Low-Cost Raspberry Pi Star Sensor for Small Satellites

Bharat Chandra P[a,b,*], Mayuresh Sarpotdar[c], Binukumar G. Nair[a], Richa Rai[a], Rekhesh Mohan[a], Joice Mathew[d], Margarita Safonova[a], Jayant Murthy[a]

[a]Indian Institute of Astrophysics, Bangalore, 560034, India.
[b]University of Calcutta, Kolkata, 700073, India.
[c]Satellite Research Centre, Nanyang Technological University, Singapore.
[d]Advanced Instrumentation and Technology Centre, RSAA, Mount Stromlo Observatory, Australian National University, Canberra, Australia.

Abstract. We present here a low-cost Raspberry Pi (RPi)-based star sensor *StarberrySense* using commercial-off-the-shelf (COTS) components, developed and built for applications in small satellites and CubeSat-based missions. A star sensor is one of the essential instruments onboard a satellite for attitude determination. However, most commercially available star sensors are expensive and bulky to be used in small satellite missions. *StarberrySense* is a configurable system – it can operate as an imaging camera, a centroiding camera, or as a star sensor. We describe the algorithms implemented in the sensor, its assembly and calibration. This payload was selected by a recent Announcement of Opportunity call for payloads to fly on the PS4-Orbital Platform by the Indian Space Research Organization (ISRO).

Keywords: Star Sensor, Attitude Sensor, Small Satellites, Raspberry Pi, CubeSat.

*Bharat Chandra P, bharat.chandra@iiap.res.in

1 Introduction

A star sensor onboard a satellite determines the satellite's attitude in space by identifying the stars in its field of view (FOV). The advantage of a star sensor over other sensors such as magnetometers, sun sensors, etc., is that it provides more precise orientation information of the satellite [1], to order of a few arcseconds to micro-arcseconds depending on the star sensor model. In recent years, small satellites are marking a new era of space exploration. Among these, a class of small satellites is called the CubeSats. They are usually used in low-Earth orbits (LEO) for communication, remote sensing, scientific missions and more. There might be a need for pointing information in these kinds of missions to perform orientation corrections to perform a particular observation. Most of the commercial radiation-hardened star sensors are made for long-duration space missions. However, using these expensive star sensors is not economical for the low-budget short-duration CubeSat or any SmallSat mission in LEO, where radiation levels are low[2]. Also, in some cases, commercial star sensors are bulky and consume a lot of power, making them impractical to be used in CubeSats[3]. To address this problem, our group started the development of a low-cost star sensor [4] using either readily available components or even building some of them in-house. The first prototype, *StarSense*, was built using a Star 1000 CMOS detector, a MIL-grade FPGA board, and a custom-fabricated lens system at the cost of around 10,000 USD [5]. Sarpotdar *et al.* had also developed a software package to evaluate the performance of star sensor algorithms, which included parameters such as attitude accuracy, sky coverage, catalogue size, etc., for different values of focal length, field of view, distortion effects and other attributes. We have continued the development of low-cost star sensors for short duration missions using readily available off-the-shelf components with a fast development cycle, modular design, and easy source code portability to different hardware. One concept of such a low-cost star sensor development, called Lab Open Star Tracker (SOST), based on Raspberry Pi and using existing open-source astronomy software,

Star Sensor	Accuracy(deg)	Update Rate(Hz)	Cost($)	Size (mm)	Weight(kg)	Power(W)
Procyon Star Tracker	0.008	$1-4$	330k	$155 \times 210 \times 56$	1.2	6.5
Comtech MST	0.02	1	150k	$50 \times 80 \times 80$	0.3	2
Sinclair Interplanetary ST-16	0.002	2	120k	$59 \times 56 \times 31.5$	0.09	1
Rigel-L	0.006	$5-16$	430k	NA	2.2	10
StarberrySense	0.008	0.2	1600	$98 \times 75 \times 69$	0.3	1.25

Table 1: Comparison between *StarberrySense* and other star sensors available in the market.

was recently described by Gutierrez *et al.* [6]. They employed the *Source Extractor* to extract the bright sources from the captured image and *Match* for matching the image of the sky with the projected segments of the sky stored onboard, followed by the attitude determination. From the real-sky test, they estimated the sensor accuracy to be about $30'' - 60''$ with an update rate of 0.05 Hz.

In *StarberrySense* we have used Raspberry Pi(RPi) Zero [7] as the main controller costing around 15 dollars, along with an RPi camera as the detector, off-the-shelf lens system for imaging, a custom-made power supply and a custom-made housing to mount all the component, bringing the total price down to about 1600 USD. Raspberry Pi is a small single-board computer (SBC) developed by the Raspberry Pi foundation[1]. It provides a compact, low-power, flexible platform to which different devices can be interfaced for applications ranging from home automation to aeronautical communication. RPi hardware design and computational capability allow it to be used in various small satellite missions, and this was the main reason for choosing the RPi. Using off-the-shelf components makes the development of a star sensor fast, cheap and easy without relying on complicated hardware and optics design. The *StarberrySense* has an accuracy of $30''$ with an update rate of 0.2 Hz, and an average power consumption under 1.25 W (see Section 2 and Table 3). Table 1 shows the comparison between the *StarberrySense* and other commercially available star sensors[8, 9, 10].

The software for the *StarberrySense* is written in standard C/C++ for faster processing on a free, open-source (FOSS) Linux platform and is easily portable to any other system such as Embedded Linux or RTOS (Real-Time Operating System), allowing faster development and deployment. The modular code architecture allows for easy customization, which is vital in the design of such subsystems in CubeSat or small satellite missions. The sensor operates in Lost-In-Space (LIS) mode [11], where centroids of possible stars in the image are identified by a region-growing algorithm. The geometric voting algorithm [11] is used to match the centroids with stars stored in the onboard catalogue. The final attitude determination is performed using the quaternion-estimator (QUEST) [12] algorithm. A minimum of three stars is required for attitude determination. The stored onboard catalogue is the Hipparchus bright star catalogue [13], listing stars with magnitudes up to 6.5.

2 The Instrument

A star sensor is essentially an imaging camera that images the stars, processes the images, identifies the bright stars and matches their positions with the stored star catalogue for determining the

[1] http://www.raspberrypi.org

look direction. As such, it consists of several subsystems that include optics, the image sensor, electronics and the housing (Fig. 1). The main controller in our star sensor is the Raspberry Pi Zero (Table 2).

Chip Broadcom	BCM2835
CPU	ARM1176JZF-S Single Core
	32-bit CPU
Operating system	Raspbian
GPU	Broadcom VideoCore IV
CPU Clock	1 GHz
Memory	512 MB DDR2
Interfaces	1×Micro USB
	1×UART
I/O	40 GPIO Pins
	CSI camera connector
Onboard storage	SD, MMC, SDIO card slot
Weight	45 gm
Power rating	250 mA (1.25 Watt)
Dimensions	66.0mm × 30.5mm × 5.0mm

Table 2: Raspberry Pi Zero technical specifications.

2.1 Design Requirements

The design requirement was to build a low-cost star sensor for small satellites and CubeSat class missions. The pointing accuracy requirement was based on the sub-arcminute pointing requirement for small satellite astronomy missions. The total mass was constrained to under 500 grams, and the system's power consumption below 2 W as per the mass and power budget of CubeSats and small satellites. Since we had a stringent need for a shorter development cycle, we chose to design the star sensor from readily available off-the-shelf electronics and ruggedized optical components.

Fig 1: Photo of the star sensor without the baffle.

