Development of Star Pattern Recognition Algorithm for Star Sensor Using $C^3_n$ Combination of Triangles Pattern

N S Ardi¹, R E Poetro¹, M A Saifudin², L Fathurohim¹

¹ Faculty of Mechanical and Aerospace Engineering, Bandung Institute of Technology, Indonesia.
² Indonesian National Institute of Aeronautics And Space (LAPAN)

E-mail: nugrahaardi53@gmail.com

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 two aspects to be considered in star sensor: hardware and software (algorithm). Star sensor’s algorithm is also known as star pattern recognition algorithm. This paper deals with development of star pattern recognition algorithm using $C^3_n$ combination of triangles pattern, that is when there are $n$ stars to be used to do pattern recognition, then $C^3_n$ combination of triangles pattern would be generated. Success rate test are conducted in digital simulation approach using various number of stars that be used to do pattern recognition and various of star image resolution to test its robustness against star spot centroiding error. Results show that this algorithm has high level of success rate even on low resolution of star image, but has increased processing time when more stars are used to do pattern recognition.

Keyword: attitude sensor, star sensor, star pattern recognition.

1. Introduction

Star sensor is the most accurate attitude sensor in satellite [1], hence it is the most researched attitude sensor among others such as sun sensor, magnetometer, horizon sensor, etc [2]. It has two aspects, namely hardware and software (algorithm) [3]. Hardware aspect consists of baffle to minimize disturbance light, optical lens to focus the image, and imager or image sensor to capture the star image [2]. Algorithm aspect consists of star spot centroiding algorithm to detect position of star’s center [4], star pattern recognition algorithm to match pattern from star image with GSC (Guide Star Catalogue) [5], and attitude estimation algorithm to estimate the attitude of star sensor [6]. In algorithm aspect, star pattern recognition algorithm is widely researched because it plays important role: Firstly, a good star sensor should be able to accurately match star image’s pattern with star pattern from GSC [7]. If matching process failed, then star sensor failed. Secondly, a good star sensor should has fast processing time in matching star image’s pattern with star pattern from GSC to provide first attitude [8]. The simplest method for star pattern recognition is geometric approach using three stars. In this paper, the development of star pattern recognition using $C^3_n$ combination of triangle patterns is presented. The triangle combination is subjected to perform the accuracy. The scope of this work is focused on star pattern recognition and attitude calculation is not presented. The algorithm is performed by digital simulation. The performance of the algorithm is measured by success rate of pattern matching and its processing time for various star image resolutions.
2. Methodology

The methodology used in this paper is digital simulation. Digital simulation is an approach in star sensor research which all of its processes are done in a software [9]. The flowchart of digital simulation is depicted by figure 1.

![Flowchart of digital simulation.](image)

Firstly, random input boresight consists of right ascension (\(\alpha\)), declination (\(\delta\)), and roll (\(\varphi\)) is used to create star image from star catalogue. We use SAO J2000 Star Catalogue with star whose magnitude is less than 6 to simulate the star image. Then the star image is processed to get star’s center position using a star spot centroiding algorithm. In this paper we use centroid method [9]. Then star pattern are generated from star image and matching process is done. If the matched pattern is within FOV, then pattern recognition process is success, and if not, then it is failed. Those processes are done 1000 times for every different boresight to represent all sky direction. Then success rate is calculated by dividing the number of success by 1000. Digital simulation is done for various star image resolutions.

3. Star Pattern Recognition Algorithm

The main idea of this algorithm is using \(C^n_3\) combination of triangles pattern, that is when there are \(n\) stars to be used to do pattern recognition, then there will be \(C^n_3 = \frac{n!}{(n-3)!3!}\) triplet combination of stars to create triangle patterns. For example if there are 4 stars, namely star 1, star 2, star 3, and star 4, then the triangle patterns is formed by star 1-2-3, 1-2-4, 2-3-4, and 1-3-4. The steps are as follows

a. Choose a closest star to the center of FOV, namely central star.

b. Choose \(n - 1\) closest stars to the central star, namely star 2, star 3, ..., star \(n\).

c. Create all possible triplet combination of stars. A simple algorithm to generate all possible triplet combination is as follows

```
for i = 1 : n-2
    for j = i+1 : n-1
        for k = j+1 : n
            combination = [i j k]
```

...
For example, if $n = 5$ then the pattern is depicted by Figure 2.

