC 09040.hdf	[Aug 05]	2018-217T13:44:44Z	2018-217T15:17:21Z	5	
ISS LIS SC P0.2 20180807 NQC 09071.hdf	[Aug 07]	2018-219T13:35:46Z	2018-219T15:08:23Z	43	
ISS LIS SC P0.2 20180808 NQC 09090.hdf	[Aug 08]	2018-220T18:55:26Z	2018-220T20:28:03Z	10	
ISS LIS SC P0.2 20180809 NQC 09105.hdf	[Aug 09]	2018-221T18:04:38Z	2018-221T19:37:15Z	43	
ISS LIS SC P0.2 20180812 NQC 09151.hdf	[Aug 12]	2018-224T17:04:51Z	2018-224T18:37:28Z	82	
ISS LIS SC P0.2 20180813 NQC 09166.hdf	[Aug 13]	2018-225T16:14:02Z	2018-225T17:46:39Z	2	
ISS LIS SC P0.2 20180816 NQC 09212.hdf	[Aug 16]	2018-228T15:14:13Z	2018-228T16:46:49Z	18	
ISS LIS SC P0.2 20180822 NQC 09304.hdf	[Aug 22]	2018-234T13:14:28Z	2018-234T14:47:05Z	12	
ISS LIS SC P0.2 20180823 NQC 09319.hdf	[Aug 23]	2018-235T12:23:38Z	2018-235T13:56:15Z	2	
ISS LIS SC P0.2 20180831 NQC 09437.hdf	[Aug 31]	2018-243T04:04:11Z	2018-243T05:36:48Z	67	

View flash data

Start a new search

3. Copy it to a text file (say yes to any dialogue appearing when closing the txt file)

Sense títol - Llibreta					—	□	×
Fitxer Edició Format Visualització Ajuda							
File name	Start time (UTC)	End time (UTC)	Flashes				
ISS_LIS_SC_P0.2_20180808_NQC_09090.hdf	[Aug 08]	2018-220T18:55:26Z	2018-220T20:28:03Z	5			
ISS_LIS_SC_P0.2_20180809_NQC_09105.hdf	[Aug 09]	2018-221T18:04:38Z	2018-221T19:37:15Z	5			
ISS_LIS_SC_P0.2_20180831_NQC_09437.hdf	[Aug 31]	2018-243T04:04:11Z	2018-243T05:36:48Z	87			
ISS_LIS_SC_P0.2_20180912_NQC_09636.hdf	[Sep 12]	2018-255T23:13:40Z	2018-256T00:46:20Z	1			
ISS_LIS_SC_P0.2_20180918_NQC_09716.hdf	[Sep 18]	2018-261T02:42:36Z	2018-261T04:15:13Z	4			

4. Select space-time domain in MATLAB code

```

%coordinates info
deltebre=[40.7212388 0.7176492];
santamarta=[11.2403547 -74.2110227];
barranca=[7.06878 -73.744418];

%scanning_area specification
centroid=barranca; %LAT/LON (remember, on the plot, this would be y and x)
range=60*sqrt(2); %range in km (the sqrt(2)) is to get for sure a squared
%time interval
starttime=datetime(2018,6,1,0,0,0);
endtime=datetime(2018,7,3,8,0,0);
timerange=[starttime endtime];

```

5. Set the variable "website_filename" as the directory where you stored the txt file with the HDF files' names.
6. Execute the code and select option 7 when asked by the command window. This will create a file named interesting URLs.txt in the directory where the txt file with the names was.
7. Open the Windows CMD and introduce the following command:
`wget -user EarthDatauser -ask-password -auth-no-challenge -no-check-certificate -i interestingURLSfile.txt2`
 The HDF files will be downloaded in your current directory.

3.2 Process HDF files to txt files

1. Set the directories: write_dir, read_dir

```

% #####USER MODIFIABLE VARIABLES #####
read_dir='C:\Users\icar\Desktop\GIS_HDF_files\Barrancabermeja\Barranca HDF';
write_dir_scilab='C:\Users\icar\Desktop\GIS_HDF_files\Barrancabermeja\Barranca_txt_Jun_Jul';
write_dir='C:\Users\icar\Desktop\GIS_HDF_files\Barrancabermeja\Barranca_txt_Jun_Jul';
read_dir_txtfiles=write_dir; %tm we read the txt files from the same place we store them
urls_file='C:\Users\icar\Google Drive\PRACTIQUES\HDF_reading\reading_txt_files\GHRC_URLs.txt';
connected_urls_file.write_dir='C:\Users\icar\Desktop\GIS_HDF_files\Barrancabermeja'; %here will also go the file twith urls from the website names file
relevant_orbits_file='C:\Users\icar\Google Drive\PRACTIQUES\HDF_reading\reading_txt_files\relevant_orbits.txt';
website_filename='C:\Users\icar\Desktop\GIS_HDF_files\Barrancabermeja\filenames_website.txt';
map_folder='C:\Users\icar\Google Drive\PRACTIQUES\LMAvsLIS (sci oscar)\Sci_program\mapfolder';

write_other_info_dir='C:\Users\icar\Desktop\GIS_HDF_files\Ebre\Ebre_other_info\'; %specify this to save the workspace and other info there

```

2. Select space-time domain in MATLAB code

²To use this command a user login has to be made into the Earth Data website

```

%coordinates info
deltebre=[40.7212388 0.7176492];
santamarta=[11.2403547 -74.2110227];
barranca=[7.06878 -73.744418];

%scanning_area specification
centroid=barranca; %LAT/LON (remember, on the plot, this would be y and x)
range=60*sqrt(2); %range in km (the sqrt(2)) is to get for sure a squared
%time interval
starttime=datetime(2018,6,1,0,0,0);
endtime=datetime(2018,7,3,8,0,0);
timerange=[starttime endtime];

```

3. Execute the code and enter option "1" when asked by command window

3.3 Process NC files

1. Set the reading/writing directories

```

% #####USER MODIFIABLE VARIABLES #####
read_dir='C:\Users\icar\Desktop\GLM_nc_files\randomdomain\ncfiles';
write_dir='C:\Users\icar\Desktop\GLM_nc_files\randomdomain\txtfiles';

```

2. select space-time domain

```

%coordinates info
deltebre=[40.7212388 0.7176492];
santamarta=[11.2403547 -74.2110227];
barranca=[7.06878 -73.744418];

%scanning_area specification
centroid=barranca; %LAT/LON (remember, on the plot, this would be y and x)
range=3000*sqrt(2); %range in km
%time interval
starttime=datetime(2011,6,1,0,0,0);
endtime=datetime(2020,7,3,8,0,0);
timerange=[starttime endtime];

```

3. Execute the program

3.4 LMA vs LIS comparator

1. Set the Reading directories

2. Set the time domain

3. Set the time step (etimestep)

```
sources_data_file = 'C:\Users\icar\Google Drive\LIGHTNING\10minPeriodData\1710181700\171018_1700.txt';
events_data_file = 'C:\Users\icar\Desktop\LIH_HDF_files\Ebre\Ebre_txt\ISS_LIS_20171018_1701_1701_events.txt';
read_workspace_dir = 'C:\Users\icar\Desktop\LIH_HDF_files\Ebre\Ebre_other_info\workspace_40.7_0.7_171018_180713.mat';
corrected_file='C:\Users\icar\Desktop\txt4database\corrected_file.txt';

% addpath('C:\Users\icar\Desktop\LIH_HDF_files\Ebre\Ebre_txt\');
% addpath('C:\Users\icar\Desktop\Data_from_scilab\');

%starttimes/endtimes
%ebre
%[18-Oct-2017 10:32:46 18-Oct-2017 10:34:31]
%[18-Oct-2017 17:01:25 17:01:28]

%barranca
%[18-Jul-03 6:40:14 6:41:18]
%[18-Jun-08 3:56:04 3:56:38]

etimestep=10e-3; %specify the bin duration
```

4. Select Modes: Recommended setup as shown in the figure below

```
correcting=0; %1 if the database is used to correct postion
toinput=0; %1 if the user wants to input the start time and endtime of fov. If 0 the program will take the min(listime) and max(endtime). THIS IS BAD
storedinput=1; %the same as above but previously saved in fovtime:

%fovtime=[datetime(2017,10,18,10,32,46) datetime(2017,10,18,10,34,31)];
fovtime=[datetime(2017,10,18,17,01,25), datetime(2017,10,18,17,01,28)];

plotting=1;
save_workspace=0; %if you want to write workspace. SAFEMODE=0
savingcsvfile=0;
```

5. Execute the program

A.6 Codes for data processing

Below are displayed the MATLAB codes produced for this project. They are also available at <https://github.com/icarfontcu/LIS-LMA-data-reading-codes>.

A.6.1 LIS_HDF_processor.m

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %// LIS HDF4 data processor      //
3  %//      MATLAB converter      //
4  %// Universitat Politcnica de Catalunya //
5  %//      icar.fontcuberta@gmail.com //
6  %//                               //
7  %// Requires: MATLAB R2017b at least //
8  %//                               //
9  %//                               //
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12
13
14
15 clear all;
16 close all;
17 clc;
18
19 % #####USER MODIFIABLE VARIABLES #####
20 read_dir='/Users/Icar/Desktop/transfer/EUMETSAT/1.DE/HDFprova/'; %from where to read ...
    HDF files
21 write_dir_scilab='C:\Users\icar\Desktop\LIS.HDF_files\Barrancabermeja\Barranca.txt-Jun-Jul';
22 write_dir='C:\Users\icar\Desktop\LIS.HDF_files\Barrancabermeja\Barranca.txt-Jun-Jul';
23 read_dir_txtfiles=write_dir; %atm we read the txt files from the same place we store ...
    them
24 urls_file='C:\Users\icar\Google ...
    Drive\PRACTIQUES\HDF.reading\reading.txt_files\GHRC.URLs.txt';
25 corrected_urls_file_write_dir='C:\Users\icar\Desktop\LIS.HDF_files\Barrancabermeja\'; ...
    %here will also go the file twith urls from the website names file
26 relevant_orbits_file='C:\Users\icar\Google ...
    Drive\PRACTIQUES\HDF.reading\reading.txt_files\relevant_orbits.txt';
27 website_filename='C:\Users\icar\Desktop\LIS.HDF_files\Barrancabermeja\filenames.website.txt';
28 map_folder='C:\Users\icar\Google Drive\PRACTIQUES\LMAs\LIS (sci ...
    oscar)\Sci.program\mapfolder';
29 %-----
30
31 %For adding LIS FOV to LMA files:
32 ISS_LIS_datafile = '/Users/Icar/Desktop/transfer/EUMETSAT/1.DE/ISS_LIS/ISS_LIS.mat'; ...
    %file containing the mat with all LIS data
33 LMA_data_dir = '/Users/Icar/Desktop/transfer/EUMETSAT/1.DE/LMA/'; %Where the LMA txt ...
    files are and the corrected ones will be written
34 HDF_file = ...
    '/Users/Icar/Desktop/transfer/EUMETSAT/1.DE/HDFprova/ISS_LIS.SC.P0.2.20180809_NQC-09105.hdf';
35 day_of_the_pass = 180809;
36
37 %Other-----
38 write_other_info_dir='C:\Users\icar\Desktop\LIS.HDF_files\Ebre\Ebre.other.info\'; ...
    %specify this to save the workspace and other info there
39 %-----
40
41 %coordinates info
42 deltebre=[40.75 0.95];
43 santamarta=[11.2403547 -74.2110227];
44 barranca=[7.06878 -73.744418];
45
46 %scanning_area specification
47 centroid=deltebre; %LAT/LON (remember, on the plot, this would be y and x)
48 range=60*sqrt(2); %range in km. With the range provided (box of LAT: 40.1 to 41.4, ...
    LON: 0.4 to 1.5) this should be 46km approx. But tolerance can be applied
49 %time interval
50 starttime=datetime(2018,6,1,0,0,0);
51 endtime=datetime(2018,7,3,8,0,0);

```

```

52 timerange=[starttime endtime];
53
54
55 #####
56
57 %correction for change in radius. extracted from https://rechneronline.de/earth-radius/
58
59
60 B=centroid(1);
61 r1= 6378.137;%radius at the equator
62 r2=6356.752;%radius at the poles
63 earth_radius=sqrt((((r1^2)*cos(B))^2+((r2^2)*sin(B))^2)/((r1*cos(B))^2+(r2*sin(B))^2)); ...
    %radius at your location
64
65 ang_range=range/earth_radius*360/(2*pi); %range in degrees
66
67 disp(' ');
68 disp('-----');
69 disp('Welcome to the ISS LIS data processor. ');
70 disp('Do you want to: ');
71 disp(' ');
72
73 disp('0: Exit Program');
74 disp('1: Write general event txt files');
75 disp('2: Write general event txt and plot them');
76 disp('3: Write event txt files for scilab (TO VERIFY) ');
77 disp('4: Plot events in interesting time-space from HDF4 files (TO VERIFY) ');
78 disp('5: Plot events in interesting time-space from .txt files ');
79 disp('6: Correct GHRCs URLs and generate a new URLs txt file');
80 disp('7: Process website filenames to list of interesting URLs');
81 disp('8: Check only for interesting files and save the workspace. ');
82 disp('9: Add LIS FOV to LMA txt files ');
83 disp(' ');
84 n=input('Enter option: ');
85
86
87
88 isok=false;
89
90 while isok==false
91     switch n
92         case 0
93             interestingfiles=[];
94             isok=true;
95         case 1
96
97             dir_list=[];
98             dir_list=check_for_folders(read_dir,dir_list);
99             interestingfiles=[]; a=1;
100             corruptfiles=[]; b=1;
101
102             for i=1:size(dir_list,1)
103
104
105
106
107                 local_read_dir=dir_list(i,:);
108                 [interestingfiles, ...
                    corruptfiles,a,b]=select_interestingfiles(local_read_dir,centroid, ...
                        timerange, ...
                        ang_range,interestingfiles,corruptfiles,a,b,write_other_info_dir);
109
110             end
111
112             w_txtfiles(interestingfiles,write_dir,centroid,ang_range,timerange,n); %in ...
                this function we will calibrate the events because we don't have the ...
                scilab post-processing program
113             isok=true;
114         case 2
115
116             dir_list=[];
117             dir_list=check_for_folders(read_dir,dir_list);
118             interestingfiles=[]; a=1;

```

