 doi:10.20944/preprints201703.0111.v1

Article

Open Hardware and Software Star Tracker: An Opportunity for Collaboration in the Emerging Cubesat Community

Samuel T. Gutiérrez [1], César I. Fuentes [2] and Marcos A. Díaz [1,*]

[1] Electrical Engineering Department, Physical and Mathematical Sciences Faculty, University of Chile, Santiago, Chile; samuel.gutierrez@usach.cl

[2] Astronomy Department, Physical and Mathematical Sciences Faculty, University of Chile, Santiago, Chile; cfuentes@das.uchile.cl

* Correspondence: mdiazq@ing.uchile.cl; Tel.: +56-2-2978-4204

Abstract: We present an open-source Star Tracker (ST) for Attitude Determination. Our implementation makes use of open source software of common usage in astronomy running on a Raspberry Pi 2 platform. The developed platform is open and available for parties interested in further development. Evaluation of the system showed that the ST is suitable for Cubesats of 2U or larger. The ST platform is capable of solving the Lost-In-Space (LIS) problem. Currently, the average accuracy reached of the algorithm is close to 5 seconds of arc with an average processing time of 75 seconds. However, a average accuracy of $\sim$ 3 minutes of arc can be reached with 35 seconds in average processing time. We also describe the evaluation procedure, the found conclusions of this procedure, in particular the section of the algorithm where most of the processing time is spent and other possible source of errors not included in this study.

Keywords: star tracker; attitude determination; CubeSat

1. Introduction

The attitude determination is a crucial subject for any satellite that requires to know its orientation while in space. Whether it is collecting maximum energy from solar panels, or for successful energy-efficient communication, or for pointing a camera to a defined target, it is important to know the precise orientation to which a satellite is pointing. There are several sensors for attitude determination, such as: sun sensors, horizon sensors, magnetometers, inertial sensors and star trackers.

The Star Tracker (ST) is considered the most precise sensor for orientation and therefore critical for this task in satellites [1,2]. STs have been part of satellite missions since the dawn of the space era [3]. Since spacecraft evolved in a polarized world and as a power demonstration the technology and algorithms have not been available for the majority of the community. After the cold war era the space knowledge was slowly transferred to the private sector. The commercial exploitation of space reduced the costs but still kept the knowledge in a limited number of actors. STs followed this trend, therefore they are mostly available as commercial black boxes. In the late 90s a standardized satellite emerged, the Cubesat [4]. The Cubesat was born as a way to improve space technology

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

education. However, Cubesats have rapidly evolved to be actively used for research, science and even commercial applications [5,6].

Cubesat is a standardized shape platform with a cubical base unit of 10 x 10 x 10 cm (1U). The standardization reaches the deployer unit in the rockets supporting sizes of 2U, 3U, 6U and even 12U until now. This standardization together with the use of smartphone electronics [7] has reduced the cost and development time of this type of vehicles allowing access to space to developing countries [8,9]. With the interest that Cubesat are gaining, with relevant actors in the space sector adopting this bus as platform for research, development and exploration, it has become relevant to increase the capabilities of subsystems such as the ST [10–13]. However, the needed acceleration in knowledge and development necessarily requires a large number of participants collaborating in this process. In order to arrive fast to novel, capable but robust subsystems many test or attempt should be happening simultaneously. One way to achieve this is by facilitating the participation of developers with an open hardware open software platform. This kind of approach is not new, already happening in the software and microcomputers communities. This approach has brought much benefit with open source platforms, where independent and much smaller actors can contribute to the development of a larger system. Notorious example of this approach is Linux operative system and the Arduino platform. In this work we propose an open ST platform where Cubesat developers can collaborate in order to accelerate the advances and the evaluation of this new development.

The presented sensor platform is based on a popular open hardware miniaturized computer, the Raspberry Pi together with a simple camera easy to integrate into this platform. For the software we propose open software commonly used for astronomy. We intensively evaluate the current capabilities that this system might offer. We expect that this open sensor can facilitate the collaborative work on it and accelerate the results in order to come with outstanding ST for future astronomy space missions based on Cubesats.

The current work is structured as follow: In section 2 we describe the platform, which includes hardware, software and the algorithm. In section 3 we describe the evaluation process of the platform and the main results. In section 4 we discuss the results and the possible improvements to the platform.

2. Platform Description

A ST is a sensor that pictures stars and compares the pattern of them with a star pattern in a stellar catalog. Through this process it is possible to obtain the camera attitude, therefore also the satellite attitude. The system requires a hardware platform, a software suite and an algorithm of how to use hardware and software to arrive to an attitude estimation.

2.1. Software and Hardware Description

In the development of this platform we worked with frequently used programs within the field of astronomy. These programs were linked using *Python* software. The used **software** suits are:

Source Extractor: Program used to detect astronomical objects (e.g. stars, planets, galaxies, etc.) in an image. In our work *Source Extractor* is used to detect stars, to calculate their position and brightness magnitude. That is, it generates a list of sources (with their positions relative to the left lower corner of the image, which is the point (0, 0)), and the relative brightness respect to the background of the image. It can be downloaded for free from [14]. Further information regarding the use and features of the program can be found in [15] and [16].

Match: Match is a program used to establish a relationship between two different lists of objects.

Match requires as input two lists of files. Each list should contain the following objects:

1. X position of objects.
2. Y position of objects.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

3. Magnitude at point X and Y. In case of the image, it represents the brightness relative to the background of the image.

