UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DEVELOPING AND TESTING AN ULTRA-LOW-COST STAR TRACKER FOR ATTITUDE DETERMINATION OF NANOSATELLITES

TESIS PARA OPTAR AL GRADO DE DOCTOR EN INGENIERÍA ELÉCTRICA

SAMUEL TOMÁS GUTIÉRREZ RUSSELL

PROFESOR GUÍA:
MARCOS DÍAZ QUEZADA
PROFESOR CO-GUÍA:
CÉSAR FUENTES GONZÁLEZ

MIEMBROS DE LA COMISIÓN:
MARCOS ORCHARD CONCHA
HOLGER DRASS
SHINICHI NAKASUKA

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE DOCTOR EN INGENIERÍA ELÉCTRICA
AUTOR: SAMUEL GUTIÉRREZ RUSSELL
FECHA: 2023
PROFESOR GUÍA: DR. MARCOS DÍAZ QUEZADA

DESARROLLO Y PRUEBA DE UN RASTREADOR DE ESTRELLAS DE COSTO ULTRA BAJO PARA LA DETERMINACIÓN DE ORIENTACIÓN DE NANOSATÉLITES

Esta Tesis de Doctorado se enfoca en el desarrollo y prueba de un rastreador de estrellas para nanosatélites de costo ultra bajo. Fue diseñado para ser usado en las misiones satelitales del Laboratorio de Exploración Espacial y Planetaria (SPEL).

Los sensores absolutos de orientación entregan capacidades de apuntamiento de alta precisión, pero constituyen un costo significativo de la misión CubeSat. Para resolver este problema desarrollé SOST (SPEL - Open Star Tracker), el cual usa componentes comerciales salidos del estante (COTS). Su uso en misiones espaciales es un área de investigación en crecimiento.

Se desarrolló un algoritmo que resuelve el problema Perdido-En-Espacio (LIS). El algoritmo y el hardware se probaron exhaustivamente. SOST provee una precisión en el eje de puntería mejor a un minuto de arco el 97.3% del tiempo, demorando cerca de 12 segundos, si se utiliza una separación de catálogo de 10 grados. La plataforma es gratuita y está disponible para investigadores del área CubeSat.

Se integró SOST en los nanosatélites SUCHAI en términos de software y hardware. Se propone una serie de experimentos para evaluar su rendimiento en el espacio. Finalmente, se estudia el comportamiento del algoritmo LIS bajo efectos de radiación que podrían aparecer en vuelo.

ABSTRACT OF THE THESIS FOR THE DEGREE OF
DOCTOR IN ELECTRICAL ENGINEERING
AUTHOR: SAMUEL GUTIÉRREZ RUSSELL
DATE: 2023
ADVISOR: DR. MARCOS DÍAZ QUEZADA

DEVELOPING AND TESTING AN ULTRA-LOW-COST STAR TRACKER FOR ATTITUDE DETERMINATION OF NANOSATELLITES

This doctoral thesis focuses on developing and testing an ultra-low-cost star tracker for attitude determination in nanosatellites. This sensor is designed to be used in the satellite missions of SPEL (Space and Planetary Exploration Laboratory).

Absolute attitude estimation sensors provide high accuracy pointing capabilities, but constitute an expensive fraction of CubeSat mission's budget. To solve this problem, I developed SOST (SPEL - Open Star Tracker). This development rests on commercial-off-the-shelf (COTS) hardware. The use of COTS in space missions is a growing research area.

A new algorithm to solve the Lost-In-Space (LIS) problem was developed. The algorithm and the hardware platform were tested extensively. SOST provides a mean precision below one arcminute for the boresight direction 97.3% of the time in almost 12 seconds when a catalog segment separation of 10 degrees is used. The platform is free and available to CubeSat researchers.

The integration of SOST into the SUCHAI nanosatellites has been done in terms of software and hardware. A list of experiments to evaluate the performance of the STT while in orbit is proposed. Finally, the behavior of the LIS algorithm under radiation effects that could appear in flight is studied.

A las personas de mi vida:
mi hijo Simón,
mi hermano Alexis,
mi madre Nancy,
y mi abuela Adriana.

"*Education is not something you can finish.*"
— Isaac Asimov

Acknowledgments

This research has been funded by grant CONICYT-PFCHA/2017-21171862.

I would like to sincerely acknowledge my two advisors, Marcos Díaz and César Fuentes. The many conversations we had were inspiring to work and imagine new solutions. They motivated me by showing me their hard work. Also, I want to acknowledge professor Shinichi Nakasuka from the University of Tokyo. He gently received and taught me when I was in my internship in Japan in 2019.

I also want to acknowledge the human group that composes the Electrical Engineering Department of the University of Chile. The professors, the administrative staff, and the assistants constantly do a great job and make the students feel welcome. Finally, I want to thank everyone who is or was part of SPEL. I learned so many things working here with them. They showed me how to work hard as a team.

Table of Content

List of Tables

List of Figures

List of Acronyms

ADCS	Attitude Determination and Control System
AO	Atomic Oxygen
APS	Active-Pixel Sensor
BSC	Bright Star Catalogue
CCD	Charge-Coupled Device
CMEs	Coronal Mass Ejections
CMOS	Complementary Metal–Oxide–Semiconductor
COTS	Commercial Off-The-Shelf
DCM	Direction Cosine Matrix
DEC	Declination (δ)
ECEF	Earth-Centered, Earth-Fixed
ECI	Earth-Centered Inertial
EPS	Electrical Power Subsystem
FOV	Field Of View
FS	Flight Software
HI	Heliospheric Imager
HST	Hubble Space Telescope
ISS	International Space Station
JPL	Jet Propulsion Laboratory
LEO	Low Earth Orbit

LIS Lost-In-Space

LP Langmuir Probe

MEMS Micro-Electro-Mechanical Systems

NASA National Aeronautics and Space Administration

OBC On-Board Computer

RA Right Ascension (α)

RAAN Right Ascension of Ascending Node

RPi Raspberry Pi

SAA South Atlantic Anomaly

SEE Single Event Effects

SOST SPEL - Open Star Tracker

SPEL Space and Planetary Exploration Laboratory

SSTL Surrey Satellite Technology Ltd

STEREO Solar TErrestrial RElations Observatory

STT STar Tracker

SUCHAI Satellite of the University of CHile for Aerospace Investigation

TEC Total Electron Content

TID Total Ionizing Dose

TLE Two-Line Element set

TRL Technology Readiness Level

TRX Transceiver

UV Ultraviolet

YPR Yaw-Pitch-Roll

Chapter 1

Introduction

Attitude determination is crucial for any spacecraft that requires knowing its orientation while flying. This attitude information is essential not only for maximizing energy collection or efficient communication but also for other payload requirements, such as a telescope or antenna orientations. Among the attitude determination sensors such as sun sensors, horizon sensors, magnetometers, and inertial sensors, the Star Tracker (STT) stands out for providing absolute attitude estimation and being the most precise [1], [2]. Thus, STT is a critical component in space missions.

Attitude sensors have been part of satellite missions since the dawn of the space era. The first CCD-based STT was developed at JPL in 1976 [3]. From this moment, image-sensor-based STT, together with the use of estimation filters [4], has been continuously used and improved over time. However, the developed technology and algorithms were not shared because the spacecraft development evolved as a demonstration of power in the polarized Cold War era. As this era came to an end, knowledge was slowly transferred to the private sector. The commercial exploitation of space has reduced the costs, but the knowledge has remained in the hands of a limited few. STTs have followed this trend; therefore, they are generally available as commercial black boxes.

In the late 90s, the CubeSat emerged as a standardized satellite [5] to improve space technology education. CubeSat is a standardized shape platform with a cubical base unit of $10\,\text{cm} \times 10\,\text{cm} \times 10\,\text{cm}$ (1U). The standardization reaches the deployer unit in rockets supporting sizes of 1U, 1.5U, 2U, 3U, 6U, and even 12U thus far. This standard has reduced the size, cost, and development time of this type of satellite. Cost reduction has resulted in growing access to space technologies in developing countries [6], [7], creating a path towards the research in CubeSat and new space-related technologies. CubeSats have rapidly evolved and been actively used for research and commercial applications [8], [9]. Its success has been of such a great level, that today a 6U CubeSat is capable of detecting exoplanet transit [10], and solutions are developed to communicate CubeSats constellations [11]. These success stories demonstrate the feasibility of doing science using CubeSats, emphasizing the importance of research and development of each of its subsystems.

1.1 Motivation

On June 23, 2017, the first CubeSat developed in SPEL (Space and Planetary Exploration Laboratory), the SUCHAI 1, was launched from India, carrying a series of "proof of concept" experiments. The SUCHAI 1 (Satellite of the University of Chile for Aerospace Investigation) was a 1U CubeSat without attitude control [12]. This development starts a space program for education and research [7] at the University of Chile, based on the use of the CubeSat standard. Many Universities around the world are using the CubeSat standard to develop Educational and Scientific Space programs, as the University of Tokyo [13], TU Delft [14], and Cal-Poly [15], among others. Continuing with the satellite program in SPEL, we have developed three 3U CubeSats: PlantSAT, SUCHAI 2, and SUCHAI 3. These satellites went into orbit on Friday, April 1, 2022, in the mission Transporter-4 from SpaceX, a Falcon-9 vehicle launched from Space Launch Complex 40 at Cape Canaveral Space Force Station in Florida [16]. At least two of these satellites are planned to work collaboratively. These new satellites include attitude estimation sensors and control actuators. The missions of these new satellites are related to the study of the space environment and its impact on electronic components and biological systems. These experiments require accurate attitude estimation at a low frequency (2 or 3 samples per minute). Therefore, the requirements of the new SUCHAI missions fall under the category of high precision absolute attitude measurement at a lower frequency where the cost of commercial STT is not justified. The description of two scientific payloads will exemplify this decision.

The first example of one of the payloads in our missions that motivated the development of an STT is a fixed-voltage flat-probe Langmuir Probe (LP), shown in Figure 1.1. The flat-probe (or patch) of the LP is located on the ram-facing side of the satellite. The LP will estimate the plasma ion density, which is assumed the same as the electron density under thermal equilibrium conditions. To measure the ion density and avoid the potential charging issue of the spacecraft, a negative potential is used. The proper estimation requires that the patch normal vector ($\hat{n}$) is parallel to the spacecraft ram velocity ($\hat{v}$) (that means, both vectors pointing in the same direction). In this configuration, the ions are likely to hit the LP patch over the surface of the CubeSat. The error in the estimation of the densities is less than 5% when the angle between $\hat{n}$ and $\hat{v}$ is less than 20°. Therefore the main requirement is to estimate the attitude of the satellite when it is spinning at a slow angular rate, but not frequently. For example, a spinning rate of 20 arcseconds/s means a rotation of 40 arcminutes after 120 seconds, which is still well inside the range ($\pm 20°$) at which the LP will deliver a proper ion/electron density measurement.

Another payload that requires accurate attitude estimation but at a low sampling rate is a phased array communication link system. With this communication system, we expect to electronically steer the antenna array beam faster than pointing by changing the satellite attitude. The phased array is not only intended for communications but also for measuring the Total Electron Content (TEC) and magnetic field if the system is appropriately designed. Similarly to the LP, the phased array experiment does not require a fast attitude sampling rate, since the available number of elements of the array in a 3U CubeSat produces a half-power beam-width of 20°.

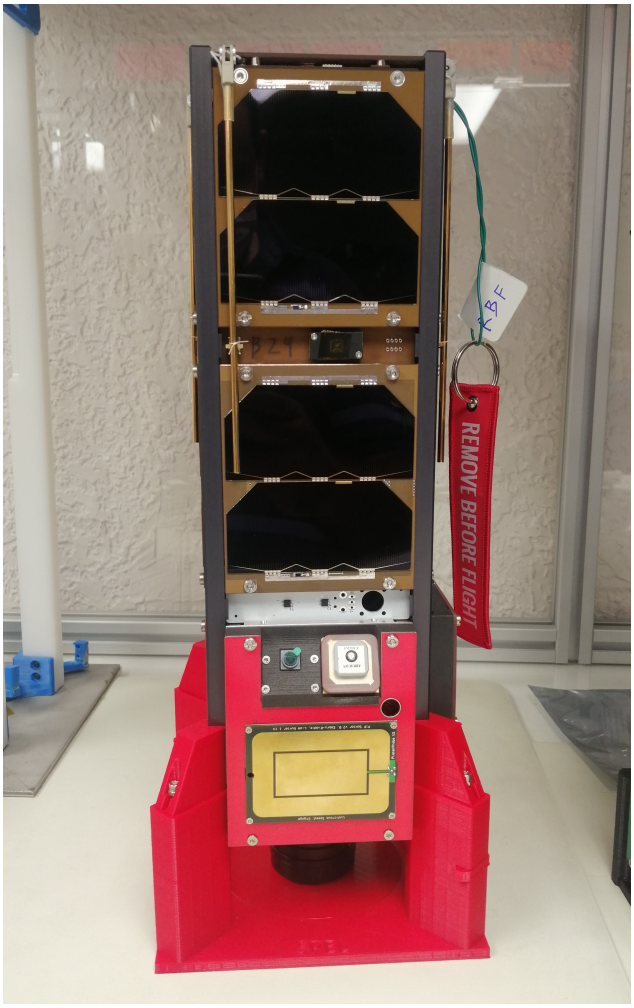


Figure 1.1: Satellite SUCHAI-2 fully assembled. At the bottom of the satellite, we can see the golden flat-probe Langmuir Probe (LP).

1.2 Problem definition

Although several STTs are available on the market, pricing for this kind of sensor is restrictive, especially for groups from developing countries where budgets are more constrained. Teams must opt for low-cost sensors, like sun sensors or magnetometers, which cannot determine attitude accurately, limiting the satellite capabilities. The restricted access to this kind of sensor is one of the main problems that this work comes to solve. Table 1.1 summarizes the main characteristics of the most commonly used attitude determination sensors [17]–[23]. From Table 1.1, it is possible to notice a "cost and technological" gap between low and high precision attitude sensors, in particular for less demanding mission requirements. This gap means, there is not currently available in the market high precision sensors (like an STT) at a low cost.

Another problem that needs to be faced is complying with the specific characteristics that a Star Tracker must fulfill to serve as one properly. The development of a full platform, considering the CubeSat restrictions, must contemplate not only a fast algorithm to solve

Attitude sensors	**Typical precision**	**Sampling rate (Hz)**	**Volume (1U)**	**Average power (W)**	**Average cost (USD)**
Gyroscope [17]	Drift rate of 1°/hr	100	1/1000	0.02	30
Magnetometer [18]	0.5° - 3°	10	1/1000	0.02	15
Earth Sensor [19]	0.25°	1	1/20	0.1	15000
Fine Sun Sensor [20]	6'	5	1/50	0.04	12000
SOST	30" - 60" (cross-boresight)	0.05	1/10	<2.6	70
Star Tracker [21]	2" (cross-boresight)	10	1/4	<1	40000

Table 1.1: This table shows different attitude determination sensors with their main features. It can be noted a performance and cost gap between the low precision sensors (in the range of degrees) with the high precision sensors (in the range of arcseconds). This gap means, there is not currently available in the market high precision sensors (like an STT) at a low cost.

the LIS problem but also the hardware that will use, an appropriate camera, the energy consumption, and mass (for board and camera), only for mention a few restrictions. Table 1.2 shows the main challenges to solve when developing a fully functional STT. Although many research papers and books focus on solving or improving a particular side of the global STT problem, there is a lack of information on developing a fully functional open-source STT platform. It is relevant to highlight that this dissertation is the first open-source, thoroughly tested STT solution put into space that exists in the literature. For that reason, I can not compare this work in terms of performance, with previously published works. This lack of literature is one of the main reasons to propose and develop a fully-functional platform, which inserts in the "cost and technological" gap previously described.

Component type	**Element**	**Challenges**
Hardware	camera	Capable of visualizing stars in a short exposure time
	microcontroller/microprocessor	It is of low electric consumption, low mass, and small volume
Software	Stars extraction	An appropriate algorithm to extract the brightest sources from image
	Stars identification	An efficient catalog searching, given the extracted stars from picture
	Finding the attitude	When stars identified, find the satellite attitude

Table 1.2: Challenges faced when developing a fully-functional STT platform under CubeSat restrictions.

To face the problems previously described, this work focuses on the development and testing of ultra-low-cost and easy-to-implement STT. A small single-board computer was adopted as the hardware platform. The proposed sensor uses an accessible miniaturized computer, the Raspberry Pi (RPi) [24], and its camera, which can be easily integrated into the RPi. The RPi was selected because of its low cost and processing power, which is enough to serve as STT, as is demonstrated in this dissertation. Another advantage of the RPi platform is that it is mostly open hardware, which means it can be adapted and modified to fit in the CubeSat standard eventually. I named this platform **SOST** (SPEL - Open

Star Tracker), an ultra-low-cost and easy to implement STT solution, based on an RPi and open source astronomy software. This platform comes to fill the "cost and technological" gap previously described. This development seems to be on the right path since new research trends point to reduce the size, weight, and power of STT through the use of COTS elements [25]. Furthermore, previous works [26]–[29] have shown that fully developing an STT brings two different benefits: first, applied knowledge in the attitude determination field. Second, a developed project is open for our SPEL group and the whole CubeSat community.

The development of SOST is in nature multidisciplinary, taking advantage of astronomy knowledge. SOST was developed from and as an open platform to facilitate the contribution of multiple groups. In this way, the features and the flight heritage of the system could be accelerating. Open source platforms enable better interaction with a broader number of experts that may provide a faster, successful outcome. This multi collaboration approach has already been used in CubeSats [30], [31]. Through this approach, smaller actors from groups like ours can contribute to the development of a more precise and robust STT platform.

1.3 Hypotheses

1. Using probed astronomy software, it is possible to develop and test new Star Tracker algorithms capable of reaching arcminutes precision or better.
2. The developed algorithm and the necessary software tools fit in a COTS platform to get a frequency better than one sample per minute. This COTS component complies with the CubeSat restriction of mass, volume, and power consumption.
3. It is possible to integrate an ultra-low-cost star tracker in a CubeSat in terms of software and hardware.
4. The effects of radiation in LEO over image detectors can be simulated, allowing the performance study and analysis of LIS algorithms.

1.4 Objectives

1.4.1 General objective

This dissertation presents the development, testing, and integration of a new ultra-low-cost attitude determination sensor under CubeSat restrictions, open to the CubeSat community researchers. This development included the software, selecting suitable hardware, and the complete testing of this software-hardware joint. This work reduced the development time and improved the performance of this kind of sensor, filling a gap between the high-precision (but high cost) and the low-cost (but low precision) attitude determination sensors.

1.4.2 Specific objectives

The following specific objectives were fulfilled to achieve the general objective:

- An algorithm capable of solving the LIS problem using probed and robust open-source software was developed.
- The performance of the developed algorithm by testing it using images from satellites currently operating in space was analyzed.
- A suitable COTS platform, complying with the CubeSat restrictions, was sought and selected. In this platform, the developed algorithm and its overall performance were tested taken night-sky pictures.
- The algorithm was evaluated by introducing radiation disturbances similar to those that probably occur in orbit, for example, single event effects (SEE).
- SOST became part of a framework that allows it to function in space inside a CubeSat, such as the SUCHAI Flight Software (FS).

1.5 Contributions

The following are the main contributions of this Thesis:

- The development of an algorithm capable of solving the LIS problem based on well-known and proven astronomical tools. This algorithm is adaptable to different fields of view.
- A methodology is proposed to validate LIS algorithms through the use of on-space images, particularly from the STEREO satellite from NASA.
- It is demonstrated that a COTS platform can be capable of carrying on the process of absolute attitude determination. This is done through an exhaustive process of testing.
- The analysis of radiation events that probably can occur in LEO and its influence over the LIS algorithm.
- The integration of a simple-to-use ultra-low-cost platform in a CubeSat platform. This allows us to currently have an STT platform in space, which enables us to deeply study the space environment and use it through a series of experiments that can be carried on from Earth.
- Regarding the existence of publications derived from this work, it is possible to mention the paper published in the journal IEEE Access (DOI: 10.1109/ACCESS.2020.3020048) titled "*Introducing SOST: An ultra-low-cost star tracker concept based on a Raspberry Pi and open-source astronomy software.*" Additionally, I presented a poster at the first postgraduate congress in engineering, science, and innovation of the Faculty of Physical and Mathematical Sciences of the University of Chile, titled "*Development, test, and integration of an ultra-low-cost star tracker in the new SUCHAI satellite missions of the University of Chile*" (https://doi.org/10.34720/ntsv-yj06).

1.6 Thesis outline

This Thesis is structured in the following way:

- Chapter one presents the introduction, where the problem is established, why it is important to solve it, and the associated restrictions. Also, there are the hypotheses, the objectives, and the contributions of this work.
- Chapter two presents the literature review. Here it is revised the fundamentals of attitude determination: sensors, representations, reference systems, and transformations. There are also presented classifications of algorithms for star identification, the use of *a priori* information, different existing tools for STT development, previous development of STT for CubeSats, the use of COTS in space projects, and a review of the space environment, focusing on effects produced by space radiation.
- Chapter three describes the developed algorithm that solves the LIS problem. Also, it is shown how this algorithm can be adapted for different fields of view, and a graphic visualization of the distribution of stars in catalogs.
- Chapter four presents the complete evaluation of the proposed solution, both in terms of the algorithm and the selected platform.
- Chapter five presents the integration of the platform in a CubeSat project in terms of hardware and software. Also, there are presented the proposed experiments to execute in space.
- Finally, chapter six contains the conclusions of this Thesis and a proposal for future work.

Chapter 2

Literature review

As an introduction to the Star Tracker subject, I will cite a beautiful definition from NASA [32]:

> "The concept of a Star Tracker probably can be traced back to the early sailors who used to navigate the open seas. The concept is they use star field patterns in order to tell them where they're pointed, while they are sailing from east to west or north to south, whatever direction they were going.
>
> The Star Tracker is a lot more complicated in that it uses the entire star field pattern to provide attitude knowledge or, in the laymen's term, more of the direction in which the spacecraft is pointing while it is orbiting the Earth or doing wherever its science mission requires it to do. So, essentially, it's an optical device to access the eyes of the spacecraft."

That is, a Star Tracker is a sensor that detects stars and compares their pattern with known stars from a stellar catalog. Then, it computes the precise orientation in which the camera is pointed at the sky and deduce the satellite's attitude.

2.1 Fundamentals of attitude determination

An STT is a sensor for precise attitude determination. But what is *attitude determination*? This section answers this question. Attitude determination is a widely developed knowledge area, in which many books have been written. I will present only the most relevant parts of each topic; more information can be found in each corresponding reference.

2.1.1 Sensors for attitude determination

According to Wertz (2012) [33], attitude determination is "the process of computing the orientation of the spacecraft relative to either an inertial reference or some object of interest, such as the Earth. This typically involves several types of sensors on each spacecraft and sophisticated data processing procedures. The accuracy limit is usually determined by a combination of processing procedures and spacecraft hardware." The different kinds of sensors were already named in Table 1.1, but now they will be described.

Sun sensors

This sensor is used to determine the direction of the sun. Usually, they have a large field of view ($> 120^\circ \times 120^\circ$) to maximize the probability of finding the sun in the sky. This sensor can be fine or coarse, digital or analog. Coarse sun sensors are based on a single photodetector, placing one of these on each face of the satellite. Its accuracy is in the order of degrees. Fine sun sensors are four-quadrant detectors, able to get more precision. Its accuracy is in the range of tenths of degrees.