```

119         corruptfiles=[]; b=1;
120
121     for i=1:size(dir_list,1)
122
123         local_read_dir=dir_list(i,:);
124         [interestingfiles, ...
            corruptfiles,a,b]=select_interestingfiles(local_read_dir,centroid, ...
            timerange, ...
            ang_range,interestingfiles,corruptfiles,a,b,write_other_info_dir);
125
126     end
127
128
129
130     w_txtfiles(interestingfiles,write_dir,centroid,ang_range,timerange,n); %in ...
        this function we will calibrate the events because we don't have the ...
        scilab post-processing program
131     plot_coastline(map_folder,centroid,ang_range);
132
133     isok=true;
134 case 3
135
136         dir_list=[];
137         dir_list=check_for_folders(read_dir,dir_list);
138         interestingfiles=[]; a=1;
139         corruptfiles=[]; b=1;
140
141     for i=1:size(dir_list,1)
142
143         local_read_dir=dir_list(i,:);
144         [interestingfiles, ...
            corruptfiles,a,b]=select_interestingfiles(local_read_dir,centroid, ...
            timerange, ...
            ang_range,interestingfiles,corruptfiles,a,b,write_other_info_dir);
145
146     end
147
148
149
150     w_txtfiles_4scilab(interestingfiles,write_dir_scilab);
151     isok=true;
152 case 4
153
154         dir_list=[];
155         dir_list=check_for_folders(read_dir,dir_list);
156         interestingfiles=[]; a=1;
157         corruptfiles=[]; b=1;
158
159     for i=1:size(dir_list,1)
160
161         local_read_dir=dir_list(i,:);
162         [interestingfiles, ...
            corruptfiles,a,b]=select_interestingfiles(local_read_dir,centroid, ...
            timerange, ...
            ang_range,interestingfiles,corruptfiles,a,b,write_other_info_dir);
163
164     end
165
166     read_hdf4_and_plot(interestingfiles,centroid,ang_range,timerange);
167     plot_coastline(map_folder,centroid,ang_range);
168
169     isok=true;
170
171 case 5 %disp('5: Plot events in interesting time-space from .txt files (TO ...
        VERIFY)');
172
173         dir_list=[];
174         dir_list=check_for_folders(read_dir_txtfiles,dir_list);
175         interestingfiles=[]; a=1;
176         corruptfiles=[]; b=1;
177
178     for i=1:size(dir_list,1)
179

```

```

180         local_read_dir=dir_list(i,:);
181
182         read_txt_and_plot(local_read_dir,centroid,ang_range,timerange);
183
184     end
185     plot_coastline(map_folder,centroid,ang_range);
186     disp('All events in the interesting space-time domain plotted');
187
188     isok=true;
189     case 6
190
191         isok=true;
192         correct_hdf_urls(urls_file,relevant_orbits_file,corrected_urls_file_write_dir);
193         %if you have a list of relevant orbits computed with the py program
194     case 7
195
196         isok=true;
197
198         disp('Have you entered a correct space-time domain?');
199         disp('Write dir for the txt file?');
200         webfilenames2urls(corrected_urls_file_write_dir,website_filename);
201         %if you have a file taken manually from the data from LIS website
202
203     case 8
204
205         isok=true;
206
207         dir_list=[];
208         dir_list=check_for_folders(read_dir,dir_list);
209         interestingfiles=[]; a=1;
210         corruptfiles=[]; b=1;
211
212         for i=1:size(dir_list,1)
213
214             local_read_dir=dir_list(i,:);
215             [interestingfiles, ...
                corruptfiles,a,b]=select_interestingfiles(local_read_dir,centroid, ...
                timerange, ...
                ang_range,interestingfiles,corruptfiles,a,b,write_other_info_dir);
216
217         end
218
219     case 9
220
221         isok = true;
222
223         add_FOV_to_LMA_function(day_of_the_pass,HDF_file, LMA_data_dir, ...
                ISSLIS_datafile);
224
225
226     otherwise
227         n=input('Bad entry. Rechoice option: ');
228
229 end
230
231 end
232
233
234 disp(' ');
235 disp('-----End of execution -----');
236 disp('Bye!');
237
238
239 %-----
240 % -----FUNCTIONS-----
241 %-----
242 %Found folders where the files are
243 function [dir_list]=check_for_folders(read_dir,dir_list)
244 disp('Looking for folders with HDF4 files inside');
245 addpath(read_dir); %current reading directory
246 folderinfoprev=dir(read_dir);
247 folderinfo=folderinfoprev(~ismember({folderinfoprev.name},{'.','..','.DS_Store'})); ...
    %DS_Store is a metadata file created by iOS environment

```

```

248
249 aretherefolders=cell2mat({folderinfo.isdir});
250
251 if any(aretherefolders)==true %check if there are folders inside the folder
252
253     namesarray={folderinfo(aretherefolders).name};
254     %gives a cell array but in char
255
256     %foldernames=convertCharsToStrings(namesarray); %lets convert it to string
257
258     for i=1:length(namesarray)
259 % new_read_dir(i)=fullfile(read_dir,foldernames(i));
260         new_read_dir=(fullfile(read_dir,(namesarray{i})));
261
262         [dir_list]=check_for_folders(new_read_dir,dir_list); %if the folder contains ...
                more folders, re-check
263
264     end
265
266 else
267
268     dir_list=[dir_list; read_dir]; %if the folder is a file folder save its ...
                directory and make it travel through the function
269
270
271 end
272
273 disp('Read directories will be:');
274     for i=1:size(dir_list,1)
275         disp(dir_list(i,:));
276     end
277     disp(' ');
278 end
279 %Select interesting files from those folders
280 function [interestingfiles, ...
            corruptfiles,a,b]=select_interestingfiles(read_dir,centroid, timerange, ...
            ang_range,interestingfiles,corruptfiles,a,b,write_other_info_dir)
281
282 %-----NOTATION-----%
283 %k=index of current file. all files inside folder, interesting files or not
284 %a=index of file in the interesting files subgroup
285 %b=index for possible cannot-read files inside the folder
286 %i = flash index inside the HDF file
287 %-----%
288 disp('Selecting the interesting files from the interesting folders. This might take ...
        a while, depending on the n of HDF4 files.');
```

```

289 addpath(read_dir);
290 folderinfoprev=dir(read_dir);
291 folderinfo=folderinfoprev(~ismember({folderinfoprev.name},{'.','..','.DS_Store'})); ...
        %.DS_Store is a metadata file created by iOS environment
292
293 totalnooffiles=size(folderinfo,1);
294 for k=1:size(folderinfo,1)
295     %get the names of all files inside your data directory
296     filename=folderinfo(k).name;
297
298     fileinfo=hdfinfo(filename);
299     %try to read, the file may be corrupt/empty
300     try
301         event_vdata=hdfread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(4));
302
303     catch
304
305         %{
306         disp(['The file ' filename ' with index ' num2str(k) ' could not be read.']);
307         disp(' ');
308         %{
309         corruptfiles(b).Filename=filename;
310         corruptfiles(b).File_index=k;
311         b=b+1;
312
313         continue
314     end

```

```

315     viewtime_vdata=hdfread(fileinfo.Vgroup.Vgroup.Vdata(2));
316     fovinfo(k).fov_coordinates=viewtime_vdata{1};
317     fovinfo(k).fovstart=viewtime_vdata{2};
318     fovinfo(k).fovend=viewtime_vdata{3};
319
320
321     event.coordinates=event_vdata{3};
322     event.tai_time=event_vdata{1};
323
324     time=datetime(1993,1,1)+seconds(event.tai_time);
325     lats=event.coordinates(1,:);
326     lons=event.coordinates(2,:);
327
328     ang_distances=sqrt((lats-centroid(1)).^2+(lons-centroid(2)).^2);
329
330     space_ind=find(ang_distances<ang_range);%get indexes of space dominum
331     time_ind=intersect(find(time>timerange(1)),find(time<timerange(2)));
332     %get indexes of time dominum. Intersect gives common values, so
333     %here its ok because finds gives indexes and we want to find
334     %common indexes
335
336     [view_ind]=intersect(space_ind,time_ind); %relevant elements of each file
337     %intersect both indexes in order to find positios that meet
338     %space-time dominum
339
340     if ~isempty(view_ind)
341
342         interestingfiles(a).Filename=filename;
343         interestingfiles(a).File_index=k;
344
345         a=a+1;
346
347     end
348
349
350     %{
351
352     i=1;
353     eventinsiderange=false; eventinsidetime=false;
354     while i<size(event.coordinates,2) && ~(eventinsiderange && eventinsidetime)
355         eventinsiderange=false; eventinsidetime=false;
356
357         %check space domain
358         coordinates=event.coordinates(:,i);
359         c_distance=sqrt((coordinates(1)-centroid(1))^2+(coordinates(2)-centroid(2))^2);
360
361         if c_distance<ang_range
362             eventinsiderange=true;
363         end
364
365         %check time domain
366         time=datetime(1993,1,1)+seconds(event.tai_time(i));
367         if time<timerange(2) && time>timerange(1)
368             eventinsidetime=true;
369         end
370
371         %if both are positive, this file is interesting, and we dont need to
372         %further loop
373         if eventinsidetime==true && eventinsiderange==true
374
375             interestingfiles(a).Filename=filename;
376             interestingfiles(a).File_index=k;
377
378             a=a+1;
379
380         end
381
382         i=i+1;
383
384     end
385     %}
386     disp([num2str(k*100/totalnoffiles) '% of the interesting files checking inside ' ...
        read_dir ' is done.']);

```

```

387
388 end
389
390 %save FOV info from these interesting files
391 disp(' ');
392 disp('Saving Workspace...');
393 save([write_other_info_dir 'workspace_' num2str(centroid(1)) '_' ...
      num2str(centroid(2)) '_' datestr(timerange(1),'yymmdd') '_' ...
      datestr(timerange(2),'yymmdd') '.mat'] , 'fovinfo', 'interestingfiles', ...
      'corruptfiles', 'centroid', 'ang_range', 'timerange');
394
395 if isempty(interestingfiles)
396     disp([num2str(length(interestingfiles)) ' interesting files have been found']);
397 else
398     disp(['No interesting files were found in ' read_dir]);
399 end
400
401 end
402 %Make .txt for scilab files from the interesting files
403 function wtxtfiles_4scilab(interestingfiles,write_dir)
404     addpath(write_dir);
405
406
407     for k=1:length(interestingfiles)
408
409         filename=interestingfiles(k).Filename;
410         fileinfo=hdffinfo(filename);
411
412         event_vdata=hdffread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(4));
413
414         filetextname=[erase(filename, '.hdf'), '_event4scilab', '.txt'];
415         fullfilename=fullfile(write_dir, filetextname);
416
417         tai.time=event_vdata{1}; %get the instant, for comparison pruposes
418         observe.time=event_vdata{2};
419         location=event_vdata{3};
420         radiance=event_vdata{4};
421         footprint=event_vdata{5};
422         address=event_vdata{6}+1; %add +1 so it doesn't start w/ 0 adrexx
423         parent.address=event_vdata{7}+1;
424         x.pixel=event_vdata{8};
425         y.pixel=event_vdata{9};
426         bg.value=event_vdata{10};
427         bg.radiance=event_vdata{11};
428         amplitude=event_vdata{12};
429         sza.index=event_vdata{13};
430         glint.index=event_vdata{14};
431         approx.threshold=event_vdata{15};
432         alert.flag=event_vdata{16};
433         cluster.index=event_vdata{17};
434         density.index=event_vdata{18};
435         noise.index=event_vdata{19};
436         bg.value.flag=event_vdata{20};
437         grouping.sequence=event_vdata{21};
438
439
440         fileID=fopen(fullfilename, 'w');
441         if fileID==-1
442             disp('Could not open file!');
443
444         else
445             fprintf(fileID, 'TAI93.time observe.time latitude longitude radiance ...
              footprint address parent.address x.pixel y.pixel bg.value bg.radiance ...
              amplitude sza.index glint.index approx.threshold alert.flag ...
              cluster.index density.index noise.index bg.value.flag ...
              grouping.sequence\r\n');
446             for i=1:length(tai.time)
447                 fprintf(fileID, '%-18.16E %2d %7.3f %7.3f %4.1d %3.1d %4u %3u %3u %3u %4u ...
              %3u %3u %3u %3u %3u %1u %2u %2u %3d %1u ...
              %6u\r\n', tai.time(i), observe.time(i), location(1,i), location(2,i), radiance(i), footprint(i),
448             end
449
450             fclose(fileID);

```

```

451         disp(['Printing file ' filename ' @ ' write_dir]);
452     end
453
454     end
455
456
457
458 end
459 function w_txtfiles(interestingfiles,write_dir,centroid, ang_range,timerange,n)
460 disp('Printing general events .txt files...');
461
462
463 ninterestingfiles=length(interestingfiles); %added constant for % use
464 for k=1:length(interestingfiles) %We will go through all interestingfiles
465     %and search for interestingevents
466     clear area.vdata flash.vdata group.vdata event.vdata
467
468     filename=interestingfiles(k).Filename;
469     fileinfo=hdfinfo(filename);
470
471     event.vdata=hdfread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(4));
472     event.location=event.vdata{3};
473     event.tai.time=event.vdata{1};
474
475     area.vdata=hdfread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(1));
476     flash.vdata=hdfread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(2));
477     group.vdata=hdfread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(3));
478     event.vdata=hdfread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(4));
479
480     event.location=event.vdata{3};
481     event.tai.time=event.vdata{1};
482     event.observe.time=event.vdata{2};
483     event.radiance=event.vdata{4};
484     event.address=event.vdata{6}+1; %add +1 so it doesn't start w/ 0 adress
485     event.parent.address=event.vdata{7}+1;
486     event.bg.radiance=event.vdata{11};
487
488     event.x.pixel=event.vdata{8};
489     event.y.pixel=event.vdata{9};
490     event.bg.rad=event.vdata{11};
491
492     group.parent.address=group.vdata{7}+1; %no need to store group.address as ...
493     %they are ordered by number inside each file
494     group.location=group.vdata{3};
495
496     flash.parent.address=flash.vdata{8}+1;
497     flash.location=flash.vdata{4};
498
499     area.location=area.vdata{4};
500     area.observetime=area.vdata{3};
501
502     orbit.vdata=hdfread(fileinfo.Vgroup.Vdata(1));
503     orbit.id=orbit.vdata{1};
504
505
506     %We could size this info for printing and plotting, but better not
507     %to change what works
508     time=datetime(1993,1,1)+seconds(event.tai.time);
509     lats=event.location(1,:);
510     lons=event.location(2,:);
511
512     ang.distances=sqrt((lats-centroid(1)).^2+(lons-centroid(2)).^2);
513
514     space_ind=find(ang.distances<ang_range);%get indexes of space dominum
515     time_ind=intersect(find(time>timerange(1)),find(time<timerange(2)));
516     %get indexes of time dominum. Intersect gives common values, so
517     %here its ok because finds gives indexes and we want to find
518     %common indexes
519
520     [view_ind]=intersect(space_ind,time_ind); %relevant elements of each file
521     %intersect both indexes in order to find positios that meet
522     %space-time dominum

```



