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 [2]. 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. 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 [11].

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 [8] [5] [6]. 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 [7]. 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 [7]. 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 on-board 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 [12].

Another approach to star identification that utilizes image processing is the approach presented by Delabie *et al*. in [4]. Similar to the method proposed by Yoon *et al*. [12], 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. Hong and Dickerson have proposed that fuzzy neural logic networks can be used in the star identification algorithm to eliminate the need for an on-board star database [5].

Kim and Bang also propose utilizing a neural network; however, rather than eliminating the star database altogether, they suggest using a neural network to construct the database to maximize lookup efficiency by grouping neighbouring stars [6].

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 is trained using a dataset of star field images with labelled right ascension and declination angles with respect to the center pixel of the image. The dataset is used to train a supervised learning model, and thus the dataset is not required on-board the satellite. 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. Additionally, as the method is described herein, the algorithm is sufficient only for vague attitude estimation as will be described in Section 5.

The algorithm involves feature extraction based on the brightest star in the binary star image, and the Euclidean distance to stars in the neighbourhood of the brightest star. This is similar to the approach presented by Delabie *et al*. in [4]. Using these feature vectors, a Support Vector Machine (SVM) is trained using a simulated dataset of star field images to maximize the distance between data clusters.

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 [10]. Stellarium is a free open-source planetarium that can be used to explore the night sky in a realistic 3D environment [1]. A useful feature of Stellarium is that it allows the user to control 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 [9].

Figure 1. Example image of star field from Stellarium

3.2. Preprocessing

All images are preprocessed by first converting to gray scale. A Gaussian kernel ($\sigma = 2$) is applied to the gray scale images to reduce normal noise which is common in star images. The kernel size of the Gaussian filter is computed based on the sigma value as shown in Equation 1 [3]. The images in the simulated dataset do not have significant white noise; however, the Gaussian blur should be sufficient to eliminate the noise if it were present. The result of applying the Gaussian filter is shown in Figure 2.

$$\begin{aligned} G_i &= \alpha * e^{-(i-(ksize-1)/2)^2/(2*\sigma^2)} \\ i &= 0..ksize-1 \\ \sum_i G_i &= 1 \end{aligned} \tag{1}$$

Figure 2. Gaussian blur applied to star image

Next, the Gaussian blurred image is binarized using a binary threshold. The Otsu algorithm is used to select the optimal threshold value [3], and the binarized image is shown in Figure 3. Each image in the dataset is passed through these steps of preprocessing before proceeding to feature extraction.

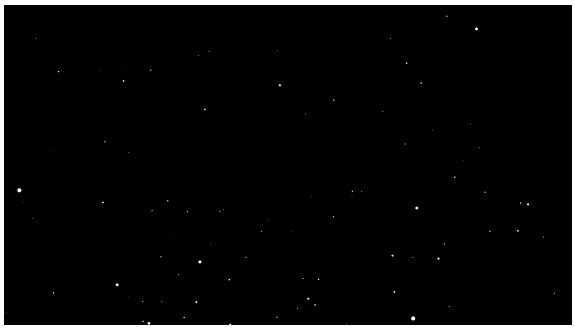


Figure 3. Result of binarizing the Gaussian blurred image

3.3. Feature extraction

After preprocessing, all of the contours are found in the image and the result is shown in red in Figure 4. All found contours are assumed to be stars in the binarized image.

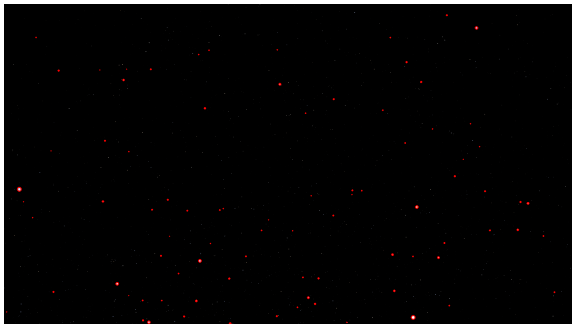


Figure 4. Result of locating global contours

The largest contour is found by comparing the areas of all the detected contours which is an O(n) operation that scales linearly with the number of detected contours. The radius of the largest contour is used as the first element of the feature vector. A circular mask of pixel radius 500 is created with its origin at the pixel location of the largest contour. The resulting mask is shown in Figure 5. The number of non-zero pixels in the masked region are counted and stored as the second element of the feature vector.

