Finding Satellite Direction from Time Difference of Arrival Data Using ResNet

Anthony Iannuzzi

***Abstract*—Satellite tracking is a key part of space operations, enabling mission success. Time Difference of Arrival (TDoA) data offers a cheap, open-source solution to tracking satellites, but suffers from input noise. Our goal is to determine whether ResNet101 can solve for the satellite direction accurately while also being resistant to uncertainties in each station's location and local time. Satellite localization with TDoA requires a network of three or more antennas. Because the satellite is far out-of-plane relative to the stations, we consider using TDoA for direction finding in the form of azimuth and elevation angles. We train ResNet101 based on hyperbola plots. Using in-house software, we create the hyperbolas using simulated time differences and constant station locations, both superimposed with random noise. We train ResNet101 on 100k generated images and evaluate it on two 2.5k image test sets: one with no error and one with error. ResNet101 performs poorly, achieving a mean azimuth error of 88° and mean elevation error of 19° for both test sets.**

I. Introduction

Before launching a CubeSat (U-class spacecraft), universities need a method of tracking their satellite. Open-source networks like Satnogs[1] can be used for downlink, but open-source stations with Orbit Determination (OD) capability are not widely accessible. OD is a key component of space situational awareness and helps decrease the risk of collisions. Universities without a station must rely on commercial and government entities for OD, which may be too infrequent. James and Maisonobe [1] show that Satnogs stations could have OD capability based on Doppler Shift measurements. Time Difference of Arrival (TDoA) is another alternative that uses several stations to find satellite direction, enabling OD capability.

TDoA works by recording the time of arrival of a signal at 3 time-synchronized stations and calculating a signed difference of that arrival time. The time difference is converted to a differential distance by multiplying by the speed of light. This differential distance constrains the possible locations of the emitter to a hyperbola [2] as opposed to a circle when using time of flight calculations. The area of intersection of three or more hyperbolas dictate the possible emitter locations. The solution to TDoA is sensitive to uncertainty in each station location and local time. Our goal is to determine whether ResNet101 [3] can reduce TDoA sensitivity to noise while maintaining similar accuracy as classic TDoA solvers for satellite direction finding.

II. Background

Prior work demonstrates that machine learning algorithms can decrease TDoA sensitivity to noise. Palaniswarmi et al. [4] increase TDoA resistance to errors using a Support Vector Regression algorithm for an unmanned aerial vehicle formation flying application. To reduce TDoA sensitivity in multipath environments, Zhang et al. [5] train an ensemble of 4 neural networks with ant lion optimization, and Niitsoo et al. [6] train a deep neural network on raw channel impulse response. Li et al. [7] optimize a kernel-based machine learning algorithm and Luo et al. [8] use an extreme learning machine to improve TDoA accuracy in non-line-of-sight environments. Cho et al. [9] train a deep network to reduce measurement errors, which in turn reduces TDoA error.

For satellite tracking, we use a topocentric coordinate system, where the coordinates represent $(x_{east}, y_{north}, z_{up})$. For satellite tracking, TDoA data produces hyperboloids, the 3D analog to a hyperbola. Three stations result in three hyperboloids. With no error, these hyperboloids intersect along a hyperbola. The TDoA symbolic solver [10] approximates the hyperbola by decomposing the 3D problem into three 2D planes, and fitting a line through the solution on each plane. Hyperbolas approach a line far from their foci. The planes are $z = 50$km, 400km, and 1200km above the primary station. This method is valid for high z values, which requires satellites with elevation $> 15^\circ$. Equation (1) describes a hyperboloid where δ_{ab} is the differential distance between station a and b and D is half the distance between a and b.

$$x = \delta_{ab}\sqrt{\frac{1}{4} + \frac{y^2}{(4D^2 - \delta_{ab}^2)} + \frac{z^2}{(4D^2 - \delta_{ab}^2)}} \tag{1}$$

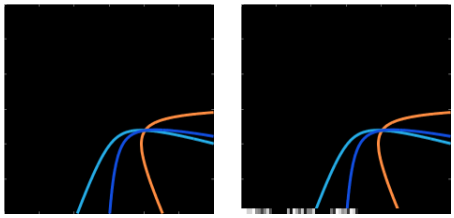
For more information on TDoA and the symbolic solver, see Knott and Edge [2] and the RIT P20151 TDoA presentation [10]. We benchmark ResNet101 performance against the symbolic solver.