Horizon sensors

This sensor determines the position of the Earth's horizon relative to the spacecraft's position in orbit. Usually, detecting the sharp thermal discontinuity at the earth-space horizon using infrared sensors [34]. Its accuracy is in the range of tenths of degrees.

Magnetometers

This sensor measures the Earth's magnetic field, and the derivative of this can also be computed. Trough the use of an accurate model of the Earth's magnetic field, this sensor is capable of doing three-axis attitude determination. Its accuracy is in the range of tenths of degrees.

Gyroscopes

This sensor measures the spinning rate of the spacecraft. It can not provide absolute attitude measurements, so usually its data can be fused with another sensors to provide absolute attitude estimation. This sensor typically presents a drift rate of 1 degree/hour.

Star Trackers

This sensor takes a picture of the stars and uses it to precisely compute the attitude. Because it uses the "fixed" positions of stars, it is the most precise attitude sensor, with precision ranging from arcminutes to tenths of arcseconds.

2.1.2 Attitude determination modes

Using an STT, there are basically two different ways of obtaining the attitude information of the satellite, and this depends on whether you have *a priori* information about it or not.

Lost-In-Space

When you do not have any previous information about the attitude of the satellite, you need to look at the full star catalog for coincidences with the image. This is known as the "Lost-In-Space" (LIS) problem. There are many different algorithms to identify stars and get the attitude, and they are described in section 2.2.

Tracking mode

When we already know the attitude at some specific time, it is possible to continuously determine it, propagating that information in time using mathematical models and fusing data from the STT with other sensors. This is known as the "tracking" mode.

2.1.3 Attitude representations

As previously stated, attitude means the orientation of the spacecraft in space, referred to some specific reference frame [33]. Generally, a reference frame is attached to the spacecraft (which is called the *body frame*), in which the base vectors of this *body frame* are aligned with the body's principal axes. Different frames can be used as a reference for attitude, some of them are inertial, and some others do not. The common ways to represent attitude are presented below.

Direction Cosine Matrix

To represent the orientation of a frame with respect to another, it is possible to reference each base vector for one frame with respect to the base vectors from another base [35]. That is, suppose that we have two reference frames $\mathcal{N}$ and $\mathcal{B}$, defined through a set of orthonormal right-handed set of vectors $\{\hat{n}\} = \{\hat{n}_1\, \hat{n}_2\, \hat{n}_3\}$ and $\{\hat{b}\} = \{\hat{b}_1\, \hat{b}_2\, \hat{b}_3\}$, respectively. The set of orthonormal base vectors $\{\hat{b}\}$ can be expressed in terms of the base vectors $\{\hat{n}\}$ as:

$$\{\hat{b}\} = \begin{pmatrix} \cos\alpha_{11} & \cos\alpha_{12} & \cos\alpha_{13} \\ \cos\alpha_{21} & \cos\alpha_{22} & \cos\alpha_{23} \\ \cos\alpha_{31} & \cos\alpha_{32} & \cos\alpha_{33} \end{pmatrix} \{\hat{n}\} = [C]\,\{\hat{n}\} \tag{2.1}$$

where the matrix $[C]$ is called the direction cosine matrix (DCM). Every matrix element has the form $\cos\alpha_{ij}$, which means the component of the vector base $\hat{b}_\mathrm{i}$ in the vector base $\hat{n}_j$. For example, $\cos\alpha_{23}$ means the component of the vector $\hat{b}_2$ in the base $\hat{n}_3$, as is shown in Figure 2.1.

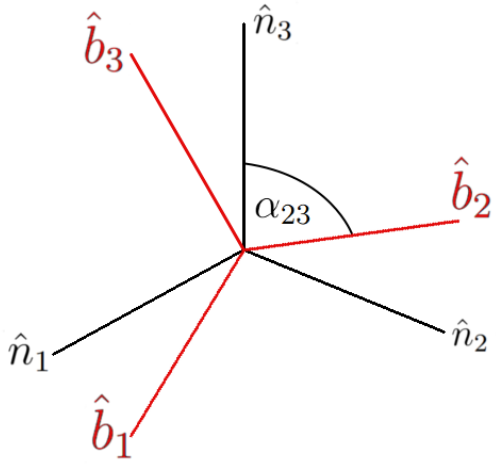


Figure 2.1: The figure shows the reference frame $\mathcal{N}$ (in black) and the rotated reference frame $\mathcal{B}$ (in red), with their corresponding base vectors. As an example, it shows the angle α_{23}, which is the angle between the vector $\hat{b}_2$ and the vector $\hat{n}_3$.

Of the nine director cosines, only three are independents due to the orthogonality of the unit vectors. This restriction leads to the orthogonality of the DCM, which means: $[C]^{-1} = [C]^T$. Therefore, $[C]$ can also represent the base vectors $\{\hat{n}\}$ in terms of the base vectors $\{\hat{b}\}$, used in the following way:

$$\{\hat{n}\} = \begin{pmatrix} \cos\alpha_{11} & \cos\alpha_{21} & \cos\alpha_{31} \\ \cos\alpha_{12} & \cos\alpha_{22} & \cos\alpha_{32} \\ \cos\alpha_{13} & \cos\alpha_{23} & \cos\alpha_{33} \end{pmatrix} \{\hat{b}\} = [C]^T\,\{\hat{b}\} \tag{2.2}$$

This DCM representation needs nine elements to define the orientation of the body with respect to a specific frame. For this reason, sometimes it is better to use other representations, with fewer elements and consequently taking fewer computing resources.

Euler angles

Euler angles describe the attitude of the reference frame $\mathcal{B}$ relative to $\mathcal{N}$, through three successive rotation angles $(\theta_1, \theta_2, \theta_3)$ about the body frame axes $\{\hat{b}\}$ [36]. There are at least twelve different Euler angle-axis sequences and can be grouped in two different sets:

Symmetric: ZXZ, ZYZ, XZX, XYX, YXY, and YZY.

Asymmetric: ZYX, ZXY, XZY, XYZ, YXZ, and YZX.

Rotation sequences can be *intrinsic* or *extrinsic*. Extrinsic means that the original coordinate system remains motionless as the rotations go. In contrast, intrinsic means the rotation is always based on the rotating coordinate system, solidary with the moving object. A useful property is that any extrinsic rotation is equivalent to an intrinsic rotation by the same angles but inverted order of elemental rotations and vice versa.

Satellite orientation (and spacecraft in general) is commonly described through the Euler angles *yaw*, *pitch*, and *roll* (ψ, θ, ϕ). To transform components of a vector in the $\mathcal{N}$ frame into the $\mathcal{B}$ frame, the reference axes are first rotated about the $\hat{b}_3$ axis by the yaw angle ψ, then about the $\hat{b}_2$ axis by the pitch angle θ and finally about the $\hat{b}_1$ axis by the roll angle ϕ. The standard *yaw-pitch-roll* angles are the (ZYX or 3-2-1) set of Euler angles, which is shown in Figure 2.2.

Quaternions

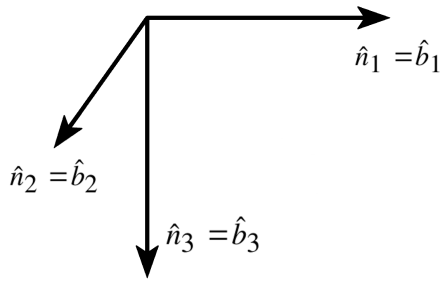
The attitude representation using Euler angles is easy to develop and to visualize, but computationally intense. The quaternion representation is based on Euler's rotational theorem, which states that the relative orientation of two coordinate systems can be described by only one rotation about a fixed axis [37], [38]. One fundamental restriction over the quaternions to represent attitude is that $|\mathbf{q}| = 1$, that is, the quaternion must be unitary.

A quaternion $\mathbf{q}$ is a 4×1 matrix which elements consists of a scalar part and a vector part. That is:

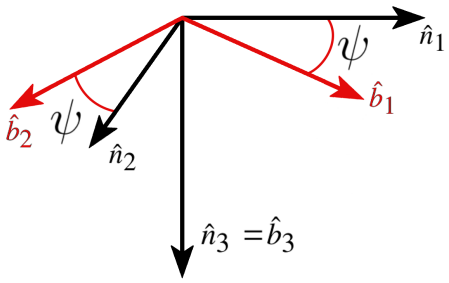
$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos \theta/2 \\ \hat{\mathrm{e}}_x \sin \theta/2 \\ \hat{\mathrm{e}}_y \sin \theta/2 \\ \hat{\mathrm{e}}_z \sin \theta/2 \end{bmatrix} \tag{2.3}$$

The quaternion representation is not very intuitive but is computationally light (only handles four elements). For that reason, it is the most used attitude presentation. If you want to have an idea of the attitude that the quaternion represent, this quaternion can be converted to Euler angles in any rotation sequence.

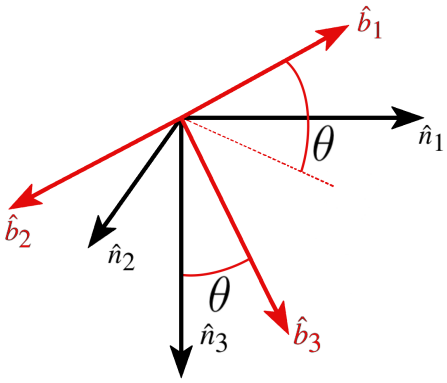
An equation that will be useful in the development of this Thesis (especially in section 2.3.3), is the derivative of the quaternion, which is given by:



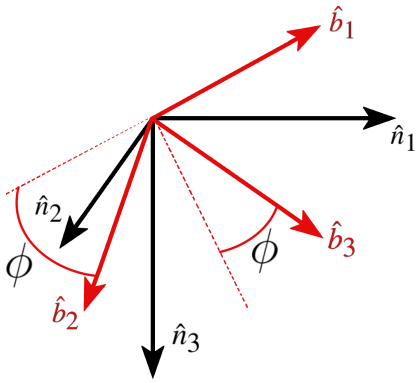
(a) Initial coordinate axis (overlapping).



(b) Rotation about $\hat{b}_3$ (*yaw*).



(c) Rotation about $\hat{b}_2$ (*pitch*).



(d) Rotation about $\hat{b}_1$ (*roll*).

Figure 2.2: Representation of Euler angles known as the *yaw-pitch-roll* sequence. Figure (a) shows the initial coordinate axis, where the two axes overlap. Then, Figure (b) shows a rotation about the $\hat{b}_3$ axis through an angle ψ (*yaw*). Following, Figure (c) shows a rotation about the $\hat{b}_2$ axis through an angle θ (*pitch*). Finally, Figure (d) shows the rotation about the $\hat{b}_1$ axis through an angle ϕ (*roll*).

$$\dot{\mathbf{q}} = \frac{1}{2}\,\Omega(\omega)\,\mathbf{q} \tag{2.4}$$

where the omega operator $\Omega(\omega)$ has been used:

$$\Omega(\omega) = \begin{bmatrix} 0 & -\omega^T \\ \omega & \lfloor\omega\rfloor_\times \end{bmatrix} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{2.5}$$

also, it has been used the expression $\lfloor \mathrm{X} \rfloor_\times$, which expands a vector X into a 3×3 skew-symmetric matrix:

$$\lfloor \mathrm{X} \rfloor_\times = \begin{bmatrix} 0 & x_3 & -x_2 \\ -x_3 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{bmatrix} \tag{2.6}$$

2.1.4 The rotation matrix

A rotation matrix is a mathematical operator used to perform a rotation. If the rotation takes a vector and rotates it at a specific angle about a defined axis, it is called an *active rotation*. If the vector remains unchanged and the rotation applies over the coordinate system, it is a *passive rotation*. The standard (passive) elemental rotation matrix in 2D is the following:

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{2.7}$$

The standard rotation matrix in 3D, in every axis, are the following:

$$R_x(\theta_1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{pmatrix} \tag{2.8}$$

$$R_y(\theta_2) = \begin{pmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{pmatrix} \tag{2.9}$$

$$R_z(\theta_3) = \begin{pmatrix} \cos\theta_3 & \sin\theta_3 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

Using these elemental rotations, the *yaw-pitch-roll* sequence (see Figure 2.2) can be computed as a series of elemental rotations:

$$
\begin{aligned}
R(\psi,\theta,\phi) &= R_x(\phi)\,R_y(\theta)\,R_z(\psi) \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ \sin\phi\sin\theta & \cos\phi & \sin\phi\cos\theta \\ \cos\phi\sin\theta & -\sin\phi & \cos\phi\cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{pmatrix}
\end{aligned}
\tag{2.11}
$$

2.1.5 Reference frames

Many reference systems can be used to specify the attitude of a spacecraft [39], [40]. The following systems are the most common but not the only ones that exist. The choice of the reference system depends on the pointing direction in which we are interested.

Earth-Centered Inertial (ECI)

The origin of this system is at the center of the Earth, and the fundamental plane is the Earth's equator. The $\hat{X}$ axis points towards the vernal equinox; the $\hat{Y}$ axis is 90° to the east in the equatorial plane; the $\hat{K}$ axis extends through the North Pole. This coordinate system is not rotating; it is assumed to be fixed in space. Using this reference frame, it is possible to define the position of a star in the *celestial sphere* using the coordinates *right ascension* (α) and *declination* (δ).

Earth-Centered, Earth-Fixed (ECEF)

If we allow that ECI rotates alongside the Earth, the result will be the Earth-Centered, Earth-Fixed coordinate system. The primary axis is always aligned with the Greenwich meridian. This system is especially useful in processing satellite observations from a site.

Perifocal Coordinate System (PQW)

In this system, the fundamental plane is the satellite's orbit, and the origin is at the center of the Earth. The **P** axis points towards perigee and the **Q** axis is 90° from the **P** axis in the direction of satellite motion. The **W** axis is then normal to the orbit. This system does not

rotate with the satellite, it always maintains an orientation towards perigee and is, therefore, best suited for orbits with a well-defined eccentricity.

Satellite Coordinate System (YPR)

This system moves with the satellite, and is sometimes called the *Gaussian coordinate system*. The YPR acronym comes from the *yaw*, *pitch*, and *roll* triad, previously mentioned. In this system, the *yaw* axis is directed toward the nadir (i.e., toward the center of the Earth), the *pitch* axis is directed toward the negative orbit normal, and the *roll* axis is perpendicular to the other two such that unit vectors along the three axes fulfill the relationship $\hat{R} = \hat{P} \times \hat{Y}$.

Body frame

The body frame is the frame attached to the spacecraft. Each satellite defines its frame, but usually, the base vectors of this frame are aligned with the body's principal axes. The center of this frame is at the center of mass of the satellite.

Camera frame

This frame is useful for STT applications. The camera of the STT defines this frame, in which, the focal plane defines the X and Y axis, and the Z axis points out in the boresight direction.

2.1.6 Transformations between frames

To transform between different reference frames, in terms of the elemental rotations described in 2.1.4, the following expressions can be used:

ECI to PQW: $R_{ECI}^{PQW} = R_z(\omega)R_x(\mathrm{i})R_z(\Omega)$

PQW to YPR: $R_{PQW}^{YPR} = R_x(-\pi/2)R_z(\nu + \pi/2)$

YPR to BF: $R_{YPR}^{BF} = R_x(\phi)R_y(\theta)R_z(\psi)$

where the orbital parameters used in these equations are:

ω Argument of perigee

i Inclination

Ω Right Ascension of Ascending Node (RAAN)

ν True anomaly

2.1.7 Attitude from the STT to other representations

If we get the attitude of the spacecraft by an STT, this can be expressed in different reference frames, depending on the extra information we have. For example, using an STT, it is necessary to perform certain mathematical operations to get the attitude with respect to the YPR frame. The STT gives the attitude of the camera frame with respect to the ECI frame, that is:

$$R_{ECI}^{CF} = R_{ECI}^{PQW} R_{PQW}^{YPR} R_{YPR}^{BF} R_{BF}^{CF} \tag{2.12}$$

where R represents a mathematical rotation operator, it can be a rotation matrix or a quaternion. To get the attitude of the satellite with respect to the YPR frame, the following operation needs to be done:

$$R_{YPR}^{BF} = {R_{PQW}^{YPR}}^{T} {R_{ECI}^{PQW}}^{T} R_{ECI}^{CF} {R_{BF}^{CF}}^{T} \tag{2.13}$$

It is necessary to perform this multiplication fast enough to obtain the attitude in real-time. Usually, the mathematics tool used is the quaternions, being the most convenient tool to solve these operations.

2.2 Algorithms for star identification

There are several algorithms for star identification to implement in an STT application [41], [42]. Based on Zhang (2011) [43], algorithms can be grouped into two main categories:

Subgraph isomorphism algorithms:
These kinds of algorithms use stars and the angular distance between them to form simple geometric figures (as triangles, quadrangles), considering the star image as a subgraph of the full-sky. Triangle [1] and pyramidal [44] algorithms are examples of these kind of algorithms.

Star pattern recognition algorithms:
These types of algorithms individualize each star and assign it a pattern, usually formed by the geometric distribution of other stars within a certain neighborhood. The use of statistical features of stars [45] and grid algorithms [46] are examples of these kinds of algorithms.

Other algorithms:
There are algorithms using approaches that do not fit in the previous classifications. This can be algorithms based on neural networks [47] or genetic algorithms [48], to mention a few.

In this dissertation, a new LIS algorithm is presented, based on the well-known open-

source astronomy software, *Source Extractor* and *Match*. The specific algorithm with which *Match* operates is based on [49]. In this newly developed algorithm, the stellar catalog is split into different segments to compare with the image. The developed algorithm is detailed in section 3.3.

2.3 Using *a priori* information

The use of *a priori* information when we determine the attitude of the satellite could lead to improve the accuracy of the determination and reduce the time of computation. First, I will present examples of using the TLE, and second, examples of using the Kalman filter.

2.3.1 Using TLE information

Almost every satellite flying in space is tracked in some manner. This tracking can identify the orbit of the satellite, and hence build a mathematical model of the satellite motion. The satellite data is expressed in a particular way named two-line elements set (TLE) [50]. The satellite position predicted from the TLE can be used when determining the satellite attitude. For example, if we know the current position of the satellite, one hemisphere can be discarded in the catalog searching since the satellite flies near to the Earth ($\approx$ 500 km). Reducing the catalog matching process to seek only in one hemisphere will reduce the overall process time. Moreover, If we can have some information about the attitude of the satellite for other means (for example, when we establish communication with the ground station), this info can also be entered in the STT algorithm to reduce even more the catalog searching.

2.3.2 Using Kalman filtering

Another way to use *a priori* information in attitude determination is by using a Kalman filter [51]. The use of a Kalman filter for attitude determination can be traced back to the origins of the space race. Early published research dates back to the 70s [52]. Kalman filter gives the attitude estimation, using data from only one sensor or fusing data between many of them. Among the many different applications that the Kalman filter may have in attitude determination, it can be mentioned the following applications:

- Estimation of angular rates of a spacecraft using STT [53].
- Calibration of STT [54].
- Integrate INS with STT and horizon sensors [55].
- Predicting star positions in STT [56].
- Using low-cost sensors [57].

Also, different kinds of filters can be used in attitude determination. It can be mentioned:

- Multiple models for precision pointing [58].

- Unscented Kalman filter for tracking under dynamic conditions [59].
- Sequential particle filter [60].
- Kalman filter with neural networks [61].

2.3.3 Kalman filter description for attitude determination

Following [51], a Kalman filter can be constructed considering the measurements taken by an absolute attitude determination sensor (like a star tracker) and a gyroscope. Usually, the star tracker has a low measurement rate, but the gyroscope has a high measurement rate, which makes them complementary. The state vector to be estimated is the quaternion $\mathbf{q}$ representing the satellite's attitude: $\mathbf{q} = [q_w \ q_x \ q_y \ q_z]^T$. The model for propagation and measurement of the quaternion is given by:

$$\mathbf{q}_K = \mathrm{F} * \mathbf{q}_{K-1} + \mathrm{W}_K \tag{2.14}$$
$$z_K = \mathrm{H} * \mathbf{q}_K + \mathrm{V}_K \tag{2.15}$$

Prediction

To find the transition matrix F, it is possible to use a polynomial linearization method from Taylor series of $\mathbf{q}(t + \Delta t)$ around the time t:

$$\mathbf{q}(t + \Delta t) = \mathbf{q}(t) + \dot{\mathbf{q}}(t) \cdot \Delta t + \frac{1}{2!} \cdot \ddot{\mathbf{q}}(t) \cdot \Delta t^2 + \cdots \tag{2.16}$$

with the aid of the equation for quaternion derivative (2.4), it turns to:

$$\mathbf{q}(t + \Delta t) = \mathbf{q}(t) + \frac{1}{2} \cdot \Omega \cdot \mathbf{q}(t) \cdot \Delta t + \frac{1}{2}\left[\frac{1}{2} \cdot \dot{\Omega} + \frac{1}{4} \cdot \Omega^2\right]\mathbf{q}(t) \cdot \Delta t^2 + \cdots \tag{2.17}$$

rearranging terms result in:

$$\mathbf{q}(t+\Delta t) = \left[I_4 + \frac{1}{2} \cdot \Omega \cdot \Delta t + \frac{1}{2}\Big(\frac{1}{2} \cdot \Omega \cdot \Delta t\Big)^2 + \cdots\right] \cdot \mathbf{q}(t) + \frac{1}{4} \cdot \dot{\Omega} \cdot \Delta t^2 \cdot \mathbf{q}(t) + \cdots \tag{2.18}$$

If the time interval $\Delta t \to 0$, and the angular rate is constant over the period $[t, t + \Delta t]$, it is possible to use a linear approximation obtaining:

$$\mathbf{q}(t + \Delta t) \approx \left[I_4 + \frac{1}{2} \cdot \Omega \cdot \Delta t\right] \cdot \mathbf{q}(t) \tag{2.19}$$

Assuming that we have a discrete system with definite time intervals, it is possible to use the following notation:

$$\begin{aligned} \mathbf{q}_{K/K-1} &= \left[I_4 + \frac{1}{2} \cdot \Omega \cdot \Delta t\right] \cdot \mathbf{q}_{K-1/K-1} \\ &= \mathrm{F} \cdot \mathbf{q}_{K-1/K-1} \end{aligned} \tag{2.20}$$

where we can obtain the transition matrix F as:

$$\mathrm{F} = I_4 + \frac{\Delta t}{2} \cdot \Omega \tag{2.21}$$

Thus, the propagation of the covariance matrix is:

$$\mathrm{P}_{K/K-1} = \mathrm{F}\,\mathrm{P}_{K-1/K-1}\,\mathrm{F}^T + \mathrm{Q}_W \tag{2.22}$$

Update

If the star tracker can provide the quaternion directly, then H = 1. Therefore, the equation for the Kalman gain is:

$$\mathrm{K}_K = \mathrm{P}_{K/K-1}\left[\mathrm{P}_{K/K-1} + \mathrm{Q}_V\right]^{-1} \tag{2.23}$$

The equations for the quaternion and the covariance matrix are:

$$\mathbf{q}_{K/K} = \mathbf{q}_{K/K-1} + \mathrm{K}_K\left[z_K - \mathbf{q}_{K/K-1}\right] \tag{2.24}$$

$$\mathrm{P}_{K/K} = \left[I_4 - \mathrm{K}_K\right]\mathrm{P}_{K/K-1} \tag{2.25}$$

It is important to recall that the quaternion must be normalized every time it is propagated to represent attitude properly.