Figure 5. Circular mask around largest detected contour

The mask is applied to the binary image in order to isolate the stars that are nearest to the brightest star in the image. The resulting masked image is shown in Figure 6.

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

Next, only the contours in the masked region are used for further feature extraction. Thus, the relevant contours to the feature vector are shown in blue in Figure 7.

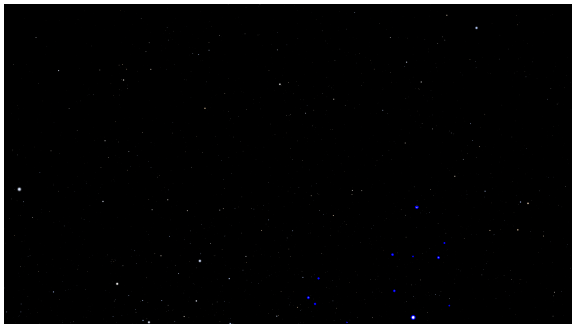


Figure 7. Result of locating contours in the masked region

The 4 next-biggest contours in the region localized around the biggest contour are used to attain the remaining feature vector elements. The contour radius of each of the 4 next-biggest contours in the masked region are used as feature vector elements 3 through 6. Next, the Euclidean distance is calculated between each of these contours and the largest contour which corresponds to feature vector elements 7 through 16. In total, 16 features are used to train the SVM.

To train the SVM, the dataset is split randomly using 20% of the images for the test set. The feature vectors are normalized using the training set as a reference, and the same norm is applied to the feature vectors of the test set. The results are discussed in Section 5 and the confusion matrices for the training and test data are shown in Figure 15. Also, Principal Component Analysis (PCA) was performed on the normalized training data and the 3D visualization of the PCA is shown in Figure 8. From the visualization, it is clear that there are some outliers to the transformed data. Each color represents one of the 4 classes and some groupings can be seen in the data.

The hyperparmeters of the SVM were tuned and the final values used were the radial basis function kernel with $\gamma = 1000$ and $C = 100$.

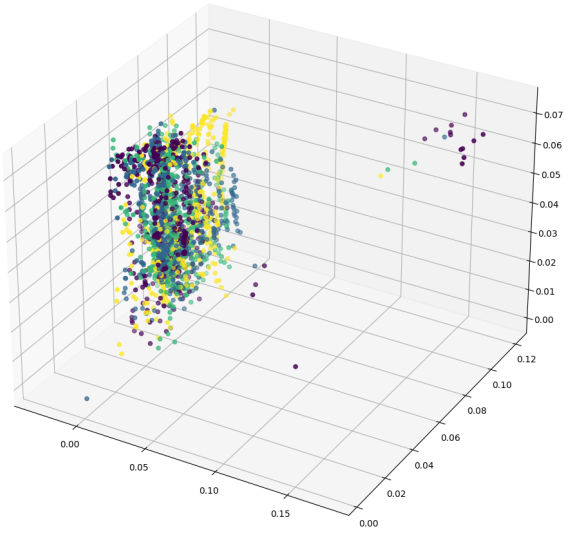


Figure 8. PCA applied to the normalized training data

4. Alternate Method

In this section, an alternative method is proposed which attempts to use a Convolutional Neural Network (CNN) to classify the attitude based on the input image without any manual feature extraction. The CNN is applied to the original dataset, then an augmented dataset is created by adding in rotated images to the dataset to simulate satellite rotation.

