Development of star image simulator for star sensor algorithm validation

N S Ardi*, M A Saifudin2, R E Poetro1, L Fathurrohim1

1 Faculty of Mechanical and Aerospace Engineering, Bandung Institute of Technology, Indonesia.
2 Satellite Technology Center, National Institute of Aeronautics And Space Indonesia (LAPAN)
*nugrahaardi53@s.itb.ac.id

Abstract. The use of star sensor, which is the most accurate attitude sensor on satellites, began to penetrate into micro satellites and nano satellites. Thus the need arises to research and develop star sensor independently to meet the specific design of micro satellites and nano satellites. There are mainly three approaches in star sensor research: digital simulation, hardware in the loop simulation, and field test of star observation. In digital simulation approach, all of processes are done in a software, including star image simulation. Hence, it is necessary to develop star image simulation software which could simulate real space environment and various star sensor’s configuration. This paper focuses on star image simulation results done in various parameters: resolution, defocus level, stars’ magnitude limit, background noises, FOV, unexpected objects and missing stars. Those parameters are needed to test stars pattern recognition’s robustness. The results show that the star image simulation is able to simulate all those parameters.

1. Introduction
The attitude determination is a critical part of any satellites that requires to know its orientation [1]. There are several sensors for attitude determination, such as: inertial sensors, horizon sensors, sun sensors, and star tracker. Star trackers are an essential sensor component of spacecraft attitude control systems due to high accuracy [2]. Therefore, ground simulation testing plays important role to reduce the riskiness of failure [3]. Affordable, robust star trackers have the potential to be an enabling technology for nano-micro satellite missions. The main disadvantage of star tracker is that they remain expensive when compared to other sensors [4]. In a quest for cost reduction, development of independence star tracker sensor have to carry out. The algorithm is the main key of the star sensor. Many methods are developed and used by researchers. There are mainly three approaches in star sensor research and development in particular for algorithm validation [5]: digital simulation, hardware in the loop simulation, and field test of star observation. The star sensor was tested by field test of star observation in the night sky necessarily. In fact, there are many constraints for star tracker developers to have a field test. First, the stars appear in the day night, then they must wait until dark sky to see the stars. Second, the weather condition is good enough, means the sky has clouds free. Third, the location of test is free from light pollution. Those problems could have the star tracker development are time consumed and costly. Therefore, the first approach that is digital simulation
using star image simulator is more suitable. In this paper, the development of star image simulator for star sensor algorithm testing is presented. The scope of this work includes developing the static star image by using SAO J2000 star catalogue and testing the star simulator by using Multitriangle 2 algorithm [6].

2. Methodology
The methodology of this work is depicted in figure 1 [7].

![Methodology flowchart](image)

**Figure 1.** Methodology flowchart.

2.1 Design
The design phase consists of features definition, mathematical modelling, and Graphical User Interface (GUI) of star image simulator.

2.1.1 Features
The star image simulator should be able to simulate some possible disturbances found in space environments and some possible image sensor’s specification. The disturbances include background noises, unexpected objects (planets, moon, space debris, etc), and missing stars (caused by bad sensitivity of image sensor or bad camera’s lens), while image sensor’s specifications are resolution, image sensor’s sensitivity, star image’s defocus level, and camera’s FOV. Hence star should have those features.

2.1.2 Mathematical modelling
Generally, there are two steps in simulating star image: converting star’s vector position from celestial sphere coordinate system (inertial coordinate system) to star sensor coordinate system, and composing every pixel of star image. Before converting star’s vector position, we need select stars which are within FOV, which satisfy [5]:

\[
\alpha \in \left\{ \alpha_0 - \frac{R}{\cos \delta_0}, \alpha_0 + \frac{R}{\cos \delta_0} \right\} \\
\delta \in \left\{ \delta_0 - R, \delta_0 + R \right\}
\]

where \((\alpha, \delta)\) is star’s right ascension and declination, \((\alpha_0, \delta_0)\) is camera’s boresight, and \(R\) is diagonal length of FOV as shown in figure 2.