One of the lists corresponds to the list of sources delivered by *Source Extractor*. The other list is a segment of the star catalog. In our platform, match is used to find a relationship between the list of source objects (image) and the objects within the catalog. Match can establish either a linear or 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 segments in the catalog has to be linearized by using a projection of tangent plane. This procedure is explained in more details in subsection 2.2.1. The linear relationship between the two lists is represented with 6 coefficients (a, b, c, d, e, and f). Thus, the relationship between the lists 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 the (x, y) pair represents an object (star) in the source list (image), and point (x', y') represents same object but in the segment of the catalog list. With respect to the quality of the relationship found, match delivers several statistical parameters. The most significant ones that were 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 case refers to the system of the projected catalog).
Nr: The number of matched pairs of objects (stars) used to define the transformation between lists.

Match can be downloaded for free from [17]. In this work we use version **0.14** of the program. In addition, it is possible to find a user manual in the same website. The specific algorithm with which *Match* operates is based on [18]. A good summary of algorithms used in Star Trackers applications can be found at [19].

For the **hardware** in this work we used the following components:

1. Raspberry Pi 2 Model B V1.1.
2. Raspberry Pi Camera V2.1.

Both are inexpensive items, and are available in the market at a cost of approximately US $ 62 in total. The main technical characteristics of the Raspberry Pi and the camera are summarized in Tables 1 and 2, respectively.

Table 1. Specifications of Raspberry Pi.

Element	Value
CPU	900 MHz ARMv7 Cortex-A7
Memory (SDRAM)iB	1024 MiB
Weight	40 g

Table 2. Specifications of Raspberry Pi camera.

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

2.2. Algorithm Description

The ST starts taking a picture from which the brightest objects (starts) in it are extracted by using *Source Extractor*. Then this list of objects from the image has to be compared with a stellar catalog, which is stored in memory. In order to find the attitude of the camera (and ultimately the vehicle associated to it) a procedure of two parts was developed. First it identifies the area or segment in the celestial catalog with the best match. The catalog has to be divided in areas or segments of similar size of the field of view of the image in order to properly use *Match*. Then, in the second part, it refines the match between the image and the catalog segment by improving the accuracy of the position within the catalog segment of the used projection point. This projection is necessary to perform the match with the image, which is a flat projection of the sky. In this section we describe with more detail this procedure. Most of the parameters described in the algorithm were achieved after testing the procedure. The evaluation procedure and the results are presented in Section 3.

2.2.1. Stellar Catalog and Tangent Plane Projection

There are many types of stellar catalogs at present. The chosen catalog to work was the **Bright Star Catalogue** from Yale University Observatory [20]. This catalog contains 9110 objects, of which 9096 are stars. It contains stars brightness up to magnitude 6.5. The catalog was obtained using the *scat* application, which is part of the WCSTools [21] program package.
When comparing the photo of the sky with the stellar catalog, it is necessary to match the geometry of both systems. The star catalog is in spherical coordinates, while the photo is in a plane. In order to relate both systems, it is necessary to project the stars that are on the sphere, in a plane tangent to this at some specific point. Following [22], this is achieved according to the equations:

$$\eta = \frac{\cos D - \cot\delta \sin D \cos(\alpha - A)}{\sin D + \cot\delta \cos D \cos(\alpha - A)} \tag{2a}$$

$$\xi = \frac{\cot\delta \sin(\alpha - A)}{\sin D + \cot\delta \cos D \cos(\alpha - A)} \tag{2b}$$

Where A, D 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 our work, we can not use the whole stellar catalog, since the image is much smaller than the whole catalog and the *Match* program requires two list of similar sizes to properly operate. For this reason the catalog was stored in memory but divided in overlapping areas or segments of $60^\circ \times 60^\circ$ (similar area of the FOV of the camera). Each segment is stored in a projected form by using the center point of the segment. The center of each segment was moved first 10° in right ascension (A) and declination (D). Thus the segment (i, j) is the one with center (tangent plane point) $A_{segment\,i,j} = i^\circ$ and $D_{segment\,i,j} = j^\circ$ with limits of the segment $A_{segment\,i,j} \in [(i-30)^\circ, (i+30)^\circ]$ and $D_{segment\,i,j} \in [(j-30)^\circ, (j+30)^\circ]$, where $i = 0^\circ, 10^\circ, 20^\circ, \ldots, 350^\circ$ and $j = -90^\circ, -80^\circ, \ldots, 0^\circ, \ldots, 80^\circ, 90^\circ$. These overlapping projected segments are stored in memory of the Raspberry Pi. A second case was also studied with a separation step between projection centers of 15°.

2.2.2. Attitude Determination Algorithm

In order to determine the attitude of the ST, the following procedure is carried out:

1. **Take the image**. The camera, with a predefined exposure time, takes a photo of the sky. In this work, exposure time is evaluated with the current camera and optics (see section 3).
2. **Generate the list of sources from the image**. The list of brightest objects in the picture is obtained by using *Source Extractor*:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

(a) The image is transformed into ".fits" format, the format recognized by *Source Extractor*, and then entered to the program.
(b) *Source Extractor* generates the list of sources, with **(1)** the positions of the stars referred to the lower left corner of the delivered image, which is the point (0, 0) and **(2)** the brightness, relative to background, of each detected object. The 40 brightest objects are selected from the list delivered by *Source Extractor*. This number of selected objects was studied in section 3.
(c) The image is scaled from pixels to *mm*, in order to accurately represent the image that is in the CMOS of the camera and to be able to use *Match* with the linear procedure. Figure 1 shows the list of objects found for one of the images taken during the evaluation of the system, which willbe used as example during the algorith description.