4.1. CNN

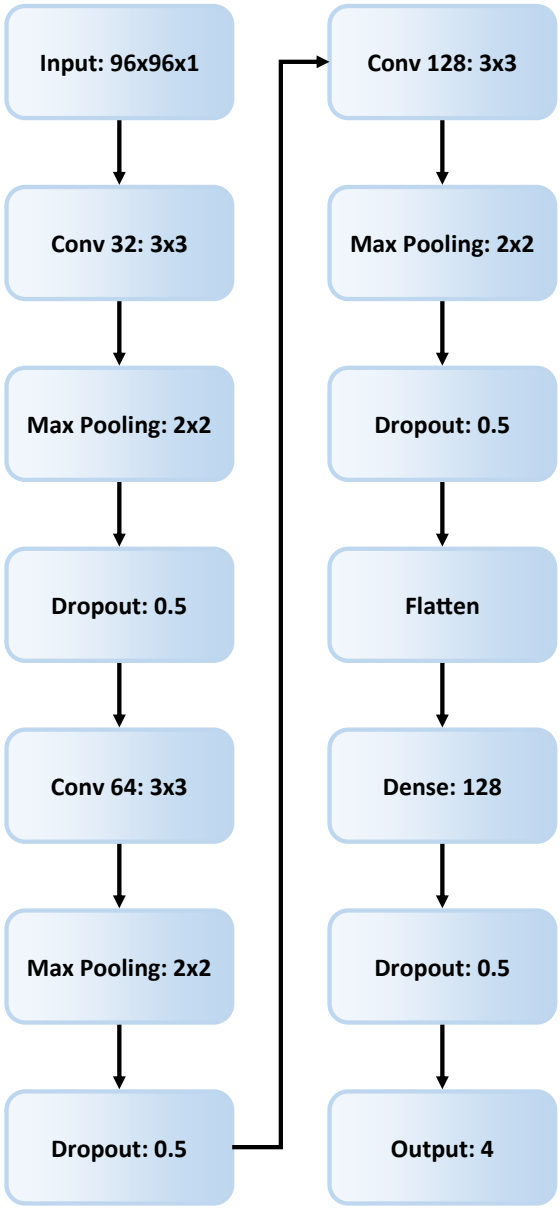


Figure 9. CNN model architecture

The proposed model architecture for the CNN is shown in Figure 9. The model includes several convolution layers, as well as max pooling and dropout. Dropout was implemented in an attempt to reduce model overfitting. The input to the CNN is a 96x96 gray scale image. Thus, the input images had to be scaled down to 96x96 and padded with zeros accordingly to fit the input image size. Also, since the input image matrices are sparse due to the nature of the stars, the images were blurred significantly and binarized in an attempt to make it easier for the CNN to recognize sig-

nificant patterns. The images were blurred using a Gaussian kernel ($\sigma = 40$) similar to the SVM method.

The original dataset was split into 20% for validation and 10% for testing. The validation test set was used to cross validate the model during training. The hyperparmeters of the model were tuned and the final values used were a learning rate of 0.001 with the Adam optimizer and a batch size of 16. The resulting loss and accuracy on the training and validation sets for the CNN are shown in Figures 10 and 11. The confusion matrices for the training, validation, and test sets are shown in Figure 16.

