IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 21, 2020, accepted August 17, 2020, date of publication August 28, 2020, date of current version September 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020048

Introducing SOST: An Ultra-Low-Cost Star Tracker Concept Based on a Raspberry Pi and Open-Source Astronomy Software

SAMUEL T. GUTIÉRREZ[1,2], CÉSAR I. FUENTES[3], AND MARCOS A. DÍAZ[1,2]

[1]Electrical Engineering Department, Faculty of Physical and Mathematical Sciences, University of Chile, Santiago 8370448, Chile
[2]Space and Planetary Exploration Laboratory (SPEL), Faculty of Physical and Mathematical Sciences, University of Chile, Santiago 8370448, Chile
[3]Astronomy Department, Faculty of Physical and Mathematical Sciences, University of Chile, Santiago 7591245, Chile

Corresponding author: Marcos A. Díaz (mdiazq@ing.uchile.cl)

This work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-18-1-0249, in part by the Comisión Nacional de Investigación Científica y Tecnològica (CONICYT) QUIMAL under grant 190004, in part by the CONICYT Fondo Nacional de Desarrollo Científico y Tecnológico (FONDECYT) under Grants 1211331 and 1151476, in part by the CONICYT Fondo de Equipamiento Científico y Tecnológico (FONDEQUIP) under Grant EQM150138, in part by the CONICYT Programa de Investigación Asociativa (PIA) under Grant ACT1405, in part by the BASAL Centro de Astrofisica y Tecnologias Afines (CATA) PFB-06/2007. F.F. and in part by the CONICYT-Programa Formación de Capital Humano Avanzado (PFCHA)/Doctorado Nacional/2017-21171862.

ABSTRACT Absolute attitude estimation sensors provide high accuracy pointing capabilities to spacecraft, but as developed thus far, they constitute a large fraction of CubeSat mission's budget. We introduce SPE Lab Open Star Tracker (SOST), an ultra-low-cost solution that currently provides sub-arcminute precision at a frequency of 1 to 3 estimations per minute in the Lost-In-Space scenario. Our Star Tracker development rests on open source astronomy software, the Raspberry Pi 3 B+, and its camera. We developed a new algorithm to solve the Lost-In-Space problem that works by acquiring an image and comparing it with different stellar catalog segments. We tested our algorithm using images from working satellites. The functioning of our platform was evaluated by using night-sky pictures taken from ground. We also conducted environmental tests of our platform by using a thermal-vacuum chamber. We optimized the catalog segment separation by analyzing the execution time, success rate, precision, and power consumption of the full platform. SOST delivers a mean precision below 1-arcminute for the boresight direction. With a segment separation of 10°, the attitude estimation is found in the 97.3% of the cases with processing time under 20s. The success rate improves to 99.8% by using 5° as segments separation but processing time doubles. This platform is open and freely available to CubeSat researchers interested in further development or deployment.

INDEX TERMS Attitude determination, CubeSat, low-cost hardware, nanosatellites, open-source, Raspberry Pi, star tracker.

I. 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. Among the attitude determination sensors such as sun sensors, horizon sensors, magnetometers, and inertial sensors, the Star Tracker (ST) stands out for providing absolute attitude estimation and being the most precise [1], [2]. Thus, the ST is a critical component in space missions.

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

Attitude sensors have been part of satellite missions since the dawn of the space era. The first CCD-based ST developed at JPL in 1976 [3] has been continuously improved over time. However, the improved 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. STs have followed this trend; therefore, they are generally available as commercial black boxes.

In the late 90s, the CubeSat emerged as a standardized satellite [4] to improve space technology education. CubeSats

TABLE 1. Different attitude determination sensors, with its main characteristics. This table shows that there are 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).

Attitude sensors	Typical precision	Frequency (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 (Proposed platform)	30" - 60" (cross-boresight)	0.05	1/10	<2.6	70
Star Tracker [21]	2" (cross-boresight)	10	1/4	<1	33000

have rapidly evolved and been actively used for research and commercial applications [5], [6]. CubeSat is a standardized shape platform with a cubical base unit of 10 cm × 10 cm × 10 cm (1U). The standardization reaches the dispenser unit in rockets supporting sizes of 1U, 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 [7], [8]. Due to the growing interest that CubeSats are gaining, it has become relevant to increase the capabilities of subsystems such as the ST [9]–[11] while keeping costs low. Many ST platforms have been tested, even using smartphones [12]. There is evidence that the use and development of sensors trough low-cost devices help to improve access to new applications and push education [13], technology development [14], and research [15], especially in developing countries.

Although several STs 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 gyroscopes or magnetometers, which cannot provide absolute attitude estimation, limiting the satellite capabilities, and then the achievable science and application. A full state-of-the-art report of different sensors and actuators for small spacecraft can be found in [16]. Table 1 summarizes the main characteristics of the most commonly used attitude determination sensors [17]–[23]. From the table, it is possible to notice a ''cost and technological'' gap between low and high precision attitude sensors, in particular for less demanding mission requirements.

The requirements of some of the new SUCHAI (Satellite of the University of Chile for Aerospace Investigation) missions fall under the category of high precision absolute attitude measurement at a lower frequency where the cost of commercial ST is not justified. Space and Planetary Exploration Laboratory (SPEL) at the University of Chile has used the CubeSat standard to develop a space program for education and research [8]. The first CubeSat developed in SPEL (SUCHAI 1), was launched on June 23, 2017, from India, carrying a series of ''proof of concept'' experiments. The SUCHAI 1 was a 1U CubeSat without attitude control. Continuing with the satellite program, SPEL is now developing three 3U CubeSats, where at least two of them are planned to work collaboratively. These new satellites will 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 (1 to 3 samples per minute).

One of the payloads in our coming missions that motivated the development of our ST is a fixed-voltage flat-probe Langmuir Probe (LP). The flat-probe (or patch) of the LP will be located in 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 we use a negative potential and we measure ion density to avoid the charging potential issue of the spacecraft. The proper estimation requires that the patch normal vector ($\hat{n}$) is parallel but against the spacecraft ram velocity ($\hat{v}$). 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 way 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 frequency 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 properly designed. Similarly to the LP, the phased array experiment does not require a high frequency when estimating attitude, since the feasible number of elements of the array in a 3U CubeSat produces a half-power beam-width of 20°.

In this article, we introduce our high-precision open ST platform as a new ultra-low-cost alternative to the devices currently available in the market. We developed it by combining commercial miniaturized computers with open software for astronomy. The developed algorithm is intended to function in multiple hardware platforms. We demonstrate in this work that the algorithm can provide attitude estimation,

with simple adaptations, from images taken and processed over multi-platforms. Our proposed platform is named *SPEL - Open Star Tracker* (**SOST**).

To get a low-cost and easy to implement attitude sensor, we adopted a small single-board-computer as hardware platform. The proposed sensor platform is based on a popular miniaturized computer, the Raspberry Pi (RPi) together with its simple camera, which is easily integrated into this platform. Similar cameras, with a large field of view (FOV), have been previously studied for ST applications [24]. The RPi was selected because of its low cost and processing capability, which is enough to serve as ST as we will show in this study. It is simple to operate and it is currently being used as an attitude sensor [25] and in a variety of space projects [26], [27]. Recently, the RPi Zero has been tested in space, taking a video of Earth [28]. Another advantage of the RPi platform is that it is mostly open hardware, which means it can be adapted and modified to eventually fit in the CubeSat standard.

