Star Tracker without a Star Database

Stephen Scott
McMaster University
1280 Main St W, Hamilton, ON L8S 4L8
scotts24@mcmaster.ca

Abstract

A star tracker is a system used to assist a satellite in attitude estimation, which has conventionally been costly to implement and requires complex algorithms to accurately detect star patterns and subsequently calculate attitude vectors. As the popularity of nanosatellites increases, there is a growing need to simplify star tracker algorithms for implementation on university-level projects where budgets are low. In the past, star tracker algorithms have relied on star matching based on star lookup tables; however, in this paper a method is investigated to perform attitude estimation using star field images without a star database. A case study is performed on a dataset of simulated star images using the proposed approach and the results are presented.

1. Introduction

Spacecraft attitude determination is the process by which a satellite uses sensors to estimate its orientation and is important to the satellite control problem because it provides a means for the satellite to adjust its trajectory with respect to a reference vector CITE. The popularity of nanosatellites like CubeSats is increasing, and typically these small platforms are much more limited both with respect to costs and available hardware when compared to conventional satellites. Attitude determination algorithms often use a fusion of data from multiple sensors to determine the best estimate of the satellite's orientation. The star tracker is a candidate for these applications; however, the star tracker can be a costly component to add to a low-budget spacecraft like a CubeSat CITE. Smith suggests that a star tracker can be made from budgetary hardware; however, the software to process the star tracker data in a meaningful manner is a non-trivial task and can propose a barrier to university teams developing nanosatellites.

Star trackers often rely on lookup tables to identify stars; however, in recent years neural networks have been incorporated into the star tracker software stack CITE. The star tracker system involves an optic system used to capture light from stars. An image is developed from the optics and is typically fed through a series of algorithms including star detection and centroiding, star identification, and finally attitude estimation CITE. A crucial part of the overall algorithm is feature extraction from the star detection and centroiding. Another important aspect of the system is the on-board database which stores star features for the identification process CITE. A magnitude threshold is used to populate the database since the photodetectors can only identify stars of a certain magnitude. Also, stars that have centroids too close to one another are ignored.

2. Related Work

One of the recent developments in star identification is a pattern-based feature extraction method called the correlation algorithm which compares the camera images to images in the database by maximizing a cost function. Star locations are represented as Gaussian distributions and the images are correlated with the database image in the image space to attain a heat map CITE.

Another approach to star identification that utilizes image processing is the approach presented by Delabie *et al.* in CITE. Similar to the method proposed by Yoon *et al.* CITE, the method uses image matching techniques to match pairs of images optimally on top of each other such that the stars are aligned. The advantage of this algorithm is that it eliminates the need for computationally expensive coordinate system conversions that are often present in an attitude estimation algorithm. The method uses a shortest distance transform to create a mapping of the inter-star pixel distances for which the Euclidean squared distance metric is used.

These two approaches are similar in that they rely on on-board datasets of images in order to identify stars and subsequently lookup the star coordinates in the database. This introduces an O(n) time complexity to the algorithm as the star lookup time scales linearly with the number of stars in the lookup table.

3. Proposed Method

The proposed method seeks to eliminate the need for a star lookup table by directly classifying attitude vectors from input star images. To accomplish this, the algorithm relies on a dataset of star field images with labelled right ascension and declination values with respect to the center pixel of the image. A limitation of this approach is that it is sensitive to the field of view of images in the dataset. Thus, for the algorithm to be useful for a particular application, the practitioner must create a dataset of images for the field of view corresponding to the camera used for the application. This field of view can be determined from the focal length of the camera CITE. Additionally, as the method is proposed herein, the algorithm is sufficient only for vague attitude estimation as will be described in 5.

3.1. Dataset Generation

For the purpose of validating the proposed algorithm, a simulated dataset of star field images was generated using Stellarium. The dataset has been released open-source with accompanying right ascension and declination labels CITE. Stellarium is a free open-source planetarium that can be used to explore the night sky in a realistic 3D environment CITE. One of the useful features of Stellarium is that it allows the user control over the visible elements in the environment. For the purpose of this simulation, the visible elements were modified to create an environment which mimics that of a satellite in space. An assumption is made that the shutter speed of the camera is low enough such that the backdrop of the milky way and deep space objects are not visible in the images, and thus the corresponding elements are disabled in Stellarium.

After preparing the simulation environment, a Stellarium script was used to sweep through right ascension and declination angles in 5 degree increments from 0 and -90 degrees to 355 and 85 degrees, respectively. Due to this angle sweeping, the image dataset covers the full spherical field of view of the sky. At each increment, an image was captured from Stellarium and the resulting dataset has 2,592 images total. For the purpose of this investigation, the resulting images were further labelled into 4 classes representing the North-East, North-West, South-East, and South-West skies using the right ascension and declination angles. Positive declination angles are used to classify the North sky and right ascension angles less than 180 degrees are used to classify the Eastern sky. An example image from the dataset is shown in Figure 1. This image will be used in further sections to visualize preprocessing and feature extraction steps of the algorithm.

