Noise Robustness of Data-Driven Star Classification

Floyd Hepburn-Dickins[a] and Michael Edwards[b]
Department of Computer Science, Swansea University, U.K.

Keywords: Neural Networks, Star-Pattern Recognition, Data Generation.

Abstract: Celestial navigation has fallen into the background in light of newer technologies such as global positioning systems, but research into its core component, star pattern recognition, has remained an active area of study. We examine these methods and the viability of a data-driven approach to detecting and recognising stars within images taken from the Earth's surface. We show that synthetic datasets, necessary due to a lack of labelled real image datasets, are able to appropriately simulate the night sky from a terrestrial perspective and that such an implementation can successfully perform star patter recognition in this domain. In this work we apply three kinds of noise in a parametric fashion; positional noise, false star noise, and dropped star noise. Results show that a pattern mining approach can accurately identify stars from night sky images and our results show the impact of the above noise types on classifier performance.

1 INTRODUCTION

Star pattern recognition dates back to the ancient era, used by a multitude of empires as part of navigation for exploration and trade. Polynesia, one of the worlds largest nations at its time, was formed in great part due to celestial navigation. This is achieved through the use of guiding stars and constellations, man-made groupings of stars easily picked out due to them containing those brightest and most obvious stars in the night sky. The advent of modern computing has supplanted these methods through a host of alternative navigation systems, most predominately global positioning systems. There is still a call for celestial navigation, most often with regard to space borne vessels and attitude determination (Rijlaarsdam et al., 2020; van Bezooijen, 1989). In recent years, when considering a desire for greater redundancy in navigational systems and the potential threats posed towards global position systems from the vulnerability of the satellites themselves to the possibility of interference from third parties, celestial navigation presents itself as a viable alternative.

At its core exist two primary problems, star detection, that is to pinpoint prospective stars within an image, and star recognition, the correct identification of a given star. Whilst the former is a widely solved issue (Stetson, 1987), the latter remains a point of active research (Rijlaarsdam et al., 2020). The computer age has brought with it a host of improvements on this ancient art, moving from identifying defined constellations and the stars within, to deriving more general solutions applicable across the night sky.

Whilst the focus of modern research is more commonly applied towards spacecraft and attitude determination, the techniques and methods used can broadly still be applied to navigation from the Earth's surface as well. The first and foremost challenge is essentially the same; any navigation requires the accurate and timely pinpointing and recognition of the stars at hand. In the past this would be achieved slowly by consulting a nautical almanac and sextant. Modern pattern recognition techniques can vastly improve both the acquisition and identification and have been shown to do so when applied to attitude determination, we further examine their efficacy on systems grounded to the earths surface alongside the challenges that entails.

2 RELATED WORK

Despite fading into the background of navigation, star identification and recognition has nonetheless seen a steady evolution of research over the decades. In (Liebe, 1993) they proposed a triangle method, forming patterns based on the angular distances between a triplet of stars, the spherical angles between them and their magnitudes. By generating a database of these geometric properties and the patterns they form they are able to match prospective identifications against them and thus identify a given star. The method is

[a] https://orcid.org/0000-0001-5943-0652
[b] https://orcid.org/0000-0003-3367-969X

Hepburn-Dickins, F. and Edwards, M.
Noise Robustness of Data-Driven Star Classification.
DOI: 10.5220/0011804000003411
In *Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2023)*, pages 176-184
ISBN: 978-989-758-626-2; ISSN: 2184-4313

both rapid and accurate, but suffered limitations with the introduction of false stars, a problem ever more apparent due to the crowding of the celestial sphere with satellites and other man-made objects that may present themselves in an image. This was improved upon in (Nabi et al., 2019) via their design of the star-triplet database construction and feature extraction methods resulting in a greater tolerance to missing stars and improved ability in handling reduced numbers of stars present within the field of view (FOV).

There are a number of papers based around these handcrafted properties between a guiding reference star (GRS) and its guiding neighbour stars (GNS). The work of (Mortari et al., 2004) utilised a pyramid schema of four stars able to resist larger numbers of false stars. The speed can then be increased via the use of hash mapping with the pyramid method when searching the space of available patterns (Wang et al., 2018) .

The grid based algorithm (Padgett and Kreutz-Delgado, 1997) utilises a primary star and its neighbours within a specified radius to form a pattern vector, providing greater tolerance towards positional noise. The modified grid algorithm (Na et al., 2009) further increased position and magnitude tolerance. Additional work here includes the use of radial and cyclical feature patterns (Zhang et al., 2008) allowing for rotation-invariant matching. With (Jiang et al., 2015) improving upon the radial patterns and their positional tolerance via utilising a redundant-coded solution.

Machine learning techniques in the work of (Xu et al., 2019) have utilised these pattern generation techniques through the lens of neural networks. This allows for the use of machine learned patterns in lieu of the handcrafted ones in prior literature, eliminating the need to search through any datasets for pattern matching whilst (Yang et al., 2022) utilises a 1D convolutional neural network to achieve increases in both speed and robustness towards various noise types.

