Star Tracker with LSTM: Satellite Attitude Determination

Jacob Heglund*
University of Illinois Urbana-Champaign, Urbana, IL, 61801

I propose a system for determining the attitude of a satellite utilizing a star tracker that feeds output images into an LSTM neural network. The implemented system simulates the rotational kinematics of a satellite experiencing zero-torque, and generates realistic star tracker images using the Tycho celestial sphere dataset. The images output by the system are then taken as a time-series input to the LSTM. This system is still in its early stages, and while it does successfully simulate a star tracker attached to a rotating satellite, inefficiencies in image generation algorithm prevent production of sufficient datasets for training the LSTM network.

I. Introduction

One of the goals of supervised learning is to learn a mapping from given input-output pairs that then allows reasonable prediction on new inputs to the system. In fact, machine learning provides some guarantees that some functions may be approximated well enough for practical purposes. However, this is contingent on having large enough sets of labeled training data to allow the neural network to converge to a cost-function-minimizing map taking inputs to outputs. In addition, further data must then be used to verify the network to ensure over-fitting does not occur. With the advent of deep reinforcement learning, neural networks can learn controllers for complex systems without creating a dynamical model of the system itself. However, controlling a system's dynamics is only half of the battle.

It is often the case that determining a system's state using sensors shares equal importance with controlling the state. In the case of a satellite, knowledge of the system's rotational position, or attitude, may be determined through a combination of magnetometers, gyroscopes, sun-sensors, Earth-sensors, and star trackers. Satellites may utilize a wide variety of of sensors in tandem using sensor-fusion algorithms, and these systems offer great robustness to sensor failure. In the pursuit of designing robust sensor systems, this paper examines the applications of machine learning to the problem of attitude determination.

The work of this paper focuses on the specific case of star trackers and LSTM neural networks, but it is as part of a larger context of developing sensor systems that are both robust to individual sensor failure and accurate in attitude determination. We will discuss the successes in implementation of the star tracker simulation, as well as future work to be done in generating datasets for training the LSTM.

II. Literature Review

This paper comprises work involving several fields of study including astrodynamics, star tracker design, and machine learning. I therefore searched for relevant papers from these fields of study in order to build a base of knowledge to draw from during the project.

A. Astrodynamics and Star Tracker Design

A major problem in the field of astrodynamics with respect to star trackers is that of finding a satellite's attitude, or rotation of the satellite with respect to some other reference frame, from images of the celestial sphere. A survey of publications in the field of astrodynamics in [1] shows that there are two classes of algorithms in the field of star tracker design. *Recursive algorithms* solve the attitude determination problem given some prior knowledge of the attitude. A version of this algorithm is proposed in [2], and shows quick convergence properties. Meanwhile, *lost-in-space algorithms* solve the same problem with no prior attitude information. The latter problem is the type for which the work of this paper proposes a solution.

A star tracker architecture comprises of two main systems: a camera system for capturing images and performing processing of the images, and an on-board database system containing information needed to determine the attitude from images produced by the camera system. This has been the standard architecture used since compact CCD sensors made star tracker systems viable for satellite use in the 1970's. Modern research in the field of star tracker design can

*Undergraduate Senior, Engineering Physics, jheglun2@illinois.edu

focus on either of the two aspects of a star tracker architecture, referred to as feature extraction and catalog search in [3]. Modern methods of feature extraction focus on ensuring quick identification of stars in the image. Several feature extraction techniques are shown in [2], [4], [5]. More comprehensive surveys of star tracking algorithms can also be found in [3] and [1].

B. Star Trackers and Neural Networks

While the architecture of a camera system and database is one that is classically used, machine learning provides another method for approaching the attitude determination problem. The earliest paper published proposing a neural network architecture for star identification was [6]. Other papers examining the topic of neural networks applied to star tracking have been steadily published throughout the 1990's and 2000's including [7], [8], and [9]. These papers show similar results in that neural networks can be used for star tracking, but concede that the well-studied "classical" method shows better results in practice. Only in recent years have neural networks been able to show similar performance as a database search, as shown in [10] that combine classical geometric methods with neural network methods.

C. Deep Neural Networks and LSTM

The most compelling argument for implementing a neural network as a method of star tracking comes from the successes of deep learning in other fields of study. Beginning in 2012 with [11], there has been a great increase in the amount of research and successful results produced in the field of machine learning and AI. In addition, recent developments in machine learning APIs such as TensorFlow, PyTorch, and Keras have allowed researchers in other fields to more easily implement machine learning solutions in their own projects. Finally, breakthroughs in parallel processing using high-powered GPUs allow modern neural networks to show never-before-seen performance.