Figure 2. Generated pattern for 5 stars in star image.

d. Compute all triplet planar angles in every triangle. Then stored in an array matrix $\theta_s$ which has size $C_n^3 \times 3$

$$\theta_s = \begin{bmatrix} \theta_{s11} & \theta_{s12} & \theta_{s13} \\ \theta_{s21} & \theta_{s22} & \theta_{s23} \\ \vdots & \vdots & \vdots \\ \theta_{sC_n^31} & \theta_{sC_n^32} & \theta_{sC_n^33} \end{bmatrix}$$ (3.1)

where the values in matrix’s row had sorted from the smallest to the largest.

e. For every star $i$ in GSC, choose its $n - 1$ neighbour stars. Create all possible triplet combination of stars using step c, and compute all planar angles and stored in an array matrix namely $\theta_{ci}$

$$\theta_{ci} = \begin{bmatrix} \theta_{c11} & \theta_{c12} & \theta_{c13} \\ \theta_{c21} & \theta_{c22} & \theta_{c23} \\ \vdots & \vdots & \vdots \\ \theta_{cC_n^31} & \theta_{cC_n^32} & \theta_{cC_n^33} \end{bmatrix}$$ (3.2)

f. Reduce every $\theta_s$’s row with every $\theta_{ci}$’s row and take its total absolute error. If the error is less than a particular value $\epsilon$, give it a label 1 and if not, then give it a label 0. In this paper we use $\epsilon = 1^\circ$. The labels are stored in an matrix $\Delta_i$ which has size $1 \times C_n^3$. Do the same step for every star $i$ from star catalogue.

g. Select $\Delta_i$ which has the most number of label 1. The selected $\Delta_i$ is the matched pattern.

In brief, this algorithm works with a voting principle. A pattern which has the most number of matched triangle is the matched pattern. Suppose there are two candidates as shown by Figure 3.
Figure 3. Pattern candidate from GSC.

Then the voting scheme is shown as on Figure 4.

Figure 4. Voting scheme between two candidates.

The first column in Fig. 4 is pattern candidate form GSC, while the second column is pattern from star image. The left side is candidate 1 and the right side is candidate 2. The green arrow is for the matched triangle, while red arrow for the unmatched triangle. Candidate 2 has more matched triangle than candidate 2, thus the matched pattern is candidate 2.

This algorithm uses the following two GSC: primary and secondary GSC. Structures are as detailed on Table 1 and Table 2 [10].

Table 1. The structure of primary GSC [10]

| ID Star | Right ascension | Declination |
|---------|-----------------|-------------|
| 1       | $\alpha_1$      | $\delta_1$  |
| 2       | $\alpha_2$      | $\delta_2$  |
| ...     | ...             | ...         |
| n       | $\alpha_n$      | $\delta_n$  |
Table 2. The structure of secondary GSC.

| ID Star | ID neighbour 1 | ID neighbour 2 | ... | ID neighbour m |
|---------|----------------|----------------|-----|----------------|
| 1       | IDn 1          | IDn 2          | ... | IDn m          |
| 2       | IDn 1          | IDn 2          | ... | IDn m          |
| ...     | ...            | ...            | ... | ...            |
| n       | IDn 1          | IDn 2          | ... | IDn m          |

where ID neighbour 1 to ID neighbour m is ordered ascendingly according to its distance from main ID star. We use SAO J2000 Star Catalogue with star whose magnitude is less than 6 as GSC.

4. Results and Discussion

Digital simulation are done using the following hardware specifications:
- **Processor**: AMD FX 9830P 3.00 – 3.50 GHz
- **RAM**: 16 GB
- **Graphic card**: Radeon RX 460 4 GB DDR 4

While the digital simulation setups are as follows:
- **FOV = 17°**
- **Star’s magnitude limit in star image = 6**
- **Number of random input boresights = 1000**
- **GSC = SAO J2000**
- **Star’s magnitude limit in GSC = 6**

Digital simulations are done in various resolution of star image to see its effect toward success rate, because the lower the resolution, star spot centroiding error will be higher thus the pattern would be mistaken, and vice versa. We varied resolution 256×256, 512×512, 1024×1024, and 2048×2048.