Throughout this paper we examine the auto-encoder and classifier outlined in (Xu et al., 2019) and its performance when operating on simulated images from the earths surface whilst parametrically exploring the models tolerance to noise.

3 PROPOSED METHODOLOGY

3.1 Detection from Image

In order to recognise the stars they must first be detected within an image. This involves accurately pinpointing the centre of a source, minimising the detection of false sources and overcoming the varying noise that may be present. To do this, we use an implementation of the DAOFIND algorithm (Stetson, 1987) that searches an image for local density maxima with a peak amplitude exceeding a chosen threshold and whose size and shape are similar to a defined 2D Gaussian kernel. Through this we can apply various thresholds and parameters relating to the the size and intensity of prospective sources in order to filter relative to our domain and conditions. By subtracting the median of our image prior to entering it into the algorithm, we can reduce the impact of background noise and light pollution, an issue that is much more prevalent in our domain of ground-borne wide angle sensors.

This allows for the accurate pinpointing of prospective sources centroids, which can then be used as the basis for the data our model will see. Whilst some stars will inevitably be lost, some centroids may be off by a small amount, and false stars will still filter through and become included in our detected sources, these are precisely the types of noise we hope to build a tolerance to within our implementation.

3.2 Constellation Representation

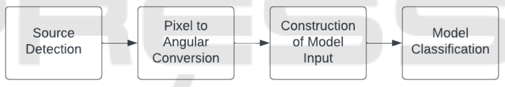

Figure 1: Pipeline.

Moving from detected sources to a classification first requires a learned representation of each star within our catalogue. To do this we need to extract information about the local features for each star. The exact definition of what is local for a star depends on our chosen parameters. We opt to utilise those stars closest to a prospective GRS as they are more likely to be captured together in an image.

When considering a given star and its neighbours, the primary points of information we can utilise are the brightness of the stars, and more critically, the local topology of points, specifically the angular distances between them. By utilising the angular distances between the stars in a radial manner we are able to make our learned representation invariant to rotation and translation within the image. The angular distance will remain constant, across images and time, forgiving the minor shifts in the stars themselves, thankfully this takes place over such a span of time as to be negligible.

The night sky is rarely perfectly clear, showing us just the stars as we want them. There is a great deal of noise, from occlusion and light pollution to all manner of satellites and astronomical phenomenon that may be present across our view. Beyond that, the imaging system itself can introduce errors or noise. To increase the robustness of our classifier it needs to overcome this noise, and as such we train our classifier on a variety of simulated noisy conditions categorised as positional noise, false star noise and dropped star noise.

Positional noise occurs when the reported centroid of the star is inaccurate, typically to the degree of a pixel or so, as a result of centroid detection in unclear images. False star noise occurs when we detect and attempt to both classify, and use in the classification of other sources, false sources. Such sources aren't stars within our image and are typically borne about through stellar artefacts or phenomenon such as passing satellites, aircraft or meteorites. Dropped star noise is where sources that should be present in the image and used for classification are omitted, typically due to occlusion of part of the image from some terrestrial artefact or weather conditions.

By introducing this noise into our training images, we can imbue a tolerance towards it in our eventual classifier. Such images can reduced into lists, D_{sn} and D_{nn}, detailing the distances between a GRS and it's nearby GNSs and the pair-wise distances between GNSs respectively. The discretized outcome of these lists, concatenated together to preserve identity between them, forms the input for our model, that which is used to discern a star pattern or representation.

The parameters for the radius, from which we consider GNSs, and discretization can be varied but in keeping with the results from (Xu et al., 2019) and experimental testing we use an input dimension of 400, derived form a radius p and discretization factor e where $p/e = 200$. The model is trained by creating a clean tensor and several noisy counterparts.

Clean tensors are formed utilising the Yale Bright star catalogue (BSC), or a magnitude threshold-ed version thereof, by choosing a star and retrieving all neighbouring stars within p. Each star is represented as a declination and right ascension and from these values we can calculate D_{sn} and D_{nn}, and thus the input tensor.

3.3 Constellation Recognition

Recognition refers to the final step, the determination of precisely which star our input tensor refers to. The star pattern generator maps from a potentially noisy input to an approximation of the clean tensor for that star. We then train a classifier to take these tensors and learn the association between them and its classification, in our case its HR designation. For a given vector we obtain a probability it belongs to each of the stars in out catalogue, we then take the highest probability result as our classification.

4 EXPERIMENTAL SETUP

4.1 Datasets

4.1.1 Training Data

In order to train the autoencoder we require a clean constellation representation and multiple noisy variants. Clean representations are obtained using the BSC, a catalogue of stars visible to the naked eye, ranging up to a visible magnitude of 6.5. From this we extract the HR number, DECJ200, RAJ2000, and visible magnitude to form a simplified revised catalogue.