III. Problem Formulation

The autonomy problem formulation goes as follows: given a satellite equipped with a star tracking system, solve the lost-in-space problem by having the satellite's estimate of its attitude and rotational rate converge to an accurate measurement of the attitude using some other external system. We are currently hopeful about the feasibility of this problem, since each sector of the sky provides a unique set of stars that can be learned and associated within the neural network. With respect to the particular problem of star tracking, a recurrent neural network (RNN) provides an excellent architecture. This is because the input to the network occurs in a time-series, meaning that each input to the network occurs a given amount of time after the previous input. In our case, a typical star tracker takes photos around two frames per second (FPS), and a sweep of the sky over some fixed amount of time would constitute a single trial.

As stated in [12] however, vanilla RNNs have the issue of vanishing gradients during back-propagation, meaning the information that changes the weights in the network disappears before reaching the beginning of the network. This paper also suggests the LSTM architecture as a solution to the vanishing gradient problem, so this will be part of our implementation. Modern research in machine learning has also developed another architecture called the Gated Recurrent Unit [13]. They show similar performance as LSTM systems, and may be an interesting future extension of the work presented in this paper.

IV. Methodology

A. Celestial Sphere Simulation

The majority of the work done in this paper focuses on the implementation of a star tracker simulation. My first choice was to directly create an image of the celestial sphere from data given by a star catalog (any star catalog typically works). However, due to my inexperience in dealing with astronomical datasets, the Tycho* † dataset was chosen instead. This dataset provided the benefit of pre-rendered images that could readily be used in the star tracker simulation.

This image utilizes a *plate carree* projection, meaning that right ascension and declination, which are the equivalent of latitude and longitude on the celestial sphere, can be found from coordinates on the image by the following. Note the

*Information about the Tycho dataset may be found here: https://svs.gsfc.nasa.gov/cgi-bin/details.cgi?aid=3572

†Images used in the star-tracker simulation may be found here: https://svs.gsfc.nasa.gov/vis/a000000/a003500/a003572/

x, y coordinate system has its origin at the center of the image.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha \\ \delta \end{bmatrix} \tag{1}$$

B. Satellite Dynamics Simulation

The equations of motion for a rotating body under constant are given by

$$\theta = \theta_{\mathrm{i}} + \omega_{\mathrm{i}} t + \frac{1}{2}\alpha_i t^2 \tag{2}$$

$$\omega = \omega_{\mathrm{i}} + \alpha_{\mathrm{i}} t \tag{3}$$

We now take θ and ω as vectors to describe the rotation of the satellite in three dimensions. We can now define the attitude of the satellite in terms of its pose with respect to a stationary, initial frame. This is given in the form of matrix

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{4}$$

where the matrix R is a rotation matrix in the form

$$R = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 \end{bmatrix} \tag{5}$$

and the translation vector p is given by

$$p = \begin{bmatrix} p_{x,1}^0 \\ p_{y,1}^0 \\ p_{z,1}^0 \end{bmatrix} \tag{6}$$

Using this formulation, we can describe an arbitrary rotation as the multiplication of the pose of the satellite by three rotation matrices. Using the Euler Angle formulation, we therefore have the pose of the satellite being rotated to any position represnted as

$$T_2 = R_x R_y R_z T_1 \tag{7}$$

where the $R_i, i \in \{x, y, z\}$ matrices are rotations about the x, y, and z axes of the stationary frame and are given by

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \text{-sin}(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \tag{8}$$

$$R_y = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ \text{-sin}(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \tag{9}$$

$$R_z = \begin{bmatrix} \cos(\theta_z) & \text{-sin}(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

By then using spherical coordinates and defining the axis along which the star tracker points as the x axis of the satellite, the right ascension α and declination δ of the star tracker can be found by the following

$$x_1^0 = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} \tag{11}$$

$$\theta_s = \arccos\left(\frac{z_s}{\sqrt{x_s^2 + y_s^2 + z_s^2}}\right) \tag{12}$$

$$\phi_s = \arctan\left(\frac{y_s}{x_s}\right) \tag{13}$$

$$\begin{bmatrix}\alpha \\ \delta\end{bmatrix} = \begin{bmatrix}90^\circ - \theta_s \\ \phi_s\end{bmatrix} \tag{14}$$

