CS230

Angles-Only Relative Orbit Determination Using Deep Recurrent Neural Networks

Josh Sullivan[*], **Simone D'Amico**[†]
Space Rendezvous Laboratory
Department of Aeronautics & Astronautics
Stanford University
jasulliv@stanford.edu

Abstract

This project focuses on the problem of vision-based relative navigation in distributed spacecraft systems, with a focus on estimating the relative orbital motion of a target space object using deep recurrent neural networks. Since the estimation problem makes use of simple measurements consisting only of the bearing angles describing the pixel coordinates of the target in the image, the problem is inherently observability constrained. This observability dilemma, compounded by the fact that the dynamical system is highly nonlinear and therefore difficult to analytically invert for a solution, has motivated the use of recurrent neural networks. The recurrent framework is chosen in order to learn the complex relationship mapping measurements to relative orbit estimate by processing temporally-ordered batch measurement data. The resulting estimates obtained after a series of training campaigns shows performance on par with, and in certain cases, even better than the current state-of-the-art approaches in the existing literature.

1 Introduction

There is an increasing interest among the space science and engineering communities in using distributed systems of small spacecraft to achieve tasks that would be difficult or impossible with traditional monolithic satellite platforms. Examples of such *distributed space systems* (DSS) include on-orbit servicing of defunct satellites, distributed aperture sensing/science, distributed observation for Earth monitoring and space situational awareness, and many others. While DSS pose many distinct advantages over single-spacecraft approaches, including improved robustness and resource efficiency, they are held to much stricter requirements on navigation and control accuracy. Vision-based navigation is considered an apt metrology system for DSS with small satellites since it provides a high-dynamic-range, passive, and robust sensing capability using a single camera with a small form-factor and low power requirement. These advantages come at the added cost of complex navigation algorithm development. In particular, when using a single monocular camera for estimating the relative motion between the observing spacecraft and target space objects, the navigation algorithm must infer 6D state information from only 2D measurement information (the bearing angles subtending the line-of-sight vector) since no range is provided by the sensor. Accordingly, the associated "observability" problem has been a major hurdle to efficient implementation of the so-called "angles-only" navigation approach.

[*]Doctoral Candidate
[†]Research Adviser, Director of the Space Rendezvous Laboratory

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

This work focuses on a completely novel application of recurrent neural networks (RNN) for estimating the target relative motion at far separations (tens of kilometers) in Earth orbit. In particular, the estimation approach is posed as a supervised learning problem using synthetically generated orbital data for the observer and target, and a realistic camera sensor emulator to provide representative noise corrupted measurements from the orbital data. The input to the algorithm for each training example is a batch sequence of temporally-ordered measurement features consisting of the the observers orbital state and attitude state with respect to an Earth-centered inertial reference frame, as well as the bearing angles describing the target centroid location in the image frame. Together, these input features account for twelve quantities at each measurement epoch in the sequence. The output of the algorithm is an estimate of the targets relative orbital motion (i.e., relative position and velocity) with respect to the observer at the last measurement epoch of the sequence. The ground truth relative state is known from simulation and used to frame the supervised learning optimization.

2 Dataset and Features

As briefly mentioned, the data being used in this project comes from a high-fidelity orbit propagation software developed in the author's laboratory. This numerical orbit simulator allows for very accurate trajectories of realistic spacecraft motion to be generated, and also contains representative sensors models to provide the pixel coordinates of the target as seen by the observers camera. Thirty thousand data examples consisting of twenty batch sequences and thirty thousand data examples consisting of forty batch sequences were generated using this framework. The discretization between measurement sequences in a batch varied between two and ten minutes from example to example (but was constant within an example). An illustration of several training example batches of bearing angles in the observers image frame are shown in Figure 1. An 80% training/10% dev/10% test split was used for algorithm optimization and validation. The training data was normalized to zero mean and unit variance for each input feature, and the same normalization transformation was applied to the dev and test sets to ensure improved algorithm optimization during gradient descent. As discussed in the introduction, the input feature data at the j^{th} measurement epoch of a batch is defined as:

$$\mathbf{x}_j = (a,\, e,\, i,\, \Omega,\, \omega,\, M,\, q_0, \mathbf{q},\, \alpha,\, \epsilon)^T \in \mathbf{R}^{12} \tag{1}$$

where the set $(a,\, e,\, i,\, \Omega,\, \omega,\, M)^T$ consists of the classical Keplerian orbital elements describing the orbit of the observer, $(q_0,\, \mathbf{q})^T$ are the observer's attitude quaternion components, and $(\alpha,\, \epsilon)^T$ are the bearing angles of the target with respect to the observer. The former two quantities are immediately available from the orbit propagation software, while the latter quantity is obtained by passing the orbital data through a camera emulator which generates the target pixel clusters in the image frame. The output vector corresponds to the relative state of the target with respect to the observer at the end of the measurement batch, and is parameterized using a set of relative orbital elements:

$$\mathbf{y} = a\,(\delta a,\, \delta\lambda,\, \delta e_x,\, \delta e_y,\, \delta i_y,\, \delta i_x)^T \in \mathbf{R}^6 \tag{2}$$