![The relation between R, FOVx, and FOVy](image)

**Figure 2.** The relation between R, FOVx, and FOVy.
where $FOV_x$ is camera’s horizontal field of view and $FOV_y$ is camera’s vertical field of view. Thus $R$ is

$$R = \sqrt{FOV_x^2 + FOV_y^2} \quad (2)$$

Then the selected star’s position vector is converted from celestial sphere coordinate system to star sensor’s coordinate system using the following steps [5]:

$$\begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ \tilde{z}_i \end{bmatrix} = \begin{bmatrix} \cos \alpha_i \cos \delta_i \\ \sin \alpha_i \cos \delta_i \\ \sin \delta_i \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = M^T \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ \tilde{z}_i \end{bmatrix} \quad (4)$$

where $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)$ is star $i$’s position vector in celestial sphere coordinate system, $(\alpha_i, \delta_i)$ is star $i$’s right ascension and declination, $(x_i, y_i, z_i)$ is star $i$’s position vector in star sensor coordinate system, and $M$ is rotational matrix defined by

$$M = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \quad (5)$$

where

$$a_1 = \sin \alpha_0 \cos \varphi_0 - \cos \alpha_0 \sin \delta_0 \sin \varphi_0$$

$$a_2 = -\sin \alpha_0 \sin \varphi_0 - \cos \alpha_0 \sin \delta_0 \cos \varphi_0$$

$$a_3 = -\cos \alpha_0 \cos \delta_0$$

$$b_1 = -\cos \alpha_0 \cos \varphi_0 - \sin \alpha_0 \sin \delta_0 \sin \varphi_0$$

$$b_2 = \cos \alpha_0 \sin \varphi_0 - \sin \alpha_0 \sin \delta_0 \cos \varphi_0$$

$$b_3 = -\sin \alpha_0 \cos \delta_0$$

$$c_1 = \cos \alpha_0 \sin \varphi_0$$

$$c_2 = \cos \alpha_0 \cos \varphi_0$$

$$c_3 = -\sin \delta_0$$

where $(\alpha_0, \delta_0, \varphi_0)$ is star sensor’s right ascension, declination, and roll relative to celestial sphere coordinate system. The relation between celestial sphere’s coordinate system and star sensor’s coordinate system as illustrated in figure 3.
Then after converting stars’ position vectors, the next step is composing star image. Generally, every pixel’s intensity in star image is composed by the sum of random background noises and an intensity that follow Gaussian distribution function. Mathematically, the composition could be written as

$$I(x, y) = f(x, y) + r(x, y)$$  

where $I(x, y)$ is intensity in every pixel of star image, $f(x, y)$ is intensity which follows Gaussian distribution function [8]

$$f(x, y) = \frac{H}{2\pi\sigma^2} \exp\left(-\frac{(x_i - x_0)^2 + (y_i - y_0)^2}{2\sigma^2}\right)$$

where $(x_i, y_i)$ is pixel’s coordinate, $(x_0, y_0)$ is their center coordinate, $\sigma$ is defocus level, and $H$ is a constant proportional to star’s magnitude. In this paper, we use form

$$H = K_1 \exp(-K_2 Mv + K_3)$$

where $K_1, K_2, K_3$ are constant, and $Mv$ is magnitude of star. We use $K_1 = 1000, K_2 = 1$, and $K_3 = 1$, but those values could be adjusted depends on image sensor’s characteristic we want to simulate. While $r(x, y)$ is random background noises, and could be written as

$$r(x, y) = f_{\text{noises min}} + (f_{\text{noises max}} - f_{\text{noises min}})\text{rand}(1, 1)$$

2.1.3 Graphical User Interface (GUI)
An early work of GUI version of this software is depicted in figure 4.
The GUI has many inputs. Right ascension, declination, and roll input define star sensor’s attitude \((\alpha_0, \delta_0, \varphi_0)\). \(FOV_x\) and \(FOV_y\) define camera’s horizontal field of view and vertical field of view. Then width and length define resolution of star image. Background noises could be turn off or turn on, and Gmin and Gmax define \(f_{\text{noises min}}\) and \(f_{\text{noises max}}\). Magnitude limit define star’s magnitude limit which would be displayed in star image. Star whose magnitude is above magnitude limit would not be displayed. In Gaussian Distribution section, we can vary the value of \(\sigma\) (defined by sigma input), and constants (defined by \(K_1, K_2, K_3\)). In Random Unexpected Objects section, we can vary its number by setting Numb min (minimum number) and Numb max (maximum number), and vary its size or radius by setting Rad min (minimum radius) and Rad max (maximum radius). In Random Missing Stars section, we can vary its minimum number (defined by n min) and maximum number (defined by n max).

2.2 Programming

The program of the star image simulator and the GUI was written in Matlab 2014a and tested on ordinary workstation.

