Learned Initial Orbit Determination from Simulated Electro-Optical Observations

Alexander Cabello, Jeff Houchard, Cameron Harris
EO Solutions Corp
J. Zachary Gazak, Jonathan Kadan, Justin Fletcher
USSF Space Systems Command (A&AS)

ABSTRACT

Accurate Initial Orbit Determination (IOD) is crucial for Space Domain Awareness (SDA). This research introduces an IOD method that aims to enhance the orbit prediction accuracy of the initial detection of unknown space objects with short-arc angle-only measurements from electro-optical (EO) sensors. The methodology integrates a machine learning model with the principles of orbital mechanics. The model is trained on a diverse dataset of simulated observations across various orbital regimes, including Low-Earth Orbit (LEO), Medium-Earth Orbit (MEO), Geostationary Orbit (GEO), and Highly Elliptical Orbit (HEO). Comparative analyses demonstrate that the proposed approach outperforms traditional angles-only methods, such as Laplace, Gauss, and Gooding methods, with a median angular error reduction relative to the observer. This improvement enhances the reliability of subsequent tracking efforts. The network architecture features two Long Short-Term Memory (LSTM) layers followed by a fully connected (dense) layer, achieving optimal results when predicting position and velocity state vectors using a physics-based loss function. These findings underscore the potential of machine learning in advancing SDA capabilities.

1. INTRODUCTION

Initial Orbit Determination (IOD) is a fundamental problem in Space Domain Awareness (SDA). It involves estimating an object's orbit using limited observational data. Traditional angles-only methods require a fairly accurate initial guess or are often sensitive to observation errors, especially over short arcs. A critical mission for SDA sensors, particularly autonomous ones, is to detect and track uncorrelated targets (UCTs). When a UCT is detected, a reliable IOD is crucial for obtaining follow-up measurements. Improved orbit estimation enhances the probability of reacquiring UCTs and provides greater flexibility for the scheduler to retask sensors due to the larger time window available.

The primary objective of this study is to develop and evaluate a machine learning framework for IOD using simulated electro-optical (EO) observations. We aim to compare its performance against traditional methods and demonstrate its applicability across different orbital regimes. An additional objective is to create a robust simulation test bed for developing and evaluating IOD models.

2. RELATED WORK

2.1 Traditional IOD Methods

Traditional angles-only methods like the Gooding [8] algorithm have been widely used for years. These methods require a fairly accurate initial guess of the slant range, while others, such as Gauss [16] and Laplace [16] algorithms, are often sensitive to observation errors, especially over short arcs. Figure 1 shows orbits predicted by various IOD models with random samples of observation errors from the same observation, highlighting the challenges angles-only IOD algorithms face in estimating orbits with short arcs.