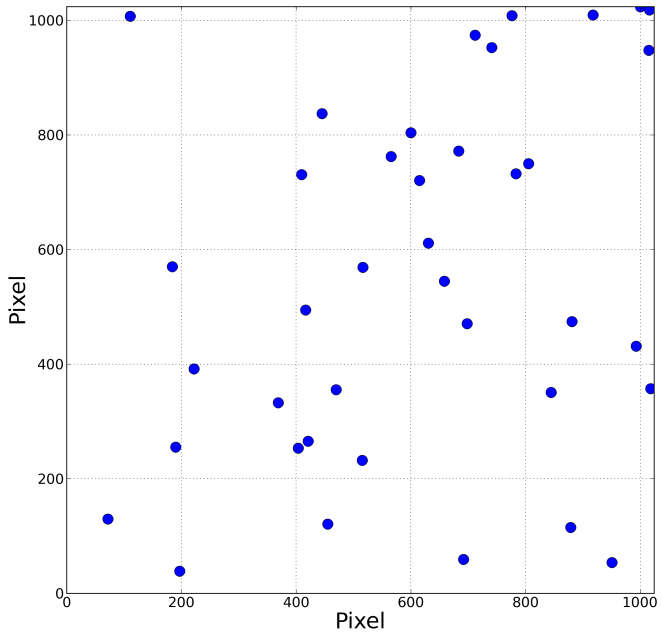


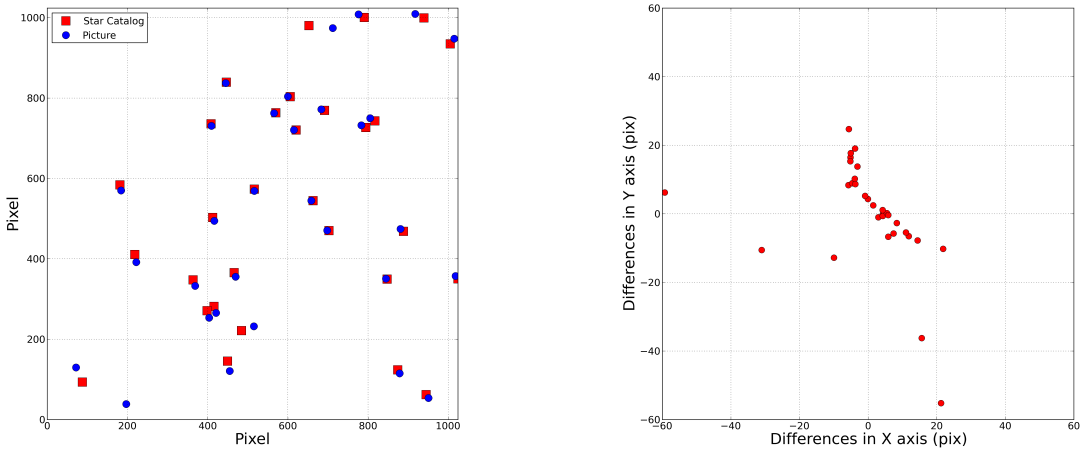
Figure 1. Sample image (in pixels) showing the 40 brightest objects extracted from the 800 ms exposure time photo.

3. **Matching: First Iteration**. Search throughout the celestial sphere. Match between the taken image and the in-memory catalog of celestial sphere.

(a) Matches are searched (using the *Match* program) between each segment of the in-memory star catalog (see subsection 2.2.1) and the list of brightest objects in the image taken (delivered by *Source Extractor*). The output of this stage is a list of candidate (matched) segments of the star catalog.
(b) The segment of the star catalog with the largest number of matched objects and with the lowest σ is selected as the final match. In our tests it is usually achieved with 20 objects (stars) or more and a $\sigma \sim 10^{-2}$. This is the output of the first stage of the algorithm. Further iterations are performed to improve the accuracy of the matching procedure, which ultimately gives the accuracy of the attitude estimation.
(c) The segments of the catalog, since they are curved, have to be projected in tangent plane. The center of this projection is arbitrarily selected in the first stage. After finding the segment of the catalog with the best match with the image the center point of the image is de-projected

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

(returned to sky coordinates) by using the linear model delivered by *Match* program. At this stage this point do not agree perfectly with the point arbitrarily selected. We use this de-projected point as a new center to improve the matching procedure in the next stages.



(a) *Match* - First iteration. **(b)** Differences between pair of points in first iteration.

Figure 2. The figure shows the results of *Match - First iteration*: (**a**) The Figure shows the objects, both from catalog and the picture of the sky, that the algorithm uses to find the relationship between both lists of objects (catalog - picture). (**b**) The Figure shows the difference (in pixels) between each pair of points that were matched to find the relationship between lists.

4. **Matching: Second Iteration**. The projection center of the selected segment of the star 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 *Match* program is used again with this corrected projection of the catalog segment and the list of objects extracted from the picture. A new list of matched objects is outed and the center is again de-projected to refine for the last time.
5. **Matching: Third and Final Iteration**. The projection center of the selected segment of the star catalog is corrected by using the matched stars found in the **second iteration** and the new center (de-projected in previous stage). The *Match* program is used again with this corrected segment of the catalog (projected with the new center) and the list of objects extracted from the picture. At this point the match between the two lists is accurate enough. The procedure is stopped here. The output of the *Match* program can be used together with the corrected projection center to find the A, D and Roll angles relative to the center of the camera, therefore the attitude of the vehicle. The Figure 3 shows in a comparative way the results obtained in the first and third iteration. Table 3 shows the effective output of the algorithm, the angles A, D and Roll, for the image of the example of Figure 1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