The script used to prepare the dataset and subsequent scripts are available on GitHub CITE.

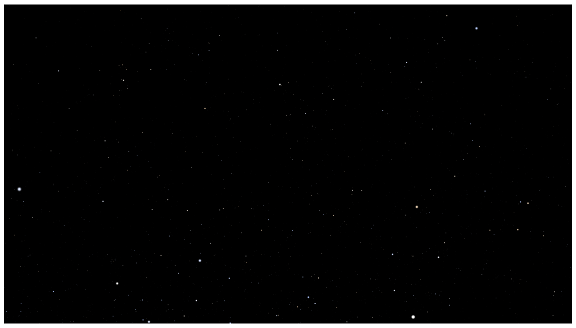


Figure 1. Example image of star field from Stellarium

3.2. Preprocessing

Figure 2. Gaussian blur applied to star image

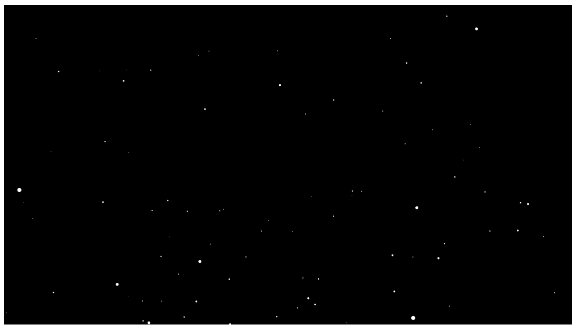


Figure 3. Result of binarizing the Gaussian blurred image

3.3. Feature extraction

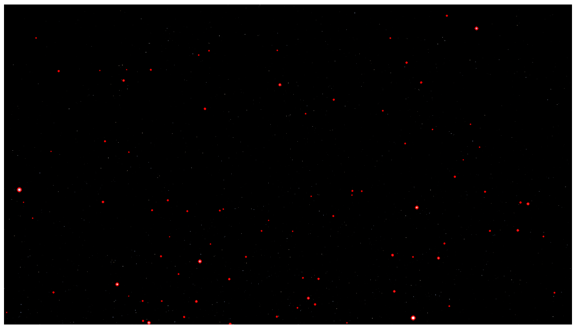


Figure 4. Result of locating global contours

Figure 5. Circular mask around largest detected contour

Figure 6. Result of applying the mask to the binarized image

Figure 7. Result of locating contours in the masked region

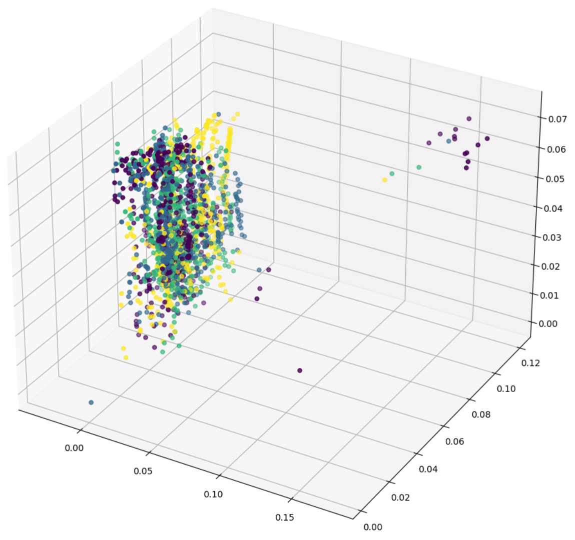


Figure 8. PCA applied to the normalized training data

4. Alternate Method