```

523         starttime=time(min(view_ind));
524         starttime=datestr(starttime,'HHMM');
525         endtime=time(max(view_ind));
526         endtime=datestr(endtime,'HHMM');
527         %-----
528
529     %prepare for writing interestingevents
530     q=erase(filename,'.hdf');
531     q=erase(q,'_NQC');
532     erasethis=['_' num2str(orbit_id,'%05i')];
533     q=erase(q,erasethis);
534     q=erase(q,'_SC.P0.2');
535     filetextname=[q,'_' starttime, '_', endtime,'_events','.txt'];
536     fullfilename=fullfile(write_dir,filetextname);
537     fileID=fopen(fullfilename,'w');
538
539     event_properties{1,17}=[];
540     if fileID==-1
541         disp('Could not open file!');
542
543     else
544
545         %disp(['Printing file n' num2str(k) ' ', ' filetextname ' @ ' write_dir']);
546         disp([num2str(k*100/ninterestingfiles) '% of .txt files printed']);
547
548         %Write Header of the file
549
550         fprintf(fileID, 'TAI93time e_lat e_lon e_radiance group g_lat g_lon flash ...
                    f_lat f_lon area a_lat a_lon a_observe_time x_pixel y_pixel ...
                    bg_radiance\r\n');
551
552
553
554
555
556
557         j=1; %n of intersting events in each file
558         for i=1:length(event.tailtime) %HERE THE PROGRAM PRINTS
559
560             eventinsidetime=false;
561             eventinsiderange=false;
562
563             %Is this event interesting?
564             %-----
565             coordinates=event.location(:,i);
566             c_distance=sqrt((coordinates(1)-centroid(1))^2+(coordinates(2)-centroid(2))^2);
567             if c_distance<ang_range
568                 eventinsiderange=true;
569             end
570
571             %check time domain
572             time=datetime(1993,1,1)+seconds(event.tailtime(i));
573             if time<timerange(2) && time>timerange(1)
574                 eventinsidetime=true;
575             end
576             %-----
577
578             if eventinsidetime==true && eventinsiderange==true
579
580
581                 %Make new clear info with time event, its radiance, adress, group
582                 %adress, flash adress and area adress
583
584                 tailtime=event.tailtime(i);
585                 event_lat=event.location(1,i);
586                 event_lon=event.location(2,i);
587                 radiance=event.radiance(i);
588                 bg_rad=event.bg_rad(i);
589                 x_pixel=event.x_pixel(i);
590                 y_pixel=event.y_pixel(i);
591
592                 egroup=event.parent_address(i);%his group address
593                 group_lat=group.location(1,egroup);

```

```

594         group_lon=group.location(2,egroup);
595
596         eflash=group.parent_address(egroup);%his flash address
597         flash_lat=flash.location(1,eflash);
598         flash_lon=flash.location(2,eflash);
599
600         earea=flash.parent_address(eflash);%his area address
601         area_lat=area.location(1,earea);
602         area_lon=area.location(2,earea);
603         area_observetime=area.observetime(earea);
604
605         fprintf(fileID, '%-18.16E %7.3f %7.3f %4.1d %u %7.3f %7.3f %u %7.3f %7.3f ...
        %u %7.3f %7.3f %u %u %u ...
        %u\r\n',tai.time,event_lat,event_lon,radiance,egroup,group_lat,group_lon,eflash,flash_lat,
606
607         a={ ...
        tai.time,event_lat,event_lon,radiance,egroup,group_lat,group_lon,eflash,flash_lat,flash_lo
608         event_properties(j,:)=a;
609         %the group, event flash and area address will not be ordered nor
610         %listed from 1 to X, maybe from 543 to 987. That's because the
611         %parent address of an event is reset in each file (1-to-end), but
612         %when we recheck if an event is inside the spacew-time dominum we
613         %will eliminate, for example, all events that were part of de
614         %groups 1 to 453.
615
616         if isempty(event_properties{j,4})
617             disp('Radiance empty. code line 598 aprox');
618         end
619
620         j=j+1;
621
622     end
623
624
625 end
626
627     fclose(fileID);
628
629
630
631 end
632
633
634     if n==2
635         disp('Plotting events form the current file...');
636         plot_events(event_properties,centroid,ang.range);
637
638     end
639 end
640
641
642 end
643
644 function read.hdf4.and.plot(interestingfiles,centroid,ang.range,timerange)
645
646 disp('Reading data from interestingfiles and plotting...');
647 for k=1:length(interestingfiles) %We will go through all interestingfiles
648     %and search for interestingevents
649
650     eventinsiderange=false;
651     eventinsidetime=false;
652
653
654     filename=interestingfiles(k).Filename;
655     fileinfo=hdffinfo(filename);
656
657     event_vdata=hdffread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(4));
658     event.location=event_vdata{3};
659     event.tai.time=event_vdata{1};
660
661     area_vdata=hdffread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(1));
662     flash_vdata=hdffread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(2));
663     group_vdata=hdffread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(3));

```

```

664         event.vdata=hdfread(fileinfo.Vgroup.Vgroup.Vgroup.Vdata(4));
665
666         event.location=event.vdata{3};
667         event.tai.time=event.vdata{1};
668         event.tai.time=event.vdata{1};
669         event.observe.time=event.vdata{2};
670         event.radiance=event.vdata{4};
671         event.address=event.vdata{6}+1; %add +1 so it doesn't start w/ 0 address
672         event.parent.address=event.vdata{7}+1;
673         event.bg.radiance=event.vdata{11};
674
675
676         group.parent.address=group.vdata{7}+1; %no need to store group.address as ...
        they are ordered by number inside each file
677         group.location=group.vdata{3};
678
679         flash.parent.address=flash.vdata{8}+1;
680         flash.location=flash.vdata{4};
681
682         area.location=area.vdata{4};
683         area.observetime=area.vdata{3};
684
685         %prepare for writing interesting events
686
687         %Write Header of the file
688
689
690         event.properties{1,14}=[];
691         for i=1:length(event.tai.time)
692
693
694         %Is this event interesting?
695         %-----
696             coordinates=event.location(:,i);
697             c.distance=sqrt((coordinates(1)-centroid(1))^2+(coordinates(2)-centroid(2))^2);
698             if c.distance<ang.range
699                 event.insiderange=true;
700             end
701
702             %check time domain
703             time=datetime(1993,1,1)+seconds(event.tai.time(i));
704             if time<timerange(2) && time>timerange(1)
705                 event.insidetime=true;
706             end
707         %-----
708             if event.insidetime==true && event.insiderange==true
709
710
711             %Make new clear info with time event, its radiance, address, group
712             %address, flash address and area address
713
714             tai.time=event.tai.time(i);
715             event.lat=event.location(1,i);
716             event.lon=event.location(2,i);
717             radiance=event.radiance(i);
718
719             egroup=event.parent.address(i); %his group address
720             group.lat=group.location(1,egroup);
721             group.lon=group.location(2,egroup);
722
723             eflash=group.parent.address(egroup); %his flash address
724             flash.lat=flash.location(1,eflash);
725             flash.lon=flash.location(2,eflash);
726
727             earea=flash.parent.address(eflash); %his area address
728             area.lat=area.location(1,earea);
729             area.lon=area.location(2,earea);
730             area.observetime=area.observetime(earea);
731
732             a={ ...
733                 tai.time,event.lat,event.lon,radiance,egroup,group.lat,group.lon,eflash,flash.lat,flas
734             end

```

```

735             eventinsiderange=false;
736             eventinsidetime=false;
737         end
738     end
739     plot_events(event_properties,centroid,ang_range);
740 end
741
742
743
744
745 end
746
747 function read_txt_and_plot(read_dir,centroid,ang_range,timerange)
748
749 addpath(read_dir);
750 folderinfoprev=dir(read_dir);
751 folderinfo=folderinfoprev(~ismember({folderinfoprev.name},{'.','..','DS_Store'})); ...
    %DS_Store is a metadata file created by iOS environment
752
753 for k=1:size(folderinfo,1)
754     filename=folderinfo(k).name;
755
756     data=import_event_LIS_txt_files(filename);
757     event_properties=[];
758     time=datetime(1993,1,1)+seconds(data(:,1));
759     lats=data(:,2);
760     lons=data(:,3);
761
762     ang_distances=sqrt((lats-centroid(1)).^2+(lons-centroid(2)).^2);
763
764     space_ind=find(ang_distances<ang_range);
765     time_ind=intersect(find(time>timerange(1)),find(time<timerange(2)));
766
767     view_ind=intersect(space_ind,time_ind); %relevant elements of each file
768
769     event_properties(:,:)=data(view_ind,:);
770
771     plot_events(event_properties,centroid,ang_range);
772
773     %{
774     for i=1:size(data,1)
775         eventinsiderange=false; %for each element we initialise inside checking
776         eventinsidetime=false;
777
778         coordinates=[data(i,2),data(i,3)];
779         c_distance=sqrt((coordinates(1)-centroid(1)).^2+(coordinates(2)-centroid(2)).^2);
780         if c_distance<ang_range
781             eventinsiderange=true;
782         end
783
784         %check time domain
785         time=datetime(1993,1,1)+seconds(data(i,1));
786         if time<timerange(2) && time>timerange(1)
787             eventinsidetime=true;
788         end
789
790         if eventinsiderange==true && eventinsidetime==true
791             event_properties(i,:)=data(i,:);
792         end
793     end
794
795 end
796
797
798 %call this function for each file. If the .txt file contains
799 %interesting events or not doesn't matter. If event_properties is
800 %empty the function plot_events will simply do nothing.
801 %This doesn't happen with HDF4 files because in that case we
802 %first search for interesting files that they already contain
803 %interesting information.
804 %}
805
806 end

```

```

807 end
808
809 function correct_hdf_urls(urls_file,relevant_orbits_file,corrected_urls_file_dir)
810
811 disp('Reading urls and relevant orbits txt file...');
812 relevant_orbits=import_relevant_orbits(relevant_orbits_file);
813 allurls=import_urls_times(urls_file);
814 urls=import_GHRCURLs(urls_file);
815
816 disp('Reading URLs timing...');
817 for i=1:length(allurls)
818
819     date=num2str(allurls(i,1));
820     year=str2double(date(1:4));
821     month=str2double(date([5 6]));
822     day=str2double(date([7 8]));
823     orbit.days_from_urls(i)=datetime(year,month,day);
824
825
826 end
827
828 days_that_LIS_passes_the_zone=relevant_orbits(:,1);
829
830 unique_days_LIS_passes=table2cell(unique(days_that_LIS_passes_the_zone));
831
832
833 disp('Checking URLs coincidence with relevant orbits...');
834 eliminate_indexes=[];
835 for i=1:size(orbit.days_from_urls,2)
836
837     finding=false;
838     for j=1:size(unique_days_LIS_passes,1)
839         if orbit.days_from_urls(i) == unique_days_LIS_passes{j,1}
840
841             finding=true;
842
843         end
844     end
845
846     if finding==false
847         try
848
849             eliminate_indexes=[eliminate_indexes; i];
850
851         catch
852
853             disp('error line 673');
854         end
855     end
856
857     disp([num2str(i*100/size(orbit.days_from_urls,2)) '% checked.']);
858
859 end
860
861 disp(['Found ' num2str(length(eliminate_indexes)) ' non-relevant URLs']);
862 urls(eliminate_indexes,:)=[];
863
864 fullfilename=fullfile(corrected_urls_file_dir,'interesting_URLs.txt');
865 writetable(urls,fullfilename);
866 disp(['Urls .txt file written at: ' corrected_urls_file_dir]);
867
868
869 end
870 %plotters
871 function plot_coastline(read_dir,centroid,ang_range)
872
873 load coastlines;
874 addpath(read_dir);
875 disp('Plotting coastline...');
876
877
878 latlim=[ centroid(1)-ang_range centroid(1)+ang_range ];
879 lonlim=[centroid(2)-ang_range centroid(2)+ang_range];

```

```

880
881 tf=ingeoquad(coastlat,coastlon,latlim,lonlim);
882 lats=coastlat(tf);
883 lons=coastlon(tf);
884
885 hold on;
886 plot(lons,lats,'k-');
887
888
889 end
890 function plot_events(event_properties,centroid,ang-range)
891
892 hold on;
893
894     if ~isempty(event_properties)
895
896         if iscell(event_properties)
897
898             size=cell2mat(event_properties(:,4));
899             color=cell2mat(event_properties(:,5));
900             lats=cell2mat(event_properties(:,2));
901             lons=cell2mat(event_properties(:,3));
902
903
904
905             else
906
907                 size=event_properties(:,4);
908                 color=event_properties(:,5);
909                 lats=event_properties(:,2);
910                 lons=(event_properties(:,3));
911             end
912
913             plot_interesting_area(centroid,ang-range);
914             scatter(lons,lats,size,color,'x');
915             xlabel('Longitude [deg.]');
916             ylabel('Latitude [deg.]');
917             grid on;
918             title('Events detected. Size -> Radiance. Color -> group');
919             axis equal;
920
921
922         else
923             disp('Event_properties vector is empty!');
924         end
925
926
927 end
928 function plot_interesting_area(centroid,r)
929 x=centroid(2);
930 y=centroid(1);
931
932 hold on;
933 grid on;
934
935 th = 0:pi/50:2*pi;
936 xunit = r * cos(th) + x;
937 yunit = r * sin(th) + y;
938 plot(xunit, yunit,'k--','LineWidth',0.01);
939 %plot(x,y,'kx','LineWidth',0.01);
940 axis equal;
941
942 end
943
944 %LMA FOV correction. Made by Prof. Montanya, J.
945 %adapted, updated and assembled to the general code by Fontcuberta .
946
947 function add_FOV_to_LMA_function(day_of_the_pass, HDF, LMA.data_dir, LIS.data_file)
948 % J. Montanya
949 %
950 % Adds a column (Nbr 8) to LMA data where 1 = that source was in the FOV
951 %                                0 = that source was NOT in the FOV
952 %

```

```

953 %
954 %   Input:   HDF of the pass
955 %           LMA file from the Batch program
956 %           ISS_LIS.mat   (ISS LIS data file)
957 %
958 %   Output:  LMA file in 'mat' format ith the extra columns
959 %
960 %   The program uses the function  check_ISS_FOV()
961 format long g
962
963 % Troba HH:MM:SS dels llamps de ISS per tal de saber franja horaria
964 % (p.e. ho faig servir per el batch program del LMA (nomes precessar el
965 % fitxer de 10 minuts que toca)
966
967 % 20181018
968 % dia=20181018;
969 % HDF=('C:\Joan\EUMETSAT\1-DE\HDF\ISS-LIS-SC-P0.2-20181018-NQC-10190.hdf');
970 % LMA=load('C:\Joan\EUMETSAT\1-DE\LMA\181018.txt');
971
972 % 20180918
973 % dia=20180918;
974 % HDF=('C:\Joan\EUMETSAT\1-DE\HDF\ISS-LIS-SC-P0.2-20181018-NQC-10190.hdf');
975 % LMA=load('C:\Joan\EUMETSAT\1-DE\LMA\181018.txt');
976
977 dia = ['20', day-of-the-pass];
978
979 disp('Loading LMA and LIS data...');
980
981 LMA = load([LMA.data_dir, num2str(day-of-the-pass), '.txt']);
982 LIS_data=load(LIS_data_file);
983 LIS_raw=LIS_data.data;
984 clear LIS_data
985 disp('Done!');
986
987 % IMPORTANT: Anar al final a posar el nom del fitxer de sortida
988 %UPDATE: Solucionat (Icar)
989
990 [X,Y]=find(LIS_raw(:,1)==dia);
991 LIS=LIS_raw(X,:);
992
993 clear X Y LIS_raw
994
995
996 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
997 %%% First check for all LMA sources is ISS-LIS was within FOV      %
998 %%% Sets a new column (Nbr 8)  with 1=YES in FOV   0: NO in FOV    %
999 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1000
1001 [fil,col]=size(LMA);
1002
1003
1004
1005
1006 % Read HDF file used to determine if each LMA is within FOV
1007 disp('Reading selected HDF file...');
1008
1009
1010 ISS_info=hdfinfo(HDF); % Read HDF file
1011 ISS_data=hdfread(ISS_info.Vgroup.Vgroup.Vdata(2)); % Read Viewtime ...
1012 % location, start time, end time, duration
1013
1014 Location=cell2mat(ISS_data(1,1,1)); % Read location LAT LON of the ...
1015 % 0.5x0.5 center of grid cells
1016 Location=Location'; %
1017 Location=double(Location); % Change to double in order to ...
1018 % use later
1019
1020 % Time start : obtains the Start Times of the FOV for each 0.5x0.5 grid
1021 Time_start_TAI= cell2mat(ISS_data(2,1)); % TAI 93 start time
1022 time=seconds(Time_start_TAI); % Converts times to format seconds

```