After selecting a candidate star from the BSC catalog we identify those stars that fall within our chosen radius p and generate the clean tensor input via discretizing D_{sn} and D_{nn}. In order to create noisy counterparts we utilise a data augmenter during the training of the star pattern auto-encoder. This augmenter randomly applies a noise type and intensity to the prospective star and its connected stars within p before re-calculating and re-sorting the angular distances in order to generate D_{sn} and D_{nn} for this new noisy variant. The structure of the data augmenter is shown in figure 2.

Dropped star noise is created by randomly choosing between 1 and a pre-determined maximum number from the neighbouring stars and removing them from the set.

False star noise is created by applying a random angular offset between 0° and 360° from the central star, as well as a random separation offset between 0° and the radius p to create a new star. The new stars RA and DEC is then calculated as an offset from the focal star.

Position noise chooses a random angular offset from 0°-360° and a separation offset from $0° - x * radius$ where x is our chosen modifier for the maximum potential offset. Each star is then shifted according to these values.

Once noise has been applied, angular distances between stars are re-calculated and the list of nearest neighbours re-sorted before discretiziation. Ultimately this process allows for the stochastic generation of noisy counterparts for each stars clean tensor.

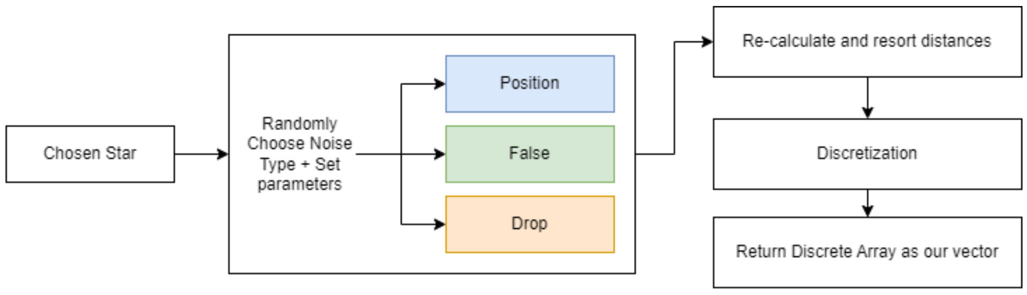
Chosen Star
Randomly Choose Noise Type + Set parameters
Position
False
Drop
Re-calculate and resort distances
Discretization
Return Discrete Array as our vector

Figure 2: Data augmenter.

4.1.2 Testing Data

In order to validate our approach we utilise the open source software Stellarium to obtain synthetic images of the night sky. Through this we can emulate a particular viewpoint and simulate the stars up to a desired visible magnitude. We can furthermore include or remove any celestial artifacts, ambient light pollution and satellites within the environment to garner clearer or more realistic images.

For each synthetic image we extract the pixel coordinates of each star alongside the catalogue information from the BSC. This results in each image having a corresponding CSV file detailing the metadata for the viewpoint, j-date, camera origin and details for each star.

For our testing we generated 11900 such images, randomly sampling longitudinal and latitudinal points across the Earth's surface.

Figure 3: A synthetic image generated within Stellarium.

4.2 Implementation

Source detection is achieved utilising DAOStarFind from the astropy library. Prior to performing the operation on the image, we subtract the median value from it in order to reduce background noise. This provides us with a list of sources typically exceeding the actual number of stars present in the image, due to the detection of satellites and other false stars.

In order to generate our model input we need to be able to convert from pixel to angular distances. In lieu of calibrating a camera we train a support vector machine (SVM) to map from the pixel co-ordinates of a star to its angular distance to the centre of the image. We are able to train the SVM from the CSV files generated alongside our synthetic images.

By imagining the centre of the image as being right ascension 0 and declination 0 we can determine a supposed RA/DEC of a particular star by calculating it is an offset from this central point, calculating the distance using our SVM. Whilst the RA/DEC is unrelated to its true RA/DEC, we can utilise these manufactured values alongside those of another star in the same frame to calculate the angular distance between them, which would be the same for their true right ascension and declination, thus allowing us to generate our D_{sn} and D_{nn} lists.

The final model used to classify sources is built from a star pattern generator and star pattern classifier the structure of which is shown in figure 4. The generator is the pre-trained auto-encoder created by training on simulated images of a GRS and its nearby GNSs. Each GRS has a clean pattern, borne from the discretization of its angular distance to its GNSs and the intra-GNS distances. Each clean pattern is augmented using the data augmenter detailed above to create noisy counterparts of varying intensity for each of the noise types. In training an auto-encoder to denoise these noisy versions we are left with an encoder capable of creating latent star-patterns for use in our classifier.

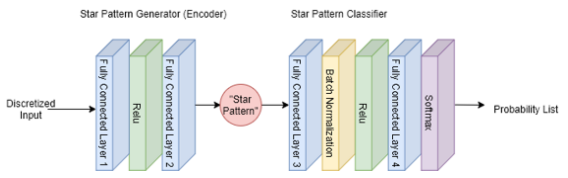

Figure 4: Combined structure of the star pattern generator and star pattern classifier.