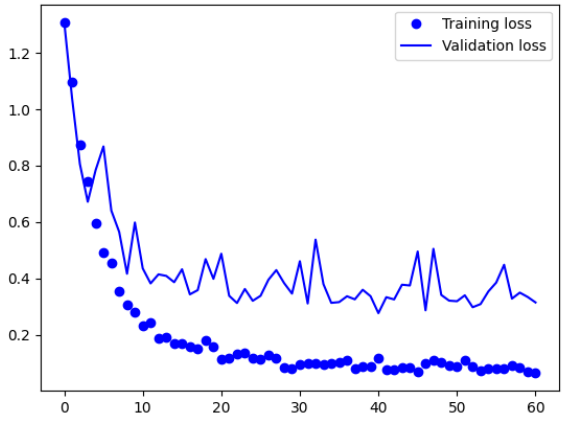


Figure 10. CNN training and validation loss

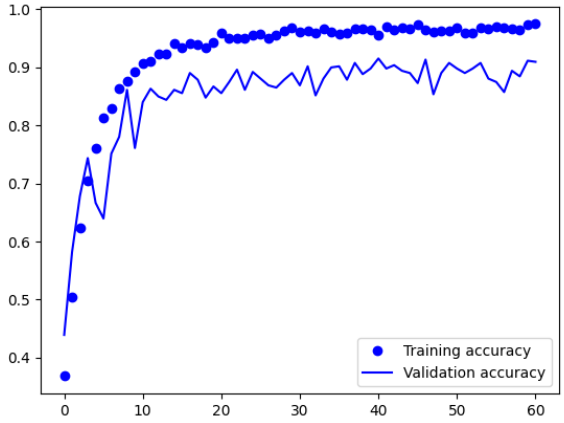
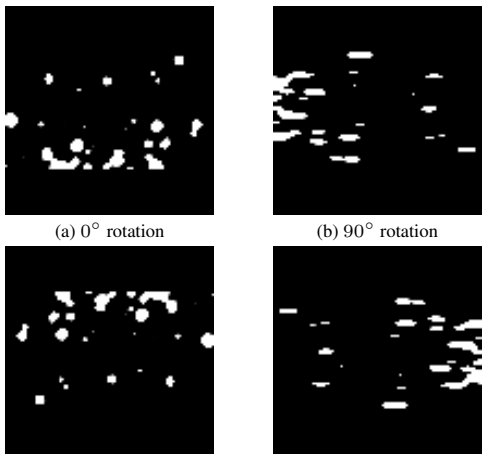


Figure 11. CNN training and validation accuracy

4.2. Dataset Augmentation

Training the CNN on the original dataset may cause the model to recognize patterns that do not solve the exact problem at hand. The nature of the attitude determination problem is such that the identified class should be robust to rotation around the axis (or around the middle pixel in this case). For the SVM, the features were chosen such that they would be the same regardless of the image's rotation. However, to replicate this in the CNN model, the dataset must be augmented with rotated images. Thus, the dataset was augmented by rotating each image by 90, 180, and 270 degrees. The final dataset for the augmented CNN has 10,368 images. An example of the augmented data is shown in Figure 12.



(a) 0° rotation (b) 90° rotation

(c) 180° rotation (d) 270° rotation

Figure 12. Example of augmented dataset for training the Convolutional Neural Network

The same model architecture was used and the hyperparameters were retuned and a learning rate of 0.001 with the Adam optimizer was used again; however, the batch size was increased to 32. The resulting loss and accuracy on the training and validation sets for the augmented CNN are shown in Figures 13 and 14. The confusion matrices for the training, validation, and test sets are shown in Figure 17.