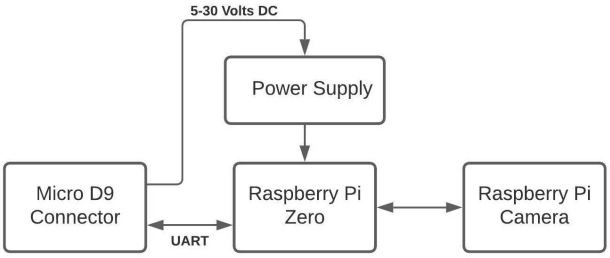

Fig 2: The electronics block diagram of the star sensor.

For the star sensor to work, three or more stars are required to be visible in its FOV. Therefore, the FOV (which depends on the focal length of the optics and the size of the image sensor) and the magnitude detection limit must be chosen carefully to meet the requirement. For our design,

the RPi V2 camera module was used as the image sensor and the Schindler Xenoplan 23-mm lens system as the imaging optics. This setup provided the system with a FOV of $9.31^\circ \times 7^\circ$, and the magnitude of the star detection limit was set to 6.5 ensuring a minimum of 3 stars in the FOV[4].

2.2 Image Sensor and Lens System

The star sensor uses the Xenoplan Schneider C-mount lens system designed to work in visible and near-IR range ($400 - 1000$ nm). The RPi V2 No-IR camera used in the star sensor has a Sony IMX219 CMOS image sensor with no IR filter that is sensitive from visible band to near-IR band. With the 23-mm lens system and the 1.12-micron pixel sensor, the star sensor has a pixel scale of $20.5''$ per pixel (with 2×2 binning). The dark noise characterisation for RPi V2 camera was done by Pagnutti *et al.* [14]. To ensure that stars with a magnitude of 6.5 are imaged with a good SNR, we chose the exposure to be 500 ms. In this setup, the mean value of background was 1.87 counts/pixel ($\sigma = 0.89$), and 6.5 mag stars were detected with $SNR = 30$.

2.3 Electronics

The Raspberry Pi Zero with ARM1176JZF-S Single Core processor is the main controller of our star sensor, which performs the image capture, image processing, star identification and attitude determination. The Raspberry Pi is interfaced with the camera module with the MIPI CSI (Camera Serial Interface), and the Broadcom VideoCore IV, onboard RPi, handles the image readout from the Sony IMX219 CMOS sensor. The universal asynchronous receiver-transmitter (UART) port of the RPi is configured for transmitting the output data (quaternions and centroids). The RPi Zero and the camera module consume less than 1.25 watts, which is within our power limit requirement. The power supply for the star sensor was designed based on a switching mode regulator LMR33630 from Texas Instruments, which provides a 5 V regulated voltage over a wide input range from 5 to 30 V. The LMR33630 has a wide operating temperature range from -40 to +125 $^\circ C$, making it suitable for our application (the summary of the technical specifications is presented in Table 3). Fig. 2 shows the overall electronics block diagram of the star sensor.

Table 3: *StarberrySense* technical specifications

Weight (without baffle)	315 gm
Weight (with baffle)	470 gm
Board	Raspberry Pi Zero
Dimensions (without baffle) (L×W×H)	$98 \times 75 \times 69$ mm
Dimensions (with baffle) (L×W×H)	$98 \times 75 \times 160$ mm
Power	1.25 W
Image Sensor	Sony IMX219
FOV	$9.31^\circ \times 7^\circ$
Wavelength range	$450 - 750$ nm
Limiting magnitude	6.5 (V)
Mode of operation	Lost-In-Space
Pointing Accuracy (3σ)	$27.18''$
Roll Accuracy (3σ)	$38.76'$

2.4 Mechanical Structure and Assembly

The mechanical enclosure for the star sensor was custom-designed and precision-manufactured for holding the camera, C-mount lens, power supply and the RPi Zero. The enclosure was made with

3-mm thick Aluminium 6061. Figure 3 shows the exploded view of the star sensor. The star sensor has a mass of about 315 gm, excluding the baffle. The entire housing is black anodized for better optical performance and to protect the surface from corrosion.

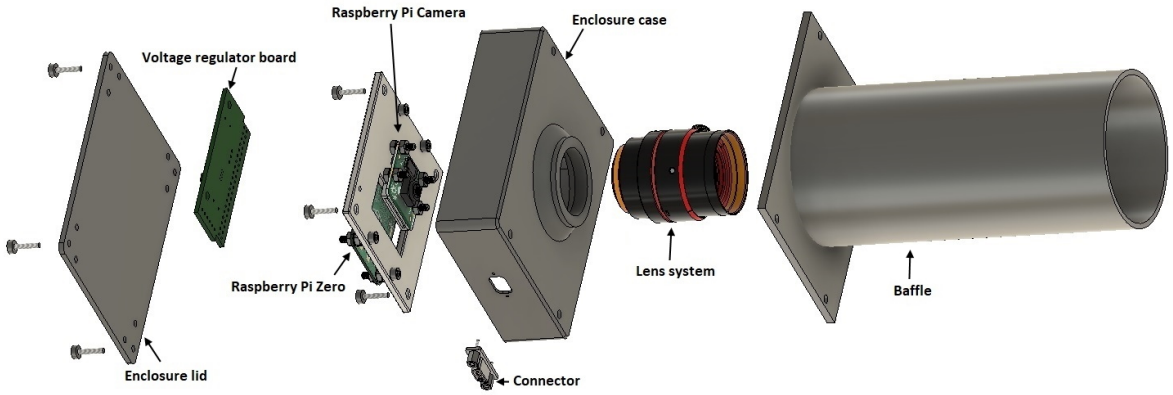

Fig 3: Exploded view of the mechanical structure.

To enhance the Sun and Moon avoidance angle, the star sensor requires a baffle. Otherwise, if the Sun appears near the FOV, it can saturate the detector, and the star sensor will not work. To prevent this, we designed a custom baffle following Asadnezhad *et al.* [15] to achieve the Sun avoidance angle of 30°, and manufactured it in-house from Aluminium 6061. The baffle was also black anodized to avoid scattering (Fig. 6).

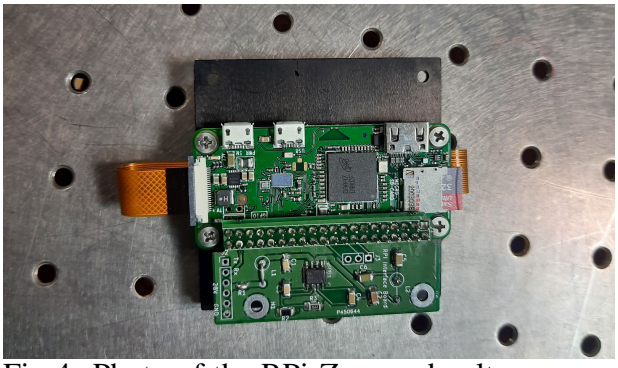

Fig 4: Photo of the RPi-Zero and voltage regulator board mounted on the intermediate plate.

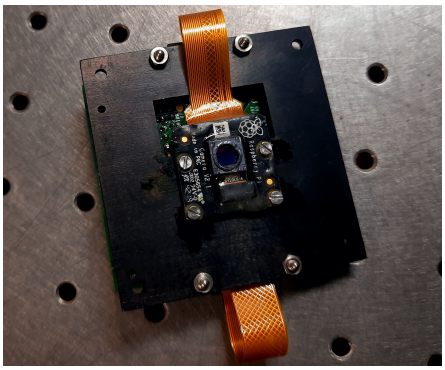

Fig 5: Photo of the RPi v2 camera mounted on the intermediate plate.