ST needs to be robust and hardware-independent. To guarantee the robustness of our ST we follow a multi-disciplinary approach by taking advantage of astronomy experience in the field. Thus, we based our algorithm on open software tools commonly used by astronomy students and researchers, *Source Extractor* [29] and *Match* [30]. The open characteristics guarantees proper operation through many contributions and independent verification uses. These two software tools are based on powerful, optimized algorithms for star detection and identification. In addition, these programs can run on diverse hardware platforms, assuring hardware-independence of the attitude estimation firmware. As suggested by [31], hardware independence is a key characteristic of the firmware to prevent obsolescence of software with hardware changes.

Besides, the development of SOST as an open platform can facilitate the contribution of multiple groups, accelerating in this manner the features and the flight heritage of the system. Open source platforms enable better interaction with a wider number of experts that may provide a faster, successful outcome. This multi collaboration approach has already been used in CubeSats [32], [33]. Through this approach, we envision that smaller actors from groups like ours can contribute to the development of a more precise and robust ST platform. The paper is organized in the following manner: First, we describe our platform (software and hardware) and the algorithm. Then, we evaluate both the algorithm with pictures from a working satellite and the full platform with ground-based experiments. Finally, we discuss the results and present avenues to improve the platform.

II. PLATFORM DESCRIPTION

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. The SOST system requires a hardware platform, a software suite, and an algorithm that can work together to fulfill the mission's requirements.

A. SOFTWARE AND HARDWARE DESCRIPTION

1) SOFTWARE

Regarding the software, we worked with frequently used programs within the field of Astronomy. The main code was developed in *Python*. It calls the following software suites:

Source Extractor:

Software used to detect astronomical sources (e.g. stars, planets, galaxies) in a ''.fits'' image. In our work, *Source Extractor* is used 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. In this work, we use version **2.19.5** of the software. *Source Extractor* can be downloaded for free from its website. Further information can be found in [34].

Match:

Software used to establish a relationship between two different lists of objects. The specific algorithm with which *Match* operates is based on [35]. To properly operate, *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). For the image, it represents the brightness relative to the background of the image.

One file corresponds to the list of sources extracted from the image. The other file is a specific segment of the star catalog. In our platform, *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. The linear relationship is used in this study since it involves the shortest calculation time. 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 explained in more detail in subsection II-B1. The linear relationship between the two data lists is defined by 6 coefficients (a, b, c, d, e, and f) delivered by *Match*. The relationship can be written as:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a \\ d \end{pmatrix} + \begin{pmatrix} b & c \\ e & f \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}, \tag{1}$$

where point (x, y) represents 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 catalog list. Additionally, *Match* delivers several statistical parameters that define the precision of the established relationship. The most significant statistical parameters used in our work are:

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

TABLE 2. Specifications of RPi 3B+ and RPi Zero W.

Element	RPi 3B+	RPi Zero W
CPU	1.4 GHz 64-bit quad-core ARMv8	1 GHz single-core ARM1176JZF-S
Memory (SDRAM)	1 GB	512 MB
Weight	45 g	9 g

Nr: 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, we used version **0.14**.

2) HARDWARE

Regarding the hardware of the platform, we used the following components:

1) Raspberry Pi 3 Model B+.
2) Raspberry Pi Zero W.
3) Raspberry Pi Camera V2.1.

These are commercial-off-the-shelf (COTS) items available on the market at a cost of approximately US $35 (or less) each one. The main technical characteristics of the RPi and the camera are summarized in Table 2 and Table 3, respectively.

TABLE 3. Specifications of Raspberry Pi camera V2.1.

Element	Value
Sensor Type	SONY IMX219PQ Color CMOS 8MPix
Sensor Size	3.674 mm x 2.760 mm
Full FOV	62.2 degrees x 48.8 degrees
Weight	3 g

B. ALGORITHM DESCRIPTION

There are several algorithms proposed to implement a ST. A summary of algorithms is presented in [36]. In [37] a summary is presented specifically for low-cost ST. The algorithm developed in this work solves the Lost-In-Space (LIS) problem, to make it comparable to the previous works. Also, our algorithm assumes that the spin rate of the spacecraft (at each axis) is less than a revolution per hour. A spin rate above this threshold can be detected by using other attitude determination sensors or even using other algorithms without going through the star identification process [38]. Then, the spin rate can be reduced by the attitude control unit. The proposed algorithm uses both star brightness information and the segmentation of the stellar catalog.

The 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, which is 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 of the FOV of the image to properly use *Match*. The concept of splitting the catalog into segments has been proposed before for STs [39]. 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 the sky. This procedure is also called camera calibration. In this section, we describe with more detail this procedure.

1) TANGENT PLANE PROJECTION OF THE STELLAR CATALOG

Our pointing precision should be determined by our images, not by the precision of the stellar catalog we use 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, we chose the Bright Star Catalogue (BSC) [40] from Yale University Observatory. The BSC was obtained using the *scat* application, part of the WCSTools [41] software package. The BSC catalog contains 9,110 objects, 9,096 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. To relate both systems, it is necessary to project the stars into the tangent plane 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 [42], this is achieved according to the equations:

$$\eta = \frac{\cos(\mathrm{DEC}) - \cot(\delta)\sin(\mathrm{DEC})\cos(\alpha - \mathrm{RA})}{\sin(\mathrm{DEC}) + \cot(\delta)\cos(\mathrm{DEC})\cos(\alpha - \mathrm{RA})}, \quad (2)$$

$$\xi = \frac{\cot(\delta)\sin(\alpha - \mathrm{RA})}{\sin(\mathrm{DEC}) + \cot(\delta)\cos(\mathrm{DEC})\cos(\alpha - \mathrm{RA})}, \quad (3)$$

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. For a visual interpretation in the use of the previous equations, the reader could refer to Fig. 107 in [42].

In our implementation, we 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 properly operate. For this reason, the catalog was stored in memory but divided into overlapping segments of $60^\circ \times 60^\circ$ (a slightly larger area than the FOV of the 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, we tested different distances between the centers of each projected segment, in RA and DEC. Thus, the segment (i, j) is the one with center (tangent plane point) $\text{RA}_{segment\ i,j} = i°$ and $\text{DEC}_{segment\ i,j} = j°$ with limits of the segment $\text{RA}_{segment\ i,j} \in [(i - 30)°, (i + 30)°]$ and $\text{DEC}_{segment\ i,j} \in [(j-30)°, (j+30)°]$, where $i = 0°, \ldots, 359°$ and $j = -90°, \ldots, 90°$. We considered three different segmentation patterns for our catalog, where we advance i and j by 5°, 10°, and 15°. These yield different numbers of segments to project and compare.

2) ATTITUDE DETERMINATION ALGORITHM

To determine the attitude of the camera, the following procedure is carried out:

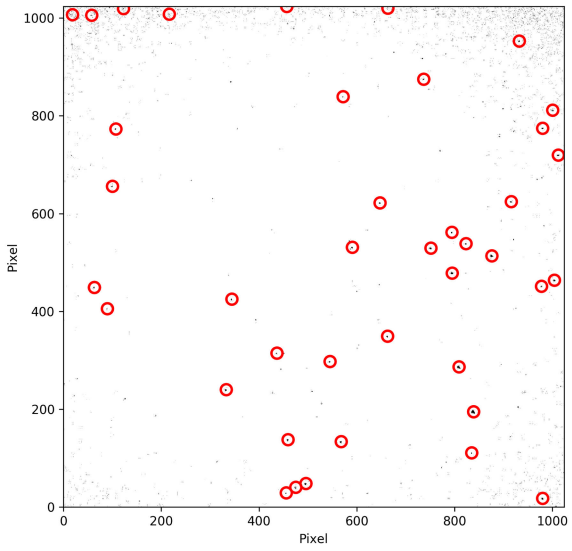
1) **Acquire an image.** The camera, with predefined exposure time, takes a picture of the sky. The exposure time was evaluated and configured with the RPi camera, presented in subsection III-B1.
2) **Generate the list of sources from the image.** The list of brightest objects in the picture is obtained by using *Source Extractor*:
   a) The ''.jpg'' image is transformed into ''.fits'' format, the defined format of *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 delivered 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 subsection III-B1.
   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. Fig. 1 (a) 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.
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 our tests, it is usually achieved with 18 objects (stars) or more. This 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 arbitrary selected in this stage (e.g. 5 degrees). We studied the effects of different distances between catalog centers in subsection III-B2. 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 1). At this stage, the projected catalog does not perfectly agree with the image. We use the relationship delivered by *Match* to find a new projection center to improve the matching procedure in the next stages. The result of this iteration is shown in Fig. 1(b).
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 Fig. 1(c). Fig. 1(d) 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 Fig. 1(a) are 192.88°, −45.10°, and −98.76°, respectively.