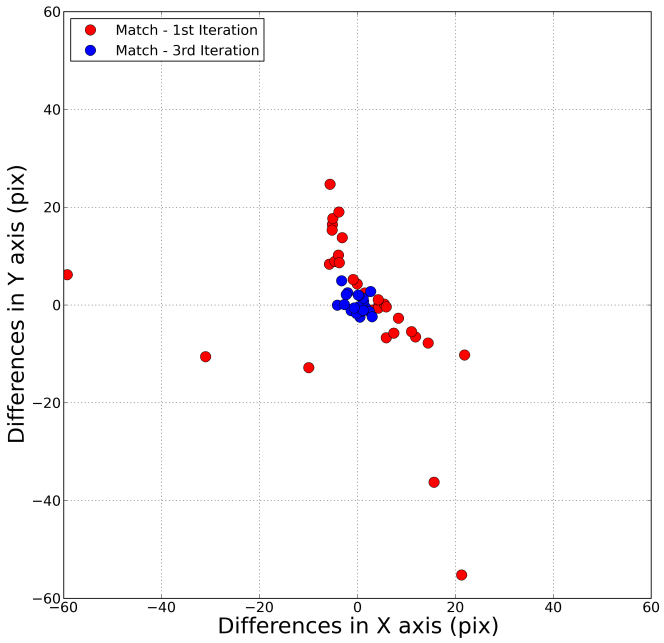


Figure 3. The figure shows the differences (in pixels) between pairs of matched objects (photo objects with those in catalog) when the right match is found. The result of the first iteration (31 matched objects) is shown in red color and the third iteration (25 matched objects) is shown in blue color. Dispersion in the first iteration is clearly greater than that obtained in the third iteration.

Table 3. Pointing coordinates for the solution found for the example presented in Figure 1, which agrees with the attitude of the camera (see Section 3).

A (°)	D (°)	Roll (°)
209.63	-52.45	-73.36

185 The process described above can be summarized in the flowchart shown in Figure 4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

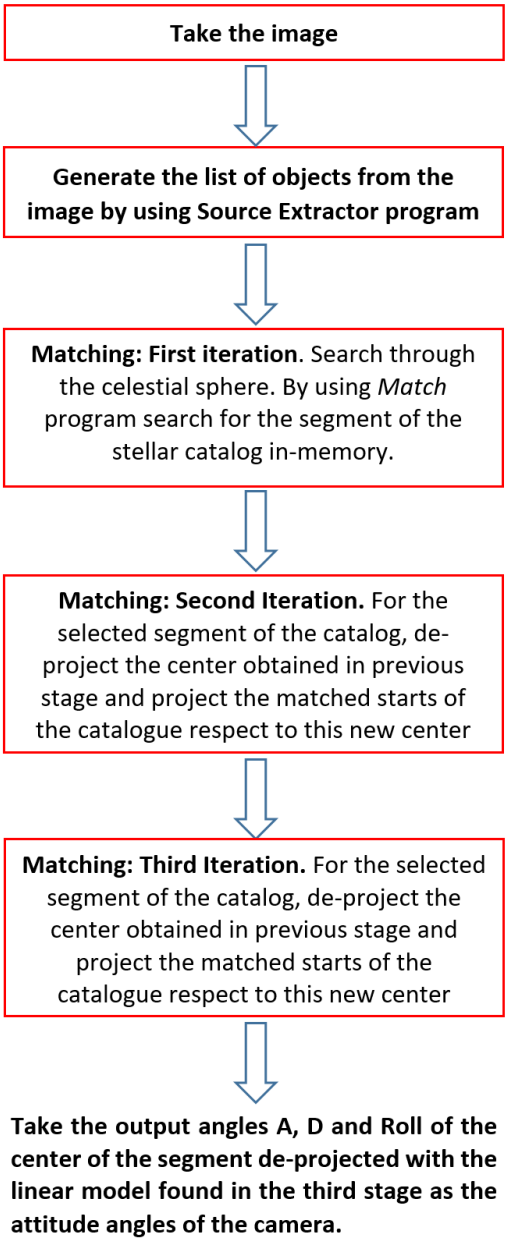


Figure 4. Flowchart of attitude determination algorithm (section 2.2).

3. Platform Evaluation

ST is one of the most accurate optical attitude determination devices. The veracity of the ST accuracy will eventually determine the satellite performance. However, how to realize and verify such accuracy remains a crucial but still on debate task. The actual accuracy is difficult to measure completely in a laboratory or in in-orbit conditions. Although, there are evaluation methods that use artificial stars within laboratory, in this work we follow the approach suggested by [23] for testing the ST by using real astronomical sources.

The platform was tested by taking photos of night sky, on Saturday, August 6th, 2016, between 16 and 18 hours UTC (20 and 22 hours local time). The geographical location where the measurements were performed is summarized in Table 4. The sky was pointed between Crux and Centaurus constellation. For the test over 50 images were used to evaluate the precision of the ST. The experimental setup with which the pictures of the sky were taken is shown in Figure 5 .

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

Figure 5. Equipment used for night sky measurements. In the image it is possible to see the tripod, the black enclosure of the Rasberry-Pi and the transparent enclosure of the camera.

Table 4. Geographic coordinates of the measurement site.

Latitude (°)	Longitude (°)	Altitude (masl)
-33.0133087	-70.9022930	720

The measurement consisted of taking pictures of night sky with different exposure times. With the measurement, the following studies were carried out:

1. To study the impact that different exposure times have on the attitude estimation accuracy, with the current optical hardware.
2. To study the impact of the star catalog segmentation on the ST algorithm. The main parameters to study were **(1)** the number of segments, **(2)** the size of the segments and **(3)** the center of each segment (segment overlapping).
3. To study the impact of the *Match* parameters on the accuracy of the estimation procedure and set up them to have a properly converging algorithm.
4. To study the feasibility of using within a Cubesat the proposed ST platform.