The assembly of the star sensor was initiated by bonding the lens mount for the C-mount lens system with the mechanical enclosure of the star sensor using epoxy Scotch-Weld 2216™. Once the glue on the mount was cured, the C-mount lens assembly was threaded onto the mount. RPi camera and RPi Zero was mounted on an intermediate aluminium plate along with the voltage regulator board as shown in Figs. 4 and 5. The intermediate aluminium plate was mounted inside the star sensor enclosure by aligning it with respect to the optomechanical axis of the lens barrel and the RPi camera using a collimated light source. The alignment precision of the system was bound by the surface finish of the mechanical components of the star sensor, which was in the range of $50 - 100$ micron. Then, the lens system was moved on the C-mount thread to bring the focal plane onto the image sensor plane, and the lens system was locked in position by using the

lock screw. Once the RPi camera was aligned, all the required electrical connections were made, and the enclosure was sealed using the enclosure lid. Once the main body of the entire star sensor unit was assembled, the baffle was fitted by using the mounting holes provided on the mounting plate of the enclosure case (Fig. 6).

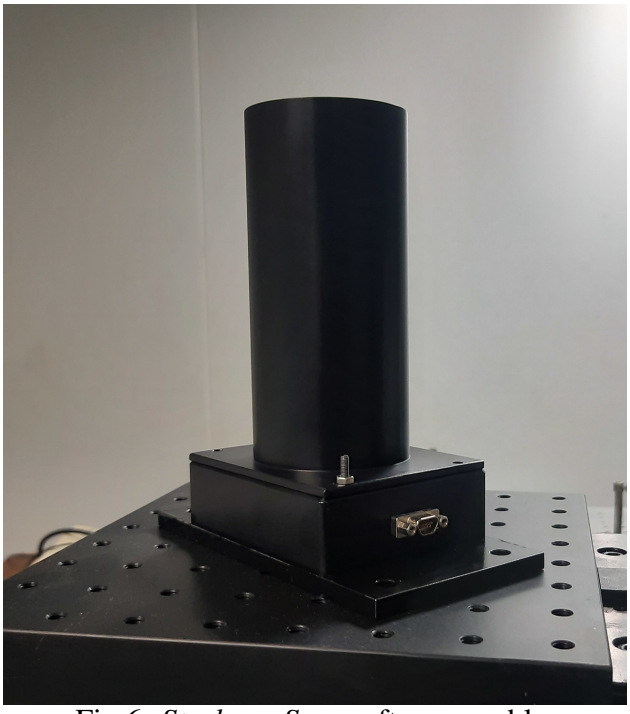

Fig 6: *StarberrySense* after assembly.

3 Software

For the onboard operating system, we have used a minimalist version of Raspbian with in-house developed algorithms written in C++ running as a standalone application. For the image capture and the camera control, a C++ API developed by the AVA research group was used [16]. The output of the sensor is attitude quaternions, which can be stored onboard and transmitted through the UART serial communication whenever needed.

3.1 Lost in Space (LIS) Algorithm

The LIS algorithm is used to determine the orientation of the spacecraft in space without the need for any initial knowledge of the satellite orientation. The algorithm identifies stars in the FOV and determines the attitude of the satellite. The algorithm uses a search table for star identification, generated and stored onboard prior to launch. During the star sensor in-flight operation, the distance value between star pairs in the image is compared with the search table for the voting. Once the voting is completed, and the actual stars are identified, the QUEST algorithm is used to determine the orientation in terms of quaternions of rotation. Fig. 7 shows the flow chart of the algorithm implemented onboard the star sensor.

3.1.1 Star Identification Process

Several methods have been developed for the identification of stars in the images [17]. Initially, stars were identified based on their magnitudes which required a flux-calibrated image sensor. The first-star sensor, based on the CCD, was developed by Salomon at JPL [18]. The major flaw with this technique was that over time, as the sensor degraded, the method was prone to errors. Then came the development of the pattern-matching based algorithms [17] for star identification. Among them, one of the well-known methods was proposed by Liebe *et al.* [1], which was based on finding the angular distance from a star to its two closest neighbours and the spherical angle between them to identify the star. This method paved the way for the development of algorithms that use star triangles for star identification [19, 20, 21, 22, 23, 24]. Another well-known method, proposed by Mortari *et al.* [23], is the pyramid algorithm which uses four stars to perform the star identification. This technique is more robust and reliable.

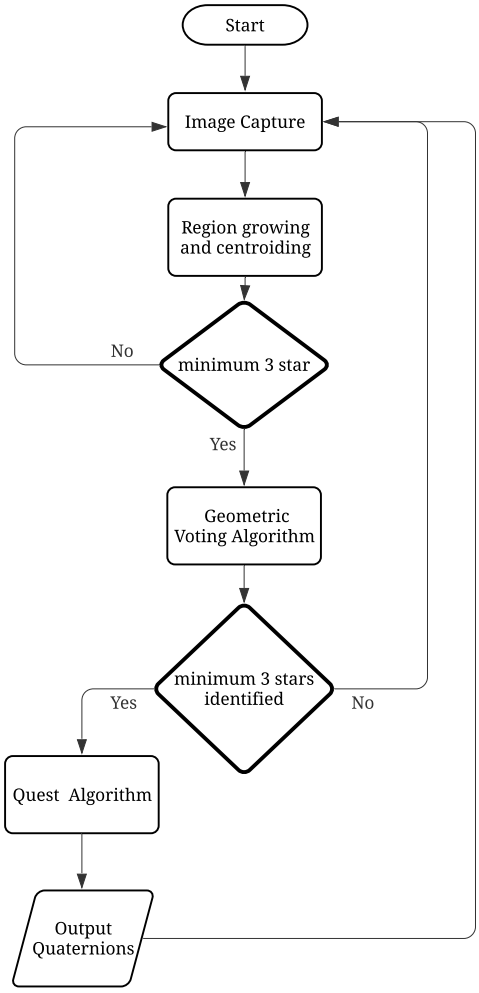

Fig 7: Flow chart of the algorithm onboard the star sensor

In our star sensor, we have used the geometric voting algorithm with a binary search algorithm for star identification [4, 11]. The geometric voting uses angular distances from a star to all-stars

in the FOV for voting to get the identity of the star. This is followed by the secondary voting to confirm these identities, where false stars are rejected, making it robust and less sensitive to noises.

3.1.2 Search Table Generation

The search table is generated manually and stored onboard before the flight. The search table was generated by calculating the angular distances between all possible star pairs in the Hipparchus catalogue and saving only those star pairs whose angular separation is less than the longest FOV D_{fov} (Table 2). Once the distances of these star pairs are calculated, they are sorted based on the increasing order of the distance between them and stored onboard the star sensor along with the unit vector of each star in the Earth-Centered inertial (ECI) coordinate system. The unit vectors of stars are calculated using the equation

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\delta \\ \sin\alpha\cos\delta \\ \sin\delta \end{bmatrix}, \tag{1}$$

where α is Right Ascension, and δ is the declination of a star. Thus, the search table stored onboard contains the list of all possible star pairs along with the angular distances between them. Algorithm 1 shown below was used to generate the search table, where D_{fov} is the longest FOV and d_{ij} is the minimum required angular separation between the stars, assumed to be $30''$ to avoid blending.