2.4 Software and hardware tools for STT development

2.4.1 Software

There exist different software options currently available, ready to use and tested, to develop an STT application. The following list presents some of them:

Astrometry.net [62]

This application return astrometric calibration meta-data, plus lists of known objects falling inside the field of view, from an input image. It can be accessed through the web page [63], the *flickr astrometry group*, or can be downloaded to use it in a PC.

OpenStarTracker [64]

OpenStarTracker is a project to make a robust, high-performance star tracker freely available under the MIT license. This open work is based on [65].

NASA - Low-cost STT [66]

It is an offer from NASA's Technology Transfer Program to license a technology based on COTS components. This public offering promises: free software for download and use, extreme-low-cost hardware, and high precision attitude data (10 arcseconds precision).

Regarding the open-software package used in this work in particular, these are suites commonly used for astronomy students and researchers, *Source Extractor* [67] and *Match* [68]. These two open software suites are powerful, optimized algorithms for star detection and identification and can be implemented on diverse hardware platforms.

Source Extractor

Software used to detect and extract astronomical points of interest (or sources) from an image (e.g., stars, planets, galaxies). This work uses *Source Extractor* to detect stars and calculate their position and brightness magnitude. It generates a list of sources with their brightness relative to the image's background. Following Bertin (1996) [67], a complete analysis of an image is done in six steps:

(a) Estimation of the sky background: As each pixel value is the sum of a background signal and light coming from the objects of interest, a precise estimation of the background level is required to measure fluxes accurately. First, an estimator of the local background is computed in each mesh of a grid that covers the whole frame. The width of the grid can vary from 32 to 128 pixels. Then, the local background histogram is clipped iteratively until convergence at $\pm 3\sigma$ around its median. The mode is estimated from the median and the mean of the local background. The resulting background map is a bilinear interpolation between the meshes of the grid.

(b) Thresholding: For this step, two classical detection techniques can be considered: peak finding and thresholding. *Source Extractor* uses the last one. It extracts 8-connected contiguous pixels from a "template frame," which results from the convolution of the original image with some appropriate convolution mask.

(c) Deblending: This step is necessary to separate neighbors extracted as a single source. Each extracted set of connected pixels is re-thresholded at 30 levels exponentially spaced between its primary extraction threshold and peak value, creating a model of the light distribution stored in a tree structure. The algorithm goes from the tips of branches to the trunk and decides at each junction whether it shall extract two (or more) objects or continue its way down.

(d) Filtering: To remove spurious detections from objects with shallow profiles, *Source Extractor* verifies whether or not there could have been a detection if there were no neighbors. It computes the contribution to the mean surface brightness of each object from the wings of its neighbor, assuming a gaussian extrapolation of their profiles, and subtracts this from the mean surface brightness.

(e) Photometry: For each detected object, two types of total magnitude are computed, one uses an adaptive aperture, and the other uses an isophotal correction. The adaptive aperture can be computed as the first moment inside an elliptic aperture with ellipticity ε and position angle θ. The corrected isophotal magnitude estimates the fraction η of the total flux enclosed within a particular isophote. For non-crowed fields, *Source Extractor* estimates the total magnitude mainly using the adaptive aperture method.

(f) Star/galaxy separation: Good discrimination between stars and galaxies is essential for extragalactic studies. Nevertheless, the reduced exposure times in the proposed STT hardware do not make it possible to observe galaxies. Therefore, this step is not used in the presented LIS algorithm.

Version **2.19.5** of *Source Extractor* is used in this work. It can be downloaded for free from its website. Further information can be found in [69].

Match

Software used to establish a relationship between two different lists of objects. To operate appropriately, *Match* requires two input files with data in columns. Each file should contain at least the following columns:

1. X position of objects.
2. Y position of objects.
3. Magnitude at point (X, Y). It represents the brightness of an object relative to the background of the image.

The first file corresponds to the list of sources extracted from the image. The second file is a specific segment of the projected star catalog. In this algorithm, *Match* is used to find a relationship between the source objects (image) and the objects within the catalog. *Match* can establish either a linear, quadratic or cubic relationship between the two lists. To use the linear relationship, the star catalog is divided into segments where each segment of the catalog is linearized by using a projection in the tangent plane. This procedure is thoroughly explained in subsection 3.2. The linear relationship between the two data lists is defined by six coefficients (a, b, c, d, e, and f) delivered by *Match*. The relationship can be written as:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a \\ \mathrm{d} \end{pmatrix} + \begin{pmatrix} b & c \\ \mathrm{e} & f \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.26}$$

where the (x, y) coordinates represent the position of an object (star) in the source list (image) and point (x', y') represents the coordinates of the same object but in the segment of the projected catalog. This relationship allows assigning a sky coordinate to every image pixel.

Additionally, *Match* delivers several statistical parameters that define the precision of the established relationship. The most significant statistical parameters are:

sig (σ): It is the standard deviation of differences between the matched pairs of items, in units of the coordinate system B (B refers to the coordinate system of the projected catalog).

Nr: It is the number of matched pairs of objects (stars) used to define the transformation between lists.

Match can be downloaded at no cost from its website, where it is possible to find a user

manual. In this work, version **0.14** is used.

As previously said, *Match* is the implementation of the algorithm proposed by Valdes et al. [49]. This algorithm mimics what an astronomer visually does when matching sky images, namely, looking for similar triangles among the brighter stars. To do this, it starts selecting a subset of the brightest objects N_{obj}, both from the image and projected catalog, and forms all the possible triangles using the selected objects as vertices of triangles. After measure the length of each side of every triangle, the triangles formed from the two lists of selected objects are represented as points in a two-dimensional space called *triangle space* (x_t, y_t), defined as:

$$x_t = b/a,\ y_t = c/a, \tag{2.27}$$

where a, b, and c are the lengths of the triangle sides in decreasing order. In the *triangle space*, a triplet of common objects in the two catalogs will appear as two nearby points, and triangles are matched when they are within a specific distance ε of each other, as is shown in Figure 2.3.

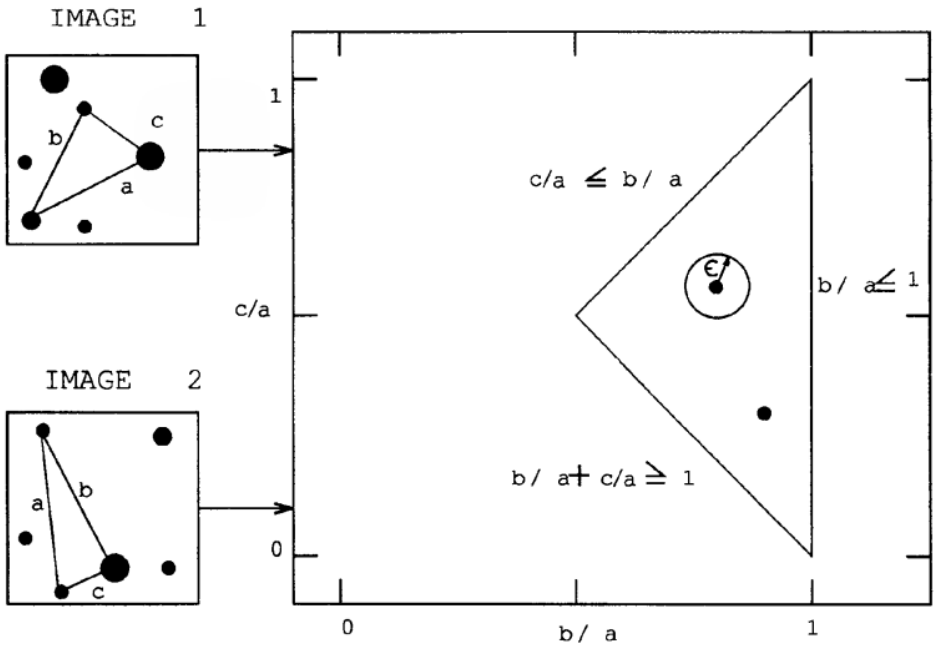


Figure 2.3: The figure (from [49]) shows the *triangle space* defined in the *Match* algorithm. The N_{obj} brightest objects coming from the image and the projected catalog are used to form triangles. After that, a two-dimensional space can be constructed considering the lengths of the triangle sides (a, b, and c), forming the ratios $x_t = b/a$ and $y_t = c/a$. It is possible to match triangles when they are within a distance ε of each other.

Next, a vote array of dimension $N_{obj} \times N_{obj}$ is defined to count the number of objects in common between lists. For each triangle matched, each of the three objects in the

triangle casts a vote. Then, a fraction of the maximum number of votes is selected as a threshold to count for matched objects. In the last step, a few matched objects are used to establish the coordinate transformation between picture and catalog objects (equation 2.26). An iterative sigma clipping algorithm is applied to reject the pairs with high residuals, setting the clipping factor as twice the 60th percentile point in a sorted residual list. The transformation coefficients (from a to f) are determined by least-squares, mainly using the HFTI algorithm from [70].

2.4.2 Hardware

It is not an easy task to select a specific hardware to develop an STT project. Most of the applications use specifically designed and manufactured devices. Many research presents and shares different cameras or algorithms but, except for new works like [71], there is a lack of information about the parts that make up a full platform in which an STT could be use or tested. Nevertheless, part of this work is to present innovative proposals. Table 2.1 presents collected information about different COTS boards that can be tested as STT hardware platforms. Table 2.2 present info about COTS cameras. The Lumenera Lw230 [72] is the camera proposed for NASA in their Technology Transfer Program [66] (previously mentioned). Both cameras and boards can be tested to form possible STT platforms.

Element	**RPi Zero W**	**RPi Zero 2 W**	**NanoPi Neo Air**	**RPi 3B+**	**ASUS Tinker board S**
CPU	1 GHz single-core ARM1176JZF-S	1 GHz quad-core 64-bit Arm Cortex-A53	Quad-core Cortex-A7 Up to 1.2 GHz	1.4 GHz 64-bit quad-core ARMv8	Quad-core 1.8 GHz ARM Cortex-A17
Memory	512 MB SDRAM	512MB SDRAM	512 MB DDR3	1 GB SDRAM	2 GB DDR3
Power consumption (W)	< 1	1	2	2.5	4
Weight (g)	9	11	10	45	55
Average cost (USD)	30	75	30	45	90

Table 2.1: Technical specifications for COTS boards on which STT applications can be tested.

Element	**RPi camera V2.1**	**RPi High Quality camera**	**RPi Global Shutter Camera**	**Lumenera Lw230M**
Sensor type	SONY IMX219PQ Color CMOS 8MPix	Sony IMX477R 12.3MPix	Sony IMX296LQR-C 1.58MPix	Sony ICX274 CCD, 1/1.8" format
Sensor size	3.674 mm x 2.760 mm	7.9 mm sensor diagonal, 1.55μm x 1.55μm pixel size	6.3mm sensor diagonal, 3.45μm $\times$ 3.45μm pixel size	1616 x 1216, 4.40μm square pixels
Full FOV (°)	62.2 x 48.8	N/A	N/A	N/A
Weight (g)	3	40	41	300
Power consumption (W)	0.5	N/F	N/F	4
Average cost (USD)	35	50	50	1950

N/A: Not applicable, lens sold separately.

N/F: Information not found.

Table 2.2: Technical specifications of COTS cameras that can be used in STT applications.

Regarding the RPi camera, there are commercial products [73]–[75] and other articles developed by makers [76], intended to spread out the RPi camera functionality. Also, two new RPi cameras have been launched; a high-quality camera in April 2020 [77] and a global shutter camera in March 2023 [78].

2.5 Developing Star Trackers for CubeSats

Remarkable examples of developing a functional satellite under strict limitations of volume, energy, and processing power are the can-sat project from the University of Tokyo [79] and the phone-sat project from NASA [80]. These kinds of projects were consolidated and aroused interest in CubeSats. For that reason, it has become relevant to increase the capabilities of the subsystems that form any CubeSat while keeping costs low, and the STT is no exception. Many studies have been developed to adapt Star Trackers to CubeSat restrictions [27]–[29], [71], [81], even testing the use of smartphones as a platform [82], [83].

Concerning the current knowledge of different sensors and actuators for small spacecraft, a full state-of-the-art report can be found in [84]. Regarding the STT performance, this report shows 25 arcseconds of pointing knowledge, with a technology readiness level (TRL) status of 9.

2.6 Use of COTS in space projects

2.6.1 Use of COTS for STT development

The use of different COTS in the development of STT is a new research trend. Many researchers are using and testing the capabilities of COTS to work as STT or other attitude sensors. The following developments can be mentioned:

- A large field of view (FOV) camera for STT [85].
- A short-wave infrared (SWIR) camera for STT [86].
- A low-cost attitude sensor with MEMS-based gyroscope and an APS CMOS image sensor [87]. However, they do not solve the LIS problem.
- An experiment with a GoPro Hero 4 camera as STT, as a backup to they MAI-400 main ADCS. They also pretend to use RPi to test SD flash memory cards [88].
- Lastly, as already mentioned, NASA is researching and evaluating the use of COTS for STT [25].

2.6.2 Using Raspberry Pi

The use of the RPi as an attitude sensor is not a new idea [89], and this component also has been used in a variety of space projects, some of this published in journals [90], [91] and others projects developed by students [92], [93]. On September 5, 2019, the news told us that an RPi Zero had been tested in space, taking a video of Earth [94], in an experiment carried out by SSTL.

2.7 Considerations about space environment

2.7.1 Environmental threats in LEO

To secure a successful mission, it is important to be aware of the main difficulties we need to cope with in the space environment. These difficulties are related to the harm that the hardware may suffer when exposed to the hostile environment that surrounds it.

This STT is designed to operate at low-earth orbit (LEO). LEO can be defined as the region that extends from 160 km above the Earth's surface up to 1000 km [95]. The use and study of this STT beyond the LEO region is out of the scope of this Thesis. According to NASA [96], the main environmental threats that degrade many materials and components in LEO can be:

Vacuum

Space vacuum (10^{-6} to 10^{-9} torr) can cause the release of volatiles from materials, a phenomenon know as *outgassing*. The outgassed molecules can deposit on cold surfaces, contaminating the spacecraft and the payloads within it. This is particularly of special care when using sensitive optics.

Atomic Oxygen

When ultraviolet radiation reacts with molecular oxygen in the upper atmosphere, atomic oxygen (AO) is produced. AO oxidizes many metals, especially silver, copper, and osmium. AO reacts strongly with any material containing carbon, nitrogen, sulfur, or hydrogen bonds, meaning that many polymers react and erode. The AO exposure will depend not only on the orientation and spacecraft altitude but also on the solar activity at the time of flight. There can be one order of magnitude difference in AO flux between solar maximum (higher flux) and solar minimum (lower flux) and significant variation between solar cycles [97].

Ultraviolet radiation

While AO may bleach materials, ultraviolet (UV) radiation generally darkens them, particularly in the presence of contamination. UV radiation also damages polymers. The UV radiation under a high vacuum can also create oxygen vacancies in oxides, leading to significant color changes.

Particulate or Ionizing Radiation

The three main sources of charged particle radiation naturally occurring in space are galactic cosmic rays, solar proton events, and the trapped radiation belts. Most of the particulate radiation dose over spacecraft in LEO occurs when flying through the South Atlantic Anomaly (SAA), depicted in Figure 2.4. It is in the form of electrons, although proton dose can be significant for some materials and components. Depending on the polymer, particulate radiation can result in cross-linking or chain scission, similar to damage by UV, resulting in polymer embrittlement. A significant effect is seen in avionics, namely single-event upsets, bit errors, and latch-ups.

Plasma

An ionized gas composed of equal numbers of positively and negatively charged particles is termed plasma. Space plasma is similar but separate from the higher energy particulate radiation. The plasma environment for spacecraft in LEO orbits is composed of approximately equal amounts of positively charged oxygen ions (O+) and free electrons varying with solar activity and altitude. Because of the differences in spacecraft velocities, ion thermal energy, and electron thermal energy, electrons can impact any spacecraft surface, while ions can only impact ram surfaces. This disparity can lead to a negative charge buildup, which can lead to ion sputtering, arcing, and parasitic currents in solar arrays, as well as re-attraction of contamination [98].

Thermal Cycling

As the spacecraft moves in and out of sunlight during its orbit around Earth, the degree to which a material experience thermal cycling temperature extremes depends mainly on its thermo-optical properties, its view of the sun/Earth, and durations of time in sunlight and in shadow. A rule of thumb for these cyclic temperature variations is -120°C to +120°C, but high solar absorptance with low infrared emittance will increase temperature swings. A spacecraft in LEO orbit can make sixteen thermal cycles a day, leading to cracking, peeling, spalling, or formation of pinholes in the coating, which then allows AO to attack the underlying material.

Micrometeoroid/Orbital Debris Impact

All areas of a spacecraft can be struck by micrometeoroids traveling as fast as 60 km/s. Surfaces facing the ram direction are more likely than those in the wake direction to be hit with space debris, traveling at an average velocity of 10 km/s. Space debris varies with the solar cycle: as the Sun's activity increases, the atmosphere heats up, increasing the drag on space debris in orbit. Micrometeoroid or space debris impacts may crater the material, spall off a coating, or short out a solar cell.

While all the environmental threats previously mentioned could be present on any spacecraft flying in LEO, I will delve into the topics of interest in this thesis: space radiation effects over COTS devices and digital cameras.

2.7.2 Effects produced by space radiation

On May 18, 2003, a disturbing anomaly occurred in the electronic voting process in Schaerbeek, Belgium, in which one of the candidates ended up with 4096 extra votes. After an extensive investigation, a possible explanation for this anomaly was found: cosmic radiation's effect on Earth, specifically, a single event upset [100]. This event reminds us how vulnerable computers can be to the effects of radiation. In the same way, not considering the impact of radiation on avionics inside a satellite can be disastrous for the mission's objectives. These radiation effects can be present in different forms [101]–[103], namely:

Single Events Effects

As its name suggests, these events are caused by a single, energetic particle and can take many forms. It is possible to distinguish at least two different effects:

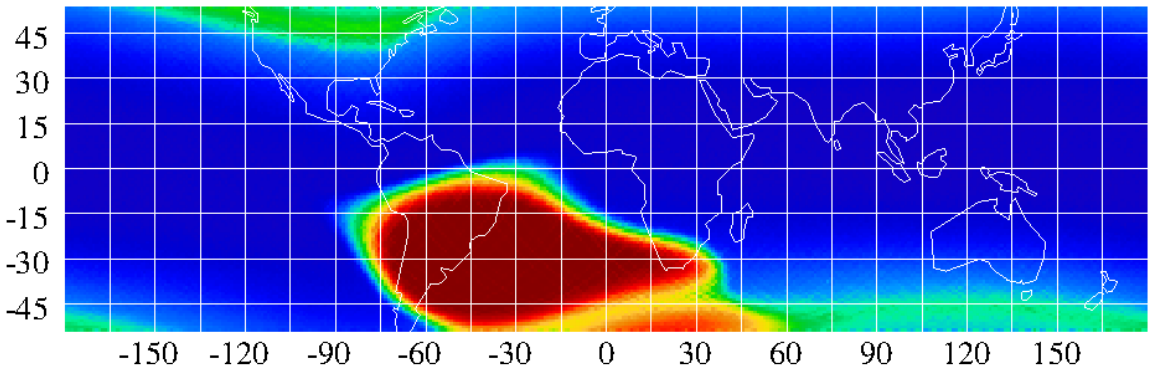


Figure 2.4: The figure shows a map of charged particles. The large red area shows the South Atlantic Anomaly (SAA). SAA is a dip in the Earth's magnetic field, allowing cosmic rays and charged particles to reach lower into the atmosphere [99].

Single Event Upsets

These are non-destructive, "soft" bit errors in that a reset or rewriting of the device causes normal behavior thereafter. They usually appear as transient pulses in logic or support circuitry or as bitflips in memory cells or registers. This may occur in digital, analog, and optical components or may have effects in surrounding circuitry.

Single Event Latchup

These errors can be potentially destructive. These effects result in a high operating current, above device specifications, and removal of power to the device is required in all non-catastrophic conditions to recover device operations.

Total Ionizing Dose

The Total Ionizing Dose (TID) measures the total energy absorbed by matter. Defines the cumulative effect that occurs through continuous radiation exposure, leading to gradual performance deterioration of the hardware that potentially leads to failure. The main sources of these particles are Solar Energetic Particle Events and the SAA.

Displacement Damage

It is the effect produced when charged particles interact with the nucleus of a target atom. A fraction of the energy of the incoming particle is transferred to the target nucleus, which is excited and can be displaced. The consequence is a local modification of the lattice structure and the creation of crystalline defects and interstitial-vacancies complexes. The accumulation of these defects, or clusters of defects, modifies the crystalline and electronic properties of the crystal. The effects range from changes in optical and mechanical properties to changes in the electronic structure of the device.

Regarding the effects of radiation over different COTS components for SmallSats, stud-

ies claim that a solution can probably be achieved with commercial parts if the lifetime radiation dose is under 30 krad [104]. Concerning the particular hardware studied in this dissertation, the RPi Zero, specific tests exist over it that analyze the performance under radiation, simulating the TID of LEO (0.4 krad/hr) [105]. Results show that the RPi has been fully operational during the test period, indicating that this hardware is an excellent candidate to work with it in space.

Another relevant and well-documented phenomenon related to high-energy particles in the space radiation environment is the phosphene or "light flashes" for astronauts in LEO. Astronauts perceived these white flashes before sleep, and several people even thought the light flashes disturbed their sleep [106]. However, apart from affecting the vision of astronauts in LEO, many different kinds of electrical equipment and sensors can be affected by radiation. For instance, radiation damages the CCD detectors of the Hubble Space Telescope (HST), generating hot pixels. Hot pixels are in general defined as those pixels having a dark rate greater than a given threshold. This failure is later corrected through the process of annealing, raising the CCD temperature to about 20°C for a few hours. Transient hot pixels that after 6-7 anneal cycles do not recover their normal status remain permanently hot [107].