2.3 Functional test

In the functional test process, star image simulator displays star image as shown in section 3.1. To test its function, a star pattern recognition algorithm is validated using one star image using digital simulation approach.

2.4 Algorithm validation

The last methodology of this work is validate the star pattern recognition algorithm by digital simulation. The flowchart of digital simulation is shown in figure 5.

![Digital simulation flowchart](image)

**Figure 5.** Digital simulation flowchart.
First, random input boresights are used to select stars which are within FOV from the star catalogue. Second, those stars are displayed in star image simulation. Third, star spot centroiding algorithm will calculate coordinates of star’s center from star image. Fourth, stars pattern is generated from those coordinates, and being matched with star pattern from stars catalogue. Finally, if the matched stars pattern from stars catalogue has been found, then attitude estimation could be done. The result is shown in section 3.2.

3. Results and discussion
In this section, both star image simulation results and star pattern recognition algorithm validation will be discussed.

3.1 Star image simulation results
This star image simulation results are done using boresight input Right Ascension (\(\alpha\)) = 0°, Declination (\(\delta\)) = 10°, and Roll (\(\varphi\)) = 30°.

3.2 Variation of resolution
In this results, star images are simulated using defocus level (\(\sigma\)) = 1.5 and star magnitude less than 6 of SAO J2000 star catalogue. Figure 5 and figure 6 show that with the increasing resolution, the stars image will be smoother. Moreover, increasing the resolution produce more accurate in centroiding results [9]. It will lead to increase the success rate of star pattern matching. However, increasing the accuracy of star spot centroiding by improving imager’s resolution has a limit. Therefore, the defocus process is done on star images to get better sub pixel accuracy [10].

![Figure 6. Star image with 1024x1024 resolution.](image)

![Figure 7. Star image with 2048x2048 resolution.](image)

3.3 Variation of defocus level (variation of \(\sigma\))
In this results, star images are simulated using 1024\times1024 resolution and stars’ magnitude are less than 6 with SAO J2000 star catalogue.
From the figure 8 and 9, higher value of $\sigma$ will lead to be more defocused, and vice versa.

3.4 Variation of star’s magnitude limit.
In this results, star images are simulated using 1024×1024 resolution, $\sigma = 1.5$, and without background noises.

Figure 10 shows that with higher stars’ magnitude limitation, there are more stars appeared compare to lower stars’ magnitude limit. Typical star sensor camera has star’s magnitude limit 6.

3.5 Variation of background noises
In these results, star images are simulated using 1024×1024 resolution, $\sigma = 1.5$, and stars’ magnitude less than 6 of SAO J2000 star catalogue.
Figure 11. Star image with various of background noises.

Figure 11 shows star image without (left) and with (right) background noises. Background noises can be adjusted by thresholding the process, i.e., if the intensity of a pixel is less than the intensity threshold, then the intensity of the pixel is set to zero.

3.6 Variation of FOV
In these results, star images are simulated using 1024×1024 resolution, σ = 1.5, and stars’ magnitude less than 6 with SAO J2000 star catalogue.

Figure 12. Star image with FOV 15° × 15° at (a), 25° × 25° at (b).

Figure 12 shows that with higher FOV value there are more stars displayed in the star image. One of the major advantages of camera with big FOV is that more bright stars are captured, so the success rate of star pattern matching is increasing. But the drawback is the star center position from star spot centroiding process is less accurate, so the result of attitude estimation also becomes less accurate.

3.7 Unexpected objects
In these results, star images are simulated using 1024×1024 resolution, σ = 1.5, stars’ magnitude less than 6 with SAO J2000 star catalogue.
Figure 13. Star image without (left) and with (right) random unexpected objects.

Figure 13 shows random unexpected objects such as planets, moon, etc. The number of unexpected objects, their size (radius in pixel), their position, and their intensity could be generated randomly or set manually. In figure 11 the unexpected objects are generated arbitrarily.

3.8 Missing stars
In these results, star images are simulated using 1024×1024 resolution, \( \sigma = 1.5 \), and stars’ magnitude less than 6 with SAO J2000 star catalogue.

Figure 14. Star image without random missing stars (left) and with random missing stars (right).

Figure 14 shows at left side star image without missing stars, and right side star image with some random missing stars. In that figure, the number of stars and the stars that are removed are chosen arbitrarily.

3.9 Star pattern recognition algorithm validation
Some star image with various configuration are used to validate the algorithm.