Algorithm 1 Search table generation

```
for i=1 to N do
    for j=i+1 to N do
        d_ij = distance between N_i and N_j
        if d_ij < D_fov and d_ij > D_min then
            Append the table row with elements i,j,d_ij.
        end if
    end for
end for
Sort the table based on d_ij.
```

3.1.3 Region Growing and Centroiding

The first part of the algorithm onboard the star sensor is finding the centroids of the stars. For this, an image of the sky is acquired and run through a region-growing algorithm to grow the brightest regions in the image above a threshold value. The threshold value is $p_{th} = \overline{p} + 5\sigma_p$, where $\overline{p}$ is the mean pixel value and σ_p is the standard deviation in the pixel values. The pixels above a threshold that lies in the neighbourhood of 2 to 3 pixels are considered as part of the same region. Here single-pixel regions are ignored since they are mostly either due to cosmic ray events or random noise. After the regions are grown, the weighted centroids for all the regions in the image are calculated using the following formula,

$$x = \frac{\sum_{i=0}^{N} x_i I_i}{\sum_{i=0}^{N} I_i}, \quad y = \frac{\sum_{i=0}^{N} y_i I_i}{\sum_{i=0}^{N} I_i}, \tag{2}$$

where (x, y) is the centroid, (x_i, y_i) is the ith pixel position and I is the intensity of the i^{th} pixel. The weighted centroiding helps us achieve sub-pixel level accuracy. Once centroids are calculated, the unit vectors for each star in the camera reference coordinate system is calculated using the formula,

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} \frac{pp_x \times (x - x_0)}{K \times f_{mm}} \\ \frac{pp_y \times (y - y_0)}{K \times f_{mm}} \\ \frac{1}{K} \end{bmatrix}, \tag{3}$$

where

$$K = \sqrt{\left((\frac{pp_x \times (x - x_0)}{f_{mm}})^2 + (\frac{pp_y \times (y - y_0)}{f_{mm}})^2 + 1 \right)}.$$

.

Here, f_{mm} is the focal length of the lens system, (x_0, y_0) is the location of the central pixel of the detector, pp_x is pixel size along x-axis and pp_y is the pixel size along z-axis. Once the unit vectors for all the stars in the image have been calculated, geometric voting will begin.

Algorithm 2 Star sensor algorithm

Calculate threshold for region growing and perform region growing
Calculate the centroids of the stars along with uncertainty ε of the centroid
Correct centroids for distortion
Assign **n** as the total number of centroids
Calculate $\vec{\mathbf{p}}$, the unit vector for the centroids in sensor body coordinate
for i=1 to n **do**
    for j=i+1 to n-1 **do**
        Compute distance $d_{ij} = \cos^{-1}(\vec{p_i} \cdot \vec{p_j})$
        if $d_{ij} < D_{fov}$ **then**
            upper_Index = binary_search($d_{ij} + \varepsilon_{ij}$)
            lower_Index = binary_search($d_{ij} - \varepsilon_{ij}$)
            for k = lower_index to upper_index **do**
                for all entries T(k) **do**
                    Append voting list V_i and V_j of possible stars S_i and S_j
                end for
            end for
        end if
    end for
end for
for i = 1 to n **do**
    Find the catalog star with maximum votes in voting list V_i
    Assign St_i to the catalogue star which got the maximal votes
end for
for i = 1 to n **do**
    for j = i+1 to n **do**
        Compute distance $d_{ij} = \cos^{-1}(\vec{p_i} \cdot \vec{p_j})$
        if distance between stars St_i and $St_j \in d_{ij}$ **then**
            Add a vote for the match of (St_i,S_i) and (St_j,S_j)
        end if
    end for
end for
Identify true stars based on primary and secondary votes.
Estimate attitude using QUEST algorithm.

Many factors cause the measured centroids to shift from their actual positions. Centroiding accuracy is proportional to the square root of the signal from the stars, and if the point-spread function (PSF) is less than 0.5 pixels, the centroiding accuracy will be limited by sampling theorem [25]. Another factor is the radial distortion caused by the lens system. This results in the centroids being displaced radially. This distortion must be accounted for during the calibration process with a suitable distortion model and corrected during the processing [11, 25] (Sec. 4.2).

3.1.4 Geometric Voting

In geometric voting, the distance between each star pair is calculated for the matching process. The distance can be calculated by finding the inverse cosine of the unit vector dot product of the star pairs. Once the distance is found, the star's ID should be in the search table in the distance range from $d_{ij} - \varepsilon$ to $d_{ij} + \varepsilon$, where ε is the uncertainty in calculated distance [11], assumed to $20''$ in our case. Using a binary search algorithm, the location on the search table is found where this distance range lies. Once the location of the distance range is known, the stars IDs, lying between the distance $d_{ij} - \varepsilon$ and $d_{ij} + \varepsilon$, will be used for the voting. After the voting, the total votes are

counted to identify the star in the image that corresponds to the star in our catalogue. The second round of voting is used to remove falsely identified stars. In the end, the total number of votes from primary and secondary rounds is used to identify the stars and to remove false detections. Once the stars in the image are identified, the star sensor proceeds to the attitude determination.

3.2 Attitude Determination

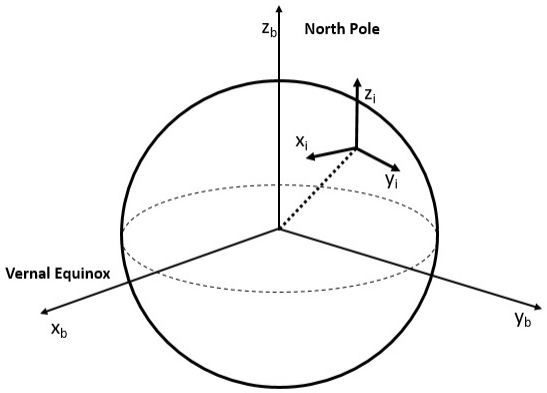

Fig 8: Figure showing the ECI-coordinate system and the sensor body coordinate system

Algorithm 2 is used to identify stars and determine the attitude. Once stars in the image are identified, the final step is to calculate the quaternion of rotation between the ECI coordinate system and the sensor body coordinate system. The geometric voting algorithm provides the unit vector of the identified star in the ECI coordinate system, and the unit vector for that star in the sensor coordinate system is calculated using Eq. (2).

Two commonly used algorithms to determine the attitude from the unit vectors are TRIAD[26] and QUEST[12]. Tri-Axial Attitude Determination (TRIAD) determines the direction cosine matrix that describes the relationship between two coordinate systems. TRIAD algorithm only uses two of the unit vector pairs and discards the rest, thus not utilizing the complete information from all the star unit vector pairs. Since each identified star has a pair of unit vectors, the total amount of pairs can be 20 or more (see Fig. 10 for an example of 18 registered stars). Since we want to use the complete information, the algorithms suited for handling such cases are Davenport's q-method[27], QUEST and SVD[28, 29]. Since QUEST bypasses the eigenvalue problem and is computationally less expensive, we have implemented it in our attitude determination algorithm.

Using the QUEST algorithm with multiple unit vector pairs as inputs, the star sensor calculates the quaternion of rotation between the two coordinate systems. The rotation can be described in many ways, such as rotation matrix, Euler angles, quaternions, etc. In our case, we have chosen quaternions as they are computationally less intensive. Now, if v_b is the unit vector for the star in the sensor body coordinate system and v_i is the unit vector for the ECI coordinate system (Fig. 8), we have

$$v_b = Rv_i \,, \tag{4}$$

where R is the rotation matrix that describes the rotation between the ECI and the sensor body coordinate systems. Once the stars in the image are identified, we know the unit vector of those

stars in both coordinate systems. Now, the star sensor needs to find the rotation matrix R from the unit vectors. The value of R must be such that it minimises the loss function

$$J = \frac{1}{2} \sum_{N}^{k=1} w_k |v_b - Rb_0|^2 \,, \tag{5}$$

where J is the loss function, w_k is the weight of each star unit vector pair, and N is the number of correctly identified stars. Since the measurements are not ideal, we will always have a value of J greater than zero. The star sensor needs to find a solution that minimises the loss function J, and the method should not be computationally intensive. There are several existing methods to solve this classic Wahba's problem [30]. We restate the loss function in terms of quaternions in such a way that it becomes an eigenvalue problem, where the largest eigenvalue is to be found. The QUEST algorithm was developed to bypass the expensive eigenvalue problem by approximating the process. It reduces to solving for parameter p, called the Rodriguez parameter, in the equation