3.1. Exposure Time

Our algorithm expects to match tens of stars with their catalog sky positions. The exposure time of the picture will set the number of stars that will be detected on average. We set out to find the minimum exposure time as to secure a sufficient number of matches in the sky catalog.

We photographed a region around Crux ranging for up to 2 seconds at intervals of 0.1 seconds. The Camera was setup up to output 1024 x 1024 pixels, in order to deal with a square field of view, around 48.8° on the side. The number of source detections increased linearly with exposure time up to 0.9 seconds, at which point the number stays constant. We decided to use 0.8 seconds as exposure time since it yields over 80 detections under the relatively poorer conditions of ground-based

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

observations. We empirically found that with 20 matched stars we could get a good fit to the pixel-to-sky transformation. For this reason we selected, with a security margin of 100%, the 40 brightest detections in the image (See Figure 1), which is similar to select stars of magnitude 4 or below. Choosing catalog stars brighter than magnitude 4 yields a good match for the detections in the image and reduces the astronomical sources stored in memory of the catalog to **518**.

3.2. Catalog segmentation

In our algorithm *Match* properly operates based on two main assumptions: **(1)** the size of both list has to be comparable and **(2)** the relation between both lists has to be linear (there are other options for the relation but the linear one is the fastest). These assumptions explain most of the decisions taken within our algorithm regarding the catalog segmentation. For instance we could not use the catalog as a whole (as just 1 segment) because in this case the picture is much smaller than the catalog segment violating assumption (1). On the other hand, assumption (2) requires a projection that depends on the center of the field in order to linearly match a set of spherical coordinates (right ascension and declination) with cartesian coordinates (pixels), precisely the value we are looking for, which is difficult to do for just one segment.

Therefore, we separated the catalog in a set of projections to be compared with every taken image. The size of the segment is selected to be $60^\circ \times 60^\circ$, which is similar to the size of the picture (given for the FOV of the camera). However, the size of the catalog by itself does not define the number of segments. The matching algorithm tolerance to an error in the sky projection finally defines the number of catalog projections that will have to be tried in order to secure a match with a given image. We tested this by shifting the projection center in both right ascension and declination and measuring the new center of the field, as can be seen in Figures 6 and 7.

We varied both A and D of the projected point independently until *Match* was unable to provide a solution. With this procedure we evaluate the impact that deviations in the projected point might have in the final estimation of the attitude, and finally the estimation error of the algorithm. Although there are other errors associated to the optics and mechanical/thermal characteristics of the system [2], we specifically focus on the algorithm error. For instance when we select a distance between segments center of 10° in the sky, or overlapping between two consecutive segments of 50°, it yields the maximum center deviation corresponded to about 1.1 pixel or 3.2 minutes of arc (see Figure 6 (a) and Figure 7 (a)). If the distance between centers is reduced to 5° in the sky, it yields a precision of $\sim$ 5 seconds of arc, although increasing by a factor of 4 the processing time of the *Match* routine (or by 2 the total processing time).

3.3. Configuration of Match parameters

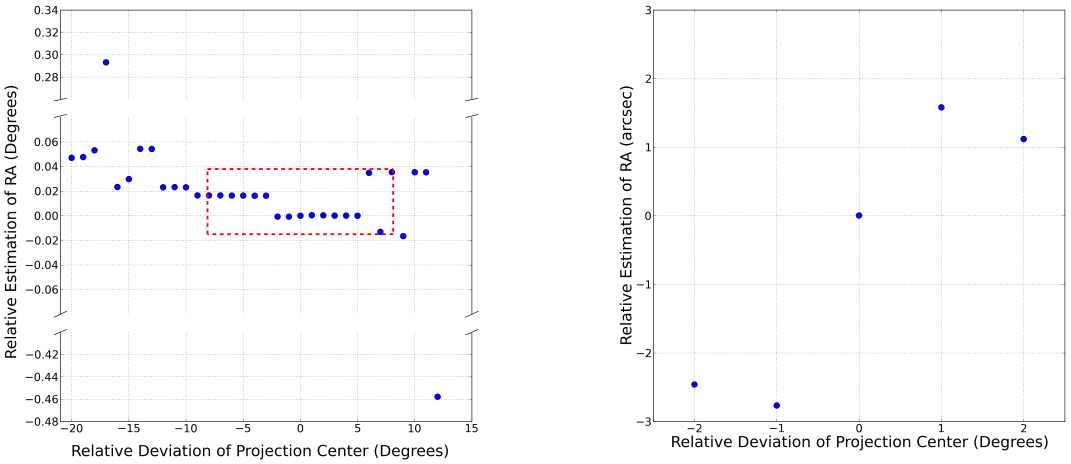
Match is a powerful and flexible software which has many parameters that need to be defined in order to properly operate in our algorithm. In this subsection we describe the main parameters used and the values given to them for our operation.

trirad: It is a parameter that defines when it is considered that the triangles formed between the objects of each lists are a 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 small values (less than 10), the found relationship between lists is not reliable. For large values (greater than 40), it takes a long time to find a match. For our program, 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 high number (for instance greater than 5) increases the execution time. For the first iteration this parameter is set to **1** and for second and third iteration it is increased to **3**.