The result is shown in Table 3.

Table 3. Digital simulation results.

| n stars          | Resolution     | Success rate (%) | Average Processing time (s) |
|------------------|----------------|------------------|----------------------------|
| 4 (4 triangles)  | 256 × 256      | 58.1             |                            |
|                  | 512 × 512      | 84.8             | 1.428472                   |
|                  | 1024 × 1024    | 93.1             |                            |
|                  | 2048 × 2048    | 94.8             |                            |
| 5 (10 triangles) | 256 × 256      | 75.6             | 4.394685                   |
|                  | 512 × 512      | 91.5             |                            |
|                  | 1024 × 1024    | 97.8             |                            |
|                  | 2048 × 2048    | 98.5             |                            |
| 6 (20 triangles) | 256 × 256      | 87.8             | 12.62772                   |
|                  | 512 × 512      | 96.8             |                            |
|                  | 1024 × 1024    | 98.7             |                            |
|                  | 2048 × 2048    | 99.5             |                            |
| 7 (35 triangles) | 256 × 256      | 92.9             | 30.15096                   |
|                  | 512 × 512      | 99.1             |                            |
|                  | 1024 × 1024    | 99.8             |                            |
|                  | 2048 × 2048    | 99.9             |                            |

From Table 3, the algorithm has an acceptable success rate (93.1%) with only 4 stars and 1024×1024 resolution. While in 5 stars and the same resolution, it has success rate 97.8% but the...
processing time is also increased significantly. If we use 6 and 7 stars in the same resolution, the success rate is not increased significantly, but the processing time is increased alot. The interesting part is this algorithm still has high success rate in 6 stars even on low resolution (512×512 with success rate 96.8%) and in 7 stars has 92.9% for 256×256 and 99.1% for 512×512.

The relation between resolution and success rate for every n stars is depicted by Fig. 5.

![Figure 5. Success rate vs n stars for every star image resolution.](image)

The increased processing time as the increased number of star is due to the fact that this algorithm use $C^n_3$ combination triangle pattern. But processing time could be reduced by eliminating step e and store $\theta_{ci}$ in an GSC, thus the calculation of $\theta_{ci}$ is done beyond algorithm.

5. Concluding Remarks

From the results and discussion above, this algorithm is capable of doing star pattern recognition and has high level of success rate even on low resolution. Hence this algorithm is expected to still work well on low resolution camera. But the processing time is also increased as the number of star that be used is increased. This increasing processing time could be reduced by calculating $\theta_{ci}$ beyond the algorithm and stored it in a GSC. In the future work, we will implement this algorithm in the hardware and test it using both hardware-in-the-loop simulation and field test of observation.

References

[1] C. C. Liebe, “Star trackers for attitude determination,” *IEEE Aerospace and Electronic Systems Magazine*, 1995.
[2] K. M. Huffman, “Designing Star Trackers to Meet Micro-satellite Requirements,” 2006.
[3] S. Dikmen, “Development of Star Tracker Attitude and Position Determination System for Spacecraft Maneuvering and Docking Facility,” 2016.
[4] T. Delabie, “Star Tracker Algorithms and a Low-Cost Attitude Determination and Control System for Space Missions,” 2016.
[5] R. H. T. M Arif Saefudin, “Algoritma Pengenalan Pola Bintang untuk Deteksi Posisi Bintang pada Star Sensor Satelit LAPAN,” *Jurnal Teknologi Dirgantara*, vol. 8, pp. 36–44, 2010.
[6] D. M. F. Landis Markley, “Quaternion Attitude Estimation Using Vector Observations”.
[7] J. A. Tappe, “Development of Star Tracker System for Accurate Estimation of Spacecraft Attitude,” 2009.
[8] J. L. C. Craig L. Cole, “Fast Star-Pattern Recognition Using Planar Triangles,” *Journal of Guidance, Control, and Dynamics*, 2006.
[9] G. Zhang, Star Identification Methods, Techniques, and Algorithms, Berlin: Springer, 2017.
[10] M. A. Saifudin, “Pembuatan Katalog Bintang untuk Aplikasi Star Sensor menggunakan Metode Magnitude Filtering”.