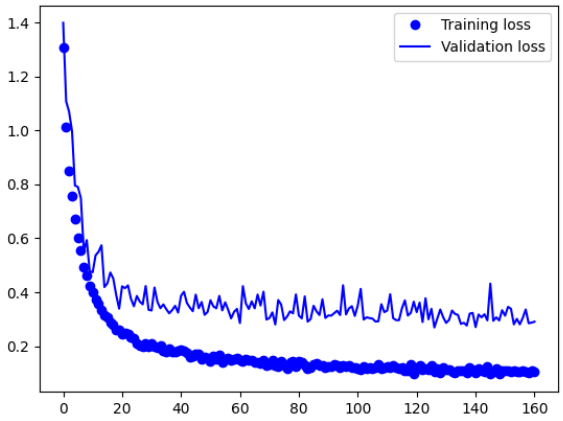


Figure 13. CNN (augmented dataset) training and validation loss

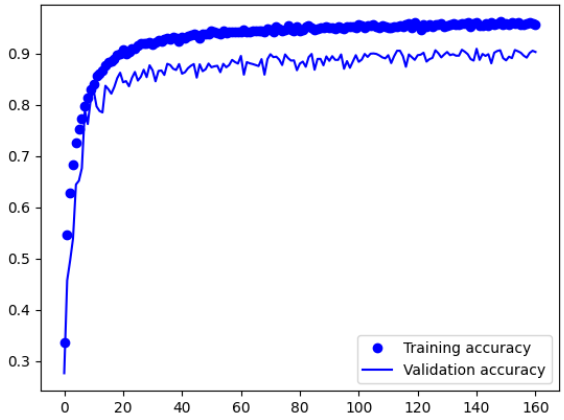


Figure 14. CNN (augmented dataset) training and validation accuracy

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

The accuracy for all proposed methods is summarized in Table 1. From the results, it appears that all 3 supervised models suffer from overfitting as the classification was much more accurate on the training sets than the validation and test sets. The CNN without augmented data performed the best on the test set with 96.2% accuracy followed by the same CNN model trained with the augmented dataset with an accuracy of 91.1%. The SVM performed the worst of the 3 models with an accuracy of 88.1% on the test set.

The primary source of error in the SVM algorithm is due to the fact that the features are not exactly unique to the class around the class boundaries. This is because the algorithm uses the brightest star and distances to the 4 brightest stars around the brightest star. For example, the brightest star could be present in one image labelled as the North-East sky, but also be visible as the brightest star in an image labelled as the North-West sky. This error is specifically noticeable around the South-East/West sky boundary, which can be observed from the high amount of misclassified stars in the training and test data confusion matrices shown in Figure 15.

A similar result is observed for the CNN trained with the augmented data set in that there is a large number of misclassified stars between the South-East/West skies. From this observation, it seems that the CNN trained on the augmented dataset has created a model which is similar to the SVM trained with the features described in Section 3. Intuitively, this observation makes sense because the dataset was augmented by rotating the images around the center pixel, which has the impact of making the model more robust with respect to axial rotation. Similarly, the features chosen to train the SVM were chosen such that they would be the same values regardless of rotation around the axis. For example, the distance of the brightest star to other stars does not change as the field of view is rotated around the center pixel.

Although the CNN trained on the original dataset shows the highest accuracy, it is likely that in practice this model would not perform the best because it will not be robust to axial rotation. This robustness is crucial to the satellite as it is unlikely that the satellite will be rotated in such a way that it always aligns with the axial rotation of the images in the supervisory dataset. For this reason, the CNN trained with the augmented dataset of rotated images is the better candidate for usage in practice.

6. Conclusion

In this paper, a method was proposed to perform satellite attitude determination without an on-board star database. Two approaches were evaluated, both of which relied on usage of supervised learning against a dataset of simulated star field images. The first approach utilized manual feature extraction and classification with an SVM, and the second used a CNN. The results showed that the CNN model performed with the highest accuracy on the test set; however, it was determined that the CNN trained on the augmented dataset would be more useful in practice due to the robustness to axial rotation.

The results indicate that the proposed methods are not sufficient to perform highly accurate attitude determination due to the fact that the accuracies on the test set are relatively poor considering that the sky was partitioned into massive ranges, resulting in only 4 classes. Further tests were performed by increasing the number of classes and hence increasing the precision of the algorithm; however, test set accuracy was reduced significantly with small increases to the number of classes.

In practice, it is useful for satellites to have arcsecond precision from the attitude estimation, which can be achieved with the usage of star lookup tables. By using the star lookup tables, the role of the image processing algorithm can be solely of identifying star patterns which can be used to lookup a star in the on-board database. The exact right ascension and declination can be taken from the database based on the identified star and from these angles, and knowing the pixel location of the star in the image, a much more accurate attitude can be determined from the

image.