- Configuration 1:
  - Right ascension, declination, roll: 30°, 70°, 30°.
  - FOV: 17°×17°.
  - Image resolution: 1024×1024.
  - Star’s magnitude limit: 6.
  - Background noises: yes.
  - Gaussian distribution: \( \sigma = 2, K_1 = 1000, K_2 = 1, K_3 = 1 \).
  - Random unexpected objects: no.
  - Random missing stars: no.
Figure 15. Star image 1 to be used as validation.

The red dots in figure 15 is the stars detected by star spot centroiding algorithm. Using the above star image configuration, the algorithm is capable to calculate attitude. The errors in right ascension, declination, and roll are 0.0013°, 0.0012°, and 0.0425°.

- Configuration 2:
  - Right ascension, declination, roll: 30°, 70°, 30°.
  - FOV: 17° × 17°.
  - Image resolution: 1024 × 1024.
  - Star’s magnitude limit: 6.
  - Background noises: yes.
  - Gaussian distribution: σ = 2, K₁ = 1000, K₂ = 1, K₃ = 1.
  - Random unexpected objects: yes.
  - Random missing stars: no.

Figure 16. Star image 2 to be used as validation.

Using figure 16, although the unexpected objects are detected as stars by star spot centroiding algorithm, the algorithm is still capable to calculate attitude. This is because Multitriangle 2 algorithm works by choosing stars whose positions are closest to the center of FOV [6]. In figure 16, the unexpected objects are far from the center of FOV, hence the algorithm still works. The errors in right ascension, declination, and roll are 0.0011°, 0.000859°, and 0.0331° respectively.

- Configuration 3:
  - Right ascension, declination, roll: 30°, 70°, 30°.
FOV: 17°×17°.
- Image resolution: 1024×1024.
- Star’s magnitude limit: 6.
- Background noises: no.
- Gaussian distribution: \( \sigma = 2, K_1 = 1000, K_2 = 1, K_3 = 1 \). 
- Random unexpected objects: yes.
- Random missing stars: no.

**Figure 17.** Star image 3 (right) with some missing stars and star image without missing stars (left)

Using figure 17 with some missing stars, the algorithm fails to do pattern recognition. This is because the missed stars’ position are close to the center of FOV [6], hence Multitriangle 2 algorithm does not work.

### 4. Conclusion

From the results and discussions above, the star image simulation shows that it may simulate some possible disturbances captured by star sensor while operating in space environment such as unexpected objects, missing stars, and background noises. Furthermore, it could simulate some possible configuration of star sensor’s camera such as image sensor’s resolution, FOV, defocus level, and image sensor’s sensitivity to star’s magnitude limit. Besides that, the simulated star image is able to be used to validate one of star pattern recognition algorithm. The algorithm works and fails in some cases. Thus those simulated star images could be used to validate various star pattern recognition. There are actually some disturbances that have not been simulated in this paper such as dead pixel and blurred star image caused by rotation of spacecraft. In the future work, we will implement this software to hardware in the loop test for star sensor.

### References

[1] T. Delabie, "Star Tracker Algorithms and a Low-Cost Attitude Determination and Control System for Space Missions," 2016.
[2] C. C. Liebe, "Accuracy Performance of Star Trackers – A Tutorial".
[3] J. A. Tappe, "Development of Star Tracker System for Accurate Estimation of Spacecraft Attitude," 2009.
[4] S. Dikmen, "Development of Star Tracker Attitude and Position Determination System for Spacecraft Maneuvering and Docking Facility." 2016.
[5] G. Zhang, Star Identification Methods, Techniques, and Algorithms, Berlin: Springer, 2017.
[6] N. S. Ardi, Characteristics Study and Robustness Comparison of Various Star Pattern Recognition Algorithm for Star Sensor using Digital Simulation, 2018.
[7] M. A. S. Muh. Sulaiman Nur Ubay, "Analisis Perangkat Lunak Simulator Bintang Untuk Pengujian Star Sensor (Analysis Software Star Simulator For Star Sensor Testing)," *Jurnal Teknologi Dirgantara*, vol. 12, pp. 92-101, 2014.

[8] L. H., W. H.-y. QIAN Hua-ming, "Design and verification of star-map simulation software," in *International Conference on Intelligent Computation Technology and Automation*, 2015.

[9] K. M. HUFFMAN, "Designing Star Trackers to Meet Micro-satellite Requirements," 2006.

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