III. Proposed Method

We train ResNet101 on a single plane, $z = 400$km. We feed it generated images of the hyperbolas on that plane. In-house MATLAB code generates the data set. The code chooses a random location of the satellite in the form of an azimuth angle between 0° and 360° and an elevation angle between 15° and 85° with respect to the primary station. Satellite range is curve fit, following [10]. The ground truth is then the (x, y) value for that direction described by (2) where El is elevation and Az is azimuth.

$$x = \frac{\sin Az}{\tan El},\ y = \frac{\cos Az}{\tan El} \tag{2}$$

[1] https://satnogs.org/



(a) Image of the three hyperboloids on the 1200km z plane.

(b) Identical image with barcode encoding the time differences used to create the hyperboloids.

Figure 1: Training image examples.

The plane does not appear in (2) because (2) is plane-normalized. The actual (x, y) point in 3D space can be calculated by multiplying the output of ResNet101 by the plane-normalized coefficient. To account for errors, the MATLAB code perturbates the TDoA data by a random value, selected from a Gaussian distribution with a mean of zero. For receivers and time differences, the standard deviations are 9m and 100ns. These are based off of realistic clock and location accuracy provided by a GPS. MATLAB plots the hyperbolas following (1) with $z = 400km$. Figure 1 shows two training images produced from the MATLAB code. Images are 224x224x3 pixels, sized for ResNet101. We try encoding the TDoA data using a barcode to improve accuracy.

All training images have the same xy axes ranges. The plane and the smallest allowable elevation, define the range of the (x, y) coordinate axis as described in (3).

$$X_{max} = \frac{400,000m}{tan(el = 15°)} \tag{3}$$

We train with the MATLAB Deep Learning Toolbox. We change the output layer of the ResNet101 network to 2 node regression. Training uses 1k, 10k, 25k, and 100k images with MATLAB's default loss function. We validate using 25% of the training image count, up to 2.5k images. The 2 final held-out test sets each have 2.5k images, one with noise and one with no noise. We train ResNet101 with and without a barcode which encodes the time differences. Using the trained ResNet101 model for 400km, we retrain using 1/10 the images for 50km and 1200km planes. We can derive direction by drawing a line from $(0, 0, 0)$ and (x, y, z) where the trained network outputs (x, y) and z is the plane the network is trained on.

IV. Results

ResNet101 learns the weights based on 3 epochs for each set of training images. For 100k images, the loss function levels out 25% of the way into the first epoch. We find that training with more than 10k images results in negligble gains in accuracy. Training on 100k images yields the lowest validation RSME, at 0.0806, a slight increase from training with 25k images at a RSME of 0.0909. Training uses a momentum ratio of 0.95 and an initial learning rate of 0.005 which drops by a factor of 10 every epoch. Training shuffles the images every epoch and uses a mini-batch size of 24 images.

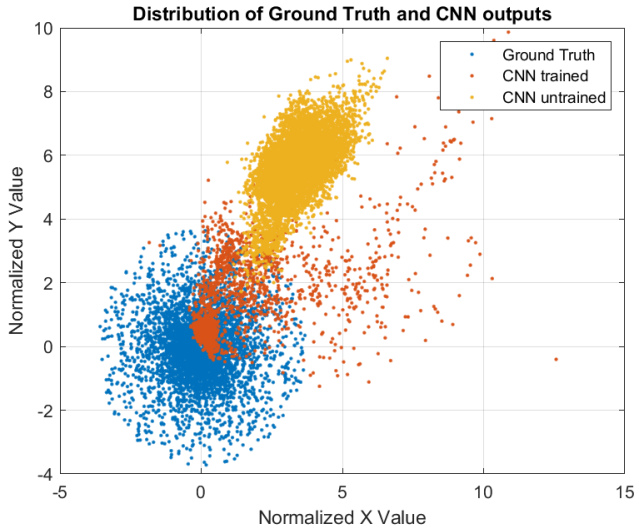


Figure 2: A map of the normalized (x, y) coordinates for each image in the held-out test set and the network estimates.