Hyper-parameters were chosen from a mix of those derived in (Xu et al., 2019), batch testing, and practical considerations. When considered in the context of navigation on the earths surface, achieved without expensive imaging systems, we limit magnitude down to 3MV on a fish-eye lens. This allows for the use of inexpensive cameras, using lower exposure times whilst trying to capture as much information as possible. The lower magnitude threshold is borne from this need for minimal exposure times, to reduce noise imparted through motion . We still adhere to the input size arrived at within (Xu et al., 2019) of 200 derived from *2[p/e]* as well as utilising 10 GNSs per GRS, but due to fewer stars being present within each image we need to expand the radius under which stars are collected to achieve this. Testing found that increasing this to 60° allowed for the capture of the 10 GNSs necessitating an increase of the e factor to 0.3. The auto-encoder is trained first with a batch size of 32 before fine tuning with a batch size of 512.

In order to maximise position noise tolerance with these new imaging parameters we exhaustively tested training the full model with differing levels of position noise applied. From these we can choose the model that is best able to handle the most position noise. Training was done via randomly applying position noise up to a maximum value. This maximum value was tested up to 6° at which point the accuracy suffered greatly. In evaluating position tolerance, the generator was adjusted to generate results only at the provided value opposed to a range up to that value, and to only apply position noise rather than the other types. We then evaluated each model at each position value allowing us to see the trade-off between tolerance to position noise and accuracy. The first results are visible in figure 5 and figure 6. The most appropriate model from these would be one trained on position noise up to 2.5° and effective up to 1° with an accuracy of $\sim$ 96.2%. However by further examining potential solutions within the optimal zone we aim to improve on this preliminary model.

The early results show the best models exist where they are trained to 2.5 to 3.5 but evaluated up to 1 to 1.5. Subsequently we search with greater granularity within this space, erring to either side, for the optimal model. This subsequent search parameters are detailed in Table 1.

Table 1: Model Training Parameters.

	Epochs	Batch Size	Early stopping
Autoencoder	4000 / 2000	32 / 512	No
Classifier	2000	512	Yes

Figure 7 shows the results of the more pinpointed training under this regime.

Table 2 showcases the results from the further testing to identify the optimal model, trading off tolerance towards pixel noise against accuracy, focused on

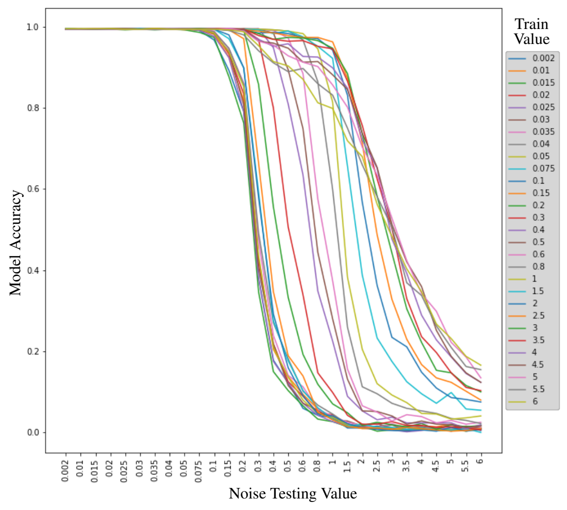

Figure 5: Position noise and its impact on accuracy.

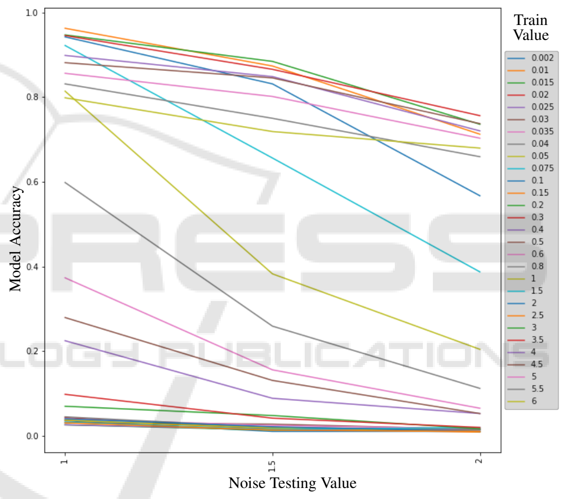

Figure 6: Position noise and its impact on accuracy close up.

due to it being the primary source of noise present within our simulated images, particularly when considering the innate error generated when converting from pixel to angular distances from the SVM. These results were generated by evaluating the model at explicitly the stated test value for pixel noise, opposed to a random distribution up to that value, as they were trained.