4.1. Convolutional Neural Network

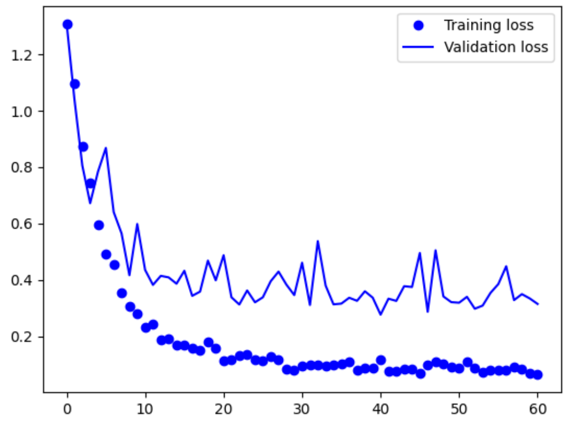


Figure 9. CNN training and validation loss

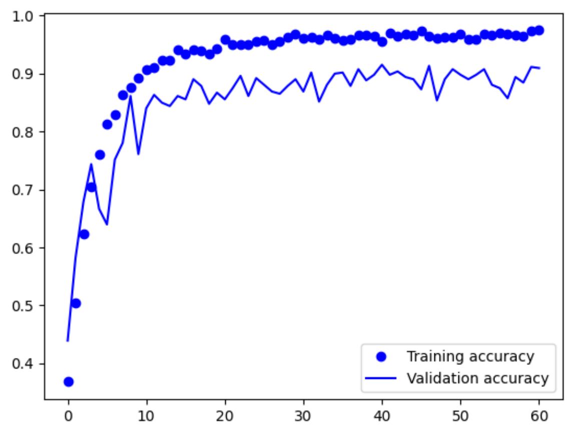


Figure 10. CNN training and validation accuracy

4.2. Dataset Augmentation

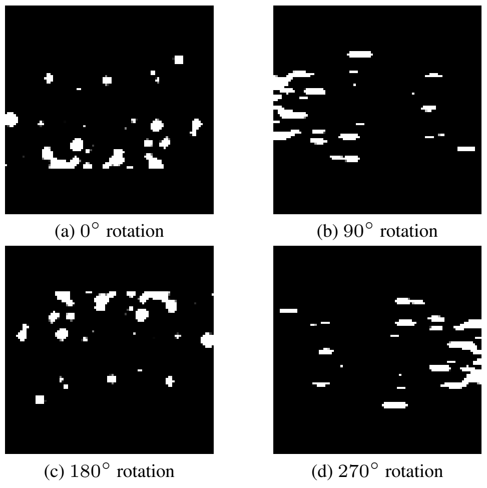


Figure 11. Example of augmented dataset for training the Convolutional Neural Network

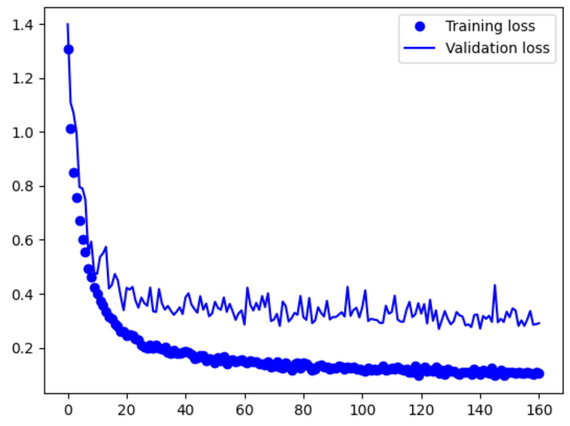


Figure 12. CNN (augmented dataset) training and validation loss

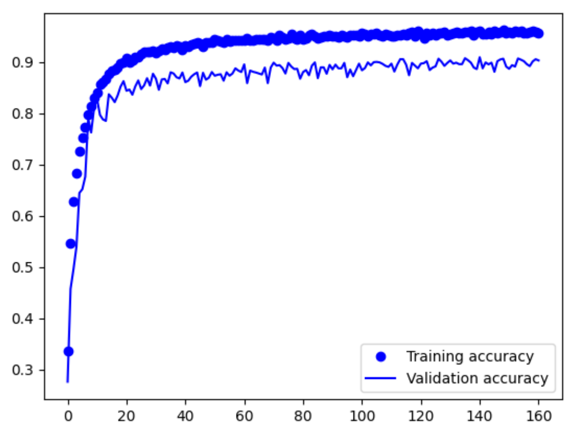


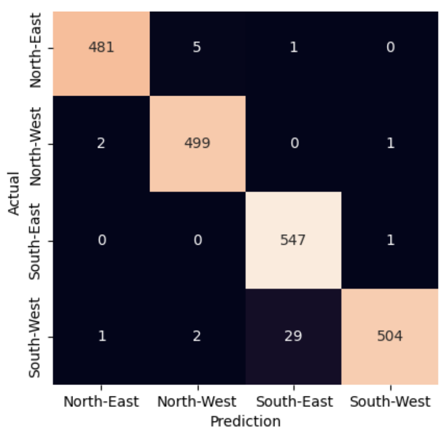
Figure 13. CNN (augmented dataset) training and validation accuracy

5. Results

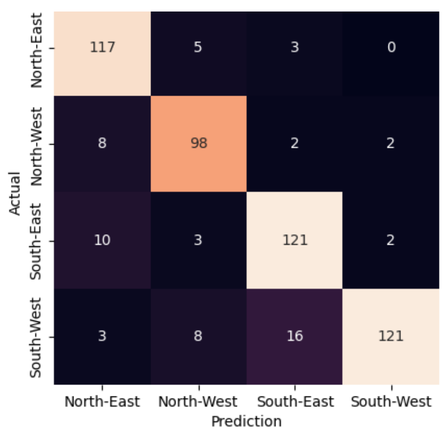
Method	Accuracy		
	Training	Validation	Test
SVM	98.0%	NA	88.1%
CNN	98.6%	90.9%	96.2%
CNN (Augmented)	98.6%	90.4%	91.1%