These quantities are shown here for completeness; an in-depth discussion of why they have been chosen can be found in several works on similar topics [1-3]. The main takeaway is that the algorithm learns the mapping $\mathbf{h} : \mathbf{R}^{12p} \mapsto \mathbf{R}^6$ for p measurement epochs per example, with $\mathbf{y} = \mathbf{h}(\mathbf{x})$.

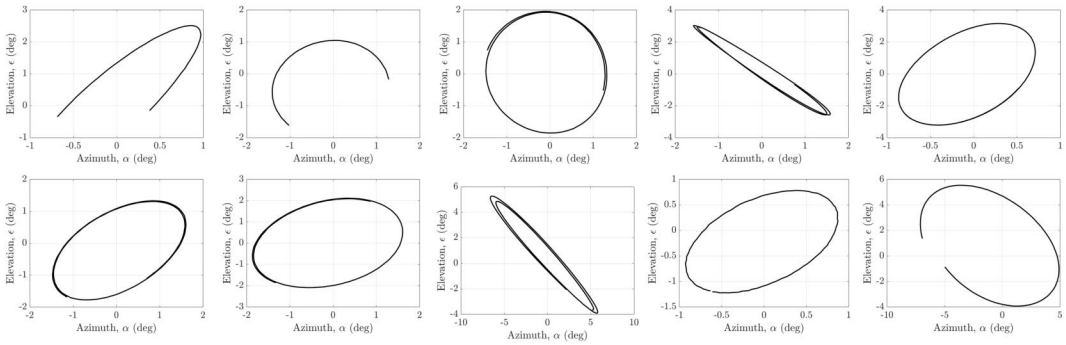

Figure 1: Sample of 10 bearing angle sequences captured by synthetic camera with simulated orbits.

3 Methods

This work leverages a deep RNN network consisting of long short-term memory (LSTM) cells followed by a single, fully-connected layer. See Figure 2 for a graphical depiction of the architecture. The use of RNNs is largely motivated by the fact that temporally-ordered sequences of data are used in each training example. LSTMs are chosen to provide persistence of information synthesized from input sequences at earlier measurement epochs when formulating estimates near the end of the measurement batch. Unlike the majority of the RNN-based problems explored in the context of CS230, the estimation is fundamentally a regression problem whereby the continuous-valued input features are used to infer continuous-valued outputs.

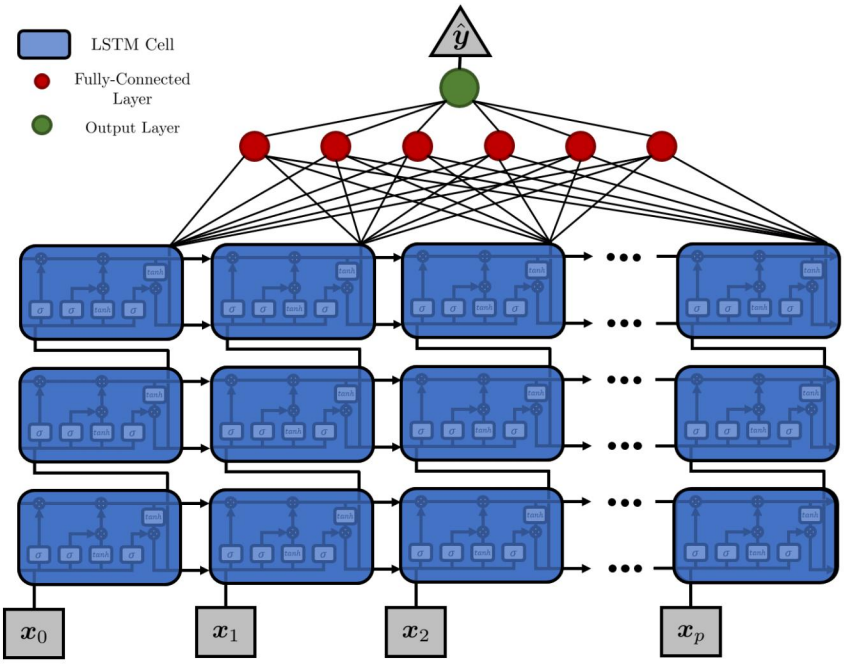

Figure 2: Deep recurrent neural network architecture with LSTM cells.