The table shows the best model able to handle the given test value of pixel noise. The desire is to choose a model tolerant to as much position noise as possible, whilst still maintaining a high degree of accuracy. To this end, the model trained at 3.342° but viable up to 1.393° is selected to move forward in our testing. When evaluated across all noise types this model scores an accuracy of 98.53% on 320000 sam-

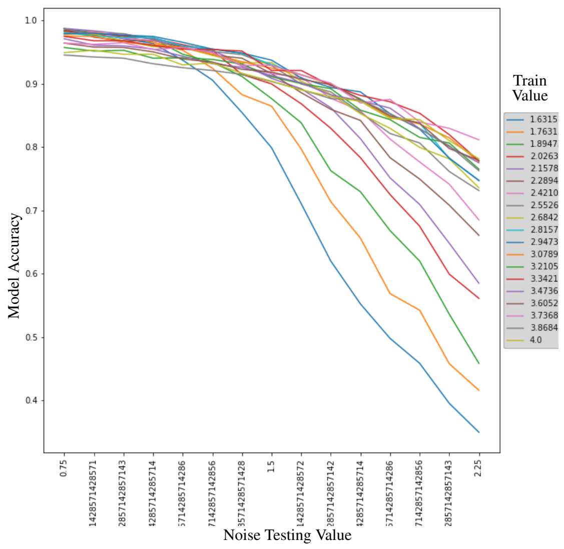

Figure 7: Position noise and its impact on accuracy, further testing.

Table 2: Showcasing the optimal model training value to maximise accuracy on a given test value when considering positional noise.

Optimal Training Max	Test Value	Accuracy
2.157	0.75	0.987
2.157	0.857	0.983
2.553	0.964	0.979
2.947	1.071	0.974
2.947	1.179	0.965
2.816	1.286	0.954
3.342	1.393	0.951
3.737	1.5	0.937
2.947	1.607	0.921
3.737	1.714	0.9
3.342	1.821	0.887
3.737	1.929	0.875
3.342	2.036	0.853
3.737	2.143	0.829
3.737	2.25	0.811

ples generated from the augmenter.

In summation, our final model chosen to be tested on real images has been trained up to noise tolerance of 3.342°, though practically it is considered accurate up to a value of 1.393°. The auto-encoder has been trained across 6000 epochs, 4000 with a batch size of 32 and 2000 with a batch size of 512 for finer adjustments. The classifier is trained for 2000 epochs at a batch size of 512 with early stopping. Early testing showed this to be the best batch size for the classifier and is in-keeping with the findings of (Xu et al., 2019) although the datasets are quite different at this point due to the differing thresholds on magnitude.

5 RESULTS

Before testing the model we can examine the results of the source detection algorithm. Across 11900 images simulated in Stellarium with a magnitude threshold of three, 1104659 sources were detected vs the true value of 1036254 stars being present within those images. The disparity stems from two primary sources, the first, and most common, is the presence of stellar phenomenon across many of the images, most common was various meteor showers that were detected in images located across the breadth of the Earths surface, these were subsequently picked up as one or more sources in each of the images in which they exist and account for the vast majority of the discrepancy between actual stars and sources. Additional detections stem from certain stars that were detected as multiple sources, though rarely, as well as a minor discrepancy between those stars simulated within Stellarium vs those that were calculated as being present in the image based off the Yale bright star catalogue. Namely where Stellarium calculated a stars apparent magnitude to be different to the catalogue, and where this value pushed a star below our magnitude threshold, it resulted in stars being shown that shouldn't be.

The SVM was trained on the angular distance from each star in an image to the center of that image. Within the 11900 synthetic images we obtain 1036254 distances. With a train test split of 0.2 it scored an accuracy of ~ 0.9988 with an average error of 0.05°.

Figure 8 showcases the auto-encoders ability to move from a noisy sample to the clean version thereof. Displaying the discretized angular distances of D_{sn} and D_{nn} concatenated together. The second chart, autoencoder output, serves as the input to the classifier and is used to determine the final prediction. In this instance a random choice of noise and magnitude of that noise has been applied, resulting in minor, but noticeable differences between the noisy sample (top) and clean version (bottom). The auto encoder is able to accurately move from this noisy sample to a very close approximation of the clean sample, highlighted in the 3rd chart by rounding the values.

Figure 9 shows a more extreme case of noise. Using our chosen auto-encoder from the final model we took forward, which is stated as effective up to an angular distance noise of 1.343°, this figure shows the results for a sample with maximum angular noise applied. Here the deformation between the noisy and clean sample is markedly more pronounced, and whilst the auto-encoder output bares less resemblance compared to our previous example, it still returns a