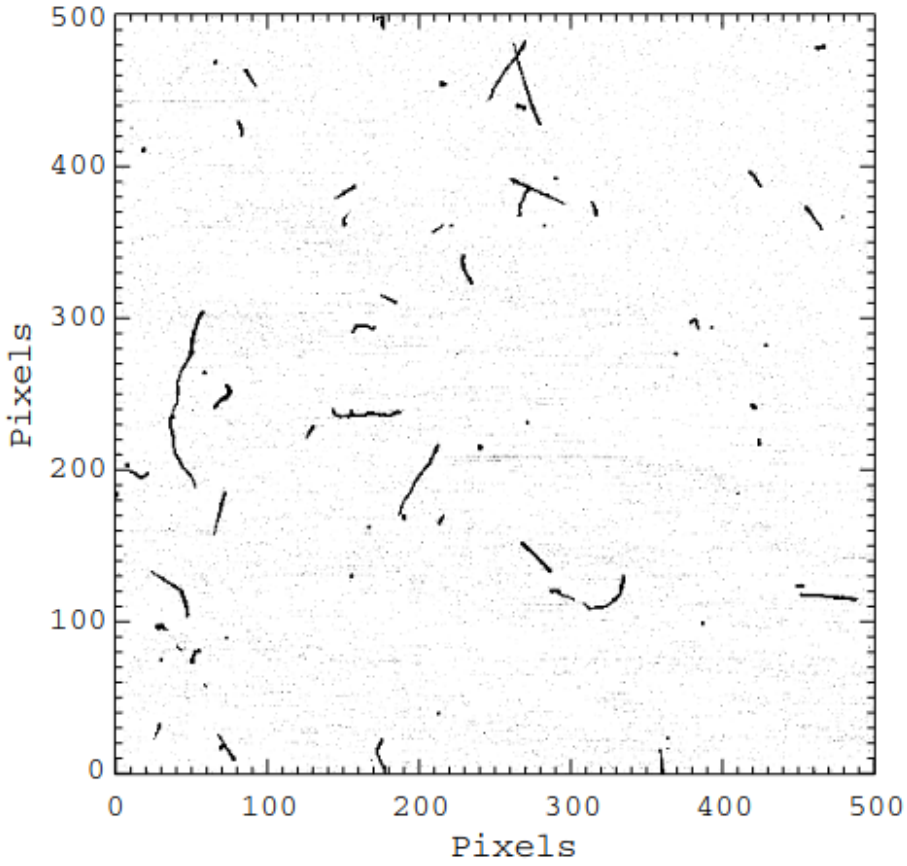


Figure 2.5: The figure, taken from [112], shows a subfield of 500×500 pixels of a 4000 s dark exposure, in which the radiation effects are visible: straight tracks, "worms", and spots.

CCDs from the HST are not the only electro-optical devices affected by radiation, as would be expected. The image quality of high-definition television cameras CCDs of the

International Space Station (ISS) is degraded by the presence of permanent bright pixels due to space radiation [108]. An alternative to CCD detector, CMOS, has also been tested under irradiation [109]. In a general way, literature stated that defects are produced over different kinds of imagers exposed to radiation produced in LEO [110], [111]. As shown in Figure 2.5, these defects can take the shape of dots (either bright or dark dot defects), "worms", or even straight lines [112].

Concerning the particular research about radiation over star trackers, a study from JPL described the development of radiation tolerant STT based on a CMOS active pixel sensor [113]. They used this sensor to avoid the degradation of the charge transfer efficiency of CCDs by proton irradiation, which results in apparent shifts in the positions of imaged objects. Furthermore, a study about the STT of the microsatellite PROBA from ESA [114] emphasizes the increase in processing time due to filtering the hot spots that the radiation induces in the CCD.

2.7.3 Study and mitigation of radiation effects

Different tools exist to estimate the radiation over the spacecraft when following its orbit in space. As an example, Figure 2.6 shows a radiation analysis made using the SPENVIS web-based tool [115]. This particular analysis uses the SHIELDOSE-II tool to calculate TID in enhanced electron and proton environments as a function of shielding depth based on spherical geometry [116].

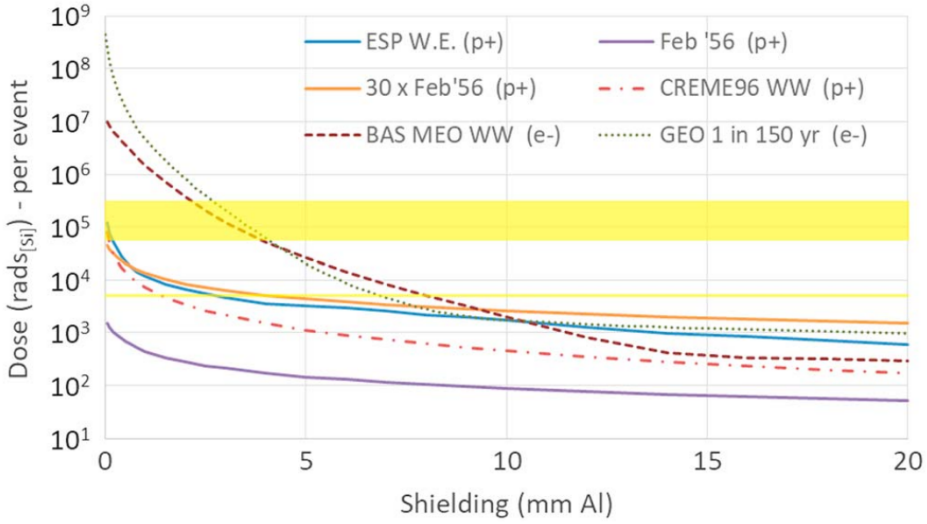


Figure 2.6: The figure, taken from [116], shows a radiation analysis made using the SPENVIS web-based tool. Dose-depth curves for four solar energetic particle event scenarios are compared to equivalent curves for two electron enhancement scenarios. Each curve is based on spherical shielding geometry with proton- and electron-based events labeled (p+) and (e-), respectively.

Different works show how to cope with the effects of the radiation environment over the

star images taken with the STT. There are methods based on using just one image or a sequence. Regarding one-image methods, a well-known method widely used in astronomy is L.A.Cosmic [117], based on a variation of Laplacian edge detection. This algorithm can identify cosmic rays of arbitrary shapes and sizes by the sharpness of their edges. Also, it distinguishes undersampled point sources from cosmic rays. Thanks to the accelerated advancement of neural networks, a reliable and robust method has been published recently [118], which uses a deep-learning-based approach for cosmic rays rejection. They claim that after proper training, it has the potential to outperform current methods in terms of both speed and accuracy in mask prediction and image inpainting.

Regarding the use of a sequence of pictures, a method proposed by Wang (2019) [119] achieves false star filtering by utilizing the difference between the motion of false stars and true stars, performing angular distance tracking and a star voting process on multiple consecutive images of stars.

Chapter 3

Algorithm Development

3.1 Description

The developed algorithm solves the LIS problem, which makes it comparable to previously published algorithms. Also, it assumes that the spin rate of the spacecraft (at each axis) is less than a revolution per hour. A spin rate over this threshold would be detected by other attitude estimation sensors and diminished by the attitude control unit. The proposed algorithm uses both star brightness information and the segmentation of the stellar catalog. The algorithm and all the related tests were programmed using *Python* [120].

SOST starts the attitude estimation process by acquiring an image from which the brightest objects (stars) in it are extracted by using *Source Extractor*. This list of objects from the image is compared with a stellar catalog stored in memory. To find the attitude of the camera, and ultimately the attitude of the vehicle associated with it, a procedure of two parts was developed. First, it identifies the segment in the celestial catalog with the best match. The catalog has to be divided into segments of similar size to the FOV of the image to use *Match* rightly. Secondly, it refines the match between the image and the catalog segment by moving the projection point of the catalog segment in the tangent plane. The projection of the catalog is necessary to perform the match with the image, which is a projection of a segment of the sky. This procedure is also called camera calibration.

3.2 Tangent plane projection of the stellar catalog

The pointing precision should be determined by the images, not by the precision of the stellar catalog that is used for comparison. The catalog should cover a stellar brightness range comparable to those available in the images to maximize the number of matched stars. Among the myriad of stellar catalogs freely available that satisfy these criteria, I chose the Bright Star Catalogue (BSC) [121] from Yale University Observatory. The BSC was obtained using the *scat* application, part of the WCSTools [122] software package. The BSC catalog

contains 9,110 objects, 9096 of which are stars. It includes stars up to brightness magnitude 6.5.

When comparing a picture of the sky with the stellar catalog, it is necessary to match the geometry of both systems. The star catalog is in spherical coordinates, whereas the photo maps the tangent plane. It is required to project the stars into the tangent plane to relate both coordinate systems, in a direction close enough to the actual pointing such that the aberration in the relative positions is low and a match is possible. Following [123], this is achieved according to the equations:

$$\eta = \frac{\cos(\mathrm{DEC}) - \cot(\delta)\sin(\mathrm{DEC})\cos(\alpha - \mathrm{RA})}{\sin(\mathrm{DEC}) + \cot(\delta)\cos(\mathrm{DEC})\cos(\alpha - \mathrm{RA})}, \tag{3.1}$$

$$\xi = \frac{\cot(\delta)\sin(\alpha - \mathrm{RA})}{\sin(\mathrm{DEC}) + \cot(\delta)\cos(\mathrm{DEC})\cos(\alpha - \mathrm{RA})}, \tag{3.2}$$

where RA and DEC are the right ascension and declination of the projection point over the celestial sphere, respectively, and α, δ the coordinates of the star (S) to be projected. Thus, η and ξ will be the so-called *standard coordinates* of the star S in the projection plane.

In this implementation, I cannot use the full stellar catalog as a whole, since the image is much smaller than the whole catalog and the *Match* software requires two data lists of similar sizes to operate properly. For this reason, the catalog was stored in memory but divided into overlapping segments of $60° \times 60°$ (a slightly larger area than the FOV of the RPi V2.1 camera). Each segment is stored in spherical coordinates and also in a projected form by using the center point of the segment. In this work, different distances between the centers of each projected segment are tested in RA and DEC. Thus, the segment (i, j) is the one with center (tangent plane point) $\mathrm{RA}_{segment\,\mathrm{i},j} = \mathrm{i}°$ and $\mathrm{DEC}_{segment\,\mathrm{i},j} = j°$ with limits of the segment $\mathrm{RA}_{segment\,\mathrm{i},j} \in [(\mathrm{i} - 30)°, (\mathrm{i} + 30)°]$ and $\mathrm{DEC}_{segment\,\mathrm{i},j} \in [(j - 30)°, (j + 30)°]$, where $\mathrm{i} = 0°, \ldots, 359°$ and $j = -90°, \ldots, 90°$. It was considered three different segmentation patterns for the catalog, where i and j are advanced by 5°, 10°, and 15°. These yield different numbers of segments to project and compare.

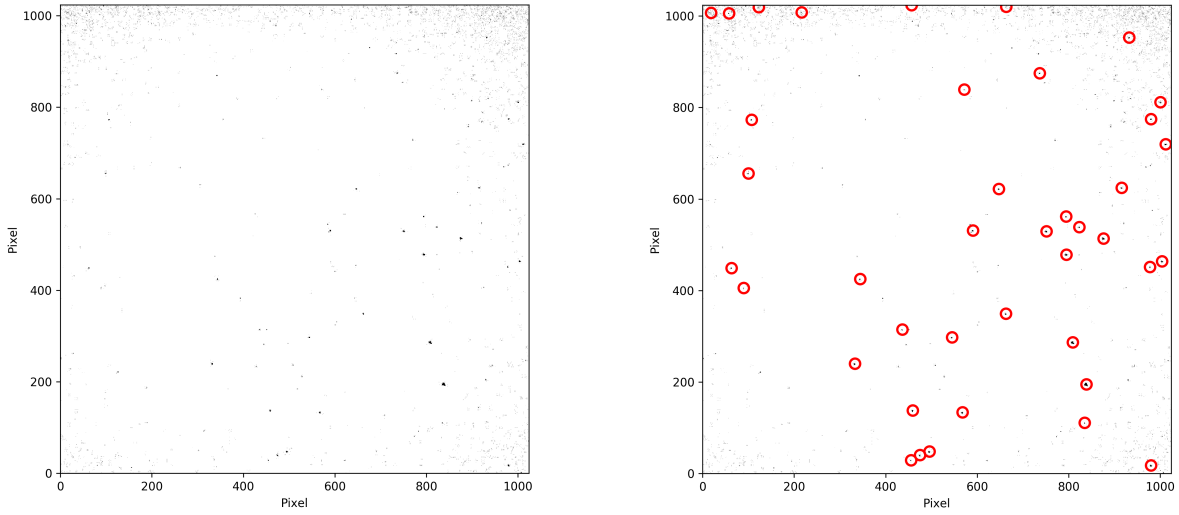
3.3 Attitude determination algorithm

The following procedure is carried out to determine the attitude of the camera:

1. **Acquire an image.** The camera, with predefined exposure time, takes a picture of the sky. The exposure time was evaluated and configured with the RPi camera, presented in section 4.2.2.
2. **Generate the list of sources from the image.** The list of brightest objects in the picture is obtained by using *Source Extractor*:
   (a) The ".jpg" image is transformed into ".fits" format, the format used by *Source Extractor*, and then entered into the software.

(b) *Source Extractor* generates a list of sources, with the position and brightness of the detected objects. The position of each detection is referenced to the lower-left corner of the image. The brightness is a value referenced to the background of each image. The forty brightest objects are selected from the full list of detected objects. This number of selected objects is studied in section 4.2.2.

(c) The image is scaled from pixels to mm, to represent the image that is formed in the CMOS of the camera and to be able to use *Match* with the linear procedure. Figure 3.1 shows the extracted sources from one of the images taken during the evaluation of the system, which will be used as an example during this algorithm description.



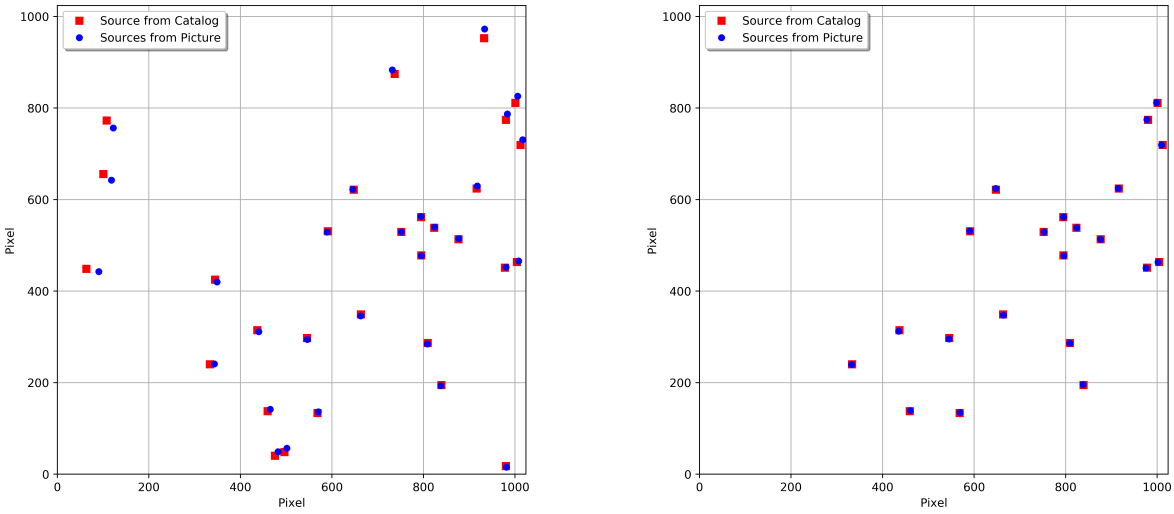
(a) Original image.

(b) Original image plus extractions.

Figure 3.1: Sample image used as an example to describe the algorithm. Figure (a) shows the negative of the original image captured by the RPi camera. Figure (b) shows the forty brightest detections extracted from the picture. Note that some detections might not be stars; these will not be matched with catalog objects.

3. **Matching: First Iteration.** Search throughout the celestial sphere. Match between the captured image and the in-memory projected segments of the catalog.
   (a) The brightest objects in the picture are searched in each segment of the in-memory projected star catalog using *Match*. The output of this stage is a list of candidates (matched) segments of the star catalog.
   (b) The segment of the star catalog with the largest number of matched objects (Nr) is selected as the best match. In tests, it is usually achieved with 18 objects (stars) or more. This selected segment is the output of the first stage of the algorithm. Further iterations are performed to improve the accuracy of the matching procedure.
   (c) The distance between the centers of the projected segments was arbitrarily selected in this stage (e.g., 5 degrees). I studied the effects of different distances between

catalog centers in subsection 4.2.3. After finding the segment of the catalog with the best match with the image, the center point of the projected catalog is de-projected (returned to sky coordinates) by using the linear model delivered by *Match* (See equation 2.26). At this stage, the projected catalog does not perfectly agree with the image. The relationship delivered by *Match* was used to find a new projection center to improve the matching procedure in the next iterations. The result of this iteration is shown in Figure 3.2 (a).



(a) Results from *Match* - First iteration.

(b) Results from *Match* - Third iteration.

Figure 3.2: Results of the algorithm. The two figures show the objects, both from the catalog and the picture of the sky, that the algorithm uses to find the relationship between both lists of objects. Figure (a) show the result of the first iteration, and Figure (b) shows the result of the third (final) iteration.

4. **Matching: Second Iteration.** The projection center of the selected segment of the catalog is corrected by using the matched stars found in the first iteration and the new center (the one de-projected using the linear model given by *Match*). The selected catalog segment is reshaped, being sure it has a size similar to the image but having as center the new center obtained with the linear model. Then, *Match* is used again with the selected catalog segment (which has been projected with the new center found in the first iteration) and the list of objects extracted from the picture. A new list of matched objects is output, and the center is again de-projected to refine for the last time.
5. **Matching: Third and Final Iteration.** Using the same approach as in the second iteration, *Match* is used again with this corrected segment of the catalog (projected with the new center found in the second iteration) and the list of extracted objects from the picture. At this point, the match between the two lists is accurate enough, and the procedure is stopped. The output of *Match* can be used together with the corrected projection center to find the RA, DEC and Roll angles relative to the center of the camera, therefore the attitude of the camera in the inertial coordinate frame. Also, the

stars used in the matching procedure are identified. The process of identifying stars and getting the attitude of the camera is simultaneously achieved by this algorithm. The result of this third iteration is shown in Figure 3.2 (b). Figure 3.3 shows comparatively the results obtained in the first and third iteration. After this stage, the effective output of the algorithm is reached. The angles RA, DEC, and Roll for the example image shown in Figure 3.1 are 192.88°, −45.10°, and −98.76°, respectively.

The process described above can be summarized in the flowchart shown in Fig. 3.4.

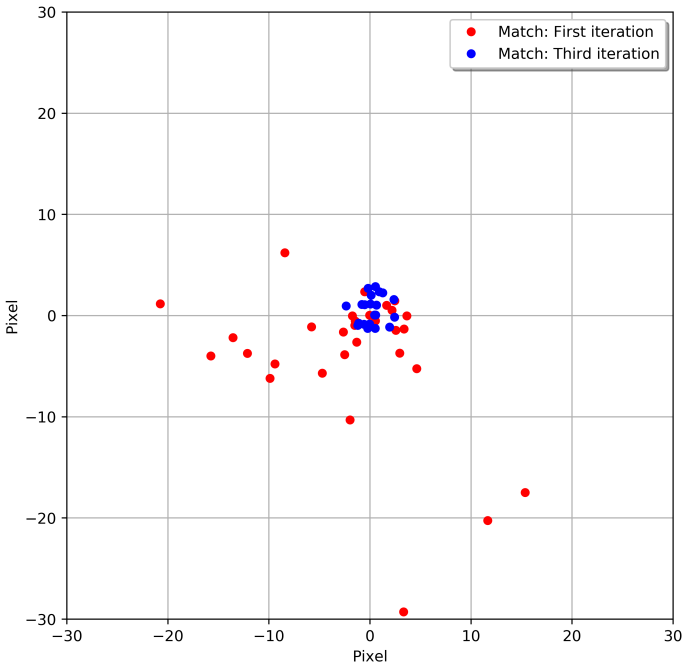


Figure 3.3: Differences (in pixels) between pairs of matched objects (picture objects with those in the catalog) found in the first and third iteration. The result of the first iteration (30 matched objects) is shown in red color and the third iteration (21 matched objects) is shown in blue color. Dispersion in the third iteration is less than that obtained in the first iteration.

3.4 Adapting the algorithm for different field of view

The algorithm previously described is intended to be used with the optical parameters of the RPi V2.1 camera. However, this algorithm can be adapted for different FOV. This algorithm has been adapted and tested for three different FOV: the RPi V2.1 camera, the HI-1 telescope from STEREO mission (described in section 4.1), and a Pixelink PL-D725 camera with a 70 mm lens. The procedure to adapt the algorithm is the following:

1. Define the cutoff magnitude of the stellar catalog.

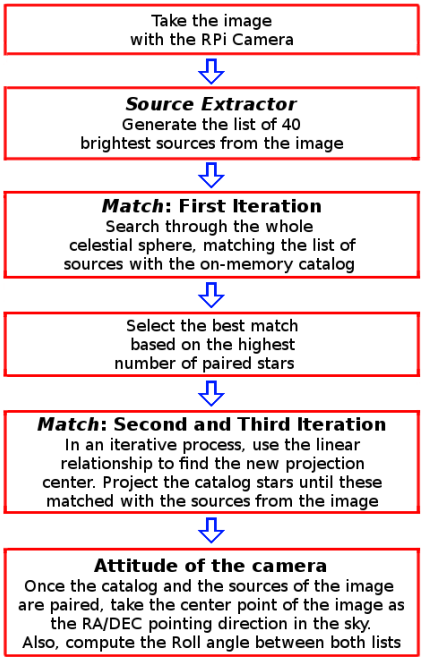


Figure 3.4: Flowchart of attitude determination algorithm (See section 3.3).

2. Define the optical parameters related to the FOV: focal length and the segment extension in RA and DEC.
3. To generate the catalog segment, select a specific (RA, DEC) center (RA can be between 0° and 360° and DEC between -90° and 90°) and then search for objects in the sky in the specific center and within the margins defined by the size of the catalog segment. Save the catalog segment in (RA, DEC) coordinates. Repeat this procedure for every integer ordinate pair (RA, DEC) in the sky.
4. Project the previously found catalog segment in the corresponding center, and save it in the *standard coordinates*. Repeat this procedure for every integer ordinate pair (RA, DEC) in the sky.

Table 3.1 shows the main parameters for catalogs adaptation for the three different cameras previously mentioned.

A novel and interesting representation arise if we look at the number of catalogs depending on the number of stars it contains. This representation is shown in Figure 3.5. In this graph, we can see the distribution of the number of stars in the catalog segments. The graph is normalized with respect to the maximum number of catalogs that contain a specific number of stars. The dashed lines show the mean number of stars computed with all the catalog segments.

This representation is useful because it graphically shows if the parameters considered to construct the catalog (the catalog segment size and the cutoff magnitude) are consistent with the number of stars expected to match between the picture and the corresponding catalog segment.

Camera	Focal length (mm)	Cutoff magnitude	FOV (°)	Segment extension (°)	Average number of stars per segment	Maximum number of stars in segment	Minimum number of stars in segment
RPi V2.1 camera	3.04	4	48.8	60	41	89	6
HI-1 telescope (STEREO mission)	78.46	6	20	20	51	144	19
Pixelink PL-D725 camera	70	6	8	12	19	70	3

Table 3.1: Main parameters for catalogs adaptation for three different cameras.