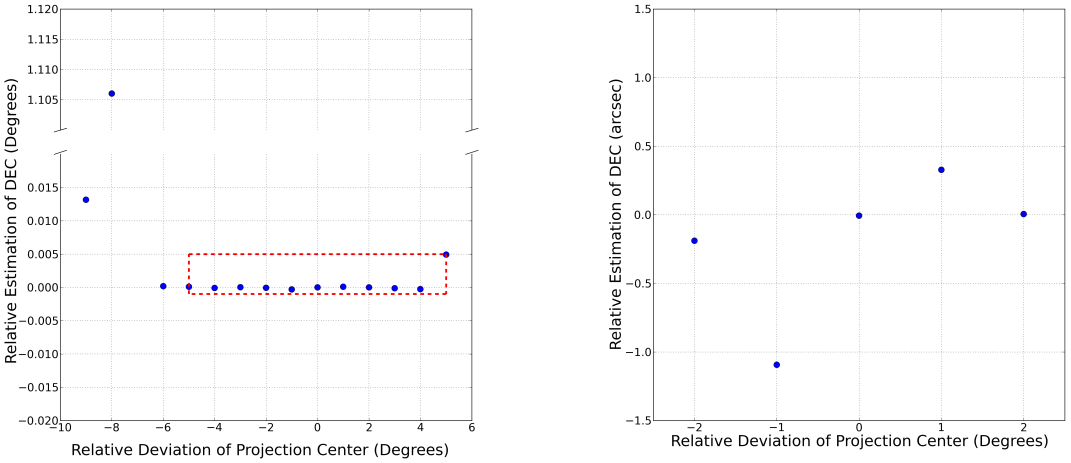
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1



(a) **(b)**

Figure 6. (**a**) Error in the estimation of right ascension (A), when projection point of the catalog segment is also deviated in right ascension for the example presented in Figure 1. The estimated solution for the example is $A = 209.63^\circ$ and $D = -52.45^\circ$ (see Table 3). The right ascention value of the projected point in the catalog segment was moved from 190° up to 222° (where 0 correspond to $A = 210^\circ$), keeping declination, D, constant at -52°. Outside this range the *Match* program was unable to deliver an answer. Inside the box (10°), the maximum variation in the estimation of D is $\sim$ 3.2 minutes of arc or $\sim$ 1.1 pixel. Figure (**b**) shows a zoom to the interval $[-2.5^\circ, 2.5^\circ]$ of Figure (a). It shows that when decreasing the distance between the projected center of each segment of the catalog, the estimation accuracy increases. In that interval the accuracy can reach $\sim$ 5 seconds of arc, however, at the cost of increasing the processing time 4 times compared to the 10° separation case (box in Figure (a)).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1



(a) **(b)**

Figure 7. (**a**) Error in the estimation of declination (D), when projection point of the catalog segment is also deviated in declination for the example presented in Figure 1. The estimated solution for the example is $A = 209.63^\circ$ and $D = -52.45^\circ$ (see Table 3). The declination value of the projected point in the catalog segment was moved from -61° up to -47° (where 0 correspond to -52°), keeping right ascension constant at 210°. Inside the box (10°), the maximum variation in the estimation of D is ~ 0.38 minutes of arc or ~ 0.1 pixel. Figure (**b**) shows a zoom to the interval $[-2.5^\circ, 2.5^\circ]$ of Figure (a). It shows that when decreasing the distance between the projected center of each segment of the catalog, the estimation accuracy increases. In that interval the accuracy can reach ~ 2 seconds of arc, however, at the cost of increasing the processing time 4 times compared to the 10° separation case (box in Figure (a)).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

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 very large (larger than 5), the linear model between lists is of poor quality (or precision) due to the large number of objects used in the relation as a match. If it is very small (less than 0.1), the linear model between lists can also be inaccurate due to the small number of objects used to establish the relation. For this reason, in our work we set an intermediate value of **1**.

scale: It defines the scale relation that exists between the lists. In our work, both lists are compared as if they were stars observed in the CMOS sensor of the camera (a plane). Since we have made everything in order to both lists correspond to each other, this value is set to **1**.

3.4. Feasibility of an ST for Cubesat: attitude determination accuracy, processing time and power consumption

The attitude estimation accuracy of the algorithm depends on how well the image matches with the catalog segment. Therefore, a large overlap among catalog segments improves the accuracy. Figures 6 and 7 show that a shorter distance among the projection centers of each catalog (or a larger overlapping area among catalog segments) increases the accuracy of the estimation (or decreases the estimation error). Thus, with a larger overlapping area it is more likely that the image has enough sources within a catalog segment. In addition, the accuracy of the algorithm is also dependent on the precision in the location of the projection point in each catalog segment, for this reason the iteration 2 and 3 of the algorithm. Figures 6 and 7 show that the intrinsic accuracy of the algorithm if other source of errors, such as optical and mechanical errors, are neglected is in average of ~ 5 seconds of arc with a distance among the catalog segment centers of 5° in D and A. For a distance among centers of 10° (in D and A) the accuracy reaches ~ 3.2 minutes of arc.

The processing time of any Star Tracker system is a relevant variable to assess its performance, which settles the type of application or science that can be achieved with it. The processing time depends on both the algorithm and the hardware. The software/algorithm and hardware are described in detail in section 2.1.

Table 5 summarizes the time taken by each process of the attitude determination of the ST, solving the LIS problem with a Raspberry Pi with overlapping between segments of 50° (separation between segment projected centers of 10°). These times were measured using the function `time.time()` from *Python*.

Table 5. Processing time of each major routing taken in Raspberry Pi. This numbers are for a separation of 10° between projection centers of each of the catalog segments stored in memory.