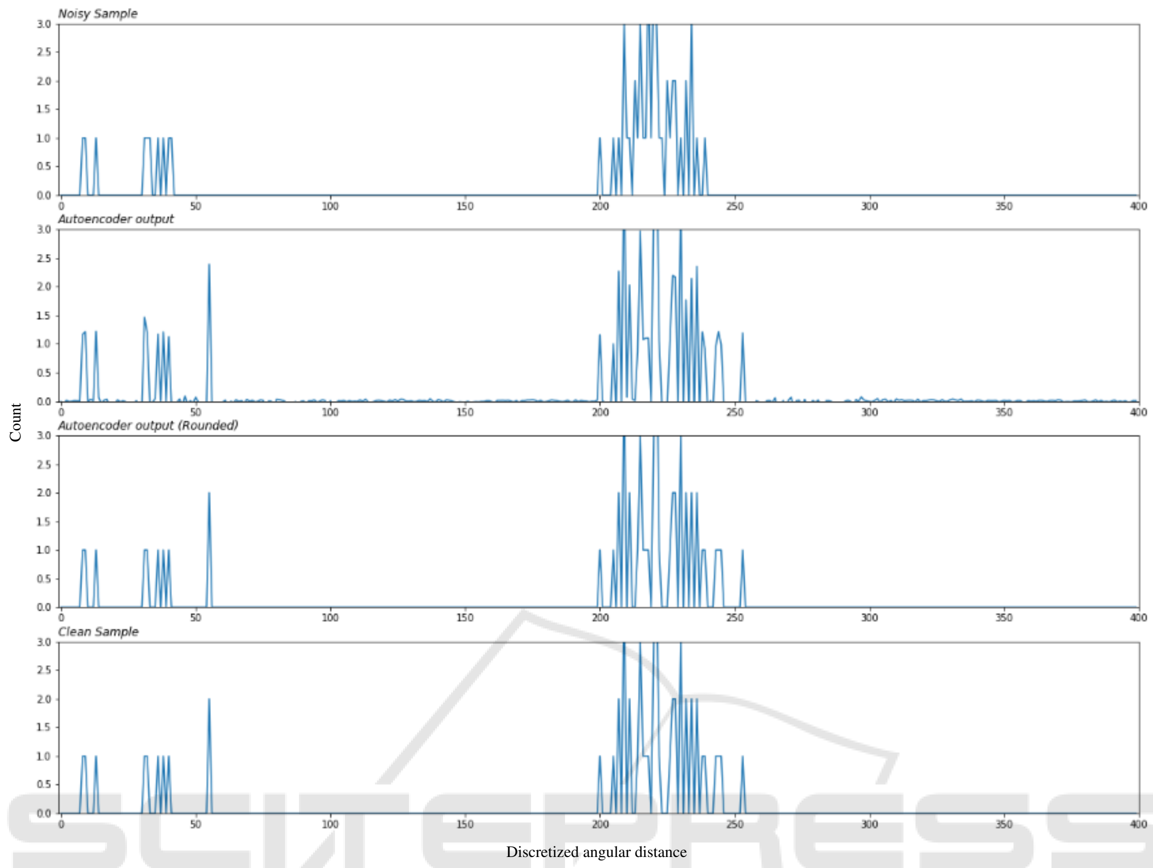

Figure 8: Autoencoder results.

good approximation of the clean sample.

The next stage of testing is carried out using the combined model on simulated images. Such images contain within them a wide variation on both the noise types and magnitude thereof. Of the 1104659 sources classified, 1036254 of which we know to be true stars present within the image, the model successfully classifies 859287 of these stars with an average confidence of ~ 0.937. In contrast to (Xu et al., 2019) we apply no final stage of weight searching, each star is identified solely based on its GNSs within the image. Within the broad-strokes of results there are some key points of note. Firstly, 68405 sources are false stars themselves as determined by cross checking known locations against detected source co-ordinates. Of the 149423 incorrectly identified stars, 90919 occur in examples of extreme pixel, and thus angular, noise. We know this by cross-checking the source location against known stars in the image, to a certain pixel tolerance here set as 8 pixels. In total $\sim 13.53\%$ of stars are miss-identified, with $\sim 8.23\%$ of these occurring in instances of extreme pixel noise in the GRSs location.

The impact of false stars and dropped stars is less easily quantifiable, but by examining images we can see that there is a much greater tendency towards miss-identifications on the periphery of the images, as would be expected in instances where a large portion of the stars GNSs are cut from view, these represent instances of significant dropped star noise.

When considering these results through the lens of navigation, wherein it is less necessary to identify all stars in the image, and more important to confidently identify a select few, we can also consider what role the confidence of our model has with respect to accuracy. The average confidence of successful identification is ~ 0.94 in contrast to the confidence of ~ 0.42 in miss-identifications. By delving into these confidence scores we can examine what impact applying confidence thresholds might have on identification throughout our image, the end goal being to reduce miss-identifications, but also to eliminate as many of the more damaging high-confidence miss-identifications as possible.