$$\left[\left(\lambda_{opt} - \sigma \right) I - S \right] p = Z \,, \tag{6}$$

where

$$\begin{aligned} Z &= \left[B_{23} - B_{32} B_{31} - B_{13} B_{12} - B_{21}^T \right] \,, \\ B &= \sum_{N}^{k=1} w_k \left(v_{kb} v_{ki}^T \right) \,, \\ \lambda_{opt} &= \sum w_k \,, \\ \sigma &= Tr(B) \,, \end{aligned}$$

and

$$S = B + B^T \,. \tag{7}$$

After finding the value of p, the attitude quaternion is given by

$$q_{quest} = \frac{1}{\sqrt{1 + p^T p}} \begin{bmatrix} p \\ 1 \end{bmatrix} \,, \tag{8}$$

where

$$q_{quest} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \,. \tag{9}$$

4 Calibration and Testing

4.1 Performance and precision limit

We calculated the theoretical limit of accuracy for noise equivalent angle (NEA) following calculations in Liebe (2002) [31]. NEA measures the ability of a star sensor to reproduce the same

attitude information for the same part of the sky. Even though NEA depends on different instrumental parameters such as dark noise, read noise, etc., it is possible to quickly estimate it. For that, we first find the average number of stars in the FOV N_{stars},

$$N_{stars} = N_{catalog}\frac{1-\cos\frac{D}{2}}{2} = 11\,, \tag{10}$$

where $N_{catalog}$=8874 is the total number of stars from the Hipparchus catalog used to generate the search table (stars with magnitudes less than or equal to 6.5). The average *StarberrySense* FOV is $D = 8.15^{\circ}$ and the average number of pixels is N_{pixels}=1436.

The cross boresight NEA (E_{CB}), which gives the bound of error, or uncertainty, in the pointing direction; and the roll axis NEA (E_{Roll}), that gives the bound of error, or uncertainty, in the roll axis, are calculated as

$$E_{CB} = \frac{DE_{centroid}}{N_{pixels}\sqrt{N_{stars}}} = 3.05^{''}\,, \tag{11}$$

and

$$E_{Roll} = \tan^{-1}\left(\frac{E_{centroid}}{0.3825N_{pixels}}\right)\frac{1}{\sqrt{N_{stars}}} = 56.06^{''}\,, \tag{12}$$

where $E_{centroid}$ is the centroiding accuracy of the star sensor, assumed to be 0.5 pixels as the worst-case scenario.

4.2 Real Sky Test and Distortion Correction

After the star sensor was assembled, a night-sky test was conducted at Vainu Bappu Observatory (VBO) facility of IIA, Kavalur, Tamil Nadu. The star sensor was mounted on a stable platform and pointed towards the night sky with no pointing knowledge. Two subsequent night-sky tests were conducted, the first to correct for the distortion of the lens system and the second to verify the pointing performance.

The main distortion that affects the angular measurement of the distance between stars is the radial distortion which causes a significant error during the voting process leading to false star identification. The obtained image of the sky in jpeg format was fed to Astrometry.net [32] for astrometric calibration and generation of distortion-correction polynomials coefficients. Thereafter, these polynomials coefficients were stored onboard the star sensor for performing distortion correction using the SIP convention method [33]. If (u, v) is the centroid of the star in the pixel coordinate system and (x, y) is the centroid of the star after the correction, we have the distortion correction equation

$$\begin{aligned} x &= u + \sum_{a,b} A\_a\_b\, u^a v^b, \\ y &= v + \sum_{a,b} B\_a\_b\, u^a v^b, \end{aligned} \tag{13}$$

where A_a_b and B_a_b are polynomial coefficients of the second order, and a and b are integers, $a + b \leq 2$. This correction was implemented in the Algorithm 2 after the centroid calculation, and the corrected centroids were used for unit vector calculation. We found that second-order polynomial correction was sufficient as the distortion was less than a $26^{''}$ even at the edges.

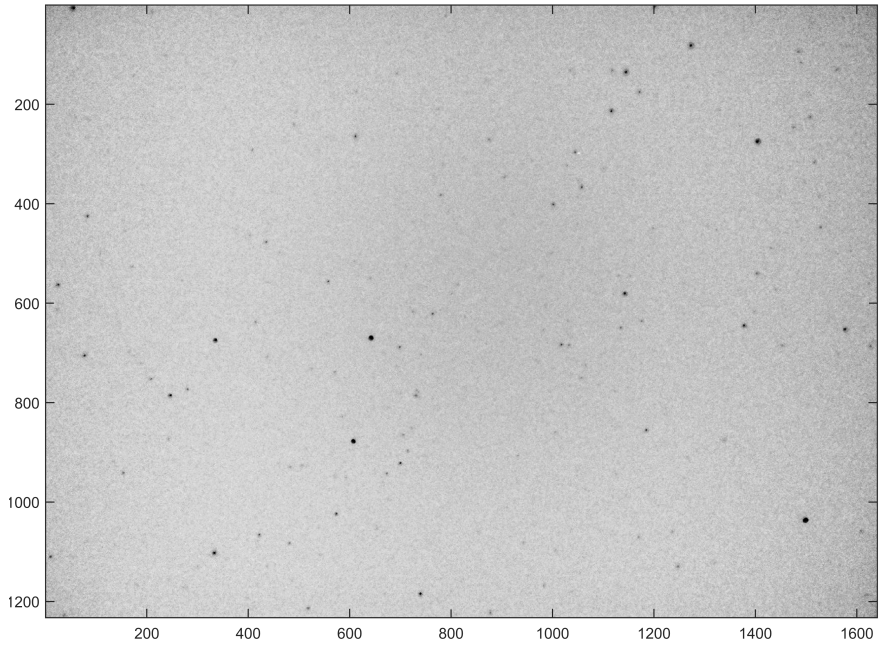

Fig 9: 500-ms exposure inverted grey-scale image of the night sky captured with the star sensor. x and y axes are pixel numbers.

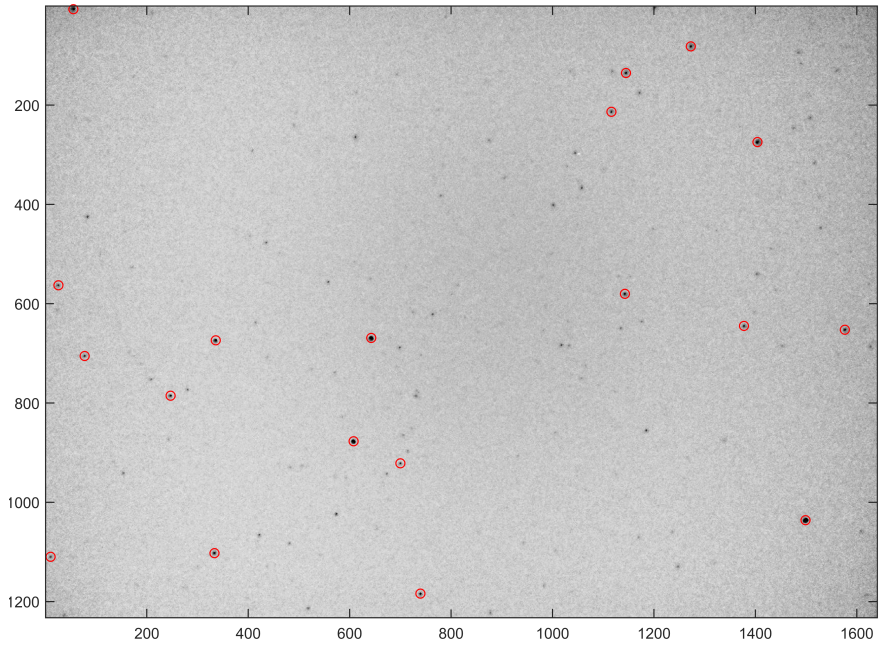

Fig 10: Same image as Fig. 9 with marked centroids of the stars above the threshold (Sec. 3.1.3) after running the region-growing algorithm . x and y axes are pixel numbers.