The process described above can be summarized in the flowchart shown in Fig. 2.

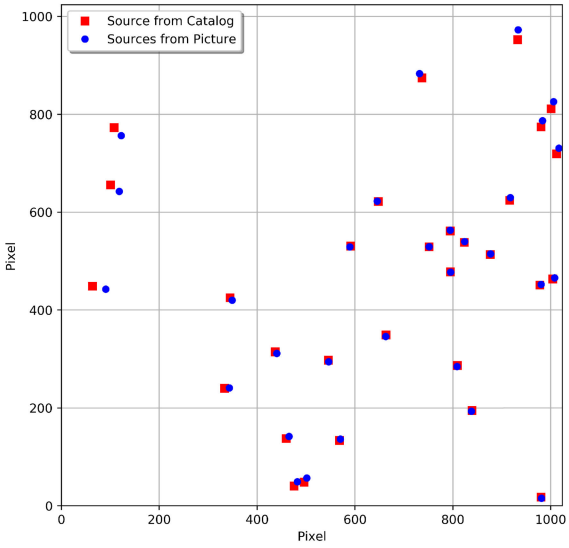
3) CONFIGURATION OF *MATCH* PARAMETERS

Match is a powerful and flexible software that finds the spatial transformation that matches most sources between two lists of objects. Its behavior is affected by several parameters, which need to be defined for the proper operation of the algorithm. In this subsection, we describe the main parameters used and the values given to it for its operation.

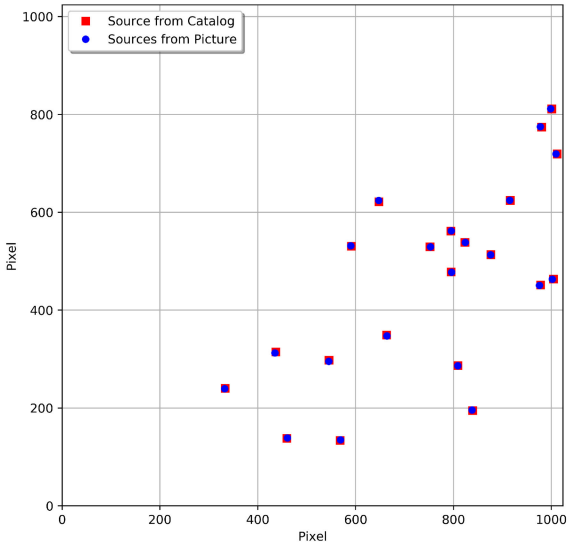
trirad: This parameter defines when the triangles formed between the objects of each list are considered a



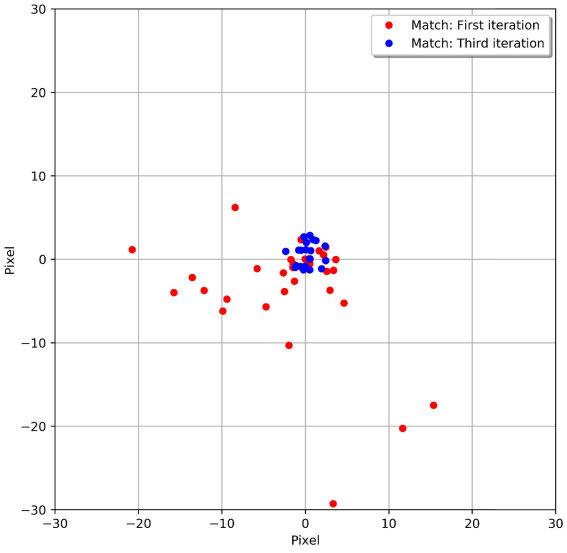
(a) Original Image plus extractions.



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



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



(d) Differences (in pixels) between pairs.

FIGURE 1. (a) shows the 40 brightest detections extracted from the picture. (b) shows the first iteration result. It shows the objects from catalog and sky picture. Fig. (c) shows the result of the third (final) iteration. (d) is the difference (in pixels) between pairs of matched objects of first and third iteration. First iteration data (30 matched objects) is shown in red and the third iteration (21 matched objects) in blue. Dispersion in the third iteration is less than that obtained in the first iteration.

match. This parameter was left in its default value which is 0.002. Modifications of this parameter produce no significant change in the estimation result unless its value is increased sharply (greater than 0.015), where no matches were found.

nobj: It defines the number of brightest objects in each list that are considered when searching for a match. For values smaller than 10, the found relationship between lists is not reliable. For values greater than 40, it takes a long time to find a match. For our software, we set this parameter to 15 for the first iteration, and it is increased to 20 for the second and third iteration.

max_iter: It defines the number of cycles that the program uses to search for matches between objects on both lists. A value greater than 5 increases the execution time. For the first iteration, this parameter is set to 1 and for the second and third iteration, it is increased to 5.

matchrad: It defines the minimum distance to which two objects from both lists must be located to be counted as a match. If this value is larger than 1, the linear model