Table 3 shows how the number of correct and incorrect identifications changes when we discard those that do not meet a given threshold. We can see an immediate reduction when limiting our identifications

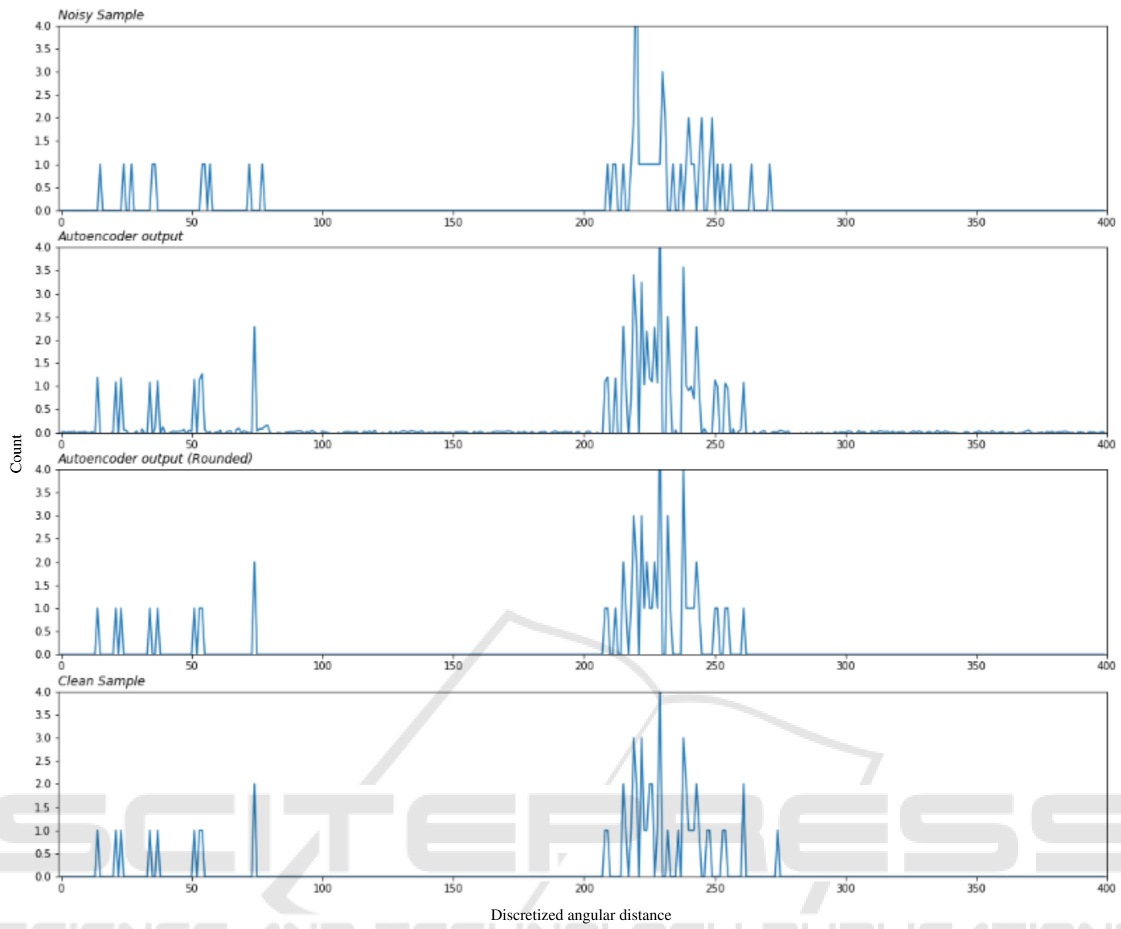

Figure 9: Auto-encoder results for max angular noise.

to a confidence score of 0.8, we are able to discard 86.70% of incorrect identifications whilst only losing 10.33% of correct ones. At the extreme, by limiting confidence to 0.999 or above we lose 39.10% of correct identifications whilst removing the vast majority, 99.85%, of incorrect identifications. This extreme, presuming clear nights, would allow for correct identification of more than enough stars for navigation, whilst drastically reducing the probability of an incorrect identification. With additional checking measures it would be reasonably easy to eliminate the remaining incorrect identifications by cross checking identifications with expected neighbouring stars in an image.

6 CONCLUSION

In this paper we demonstrate the viability of neural network star-pattern recognition when applied to a synthetic dataset emulating the viewpoint at the Earths surface. The synthetic data represents a new

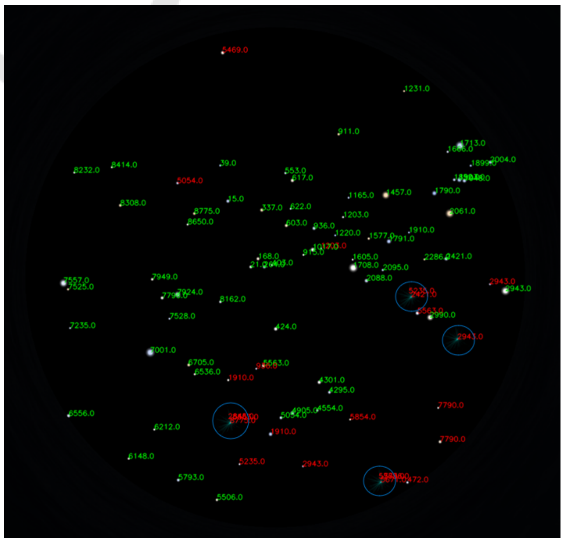

Figure 10: Application of model to synthetic image, correct identifications in green and incorrect in red. Highlighted are 4 regions of false star noise due to meteor showers.