```

1022     TAI-UTC = seconds(10) ; % This is the delay that matches ...
1023     with the ISS LIS data of the GHRC website
1024     time = (time - TAI-UTC);
1025
1026     date = time + datenum([1993,1,1]);
1027     date = datenum(date);
1028     myDateTime = datetime(date, 'ConvertFrom', 'datenum');
1029     myDateTime.Format = 'dd-MMM-uuuu HH:mm:ss.SSSS'; % Format of the ...
1030     myDateTime
1031
1032     % Creates the Start Time in seconds of the day (UTC)
1033     MyTime_start = hour(myDateTime)*3600 + ...
1034     minute(myDateTime)*60+second(myDateTime); % Time in seconds of the day UTC
1035
1036     % Get the day in numeric format
1037     Date.dia= year(myDateTime(1,1))*10000+month(myDateTime(1,1))*100 ...
1038     +day(myDateTime(1,1));
1039
1040     clear time date myDateTime checkDate
1041
1042     % Time end : obtains the End Times of the FOV for each 0.5x0.5 grid
1043     Time_end.TAI= cell2mat(ISS_data(3,1)); % TAI 93 end time
1044     time=seconds(Time_end.TAI);
1045     TAI-UTC = seconds(10) ;
1046     time = (time - TAI-UTC);
1047
1048     date = time + datenum([1993,1,1]);
1049     date = datenum(date);
1050     myDateTime = datetime(date, 'ConvertFrom', 'datenum');
1051     myDateTime.Format = 'dd-MMM-uuuu HH:mm:ss.SSSS';
1052
1053     % Creates the End Time in seconds of the day (UTC)
1054     MyTime_end = hour(myDateTime)*3600 + ...
1055     minute(myDateTime)*60+second(myDateTime); % Time in seconds of the day UTC
1056
1057     % Duration of observation when the ISS LIS is withtin the FOV of the ...
1058     0.5x0.5 grid cell %%%
1059
1060     Duration= cell2mat(ISS_data(4,1)); % Duration in seconds
1061
1062     % We obtained: Location MyTime_end MyTime_start
1063
1064     disp('Done!');
1065
1066     disp('Checking which sources were inside LIS FOV...')
1067     for i=1:1:fil
1068
1069         time_LMA=LMA(i,2);
1070         LATLMA=LMA(i,4);
1071         LONLMA=LMA(i,5);
1072
1073         FOV = check_ISS_FOV(time_LMA,LATLMA,LONLMA,Location,MyTime_start,MyTime_end);
1074
1075         if FOV==1
1076             LMA(i,8)=1;
1077         end
1078
1079         if FOV==0
1080             LMA(i,8)=0;
1081         end
1082
1083         clear FOV LATLMA LONLMA time_lma
1084
1085         %i
1086     end
1087     disp('Done!');
1088

```



```

1089
1090
1091
1092
1093 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1094 % PART 2: Adds Column 9 with the ID (also for the sources classified as
1095 % noise)
1096 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1097 disp('Adding a 9th column to the LMA data with the ID...');
1098 disp(' Start to relate LMA noisy sources with the flash ID')
1099 clear i X Y FOV fil col
1100
1101
1102 LMA(:,9)=LMA(:,1); % Firt copy the flash ID of column 1 to column 9
1103 % that is beacuse sources with ID not 0 will not be
1104 % analyzed
1105
1106 [fil,col]=size(LMA);
1107
1108
1109
1110 End_ID_flash=max(LMA(:,1));
1111
1112 for i=1:1:End_ID_flash
1113
1114     ID=i;
1115
1116     [X,Y]=find(LMA(:,1)==ID);
1117
1118     TimeLMAstartID=min(LMA(X(:,1),2));
1119     TimeLMAendID=max(LMA(X(:,1),2));
1120
1121     clear X Y
1122
1123     [X,Y]=find(LMA(:,1)==0 & LMA(:,2)≥TimeLMAstartID & LMA(:,2)≤TimeLMAendID );
1124
1125     LMA(X,9)=ID;
1126
1127     clear X Y ID
1128
1129
1130
1131
1132 end
1133 disp('Done!');
1134
1135 disp(['Writing data at ' LMA_data_dir '... ']);
1136
1137 LMA_txt_filename = [LMA_data_dir, 'LMA.FOV-', num2str(day-of-the-pass), '_wFOV'];
1138
1139 save([LMA_txt_filename '.mat'],'LMA');
1140
1141 save([LMA_txt_filename '.txt'],'LMA','-ascii');
1142 disp('Done!');
1143
1144 end
1145
1146
1147 function FOV = check_ISS_FOV(time_LMA,LATLMA,LONLMA,Location,MyTime_start,MyTime_end)
1148
1149 %
1150 % Cheks if a source is within the fiel of view
1151 %
1152 % Returns:
1153 %
1154 % FOV =1 if LMA source is within fov
1155 % FOV = 0 if LMA source is not within fov
1156 %
1157
1158
1159
1160
1161

```

```

1162 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1163 % Check if LMA flash could be seen by the ISS LIS
1164
1165 % Flash LMA location
1166 LAT_LMA=LATLMA;
1167 LON_LMA=LONLMA;
1168
1169 %TimeLMA= 3* 3600 + 37* 60 + 34.9; % Time of the LMA flash
1170 TimeLMA=time_LMA;
1171
1172 % Finds the closest grid square of 0.5 x0.5 in the FOV of ISS at
1173 % the LAT LON of the LMA flash location
1174 % a difference of 0.25 is allowed
1175 clear X Y
1176 [X,Y]=find( Location(:,1)<(LAT_LMA+0.25) & Location(:,1)>(LAT_LMA-0.25) & ...
1177            Location(:,2)<(LON_LMA+0.25) & Location(:,2)>(LON_LMA-0.25) );
1178 [fX,cX]=size(X);
1179
1180 % If one or several 0.5x0.5 square grids are found
1181 % selects the absolute close and then checks if the time
1182 % of the LMA flash was within the start and end time of the
1183 % ISS FOV over that square grid.
1184
1185
1186
1187
1188
1189
1190 if isempty(X)
1191
1192     %disp('No Location is found !');
1193     FOV=0;
1194 else
1195
1196     if fX==1 | Location(X(1,1),:)==Location(X,:)
1197
1198         % Pot passar(ho he comprovat) que hi ha varies
1199         % locations amb el mateix FOV de LIS, es a dir varis
1200         % Location que tenen exactament la mateixa LAT LON
1201         % i que compleixen amb distancia amb LAT/LON LMA.
1202
1203         Location(X,:);
1204
1205         MyTime.start(1,X);
1206         MyTime.end(1,X);
1207
1208         % Diff_Location= abs(Location(X,1)-LAT_LMA) + ...
1209             abs(Location(X,2)-LON_LMA) ;
1210
1211         % [MinLoc,IndexMinLocation]=min(Diff_Location(:,1));
1212
1213         % Index=X(IndexMinLocation);
1214
1215         FOV = 0;
1216
1217         for ii=1:fX
1218
1219             if TimeLMA>MyTime.start(1,X(ii,1)) && TimeLMA <MyTime.end(1,X(ii,1))
1220                 %disp('The source WAS within the field of view of the ISS');
1221                 FOV=1;
1222             end
1223
1224         end
1225
1226     end
1227
1228     if Location(X(1,1),:)!=Location(X,:)
1229
1230         disp('Source vista per varies locations (FOV de 0.5x0.5 ) diferents ...
1231             !'); % En el cas que una source estigui a multiples Location ...
1232             diferents de LIS seria un error i para el programa

```



```

1300 % Close the text file.
1301 fclose(fileID);
1302
1303 %Post processing for unimportable data.
1304 %No unimportable data rules were applied during the import, so no post
1305 % processing code is included. To generate code which works for
1306 % unimportable data, select unimportable cells in a file and regenerate the
1307 % script.
1308
1309 % Create output variable
1310 data = [dataArray{1:end-1}];
1311 end
1312
1313 function GHRCURLs = import_urls.times(filename, startRow, endRow)
1314 %IMPORTFILE Import numeric data from a text file as a matrix.
1315 % GHRCURLs = IMPORTFILE(FILENAME) Reads data from text file FILENAME for
1316 % the default selection.
1317 %
1318 % GHRCURLs = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
1319 % STARTROW through ENDROW of text file FILENAME.
1320 %
1321 % Example:
1322 % GHRCURLs = importfile('GHRC_URLs.txt', 1, 433);
1323 %
1324 % See also TEXTSCAN.
1325
1326 % Auto-generated by MATLAB on 2018/07/23 16:23:22
1327
1328 % Initialize variables.
1329 delimiter = {' ','.'};
1330 if nargin<2
1331     startRow = 1;
1332     endRow = inf;
1333 end
1334
1335 % Read columns of data as text:
1336 % For more information, see the TEXTSCAN documentation.
1337 formatSpec = '%s%s%s*s*s*s*s*s*s*s*s*s%s%[\n\r]';
1338
1339 % Open the text file.
1340 fileID = fopen(filename,'r');
1341
1342 % Read columns of data according to the format.
1343 % This call is based on the structure of the file used to generate this
1344 % code. If an error occurs for a different file, try regenerating the code
1345 % from the Import Tool.
1346 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', 'HeaderLines', ...
    startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1347 for block=2:length(startRow)
1348     frewind(fileID);
1349     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', ...
        'HeaderLines', startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1350     for col=1:length(dataArray)
1351         dataArray{col} = [dataArray{col};dataArrayBlock{col}];
1352     end
1353 end
1354
1355 %Close the text file.
1356 fclose(fileID);
1357
1358 % Convert the contents of columns containing numeric text to numbers.
1359 % Replace non-numeric text with NaN.
1360 raw = repmat({''},length(dataArray{1}),length(dataArray)-1);
1361 for col=1:length(dataArray)-1
1362     raw(1:length(dataArray{col}),col) = mat2cell(dataArray{col}, ...
        ones(length(dataArray{col}), 1));
1363 end
1364 numericData = NaN(size(dataArray{1},1),size(dataArray,2));
1365
1366 for col=[1,2,3]
1367     % Converts text in the input cell array to numbers. Replaced non-numeric

```

```

1368 % text with NaN.
1369 rawData = dataArray{col};
1370 for row=1:size(rawData, 1)
1371     % Create a regular expression to detect and remove non-numeric prefixes and
1372     % suffixes.
1373     regexstr = ...
1374         '(?<prefix>.*?)(?<numbers>([-]*(\d+[\,]*)+[\.]{0,1}\d*[eEdD]{0,1}[-+]*\d*[i]{0,1})|([-]*(\d
1375     try
1376         result = regexp(rawData(row), regexstr, 'names');
1377         numbers = result.numbers;
1378
1379         % Detected commas in non-thousand locations.
1380         invalidThousandsSeparator = false;
1381         if numbers.contains(',')
1382             thousandsRegExp = '^(\d+[\,]\d{3})*\.{0,1}\d*$';
1383             if isempty(regexp(numbers, thousandsRegExp, 'once'))
1384                 numbers = NaN;
1385                 invalidThousandsSeparator = true;
1386             end
1387         end
1388         % Convert numeric text to numbers.
1389         if ~invalidThousandsSeparator
1390             numbers = textscan(char(strrep(numbers, ',', '')), '%f');
1391             numericData(row, col) = numbers{1};
1392             raw{row, col} = numbers{1};
1393         end
1394     catch
1395         raw{row, col} = rawData{row};
1396     end
1397 end
1398
1399 % Exclude columns with non-numeric cells
1400 I = ~all(cellfun(@(x) (isnumeric(x) || islogical(x)) && ~isnan(x),raw),1); % Find ...
1401     columns with non-numeric cells
1402 raw(:,I) = [];
1403
1404 %Initialize column outputs.
1405 columnIndices = cumsum(~I);
1406
1407 % Create output variable
1408 GHRCURLs = cell2mat(raw);
1409
1410 end
1411
1412 function relevantorbits = import_relevant_orbits(filename)
1413 %IMPORTFILE Import numeric data from a text file as a matrix.
1414 % RELEVANTORBITS = IMPORTFILE(FILENAME) Reads data from text file
1415 % FILENAME for the default selection.
1416 %
1417 % RELEVANTORBITS = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from
1418 % rows STARTROW through ENDROW of text file FILENAME.
1419 %
1420 % Example:
1421 % relevantorbits = importfile('relevant_orbits.txt', 1, 1021);
1422 %
1423 % See also TEXTSCAN.
1424
1425 % Auto-generated by MATLAB on 2018/07/23 15:13:23
1426
1427 % Initialize variables.
1428 delimiter = ' ';
1429 if nargin<2
1430     startRow = 1;
1431     endRow = inf;
1432 end
1433
1434 % Format for each line of text:
1435 % column1: datetimes ({yyyy-MM-dd}D)
1436 % column2: datetimes ({HH:mm:ss}D)
1437 % For more information, see the TEXTSCAN documentation.
1438 formatSpec = '{yyyy-MM-dd}D{HH:mm:ss}D%[^\\n\\r]';

```

```

1439
1440 % Open the text file.
1441 fileID = fopen(filename,'r');
1442
1443 % Read columns of data according to the format.
1444 % This call is based on the structure of the file used to generate this
1445 % code. If an error occurs for a different file, try regenerating the code
1446 % from the Import Tool.
1447 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', 'HeaderLines', ...
    startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1448 for block=2:length(startRow)
1449     frewind(fileID);
1450     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', ...
        'HeaderLines', startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1451     for col=1:length(dataArray)
1452         dataArray{col} = [dataArray{col};dataArrayBlock{col}];
1453     end
1454 end
1455
1456 % Close the text file.
1457 fclose(fileID);
1458
1459 % Post processing for unimportable data.
1460 % No unimportable data rules were applied during the import, so no post
1461 % processing code is included. To generate code which works for
1462 % unimportable data, select unimportable cells in a file and regenerate the
1463 % script.
1464
1465 % Create output variable
1466 relevantorbits = table(dataArray{1:end-1}, 'VariableNames', {'year_day','hour'});
1467
1468 % For code requiring serial dates (datenum) instead of datetime, uncomment
1469 % the following line(s) below to return the imported dates as datenum(s).
1470
1471 % relevantorbits.year_day=datenum(relevantorbits.year_day);
1472 % relevantorbits.hour=datenum(relevantorbits.hour);
1473 end
1474 function GHRCURLs = import_GHRCURLs(filename)
1475 %IMPORTFILE Import numeric data from a text file as a matrix.
1476 %   GHRCURLs = IMPORTFILE(FILENAME) Reads data from text file FILENAME for
1477 %   the default selection.
1478 %
1479 %   GHRCURLs = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
1480 %   STARTROW through ENDROW of text file FILENAME.
1481 %
1482 % Example:
1483 %   GHRCURLs = importfile('GHRCURLs.txt', 1, 433);
1484 %
1485 %   See also TEXTSCAN.
1486
1487 % Auto-generated by MATLAB on 2018/07/23 15:39:58
1488
1489 % Initialize variables.
1490 delimiter = {' '};
1491 if nargin<2
1492     startRow = 1;
1493     endRow = inf;
1494 end
1495
1496 % Format for each line of text:
1497 %   column1: text (%s)
1498 % For more information, see the TEXTSCAN documentation.
1499 formatSpec = '%s[\n\r]';
1500
1501 % Open the text file.
1502 fileID = fopen(filename,'r');
1503
1504 % Read columns of data according to the format.
1505 % This call is based on the structure of the file used to generate this
1506 % code. If an error occurs for a different file, try regenerating the code
1507 % from the Import Tool.

```