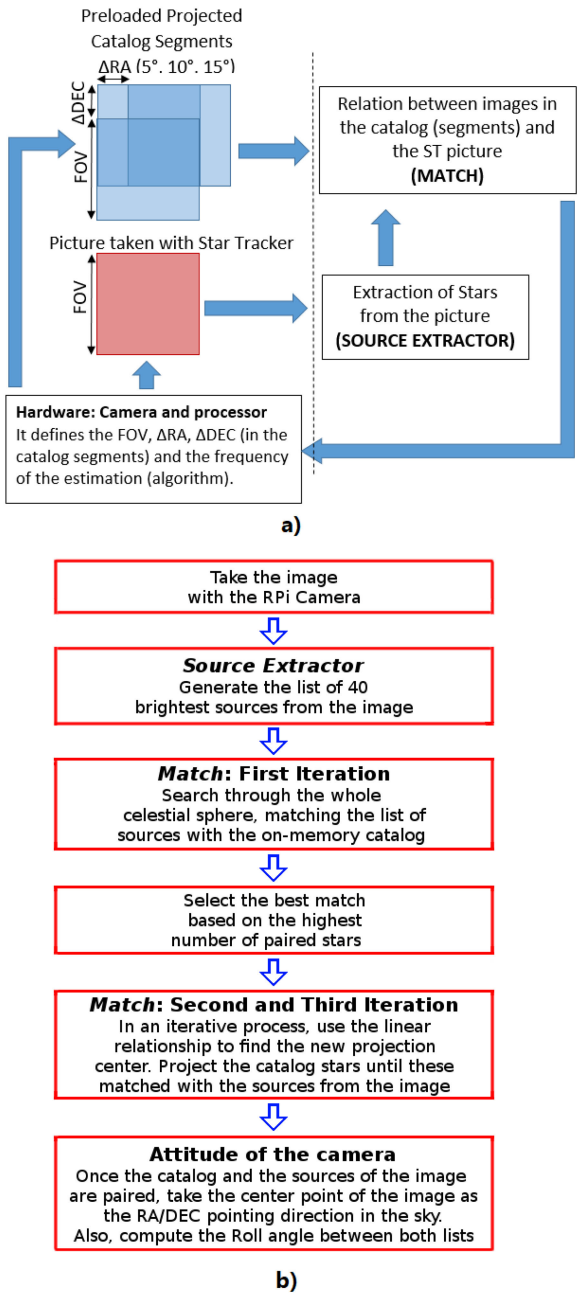


FIGURE 2. a) shows a scheme of the algorithm architecture based on catalog segmentation. It presents what is developed in this work (left side of the dashed line) and what comes form other sources (right side of dashed line). The right side can be easily changed to customized tools if preferred. It also shows the independence of the tools selected (hardware and software). b) presents a flowchart of attitude determination algorithm (See subsection II-B2). It shows the iterative process which is a novel approach to improve accuracy while being independent of the pre-defined/pre-loaded segments centers.

between lists is of poor quality due to a large number of objects used in the relationship as a match. If its value is less than 0.01, the linear model between lists can also be inaccurate due to the small number of objects used to establish the relation. We set an intermediate value of 0.1.

scale: It defines the scale relationship that exists between lists. In our work, both lists are compared as if they were stars observed in the CMOS sensor of the camera. Since we have made every task to both data lists correspond to each other (using the procedure described in section II-B1), this value is set to 1.

III. EVALUATION

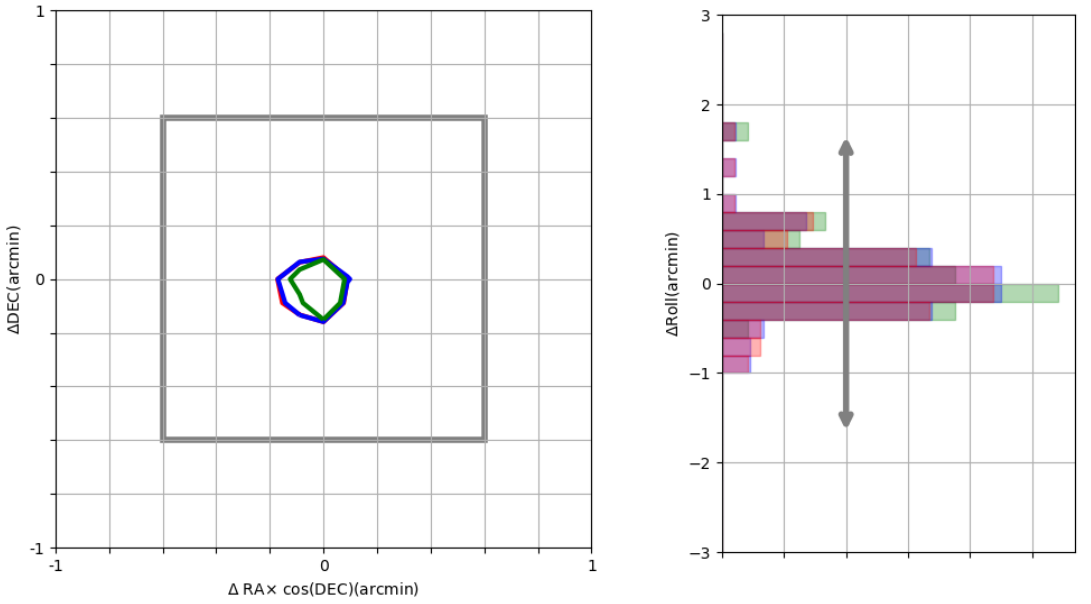
We present the results of the full evaluation of our proposed ST. First, we evaluate the algorithm with pictures coming from a working satellite (STEREO mission). The main idea of this evaluation is to isolate the hardware from the platform to focus only on the results given by the algorithm. Secondly, we fully evaluate the proposed platform (RPi plus camera).

A. ALGORITHM EVALUATION BY USING ON-SPACE IMAGES (FROM STEREO MISSION)

STEREO (Solar TErrestrial RElations Observatory) [43] 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). We use images from STEREO, specifically from the HI-1 telescope [44]. This telescope observes the space between the Earth and the Sun. There are hundreds of stars in each image, making them suitable to test our algorithm.

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

1) Get catalog patches similar in size to the FOV of the HI-1 detector (20° × 20°). We consider stars from the BSC catalog until magnitude six, this implies a total of 5080 catalog stars. We tested the algorithm 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 [45]. We select random images from the year 2007 to the year 2010 from the observatory STEREO A. Attitude is computed from this HI Level 0 images using our proposed 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 [46]. 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 our algorithm and the header data of every image are shown in Fig. 3, for different catalog segmentation. The success rate given by the algorithm in this evaluation is 96%, 89%, and 48% for a 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 3. 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 our algorithm. Fig. (a) show 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. Fig. (b) show 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.

B. PLATFORM EVALUATION

ST 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 mimic the in-orbit conditions in a laboratory capable to put to test the attitude estimation performance of the ST. Some evaluation methods use artificial stars in the laboratory to solve this problem. Instead, we follow a similar approach to that suggested by [47], which uses real astronomical sources from ground.

Thus, we tested our platform's performance with images captured by the RPi camera by 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, pointing at different parts of the sky. The experimental setup used for this task is shown in Fig. 5(a). The attitude of the camera for every captured image was computed using three different processors: RPi 3 B+, RPi Zero W, and a personal computer. We used different:

1) Exposure time, and
2) Star catalog segmentation: number of catalog segments, size of the segments, and the distance between segments centers (segment overlapping),

to study the effects on:

1) Precision,
2) Success rate,
3) Processing time, and
4) Power consumption.

In addition, we also evaluated the response of the SOST platform to environmental conditions similar to those found in LEO orbit (Thermal-vacuum test).

1) EXPOSURE TIME IMPACT

The quality of the image directly affects the number of stars that are extracted from it. Our algorithm expects to match tens of stars from the picture with their catalog sky positions to converge to a solution (a success). The exposure time of the picture will set the number of stars that will be detected on average. We used different exposure times to find the minimum one that yields enough number of extractions that produce reliable matches with catalog stars.

We photographed different regions of the sky, with exposure time ranging from 0.1 to 2 seconds at intervals of 0.1 seconds. The camera was configured to produce 1024 pixels × 1024 pixels images 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 that time there is no significant improvement in the image quality,

then the number of real extracted sources remains constant. Although the algorithm converges with images with an exposure time of 0.6 seconds, we 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 images.

Regarding the matched stars, we empirically found that with $\approx$ 20 stars we could get a good fit for the pixel-to-sky transformation. For this reason, we selected the 40 brightest detections in the image (see Fig. 1(a)) as a limit. More extractions provide only noise from the image. This criterion selects catalog stars brighter than magnitude 4, reducing the astronomical sources from the BSC catalog to 518 stars. Thus, the exposure time impact mostly on processing time, because the precision and success rate are independent of exposure time over 0.9 seconds.

2) CATALOG SEGMENTATION IMPACT

In our 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 our algorithm. For instance, we 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, which would break 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).