The LSTM cells consist of several parameters which are optimized over the course of the training campaign. Figure 2 shows an unrolled (left-to-right) view of three RNN layers composed of LSTMs. There are essentially two "streams" of information flowing through the LSTM cell: a cell-state and the output estimate from the previous LSTM cell (or the initialized state). Internally, the LSTM decides how to propagate information from the previous input epoch into the current cell state by using a series of gates, denoted as the forget gate, the input gate, and the output gate [4, 5]. The weights corresponding to each of these gate paths are what is optimized. For this architecture, the loss function is chosen as the *least absolute deviations*, or the ℓ-1 loss:

$$\mathcal{L}\left(\mathbf{y}^{(j)}, \hat{\mathbf{y}}^{(j)}\right) = ||\mathbf{y}^{(j)} - \hat{\mathbf{y}}^{(j)}||_1 = |\mathbf{y}^{(j)} - \hat{\mathbf{y}}^{(j)}| \tag{3}$$

This loss function is chosen primarily for robustness to potential data outliers, since penalties scale linearly even for large residuals (as compared with the quadratic scaling in the well-known least-squares, or ℓ-2, loss). The Adam protocol is chosen as the workhorse numerical optimizer.

4 Experiments, Results, and Discussion

The main experiment conducted in this project is a hyperparameter search over learning rate, RNN depth, number of LSTM units, and length of measurement batch for each example. This experiment is intended to provide a high-level intuition into what the influencing (or non-influencing) hyper-parameters are for this scenario. A nominal test scenario is setup with an optimization learning rate of 0.1, an RNN depth of 5 layers consisting of 128 LSTM units, to process a batch of twenty

sequences of measurement input features. Using this nominal configuration, each hyperparameter is varied independently while holding the others constant. The learning rate is varied over the values $\{0.001,\ 0.01,\ 0.1\}$. The RNN depth is varied over the values $\{1,\ 5,\ 10\}$. The number of LSTM units is varied over the set $\{64,\ 128,\ 256\}$. Finally, the number of measurement epochs per batch is varied over the set $\{20,\ 40\}$. In each hyperparameter configuration, training is conducted for 500 epochs.

The quantitative results for this hyperparameter search are included in the table shown in Figure 3. These results consist of the average test accuracy computed over all three thousand test examples. The accuracy metric used in this work is similar to the loss metric; that is, the accuracy is given by the average ℓ-1 norm of the estimation error:

$$\mathcal{A} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} |\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}| \tag{4}$$

where m_test is the number of training examples. This accuracy metric is chosen because it gives a good indication of how well the algorithm is estimating the range (in units of distance) between the target and the observer. Recall that no inter-spacecraft range information is provided as a feature, and in general it is this lack of range information that causes the problem to be very poorly observable. The ability to reconstruct range information from only a sequence of images represents a distinctly advantageous estimation capability in space. For the knowledge of the reader, nearly all objects in space are cataloged and coarsely tracked by the North American Aerospace Defense Command (NORAD), and generally their orbits are known to within approximately five kilometers (km). Instead, some algorithms for space-based imaging and orbit estimation have shown accuracy at the two to three kilometer level in certain constrained circumstances.

Hyperparameter	Case 1	Case 2	Case 3
Learning Rate	5.37/5.09/5.13 km	3.85/3.64/3.78 km	2.66/2.44/2.45 km
RNN Depth	7.45/7.39/7.42 km	2.66/2.44/2.45 km	2.01/1.94/1.89 km
LSTM Units	2.71/2.69/2.86 km	2.66/2.44/2.45 km	2.76/2.71/2.63 km
Temporal Sequences	2.66/2.44/2.45 km	1.70/1.68/1.64 km	-

Figure 3: Average relative orbit estimation train/dev/test accuracy results. Each hyperparameter was varied while the others remained fixed at the nominal test values. 500 training epochs were used.

In general, there are some interesting trends to be noted in the results. First, the choice of learning rate has substantial influence on the test accuracy using a model trained over 500 epochs. Increasing the learning rate from 0.001 to 0.1 produces an approximately 52% improvement in test accuracy. This result alone, however, is not too indicative of whether 0.1 is a "more optimal" choice for the learning rate since only 500 training epochs were conducted (these ran fairly fast). Note that, while 2.45 km of error is on par with what current state-of-the-art is able to deliver and generally a bit better than NORAD tracking, this indicates a fair amount of bias in the model. Since the train and dev accuracy indicate the network was not overfitting the data, a working hypothesis is that training the network for more epochs with a lower learning rate may indeed result in better performance.