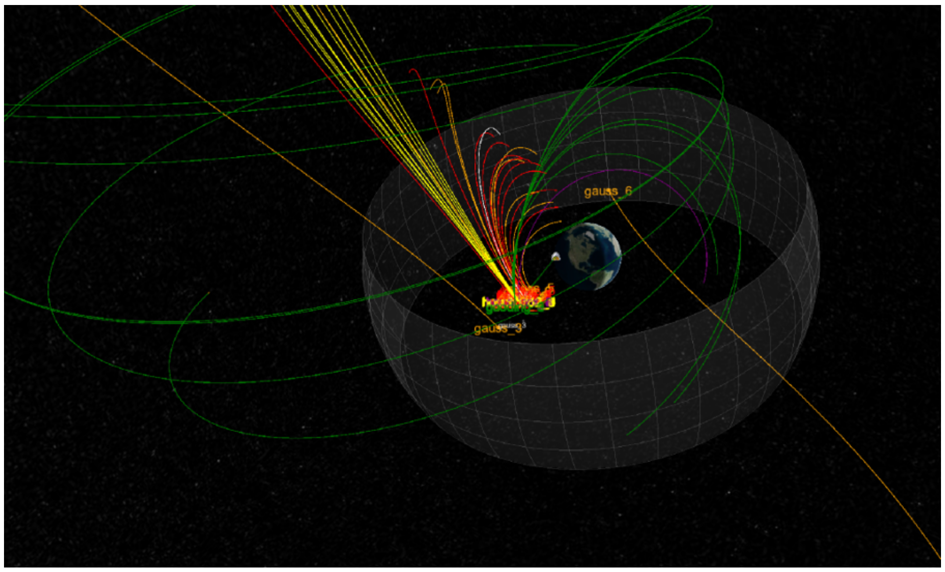


Fig. 1: Visualization of orbits predicted by various IOD models with random observation errors. The white orbit line represents the true orbit, while deviations in the predicted orbits illustrate the impact of observational noise on orbit parameters.

2.2 MACHINA IOD

In addition, we introduce a new heuristic IOD algorithm we developed for MACHINA [6]. This algorithm combines traditional IOD techniques with optimization methods that are particularly effective in estimating an initial orbit suitable for follow-up observations. Below is a description of the algorithm.

The algorithm uses sensor position vectors $\vec{r}_{\text{obs}}$ and the line-of-sight (LOS) vectors $\vec{s}$ at two times to estimate the satellite's position $\vec{r}$. The core idea is to determine coefficients k_0 and k_1 that scale the LOS vectors, such that the sum of the scaled LOS vectors and the sensor positions constrain the satellite's position to a circular orbit.

Given:

- $\vec{r}_{\text{obs},0}$ and $\vec{r}_{\text{obs},1}$ are the positions of the observers at times t_0 and t_1, respectively.
- $\vec{s}_0$ and $\vec{s}_1$ are the LOS vectors at times t_0 and t_1, respectively.
- k_0 and k_1 are the coefficients to be determined.
- Δt is the time of flight between observations.

The positions $\vec{r}_0$ and $\vec{r}_1$ of the satellite at times t_0 and t_1 can be expressed as:

$$\vec{r}_0 = \vec{r}_{\text{obs},0} + k_0\vec{s}_0$$

$$\vec{r}_1 = \vec{r}_{\text{obs},1} + k_1\vec{s}_1$$

An initial guess is made for k_0. Then, the value of k_1 is based on the requirement that the new vector $\vec{r}_1$ should have the same magnitude as $\vec{r}_0$. This can be used to derive a circular orbit, simplifying the problem and ensuring consistency in the orbit determination process.

Given:

- $r_{0,\text{mag}}$: The magnitude of the original vector $\vec{r}_0$.
- $\vec{r}_{\text{obs},1}$: The sensor position vector at the second observation.

- $\vec{s}_1$: The LOS vector at the second observation.

The equation that needs to be satisfied is:

$$\|\vec{r}_{\text{obs},1} + k_1\vec{s}_1\| = r_{0,\text{mag}}$$

Expanding this equation:

$$\|\vec{r}_{\text{obs},1} + k_1\vec{s}_1\|^2 = r_{0,\text{mag}}^2$$

This leads to a quadratic equation in terms of k_1:

$$ak_1^2 + bk_1 + c = 0$$

where:

$$a = \|\vec{s}_1\|^2$$

$$b = 2(\vec{r}_{\text{obs},1} \cdot \vec{s}_1)$$

$$c = \|\vec{r}_{\text{obs},1}\|^2 - r_{0,\text{mag}}^2$$

Solving this quadratic equation gives the possible values for k_1:

$$k_1 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

To determine the transfer trajectory between $\vec{r}_0$ and $\vec{r}_1$, the algorithm uses the Izzo algorithm to solve the Lambert problem, which finds the velocity vectors $\vec{v}_0$ and $\vec{v}_1$ given the positions and the time of flight [11].