We divided the catalog in a set of projected segments to be compared with the image. The number of catalog segments is directly related to the success rate that we can obtain with our algorithm. We tested the influence that catalog segmentation on the algorithm has on the 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 images 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 to find a pointing solution in every image.
2) Run the ST algorithm over every image, with the center of the catalog patches separated by 5°, 10°, and 15°.
3) For each catalog segmentation, we change the catalog starting point in (RA, DEC) in the matching process. We start in (0, 0) degrees and increase one by one at each coordinate, sweeping all possible integer values until $([p-1]^\circ,[p-1]^\circ)$, with $p = 5^\circ$, 10°, and 15°.

For the algorithm a **successful estimation** is achieved when software *Match* finds a valid relationship between the selected catalog segment and an image's star list. We regard a ''valid relationship'' is achieved when the position of at least 15 objects can be transformed between lists with a standard deviation σ lower than the size of a pixel. On the other hand, when *Match* is not capable of finding and delivering a valid relationship, the algorithm fails in achieving an attitude estimation. Thus, the **success rate** was computed as follows: For a given catalog segmentation, and a specific starting point of the catalog, we count the times that the algorithm gives a valid pointing solution, for the selected subset of images. After that, we average the success rate over different starting points, for every catalog segmentation. The result is shown in Table 4.

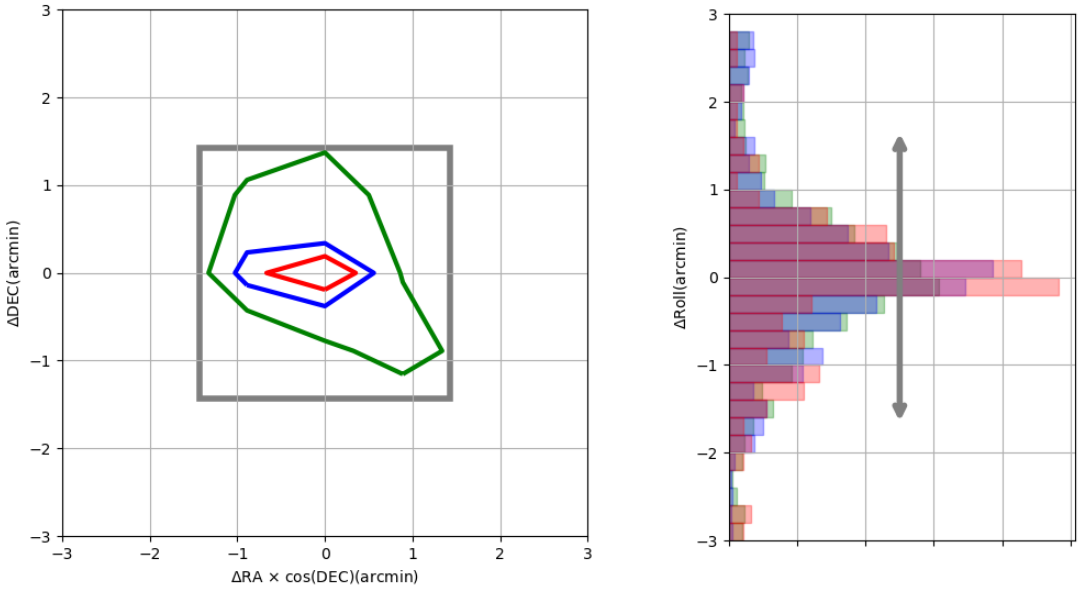
TABLE 4. Success rate and the number of related objects between lists, for different catalog segmentations.

	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

The number of related objects between lists was computed as follows: For a given catalog segmentation, and a specific starting point of the catalog, we save the number of matched objects between picture and catalog, for the selected subset of images. After that, we average the number of objects over all the different starting points, for every catalog segmentation. The result is shown in Table 4. The uncertainties show the standard deviation of the success rate and the number of related objects when we change the starting point of the catalog segmentation.

Regarding the precision in the pointing coordinates (Right Ascension, Declination, and Roll), it is important to notice that for this type of experiment, contrary to the test with STEREO images, we do not know the ''true value'' of the attitude solution in the selected subset of pictures. For that reason, our analysis aims to get an idea of the precision, not the accuracy, of our proposed platform. In that way, we assume that the segmentation pattern at 5° brings an attitude solution near the ''true value''. For that reason, we computed an assumed ''true value'' based on the average of the pointing solution obtained for different starting point 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. 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.

Besides the catalog segmentation and the pointing precision, the processing time of an ST is a relevant variable to assess its performance. This settles the type of application or science that can be achieved with it. Processing time depends on both the algorithm and the hardware. We evaluated the algorithm using the *multiprocessing* package from *Python*,



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

FIGURE 4. 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. Fig. (a) show 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 our images. Fig. (b) show 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.

taking advantage of the multi-core capability of the RPi 3 B+. We use this multiprocessing capability specifically in the *Match* routine of the algorithm. Table 5 summarizes the time taken by each process of the attitude determination of the ST, 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*.

TABLE 5. Processing time of each major routine in Raspberry Pi 3 B+. These numbers are for a 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.

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

3) VACUUM CHAMBER TEST

To partially simulate the spatial environment and its effects on the RPi, we tested the RPi 3 B+ and its camera inside a vacuum chamber. The vacuum chamber system is an NDT-4000 from Nano-Master Inc [48]. 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.
2) Run the ST 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.

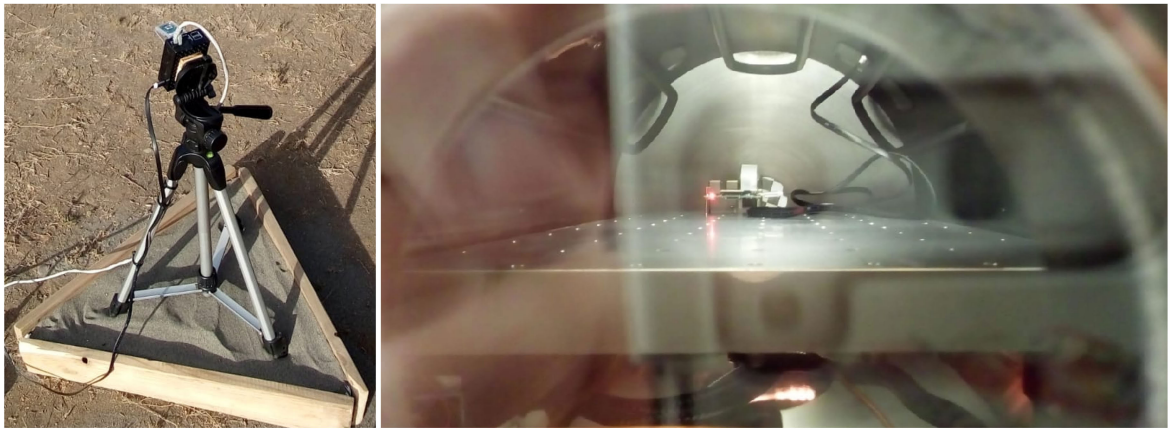
This test was performed for twelve hours. The chamber remained at a nearly constant temperature of 27°C, and the minimum pressure reached was $8.89 \cdot 10^{-4}$ Pa ($6.67 \cdot 10^{-6}$ Torr). An image of the RPi in the vacuum chamber test is shown in Fig. 5(b). During the test, the RPi and the camera worked normally. It showed no sign of failure, and the taken images were not distorted. Also, the temperature of the RPi-core did not overcome the 30°C.

4) POWER CONSUMPTION

The power consumption is an important variable of ST, which can define the feasibility of being used in CubeSat missions. The power consumption of the ST platform during the operation is summarized in Table 6. It shows that in the current stage the ST system 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.