```

1508 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'TextType', 'string', 'HeaderLines', startRow(1)-1, 'ReturnOnError', ...
    false, 'EndOfLine', '\r\n');
1509 for block=2:length(startRow)
1510     frewind(fileID);
1511     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'TextType', 'string', 'HeaderLines', ...
        startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1512     dataArray{1} = [dataArray{1};dataArrayBlock{1}];
1513 end
1514
1515 % Close the text file.
1516 fclose(fileID);
1517
1518 %Post processing for unimportable data.
1519 % No unimportable data rules were applied during the import, so no post
1520 % processing code is included. To generate code which works for
1521 % unimportable data, select unimportable cells in a file and regenerate the
1522 % script.
1523
1524 % Create output variable
1525 GHRCURLs = table(dataArray{1:end-1}, 'VariableNames', ...
    {'httpsghrcnsstcnasagovpublisissdatasciencengchdf20170401ISS_LIS-'});
1526
1527 end
1528
1529 function webfilenames2urls(write_dir,read_filename)
1530
1531 website='https://ghrc.nsstc.nasa.gov/pub/lis/iss/data/science/nqc/hdf/';
1532
1533 files= import_web_filenames(read_filename);
1534 %urls generation
1535 disp('Creating url vector...');
1536
1537 write_filename=fullfile(write_dir,'interesting_URLs.txt');
1538 fileID=fopen(write_filename,'w');
1539
1540 disp(['Printing @ ' write_dir]);
1541 for i=1:size(files,1)
1542
1543     name=char(files{i,1});
1544     year=name([17:20]);
1545     month=name([21 22]);
1546     day=name([23 24]);
1547     url(i,:)=[website year '/' month day '/' name];
1548     fprintf(fileID,[url(i,:) '\r\n']);
1549
1550 end
1551 fclose(fileID);
1552 end
1553 function files = import_web_filenames(filename)
1554 %IMPORTFILE Import numeric data from a text file as a matrix.
1555 % FILES = IMPORTFILE(FILENAME) Reads data from text file FILENAME for the
1556 % default selection.
1557 %
1558 % FILES = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
1559 % STARTROW through ENDROW of text file FILENAME.
1560 %
1561 % Example:
1562 % files = importfile('filenames.website.txt', 2, 30);
1563 %
1564 % See also TEXTSCAN.
1565
1566 % Auto-generated by MATLAB on 2018/07/24 10:17:13
1567
1568 % Initialize variables.
1569 delimiter = {'\t','?','[',']',' '};
1570 if nargin<2
1571     startRow = 2;
1572     endRow = inf;
1573 end
1574
1575 %Format for each line of text:

```

```

1576 % column1: text (%s)
1577 % column2: datetimes ({MMM}D)
1578 % column3: datetimes ({dd}D)
1579 % For more information, see the TEXTSCAN documentation.
1580 formatSpec = '%s{MMM}D%{dd}D%s*s*s*s*s*s*s*s%[\n\r]';
1581
1582 % Open the text file.
1583 fileID = fopen(filename,'r');
1584
1585 % Read columns of data according to the format.
1586 % This call is based on the structure of the file used to generate this
1587 % code. If an error occurs for a different file, try regenerating the code
1588 % from the Import Tool.
1589 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', 'HeaderLines', ...
    startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1590 for block=2:length(startRow)
1591     frewind(fileID);
1592     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', ...
        'HeaderLines', startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1593     for col=1:length(dataArray)
1594         dataArray{col} = [dataArray{col};dataArrayBlock{col}];
1595     end
1596 end
1597
1598 % Close the text file.
1599 fclose(fileID);
1600
1601 % Post processing for unimportable data.
1602 % No unimportable data rules were applied during the import, so no post
1603 % processing code is included. To generate code which works for
1604 % unimportable data, select unimportable cells in a file and regenerate the
1605 % script.
1606
1607 % Create output variable
1608 files = table(dataArray{1:end-1}, 'VariableNames', {'name','month','month_day'});
1609
1610 % For code requiring serial dates (datenum) instead of datetime, uncomment
1611 % the following line(s) below to return the imported dates as datenum(s).
1612
1613 % files.month=datenum(files.month);
1614 % files.month_day=datenum(files.month_day);
1615
1616 end

```


A.6.2 LIS_vs_LMA_comparator.m

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %// LIS HDF4 data processor //
3  %// MATLAB converter //
4  %// Universitat Politecnica de Catalunya //
5  %// icar.fontcuberta@gmail.com //
6  %// //
7  %// Requires: MATLAB R2017b at least //
8  %// //
9  %// //
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12
13
14 %% Program Presets
15 clear all;
16 close all;
17 clc;
18
19 % To use the program:
20 %1. select LMA.filename of the day you want to study the sources
21 %2. Set the "examined.day" variable as the same day.
22 %3. Set timestep
23 %4. Set plotting options: histograms/plotting (sources in
24 %time)/sources+events in time
25 %5.RUN
26
27 disp("##### START OF EXECUTION #####");
28 %{
29 sources_data_file = 'C:\Users\icar\Google ...
    Drive\LIGHTNING\10minPeriodData\1710181700\171018.1700.txt';
30 events_data_file = ...
    'C:\Users\icar\Desktop\LIS-HDF_files\Ebre\Ebre.txt\ISS-LIS.20171018.1701.1701.events.txt';
31 read_workspace_dir = ...
    'C:\Users\icar\Desktop\LIS-HDF_files\Ebre\Ebre.other.info\workspace.40.7-0.7-171018.180713.mat';
32 corrected_file='C:\Users\icar\Desktop\txt4database\corrected_file.txt';
33 %}
34 LIS.total_filename = '/Users/Icar/Google Drive/TFG/LMA.LIS.DATA/ISS-LIS.txt';
35 LMA.filename = '/Users/Icar/Google Drive/TFG/LMA.LIS.DATA/LMA_FOV_20171018_pass1.txt';
36 examined.day = datetime(2017,10,18); %used to get the LMA date correctly
37
38
39 disp("###Remember to set the examined.day to the one from where the LMA is coming");
40
41 timestep = 2e-3; %sec
42
43 %{
44 % addpath('C:\Users\icar\Desktop\LIS-HDF_files\Ebre\Ebre.txt\');
45 % addpath('C:\Users\icar\Desktop\Data.from.scilab\');
46
47 %starttimes/endtimes
48 %ebre
49 %[18-Oct-2017 10:32:46 18-Oct-2017 10:34:31]
50 %[18-Oct-2017 17:01:25 17:01:28]
51
52 %barranca
53 %[18-Jul-03 6:40:14 6:41:18]
54 %[18-Jun-08 3:56:04 3:56:38]
55
56 %lisdata = import_lis(events_data_file);
57 correcting=0; %1 if the database is used to correct postion
58 toinput=0; %1 if the user wants to input the start time and endtime of fov. If 0 the ...
    program will take the min(listime) and max(endtime). THIS IS BAD
59 storedinput=1; %the same as above but previously saved in fovtime:
60 %}
61
62 %Plotting Options
63 histograms=0;
64 plotting=1;
65 sources_and_events = 0; %plot sources and events over time. COMPUTATION LOADING

```

```

66 comparing_length_section=0;
67
68 %Other Options
69 save_workspace=0; %if you want to write workspace. SAFEMODE=0
70 saving_csvfile=0;
71
72
73
74 disp(' ');
75 disp('-----');
76 %% Get LMA data
77 ouput_data = get_LMA_data(LMA_filename);
78
79 disp("Reading LMA data...");
80
81
82 sources_data = get_LMA_data(LMA_filename);
83
84 lma.flash = sources_data(:,1);
85 lma.sources_time = examined_day + seconds(sources_data(:,2));
86 lma.slats = sources_data(:,4);
87 lma.slons = sources_data(:,5);
88 lma.salts = sources_data(:,6);
89 lma.pwr = sources_data(:,7); %power
90 lma.LIS_FOV = sources_data(:,8);
91 lma.noise_fID = sources_data(:,9);
92
93 %eliminate those sources that were not under LIS FOV:
94 indexes = find(lma.LIS_FOV == 0);
95
96         firstsize = length(lma.sources_time);
97         secondsize = length(indexes);
98         disp(['Eliminating ' 'sources outside the LIS FOV time...']);
99         disp([num2str(firstsize-secondsize) ' elements within the FOV']);
100
101 lma.flash(indexes) = [];
102 lma.sources_time(indexes) = [];
103 lma.slats(indexes) = [];
104 lma.slons(indexes) = [];
105 lma.salts(indexes) = [];
106 lma.pwr(indexes) = [];
107 lma.LIS_FOV(indexes) = [];
108 lma.noise_fID(indexes) = [];
109
110 %The starttime of that day will be the minum time of the sources that were
111 %inside the FOV, that day.
112
113 starttime = min(lma.sources_time);
114
115 endtime = max(lma.sources_time);
116
117
118
119
120 %% Get LIS data
121
122 lisdata=import_LIS_total_data(LIS_total_filename);
123
124         disp('Reading LIS data...');
125
126         %LIS data
127         day_string = num2str(lisdata(:,1));
128         date_day = datetime(day_string, 'InputFormat', 'yyyyMMdd');
129         [lis.events_time, I] = sort(date_day+seconds(lisdata(:,2)));
130
131         lis.elats = lisdata(I,3);
132         lis.elons = lisdata(I,4);
133         lis.erad = lisdata(I,5);
134         lis.group = lisdata(I,6);
135         lis.glat = lisdata(I,7);
136         lis.glon = lisdata(I,8);
137         lis.flash = lisdata(I,9);
138         lis.flat = lisdata(I,10);

```

```

139     lis.flon = lisdata(I,11);
140     lis.area = lisdata(I,12);
141     lis.alat = lisdata(I,13);
142     lis.alon = lisdata(I,14);
143     lis.aobstime = lisdata(I,15);
144     lis.xpixel = lisdata(I,16);
145     lis.ypixel = lisdata(I,17);
146     lis.bgrad = lisdata(I,18);
147     lis.dis2lma = lisdata(I,19);
148
149
150
151     lis.starttime_address=find(lis.events.time>starttime,1,'first'); %first source ...
152                               time where LIS had it under its FOV
153     lis.endtime_address=find(lis.events.time<endtime,1,'last');
154
155     erase_indexes = [find(lis.events.time<starttime); find(lis.events.time>endtime)];
156
157
158     lis.events.time(erase_indexes)=[];
159     %lma.seconds_flash(erase_indexes)=[];
160     lis.elats(erase_indexes)=[];
161     lis.elons(erase_indexes)=[];
162     lis.erad(erase_indexes)=[];
163     lis.group(erase_indexes)=[];
164     lis.glat(erase_indexes)=[];
165     %lma.cstp(erase_indexes)=[];
166     lis.glon(erase_indexes)=[];
167     lis.flash(erase_indexes)=[];
168     lis.flat(erase_indexes)=[];
169     lis.flon(erase_indexes)=[];
170     lis.area(erase_indexes)=[];
171     lis.alat(erase_indexes)=[];
172     lis.alon(erase_indexes)=[];
173     lis.aobstime(erase_indexes)=[];
174     lis.xpixel(erase_indexes)=[];
175     lis.ypixel(erase_indexes)=[];
176     lis.bgrad(erase_indexes)=[];
177     lis.dis2lma(erase_indexes)=[];
178
179     disp(['Start time: ']);
180     disp(starttime);
181     disp(['End time: ']);
182     disp(endtime);
183
184     %% Make chunkinfo structures
185
186     %{
187     %load(read_workspace_dir,'fovinfo');
188     %[min_starttime_on_area, max_endtime_on_area]=lisboundaries(fovinfo,lma);
189     %}
190
191     %{
192     %Let's set a function that leaves only the data that should be seen from
193     %both sensors simultaneously
194
195     %Analyze all the LMA detections and eliminate those that are outside the
196     %field of view. Create new lma struct and do the following:
197     %}
198
199
200     timeperiod = seconds(endtime-starttime);
201
202     %ftimestep = timeperiod/(lma.total_nflashes.*10); %an adimensional number.
203                                     %The flashes may not fit in the same chunk but
204                                     %but w/ this criteria we got a avalue
205                                     %that changes with no of flashes
206
207
208     %etimestep = timeperiod/(lis.nevents*2); %check if inside a eventstime chunk therees ...
209                               a LMA detection!
210     etimestep = timestep;

```

```

210
211 disp('Dividing time in chunks...');
212 % Preccence
213 [chunksinfo, chunksdata]=check_detections(etimestep, starttime, endtime, lis, lma);
214
215 chunksinfo=e_vs_s_presence(chunksdata, chunksinfo);
216
217 chunksinfo.annotation='.times gives the minutes coordinates, from the hour we are ...
    regarding, of all chunks that divide the detection period. Other .chunk_ guive ...
    the ADDRESS of the chunks.';
218
219
220 %physical properties of chunks where events and/or sources where detected
221 [secproperties, scproperties]=chunk_physical_properties(chunksinfo, lis, lma);
222 disp('Info about the time chunks saved');
223
224
225 chunksinfo
226
227 %% Statistics Mean, Medians...
228 %power
229 secmaxpwrarray=array_max_power(secproperties);
230 scmaxpwrarray=array_max_power(scproperties);
231
232 semeanmaxpower=mean(secmaxpwrarray);
233 semedianmaxpower=median(secmaxpwrarray);
234
235 smeanmaxpower=mean(scmaxpwrarray);
236 smedianmaxpower=median(scmaxpwrarray);
237
238 secpwrcentroid=array_power_centroid(secproperties);
239 scpwrcentroid=array_power_centroid(scproperties);
240
241 semeanpowercentroid=mean(secpwrcentroid);
242 semedianpowercentroid=median(secpwrcentroid);
243
244 smeanpowercentroid=mean(scpwrcentroid);
245 smedianpowercentroid=median(scpwrcentroid);
246
247
248 %height
249 %secmedian, secmen, ... are vectors taht contain the mean height of all the
250 %sources inside each chunk, one value per chunk.
251 secmedian=array_h_median(secproperties);
252 scmedian=array_h_median(scproperties);
253
254 secmean=array_h_mean(secproperties);
255 scmean=array_h_mean(scproperties);
256
257 sheightmedianmean=mean(scmedian);
258 seheightmedianmean=mean(secmedian);
259
260 seheightmeanmean=mean(secmean);
261 sheightmeanmean=mean(scmean);
262
263 seheightmeanmedian=median(secmean);
264 sheightmeanmedian=median(scmean);
265
266 %number of sources
267 %nsources/nesources is the vector that contain information about the number
268 %of sources in each chunk that contains only/with events sources
269
270 [nesources, nsources]=nsourcesxchunks(chunksinfo);
271 senumsourcesmean=mean(nesources);
272 senumsourcesmedian=median(nesources);
273
274 snumsourcesmean=mean(nsources);
275 snumsourcesmedian=median(nsources);
276
277 %Sum of powers in each chunk
278 %sesumpower/ssumpower are the vectors that contain the simple sum of all
279 %the powers in each chunk with_events/only sources
280 [sesumpower, sumpower]=sumpowersinchunks(chunksinfo, lma);