Further research on this topic could be undertaken to determine if there are more suitable features that can be extracted from the star field images to perform attitude regression without a star database. Perhaps some geometric transformations could be applied to the images, or graph theory could be utilized for more descriptive feature extraction. Graphs could be created based on the interrelationships between identified star contours, and further analysis could be undertaken.

References

[1] Stellarium 0.22.0. https://stellarium.org/, 2022. 2

[2] E. Bolandi, M. Haghparast, F.F. Saberi, B.G. Vaghei, and S.M. Smailzadeh. Spacecraft attitude determination and control. *Measurement and Control*, 45(5):151–157, 2012. 1

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 2, 3

[4] T. Delabie, J.D. Schutter, and B. Vandenbussche. Highly efficient attitude-estimation algorithm for star trackers using optimal image matching. *Journal of Guidance, Control, and Dynamics*, 36(6), 2013. 1, 2

[5] J. Hong and J.A. Dickerson. Neural-network-based autonomous star identification algorithm. *Journal of Guidance, Control, and Dynamics*, 23(4), 2000. 1

[6] K.T. Kim and H. Bang. Reliable star pattern identification technique by using neural networks. *The Journal of the Astronautical Sciences*, 52:239–249, 2004. 1, 2

[7] D. Rijlaarsdam, H. Yous, J. Byrne, D. Oddenino, G. Furano, and D. Moloney. A survey of lost-in-space star identification algorithms since 2009, 2020. 1

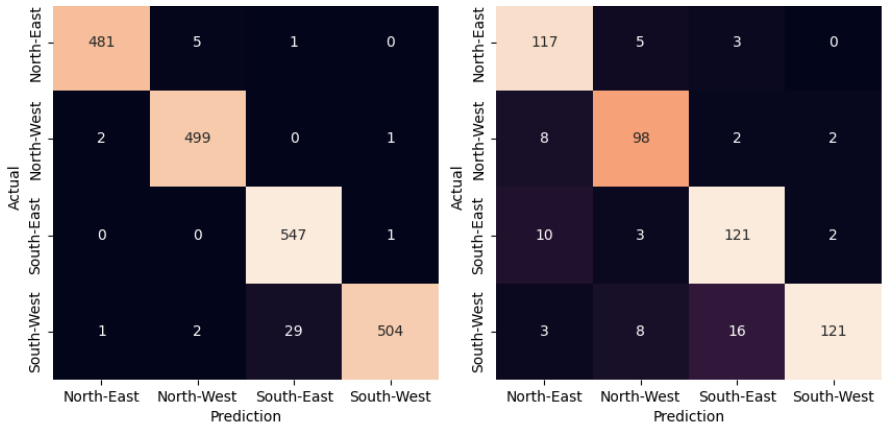
[8] Agarwal S., E. Hervas-Martin, J. Byrne, A. Dunne, J.L. Espinosa-Aranda, and D. Rijlaarsdam. An evaluation of low-cost vision processors for efficient star identification, 2020. 1

[9] S. Scott. Star tracker. https://github.com/stephen99scott/star-tracker, 2022. 2

[10] S. Scott. Stellarium star fields with labelled coordinates. https://www.kaggle.com/datasets/stephenscott99/star-fields, 2022. 2

[11] C.G. Smith. Development and implementation of star tracker based attitude determination, 2017. 1