Table I: Summary of Az and El errors for various networks and the symbolic solver. BC is barcode.

	Symbolic Solver	No training	10k	BC, 10k	BC, 100k
$\Delta Az°$	19.9	90.2	87.8	88.1	88.7
$\Delta El°$	7.3	44.1	19.3	19.3	18.0

Figure 2 shows the the resulting error for the noisy held-out test set for the network trained on 100k images. The network cannot calculate $(-x, -y)$ values. Adding a barcode and an esemble of different planes do not correct this problem. This results in nearly all line fits to have an azimuth between $0°$ and $90°$, resulting in $\sim 90°$ error in azimuth. The maximum error in azimuth is $180°$. Table I tabulates network error for the noisy test set. The results from the no error test set are within $1°$ of Table I for any networks whereas the symbolic solver error decreases to $1.5°$. The untrained ResNet101 network (i.e. random weights) serves as a benchmark since the networks perform poorly. Compared to randomly guessing, training halves elevation error by more than 50%, but reduces azimuth error by only 3%. The network shows robustness to noise for elevation.

V. Conclusion

The architecture used in this paper is applicable to any 2D or 3D TDoA environment that has known receiver locations and time differences. The network may be retrainable to solve other graph intersection problems. The network has poor accuracy because it cannot calculate $-x$ and $-y$ values. Another approach might include two convolutional neural networks. The first net classifies which quadrant the intersection is in. Based on that output, MATLAB rotates the image to the first quadrant and uses the ResNet101 architecture presented here. At present, ResNet101 cannot accurately solve the TDoA problem.

REFERENCES

[1] N. Janes and M. Luc, "Orbital determination using satnogs and orekit." under review.

[2] M. Knott and A. Edge, "Passive geolocation with 3d tdoa: A crfs white paper," tech. rep., CRFS Inc., Chantilly, Virginia, 2019.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[4] M. Palaniswami, B. Sundaram, R. G. L. Jayavardhana, and A. Shilton, "Target localization using machine learning," in *Informatics in Control, Automation and Robotics II* (J. Filipe, J.-L. Ferrier, J. A. Cetto, and M. Carvalho, eds.), (Dordrecht), pp. 27–33, Springer Netherlands, 2007.

[5] Z. Zhang, F. Jiang, B. Li, and B. Zhang, "A novel time difference of arrival localization algorithm using a neural network ensemble model," *International Journal of Distributed Sensor Networks*, vol. 14, no. 11, p. 1550147718815798, 2018.

[6] A. Niitsoo, T. Edelhäußer, E. Eberlein, N. Hadaschik, and C. Mutschler, "A deep learning approach to position estimation from channel impulse responses," *Sensors*, vol. 19, p. 1064, Mar 2019.

[7] J. Li, I.-T. Lu, J. Lu, and L. Zhang, "Robust kernel-based machine learning localization using nlos toas or tdoas," pp. 1–6, 05 2017.

[8] X. Luo, X. CHANG, and H. LIU, "A taylor based localization algorithm for wireless sensor network using extreme learning machine," *IEICE Transactions on Information and Systems*, vol. E97.D, pp. 2652–2659, 10 2014.

[9] J. Cho, D. Hwang, and K.-H. Kim, "Improving tdoa based positioning accuracy using machine learning in a lorawan environment," pp. 469–472, 01 2019.

[10] A. Iannuzzi, "P20151 localize attribrute satellites in orbit: Tdoa algorithm." http://edge.rit.edu/edge/P20151/public/TDoA