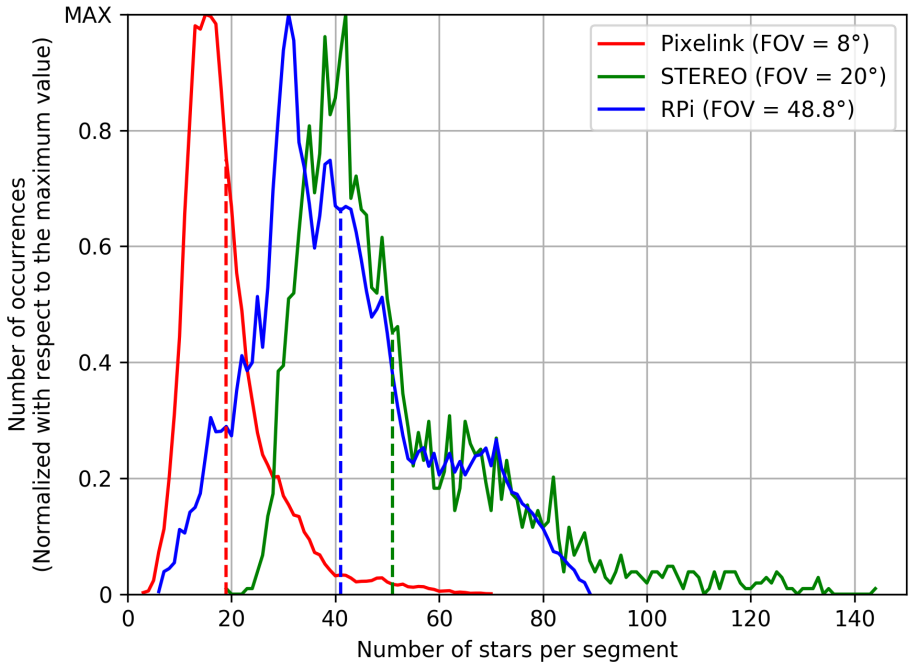


Figure 3.5: The graph shows the distribution of the number of stars in the catalog segments. The graph is normalized with respect to the maximum number of catalogs that contain a specific number of stars. The dashed lines show the mean number of stars computed with all the catalog segments for every considered camera.

Chapter 4

Evaluation

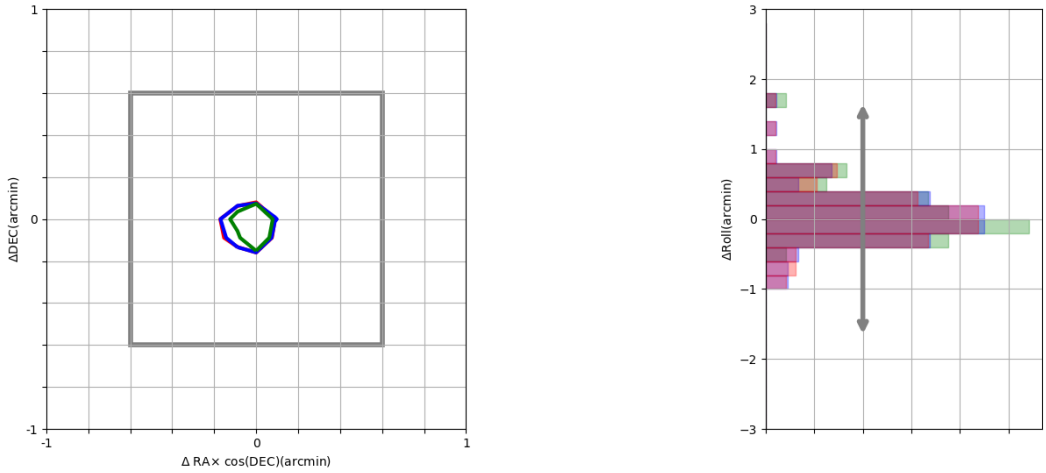
4.1 Algorithm evaluation by using on-space images

To isolate the algorithm from the platform, I will focus first on evaluating only the developed algorithm, using pictures coming from the STEREO (Solar TErrestrial RElations Observatory) [124] mission. STEREO is the third mission in NASA's Solar Terrestrial Probes program (STP). It employs two nearly identical space-based observatories - one ahead of Earth in its orbit, the other trailing behind - to provide the first-ever stereoscopic measurements to study the Sun and the nature of its coronal mass ejections (CMEs). I use images from STEREO, specifically from the HI-1 telescope [125]. This telescope observes the space between the Earth and the Sun. There are hundreds of stars in each image, making them suitable to test the algorithm.

To properly use the algorithm with HI-1 STEREO images, the procedure was the following:

1. Adapt the catalog to the FOV of the HI-1 detector ($20° \times 20°$). As previously mentioned in section 3.4, it was considered stars from the BSC catalog until magnitude six; this implies a total of 5080 catalog stars. The algorithm was tested with the center of catalog patches separated 5°, 10°, and 15°.
2. Get 100 HI-1 "L0" images from the \pub subdirectory of the STEREO SCIENCE CENTER web page [126]. Random images were selected from the year 2007 to the year 2010 from the observatory STEREO A. Attitude is computed from this HI Level 0 images using the developed algorithm. An example name of a L0 image downloaded in this stage is "20070130_080100_s4h1A.fts".
3. Get the "L2" images from the corresponding "L0" images previously obtained [127]. The "L0" and "L2" processing levels is well-explained in [128]. The attitude obtained in the previous step is compared with the attitude information from the header data of the HI Level 2 images. For the example name previously shown, the name of the corresponding L2 image is "20070130_080100_2bh1A_br01.fts".
4. Collect the results. The differences in the pointing coordinates between the developed algorithm and the header data of every image are shown in Figure 4.1, for different

catalog segmentations. The success rate given by the algorithm in this evaluation is 96%, 89%, and 48% for separation of 5°, 10°, and 15° between catalog centers, respectively.



(a) STEREO accuracy in Right Ascension and Declination coordinates. **(b)** STEREO accuracy in Roll coordinate.

Figure 4.1: Results from the pointing analysis with STEREO images. These graphs show the difference between the attitude data from the header of STEREO images and the attitude data obtained with the proposed algorithm. Figure (a) shows the pointing accuracy for Right Ascension and Declination in arcminutes. The contour lines enclose 90% of successful runs with catalog separations at 5°, 10°, and 15° shown in red, blue, and green respectively. The gray square is the angular size of a pixel in STEREO images. Figure (b) shows the pointing accuracy distribution for the Roll angle. The gray arrow is the angle subtended by a pixel over the side of an image, i.e., 1/1024 rad.

4.2 Platform evaluation

4.2.1 Description

The COTS hardware selected to work as a platform of the developed STT algorithm are the known family of small single-board computers Raspberry Pi and the RPi camera V2.1. This platform was selected mainly due to the simplicity of use, it is popular among researchers, educators, and makers, and it is ultra-low-cost. The main features of these components was shown in Tables 2.1 and 2.2, respectively.

An STT is the most precise optical attitude determination device. However, there is still debate on how to realize and verify such precision. It is hard to set up a similar to in-orbit conditions to test the performance of the STT. Some evaluation methods use artificial stars in the laboratory to solve this problem. I followed a similar approach that the suggested by Sun (2016) [129] using real astronomical sources.

To test that the results with STEREO HI-1 images can similarly be obtained with this platform, the platform's performance was tested with images captured by the RPi camera. This test consisted of capturing images of night-sky, from two different geographic locations in two observing runs:

- The first data set was obtained at 33°00′48″S 70°54′08″W on August 6th, 2016 from 22:00 to 05:00 hours, local time (CLT).
- The second data set was obtained at 34°25′42″S 70°49′22″W on February 6th, 2017 from 22:00 to 05:00 hours, local time (CLT).

Over 500 images were taken in each site to evaluate the overall performance of the SOST. In the test, the camera was pointing at different directions of the sky, with a set of different exposure times. The experimental setup used for this task is shown in Figure 4.2. The effects of the following variables were studied:

1. Exposure time.
2. Star catalog segmentation. The main parameters to study were the number of catalog segments, the size of the segments, and the distance between segment centers (segment overlapping).
3. Processing time.

It was also evaluated the performance of the platform that will host the first version of SOST:

1. Power consumption.
2. Environmental studies, mainly its response to vacuum.

4.2.2 Exposure Time

The proposed algorithm expects to match tens of stars from the picture with their catalog sky positions. The exposure time of the picture will set the number of stars that will be detected on average. I set out to find the minimum exposure time that yields enough number of extractions that match catalog stars.

Different regions of the sky were photographed, with exposure time ranging from 0.1 to 2 seconds at intervals of 0.1 seconds. The RPi V2.1 camera was set up to output 1024 pixels × 1024 pixels to deal with a square FOV with 48.8° on each side. The number of extractions increased linearly with the exposure time, up to 0.9 seconds, after which the number of real extracted sources remains constant without any significant improvement to the image quality. Although the algorithm works with images with an exposure time of 0.6 seconds, I decided to set the minimum exposure time at 0.8 seconds, since *Source Extractor* yields over 40 detections under the relatively poor conditions of ground-based observations.

Regarding the matched stars, it was empirically found that with ≈ 20 stars, a good fit for the pixel-to-sky transformation can be obtained. For this reason, the 40 brightest detections was selected in the image (see Figure 3.1). This criterion selects catalog stars brighter than

Figure 4.2: Equipment used for night sky measurements. This image shows the tripod, the black enclosure of the Raspberry Pi, and the transparent enclosure of the camera. The measurements were taken at 33°00′48″S 70°54′08″W on August 6th, 2016, and at 34°25′42″S 70°49′22″W on February 6th, 2017. Both measurements were made from 22:00 to 05:00 hours, local time (CLT).

magnitude 4, reducing the astronomical sources from the BSC catalog to 518 stars.

4.2.3 Success rate, precision, catalog segmentation and processing time

In the algorithm *Match* properly operates based on two main assumptions: (1) the size of both lists has to be comparable and (2) the relation between both lists has to be linear. These assumptions guide most of the decisions regarding the catalog segmentation in this algorithm. For instance, I could not use the catalog as a whole (as just one segment) because in this case, the picture is much smaller than the catalog segment, breaking our first assumption. On the other hand, assumption (2) requires a projection of the catalog to linearly match a set of spherical coordinates (right ascension and declination) with Cartesian coordinates (pixels).

The catalog was segmented in a set of projections to be compared with the image. The number of segments in the catalog is directly related to the success rate that can be obtained with the developed algorithm. I tested the influence of catalog segmentation on the algorithm success rate, the number of related objects (stars) between lists, the precision of the obtained

pointing solution and the processing time, in the following way:

1. Select a subset of 50 images of the total taken pictures from the night-sky measurements. In this selected subset of images, we can extract enough numbers of stars (at least 20 stars). The algorithm could be capable of finding a pointing solution in every image.
2. Run the STT algorithm over every image, with the center of the catalog patches separated by 5°, 10°, and 15°.
3. The catalog starting point in (RA, DEC) was changing in the matching process for each catalog segmentation. It starts in (0, 0) degrees and increases one by one at each coordinate, sweeping all possible integer values until $\big([p-1]^\circ,[p-1]^\circ\big)$, with $p = 5^\circ, 10^\circ,$ and 15°.
4. Save all the pointing results given by the algorithm for every image.

The success rate was computed as follows: For a given catalog segmentation, and a specific starting point of the catalog, I count the times that the algorithm gives a correct pointing solution, for the selected subset of images. After that, the success rate was averaged over different starting points, for every catalog segmentation. The result is shown in Table 4.1.

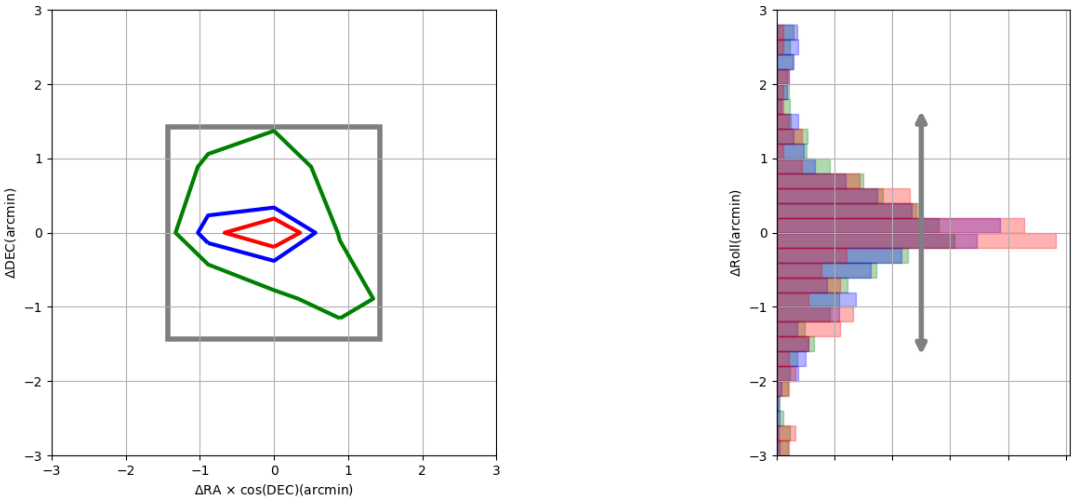
The number of related objects between lists was computed as follows: For a given catalog segmentation, and a specific starting point of the catalog, the number of matched objects between picture and catalog was saved, for the selected subset of images. After that, the number of objects over all the different starting points was averaged for every catalog segmentation. The result is shown in Table 4.1. The uncertainties show the variability of the success rate and the number of related objects when the starting point of the catalog segmentation is changed.

	Catalog segmentation		
	5°	10°	15°
Success rate (%)	99.8 ± 0.6	97.3 ± 2.5	85.6 ± 7.0
Number of related objects	18.0 ± 0.0	17.9 ± 0.3	17.7 ± 0.5

Table 4.1: Success rate and the number of related objects between lists, for different catalog segmentations.

Regarding the precision in the pointing coordinates (Right Ascension, Declination, and Roll), it is important to note that for this type of experiment, contrary to the test with STEREO images, I do not know the "true value" of the attitude solution in the selected subset of pictures. For that reason, this analysis aims to get an idea of the precision, not the accuracy, of the proposed platform. In that way, I assume that the segmentation pattern at 5° brings an attitude solution near the "true value". For that reason, I computed an assumed "true value" based on the average of the pointing solution obtained for different starting points in (RA, DEC) when the catalog segmentation pattern is 5°. Then, we compute the differences between this averaged value and the attitude solution for every pointing solution at the different starting points of the catalog and different catalog segmentations. This procedure is repeated for every picture. The result is graphically shown in Fig. 4.3. The contour lines in the graph enclose 90% of successful runs for each catalog segmentation (5°, 10°, and 15°). We can see that the mean precision of the pointing coordinates decreases with a larger catalog separation. For the Right Ascension and Declination coordinates, the mean

precision ranges from 0.5 to 1 arcminute. For the Roll coordinate, the mean precision ranges from 2 to 4 arcminutes.



(a) SOST precision in Right Ascension and Declination coordinates. **(b)** SOST precision in Roll coordinate.

Figure 4.3: Results from the pointing analysis with RPi images. Both figures represent the deviation of the different attitude solutions due to the change of the starting points of the catalog, compared with the average value. Figure (a) shows the pointing precision for Right Ascension and Declination in arcminutes. The contour lines enclose 90% of successful runs with catalog separations at 5°, 10°, and 15° shown in red, blue, and green respectively. The gray square is the angular size of a pixel in the images. Figure (b) shows the pointing precision distribution for the Roll angle. The gray arrow is the angle subtended by a pixel over the side of an image, i.e., 1/1024 rad.

Besides the catalog segmentation and the pointing precision, the processing time of an STT is a relevant variable to assess its performance. Processing time settles the type of application or science that can be achieved with the STT. Processing time depends on both the algorithm and the hardware. The algorithm was tested using the *multiprocessing* package from *Python*, taking advantage of the multi-core capability of the RPi 3 B+. This multi-processing capability was used specifically in the *Match* routine of the algorithm. Table 4.2 summarizes the time taken by each process of the attitude determination of the STT, solving the LIS problem with the RPi with separation between catalog centers of 5°, 10°, and 15°. These times were measured using the function `time.time()` from *Python*.

4.2.4 Vacuum chamber test

The RPi 3 B+ and its camera were tested inside a vacuum chamber, to partially simulate the spatial environment and its effects. The vacuum chamber system is an NDT-4000 from Nano-Master Inc [130]. When the RPi was inside the chamber, the full test consisted of performing the following tasks every five minutes:

1. Take a picture with the RPi camera and measure the execution time.

Process	Average time (s)		
	5°	10°	15°
Importing libraries		4.07 ± 0.03	
Picture acquisition		8.68 ± 0.01	
Source extraction		2.36 ± 0.04	
Catalog matching	24.6 ± 0.1	7.2 ± 0.1	3.9 ± 0.1
Total	39.7 ± 0.1	22.3 ± 0.1	19.0 ± 0.1

Table 4.2: Processing time of each major routine in Raspberry Pi 3 B+. These numbers are for separation of 5°, 10°, and 15° between centers of the catalog patches stored in memory. Due to the different catalog segmentation, there are various time measurements for the matching routine.

2. Run the STT routine with a sample image of the sky and measure the execution time.
3. Measure the core temperature of the RPi using the on-board sensor.

This test was performed for twelve hours. The chamber remained at a nearly constant temperature of 27°C, and the minimum pressure reached was $8.89 \cdot 10^{-4}$ Pa ($6.67 \cdot 10^{-6}$ Torr). An image of the RPi in the vacuum chamber test is shown in Figure 4.4. During the test, the RPi and the camera worked as expected. It showed no sign of failure, and the taken images were not distorted. Also, the temperature of the RPi-core did not show sharp changes.

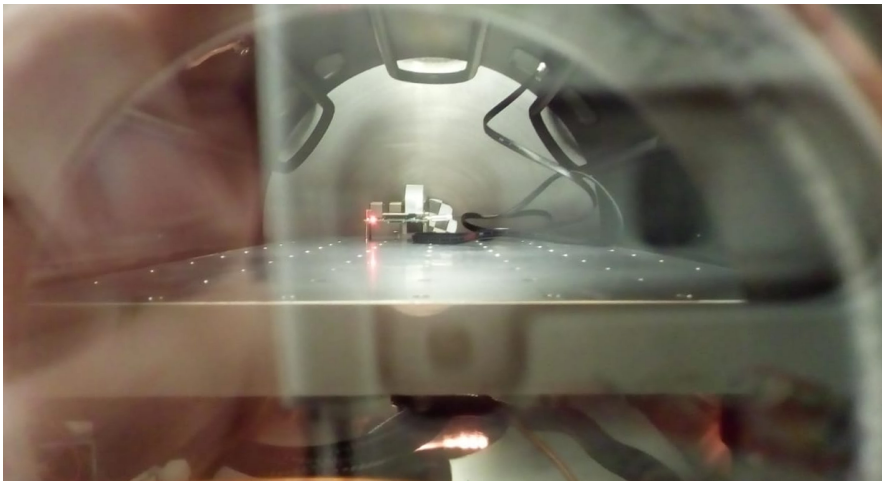


Figure 4.4: Raspberry Pi and its camera inside an NDT-4000 vacuum chamber. During the test, which lasted twelve hours, the temperature inside the chamber remained nearly constant at 27°C, and the minimum pressure reached was $8.89 \cdot 10^{-4}$ Pa ($6.67 \cdot 10^{-6}$ Torr).

4.2.5 Power consumption

The power consumption is an essential variable to consider when developing an STT. It can define the feasibility of being used in CubeSat missions. The power consumption of the STT platform during the operation is summarized in Table 4.3. It shows that in the current stage, the STT can be implemented in CubeSats of 3U or above, although no action has

been done thus far to diminish the power consumption. Further optimization might reduce consumption, especially during the *idle* state.

Process	Maximum power consumption (W)
Idle	2.0
Attitude computation	2.6
Image acquisition	2.5

Table 4.3: Power consumption in each STT stage using a Raspberry Pi 3 B+.

4.2.6 Discussion

The expected performance of SOST is based on the previously shown results, which will be discussed below. This discussion is worth it since it is possible to contrast the experimental results with theoretical ones.

SOST precision

The proposed STT was tested from ground, enabling to detect Earth's rotation, which is close to 15 arcseconds per second. Tests with angular rates below that value were not performed. Although the total processing time affects the attitude estimation, the limit precision of the SOST is given by the exposure time of the camera. The optimal exposure time to properly operate at ground level (see section 4.2.2) is 0.8 seconds.

It was not expected a precision better than 12 arcseconds (15 arcseconds/s $\times$ 0.8 s) with its platform (or equivalently with separation between projection centers of catalog segments below 5°). For this reason, 5° is the minimum separation between centers of catalog segments tested in this work. On the other hand, with 15° of separation between projection centers of catalog segments, the success rate is 85.6%. Larger separations between patches produce lower success rates.

The attitude estimation accuracy of the algorithm depends on how well the image matches the catalog segment. Therefore, a large overlap among catalog segments improves the success rate, as shown in Table 4.1. Thus, with a larger overlapping area, it is more likely that the image has enough sources within a catalog segment. Also, the accuracy of the algorithm is affected by the projection point in each catalog segment. For this reason, iterations 2 and 3 were included in the algorithm.

Besides the empirical approach I have previously described to estimate SOST precision, it was also consider theoretical methods previously proposed [2] that allow to assess several parameters of the STT. For example, consider the average number of stars in a circular FOV with A degrees wide. In the following equations the parameters used in this study are replaced, i.e., A = 48.8°, $N_{M_4} = 518$ stars, and $N_{\text{pixel}} = 1024$:

$$N_{\text{FOV}} = N_{M_4} \cdot \frac{1 - \cos(\frac{\text{A}}{2})}{2} = 518 \cdot \frac{1 - \cos(24.4)}{2} = 23 \text{ stars} \tag{4.1}$$

Using the obtained value ($N_{\text{FOV}} = 23$ stars), the error in the cross-boresight and the roll direction for the SOST can be estimated. It was assumed that the centroiding accuracy is 1 pixel since it is not allowed to use the hyperacuity technique (defocusing) with the RPi camera:

$$E_{\text{cross-boresight}} = \frac{\text{A} \cdot E_{\text{centroid}}}{N_{\text{pixel}} \cdot \sqrt{N_{\text{FOV}}}} = \frac{48.8 \cdot 1}{1024 \cdot \sqrt{23}} = 36 \text{ arcseconds} \tag{4.2}$$

$$\begin{aligned} E_{\text{roll}} &= \arctan\left(\frac{E_{\text{centroid}}}{0.3825 \cdot N_{\text{pixel}}}\right) \cdot \frac{1}{\sqrt{N_{\text{FOV}}}} \\ &= \arctan\left(\frac{1}{0.3825 \cdot 1024}\right) \cdot \frac{1}{\sqrt{23}} = 1.8 \text{ arcminutes} \end{aligned} \tag{4.3}$$

It is worth noting the similarity of these results to those obtained in the ground-based night-sky test, detailed in section 4.2.3 and shown in Figure 4.3.

Time of the estimation process

One of the main goals is to develop an easy-to-implement and ultra-low-cost, functional STT for some specific type of payload where the speed of the attitude estimation is not critical. The SOST has enough precision to fulfill its purposes, although the processing time can still be improved. As mentioned in section 1.1, the current SOST performance is enough for a couple of experiments integrated into our CubeSats [7]: the first experiment is a fixed-voltage flat-probe Langmuir Probe, which will estimate the plasma ion density. The second experiment is a phased array communication link system for studies of TEC. For both payloads, the attitude estimation does not need to be fast if the satellite is slowly spinning.

The evaluation (at ground level) of the STT shows that processing time is the critical feature to improve. For this reason, the impact of using a faster processor (a desktop PC) was evaluated. The software was installed, run, and tested in a dual-core Intel i5-7200U at 2.5GHz, 64 bits processor with 7.6 GB in memory RAM and Linux Ubuntu 18.04 as Operative System. The time needed by the different processes in the attitude estimation was measured, except the picture acquisition process, which is a specific module in the RPi. The processing time for three different catalog segmentation patterns (5°, 10°, and 15°) is presented in Table 4.4. The results consistently show that the processes are performed faster with more powerful processors.