C. Transforming to Image Coordinates

We now find a transformation between the coordinates with origin at the center of the image and the pixel coordinates centered at the top-left of the image. Given a vector containing the right ascension and declination

$$q = \begin{bmatrix}\alpha \\ \delta \\ 0 \\ 1\end{bmatrix} \tag{15}$$

we first scale by the ratio of number of pixels in the image to the angular distance spanned by the image.

$$q_{pixelConversion} = \begin{bmatrix}w/360 \\ h/180 \\ 0 \\ 1\end{bmatrix} \tag{16}$$

where w is the width of the image in pixels, and h is the height of the image in pixels. Taking the Hadamard product of q and $q_{pixelConversion}$, then taking the floor of each entry in the result yields a vector in pixel coordinates with origin at the center of the image. We finally transform to pixel coordinates centered at the top left of the image by multiplying by a homogeneous transformation matrix

$$T = \begin{bmatrix}1 & 0 & 0 & w/2 \\ 0 & -1 & 0 & h/2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1\end{bmatrix} \tag{17}$$

We now have all the necessary mathematical machinery to go from continuous, celestial coordinates α and δ to discrete pixel coordinates centered at the top-left of the image . 1. This means we can find the field of view of the star tracker from a given pose of the satellite, exactly what we wanted!

D. LSTM Data Manipulation

The data from the simulation is presented as a time-series, and a single trial has twenty, 920x920 pixel images. Each trial has an associated rotational velocity ω, given by

$$\omega = \begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix} \tag{18}$$

The input-output pairs of each trial can then be compressed together into a tensor using TensorFlow's data handling functions. Since the data is created in a controlled environment, and each image is the same size, there is little other data cleanup that is required before inputting the data to the LSTM. A TensorFlow Dataset object can then easily be generated and acts as the interface between the data and the LSTM network.

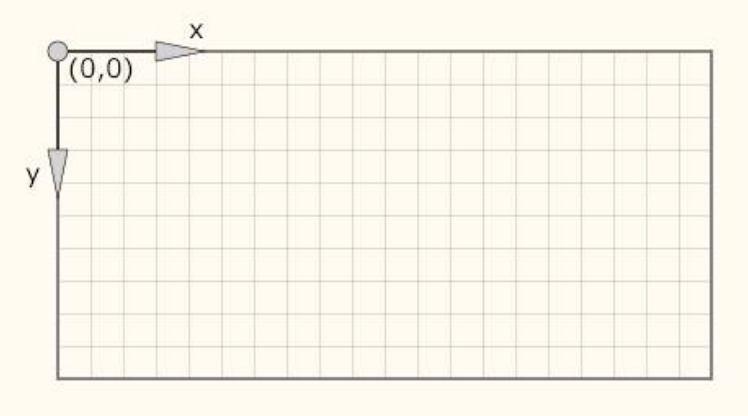

Fig. 1 The coordinate system for pixels in the image.

V. Experiments and Results

A. Star Tracker Simulation

At a single time-step in the star tracker simulation, the image of the celestial sphere and rotational kinematics shown in IV are used to produce an image of the star tracker's field of view (FOV). The output image is 920x920 pixels, and an example of the output can be seen in 2. Additionally, another image showing the satellite's FOV in the context of the celestial sphere can be specified as an output. In this image, the black square is once again 920x920 pixels, and the entire image is 16384x8192 pixels. An example of this output can be seen in 3.