Table 3: Classification results and thresholding impact.

Threshold	Correct Identifications	% Change (Total)	Incorrect Identifications	% Change (Total)
Baseline	859287	N/A	149423	N/A
0.8	770485	10.33%	19874	86.70%
0.9	741632	13.69%	12039	91.94%
0.95	713120	17.01%	7184	95.19%
0.99	640618	25.45%	2119	98.51%
0.999	523322	39.10%	221	99.85%

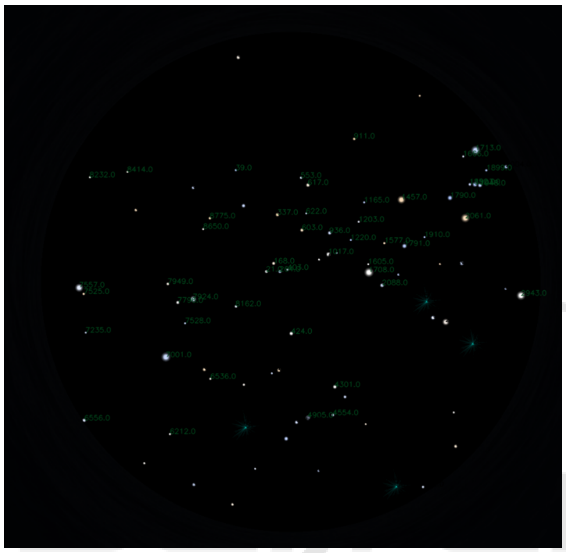
Figure 11: Synthetic image results with 0.999 confidence thresholding, showing the removal of all miss-identifications in the image.

method by which we can garner realistic night-sky images from a variety of viewpoints as well as the corresponding information required to utilise them as a labelled dataset of stars.

We create a data augmenter to avoid pre-computing training patterns whilst allowing for greater freedoms with training and testing and parametrically explore tolerance towards noise. The final implementation performs well on simulated star patterns and is still able to correctly identify the majority of stars in the markedly noisier synthetic images, we threshold our images to a visible magnitude of 3.0 in order to appropriately simulate the capabilities of inexpensive cameras with low exposure times, an important consideration when examining this technology through the lens of celestial navigation from the Earths surface. Whilst miss-identifications do occur, we are able to drastically reduce these by utilising confidence thresholds, a necessary step if the method were to be used for navigational purposes where miss-identifications can be damaging.

REFERENCES

Jiang, J., Ji, F., Yan, J., Sun, L., and Wei, X. (2015). Redundant-coded radial and neighbor star pattern identification algorithm. *IEEE Transactions on Aerospace and Electronic Systems*, 51(4):2811–2822.

Liebe, C. (1993). Pattern recognition of star constellations for spacecraft applications. *IEEE Aerospace and Electronic Systems Magazine*, 8(1):31–39.

Mortari, D., Samaan, M. A., Bruccoleri, C., and Junkins, J. L. (2004). The pyramid star identification technique. *NAVIGATION*, 51(3):171–183.

Na, M., Zheng, D., and Jia, P. (2009). Modified grid algorithm for noisy all-sky autonomous star identification. *IEEE Transactions on Aerospace and Electronic Systems*, 45(2):516–522.

Nabi, A., Foitih, Z., and Mohammed El Amine, C. (2019). Improved triangular-based star pattern recognition algorithm for low-cost star trackers. *Journal of King Saud University*.

Padgett, C. and Kreutz-Delgado, K. (1997). A grid algorithm for autonomous star identification. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):202–213.

Rijlaarsdam, D., Yous, H., Byrne, J., Oddenino, D., Furano, G., and Moloney, D. (2020). A survey of lost-in-space star identification algorithms since 2009. *Sensors*, 20(9).

Stetson, P. B. (1987). DAOPHOT: A Computer Program for Crowded-Field Stellar Photometry. *Publications of the Astronomical Society of the Pacific*, 99:191.

van Bezooijen, R. (1989). A star pattern recognition algorithm for autonomous attitude determination. *IFAC Proceedings Volumes*, 22(7):51–58. IFAC Symposium on Automatic Control in Aerospace, Tsukuba, Japan, 17-21 July 1989.

Wang, G., Li, J., and Wei, X. (2018). Star identification based on hash map. *IEEE Sensors Journal*, 18(4):1591–1599.

Xu, L., Jiang, J., and Liu, L. (2019). Rpnet: A representation learning based star identification algorithm. *IEEE Access*, PP:1–1.

Yang, S., Liu, L., Zhou, J., Zhao, Y., Hua, G., Sun, H., and Zheng, N. (2022). Robust and efficient star identification algorithm based on 1-d convolutional neural network. *IEEE Transactions on Aerospace and Electronic Systems*, 58(5):4156–4167.

Zhang, G., Wei, X., and Jiang, J. (2008). Full-sky autonomous star identification based on radial and cyclic features of star pattern. *Image and Vision Computing*, 26(7):891–897.