Process	Average time (s)		
	5°	10°	15°
Importing libraries		0.47 ± 0.01	
Source extraction		0.5 ± 0.1	
Catalog matching	3.52 ± 0.07	0.94 ± 0.01	0.53 ± 0.01
Total	4.5 ± 0.1	1.9 ± 0.1	1.5 ± 0.1

Table 4.4: Processing time of each major routine of the attitude determination algorithm on a personal computer. These numbers are for distances of 5°, 10°, and 15° between centers of the catalog patches stored in memory. The picture acquisition process cannot be executed since it is an RPi specific process. The PC main characteristics are detailed in section 4.2.6. Due to the different catalog segmentation, there are various time measurements for the matching routine.

Power consumption of the Raspberry Pi

The power consumption of the STT routine in the RPi 3 B+ was shown in Table 4.3. These values can be considered dangerously high if we think of the limited energy capabilities of a CubeSat. Looking for a way to solve this issue, I also try the STT software in related hardware, the RPi Zero W and the RPi Zero 2 W. The main features of both hardware were previously shown in Table 2.1.

The power consumption for the RPi Zero W/Zero 2 W is shown in Table 4.5. Due to the different computing capabilities, the times of the STT routine changes compared with the RPi 3 B+. Table 4.6 shows the time of each stage of the STT routine on the Raspberry Pi Zero W/Zero 2 W.

Process	Average power consumption (W)	
	RPi Zero W	RPi Zero 2 W
Idle	0.5	0.7
Attitude computation	1.0	2.5

Table 4.5: Power consumption using a Raspberry Pi Zero W/ Zero 2 W as STT.

Process	Average time (s) - RPi Zero W			Average time (s) - RPi Zero 2 W		
	5°	10°	15°	5°	10°	15°
Importing libraries		26.2 ± 0.3			3.81 ± 0.02	
Picture acquisition		8.73 ± 0.03			1.69 ± 0.01	
Source extraction		8.7 ± 0.6			2.84 ± 0.01	
Catalog matching	114 ± 3	32 ± 1	17.8 ± 0.6	12.5 ± 0.2	3.9 ± 0.2	2.2 ± 0.1
Total	158 ± 3	76 ± 1	61.4 ± 0.6	20.8 ± 0.2	12.2 ± 0.2	10.5 ± 0.1

Table 4.6: Processing time of each major routine in Raspberry Pi Zero W/Zero 2 W. These numbers are for distances of 5°, 10°, and 15° between projection centers of the catalog segments stored in memory. Due to the different catalog segmentation, there are various time measurements for the matching routine.

About the developed open-source platform

There was developed a platform, open to everyone interested in a simple and easy-to-use STT for CubeSats. The platform and test images can be downloaded from `http://github.com/spel-uchile/Star_Tracker`. I hope that this platform might be helpful for the development of an STT in new small groups of CubeSat developers, and for experienced developers who need a simple and fast solution. Trough the GitHub page, people for different groups put in contact with me to talk about the development. Firstly, I will mention Mr. Kamil Borkowsky from the HyperSat team in Warsaw. He tried this open platform in the testing phase of their satellite development. Secondly are Mr. Nathan Raposo and Mr. Daniel Gatti from Maúa Institute of Technology in São Paulo, Brazil. They also tried the code and are developing new applications with it.

4.3 Radiation effects assessment

To deeply study and understand the possible effects produced on the camera sensor due to the radiation that affects every spacecraft in LEO (previously described in section 2.7), radiation effects are simulated on the previously acquired night sky images. The same subset of 50 night-sky pictures selected for the previous evaluation (used for the analysis presented in subsection 4.2.3) was used in this experiment. Three different simulations were carried on over the image set:

- Adding random spots
- Adding straight lines
- Removing stars

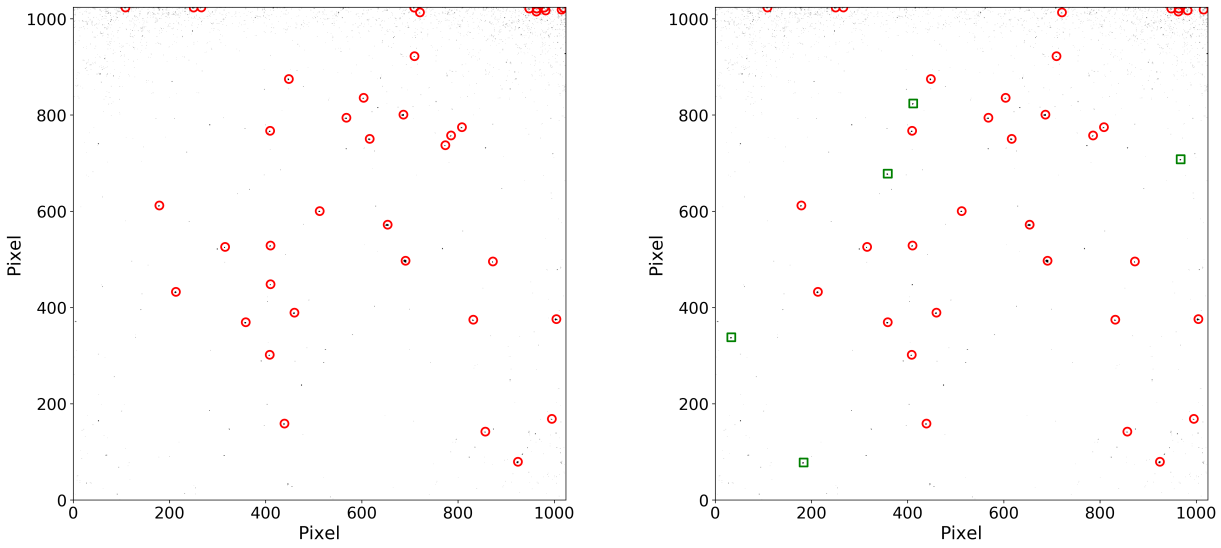
These experiments represent the following scenarios: adding spots and lines represents the collision of high-energy particles with the camera sensor. This collision can be direct or oblicuous, generating dots or lines, respectively. The experiment of removing stars represents the possibility that some stars fall into a dead pixel. I will describe the methods used in each simulation and then present the obtained results.

4.3.1 Method description

Adding random spots

In this experiment, white spots were randomly added to an image. The added white spots have the maximum possible intensity value, which is 255. For every image, the number of added spots ranges from 1 to 30, and the spot sizes range from 1 pixel to 10 pixels. For a specific image, a particular number of spots, and a particular spot size, white dots were added in random positions 20 times, and the LIS problem was tried to solve in each of the cases. As an example of an evaluated image, Figure 4.5 shows a night-sky image in which

five spots of size 7 pixels were randomly added.



(a) Original image and its extractions.

(b) Original image plus white spots.

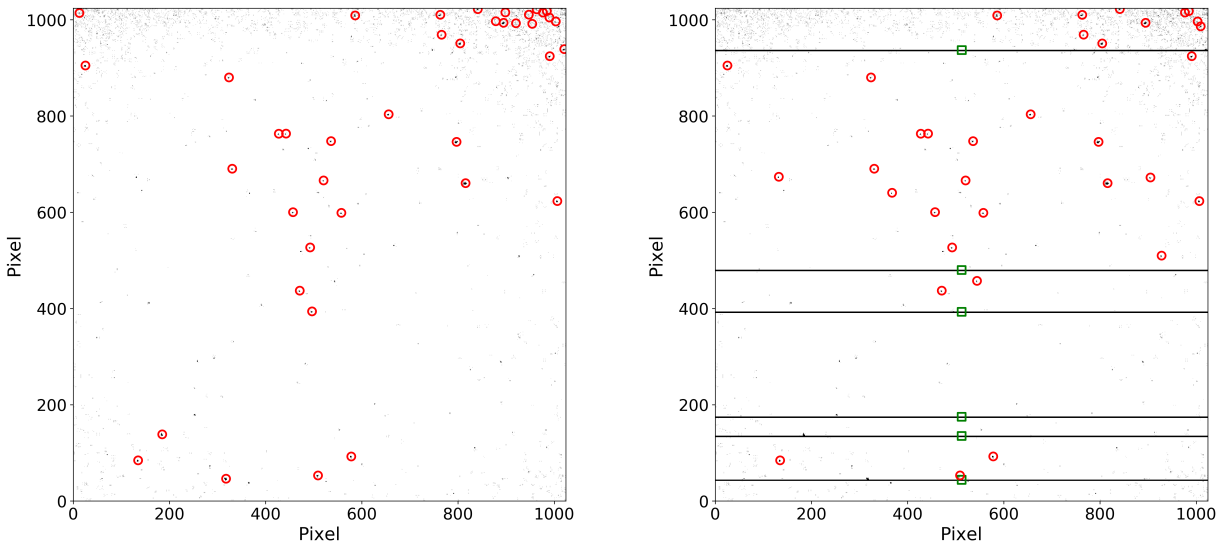
Figure 4.5: Sample image of the night sky taken with the Raspberry Pi camera. Figure (a) shows the negative of the original image plus the objects extracted from it (marked with a red circle). Figure (b) shows the extractions of the image in which five white spots of size seven pixels were added in random positions. The white spots are detected and marked with a green square. The rest of the extractions from the image are marked with a red circle.

The studied parameters were the change in the success percentage of the LIS problem, and the number of related objects between picture and catalog as the number of white spots and their size increased. Finally, the results for every spot size and number of spots were averaged over the total subset of 50 images.

Adding straight lines

In this experiment, straight white lines were randomly added to an image. The added lines have constant intensity with its maximum possible value, 255. For every image, the number of added lines ranges from 1 to 10, and the line width range from 1 pixel to 5 pixels. For a specific image, a particular number of lines, and a particular line width, straight white lines were added in random positions 20 times, and the LIS problem was tried to solve in each of the cases. As an example of an evaluated image, Figure 4.6 shows a night-sky image with six lines of width 3 pixels randomly added.

Similar to the previous analysis, the studied parameters were the change in the success percentage of the LIS problem and the number of related objects between picture and catalog as the number of white lines and their width increased. Finally, the results for every line width and number of lines were averaged over the total subset of 50 images.



(a) Original image and its extractions.

(b) Original image plus white lines.

Figure 4.6: Sample image of the night sky taken with the Raspberry Pi camera. Figure (a) shows the negative of the original image plus the objects extracted from it (marked with a red circle). Figure (b) shows the extractions of the image in which six white lines of width three pixels were added in random positions. The white lines are detected and marked with a green square. The rest of the extractions from the image are marked with a red circle. Notice that all the extractions coming from lines are detected in the middle of the picture; this is due to the uniformly distributed intensity of the line.

Removing stars

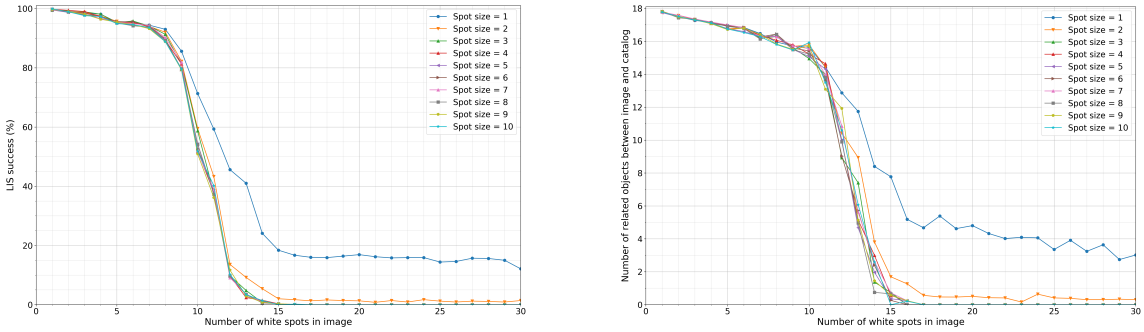
In this experiment, extractions from the picture are removed. The original extractions from the image are sorted from the brightest to the dimmest. Then, extracted objects are removed one by one. This removal has been done in two ways: start removing from the brightest extractions and the dimmest extractions. When **removing brightest stars**, the object removal starts from the brightest objects towards the dimmest. In the case of **removing dimmest stars**, the object removal starts from the dimmest objects towards the brightest. In both cases, objects are extracted until there is not possible to get a solution to the LIS problem in no picture.

Similar to the last analysis, the studied parameters were the change in the success percentage of the LIS problem and the number of related objects between picture and catalog as the extractions from the image were removed. Finally, the results for each removed star were averaged over the total subset of 50 pictures.

4.3.2 Results

Adding random spots

Figure 4.7 shows the change in both the LIS success percent and the number of objects related between the catalog and picture as the number of white spots in the image increases. Figure (a) shows a slight decrease in the LIS success rate as the number of white spots increases, reaching 90% when eight white spots are added to the images. After that, the LIS success decreases dramatically until just 10%, when 12 white spots are added. The results are almost the same, independent of the size of the spots. The only remarkable difference is when the spot size is one pixel; in this case, the decrease in the LIS success is not so abrupt compared with the other white spot sizes. Figure (b) shows a similar trend as Figure (a), but in this case, it shows the number of related objects between the catalog and the image. In this Figure, a constant slight decrease last until ten white spots are added to the picture. After that, the number of related objects decreases sharply, nearly to 0, when 15 white spots are added to the image.

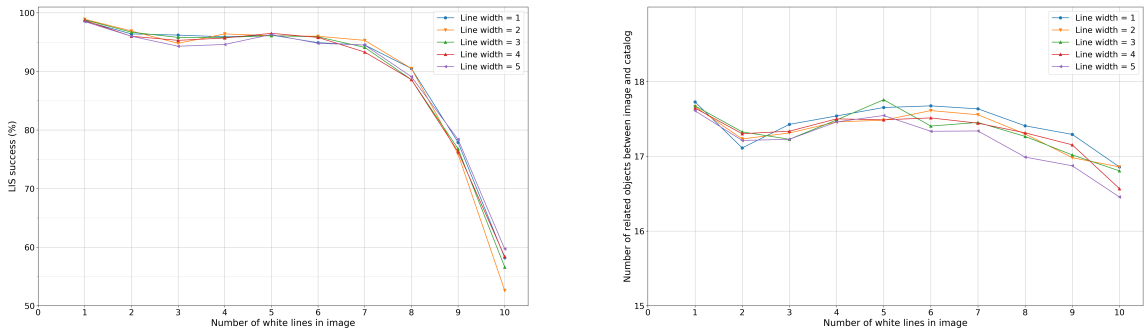


(a) Change in LIS success (%). **(b)** Change in the number of related objects.

Figure 4.7: The Figure shows the results of adding white spots of different sizes over the night-sky images. Figure (a) shows the change in the LIS percent success, and Figure (b) shows the change in the number of related objects between the catalog and picture due to the addition of white spots.

Adding straight lines

Figure 4.8 shows the change in both the LIS success percent and the number of objects related between the catalog and picture as the number of white lines in the image increases. Figure (a) shows that the LIS success is almost constant until seven white lines are added to the images. After that, the LIS success decreases sharply until near 60%, when ten white lines are added. It is possible to notice that the trend is the same as the observed in the experiment with white spots. Figure (b) shows that the number of related objects between the catalog and the image is almost constant. A slight decrease is observed in this figure from adding eight white lines to the picture. In both Figures, we can see that the result is practically independent of the width of the lines added to the images.



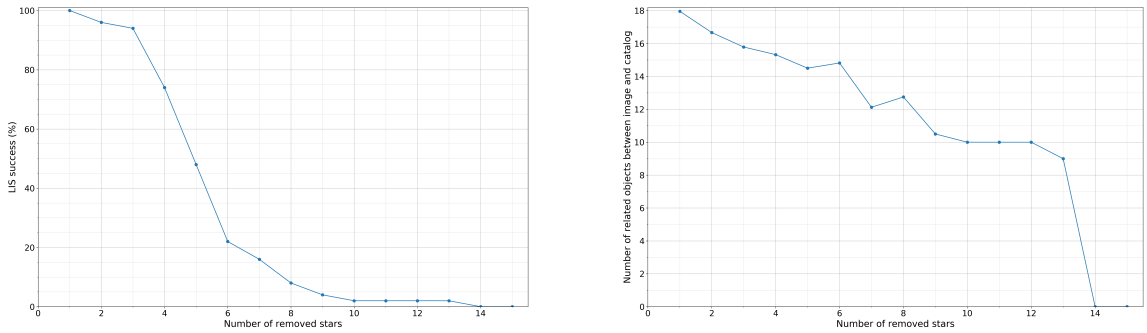
(a) Change in LIS success (%). **(b)** Change in the number of related objects.

Figure 4.8: The Figure shows the results of adding white lines of different sizes over the night-sky images. Figure (a) shows the change in the LIS percent success, and Figure (b) shows the change in the number of related objects between the catalog and picture due to the addition of white lines.

Removing stars

Removing brightest stars

Figure 4.9 shows the change in both the LIS success percent and the number of objects related between the catalog and picture as the number of brightest extracted objects from the pictures is removed. Figure (a) shows a sharp decrease in the LIS success percent after removing three extractions. The LIS success constantly decreases, reaching less than 20% when seven extractions are removed. Finally, the LIS percent success drops to 0% when fourteen extractions are removed. Figure (b) shows a steady decreasing trend in the number of related objects between the catalog and the image. This number goes to 0 when fourteen extractions are removed.



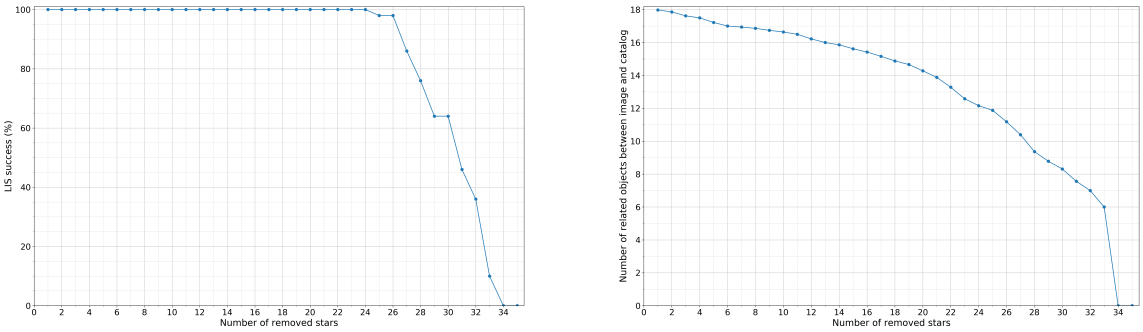
(a) Change in LIS success (%). **(b)** Change in the number of related objects.

Figure 4.9: The Figure shows the results of removing extracted objects from the night-sky images, starting with the brightest extractions. Figure (a) shows the change in the LIS percent success, and Figure (b) shows the change in the number of related objects between the catalog and picture due to the removal of extractions.

Removing dimmest stars

Figure 4.10 shows the change in both the LIS success percent and the number of objects

related between the catalog and picture as the number of dimmest extracted objects from the pictures is removed. Figure (a) shows an almost constant value of the LIS success until 26 extractions are removed. After that, The LIS success constantly decreases, reaching 0% when 34 extractions are removed. Figure (b) shows a steady declining trend in the number of related objects between the catalog and the image. This number goes to 0 when 34 extractions are removed.



(a) Change in LIS success (%). **(b)** Change in the number of related objects.

Figure 4.10: The Figure shows the results of removing extracted objects from the night-sky images, starting with the dimmest extractions. Figure (a) shows the change in the LIS percent success, and Figure (b) shows the change in the number of related objects between the catalog and picture due to the removal of extractions.

The algorithm seems relatively robust to radiation events over the image sensor. It works well until eight spots/lines are added to the image. After that, the effect on the LIS algorithm starts to be considerable (success rate under 90%). In the case of removing stars, the algorithm seems very sensitive to the presence of bright stars, capable of tolerating the absence of just three bright stars until the success rate decays significantly (under 80%).

Finally, I want to mention that this robustness analysis under radiation is open to everyone interested in reviewing it and replicating it in the same way as the previously proposed star tracker algorithm. This can be accessed through the GitLab website in `http://gitlab.com/spel-uchile/testing-stt-robustness`.

Chapter 5

Star tracker integration and set up for on-flight evaluation

5.1 Being part of a CubeSat project

There is one STT already flying in space in each of the three 3U CubeSats developed at SPEL. As it is developed under the CubeSat standard, it must fulfill the characteristics of low mass, small volume, and low energy consumption. Also, the design of the PCB, which holds the STT, must satisfy the PC-104 standard. It is necessary to adapt the payload to the requirements imposed by the satellite bus. The STT payload, while it was being integrated into the satellite SUCHAI-2 from SPEL, is shown in Figure 5.1.

5.2 Payload integration with Flight Software (FS)

For the payload to operate within the satellite, it needs to fulfill the hardware requirements. Still, it also needs to communicate with the on-board computer (OBC) and the rest of the main subsystems that compound the satellite. These subsystems include, for example, the electrical power subsystem (EPS), the transceiver (TRX), and possibly other payloads. To receive commands from the ground station (GS), the EPS needs to energize the payload, and the TRX (which receives the message) sends it to the payload. Under the command-based pattern in which the SUCHAI operates [131], the payload needs to run an application similar to the FS running on the OBC. I set a series of requirements that the payload needs to fulfill. These requirements are applied to the payload as commands sent through the FS. These requirements and the basic commands created to satisfy the requirements are shown below.

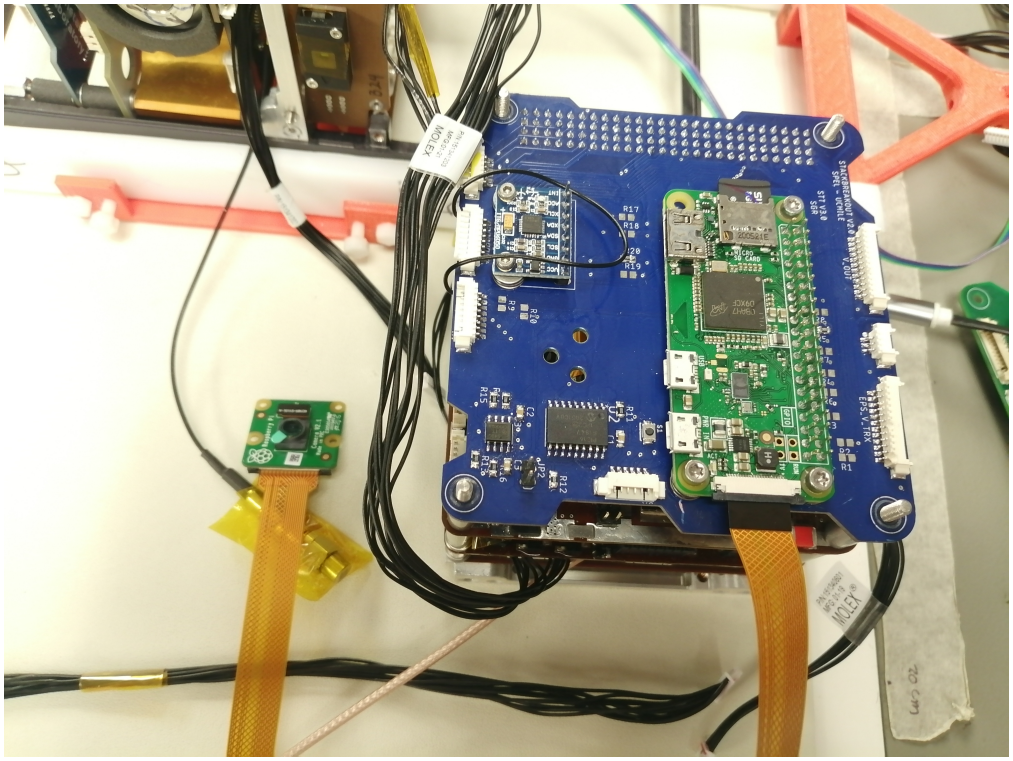


Figure 5.1: STT payload integrated into SUCHAI-2. The picture shows the PCB based on the PC-104 standard that holds the experiment: the Raspberry Pi, the gyroscope, and the associated electronics. Also, the RPi camera and its cable are visible.