Process	Average Time (s)	Standard Deviation (s)
Importing libraries	5.39	0.03
Picture acquisition	9.381	0.003
Source Extractor	5.3	0.8
Match	14.5	0.4
Total	34.7	0.9

Table 6. Processing time of each major routing taken in Raspberry Pi. This numbers are for a separation of 5° between projection centers of each of the catalog segments stored in memory.

Process	Average Time (s)	Standard Deviation (s)
Importing libraries	5.42	0.05
Picture acquisition	9.393	0.007
Source Extractor	4.7	0.4
Match	53.4	0.9
Total	72.9	0.9

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 March 2017 doi:10.20944/preprints201703.0111.v1

The power consumption is another important variable of the ST system which can define the feasibility of being used in Cubesat missions. The power consumption of the ST system during the operation is summarized in Table 7. It shows that in the current stage the ST system can be implemented in Cubesats of 2U or above, although no action has been done thus far to diminish the power consumption. Further optimization might reduce the consumption especially during the *idle* state.

Table 7. Power consumption in each Star Tracker stage within Raspberry Pi.

Process	Maximum Power Consumption (W)
Idle	1.55
Attitude algorithm calculations	2.05
Image acquisition	2.40

4. Conclusions and discussion

In this work, we propose an open Star Tracker platform in order to accelerate the development of this type of sensor, which it is close to satisfy the constrains of weight, volume and energy of CubeSats of 2U and above. The ST operation is tested with ground campaigns. Taking pictures of known areas of night sky. The system uses as hardware a Raspberry-Pi and its camera. As software, it uses two common astronomy open software to perform the process: *Source Extractor* and *Match*. The system described in section 2.2 has 3 major steps (see Figure 4). First it takes a picture and extracts the $\sim$ 40 brightest starts from that image using *Source Extractor* software. Second, it looks for the proper area or segment in the whole sky by using the *Match* software, comparing the list of brightest objects extracted from the image with each catalog segment. Finally, the point of projection of the selected area or catalog segment is improved iteratively to achieve a better precision in the the attitude estimation. The best average precision reached of this system is of the order of $\sim$ 5 seconds of arc solving the lost-in-space problem in about 73 seconds. Nevertheless, an average precision of $\sim$ 3 minutes of arc can be reached in about 35 seconds. These processing times includes taking the picture, extracting the bright sources or stars, finding the correct segment of the sky catalog and estimating the attitude of the camera. Comparing this result with current comercial systems such as the Nano Star Tracker 1 (NST-1) from TY-Space, our system has a similar accuracy (although the optics and mechanics has not been yet evaluated in a hostaile environment) but the processing time still has to be improved.

To achieve this result, the following design decisions were made as the work evolved:

1. The celestial sphere catalog was divided in segments of size $60^\circ \times 60^\circ$. At the beginning, the list of brightest sources (stars) of the picture, extracted by using *Source Extractor*, was compared by using *Match* with the whole celestial catalog giving poor results. *Match* is intended to compare two list of similar number of objects. When comparing the list of sources from the picture with the objects of the whole catalog this principle was violated giving either low precision results or even converging to no solution. For this reason the celestial catalog was divided of segments of size $60^\circ \times 60^\circ$, which is similar to the field of view of the camera.
2. Configuring *Source Extractor* to extract no more than 40 objects from the picture. As highlighted in the previous point, in order to have good results with *Match*, both list has to have similar number of objects, which requires to divide the catalog in segments of size $60^\circ \times 60^\circ$. However, this is not sufficient to have a similar number of objects. It also requires to configure the number of objects extracted from the picture and in the catalog or the minimum magnitude in brightness of the stars used. Taking the 40 brightest objects from the picture, which is equivalent to stars of magnitude 4 or above within the catalog, yields the best precision for the ST.
3. The segments had to have an overlapping area among them. A first approach of the distribution of the catalog segments might be with segments as patches of the celestial sky with no

 doi:10.20944/preprints201703.0111.v1

overlapping area. The problem of this approach is that the image can fall in the boundary among adjacent catalog containing too few matching stars with each of the catalog. But the main problem of falling among catalogs implies that the matching stars of each catalog segment are in the edges of the image projected under a tangent point far from the projected point of the image delivering as an answer a no-match of the segment catalog. In this work we tested two separation among centers of catalog segments: 10° and 5°. Evaluation of the ST shows that shorter the distance among centers higher the accuracy at the cost of a higher processing time.

4. The minimum number of matching sources (or stars) had to be set in *Match*. Regarding the number of stars used to consider a match, for small values (less than 10), the found relationship between lists is not reliable. For large values (greater than 40), it takes a long time to find a match. For our algorithm, we set this parameter to **15** for the first iteration (searching the whole catalog), and it is increased to **20** for the second and third iteration (refining the projection point).
5. Each catalog segment was stored in-memory projected to a plane to facilitate the comparison with the image. It was also evaluated the possibility of de-projecting the objects observed in the photo, and compare them to each segment catalog in sky coordinates. However, this approach did not work, since *Match* is not the right program to relate lists of objects that are in spherical coordinates and in addition, it requires to do an extra operation to the image, de-project it, increasing even more the processing time.

Although the platform is fully described and tested at ground conditions, it is a first approximation to a open star tracker platform, and it can be subjected to further optimizations. For instance, it is necessary to include environmental considerations to to the system. Error of the optics has to be quantified and corrected [24]. In addition, the environmental conditions will add errors which might be diminished with mechanical subsystems as well as with compensation in the algorithm [25].

Although precision might be affected by space conditions thus far processing time appears as the poorer feature of this system. Based on the results presented in Tables 5 and 6 the following priority optimizations are suggested:

- To reduce the exposure time of the photo. To achieve this, the optics of the system might need improvement [24].
- To optimize the way in which the attached programs (*SExtractor* and *Match*) are used. It is necessary to introduce improvements in the way these programs operate, avoiding repetition of tasks that only need to be executed once.