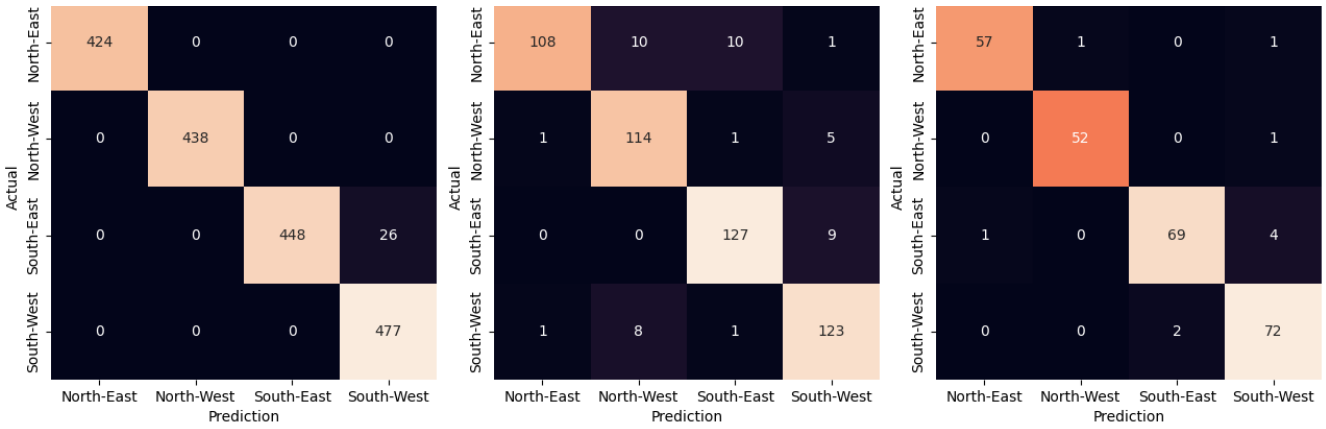
[12] H. Yoon, Y. Lim, and H. Bang. New star-pattern identification using a correlation approach for spacecraft attitude determination. *Journal of Spacecraft and Rockets*, 48(1), 2011. 1



(a) Training data confusion matrix

(b) Test data confusion matrix

Figure 15. Support Vector Machine classification results

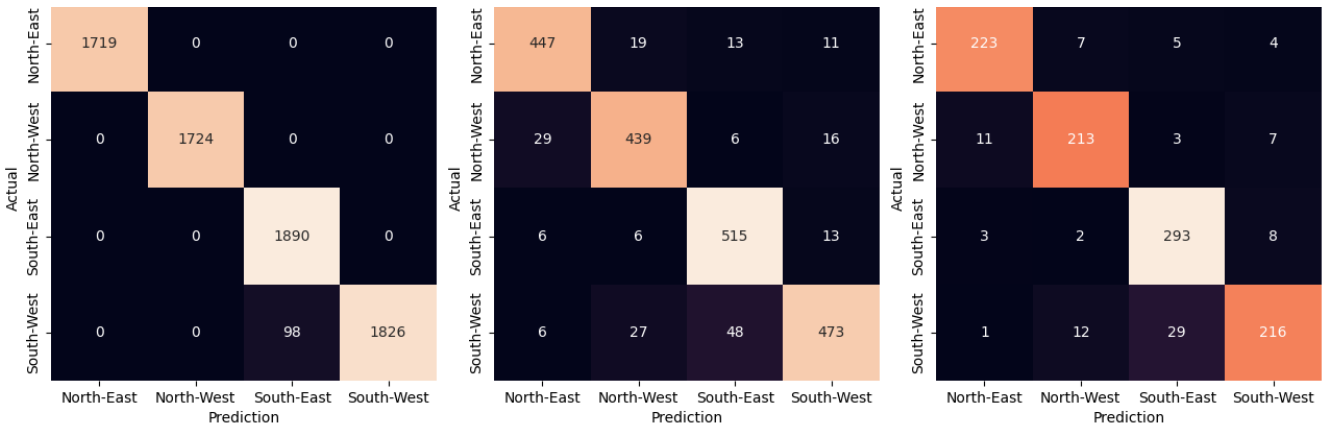


(a) Training data confusion matrix

(b) Validation data confusion matrix

(c) Test data confusion matrix

Figure 16. CNN classification results



(a) Training data confusion matrix

(b) Validation data confusion matrix

(c) Test data confusion matrix

Figure 17. CNN (augmented dataset) classification results