Main requirements

To develop an essential functionality in the STT, I establish a set of requirements that shape it. These requirements are shown in Table 5.1.

Commands

To satisfy the previously established requirements, I developed a series of commands shown in Table 5.2.

5.3 Validation through on-flight experiments

As this will be the first CubeSat from SUCHAI missions that will carry on payloads to determine and control attitude, it becomes necessary to utilize this as a study and learning platform. In that way, it becomes exciting and essential to developing experiments to study the hardware and software behavior of the STT in space to apply this knowledge on future missions. The experiments proposed to be carried out in space are detailed below. The command referred to are those presented in Table 5.2.

ID	Requirement
1	The payload is able to turn on and turn off through the satellite power bus.
2	The payload is able to reboot.
3	The payload shows that it is ready to be operated.
4	It is possible to set the time in the payload.
5	The payload can take a picture of the night-sky.
6	The payload can take a picture of the night-sky and analyze if there are stars in it.
7	The payload can take a picture of the night-sky and solve the LIS problem.
8	The payload can take a series of night-sky pictures and apply different tracking methods.
9	The payload can determine the optimal exposure time for taking night-sky pictures.
10	The payload can measure the rotation speed of the satellite.
11	The payload can measure temperature.
12	The payload provides tools to handle in-memory pictures or files.

Table 5.1: Set of requirements established to develop a basic functionality as STT in the payload. These requirements are later traduces into commands in the payload.

1. **Survival in space:** One of the first elements to evaluate is the capacity of the hardware to survive and work properly in space. If the payload is energized and the hardware and software are working correctly, the payload can give a predefined short response to the command `stt_test`.
2. **Taking pictures from space:** The first step to making a star tracker work is the capability to take pictures from the night-sky. This ability can be evaluated with the command `stt_take_pic`. The taken pictures are saved to a predefined folder.
3. **Evaluating radiation effects on camera:** The pictures taken with the star tracker can help evaluate the effects of radiation on the camera sensor. We can take sky photos from a low radiation site and compare these with others taken in a high radiation environment (as the SAA, depicted in Figure 2.4). The existence of artifacts in the pictures can give evidence of the effects previously described in subsection 2.7.2.
4. **Finding the right exposure time:** Although the exposure time to find enough stars in a picture to solve the LIS problem on Earth is known (see section 4.2.2), this is not known in space. I hope that the exposure time in space will be lower than on Earth, but the specific value is unknown. For that reason, an experiment is proposed to determine a pseudo-optimal exposure time. While in eclipse, take many pictures in a range of selected exposure times. Also, an expected value of extracted objects is given. The minimal exposure time in which the number of extracted objects is closer than the expected value is considered the pseudo-optimal exposure time. This experiment is executed with the command `stt_exp_time_eval`.
5. **Evaluating space pictures:** During the first operations of the star tracker, we will not know the pointing direction of the payload. The pictures might not be of stars. Therefore, it becomes convenient to evaluate the taken pictures before downloading them to the ground station. For this reason, it is possible to assess the images using

Command ID	Associated requirement ID	Command	Parameters description	Brief command description
1	1	`eps_set_output A B`	**A:** Payload number **B:** Energy selector	Turn on/off payload
2	2	`stt_reboot A`	**A:** Communication node	Reboot the payload
3	3	`stt_test A`	**A:** Communication node	Test if payload is ready to operate
4	4	`com_set_time_node A`	**A:** Payload node	Set time from OBC to the payload
5	5	`stt_take_pic A B`	**A:** Communication node **B:** Exposure time	The payload takes a picture
6	6	`stt_eval_pic A B C`	**A:** Communication node **B:** Exposure time **C:** Darkness threshold	The payload takes a picture and evaluate it
7	7	`stt_solve_lis A B C`	**A:** Communication node **B:** Exposure time **C:** Catalog division	Take a picture and apply the LIS algorithm over it
8	8	`stt_solve_lis_and_track A B C D`	**A:** Communication node **B:** Exposure time **C:** Catalog division **D:** Total steps	Take a series of pictures and apply the LIS algorithm over them
9	9	`stt_exp_time_eval A B C D E`	**A:** Communication node **B:** Initial exposure time **C:** Final exposure time **D:** Time step **E:** Expected stars	Take a series of pictures and determine the best exposure time
10	10	`stt_gyro_n A B C`	**A:** Communication node **B:** Number of measurements **C:** Time step	Take a series of measurements with the gyroscope
11	11	`stt_rpi_temp A`	**A:** Communication node	Measures the RPi-core temperature
12	12	`stt_ls A B C`	**A:** Communication node **B:** Selected folder **C:** Number of files	List files of a specific folder
13	12	`stt_folder_size A B`	**A:** Communication node **B:** Selected folder	Get size of a specific folder
14	12	`stt_erase A B`	**A:** Communication node **B:** Selected folder	Erase data from a specific folder
15	12	`stt_thumb A B C D`	**A:** Communication node **B:** Full path of image **C:** Reduction factor **D:** Quality	Create a thumbnail of a picture

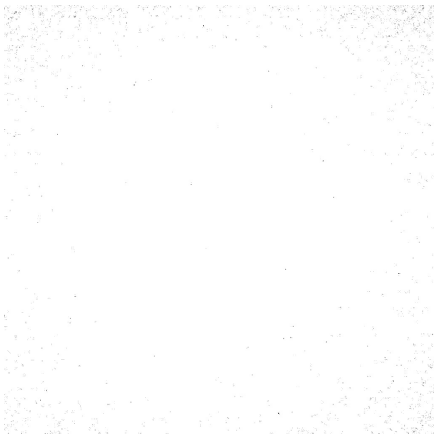
Table 5.2: Set of commands developed to fulfill the requirements previously described in Table 5.1. The parameters of every command and a brief description are presented. These commands are used to perform the experiments detailed in section 5.3.

the command `stt_eval_pic`. With this aid, it becomes easier to select suitable images to download. Additionally, the file size of the image is also a good indicator of what is probably seeing in the image. Figure 5.2 shows that different images can be related with different file sizes. It introduces another comparison point into this evaluation.

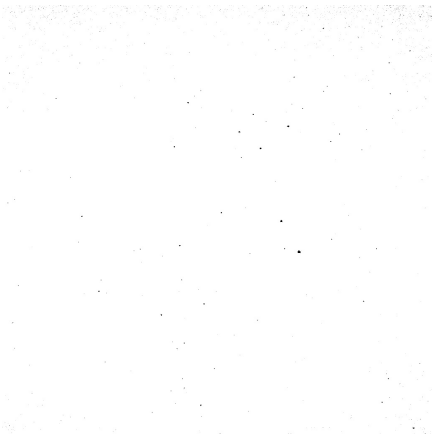
6. **Finding the attitude by solving the LIS problem:** The core ability of any star tracker is to find the attitude from a picture of the stars. This ability is tested with the command `stt_solve_lis`. The taken pictures are saved from being downloaded later to the ground station and then compared the solution obtained in space with the solution obtained on Earth with the same image. Also, the time of LIS computation is saved for later analysis.
7. **Upgrading star tracker performance:** Besides getting the attitude from a picture of the stars, the star tracker can propagate attitude measurement using different kinds of filters. A method using searching in catalog segments near the previous attitude

solution was programmed in the command `stt_solve_lis_and_track`.

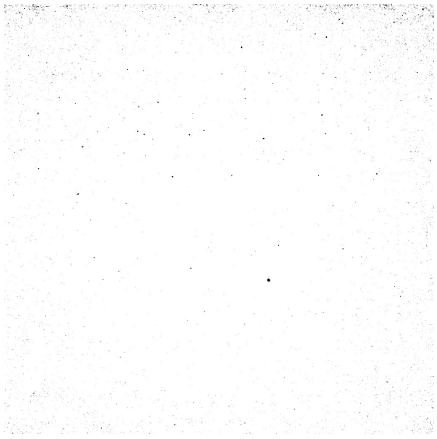
8. **Study hardware temperature:** Due to the extreme temperature values presented in space, it is essential to know how this will affect the hardware of the payload. For that reason, the command `stt_rpi_temp` was developed to measure the RPi-core temperature. This measurement will help to check the ranges of temperature that affect the RPi and monitor if this can cause some damage to the hardware.
9. **Operating STT from ground station:** There will be a lot of data from the different experiments proposed here. For that reason, it is reasonable to have a simple interface to navigate between files, get the file size, and have the ability to erase them (in case of memory limitations). This file manipulation is done with the commands `stt_ls`, `stt_folder_size`, and `stt_erase`, respectively.
10. **Downloading space pictures:** Downloading a high-quality image from the satellite to the GS can be time-consuming. This process can not be done in a parallel way, thus limiting the time and data we can access on the satellite. For that reason, it becomes necessary to develop a method to download images in a fast way. This task is done by creating a thumbnail of it using the command `stt_thumb`. This strategy allows us a fast download and thus be able to discriminate if that is a valuable image rapidly. If the downloaded image is interesting for research purposes, then the high-quality version of it can be downloaded later. The developed method is depicted in Figure 5.3.



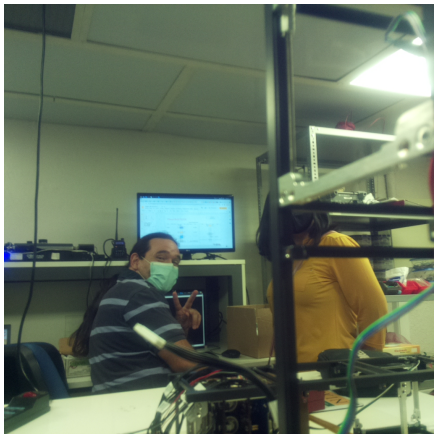
(a) Full-dark image. Image size: 54,7 KB.



(b) Night-sky image with 1179 extractions. Image size: 74,6 KB.

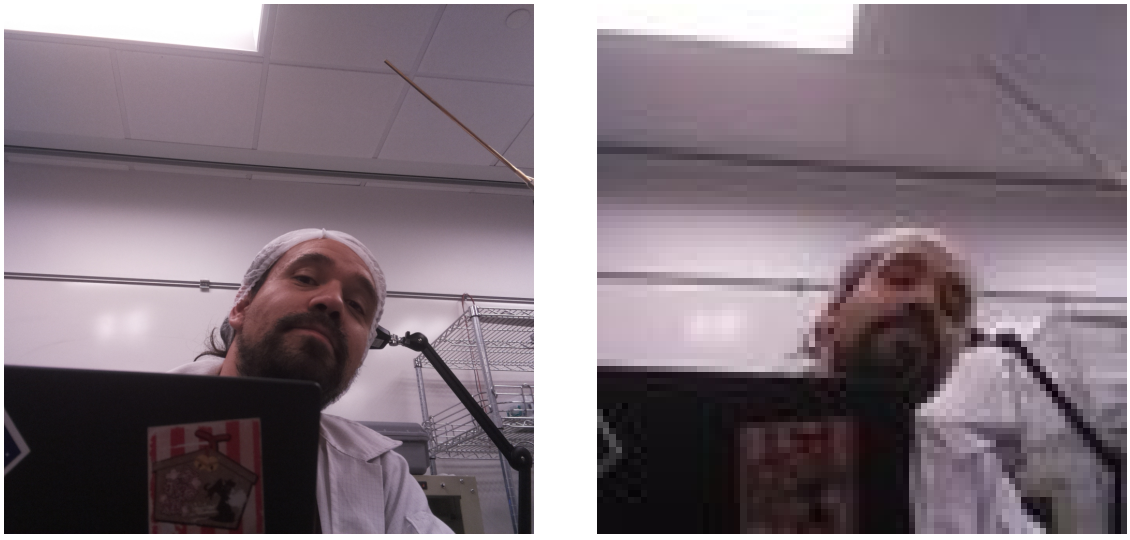


(c) Night-sky image with 5303 extractions. Image size: 134,9 KB.



(d) Full-color image. Image size: 692,4 KB.

Figure 5.2: It is shown different images acquired with the RPi. The file size of the image can be used as a discriminator to know if it is worth downloading. Figure (a) shows an entirely dark image (54,7 KB). Figure (b) shows a night-sky image with 1179 full extractions (74,6 KB). Figure (c) also shows a night-sky image with more extractions (5303), and the file size increases (134,9 KB). Lastly, for comparison purposes, Figure (d) shows a full-color image (692,4 KB). Figures (a), (b), and (c) are in inverse color for clarity purposes.



(a) Original image (634,1 KB). **(b)** Thumbnail (1,9 KB).

Figure 5.3: The Figure shows a 15 ms exposure time picture taken with the RPi camera of the SUCHAI-3 nanosatellite (634,1 KB) and the thumbnail of that picture (1,9 KB). The reduction factor used is 12, and the quality is 80% of the original image. Although the image size is strongly reduced, the main details are still clearly visible. This procedure demonstrates that reducing the image size before downloading is feasible without considerably affecting the image quality.

Chapter 6

Conclusions and future work

This dissertation presents the development of a novel ultra-low-cost star tracker. This development considers all stages, from creating an algorithm capable of solving the LIS problem to the proposal of experiments to test the STT performance in space, going through the test of the algorithm and the hardware, and the integration into a CubeSat.

An attitude determination algorithm that uses proven common-use astronomy software was developed, capable of solving the LIS problem by reaching arcminutes precision or better. The algorithm is tested with images from currently operative satellites (STEREO mission of NASA). This algorithm can be adapted to different fields of view.

A COTS hardware was chosen to test the STT software, the Raspberry Pi Zero W/Zero 2 W. It satisfies the requirements imposed by the CubeSat standard in terms of mass, volume, and energy consumption. To fully solve the LIS problem, the RPi Zero W takes 76 ± 1 s, considering a distance of 10° between projection centers of the catalog segments in the matching routine, consuming 1.0 W during attitude computation. When a distance of 5° between projection centers of the catalog segments is considered, processing times increase to 158 ± 3 s. In the case of the RPi Zero 2 W, it takes 12.2 ± 0.2 s considering a distance of 10° between projection centers of the catalog segments, consuming 2.5 W during attitude computation. When a distance of 5° between projection centers of the catalog segments is considered, processing times increase to 20.8 ± 0.2 s. In both cases, the success rate is $97.3 \pm 2.5\%$ and $99.8 \pm 0.6\%$ when considering a distance of 10° and 5° between projection centers of the catalog segments, respectively. Additionally, the precision that is possible to achieve with this hardware was tested in two different ways: using night-sky images and based on theoretical considerations. Both results converge.

The algorithm and the hardware components make up an open-source and ultra-low-cost STT platform. This platform is available to interested developers and researchers in nanosatellites in the GitHub page of the SPEL group: `http://github.com/spel-uchile/Star_Tracker`, and it is continuously improved.

A relevant concern of an STT while on orbit is its behavior under possible radiation effects (SEE), mainly coming from anomalies like the SAA. Under this scenario, the robustness of

the algorithm was evaluated. The algorithm's behavior was assessed when simulating the effect of adding spots and lines over the image and removing stars. When adding spots, the algorithm runs well until eight white spots are added to the image, when the success rate decay to 90%. After that, the decline in the success rate is significant. When adding white lines, a similar trend is observed. When removing objects from the brightest to the dimmest, it is observed that after removing three objects, the success rate decay under 80%. Removing objects in the opposite direction has almost no impact on the algorithm until 26 objects are removed. This robustness analysis under radiation is open to everyone interested in reviewing it and replicating it in the same way as the SOST platform through the GitLab page of the SPEL group:: `http://gitlab.com/spel-uchile/testing-stt-robustness`.

Furthermore, it is presented the integration of the STT into the new series of SUCHAI nanosatellites. This integration involves the software integration into a flight software framework and a hardware integration following the PC-104 standard. The main requirements are discussed, and commands which fulfill the requirements are developed. Also, it is presented the proposed experiments to be set while operating the STT, which is currently in orbit in our new CubeSats.

Regarding future work, three different roads are currently being developed. The first is about testing the STT in the three new 3U CubeSats orbiting the Earth. The proposed experiments exposed in this dissertation are being carried out, and the results of the survival of the RPi in the harsh space environment are promising. The second avenue is the development of a new matching algorithm to make the LIS algorithm faster, more efficient, and more robust. To do this, an algorithm taken from the literature based on rarely occurring triangles formed with extractions from the image is in the implementation process. The final route worth mentioning is the study of the effects of the radiation over STT and the LIS algorithm while in orbit. This is done primarily by analyzing the night-sky pictures from the STT in orbit.

Bibliography

[1] C. C. Liebe, "Star trackers for attitude determination", *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, no. 6, pp. 10–16, 1995.

[2] C. C. Liebe, "Accuracy performance of star trackers-a tutorial", *IEEE Transactions on aerospace and electronic systems*, vol. 38, no. 2, pp. 587–599, 2002.

[3] P. Salomon and W. Goss, "A microprocessor-controlled CCD star tracker", in *14th Aerospace Sciences Meeting*, 1976, p. 116. DOI: 10.2514/6.1976-116.

[4] J. L. Junkins, C. C. White, and J. D. Turner, "Star pattern recognition for real time attitude determination", *Journal of the Astronautical Sciences*, vol. 25, no. 3, pp. 251–270, 1977.

[5] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka, and R. Twiggs, "Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation", *Proceedings of the AIAA/USU Conference on Small Satellites*, 2000.

[6] K. Woellert, P. Ehrenfreund, A. J. Ricco, and H. Hertzfeld, "Cubesats: Cost-effective science and technology platforms for emerging and developing nations", *Advances in Space Research*, vol. 47, no. 4, pp. 663–684, 2011. DOI: 10.1016/j.asr.2010.10.009.

[7] M. Diaz, J. Zagal, C. Falcon, M. Stepanova, J. Valdivia, M. Martinez-Ledesma, J. Diaz-Peña, F. Jaramillo, N. Romanova, E. Pacheco, *et al.*, "New opportunities offered by cubesats for space research in latin america: The SUCHAI project case", *Advances in Space Research*, vol. 58, no. 10, pp. 2134–2147, 2016. DOI: 10.1016/j.asr.2016.06.012.

[8] National Academies of Sciences, Engineering, and Medicine and Space Studies Board and others, *Achieving Science with CubeSats: Thinking Inside the Box*. National Academies Press, 2016. DOI: 10.17226/23503.

[9] C. Boshuizen, J. Mason, P. Klupar, and S. Spanhake, "Results from the planet labs flock constellation", *Proceedings of the AIAA/USU Conference on Small Satellites*, 2014.

[10] M. Knapp, S. Seager, B.-O. Demory, A. Krishnamurthy, M. W. Smith, C. M. Pong, V. P. Bailey, A. Donner, P. Di Pasquale, B. Campuzano, *et al.*, "Demonstrating high-precision photometry with a cubesat: Asteria observations of 55 cancri e", *The astronomical journal*, vol. 160, no. 1, p. 23, 2020.

[11] A. Kak and I. F. Akyildiz, "Large-scale constellation design for the internet of space things/cubesats", in *2019 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2019, pp. 1–6.

[12] C. Gonzalez, C. Rojas, A. Becerra, J. Rojas, T. Opazo, and M. Diaz, "Lessons learned from building the first chilean nano-satellite: The SUCHAI project", *AIAA/USU Conference on Small Satellites*, 2018.

[13] S. Nakasuka, N. Sako, H. Sahara, Y. Nakamura, T. Eishima, and M. Komatsu, "Evolution from education to practical use in university of tokyo's nano-satellite activities", *Acta Astronautica*, vol. 66, no. 7-8, pp. 1099–1105, 2010.

[14] J. Guo, J. Bouwmeester, and E. Gill, "In-orbit results of delfi-n3xt: Lessons learned and move forward", *Acta Astronautica*, vol. 121, pp. 39–50, 2016.

[15] W. Lan, J. Brown, A. Toorian, R. Coelbo, L. Brooks, J. Puig-Suari, and R. Twiggs, "Cubesat development in education and into industry", in *Space 2006*, American Institute of Aeronautics and Astronautics, 2006, p. 7296.

[16] 2. Horas, *Three satellites made in chile launched into space*, Available: `http://www.24horas.cl/nacional/lanzan-al-espacio-tres-satelites-fabricados-en-chile-5247020`, [Online]. Accessed on: April 13, 2023.

[17] Sparkfun, *Sparkfun triple axis accelerometer and gyro breakout - MPU-6050*, Available: `http://www.sparkfun.com/products/11028`, [Online]. Accessed on: April 13, 2023.

[18] Sparkfun, *Sparkfun triple axis magnetometer breakout - MLX90393*, Available: `http://www.sparkfun.com/products/14571`, [Online]. Accessed on: April 13, 2023.

[19] SatCatalog, *MAI-SES IR Earth sensor*, Available: `http://www.satcatalog.com/component/ai-ses-ir-earth-sensor/`, [Online]. Accessed on: April 13, 2023.

[20] CubeSatShop, *NSS Fine sun sensor*, Available: `http://www.cubesatshop.com/product/digital-fine-sun-sensor/`, [Online]. Accessed on: April 13, 2023.

[21] CubeSatShop, *Arcsec sagitta star tracker*, Available: `http://www.cubesatshop.com/product/sagitta-star-tracker/`, [Online]. Accessed on: April 13, 2023.

[22] J. C. Springmann, "Satellite attitude determination with low-cost sensors", `http://hdl.handle.net/2027.42/102312` [Online]. Accessed on: April 13, 2023, PhD thesis, University of Michigan, 2013.

[23] R. Burton, S. Weston, and E. Agasid, "State of the art in guidance, navigation and control: A survey of small satellite GNC components", in *Proc. Adv. Astron. Sci.*, vol. 157, 2016.

[24] R. P. Foundation, *Teach, learn and make with the raspberry pi foundation*, Available: `http://www.raspberrypi.org/`, [Online]. Accessed on: April 13, 2023.

[25] S. Pedrotty, R. Lovelace, J. Christian, D. Renshaw, and G. Quintero, "Design and performance of an open-source star tracker algorithm on commercial off-the-shelf cameras and computers", in *43rd Annual AAS Guidance, Navigation and Control Conference*, 2020.

[26] M. Samaan and S. Theil, "Development of a low cost star tracker for the SHEFEX mission", *Aerospace science and technology*, vol. 23, no. 1, pp. 469–478, 2012.

[27] C. R. McBryde and E. G. Lightsey, "A star tracker design for cubesats", in *Aerospace Conference, 2012 IEEE*, IEEE, 2012, pp. 1–14. DOI: `10.1109/AERO.2012.6187242`.

[28] A. O. Erlank and W. H. Steyn, "Arcminute attitude estimation for cubesats with a novel nano star tracker", *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 9679–9684, 2014.