```

```
281
282
283 sesumpowermean=mean(sesumpower);
284 ssumpowermean=mean(ssumpower);
285
286 sesumpowermedian=median(sesumpower);
287 ssumpowermedian=median(ssumpower);
288
289 if save_workspace==1
290 disp('saving workspace...');
291 save([erase(sources_data_file, '.txt') '.mat']);
292 disp('done');
293
294 end
295
296
297
298
299
300 %% Plotting representative values
301 n=1;
302 if plotting==1
303 disp('Plotting over time started...');
304 close all;
305 %x axis limits:
306 [h,m,s]=hms([starttime-seconds(etimestep) endtime+seconds(etimestep)]);
307 lims=minutes(minutes(m)+seconds(s));
308
309 % Power may be related to the detection from LIS
310
311 %maximum power
312 figure(n);
313 n=n+1;
314
315 scatter(minutes(chunksinfo.times(chunksinfo.chunks_w.both)),secmaxpwrarray,'*b');
316 hold on;
317 scatter(minutes(chunksinfo.times(chunksinfo.chunks_only-sources)),scmaxpwrarray,'*k');
318 grid on;
319 title('Maximum power from the sources registered in each time chunk');
320 xlabel('Time from the current hour [min.]');
321 ylabel('Max power [dW]');
322 legend('Chunk w/ events+sources','Chunk w/ only sources');
323 %xlim(lims);
324 colormap(jet);
325
326 figure(n);
327 n=n+1;
328 scatter(minutes(chunksinfo.times(chunksinfo.chunks_w.both)),secpwrcentroid,'*b');
329 hold on;
330 scatter(minutes(chunksinfo.times(chunksinfo.chunks_only-sources)),scpwrcentroid,'*k');
331 grid on;
332
333 title('Height of power centroid in the chunk');
334 xlabel('Time from the current hour [min.]');
335 ylabel('Height [m]');
336 legend('Chunk w/ events+sources','Chunk w/ sources');
337 %xlim(lims);
338 colormap(jet);
339
340 %-----
341 figure(n);
342 n=n+1;
343 [or_times, indexv]=sort(lis.events.time,1);
344 or_pixels(:,1)=lis.xpixel(indexv);
345 or_pixels(:,2)=lis.ypixel(indexv);
346
347 or_times=datenum(or_times);
348 scatter(or_pixels(:,1),or_pixels(:,2),1,or_times,'.');
349 axis equal;
350 grid on;
351 axis([0 128 0 128]);
352 title('Pixels excited on the CCD. Colored by time');
353 colormap(jet);
```

```

354
355 %-----
356
357 disp('Continuing plotting...');
358 % Comparing Altitudes in detection
359 %Median of the height
360 figure(n);
361 n=n+1;
362
363
364
365 scatter(minutes(chunksinfo.times(chunksinfo.chunks.w.both)),secmedian,'*b');
366 hold on;
367 scatter(minutes(chunksinfo.times(chunksinfo.chunks.only.sources)),scmedian,'.k');
368 grid on;
369 title('Median of sources height in each time chunk');
370 xlabel('Time from the current hour [min.]');
371 ylabel('Height [m]');
372 legend('Chunk w/ events+sources','Chunk w/ sources');
373 xlim(lims);
374 colormap(jet);
375
376
377
378 figure(n);
379 n=n+1;
380
381 scatter(minutes(chunksinfo.times(chunksinfo.chunks.w.both)),secmean,'*b');
382 hold on;
383 scatter(minutes(chunksinfo.times(chunksinfo.chunks.only.sources)),scmean,'.k');
384 grid on;
385 title('Mean of sources height in each time chunk');
386 xlabel('Time from the current hour [min.]');
387 ylabel('Height [m]');
388 legend('Chunk w/ events+sources','Chunk w/ sources');
389 %xlim(lims);
390
391 colormap(jet);
392
393 end
394
395 if sources.andevents ==1
396 disp("Plotting events & sources over time...");
397 % plot events with sources
398 figure(n);
399 n=n+1;
400 k=0;
401 grid on;
402 hold on;
403
404 [C,ia,ic]=unique(lis.flash);
405 colorvalues(1)=5;
406 for j=2:length(C)
407
408     colorvalues(1,j)=colorvalues(j-1)+30;
409
410 end
411
412 colorvector=colorvalues(ic)';
413
414
415 for i=1:length(chunksinfo.events)
416
417     local_addresses=chunksinfo.events{i};
418     if ~isempty(local_addresses)
419         scatter(lis.events.time(local_addresses),linspace(500,500,length(local_addresses)),lis.eras
420             k=k+length(local_addresses);
421
422     end
423
424
425 end
426

```

```

427     [C,ia,ic]=unique(lma.flash);
428     colorvalues(1)=5;
429     for j=2:length(C)
430
431         colorvalues(1,j)=colorvalues(j-1)+30;
432
433     end
434
435     colorvector=colorvalues(ic)';
436
437     for i=1:length(chunksinfo.sources)
438
439         local_addresses=chunksinfo.sources{i};
440         if ~isempty(local_addresses)
441             try
442                 scatter(lma.sources_time(local_addresses),lma.salts(local_addresses),abs(lma.pwr(local_addresses)),colorvector);
443             catch
444                 warning("Unable to plot Events & Sources");
445             end
446         end
447
448
449     end
450     ylabel('Height [m]');
451     title('Events and sources printed over time');
452     legend('x Events. Size==radiance. Color==flash.','+ Sources. Size==power. ... Color==flash.');
```

```

453
454     %etimestep=seconds(etimestep);
455     %timev=(starttime-etimestep/2):etimestep:(endtime+etimestep/2);
456     %xticks(timev);
457
458     xlim([starttime-seconds(etimestep) endtime+seconds(etimestep)]);
459
460 end
461 cif savingcsvfile==1
462 % Write CSV file to open with excel
463 disp('Writing csv file...');
464 M=[semeanmaxpower semedianmaxpower smeanmaxpower smedianmaxpower semeanpowercentroid ...
    semedianpowercentroid smeanpowercentroid smedianpowercentroid sheightmedianmean ...
    seheightmedianmean seheightmeanmean sheightmeanmean seheightmeanmedian ...
    sheightmeanmedian senumsourcesmean senumsourcesmedian snumsourcesmean ...
    snumsourcesmedian sesumpowermean ssumpowermean sesumpowermedian ssumpowermedian];
465 csvwrite([erase(sources_data_file, '.txt') '.csv'],M);
466 disp('Done.');
```

```

467
468 end
469
470 %% Plot histograms
471 if histograms == 1
472     disp('Plotting histograms...');
```

```

473
474
475
476     %maximum power
477     figure(n);
478     n=n+1;
479
480     histogram(secmaxpwrarray,30);
481     hold on;
482     histogram(scmaxpwrarray,30);
483     title('Bins max power histogram');
484     xlabel('Max power in the bin [dBW]');
485     ylabel('Counts');
486     legend('Sources + events bins','only sources bins');
```

```

487
488
489     %Heights
490
491     figure(n);
492     n=n+1;
493     histogram(secmedian,30);
494     hold on;
```

```
495 histogram(scmmedian,30);
496
497 title('Bins median height histogram');
498 ylabel('Counts');
499 xlabel('Height [m]');
500 legend('sources+events bins','only sources bins');
501
502
503 figure(n);
504 n=n+1;
505
506 histogram(secmean,30);
507 hold on;
508
509 histogram(scmean,30);
510 title('Bins height mean histogram');
511 ylabel('Counts');
512 xlabel('Height [m]');
513 legend('sources+events bins','only sources bins');
514
515
516
517
518 %sources densities per bin
519 figure(n);
520 n=n+1;
521 histogram(nesources);
522 hold on;
523 histogram(nsources);
524 title('Density of sources per bin');
525 ylabel('Counts');
526 xlabel('Number of sources per bin');
527 legend('Bins with sources+events','Bins with only sources');
528
529
530 figure(n);
531 n=n+1;
532 histogram(secpwrcentroid,30);
533 hold on;
534 histogram(scpwrcentroid,30);
535 title('Bins power centroid histogram');
536 ylabel('Counts');
537 xlabel('Height [m]');
538 legend('sources+events bins','only sources bins');
539
540
541 figure(n);
542 n=n+1;
543 histogram(sesumpower)
544 hold on;
545 histogram(ssumpower)
546 title('Distributions for the sum of power in each bin');
547 xlabel('Power [dBW]');
548 ylabel('Counts');
549 legend('sources+events bins','only sources bins');
550
551 %Power population distribution
552 figure(n);
553 n=n+1;
554
555 formatOut='yyyy-mm-dd HH:MM';
556 histogram(lma.pwr)
557 title(['Power Histogram for ' datestr(starttime,formatOut)]);
558 xlabel('Power [dBW]');
559 ylabel('Counts');
560
561 figure(n);
562 n=n+1;
563
564 histogram(lis.erad)
565 title(['Radiance histogram for ' datestr(starttime,formatOut)]);
566 ylabel('Counts');
567 xlabel('Radiance [J/(m^2 * sr *m)]');
```



```

568
569
570
571
572 figure(n);
573 n=n+1;
574 histogram(lma.salts);
575 title(['Heights histogram for ' datestr(starttime,formatOut)]);
576 ylabel('Counts');
577 xlabel('Height [m]');
578 disp('plotting ended');
579
580
581 end
582
583 %% Power of events that had a source at 2500m associated
584
585 DISCHARGE_ALT = 2500; %m
586 tolerance = 50; %m. Tolerance for the search
587
588 [rad.dischargealt] = ...
    typical_radiance_at_height(DISCHARGE_ALT,tolerance,lma,lis,chunksinfo);
589
590 disp(['Radiances at discharge height: ' num2str(DISCHARGE_ALT) ...
    'm with tolerance: ' num2str(tolerance) 'm:']);
591
592 disp(['Mean: ' num2str(mean(rad.dischargealt.averages)) ' [\mu J/sr/m^2/\mu m]']);
593 disp(['Median: ' num2str(median(rad.dischargealt.medians)) ' [\mu J/sr/m^2/\mu m]']);
594
595 figure(n);
596 n= n+1;
597
598 histogram(rad.dischargealt.medians,30);
599 hold on;
600 histogram(rad.dischargealt.averages,30);
601 title(['Typical values for the radiance of events that had sources associated at ' ...
    num2str(DISCHARGE_ALT) ...
    'm with tolerance: ' num2str(tolerance) 'm:']);
602
603
604 xlabel('Radiance [\mu J/sr/m^2/\mu m] ');
605 ylabel('Time bin counts');
606
607 text = {strcat(['Altitude: ' num2str(rad.dischargealt.alt) 'm ']); ...
    strcat(['Tolerance: ' num2str(rad.dischargealt.tol) 'm ']);...
    strcat(['Total n of bins: ' num2str(rad.dischargealt.nbins)]);}
608
609
610
611 dim = [.2 .55 .3 .3];
612 annotation('textbox',dim,'String',text,'FitBoxToText','on');
613
614 legend('Radiance median in the bin','Radiance average in the bin');
615
616
617 %% Comparing lightning length
618 if comparing_length_section == 1
619     disp('comparing flash lengthes...');
620 % GET THE LENGTH OF THE FLASHES. WE HAVE TO ASSIGN TIMINGS TO FLASHES AND
621 % THEN ASSIGN FLASHES OF LMA TO FLASHES OF LIS, SO WE COMPARE THE SAME
622 % FLASH. The LIS and LAST
623
624 %get the LIS flashes
625 %start times (and positions)
626 [flashnst,ilistst]=unique(lis.flash,'first'); %"flash number starts / index list ...
    starts]
627 %end times (and positions)
628 [flashnend,ilistend]=unique(lis.flash,'last'); %"flash number ends / index list ...
    ends"
629
630 lisflashprops.nflash=flashnst;
631 lisflashprops.ilistst=ilistst;
632 lisflashprops.ilistend=ilistend;
633 lisflashprops.times(:,1)=lis.events.time(ilistst);
634 lisflashprops.times(:,2)=lis.events.time(ilistend);
635
636

```

```

637 %get the LMA flashes
638 %start times (and positions)
639 [flashnst,ilistst]=unique(lma.flash,'first'); %"flash number starts / index list ...
        starts]
640 %end times (and positions)
641 [flashnend,ilistend]=unique(lma.flash,'last'); %"flash number ends / index list ...
        ends"
642
643 lmaflashprops.nflash=flashnst;
644 lmaflashprops.ilistst=ilistst;
645 lmaflashprops.ilistend=ilistend;
646 lmaflashprops.times(:,1)=lma.sources_time(ilistst);
647 lmaflashprops.times(:,2)=lma.sources_time(ilistend);
648
649
650 %there might be flashes that lis did not detect (usually the case).
651 %Associate the flashes:
652
653 flashesrelated=zeros(size(lisflashprops.nflash,1),4);
654 flashesrelated(:,1)=lisflashprops.nflash;
655
656
657 %now we have to associate the LIS flashes to LMA flashes
658
659 %4 options: 1    start time of LIS inside LMA
660              %2    end time of LIS inside LMA (the first two cases already
661              %comprehend the case of a LIS flash inside a LMA flash
662              %3    LMA flash inside LIS flash
663 %the starttime of lis flash has to be higher or equal
664 for i=1:size(lisflashprops.times,1)
665
666     lmaindex=intersect(find(lmaflashprops.times(:,1)≤lisflashprops.times(i,1)), ...
        find(lmaflashprops.times(:,2)≥lisflashprops.times(i,1))); %the start of ...
        LIS is comprehended in the LMA flash
667
668     if isempty(lmaindex) %in case that it is not the start, but the end of a LIS ...
        flash that is comprehended inside a LMA flash
669
670         lmaindex=intersect(find(lmaflashprops.times(:,1)≤lisflashprops.times(i,2)), ...
        find(lmaflashprops.times(:,2)≥lisflashprops.times(i,2)));
671
672         if isempty(lmaindex) %repeat the process for in case that the LMA flash ...
        is comprehended inside LIS flash
673
674
675             lmaindex=intersect(find(lmaflashprops.times(:,1)≥lisflashprops.times(i,1)), ...
        find(lmaflashprops.times(:,2)≤lisflashprops.times(i,2))); %the ...
        start of LIS is comprehended in the LMA flash
676
677
678         end
679
680     end
681
682     if isempty(lmaindex)
683
684         disp('error on comparing flashes times. lma index not assigned');
685         disp('It also can be that there are non-simultaneous flashes. (This can ...
        be seen in the data)');
686
687     else
688
689         flashesrelated(i,2)=lmaflashprops.nflash(lmaindex);
690
691     end
692
693 end
694
695 end
696
697
698 for i=1:size(flashesrelated,1)
699