IV. DISCUSSION

The proposed ST (SOST) is based on two open software suites commonly used in astronomy and an educational



(a) On-ground evaluation of SOST platform (b) Environmental evaluation of SOST platform inside a thermal vacuum chamber

FIGURE 5. (a) 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). (b) 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).

TABLE 6. Power consumption in each ST stage within Raspberry Pi 3 B+.

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

hardware platform popular among educators and embedded systems hobbyist. We have developed an attitude estimation algorithm to solve the Lost-In-Space problem. The problem we are solving assumes that the satellite was stabilized by some means. For instance, if the satellite is spinning fast, other attitude sensors such as an IMU can detect that movement. These sensors provide information to the attitude control unit which compensates that rotation by using some actuators. The expected performance of SOST is based on the previously shown results, which will be further discussed in the coming subsections.

A. SOST PRECISION

Our proposed ST was tested from the ground, enabling us to detect Earth's rotation, which is close to 15 arcseconds per second. We did not perform tests with angular rates below that value. 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 III-B1) is 0.8 seconds.

We do not expect a precision better than 12 arcseconds (15 arcseconds/s × 0.8 s) with our 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. 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, we include iterations 2 and 3 in the algorithm.

Besides the empirical approach we have previously described to estimate SOST's precision, we also consider theoretical methods previously proposed [2] that allow us to find bounds to the features of the ST. For instance, the average number of stars in a circular FOV with A degrees wide, or N_{FOV}, can be obtained by using the parameters of our system: A = 48.8° (FOV of the RPi camera), $N_{M_4} = 518$ (number of stars in the BSC catalog of magnitude 4 or less), and $N_{\text{pixel}} = 1024$ (the number of pixels in RPi camera):

$$N_{\text{FOV}} = N_{M_4} \cdot \frac{1 - \cos(\frac{\text{A}}{2})}{2} = 518 \cdot \frac{1 - \cos(24.4)}{2} = 23 \text{ stars} \quad (4)$$

Using the obtained value ($N_{\text{FOV}} = 23$ stars), we can find the bounds for the error in the cross-boresight direction ($E_{\text{cross-boresight}}$) and for the error in the roll direction (E_{roll}) of the SOST. We assume that the centroiding accuracy is $E_{\text{centroid}} = 1$ pixel since we cannot use the hyperacuity technique (defocusing) with the RPi camera:

$$E_{\text{cross-boresight}} = \frac{\text{A} \cdot E_{\text{centroid}}}{N_{\text{pixel}} \cdot \sqrt{N_{\text{FOV}}}} = \frac{48.8 \cdot 1}{1024 \cdot \sqrt{23}} = 36 \text{ arcseconds} \quad (5)$$

$$E_{\text{roll}} = \arctan\left(\frac{E_{\text{centroid}}}{0.3825 \cdot N_{\text{pixel}}}\right) \cdot \frac{1}{\sqrt{N_{\text{FOV}}}} = \arctan\left(\frac{1}{0.3825 \cdot 1024}\right) \cdot \frac{1}{\sqrt{23}} = 1.8 \text{ arcminutes} \quad (6)$$

It is worth noting the similarity of these results to those obtained in our ground-based night-sky test, detailed in section III-B2 and shown in Fig. 4.

B. TIME OF THE ESTIMATION PROCESS

The evaluation (at ground level) of the ST shows that processing time is the critical feature to improve. For this reason, we evaluated the impact of using a faster processor (a desktop PC). 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 7. The results consistently show that the processes are performed faster with more powerful processors.

TABLE 7. 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 IV-B. Due to the different catalog segmentation, there are various time measurements for the matching routine.

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

C. POWER CONSUMPTION OF THE RASPBERRY PI

The power consumption of the ST routine in the RPi 3 B+ was shown in Table 6. 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, we also try the ST software in related hardware, the RPi Zero W. The main features of the RPi Zero are shown in Table 2.

The power consumption in the idle state was 0.5 W on average. While the ST routine is running on, the average power consumption is 1.0 W. Nonetheless, due to the lower computing capabilities, the times of the ST routine increased. Table 8 shows the time of each stage of the ST routine on the Raspberry Pi Zero W.

D. ABOUT THE DEVELOPED OPEN PLATFORM

We developed an ST platform, open to everyone interested in a simple and easy-to-use ST for CubeSats. Our ST platform and test images can be downloaded from `https://github.com/spel-uchile/Star_Tracker`. We hope that our platform might be helpful for the development of an ST in new small groups of CubeSat developers, and for experienced developers who need a simple and fast ST solution. We encourage readers to install our software. It can be installed on a Linux-based personal computer or an RPi plus camera platform, to test it and also reproduce all the shown results in this work.

TABLE 8. Processing time of each major routine in Raspberry Pi Zero 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.

Process	Average time (s)		
	5°	10°	15°
Importing libraries		26.2 ± 0.3	
Picture acquisition		8.73 ± 0.03	
Source extraction		8.7 ± 0.6	
Catalog matching	114 ± 3	32 ± 1	17.8 ± 0.6
Total	158 ± 3	76 ± 1	61.4 ± 0.6

V. CONCLUSION

We present a ready to use ultra-low-cost open star tracker platform suitable for medium Cubesats, which is based on a quasi-open hardware platform and a algorithm developed as an open software. Although, the platform rests on the integration of existing tools the proposed implementation presents multiple advantages that might improve Cubesat missions. To the best of our knowledge, there is no such system in the CubeSat community. This platform is easy to implement and has been conceived for payloads and/or experiments that require precision in the attitude determination but at low frequency (1 to 3 samples per minute solving the LIS problem). The system is designed to operate under the restrictions of mass, volume, and energy of a 3U CubeSat, however, it can be adapted to work in other platforms.

The algorithm uses a catalog segmentation process to facilitate the independence of the selected hardware. The segmentation approach also facilitates the independence of the tools used to extract objects of images and for the searching-in-the-catalog process for the best match with the image (see Fig. 2 a)). Thus, personalized tools could be used and exchanged easily for these tasks. It also uses an iterative approach to improve the accuracy of the attitude estimation, which makes the estimation independent of the center-location on each preloaded segment (see Fig. 2 b)). To the best of our knowledge, these approaches (segmentation and iterations) are also a novel approach in the field. In the current algorithm, we use two open software suites commonly used in astronomy (as shown to the right of the dashed line in Fig. 2 a)): *Source extractor* (for extracting objects/stars from star tracker images) and *Match* (for finding a linear relationship between images and preloaded catalog segments). The use of these tools makes the algorithm robust since their performances are well known and tested within

the astronomy community. The use of these tools can also facilitate improvements since they are open source tools that run over multiple platforms. This also makes the algorithm independent of the selected hardware.

To evaluate the algorithm performance, we proposed a novel procedure that uses images from the HI-1 telescope of STEREO mission. These images come not only with the objects/stars but also with attitude information, which were used for verification of the attitude estimation obtained with our algorithm (see section III-A). This evaluation process also shows that the proposed algorithm is easily adapted to other hardware platforms. In addition, this analysis shows that if the algorithm is used on an optimized hardware, the accuracy of the estimation is independent of the inter-segments overlapping. However, it also shows that the accuracy and the processing time are not the only relevant figure of merits to evaluate the algorithm performance. The success rate is also key in the evaluation. Since the accuracy does not change with inter-segments overlapping, the attitude estimation can be accelerated if less overlapping among segments is used. However, the use of less inter-segments overlapping shows a reduction in the probability of getting a valid attitude estimation per image or success rate (see Table 4). The rate of success is rarely reported by commercial STs, which might lead to partial evaluation of the STs performance.