Table 1. Comparison of classification accuracy for all methods

6. Conclusion

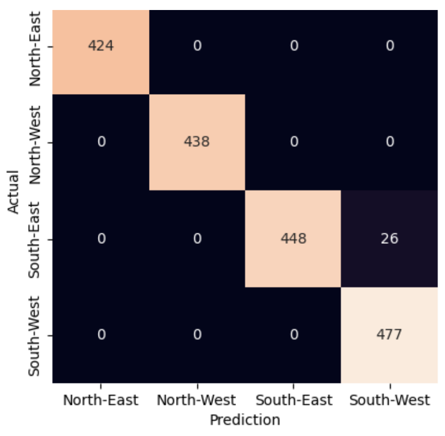


(a) Training data confusion matrix

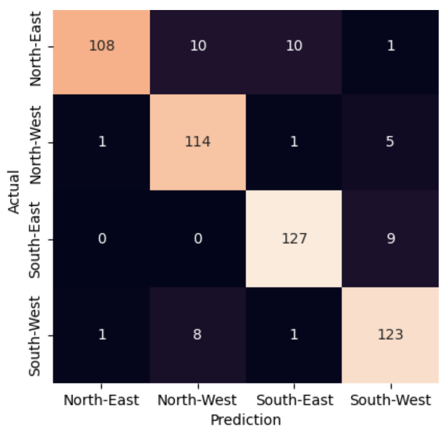


(b) Test data confusion matrix

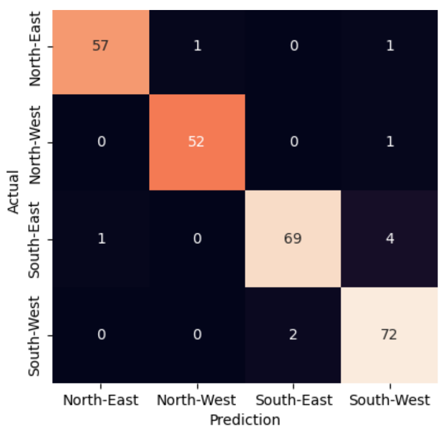
Figure 14. Support Vector Machine classification results



(a) Training data confusion matrix

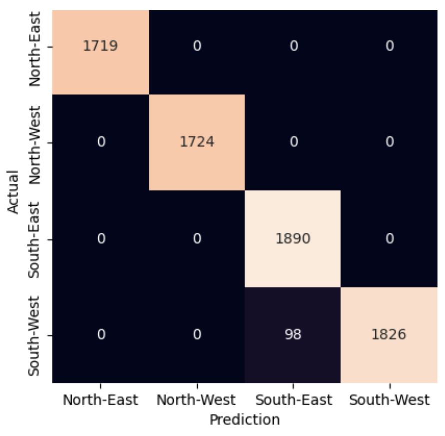


(b) Validation data confusion matrix

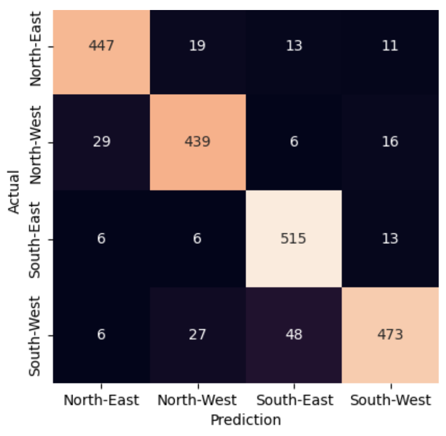


(c) Test data confusion matrix

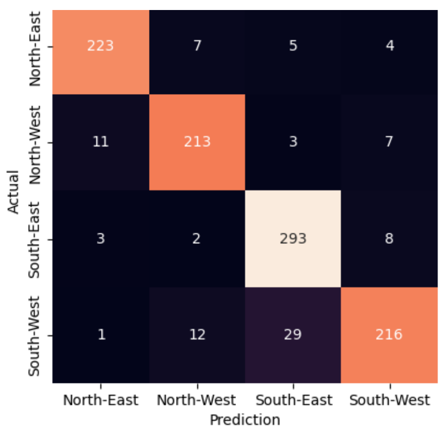
Figure 15. CNN classification results



(a) Training data confusion matrix



(b) Validation data confusion matrix



(c) Test data confusion matrix

Figure 16. CNN (augmented dataset) classification results