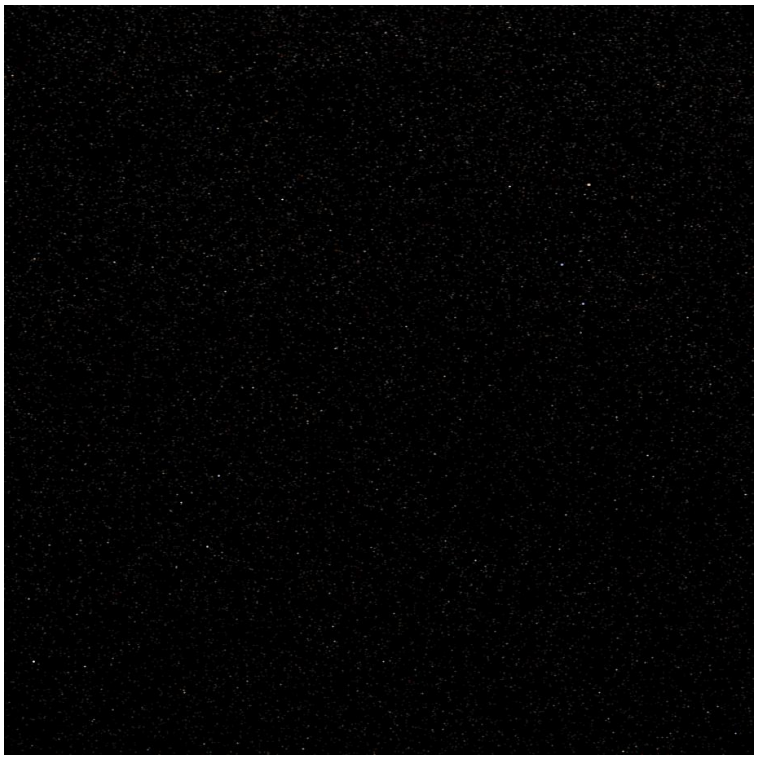

Fig. 2 The star tracker's field of view as an output of the satellite rotational dynamics simulation.

Fig. 3 The star tracker's field of view in the context of the celestial sphere as an output of the star tracker simulation.

B. LSTM - The Curse of Dimensionality

While a small amount of randomized training data was successfully produced by the star tracker simulation, it must be noted that a 10 second simulation of the star tracker's FOV, which produces a total of 20 images at 2 frames per second (FPS), takes around 9 real-world minutes to complete. Therefore, a set of 10 trials took 80-100 minutes to render. This inefficiency stems from the methods used to produce the star tracker FOV in the simulation and must be improved before any neural network testing can begin. In training a neural network, more data often leads to better performance. In this case, several hundred trials with randomized initial conditions would have to be generated as training and validation datasets, and given the performance of the star tracker simulation, this is not currently feasible.

VI. Conclusion

Before a proper test of the neural network architecture as a means of determining attitude and rate from star tracker images, a faster method of simulating the FOV of the star tracker must be developed. While it is disappointing no conclusive results from the neural network could be produced, this project was an overall success in being one of my first independent projects. In addition, this project provided real-world experience in debugging problems as they arose, and thinking about the high-level structure of the project itself.

I was also able to apply many of the techniques I have learned in courses over the past fours years of my undergraduate career, making it a satisfying conclusion to one of the chapters in my life.

Acknowledgments

I would like to thank Dr. Alexander Ghosh for his support throughout the course of this project. His expertise and guidance in the field of satellite navigation served as inspiration for this project. I also thank Vedant for bringing the idea of neural networks and their applications to attitude determination and control systems to my attention, as well as for his support in the early stages of the project.

References

[1] Spratling, B. B., and Mortari, D., "A survey on star identification algorithms," *Algorithms*, Vol. 2, No. 1, 2009, pp. 93–107.

[2] Samaan, M. A., Mortari, D., and Junkins, J. L., "Recursive mode star identification algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 41, No. 4, 2005, pp. 1246–1254.

[3] Ho, K., "A survey of algorithms for star identification with low-cost star trackers," *Acta Astronautica*, Vol. 73, 2012, pp. 156–163.

[4] Delabie, T., Durt, T., and Vandersteen, J., "Highly robust lost-in-space algorithm based on the shortest distance transform," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 476–484.

[5] Pham, M. D., Low, K.-S., and Chen, S., "An autonomous star recognition algorithm with optimized database," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 49, No. 3, 2013, pp. 1467–1475.

[6] Alvelda, P., and San Martin, A. M., "Neural network star pattern recognition for spacecraft attitude determination and control," *Advances in Neural Information Processing Systems*, 1989, pp. 314–322.

[7] Hong, J., and Dickerson, J. A., "Neural-network-based autonomous star identification algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 4, 2000, pp. 728–735.

[8] Trask, A., and Coverstone, V., "Autonomous artificial neural network star tracker for spacecraft attitude determination," *Advances in the Astronautical Sciences*, Vol. 114, No. SUPPL., 2003, pp. 1315–1335.

[9] Li, C., Li, K., Zhang, L., Jin, S., and Zu, J., "Star pattern recognition method based on neural network," *Chinese Science Bulletin*, Vol. 48, No. 18, 2003, pp. 1927–1930.

[10] Miri, S. S., and Shiri, M. E., "Star identification using Delaunay triangulation and distributed neural networks," *International Journal of Modeling and Optimization*, Vol. 2, No. 3, 2012, p. 234.

[11] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] Gers, F. A., Schmidhuber, J., and Cummins, F., "Learning to forget: Continual prediction with LSTM," 1999.

[13] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y., "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.