```

```

700         if flashesrelated(i,2) ~=0
701             locallisflash=flashesrelated(i,1);
702             locallmaflash=flashesrelated(i,2);
703
704             indexlis=find(lisflashprops.nflash==locallisflash);
705
706             timeslis=(lisflashprops.times(indexlis,:));
707
708             durationlis=second(timeslis(2))-second(timeslis(1));
709
710             indexlma=find(lmaflashprops.nflash==locallmaflash);
711
712             timeslma=(lmaflashprops.times(indexlma,:));
713
714             durationlma=second(timeslma(2))-second(timeslma(1));
715
716
717             flashesrelated(i,3)=durationlis;
718             flashesrelated(i,4)=durationlma;
719
720
721         end
722     end
723
724     diffdurations=flashesrelated(:,4)-flashesrelated(:,3);
725
726
727     figure(n);
728     n=n+1;
729     histogram(flashesrelated(:,3),30);
730     hold on;
731     histogram(flashesrelated(:,4),30);
732     legend('LIS duration count','LMA duration count');
733     ylabel('Counts');
734     xlabel('Duration [sec.]');
735
736
737
738
739
740
741
742
743     disp('END at line 660');
744     disp(' '); disp(' '); disp(' ');
745     %flashesrelated content: /LIS#/ LMA#/ LIS durat./ LMA durat./
746
747
748
749
750 end
751
752
753 %% END
754 disp("#####END OF EXECUTION #####");
755 disp(" ");
756 %% Functions
757
758
759 function [chunksinfo,chunksdata] = check_detections(timestep,starttime,endtime,lis,lma)
760 timestep=seconds(timestep);
761
762
763 timev=(starttime-timestep/2):timestep:(endtime+timestep/2);
764 %the way the vector is constructed will assure that the first and last
765 %event/source is within the time perdioid. This vector has the values of the
766 %surrounding positions of timechunkns. It has thee end and final time in
767 %it.
768
769 central_chunk_times=starttime:timestep:endtime; %will have the same size as the ...
       chunk vector
770
771 %now we should assign events and sources to each chunk. We will assign, to

```

```

772 %each chunk, what events and sources index there are. In the first row we
773 %will put the events and in the second one the sources
774
775 %{
776 chunksdata{2,length(timev)-1}=[];
777
778     for i = 1:lis.nevents %check in what chunks events were detected
779
780         subind=find(lis.events.time(i)>timev,1,'last');
781
782         if ~isempty(subind)
783             chunksdata{1,subind} = [chunksdata{1,subind}, i];
784         end
785     end
786
787     for i = 1:lma.nsources %check in what chunks sources were detected
788
789         subind=find(lma.sources.time(i)>timev,1,'last'); %notice here the vector is ...
790         %the "timev". There is no ≥ because in case that the source.time would be ...
791         %equal to timev(end) it would assign to the last chunk, indexed through ...
792         %the left time limit of the chunk
793
794         if ~isempty(subind) %if it does find something
795             chunksdata{2,subind} = [chunksdata{2,subind}, i];
796         end
797     end
798
799     %}
800
801     %for each time chunk check if event and/or source was recorded
802
803     %lets store only the minutes and seconds of the chunk since the time is
804     %obvious
805     [h,m,s]=hms(central_chunk.times);
806     times=minutes(m)+seconds(s);
807
808
809
810
811 lis_addresses=knnsearch(seconds(central_chunk.times-datetime(1993,1,1))',seconds(lis.events.time-datetime(1993,1,1))');
812 lma_addresses=knnsearch(seconds(central_chunk.times-datetime(1993,1,1))',seconds(lma.sources.time-datetime(1993,1,1))');
813 %what we do is to set those datetime vectors as duration vectors (with the
814 %same time reference) to afterwards pass them to numeric seconds with
815 %seconds() function
816
817 chunksdata{2,length(central_chunk.times)}=[];
818
819 for i=1:length(lis_addresses)
820     chunksdata{1,lis_addresses(i)}=[chunksdata{1,lis_addresses(i)} i];
821 end
822
823 for i=1:length(lma_addresses)
824     chunksdata{2,lma_addresses(i)}=[chunksdata{2,lma_addresses(i)} i];
825 end
826
827 chunksinfo.times=times;
828 chunksinfo.events=chunksdata(1,:);
829 chunksinfo.sources=chunksdata(2,:);
830 chunksinfo.timestep=timestep;
831 chunksinfo.timeperiod=[starttime, endtime];
832
833
834
835
836 end
837
838 function chunksinfo=e_vs_s_presence(chunksdata,chunksinfo)
839
840 %this function only checks if each chunk is full of sources/events. It is,

```

```

841 %therefore, realted to its presence on the chunks.
842
843     chunks_w_events=find(~cellfun(@isempty,chunkdata(1,:)));%logical vectors. ...
844     l==has_events/sources. 0==empty
845     chunks_w_sources=find(~cellfun(@isempty,chunkdata(2,:)));
846
847     chunks_w_both=intersect(chunks_w_events,chunks_w_sources);
848
849     chunks_only_events=setdiff(chunks_w_events,chunks_w_sources);
850     chunks_only_sources=setdiff(chunks_w_sources,chunks_w_events);
851
852     empty_chunks=setdiff(1:length(chunkdata),[chunks_w_events chunks_w_sources]);
853
854     chunksinfo.chunks_w_events=chunks_w_events;
855     chunksinfo.chunks_w_sources=chunks_w_sources;
856     chunksinfo.chunks_w_both=chunks_w_both;
857     chunksinfo.chunks_only_events=chunks_only_events;
858     chunksinfo.chunks_only_sources=chunks_only_sources;
859     chunksinfo.empty_chunks=empty_chunks;
860
861 end
862
863 function [secproperties, scproperties]=chunk.physical_properties(chunksinfo,lis,lma)
864
865     secproperties=[];
866     scproperties=[];
867
868     s_and_e=chunksinfo.sources(chunksinfo.chunks_w_both); %sources address that appeared ...
869     alongside with events
870
871     %see what's the maximum height of each chunk and its median. (Half sources
872     %will be up that value and half will be under that value).
873
874     %lets store some physical properties of each chunk where sources and events
875     %were detected
876
877     %secproperties stands for "sources with events chunk properties"
878     %each "i" relates to a chunk of time!
879     for i=1:length(s_and_e)
880
881         local_addresses=s_and_e{1,i};
882         local_heights=lma.salts(local_addresses);
883         secproperties(i).mean.s_h=mean(local_heights);
884         secproperties(i).median.s_h=median(local_heights);
885
886         local_pwr=lma.pwr(local_addresses);
887         secproperties(i).max.pwr=max(local_pwr);
888
889         %influence of the power on the detection
890         local_pwr = 10.^(local_pwr/10); % temporal conversion to linear units to ...
891         weight the centroid
892         secproperties(i).pwr.centroid=sum(local_pwr.*local_heights)/sum(local_pwr);
893
894         %properties coming from LIS detection
895
896     end
897
898     os=chunksinfo.sources(chunksinfo.chunks_only_sources);
899
900     %store the same physical variables of chunks where only sources were
901     %detected (and so events should have been detected also)
902
903     %secproperties stands for "sources chunk properties"
904
905     for i=1:length(os)
906
907         local_addresses=os{1,i};
908         local_heights=lma.salts(local_addresses);
909         scproperties(i).mean.s_h=mean(local_heights);
910         scproperties(i).median.s_h=median(local_heights);

```

```
911         local_pwr=lma.pwr(local.addresses);
912         scproperties(i).max_pwr=max(local_pwr);
913
914         %influence of the power on the detection
915
916         local_pwr = 10.^(local_pwr/10); % temporal conversion to linear units to ...
917         weight the centroid
918         scproperties(i).pwr_centroid=sum(local_pwr.*local.heights)/sum(local_pwr);
919
920     end
921
922
923     disp('Physical Poperties of chunks were events and/or sources were detected done');
924 end
925
926
927 function property=array.power.centroid(struct)
928
929 property=zeros(1,length(struct));
930
931 maximum = 0;
932 indexmax = 0;
933     for i=1:length(struct)
934
935         property(i)=struct(i).pwr.centroid;
936
937         %         if property(i)>maximum
938         %             maximum = property(i);
939         %             indexmax = i;
940         %         end
941     end
942 % disp(maximum);
943 % disp(indexmax);
944
945
946
947 end
948
949 function property=array.max.power(struct)
950
951 property=zeros(1,length(struct));
952
953     for i=1:length(struct)
954
955         property(i)=struct(i).max.pwr; %the array property will have the maximum ...
956         power of each chunk
957     end
958
959 end
960
961 function property=array.h.mean(struct)
962
963 property=zeros(1,length(struct));
964
965     for i=1:length(struct)
966
967         property(i)=struct(i).mean.s_h;
968
969     end
970
971 end
972
973 function property=array.h.median(struct)
974
975 property=zeros(1,length(struct));
976
977     for i=1:length(struct)
978
979         property(i)=struct(i).median.s_h;
980
981     end
```

```
982
983 end
984
985 function write_txt_files(lma,write_dir)
986
987
988 addpath(write_dir);
989
990 %processing sources times to let them in a OK format for Paulino
991
992 fullfilename=fullfile(write_dir,'sources2database.txt');
993
994 disp('Opening .txt writed file...');
995 stringtimes=datestr(lma.sources_time,'yyyy-mm-dd;HH:MM:SS');
996 fileID=fopen(fullfilename,'w');
997
998 if fileID == -1
999
1000     disp('Could not open file. ');
1001 else
1002
1003     disp('File opened. Starting to print');
1004
1005     for i=1:length(lma.sources_time)
1006
1007         fprintf(fileID, '%s;%f;%f\r\n',stringtimes(i,:),lma.slats(i),lma.slons(i));
1008
1009
1010     end
1011
1012 fclose(fileID);
1013
1014 disp('Printing ended and file closed. ');
1015 end
1016 end
1017
1018 function [nesources,nsources]=nsourcesxchunks(chunksinfo)
1019
1020
1021 global_addresses=chunksinfo.sources(chunksinfo.chunks_w_both);
1022
1023 nesources=zeros(1,length(global_addresses));
1024
1025 for i=1:length(global_addresses)
1026
1027     local_addresses=global_addresses{i};
1028
1029     nesources(i)=length(local_addresses);
1030
1031 end
1032
1033 global_addresses=chunksinfo.sources(chunksinfo.chunks_only_sources);
1034
1035 nsources=zeros(1,length(global_addresses));
1036
1037 for i=1:length(global_addresses)
1038
1039     local_addresses=global_addresses{i};
1040
1041     nsources(i)=length(local_addresses);
1042
1043 end
1044
1045 end
1046
1047 function [sesumpower,ssumpower]=sumpowersinchunks(chunksinfo,lma)
1048
1049
1050
1051 global_addresses=chunksinfo.sources(chunksinfo.chunks_w_both);
1052
1053 sesumpower=zeros(1,length(global_addresses));
1054
```

```

1055     for i=1:length(global_addresses)
1056         local_addresses=global_addresses{i};
1057
1058         pwr_sumable = 10.^(lma.pwr(local_addresses)/10); %from dB to W
1059
1060         sesumpower(i)=10*log10((sum(pwr_sumable))); %from W to dB
1061
1062     end
1063     global_addresses=chunksinfo.sources(chunksinfo.chunks_only.sources);
1064
1065     ssumpower=zeros(1,length(global_addresses));
1066
1067     for i=1:length(global_addresses)
1068         local_addresses=global_addresses{i};
1069
1070         pwr_sumable = 10.^(lma.pwr(local_addresses)/10); %from dB to W
1071
1072         ssumpower(i)=10*log10((sum(pwr_sumable))); %from W to dB
1073
1074     end
1075 end
1076
1077 %Reading data
1078 function sources_data = import_sourcesdata(filename)
1079 %IMPORTFILE Import numeric data from a text file as a matrix.
1080 % SOURCES_DATA = IMPORTFILE(FILENAME) Reads data from text file FILENAME
1081 % for the default selection.
1082 %
1083 %
1084 % SOURCES_DATA = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from
1085 % rows STARTROW through ENDROW of text file FILENAME.
1086 %
1087 % Example:
1088 % sources_data = importfile('2017-10-06-10-30.txt', 2, 4856);
1089 %
1090 % See also TEXTSCAN.
1091
1092 % Auto-generated by MATLAB on 2018/07/27 12:20:39
1093
1094 % Initialize variables.
1095 delimiter = ' ';
1096 if nargin< 2
1097     startRow = 2;
1098     endRow = inf;
1099 end
1100
1101 % Format for each line of text:
1102 % column1: double (%f)
1103 % column2: double (%f)
1104 % column3: double (%f)
1105 % column4: double (%f)
1106 % column5: double (%f)
1107 % column6: double (%f)
1108 % column7: double (%f)
1109 % column8: double (%f)
1110 % For more information, see the TEXTSCAN documentation.
1111 formatSpec = '%f%f%f%f%f%f%f*s%[\n\r]';
1112
1113 % Open the text file.
1114 fileID = fopen(filename,'r');
1115
1116 % Read columns of data according to the format.
1117 % This call is based on the structure of the file used to generate this
1118 % code. If an error occurs for a different file, try regenerating the code
1119 % from the Import Tool.
1120 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', 'EmptyValue', NaN, ...
    'HeaderLines', startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1121 for block = 2:length(startRow)
1122     frewind(fileID);
1123     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', ...
        'EmptyValue', NaN, 'HeaderLines', startRow(block)-1, 'ReturnOnError', false, ...

```



```

    'EndOfLine', '\r\n');
124     for col = 1:length(dataArray)
125         dataArray{col} = [dataArray{col};dataArrayBlock{col}];
126     end
127 end
128
129 % Close the text file.
130 fclose(fileID);
131
132 % Post processing for unimportable data.
133 % No unimportable data rules were applied during the import, so no post
134 % processing code is included. To generate code which works for
135 % unimportable data, select unimportable cells in a file and regenerate the
136 % script.
137
138 % Create output variable
139 sources_data = [dataArray{1:end-1}];
140 end
141 function sources_header = import_sources_header(filename)
142 %IMPORTFILE Import numeric data from a text file as a matrix.
143 % SOURCES_DATA = IMPORTFILE(FILENAME) Reads data from text file FILENAME
144 % for the default selection.
145 %
146 % SOURCES_DATA = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from
147 % rows STARTROW through ENDROW of text file FILENAME.
148 %
149 % Example:
150 % sources_data = importfile('2017_10_06_10_30.txt', 1,1);
151 %
152 % See also TEXTSCAN.
153
154 % Auto-generated by MATLAB on 2018/07/27 10:45:10
155
156 % Initialize variables.
157 delimiter = ' ';
158 if nargin ≤ 2
159     startRow = 1;
160     endRow = 1;
161 end
162
163 % Format for each line of text:
164 % column1: double (%f)
165 % column2: double (%f)
166 % column3: double (%f)
167 % column4: double (%f)
168 % column5: double (%f)
169 % column6: double (%f)
170 % column7: double (%f)
171 % column8: double (%f)
172 % column9: double (%f)
173 % For more information, see the TEXTSCAN documentation.
174 formatSpec = '%f%f%f%f%f%f%f%f%f%\n\r';
175
176 % Open the text file.
177 fileID = fopen(filename,'r');
178
179 % Read columns of data according to the format.
180 % This call is based on the structure of the file used to generate this
181 % code. If an error occurs for a different file, try regenerating the code
182 % from the Import Tool.
183 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', 'HeaderLines', ...
    startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
184 for block = 2:length(startRow)
185     frewind(fileID);
186     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'MultipleDelimsAsOne', true, 'TextType', 'string', ...
        'HeaderLines', startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
187     for col = 1:length(dataArray)
188         dataArray{col} = [dataArray{col};dataArrayBlock{col}];
189     end
190 end
191