[29] Y. Jianan, L. Yong, H. Fei, F. Qian, and P. Quan, "Pico-satellite attitude determination using a star tracker with compressive sensing", in *Control Conference (CCC), 34th Chinese*, IEEE, 2015, pp. 5331–5336. DOI: `10.1109/ChiCC.2015.7260472`.

[30] J. Straub, C. Korvald, A. Nervold, A. Mohammad, N. Root, N. Long, and D. Torgerson, "Openorbiter: A low-cost, educational prototype cubesat mission architecture", *Machines*, vol. 1, no. 1, p. 1, 2013. DOI: `10.3390/mach1010001`.

[31] A. Scholz and J.-N. Juang, "Toward open source cubesat design", *Acta astronautica*, vol. 115, pp. 384–392, 2015. DOI: 10.1016/j.actaastro.2015.06.005.

[32] NASA, *Star trackers lights the way*, Available: http://www.nasa.gov/multimedia/podcasting/StarTrackers.html, [Online]. Accessed on: April 13, 2023, 2018.

[33] J. R. Wertz, *Spacecraft attitude determination and control*. Springer Science & Business Media, 2012, vol. 73.

[34] R. W. Astheimer, "Infrared horizon sensors for attitude determination", *IFAC Proceedings Volumes*, vol. 2, no. 1, pp. 381–396, 1965.

[35] J. L. Junkins and H. Schaub, *Analytical mechanics of space systems*. American Institute of Aeronautics and Astronautics, 2009.

[36] F. L. Markley and J. L. Crassidis, *Fundamentals of spacecraft attitude determination and control*. Springer, 2014, vol. 33.

[37] J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999.

[38] E. Canuto, C. Novara, D. Carlucci, C. P. Montenegro, and L. Massotti, *Spacecraft Dynamics and Control: The Embedded Model Control Approach*. Butterworth-Heinemann, 2018.

[39] D. A. Vallado, *Fundamentals of astrodynamics and applications*. Springer Science & Business Media, 2001, vol. 12.

[40] H. D. Curtis, *Orbital mechanics for engineering students*. Butterworth-Heinemann, 2013.

[41] B. B. Spratling and D. Mortari, "A survey on star identification algorithms", *Algorithms*, vol. 2, no. 1, pp. 93–107, 2009. DOI: 10.3390/a2010093.

[42] K. Ho, "A survey of algorithms for star identification with low-cost star trackers", *Acta Astronautica*, vol. 73, pp. 156–163, 2012.

[43] G. Zhang, "Star identification", *National Defense Industry Press, Beijing*, 2011.

[44] D. Mortari, J. L. Junkins, and M. Samaan, "Lost-in-space pyramid algorithm for robust star pattern recognition", in *Guidance and control 2001*, 2001, pp. 49–68.

[45] S. Udomkesmalee, J. W. Alexander, and A. F. Tolivar, "Stochastic star identification", *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 6, pp. 1283–1286, 1994.

[46] C. Padgett and K. Kreutz-Delgado, "A grid algorithm for autonomous star identification", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 1, pp. 202–213, 1997.

[47] C. S. Lindsey, T. Lindblad, and A. J. Eide, "Method for star identification using neural networks", in *Applications and Science of Artificial Neural Networks III*, International Society for Optics and Photonics, vol. 3077, 1997, pp. 471–478.

[48] L. Li, F. Zhang, and T. Lin, "An all-sky autonomous star map identification algorithm based on genetic algorithm", *Opto-Electron Eng*, vol. 27, no. 5, pp. 15–18, 2000.

[49] F. G. Valdes, L. E. Campusano, J. D. Velasquez, and P. B. Stetson, "Focas automatic catalog matching algorithms", *Publications of the Astronomical Society of the Pacific*, vol. 107, no. 717, p. 1119, 1995.

[50] CelesTrak, *NORAD two-line element set format*, Available: http://www.celestrak.com/NORAD/documentation/tle-fmt.php, [Online]. Accessed on: April 13, 2023.

[51] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation", *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[52] J. L. Farrell, "Attitude determination by kalman filtering", *Automatica*, vol. 6, no. 3, pp. 419–430, 1970.

[53] P. Singla, J. L. Crassidis, and J. L. Junkins, "Spacecraft angular rate estimation algorithms for star tracker-based attitude determination", *Advances in the Astronautical Sciences*, vol. 114, pp. 1303–1316, 2003.

[54] P. Singla, D. T. Griffith, J. L. Crassidis, and J. L. Junkins, "Attitude determination and autonomous on-orbit calibration of star tracker for the GIFTS mission", *Advances in the Astronautical Sciences*, vol. 112, pp. 19–38, 2002.

[55] A. M. Rad, J. H. Nobari, and A. A. Nikkhah, "Optimal attitude and position determination by integration of INS, star tracker, and horizon sensor", *IEEE Aerospace and Electronic Systems Magazine*, vol. 29, no. 4, pp. 20–33, 2014.

[56] H.-b. Liu, J.-k. Yang, J.-q. Wang, J.-c. Tan, and X.-j. Li, "Star spot location estimation using kalman filter for star tracker", *Applied optics*, vol. 50, no. 12, pp. 1735–1744, 2011.

[57] H. Lee, Y.-H. Choi, H.-C. Bang, and J.-O. Park, "Kalman filtering for spacecraft attitude estimation by low-cost sensors", *International Journal of Aeronautical and Space Sciences*, vol. 9, no. 1, pp. 147–161, 2008.

[58] K. Xiong, T. Liang, and L. Yongjun, "Multiple model kalman filter for attitude determination of precision pointing spacecraft", *Acta Astronautica*, vol. 68, no. 7-8, pp. 843–852, 2011.

[59] J. Wang, X. He, Z.-h. Wei, Y. Lü, D.-l. He, Z.-y. Mu, J.-y. Ling, and Z.-l. Ma, "An attitude tracking method for star sensor under dynamic conditions", *Optoelectronics Letters*, vol. 15, no. 5, pp. 368–373, 2019.

[60] Y. Cheng and J. Crassidis, "Particle filtering for sequential spacecraft attitude estimation", in *AIAA guidance, navigation, and control conference and exhibit*, 2004, p. 5337.

[61] S. Haykin, *Kalman filtering and neural networks*. John Wiley & Sons, 2004, vol. 47.

[62] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, and S. Roweis, "Astrometry.net: Blind astrometric calibration of arbitrary astronomical images", *The astronomical journal*, vol. 139, no. 5, p. 1782, 2010.

[63] Astrometry.net, *Astrometry - astrometric calibration meta-data and more.* Available: `http://nova.astrometry.net/`, [Online]. Accessed on: April 13, 2023.

[64] A. Tennenbaum, *Open star tracker*, Available: `http://openstartracker.org/`, [Online]. Accessed on: April 13, 2023.

[65] A. Tennenbaum, "Automatic star-tracker optimization framework", *Proceedings of the AIAA/USU Conference on Small Satellites*, 2017.

[66] NASA, *Low cost star tracker software*, Available: `http://technology.nasa.gov/patent/TOP2-265`, [Online]. Accessed on: April 13, 2023.

[67] E. Bertin and S. Arnouts, "Sextractor: Software for source extraction", *Astronomy and Astrophysics Supplement Series*, vol. 117, no. 2, pp. 393–404, 1996.

[68] M. Richmond, *Match – a program for matching star list*, Available: `http://spiff.rit.edu/match/`, [Online]. Accessed on: April 13, 2023, 2012.

[69] E. Bertin, *Sextractor*, Available: `http://www.astromatic.net/software/sextractor`, [Online]. Accessed on: April 13, 2023, 2016.

[70] C. L. Lawson and R. J. Hanson, *Solving least squares problems*. SIAM, 1995.

[71] M. N. Sarvi, D. Abbasi-Moghadam, M. Abolghasemi, and H. Hoseini, "Design and implementation of a star-tracker for LEO satellite", *Optik*, p. 164 343, 2020.

[72] A. E. G. Inc, *Lumenera LW230/LW235 USB camera*, Available: http://aegis-elec.com/lumenera-lw230-lw235.html, [Online]. Accessed on: April 13, 2023.

[73] DesignSpark, *New raspberry pi camera module kits and lenses*, Available: http://www.rs-online.com/designspark/new-raspberry-pi-camera-module-kits-and-lenses, [Online]. Accessed on: April 13, 2023.

[74] M. L. Inc, *PT-LH023RPP lens holder for raspberry pi v2 camera*, Available: http://www.m12lenses.com/CNC-Machined-Raspberry-Pi-M12-Lens-Holder-Plastic-p/pt-lh023rpp.htm, [Online]. Accessed on: April 13, 2023.

[75] M. L. Inc, *Board lenses*, Available: http://www.m12lenses.com/Board-Lenses-s/12.htm, [Online]. Accessed on: April 13, 2023.

[76] F. Nelli, *Using canon and nikon lenses with raspberry pi*, Available: http://www.meccanismocomplesso.org/en/using-canon-and-nikon-lenses-with-raspberry-pi/, [Online]. Accessed on: April 13, 2023.

[77] R. P. Foundation, *Raspberry pi high quality camera*, Available: http://www.raspberrypi.org/products/raspberry-pi-high-quality-camera/, [Online]. Accessed on: April 13, 2023.

[78] R. P. Foundation, *Raspberry pi global shutter camera*, Available: http://www.raspberrypi.com/products/raspberry-pi-global-shutter-camera/, [Online]. Accessed on: April 13, 2023.

[79] N. Sako, Y. Tsuda, S. Ota, T. Eishima, T. Yamamoto, I. Ikeda, H. Ii, H. Yamamoto, H. Tanaka, A. Tanaka, *et al.*, "Cansat suborbital launch experiment-university educational space program using can sized pico-satellite", *Acta Astronautica*, vol. 48, no. 5-12, pp. 767–776, 2001.

[80] NASA, *Phonesat, the smartphone nanosatellite*, Available: http://www.nasa.gov/centers/ames/engineering/projects/phonesat.html, [Online]. Accessed on: April 13, 2023, 2013.

[81] G. Roux, "Augmented stellar sensor for a small spacecraft", http://hdl.handle.net/10019.1/106199 [Online]. Accessed on: April 13, 2023, Master's thesis, Stellenbosch University, 2019.

[82] A. Khorev and L. Torres, "Performance of a smartphone based star tracker", in *4th Interplanetary CubeSat Workshop*, South Kensington, London, United Kingdom, 2015.

[83] J. Critchley-Marrows and X. Wu, "Investigation into star tracker algorithms using smartphones with application to high-precision pointing cubesats", *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, pp. 327–332, 2019.

[84] NASA, *State-of-the-art of small spacecraft technology*, Available: http://www.nasa.gov/smallsat-institute/sst-soa, [Online]. Accessed on: April 13, 2023.

[85] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, C. Pernechele, and C. Dionisio, "A new star tracker concept for satellite attitude determination based on a multi-purpose panoramic camera", *Acta Astronautica*, vol. 140, pp. 166–175, 2017.

[86] K. Gudmundson, "Ground based attitude determination using a SWIR star tracker", http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1330406&dswid=1883 [Online]. Accessed on: April 13, 2023, Master's thesis, Linköping University, 2019.

[87] S. Scibelli, V. Ferrara, and F. Bernardini, "Low-cost stellar sensor for attitude control of small satellites", in *2019 PhotonIcs & Electromagnetics Research Symposium-Spring (PIERS-Spring)*, IEEE, 2019, pp. 3500–3506.

[88] M. Nixon, G. Day, J. Gould, B. Holden, J. Shrontz, J. Smith, E. Swinney, T. Taylor, and A. Webb, "Army cost efficient spaceflight research experiments and demonstrations attitude determination experiment", *Proceedings of the AIAA/USU Conference on Small Satellites*, 2017.

[89] A. Sreejith, J. Mathew, M. Sarpotdar, R. Mohan, A. Nayak, M. Safonova, and J. Murthy, "A raspberry pi-based attitude sensor", *Journal of Astronomical Instrumentation*, vol. 3, no. 02, p. 1 440 006, 2014.

[90] C. Underwood, S. Pellegrino, V. J. Lappas, C. P. Bridges, and J. Baker, "Using cubesat/micro-satellite technology to demonstrate the autonomous assembly of a reconfigurable space telescope (AAReST)", *Acta Astronautica*, vol. 114, pp. 112–122, 2015. DOI: 10.1016/j.actaastro.2015.04.008.

[91] D. Honess and O. Quinlan, "Astro pi: Running your code aboard the international space station", *Acta Astronautica*, 2017. DOI: 10.1016/j.actaastro.2017.05.023.

[92] Y. H. Lee, *Design & development of a prototype star tracker using raspberry pi*, http://hdl.handle.net/10356/71180, [Online]. Accessed on: April 13, 2023, 2017.

[93] S. F. Lopes, "Development of a low-cost star trackers for cubesats", http://hdl.handle.net/10451/38246 [Online]. Accessed on: April 13, 2023, Master's thesis, University of Lisbon, 2019.

[94] BBC, *Raspberry pi computer looks down on earth*, Available: http://www.bbc.com/news/science-environment-49584941, [Online]. Accessed on: April 13, 2023.

[95] ESA, *Types of orbits*, Available: http://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits, [Online]. Accessed on: April 13, 2023.

[96] NASA, *A Researcher's Guide to: Space Environmental Effects*, Available: http://www.nasa.gov/connect/ebooks/researchers_guide_space_environment_detail.html, [Online]. Accessed on: April 13, 2023.

[97] S. Samwel, "Low earth orbital atomic oxygen erosion effect on spacecraft materials", *Space Res. J*, vol. 7, no. 1, pp. 1–13, 2014.

[98] B. F. James, *The natural space environment: Effects on spacecraft*. National Aeronautics and Space Administration, Marshall Space Flight Center, 1994, vol. 1350.

[99] ROSAT Guest Observer Facility, *South atlantic anomaly*, Available: http://heasarc.gsfc.nasa.gov/docs/rosat/gallery/misc_saad.html, [Online]. Accessed on: April 13, 2023.

[100] B. Bhuva, "Control, alt, delete? the impact of space weather in the air and on the ground", in *2017 AAAS Annual Meeting (February 16-20, 2017)*, AAAS, 2017.

[101] NASA, *Radiation effects and analysis*, Available: https://radhome.gsfc.nasa.gov/top.htm, [Online]. Accessed on: April 13, 2023.

[102] N. Ya'acob, A. Zainudin, R. Magdugal, and N. F. Naim, "Mitigation of space radiation effects on satellites at low earth orbit (LEO)", in *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, IEEE, 2016, pp. 56–61.

[103] R. Ecoffet, "Overview of in-orbit radiation induced spacecraft anomalies", *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 1791–1815, 2013.

[104] D. Sinclair and J. Dyer, "Radiation effects and COTS parts in smallsats", *Proceedings of the AIAA/USU Conference on Small Satellites*, 2013.

[105] J. Mojica Decena, B. Wood, R. J. Martineau, M. Tayor, and J. Dennison, "Radiation damage threshold of satellite COTS components: Raspberry pi zero for OPAL cubesat", in *Utah State University Student Research Symposium*, 2018.

[106] C. Fuglesang, L. Narici, P. Picozza, and W. G. Sannita, "Phosphenes in low earth orbit: Survey responses from 59 astronauts", *Aviation, space, and environmental medicine*, vol. 77, no. 4, pp. 449–452, 2006.

[107] M. Sirianni, M. Mutchler, R. Gilliland, J. Biretta, and R. Lucas, "Radiation damage in hubble space telescope detectors", in *2007 IEEE Radiation Effects Data Workshop*, IEEE, 2007, pp. 9–15.

[108] A. Nagamatsu, K. Murakami, A. Yokota, J. Yamazaki, M. Yamauchi, K. Kitajo, H. Kumagai, and H. Tawara, "Space radiation damage to HDTV camera CCDs onboard the international space station", *Radiation Measurements*, vol. 46, no. 2, pp. 205–212, 2011.

[109] G. R. Hopkinson, "Radiation effects in a CMOS active pixel sensor", *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2480–2484, 2000.

[110] L. Pinheiro da Silva, G. Rolland, V. Lapeyrere, and M. Auvergne, "Radiation effects on space-based stellar photometry: Theoretical models and empirical results for corot space telescope", *Monthly Notices of the Royal Astronomical Society*, vol. 384, no. 4, pp. 1337–1343, 2008.

[111] G. H. Chapman, R. Thomas, R. Thomas, K. J. C. S. Meneses, T. Q. Yang, I. Koren, and Z. Koren, "Single event upsets and hot pixels in digital imagers", in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, IEEE, 2015, pp. 41–46.

[112] A. R. Smith, R. J. McDonald, D. C. Hurley, S. E. Holland, D. E. Groom, W. E. Brown, D. K. Gilmore, R. J. Stover, and M. Wei, "Radiation events in astronomical CCD images", in *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications III*, International Society for Optics and Photonics, vol. 4669, 2002, pp. 172–183.

[113] B. R. Hancock, R. C. Stirbl, T. J. Cunningham, B. Pain, C. J. Wrigley, and P. G. Ringold, "CMOS active pixel sensor specific performance effects on star tracker/imager position accuracy", in *Functional Integration of Opto-Electro-Mechanical Devices and Systems*, SPIE, vol. 4284, 2001, pp. 43–53.

[114] J. L. Jørgensen, T. Denver, M. Betto, and P. Van den Braembussche, "The PROBA satellite star tracker performance", *Acta Astronautica*, vol. 56, no. 1-2, pp. 153–159, 2005.

[115] ESA, *SPENVIS - The Space Environment Information System*, Available: `http://www.spenvis.oma.be/`, [Online]. Accessed on: April 18, 2023.

[116] A. D. Hands, K. A. Ryden, N. P. Meredith, S. A. Glauert, and R. B. Horne, "Radiation effects on satellites during extreme space weather events", *Space Weather*, vol. 16, no. 9, pp. 1216–1226, 2018.

[117] P. G. Van Dokkum, "Cosmic-ray rejection by laplacian edge detection", *Publications of the Astronomical Society of the Pacific*, vol. 113, no. 789, p. 1420, 2001.

[118] K. Zhang and J. S. Bloom, "Deepcr: Cosmic ray rejection with deep learning", *The Astrophysical Journal*, vol. 889, no. 1, p. 24, 2020.

[119] G. Wang, W. Lv, J. Li, and X. Wei, "False star filtering for star sensor based on angular distance tracking", *IEEE Access*, vol. 7, pp. 62 401–62 411, 2019.

[120] P. S. Foundation, *The python language reference, version 3.7*, Available: `https://docs.python.org/3.7/reference/`, [Online]. Accessed on: April 13, 2023.

[121] D. Hoffleit, *Catalogue of bright stars*. New Haven, Conn.: Yale University Observatory, 3rd rev.ed., 1964.

[122] D. Mink, "Wcstools 4.0: Building astrometry and catalogs into pipelines", in *Astronomical Data Analysis Software and Systems XV*, vol. 351, 2006, p. 204.

[123] W. M. Smart and R. Green, *Textbook on spherical astronomy*. Cambridge University Press, 1977.

[124] NASA, *STEREO website*, Available: http://stereo.gsfc.nasa.gov/, [Online]. Accessed on: April 13, 2023.

[125] C. Eyles, R. Harrison, C. J. Davis, N. Waltham, B. Shaughnessy, H. Mapson-Menard, D. Bewsher, S. Crothers, J. Davies, G. Simnett, *et al.*, "The heliospheric imagers onboard the STEREO mission", *Solar Physics*, vol. 254, no. 2, pp. 387–445, 2009.

[126] NASA, *STEREO SCIENCE CENTER - L0 Images*, Available: http://stereo-ssc.nascom.nasa.gov/pub/ins_data/secchi/L0/a/img/hi_1/, [Online]. Accessed on: April 13, 2023.

[127] NASA, *STEREO SCIENCE CENTER - L2 Images*, Available: http://stereo-ssc.nascom.nasa.gov/pub/ins_data/secchi_hi/L2/a/img/hi_1/, [Online]. Accessed on: April 13, 2023.

[128] U. S. S. D. Centre, *STEREO HI data processing documentation*, Available: http://www.ukssdc.ac.uk/solar/stereo/documentation/HI_processing.html, [Online]. Accessed on: April 13, 2023.

[129] T. Sun, F. Xing, X. Wang, Z. You, and D. Chu, "An accuracy measurement method for star trackers based on direct astronomic observation", *Scientific reports*, vol. 6, 2016. DOI: 10.1038/srep22593.

[130] N.-M. Inc, *Space simulation systems*, Available: http://www.nanomaster.com/spacesimulation.html, [Online]. Accessed on: April 13, 2023.

[131] C. E. Gonzalez, C. J. Rojas, A. Bergel, and M. A. Diaz, "An architecture-tracking approach to evaluate a modular and extensible flight software for cubesat nanosatellites", *IEEE Access*, vol. 7, pp. 126 409–126 429, 2019.

Annexes

Annex A

Extended abstract

This doctoral thesis focuses on developing and testing an ultra-low-cost and simple-to-use star tracker for attitude determination in nanosatellites. In addition, it embraces the upgrades and the applications derived from this development. This sensor is designed to be used in the satellite missions of SPEL (Space and Planetary Exploration Laboratory) and as a free-to-use tool for the CubeSat community of researchers and developers.

Absolute attitude estimation sensors provide high accuracy pointing capabilities to spacecraft, but as developed until today, they constitute an expensive fraction of CubeSat mission's budget. The cost skyrockets if a sub-arcminute precision is required. To solve this problem, I developed SOST (SPEL - Open Star Tracker), an ultra-low-cost attitude determination solution for satellite missions requiring low sampling rates. Due to its low cost, this platform fills a technological gap for CubeSats, a gap that considers high precision sensors, ultra-low-cost but low frequency. This Star Tracker development rests on open-source astronomy software, hardware from the Raspberry Pi family of microcomputers, and the Raspberry Pi camera (V2.1). Currently, using commercial-off-the-shelf (COTS) hardware as Star Tracker is a growing research area in which new works have been recently published, and a new research line is being consolidated.

To create SOST, I started by developing a new algorithm to solve the Lost-In-Space (LIS) problem that works by acquiring an image and comparing it with many stellar catalog segments. I tested this algorithm using images from the STEREO satellite. Also, the platform which holds this development was selected and tested in the following terms: the overall execution time, success rate, precision, power consumption, the capability of observing stars, and the performance inside a vacuum chamber. SOST provides a mean precision below one arcminute for the boresight direction 97.3% of the time in almost 12 seconds when a segment separation of 10 degrees is used. The success rate improved to 99.8% by using a finer grid of segments, but processing time increased to 20.8 s. The platform is free, available to CubeSat researchers, and has been used by parties interested in further development or deployment.

As SOST is part of the SUCHAI satellite missions, it is essential to demonstrate its integration into the nanosatellites. In terms of software, this integration is done within a framework that allows its operation while in space. In terms of hardware, the integration is

accomplished using the PC-104 standard. The main requirements and commands developed to make the payload work are presented, and a list of experiments to evaluate the performance of the STT while in orbit is proposed.

Finally, since the hostile ambient in LEO can damage the STT, particularly by different kinds of radiation, the behavior of the LIS algorithm under radiation effects that could appear in flight is studied and discussed.

Annex B

Code links

The open-source and ultra-low-cost STT platform is available on the GitHub page of the SPEL group:

`http://github.com/spel-uchile/Star_Tracker`

The robustness analysis under radiation events is open through the GitLab page of the SPEL group:

`http://gitlab.com/spel-uchile/testing-stt-robustness`.