In the second test, the star sensor was pointed at different parts of the sky, and the captured images were stored onboard along with calculated quaternions. Fig. 9 shows one of the captured images, pointed at a random part of the sky, and the calculated centroids of the stars after running the region-growing algorithm are shown in Fig. 10. Quaternions, obtained from the star sensor,

were used to find the pointing [34], which in case of Fig. 9 was $\alpha = 91.0176°$ and $\delta = 14.1498°$, while the actual pointing from Astrometry.net was $\alpha = 91.019°$ and $\delta = 14.149°$. The ground-based observations error was found to be $6''$ when pointing from the star sensor was compared with the pointing obtained from Astrometry.net.

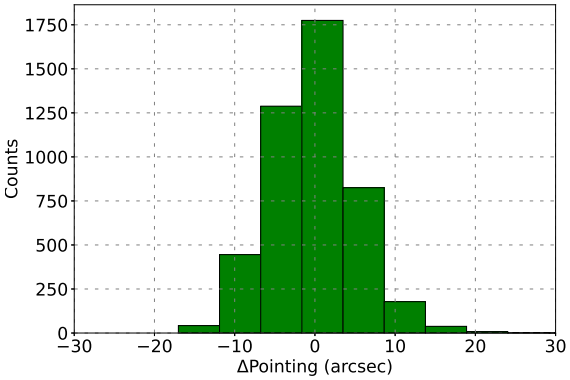

Fig 11: Plot shows the histogram of error for the *StarberrySense* in the pointing axis.

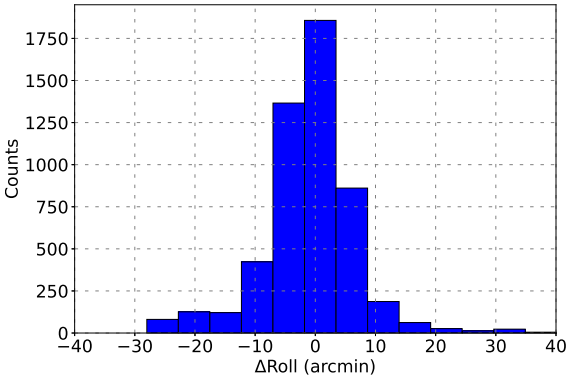

Fig 12: Plot shows the histogram of error for the *StarberrySense* in the roll axis.

To determine the accuracy of the *StarberrySense* in the pointing and roll direction, the star sensor was mounted on a stable platform and pointed towards the sky. The star sensor was set to calculate the quaternions and store them on board along with the sky images for a long duration. The actual pointing obtained from the Astrometry.Net using the stored images and the pointings obtained from the star sensor was used to determine the accuracy in the pointing and in the roll axis. Fig. 11 and Fig. 12 show the histogram of error in the pointing and roll axis, respectively. The measured accuracy in the pointing axis was 3σ = 27.18 arcsec, and in the roll axis was 3σ= 38.76 arcmin. The success rate of the star sensor was found to be 94.69% from the real-sky test. Also, the sky condition during the test was close to 5 on the Bortle scale, which is a measure of the night sky brightness [35]. We also determined the slew rate limit by simulating the star trailing on the images captured by the star sensor. We found that after the simulated motion rate at 0.2°/s, the algorithm failed to determine the quaternions.

4.3 Power Consumption and Processing Time

Table 4 shows the average power consumption during different processes: when the star sensor is idle, capturing the image, processing the image, or during UART transmission. The star sensor draws an average current of 140 mA when idle, and the current can peak to 220 mA during the image capture.

Table 4: Star sensor power consumption.

Process	Power
Idle	0.7 Watts
Image capture and readout	1.1 Watts
Image Processing	0.9 Watts
UART transmission	0.75 Watts

Table 5 shows the processing time taken by each of the subroutines in the star sensor algorithm. The image capture and readout take the longest amount of time, and the star sensor can provide 14 quaternions per minute.

Table 5: Star sensor processing time for the different subroutines.

Process	Time
Image capture and readout	3.6 s
Threshold determination	240 ms
Centroiding	220 ms
Geometric Voting	50 ms
QUEST	50 ms

5 Flight Qualification

To qualify for the space flight, payloads have to undergo the standard environmental tests: vibration and thermal-vacuum tests. This is to ensure that the payload will be able to withstand all launch loads and operate in the space environment. The most stringent vibration requirements for launch vehicle platforms are the following: the natural frequency of the payload must be above 100 Hz, and the instrument should be able to withstand acceleration loads up to 10g[36, 37]. The thermal-vacuum and vibration tests for the sensor were performed in the M. G. K. Menon Lab, CREST Campus, IIA, Bangalore, as per the Polar Satellite Launch Vehicle (PSLV) Stage 4 flight requirements.

5.1 Thermal-vacuum test

A thermal-vacuum chamber was used to simulate the thermal environment that the sensor will be subjected to in space. Four temperature probes were mounted on the *StarberrySense* body as shown in Figs.13 and 14 using aluminium tape, then the pressure inside the chamber was pumped down to $10^{-6}\,mbars$ with the help of a roughing pump and a turbo-molecular pump. The temperature in the chamber was varied through seven cold and hot cycles to simulate the orbital environment in the LEO. Fig. 15 shows the temperature measurement from the four sensors mounted on the star sensor overplotted on the programmed chamber profile[38].

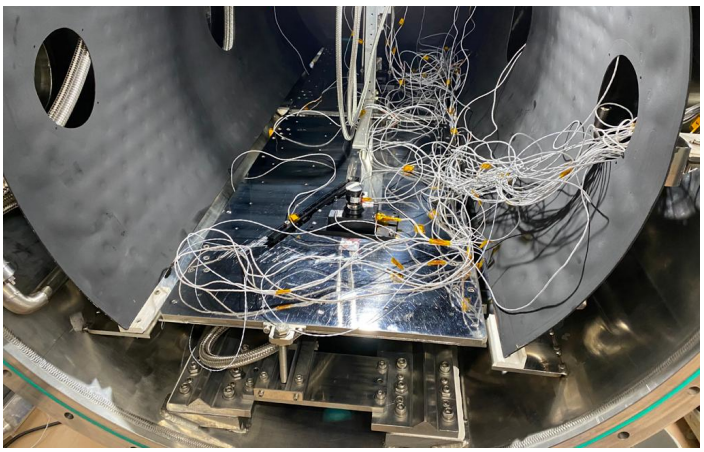

Fig 13: Star sensor mounted inside the thermal-vacuum chamber in MGKM lab, CREST, IIA.

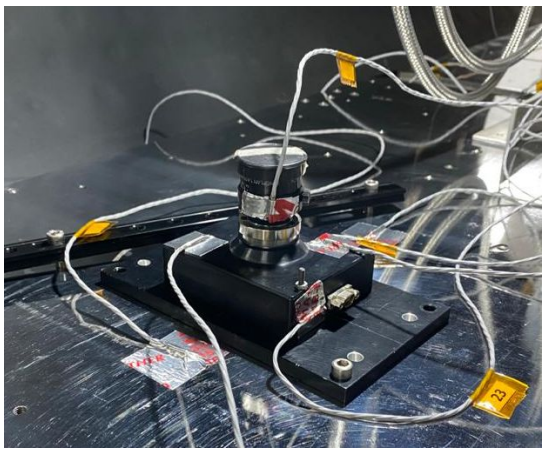

Fig 14: Star sensor with temperature sensors taped on with aluminium tape.