```



```

1260     end
1261 end
1262
1263 % Close the text file.
1264 fclose(fileID);
1265
1266 % Post processing for unimportable data.
1267 % No unimportable data rules were applied during the import, so no post
1268 % processing code is included. To generate code which works for
1269 % unimportable data, select unimportable cells in a file and regenerate the
1270 % script.
1271
1272 % Create output variable
1273 lisdata = [dataArray{1:end-1}];
1274 end
1275
1276
1277 function [min.starttime.on.area, max.endtime.on.area]=lisboundaries(fovinfo,lma)
1278
1279 %which file are we lookin in?
1280
1281 k=1; %i know "manually that in this case is the first interesting file
1282     coordinates=fovinfo.fov.coordinates;
1283
1284     fovlats=coordinates(1,:);
1285     fovlons=coordinates(2,:);
1286
1287     minlat=lma.minlat;
1288     minlon=lma.minlon;
1289     maxlat=lma.maxlat;
1290     maxlon=lma.maxlon;
1291
1292     disp('Checking which points are inside the interesting area...');
1293     inside_range_index=intersect(intersect(find(fovlats>minlat),find(fovlats<maxlat)),intersect(find(fo
1294
1295     %check at when the lis will start to see and leave the area. This is not
1296     %exact due to the fact that maybe the centroids are outside but the fov
1297     %cell has some part inside or viceversa.
1298
1299     interestingfovtimes=[fovinfo(k).fovstart(inside_range_index); ...
1300         fovinfo(k).fovend(inside_range_index)];
1301
1302     min_starttime_on_area=min(interestingfovtimes(1,:)); %minimum time where the lis ...
1303         is sensing the area
1304     max_endtime_on_area=max(interestingfovtimes(2,:)); %last time when LIS saw info ...
1305         on the area
1306
1307     min_starttime_on_area=datetime(1993,1,1)+seconds(min_starttime_on_area);
1308     max_endtime_on_area=datetime(1993,1,1)+seconds(max_endtime_on_area);
1309
1310     %we are assuming that LIS will record until the last moment on the
1311     %interesting area. Truly, during this last time the LIS only would be
1312     %able to see the little, last, only "franja" of the area.
1313 end
1314
1315 function [date,time,lat,lon] = importcorrected(filename)
1316 %IMPORTFILE Import numeric data from a text file as column vectors.
1317 % [DATE1,TIME,LAT,LON] = IMPORTFILE(FILENAME) Reads data from text file
1318 % FILENAME for the default selection.
1319 %
1320 % [DATE1,TIME,LAT,LON] = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads
1321 % data from rows STARTROW through ENDROW of text file FILENAME.
1322 %
1323 % Example:
1324 % [date1,time,lat,lon] = importfile('paulino.txt.txt',1, 1713);
1325 %
1326 % See also TEXTSCAN.
1327
1328 % Auto-generated by MATLAB on 2018/07/31 13:20:32
1329

```

```

1330 % Initialize variables.
1331 delimiter = ';';
1332 if nargin<2
1333     startRow = 1;
1334     endRow = inf;
1335 end
1336
1337 % Format for each line of text:
1338 %   column1: datetimes ({yyyy-MM-dd}D)
1339 %   column2: datetimes ({HH:mm:ss}D)
1340 %   column3: double (%f)
1341 %   column4: double (%f)
1342 % For more information, see the TEXTSCAN documentation.
1343 formatSpec = '%{yyyy-MM-dd}D%{HH:mm:ss}D%f%f%[\n\r]';
1344
1345 % Open the text file.
1346 fileID = fopen(filename,'r');
1347
1348 % Read columns of data according to the format.
1349 % This call is based on the structure of the file used to generate this
1350 % code. If an error occurs for a different file, try regenerating the code
1351 % from the Import Tool.
1352 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ...
    delimiter, 'TextType', 'string', 'HeaderLines', startRow(1)-1, 'ReturnOnError', ...
    false, 'EndOfLine', '\r\n');
1353 for block=2:length(startRow)
1354     frewind(fileID);
1355     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
        'Delimiter', delimiter, 'TextType', 'string', 'HeaderLines', ...
        startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1356     for col=1:length(dataArray)
1357         dataArray{col} = [dataArray{col};dataArrayBlock{col}];
1358     end
1359 end
1360
1361 % Close the text file.
1362 fclose(fileID);
1363
1364 % Post processing for unimportable data.
1365 % No unimportable data rules were applied during the import, so no post
1366 % processing code is included. To generate code which works for
1367 % unimportable data, select unimportable cells in a file and regenerate the
1368 % script.
1369
1370 % Allocate imported array to column variable names
1371 date = dataArray{:, 1};
1372 time = dataArray{:, 2};
1373 lat = dataArray{:, 3};
1374 lon = dataArray{:, 4};
1375
1376 % For code requiring serial dates (datenum) instead of datetime, uncomment
1377 % the following line(s) below to return the imported dates as datenum(s).
1378
1379 % datel=datenum(datel);
1380 % time=datenum(time);
1381 end
1382
1383
1384 function [dir_list]=check_for_folders(read_dir,dir_list)
1385 disp('Looking for folders with HDF4 files inside');
1386 addpath(read_dir); %current reading directory
1387 folderinfoprev=dir(read_dir);
1388 folderinfo=folderinfoprev(~ismember({folderinfoprev.name},{'.','..','DS_Store'})); ...
    %DS_Store is a metadata file created by iOS environment
1389
1390 aretherefolders=cell2mat({folderinfo.isdir});
1391
1392 if any(aretherefolders)==true %check if there are folders inside the folder
1393
1394     namesarray={folderinfo(aretherefolders).name};
1395     %gives a cell array but in char
1396
1397     %foldernames=convertCharsToStrings(namesarray); %lets convert it to string

```

```

1398     for i=1:length(namesarray)
1399         % new_read_dir(i)=fullfile(read_dir,foldernames(i));
1400         new_read_dir=(fullfile(read_dir,(namesarray{i})));
1401
1402         [dir_list]=check_for_folders(new_read_dir,dir_list); %if the folder contains ...
1403             more folders, re-check
1404     end
1405
1406 else
1407     dir_list=[dir_list; read_dir]; %if the folder is a file folder save its ...
1408         directory and make it travel through the function
1409
1410 end
1411
1412 disp('Read directories will be:');
1413 for i=1:size(dir_list,1)
1414     disp(dir_list(i,:));
1415 end
1416 disp(' ');
1417 end
1418
1419 %NEW VERSION FUNCTIONS
1420 function ouput_data = get_LMA_data(filename, startRow, endRow)
1421 %IMPORTFILE Import numeric data from a text file as a matrix.
1422 % OUPUT_DATA = IMPORTFILE(FILENAME) Reads data from text file FILENAME
1423 % for the default selection.
1424 %
1425 % OUPUT_DATA = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from
1426 % rows STARTROW through ENDROW of text file FILENAME.
1427 %
1428 % Example:
1429 % ouput_data = importfile('total.LMA.file.noNoise.txt', 1, 4178);
1430 %
1431 % See also TEXTSCAN.
1432
1433 % Auto-generated by MATLAB on 2019/03/16 10:47:29
1434
1435 %% Initialize variables.
1436 delimiter = '\t';
1437 if nargin<2
1438     startRow = 1;
1439     endRow = inf;
1440 end
1441
1442 %% Format for each line of text:
1443 % column1: double (%f)
1444 % column2: double (%f)
1445 % column3: double (%f)
1446 % column4: double (%f)
1447 % column5: double (%f)
1448 % column6: double (%f)
1449 % column7: double (%f)
1450 % column8: double (%f)
1451 % column9: double (%f)
1452 % For more information, see the TEXTSCAN documentation.
1453 formatSpec = '%f%f%f%f%f%f%f%f%f%\n\r';
1454
1455 %% Open the text file.
1456 fileID = fopen(filename,'r');
1457
1458 %% Read columns of data according to the format.
1459 % This call is based on the structure of the file used to generate this
1460 % code. If an error occurs for a different file, try regenerating the code
1461 % from the Import Tool.

```



```

1537 %% Read columns of data according to the format.
1538 % This call is based on the structure of the file used to generate this
1539 % code. If an error occurs for a different file, try regenerating the code
1540 % from the Import Tool.
1541 dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', ',', ...
    'WhiteSpace', '', 'TextType', 'string', 'EmptyValue', NaN, 'HeaderLines', ...
    startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1542 for block=2:length(startRow)
1543     frewind(fileID);
1544     dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, ...
    'Delimiter', ',', 'WhiteSpace', '', 'TextType', 'string', 'EmptyValue', NaN, ...
    'HeaderLines', startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
1545     for col=1:length(dataArray)
1546         dataArray{col} = [dataArray{col};dataArrayBlock{col}];
1547     end
1548 end
1549
1550 %% Close the text file.
1551 fclose(fileID);
1552
1553 %% Post processing for unimportable data.
1554 % No unimportable data rules were applied during the import, so no post
1555 % processing code is included. To generate code which works for
1556 % unimportable data, select unimportable cells in a file and regenerate the
1557 % script.
1558
1559 %% Create output variable
1560 output_data = [dataArray{1:end-1}];
1561 end
1562
1563 function [rad.dischargealt] = ...
    typical_radiance_at_height(DISCHARGE_ALT,tolerance,lma,lis,chunksinfo)
1564
1565
1566 rad.dischargealt.medians = [];
1567 rad.dischargealt.averages = [];
1568 counter = 0;
1569 for i = 1:length(chunksinfo.chunks_w.both)
1570
1571     chunk_address = chunksinfo.chunks_w.both(i);
1572
1573     local.sources = chunksinfo.sources{chunk_address};
1574
1575
1576     if any(lma.salts(local.sources)>DISCHARGE_ALT-tolerance) ...
1577         && any(lma.salts(local.sources)<DISCHARGE_ALT +tolerance) %The ...
1578         heights of the sources is comprised in the tolerance
1579
1580         counter = counter +1;
1581
1582         local.events = chunksinfo.events{chunk_address}; %access the content of ...
1583         the cell at that address;
1584
1585         local.rads = lis.erad(local.events);
1586
1587         median.rads = median(local.rads);
1588         average.rads = mean(local.rads);
1589
1590         rad.dischargealt.medians(counter) = median.rads;
1591         rad.dischargealt.averages(counter) = average.rads;
1592
1593     end
1594
1595     rad.dischargealt.alt = DISCHARGE_ALT;
1596     rad.dischargealt.tol = tolerance;
1597     rad.dischargealt.nbins = counter;
1598 end

```

Bibliography

- [1] H. J. Kramer. *ISS Utilization: LIS (Lightning Imaging Sensor) on STP-H5's investigations of DoD*. Retrieved from <https://directory.eoportal.org/web/eoportal/satellite-missions/i/iss-lis>.
- [2] NASA,CHRC. *ISS Lightning Sensors Data Sets*. Retrieved from https://lightning.nsstc.nasa.gov/data/data_iss_lis.html.
- [3] Hugh J. Christian, Richard J. Blakeslee, and Steven J. Goodman. *Lightning Imaging Sensor (US) for the Earth Observing System*. NASA, February 1992.
- [4] P. M. Bitzer, H. Christian. *Timing Uncertainty of the Lightning Imaging Sensor*. J. Atmos. Oceanic Technol., 32, 453–460, <https://doi.org/10.1175/JTECH-D-13-00177.1>
- [5] Richard J. Blakeslee. *Lightning Imaging Sensor (LIS) on ISS and Plans for Sustained Ground Measurements in Support of GLM Cal/Val*. Joint MTG LI Mission Advisory Group and GOES-R GLM Science Team Workshop, Rome, Italy. May 2015
- [6] E.P. Krider. *Atmospheric Electricity On-line Course*. University of Arizona, 2015. Retrieved from: http://www.atmo.arizona.edu/students/courselinks/spring15/atmo589/ATMO489_on-line/CONTENTS.html.
- [7] Blumenfeld, J. *New Lightning Imaging Sensor to be Installed on the International Space Station*. Retrieved from <https://earthdata.nasa.gov/lis-on-iss>.
- [8] S. Goodman, D. Mach, W. Koshak, R. Backslee. *Algorithm Theoretical Basis Document*. Center for Satellites Applications and Research. September 24, 2010.
- [9] R. J. Blakslee, H. Christian et al. *Lightning Imaging Sensor (LIS) for the International Space Station (ISS): Mission Description and Science Goals*. XV International Conference on Atmospheric Electricity, 15-20 June 2014, Norman, Oklahoma, U.S.A.
- [10] S.J. Goodman et al. *The GOES-R Geostationary Lightning Mapper (GLM)*. Atmospheric Research 125–126 (2013) 34–49.
- [11] C. Lennon, L. Maier. *LIGHTNING MAPPING SYSTEM* NASA, Florida.
- [12] R.J. Thomas et al. *Observations of VHF Source Powers Radiated by Lightning*. Geophysical Research Letters, Vol. 28, NO. 1, pages 143-146, January 1,2001.
- [13] F. Fabr , J. Montany  et al. *Analysis of energetic radiation associated with 1 thunderstorms in the Ebro delta region in Spain*. Geophys. Res. Atmos., 121, doi:10.1002/2015JD024573.
- [14] The HDF Group. *HDF User's Guide*. Retrieved from <https://www.hdfgroup.org/solutions/hdf4/>, June 2017.
- [15] NetCDF 4.6.1. *An introduction to NetCDF* Retrieved from: <https://www.unidata.ucar.edu/software/netcdf/docs/netcdf.introduction.html>.
- [16] Blakeslee Richard J., Douglas M. Mach, Michael F. Stewart, Dennis Buechler and Hugh Christian. 2017. *Non-Quality Controlled Lightning Imaging Sensor (LIS) on International Space Station (ISS) Provisional Science Data*. Dataset available online from the NASA Global Hydrology Center DAAC, Huntsville, Alabama, U.S.A. DOI: <http://dx.doi.org/10.5067/LIS/ISSLIS/DATA204>

- [17] H. Christian, M. McCook et al. *A Lightning Primer*. Retrieved from <https://lightning.nsstc.nasa.gov/primer/index.html>. September 2018.
- [18] R. J. Thomas, Paul L. Krehbiel et al. *Comparison of Ground-based 3-dimensional lightning mapping observations with satellite-based LIS observations in Oklahoma* Geophysical Research Letters, vol. 27 n° 12, pages 1703-1706, June 15, 2000.
- [19] Liu Feng et al. *The Study of Active Atoms in High-Voltage Pulsed Coronal Discharge by Optical Diagnostics*. 2005 Plasma Sci. Technol. 7 2851
- [20] Servei Meteorològic de Catalunya. *Normals Climàtiques Recents*. Retrieved from www.meteo.cat/wpweb/climatologia/serveis_i_dades_climatiques/normals_climatiques_recents/. April 2019.
- [21] P André et al. *The calculation of monatomic spectral lines' intensities and composition in plasma out of thermal equilibrium; evaluation of thermal disequilibrium in ICP torches*. J. Phys. D: Appl. Phys. 30 (1997) 2043-2055.
- [22] E.V. Koryukina. *Calculation of the Emission Spectra of Atoms and Ions in the External Electric field*. 2nd International Congress on radiation physics, high current electronics and modification of materials. Tomsk Polytechnic University (2006).
- [23] C. Trassy, A. Tazeem. *Simulation of atomic and ionic absorption and emission spectra for thermal plasma diagnostics: application to a volatilisation study in a plasma jet*. Spectrochimica Acta Part B: Atomic Spectroscopy 54 (1999) 581-602.