Finally, it is proposed the use of an ST hardware based on the Raspberry Pi 3 B+ platform and its camera. We also showed that the Raspberry Pi Zero can also be used, although doubling the processing times. The use of the Raspberry Pi hardware dramatically reduces the cost of the ST (at least two orders of magnitude from the current commercial options). This critical reduction in cost, together with the development of the algorithm as an open software tool, promise not only a rapid improvement of the system, by allowing contributions from users, but also by accelerating its robustness in space if it is more frequently used in Cubesat missions.

The average precision reached with this version of the platform is in the order of $\sim$ 30 arcseconds (Figure 4) solving the Lost-In-Space problem in about 40 seconds (Table 5), with a success rate of 99.8% (Table 4), when using a catalog segmentation of 5°. Nevertheless, a similar pointing accuracy can be reached in about 20 seconds, with a success rate of 97.3% by using a catalog segmentation of 10° (Table 4). These processing times include loading *Python* libraries, taking a picture of the sky, extracting bright stars, finding the correct segment of the sky catalog, and estimating the attitude of the camera.

Although the platform has not been yet evaluated in space it has been fully tested on laboratory for launch (vibration and bake-out) and operational (thermal-vacuum) conditions. The attitude estimation of the SOST platform was also tested at ground campaigns, taking pictures of different areas of night-sky, as suggested in previous works (see section III-B). To the best of our knowledge, SOST is the first approximation to an open ST platform, and it is integrated into the coming SUCHAI-2 and -3 missions [8] for evaluation in space. Nevertheless, it can be subjected to further optimizations. For instance, it might be necessary to include environmental considerations in the platform, in particular to evaluate and compensate the effect of radiation. Optical aberrations could be quantified and corrected if needed [49]. These conditions and their effects can be mitigated with the aid of mechanical subsystems and compensations in the algorithm [50]. Also, passive thermal control can facilitate the stability of the optics without increasing energy consumption [51].

Even though precision might be affected by space conditions thus far processing time appears to be the most significant limitation of the platform. Based on the results presented in Table 5 we prioritize and suggest the following optimizations:

- Reduction of the image's exposure time. Recently, a new RPi high-quality camera has been launched to the market [52]. This new camera's sensor has more pixels than camera V2.1, and its lens is customizable. It would be worth testing its capability as ST, looking for a diminish in the exposure time.
- Optimization of how the third-party software (*Source Extractor* and *Match*) is used. To introduce improvements in the operation of these we could modify their source code or port their core functionality.
- Explore other platforms such as the NanoPi NEO Air from NanoPi family [53]. Also, there are several newer cellphone multi-cores processors optimized to operate with cameras that might be explored. Some smartphones have already been deployed in space; an example of that is the NASA phone-sat project [54].

All code described in this work is open source and available for further development at the University of Chile SPEL group's GitHub site:`https://github.com/spel-uchile/Star_Tracker` and is also available as supplementary material of this article.

ACKNOWLEDGMENT

The authors want to acknowledge the following researchers who contacted them through the work published on the GitHub page: First is Kamil Borkowsky from the HyperSat team in Warsaw. He tried this open platform in the testing phase of their satellite development. Secondly is Nathan Raposo and Daniel Gatti from Mauá Institute of Technology. They also tested the code and are developing new applications with it. From SPEL, they want to acknowledge two people: First is Tomás I. Opazo, for assistance with Raspberry Pi programming. Secondly is Matías G. Vidal for his help in this article's review.

REFERENCES

[1] C. C. Liebe, "Star trackers for attitude determination," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 10, no. 6, pp. 10–16, Jun. 1995.
[2] C. C. Liebe, "Accuracy performance of star trackers—A tutorial," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 2, pp. 587–599, Apr. 2002.
[3] P. Salomon and W. Goss, "A microprocessor-controlled CCD star tracker," in *Proc. 14th Aerosp. Sci. Meeting*, Jan. 1976, p. 116.
[4] 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," in *Proc. AIAA/USU Conf. Small Satell.*, 2000, pp. 1–19.

[5] Board, Space Studies and National Academies of Sciences, Engineering, and Medicine, *Achieving Science With CubeSats: Thinking Inside the Box*. Washington, DC, USA: The National Academies Press, 2016, doi: 10.17226/23503.

[6] C. Boshuizen, J. Mason, P. Klupar, and S. Spanhake, "Results from the planet labs flock constellation," in *Proc. AIAA/USU Conf. Small Satell.*, 2014.

[7] K. Woellert, P. Ehrenfreund, A. J. Ricco, and H. Hertzfeld, "Cubesats: Cost-effective science and technology platforms for emerging and developing nations," *Adv. Space Res.*, vol. 47, no. 4, pp. 663–684, Feb. 2011.

[8] M. A. Diaz, J. C. Zagal, C. Falcon, M. Stepanova, J. A. Valdivia, M. Martinez-Ledesma, J. Diaz-Peña, F. R. Jaramillo, N. Romanova, E. Pacheco, M. Milla, M. Orchard, J. Silva, and F. P. Mena, "New opportunities offered by cubesats for space research in latin america: The SUCHAI project case," *Adv. Space Res.*, vol. 58, no. 10, pp. 2134–2147, Nov. 2016.

[9] C. R. McBryde and E. G. Lightsey, "A star tracker design for CubeSats," in *Proc. IEEE Aerosp. Conf.*, Mar. 2012, pp. 1–14.

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

[11] Y. Jianan, L. Yong, H. Fei, F. Qian, and P. Quan, "Pico-satellite attitude determination using a star tracker with compressive sensing," in *Proc. 34th Chin. Control Conf. (CCC)*, Jul. 2015, pp. 5331–5336.

[12] A. Khorev and L. Torres, "Performance of a smartphone based star tracker," in *Proc. 4th Interplanetary CubeSat Workshop*, London, U.K.: South Kensington, 2015, pp. 1–4.

[13] Y. Linn, "An ultra low cost wireless communications laboratory for education and research," *IEEE Trans. Educ.*, vol. 55, no. 2, pp. 169–179, May 2012.

[14] Y. Liao, N. Yu, D. Tian, C. Wang, S. Li, and Z. Li, "An intelligent low-power low-cost mobile Lab-On-Chip yeast cell culture platform," *IEEE Access*, vol. 8, pp. 70733–70745, 2020.

[15] E. I. Al Khatib, M. A. K. Jaradat, and M. F. Abdel-Hafez, "Low-cost reduced navigation system for mobile robot in Indoor/Outdoor environments," *IEEE Access*, vol. 8, pp. 25014–25026, 2020.

[16] NASA. *State of the Art of Small Spacecraft Technology*. Accessed: Mar. 30, 2020[Online]. Available: https://sst-soa.arc.nasa.gov/

[17] Sparkfun. *Sparkfun Triple Axis Accelerometer and Gyro Breakout—Mpu-6050*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.sparkfun.com/products/11028

[18] Sparkfun. *Sparkfun Triple Axis Magnetometer Breakout—Mlx90393*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.sparkfun.com/products/14571

[19] CubeSatShop. *Mai-Ses Ir Earth Sensor*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.cubesatshop.com/product/mai-ses-ir-earth-sensor/

[20] CubeSatShop. *Nss Fine Sun Sensor*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.cubesatshop.com/product/digital-fine-sun-sensor/

[21] CubeSatShop. *Ku Leuven Star Tracker*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.cubesatshop.com/product/kul-star-tracker/

[22] J. C. Springman, "Satellite attitude determination with low-cost sensors," Ph.D. dissertation, Dept. Aerosp. Eng., Univ. Michigan, Ann Arbor, MI, USA, 2013. [Online]. Available: http://hdl.handle.net/2027.42/102312

[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. AIAA/AAS Guid., Navigat. Control Conf.*, vol. 157, 2016, p. 170.