After each hot and cold cycle, the functionality of the payload was checked by turning on the system and capturing dark frames, which were stored onboard the system memory. The camera module and RPi functioned without any errors, and the dark frames were consistent without any

abnormalities. Thus, the payload remained functional and was able to handle the temperature variations.

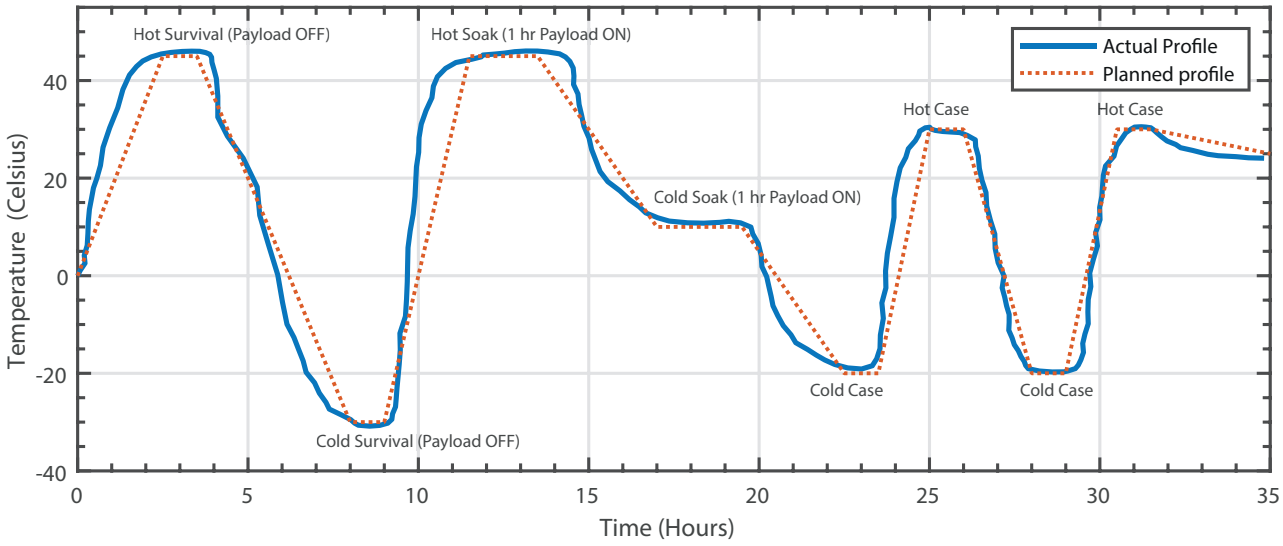

Fig 15: Plot shows the average temperature values recorded from the four sensors mounted on star sensor body during the thermal-vacuum test (Blue continuous) overplotted on the programmed temperature profile (Red dotted).

5.2 Vibration test

The vibration test was conducted as per the requirement of the launch provider (ISRO-PSLV). The launch load profiles are shown in Tables 6 and 7. The payload was subjected to sine and random vibrations according to the launch requirements. Accelerometers were mounted on the *StarberrySense* body as shown in Fig. 16. The *StarberrySense* successfully passed the vibration test without any structural damage, with all the connecting bolts remaining intact. The first natural frequency was found to be 385 Hz (well above the frequency level requirement of PSLV stage 4), and the next five successive frequencies were 585Hz, 911Hz, 1327Hz, 1376Hz and 1655Hz, respectively. After the vibration test, the star sensor was tested in the open sky. We verified that the star sensor operates normally as expected, and distortion parameters were unchanged.

Frequency (Hz)	PSD (g^2/Hz)	Level	Duration	Axis
20	0.001	$9g_{rms}$	1 min	All three axis
68	0.001			
250	0.062			
1000	0.062			
2000	0.015			

Table 6: Random vibration specifications for *StarberrySense*

Frequency	Longitudinal axis		Lateral axis	
	Level	Sweep rate	Level	Sweep rate
10 Hz - 16 Hz	20mm DA	2 oct/min	15mm DA	2 oct/min
16 Hz - 100 Hz	10g	2 oct/min	6g	2 oct/min

Table 7: Sinusoidal specifications for *StarberrySense*

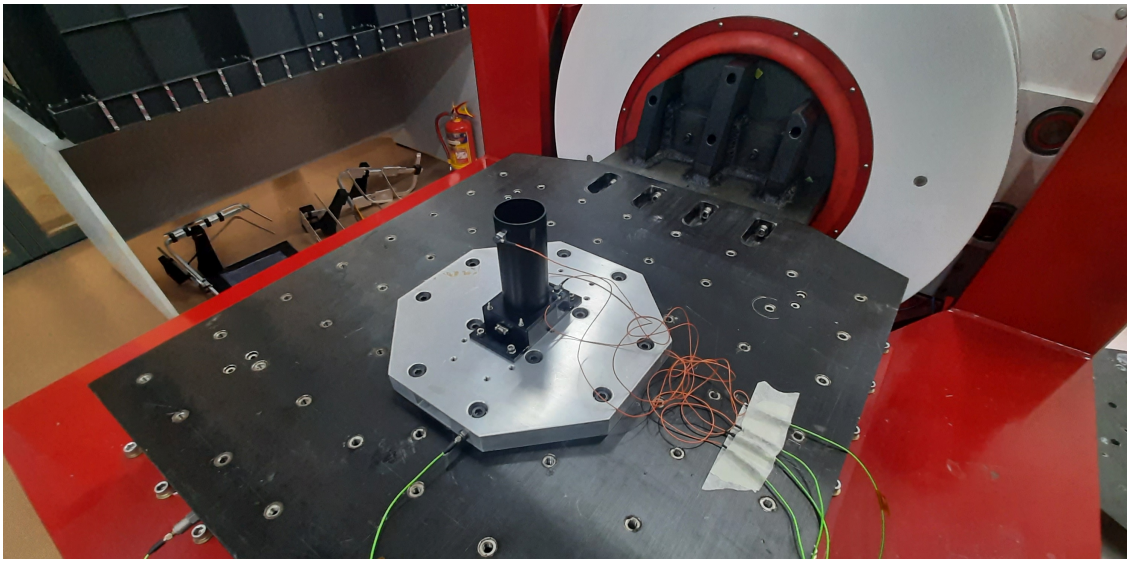

Fig 16: *StarberrySense* mounted on the vibration table with accelerometers attached to the body in the MGKM lab, CREST, IIA.

6 Conclusions and Future Work

In this paper, we have presented the design and development of *StarberrySense*, a low-cost star sensor based on Raspberry Pi Zero. The star sensor was assembled, aligned, calibrated and tested in-house. A computationally-efficient algorithm has been implemented and tested against the real sky conditions, and the results were favourably compared with obtained values from Astrometry.net. The thermal-vacuum test for the star sensor was completed successfully, along with the vibration test as per PSLV launch requirements. The *StarberrySense* was selected by the recent Announcement of Opportunity call for payloads to fly on the PS4-Orbital Platform by ISRO. PS4-Orbital Platform is an innovative use of the spent PS4 stage (fourth stage of the PSLV) as it has standard interfaces/packages for power and telemetry and can be stabilized. Thus, it can carry the scientific payloads to perform scientific experiments for up to 6 months in LEO. We are preparing the payload for flight before the end of 2022 on board the ISRO PSLV stage-4 platform.
Further improvements in the next version of the *StarberrySense* include an up-gradation of the hardware processing capability with the newer Rpi Zero 2 as well as reduction in form factor and weight of the payload. In addition to that, an improvement in the optics is envisaged to enhance the light collection area and thereby reducing the exposure time.