The most surprising result comes from comparing the accuracies over the different RNN depth choices. By going from one RNN layer to five layers, the test accuracy improves by a factor of just under 75%. Instead, the improvement gained by using ten layers instead of five is only about 23%. Accordingly, there are two effects to be seen here: (1) adding more layers seems to improve estimation accuracy, and (2) this improvement effect has diminishing returns with deeper networks. Speaking to the first effect, the interpretation is that the network needs more layers (at least more than one) to reasonably learn the complex nonlinear relationship between the underlying spacecraft relative motion dynamics and the input measurement sets.

Unlike the previous hyperparameter sweep, varying the number of LSTM units in each cell (while using five RNN layers) does not appear to have a substantial effect on the accuracies overall. A likely explanation for this comes simply from the fact that the system under consideration is fairly low dimensional. For the nominal case, the entire input batch consists of only 240 features (twenty sequential twelve-dimension measurement vectors). Thus it is somewhat expected that adding more

LSTM units does not heavily influence accuracy, since the measurement information is not "rich" enough to be useful to many more units.

Finally, the last hyperparameter that is varied is the number of measurement epochs per input feature batch. In an operational scenario, this is equivalent to allowing the observing spacecraft to accumulate more measurements of the target prior to making an estimate. As expected, the accrual of more feature data in each batch enables improves accuracy overall- by doubling the number of measurement epochs, the network obtains a 33% accuracy improvement. This is generally to be expected in any batch estimation problem, but it is important to note that adding more measurements generally comes with diminishing returns as the computational complexity eventually outweighs the marginal improvements in estimation accuracy.

5 Conclusion and Future Work

Overall, the intent of this project was to develop a framework for relative orbit estimation from a space-based observer spacecraft using recurrent neural networks composed of long short-term memory cells. The general incentive was to: (1) gain an intuition for applying recurrent neural networks to a regression problem, (2) understand which hyperparameters are (non-)influencing on the estimation accuracy, and (3) comparing the overall test performance with the current state-of-the-art. Note that, to the author's knowledge, no other works in literature have applied such an approach to the problem at hand. The validation results presented here indicate the ability to return estimation results that are on par with, and in several cases, better than other proposed methods or current NORAD tracking capability. In the best cases, the neural network approach developed in this project provided average errors that were under two kilometers, which is more than sufficient to initialize a sequential estimation filter (Kalman filter or otherwise). The most drastic improvements were seen when several RNN layers were used to learn the complex relationship between the underlying dynamics and measurements. The best performance was seen when the five layer network consisting of 128-unit LSTM cells was provided double the measurements that the nominal case was given. Future work will expand upon the hyperparameter search to a more refined parameterization (instead of just three test cases per hyperparamter) in order to better understand the lasting implications in a more widespread sense. Additionally, a lengthier training campaign will be done over many more epochs using smaller learning rates to better characterize the convergence properties of the algorithm. Convergence plots and trends will be analyzed (the author had terrible luck with TensorBoard while conducting this experiment). Finally, the author would like to explore the option of creating a "customized" LSTM cell that uses some domain-specific knowledge of the underlying dynamics to perhaps improve performance more efficiently.

6 Contributions

This work was solely conducted by Josh Sullivan, PhD candidate, with general research advisement coming from Professor Simone D'Amico of the Space Rendezvous Laboratory in the Aeronautics & Astronautics Department of Stanford. Code for this project can be found at:
https://github.com/joshua-sullivan/sw-deep-irod

References

[1] Sullivan, J. & D'Amico, S. (2017) Nonlinear Kalman Filtering for Improved Angles-Only Navigation Using Relative Orbital Elements. *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 9, pp. 2183-2200

[2] Sullivan, J. Grimberg, S., & D'Amico, S. (2017) Comprehensive Survey and Assessment of Spacecraft Relative Motion Dynamics Models. *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 8, pp. 1837-1859

[3] D'Amico, S. (2010) Autonomous Formation-Flying in Low Earth Orbit. Ph.D. Thesis *TU Delft*

[4] Olah, C. (2015) Understanding Long Short-Term Memory Networks. *Colah's Blog: http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

[5] Graves, A., Mohamed, A.R. & Hinton, G. (2013) Speech Recognition with Deep Recurrent Neural Networks. In Acoustics, Speech and Signal Processing (ICASSP), *2013 IEEE International Conference* pp. 6645-6649.