[24] 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, Nov. 2017.

[25] A. G. Sreejith, J. Mathew, M. Sarpotdar, R. Mohan, A. Nayak, M. Safonova, and J. Murthy, "A raspberry pi-based attitude sensor," *J. Astronomical Instrum.*, vol. 03, no. 02, Nov. 2014, Art. no. 1440006.

[26] 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, Sep. 2015.

[27] D. Honess and O. Quinlan, "Astro pi: Running your code aboard the international space station," *Acta Astronautica*, vol. 138, pp. 43–52, Sep. 2017.

[28] BBC. *Raspberry Pi Computer Looks Down on Earth*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.bbc.com/news/science-environment-49584941

[29] E. Bertin. (2016). *Sextractor*. Accessed: Mar. 30, 2020. [Online]. Available: http://www.astromatic.net/software/sextractor

[30] M. Richmond. (2012). *Match—A Program for Matching Star List*. Accessed: Mar. 30, 2020. [Online]. Available: http://spiff.rit.edu/match/

[31] 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 *Proc. 43rd Annu. AAS Guid., Navigat. Control Conf.*, 2020, pp. 1–12.

[32] 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, pp. 1–32, Jan. 2013.

[33] A. Scholz and J.-N. Juang, "Toward open source CubeSat design," *Acta Astronautica*, vol. 115, pp. 384–392, Oct. 2015.

[34] E. Bertin, "Sextractor user's manual," Inst. dAstrophys. Observatoire Paris, Paris, France, Tech. Rep., 2006. [Online]. Available: https://www.astromatic.net/pubsvn/software/sextractor/trunk/doc/sextractor.pdf

[35] F. G. Valdes, L. E. Campusano, J. D. Velasquez, and P. B. Stetson, "FOCAS automatic catalog matching algorithms," *Publications Astronomical Soc. Pacific*, vol. 107, p. 1119, Nov. 1995.

[36] B. Spratling and D. Mortari, "A survey on star identification algorithms," *Algorithms*, vol. 2, no. 1, pp. 93–107, Jan. 2009.

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

[38] G. Fasano, G. Rufino, D. Accardo, and M. Grassi, "Satellite angular velocity estimation based on star images and optical flow techniques," *Sensors*, vol. 13, no. 10, pp. 12771–12793, Sep. 2013.

[39] J. L. Junkins, C. C. White, III, and J. D. Turner, "Star pattern recognition for real time attitude determination," *J. Astron. Sci.*, vol. 25, pp. 251–270, Jan. 1977.

[40] D. Hoffleit, *Catalogue Bright Stars*, 3rd ed. New Haven, CT, USA: Yale Univ. Observatory, 1964.

[41] D. Mink, "WCSTools 4.0: building astrometry and catalogs into pipelines," in *Proc. Astronomical Data Anal. Softw. Syst. XV*, vol. 351, 2006, p. 204.

[42] W. M. Smart and R. Green, *Textbook on Spherical Astronomy*. Cambridge, U.K.: Cambridge Univ. Press, 1977.

[43] NASA. *STEREO Website*. Accessed: Mar. 30, 2020. [Online]. Available: https://stereo.gsfc.nasa.gov/

[44] C. J. Eyles, R. A. Harrison, C. J. Davis, N. R. Waltham, B. M. Shaughnessy, H. C. A. Mapson-Menard, D. Bewsher, S. R. Crothers, J. A. Davies, G. M. Simnett, R. A. Howard, J. D. Moses, J. S. Newmark, D. G. Socker, J.-P. Halain, J.-M. Defise, E. Mazy, and P. Rochus, "The heliospheric imagers onboard the STEREO mission," *Sol. Phys.*, vol. 254, no. 2, pp. 387–445, Feb. 2009.

[45] NASA. *Stereo Science Center—L0 Images*. Accessed: Mar. 30, 2020. [Online]. Available: https://stereo-ssc.nascom.nasa.gov/pub/ins_data/secchi/L0/a/img/hi_1/

[46] NASA. *Stereo Science Center—L2 Images*. Accessed: Mar. 30, 2020. [Online]. Available: https://stereo-ssc.nascom.nasa.gov/pub/ins_data/secchi_hi/L2/a/img/hi_1%/

[47] T. Sun, F. Xing, X. Wang, Z. You, and D. Chu, "An accuracy measurement method for star trackers based on direct astronomic observation," *Sci. Rep.*, vol. 6, no. 1, Sep. 2016, Art. no. 22593.

[48] N.-M. Inc. *Space Simulation Systems*. Accessed: Mar. 30, 2020. [Online]. Available: http://www.nanomaster.com/spacesimulation.html

[49] T. Sun, F. Xing, and Z. You, "Optical system error analysis and calibration method of high-accuracy star trackers," *Sensors*, vol. 13, no. 4, pp. 4598–4623, Apr. 2013.

[50] Y. Lai, D. Gu, J. Liu, W. Li, and D. Yi, "Low-frequency error extraction and compensation for attitude measurements from STECE star tracker," *Sensors*, vol. 16, no. 10, p. 1669, Oct. 2016.

[51] E. Escobar, M. Diaz, and J. C. Zagal, "Evolutionary design of a satellite thermal control system: Real experiments for a CubeSat mission," *Appl. Thermal Eng.*, vol. 105, pp. 490–500, Jul. 2016.

[52] R. P. Foundation. *Raspberry Pi High Quality Camera*. Accessed: May 9, 2020. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-high-quality-camera/

[53] FriendlyARM. *Nanopi*. Accessed: Mar. 30, 2020. [Online]. Available: http://nanopi.io/

[54] NASA. *Phonesat, the Smartphone Nanosatellite*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.nasa.gov/centers/ames/engineering/projects/phonesat.html

SAMUEL T. GUTIÉRREZ received the B.S. degree in applied physics from the University of Santiago of Chile, Santiago, Chile, in 2011. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Chile, Santiago. Since 2013, he has been a Lecturer of the introductory courses of physics for engineering students with the University of Santiago of Chile. Since 2016, he has been a part of the Space and Planetary Exploration Laboratory, University of Chile, collaborating in the nanosatellite-based space program of the University. His research interests include study and development of low-cost attitude sensors for nanosatellites.

CÉSAR I. FUENTES received the B.S. degree in physics and astronomy from the University of Chile and the Ph.D. degree in astronomy from Harvard University, in 2010. He held a postdoctoral position at AURA and Northern Arizona University. He is currently an Assistant Professor with the Astronomy Department, University of Chile. He heads the Dynamic Data Laboratory, University of Chile. His primary research interests include population of trans-Neptunian objects and near to Earth objects using targeted observations and big data analysis of ground and space-based observatories.

MARCOS A. DÍAZ received the degree in electrical engineering from the University of Chile, Santiago, Chile, and the M.S. and Ph.D. degrees in electrical engineering from Boston University, Boston, MA, USA, in 2004 and 2009, respectively. He is currently an Assistant Professor with the Electrical Engineering Department, University of Chile. His research interests include the study of ionospheric turbulent plasma, incoherent scatter radar techniques, low-frequency-radio-astronomy/space instrumentation, and nano-satellite technologies. He is in charge of the Space and Planetary Exploration Laboratory (SPEL), a multidisciplinary laboratory located in the Faculty of Physical and Mathematical Sciences, University of Chile, where the nanosatellite-based space program at the University is being developed.