7 Acknowledgement

We wish to thank Mr. S. Sriram of the Indian Institute of Astrophysics (IIA) and Mr. Ajin Prakash from Arksa Research Labs, Bangalore, India, for their valuable suggestions and help. MS acknowl-

edges the financial support by the Department of Science and Technology (DST), Government of India, under the Women Scientist Scheme (PH), project reference number SR/WOS-A/PM-17/2019. We also thank all the staff at the M. G. K. Menon laboratory (CREST) for helping us with assembly and storage of payload components in the cleanroom environment.

References

[1] C. Liebe, "Star trackers for attitude determination," *Aerospace and Electronic Systems Magazine, IEEE* **10**, 10 – 16 (1995).

[2] T. Reid, *Orbital Diversity for Global Navigation Satellite Systems*. PhD thesis (2017).

[3] S. Lee, R. Saleem, and S.-S. Lee, "Micro star tracker with a curved vane for a short baffle length and sharp stray light attenuation," *Appl. Opt.* **59**, 4131–4142 (2020).

[4] M. Sarpotdar, J. Mathew, A. Sreejith, *et al.*, "A software package for evaluating the performance of a star sensor operation," *Experimental Astronomy* **43** (2016).

[5] M. Sarpotdar, J. Mathew, A. Sreejith, *et al.*, "Design and development of a star sensor cum asteroid tracker," *Space Telescopes and Instrumentation 2016: Optical, Infrared, and Millimeter Wave, SPIE Astronomical Telescopes + Instrumentation* , 9904–195 (2016).

[6] S. Gutiérrez, C. Fuentes, and M. Diaz, "Introducing SOST: An Ultra-Low-Cost Star Tracker Concept Based on a Raspberry Pi and Open-Source Astronomy Software," *IEEE Access* **8**, 166320–166334 (2020).

[7] Raspberry Pi Zero, "Raspberry Pi Foundation.," *http://raspberrypi.org* (2000).

[8] T. Dzamba, J. Enright, D. Sinclair, *et al.*, "Success by 1000 improvements: Flight qualification of the st-16 star tracker," (2014).

[9] B. Seng, "Space Sensor Commercialization – A small company approach." https://www.osti.gov/servlets/purl/1249037.

[10] "Highly accurate, flexible, robust and scalable multicamera system for spacecraft autonomous attitude determination through low cost cameras." https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5adea83b5&appId=PPGMS.

[11] M. Kolomenkin, S. Pollak, I. Shimshoni, *et al.*, "Geometric voting algorithm for star trackers," *Aerospace and Electronic Systems, IEEE Transactions on* **44**, 441 – 456 (2008).

[12] G. Wahba, "A least squares estimate of satellite attitude," *Society for Industrial and Applied Mathematics* , Problem 65–1 (1966).

[13] M. A. C. Perryman, L. Lindegren, J. Kovalevsky, *et al.*, "The hipparcos catalogue.," *Astronomy and Astrophysics* **500**, 501–504 (1997).

[14] M. Pagnutti, R. Ryan, G. Cazenavette, *et al.*, "Laying the foundation to use raspberry pi 3 v2 camera module imagery for scientific and engineering purposes," *Journal of Electronic Imaging* **26**, 013014 (2017).

[15] M. Asadnezhad, A. Eslamimajd, and H. Hajghassem, "Optical system design of star sensor and stray light analysis," *Journal of the European Optical Society* **14** (2018).

[16] R. M. Salinas, "RaspiCam: C++ API for using Raspberry camera with/without OpenCv." http://www.uco.es/investiga/grupos/ava/node/40.

[17] B. Spratling and D. Mortari, "A survey on star identification algorithms," *Algorithms* **2** (2009).

[18] P. SALOMON and W. GOSS, "A microprocessor-controlled ccd star tracker," *14th Aerospace Sciences Meeting* (1976).

[19] B. Quine and H. F. Durrant-Whyte, "Rapid star-pattern identification," *Acquisition, Tracking, and Pointing X, Proc. SPIE* **2739**, 351 – 360 (1996).

[20] E. Heide, M. Kruijff, S. Douma, *et al.*, "Development and validation of a fast and reliable star sensor algorithm with reduced data base," *International Astronautical Congress* (1998).

[21] A. Nabi, Z. Ahmed-Foitih, and M. E.-A. Cheriet, "Improved triangular-based star pattern recognition algorithm for low-cost star trackers," *Journal of King Saud University - Computer and Information Sciences* **33**(3), 258–267 (2021).

[22] G. Lamy Au Rousseau, J. Bostel, and B. Mazari, "Star recognition algorithm for aps star tracker: oriented triangles," *IEEE Aerospace and Electronic Systems Magazine* **20**(2), 27–31 (2005).

[23] D. Mortari, M. Samaan, C. Bruccoleri, *et al.*, "The pyramid star identification technique," *NAVIGATION* **51** (2004).

[24] Q. Fan and X. Zhong, "A triangle voting algorithm based on double feature constraints for star sensors," *Advances in Space Research* **61**(4), 1132–1142 (2018).

[25] C. Liebe, "Accuracy performance of star trackers - a tutorial," *IEEE Transactions on Aerospace and Electronic Systems* **38**(2), 587–599 (2002).

[26] H. D. BLACK, "A passive system for determining the attitude of a satellite," *AIAA Journal* **2**(7), 1350–1351 (1964).

[27] J. E. Keat, "Analysis of least-squares attitude determination routine doaop," *Computer Sciences Corporation* (1977).

[28] F. Markley, "Attitude determination using vector observations and the singular value decomposition," *The Journal of the Astronautical Sciences* **36**(3), 245–258 (1988).

[29] A. O. Erlank, *Development of CubeStar A CubeSat-Compatible Star Tracker*. PhD thesis (2013).

[30] G. Wahba, "A least squares estimate of satellite attitude," *SIAM Review* **7**(3), 409–409 (1965).

[31] C. C. Liebe, "Accuracy performance of star trackers - a tutorial," *IEEE Transactions on Aerospace and Electronic Systems* **38**(2), 587–599 (2002).

[32] D. Lang, D. W. Hogg, K. Mierle, *et al.*, "Astrometry. net: Blind astrometric calibration of arbitrary astronomical images," *The Astronomical Journal* **139**(5), 1782 (2010).

[33] D. Shupe, M. Moshir, J. Li, *et al.*, "The sip convention for representing distortion in fits image headers," *ASP Conf. Ser.* **347**, 491 (2005).

[34] F. Markley and J. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*, Space Technology Library, Springer New York (2014).

[35] J. E. Bortle, "Gauging light pollution: The bortle dark-sky scale," *Sky & Telescope* **Retrieved 2020-05-29** (2001).

[36] J. Mathew, A. Prakash, M. Sarpotdar, *et al.*, "Prospect for UV observations from the Moon. II. Instrumental design of an ultraviolet imager LUCI," *Astrophysics and Space Science* **362**, 1–11 (2017).

[37] A. Kumar, S. K. Ghosh, J. Hutchings, *et al.*, "Ultra Violet Imaging Telescope (UVIT) on ASTROSAT," in *Space Telescopes and Instrumentation 2012: Ultraviolet to Gamma Ray*, T. Takahashi, S. S. Murray, and J.-W. A. den Herder, Eds., *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* **8443**, 84431N (2012).

[38] A. Kandala, P. Hari, H. Simha, *et al.*, "Design and development of a ps4-op payload for solar spectral irradiance measurements and technology demonstration of small-satellite subsystems," (2021).