All code described in this paper is open source and available to further development at Universidad de Chile's SPEL group's GitHub site: https://github.com/spel-uchile/Star_Tracker and it is available as supplementary material of this article.

Acknowledgments: This research was partially supported by Grants CONICYT PIA ACT1405 and CONICYT FONDECYT 1151476. Also, we thank Tomás Opazo for assistance with Raspberry Pi programming.

Author Contributions: The work presented in this paper was carried out between the 3 authors. S.G. programmed the Star Tracker in the Raspberry Pi using *Python* and performed the night sky measurements. C.F. gave his knowledge regarding the common programs used in Astronomy. M.D. contributed with his knowledge regarding the use and state-of-the-art of CubeSats. The algorithm design, the data analysis of the ground campaigns and paper writing were done in a collaborative manner among the 3 authors.

Conflicts of Interest: The authors declare no conflict of interest.

Bibliography

1. Liebe, C.C. Star trackers for attitude determination. *IEEE Aerospace and Electronic Systems Magazine* **1995**, *10*, 10–16.
2. Liebe, C.C. Accuracy performance of star trackers-a tutorial. *IEEE Transactions on aerospace and electronic systems* **2002**, *38*, 587–599.

3. Salomon, P.; Goss, W. A microprocessor-controlled CCD star tracker. 14th Aerospace Sciences Meeting, 1976, p. 116.
4. Heidt, H.; Puig-Suari, J.; Moore, A.; Nakasuka, S.; Twiggs, R. CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation. *Proceedings of the AIAA/USU Conference on Small Satellites* **2000**.
5. National Academies of Sciences, Engineering, and Medicine.; Board, S.S.; others. *Achieving Science with CubeSats: Thinking Inside the Box*; National Academies Press, 2016.
6. Boshuizen, C.; Mason, J.; Klupar, P.; Spanhake, S. Results from the planet labs flock constellation. *Proceedings of the AIAA/USU Conference on Small Satellites* **2014**.
7. Khorev, A.; Torres, L. Performance of a Smartphone based Star Tracker. 4th Interplanetary CubeSat Workshop; , 2015.
8. Woellert, K.; Ehrenfreund, P.; Ricco, A.J.; Hertzfeld, H. Cubesats: Cost-effective science and technology platforms for emerging and developing nations. *Advances in Space Research* **2011**, *47*, 663–684.
9. Diaz, M.; Zagal, J.; Falcon, C.; Stepanova, M.; Valdivia, J.; Martinez-Ledesma, M.; Diaz-Peña, J.; Jaramillo, F.; Romanova, N.; Pacheco, E.; others. New opportunities offered by Cubesats for space research in Latin America: The SUCHAI project case. *Advances in Space Research* **2016**, *58*, 2134–2147.
10. McBryde, C.R.; Lightsey, E.G. A star tracker design for CubeSats. Aerospace Conference, 2012 IEEE. IEEE, 2012, pp. 1–14.
11. Bezold, M. An Attitude Determination System with MEMS Gyroscope Drift Compensation for Small Satellites. Master's thesis, University of Kentucky, 2013.
12. Erlank, A.O.; Steyn, W.H. Arcminute Attitude Estimation for CubeSats with a Novel Nano Star Tracker. *IFAC Proceedings Volumes* **2014**, *47*, 9679–9684.
13. Jianan, Y.; Yong, L.; Fei, H.; Qian, F.; Quan, P. Pico-satellite attitude determination using a star tracker with compressive sensing. Control Conference (CCC), 2015 34th Chinese. IEEE, 2015, pp. 5331–5336.
14. Bertin, E. SExtractor. http://www.astromatic.net/software/sextractor, 2016. [Online; accessed 13-December-2016].
15. Bertin, E. SExtractor user's manual, 2006.
16. Holwerda, B.W. Source extractor for dummies v5. *arXiv preprint astro-ph/0512139* **2005**.
17. Richmond, M. Match – a program for matching star list. http://spiff.rit.edu/match/, 2012. [Online; accessed 13-December-2016].
18. Valdes, F.G.; Campusano, L.E.; Velasquez, J.D.; Stetson, P.B. FOCAS automatic catalog matching algorithms. *Publications of the Astronomical Society of the Pacific* **1995**, *107*, 1119.
19. Spratling, B.B.; Mortari, D. A survey on star identification algorithms. *Algorithms* **2009**, *2*, 93–107.
20. Hoffleit, D. *Catalogue of bright stars*; New Haven, Conn.: Yale University Observatory, 1964, 3rd rev.ed., edited by Hoffleit, Dorrit, 1964.
21. Mink, D. WCSTools 4.0: building astrometry and catalogs into pipelines. Astronomical Data Analysis Software and Systems XV, 2006, Vol. 351, p. 204.
22. Green, R.M.; Smart, W. Textbook on Spherical Astronomy. *Cambridge University* **1985**.
23. Sun, T.; Xing, F.; Wang, X.; You, Z.; Chu, D. An accuracy measurement method for star trackers based on direct astronomic observation. *Scientific reports* **2016**, *6*.
24. Sun, T.; Xing, F.; You, Z. Optical system error analysis and calibration method of high-accuracy star trackers. *Sensors* **2013**, *13*, 4598–4623.
25. Lai, Y.; Gu, D.; Liu, J.; Li, W.; Yi, D. Low-Frequency Error Extraction and Compensation for Attitude Measurements from STECE Star Tracker. *Sensors* **2016**, *16*, 1669.