The variables formulation of the Lambert problem provides the orbital state vector based on $\vec{r}_0$, $\vec{r}_1$, and Δt:

$$\vec{v}_0, \vec{v}_1 = \text{lambert}(\vec{r}_0, \vec{r}_1, \Delta t)$$

The resulting orbital state vector $\vec{r}_0$, $\vec{v}_0$ is two-body propagated forward by 120 seconds to compute $\vec{r}_3$. The residual r_{err} is used to derive a circular orbit given by:

$$\text{r}_{\text{err}} = |\|\vec{r}_0\| - \|\vec{r}_3\||$$

The algorithm uses the Nelder-Mead optimization method to optimize the value of k_0 with an initial guess of 100 km. This widely used numerical method finds the minimum of an objective function in a multidimensional space. It is particularly effective for non-linear optimization problems where the gradient of the objective function may not be available [15]. In the context of the algorithm, the objective function to be minimized is the residual r_{err}, which constrains the prediction to a circular orbit.

3. METHODOLOGY

3.1 Data Generation and Simulation

For this study, we generated simulated short-arc angles-only observations of satellites from Earth observatories using two-line element set (TLE) catalogs and the simplified perturbation model (SGP4) [10]. Angles-only measurements or LOS vectors are generated by randomly selecting an observatory and a random satellite in its field of regard at random times no later than seven days from the TLE epoch. Five LOS vectors in sequence are generated, separated by a random frame rate to simulate common collection modes for the Raven-class telescopes [13]. To simplify the evaluation of the models, we do not simulate light transit time or aberration of light.

The LOS vector $\vec{s}$ is calculated by propagating the satellite and the observatory to time t to determine their Earth Centered Inertial (ECI) positions $\vec{r}_{\text{sat}}$ and $\vec{r}_{\text{obs}}$ respectively, then computing the vector difference:

$$\vec{s} = \vec{r}_{\text{sat}} - \vec{r}_{\text{obs}}$$

$$\vec{s} = \frac{\vec{s}}{||\vec{s}||}$$

To inject measurement noise into $\vec{s}$, we first construct an orthogonal basis with $\vec{s}$ by finding an orthogonal vector $\vec{x}$ and constructing the third orthogonal vector $\vec{y}$:

$$\vec{y} = \vec{s} \times \vec{x}$$
$$\vec{y} = \frac{\vec{y}}{||\vec{y}||}$$

These orthogonal vectors $\vec{x}$ and $\vec{y}$ are combined with $\vec{s}$ into a rotation matrix R_0:

$$R_0 = \begin{bmatrix} \vec{x} & \vec{y} & \vec{s} \end{bmatrix}$$

Noise is applied with random rotations around the z and x axis. First, we rotate around the z-axis by a uniform random angle θ_z:

$$R_z(\theta_z) = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$R_1 = R_0 \cdot R_z(\theta_z)$$

Then, we rotate around the x-axis by a normally distributed random angle θ_x:

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix}$$
$$R_2 = R_1 \cdot R_x(\theta_x)$$

The resulting noisy vector $\vec{S}$ is obtained by applying the final rotation to the unit vector $\vec{u} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$:

$$\vec{S} = R_2 \cdot \vec{u}$$

3.2 Two-Body Propagator for Loss Function and Evaluation

The two-body problem describes the motion of a satellite under the gravitational influence of a central body (e.g., Earth). The state vector $\vec{y}$ includes the position vector $\vec{r}$ and the velocity vector $\vec{v}$:

$$\vec{y} = \begin{pmatrix} \vec{r} \\ \vec{v} \end{pmatrix}$$

The derivative of the state vector, $\frac{d\vec{y}}{dt}$, is given by:

$$\frac{d}{dt}\begin{pmatrix} \vec{r} \\ \vec{v} \end{pmatrix} = \begin{pmatrix} \vec{v} \\ -\frac{\mu}{|\vec{r}|^3}\vec{r} \end{pmatrix}$$

where μ is the standard gravitational parameter for Earth [16].

To propagate the state vector over time, we use the differentiable Dormand-Prince solver from the TensorFlow Probability module [4], an adaptive step-size numerical integrator. It implements a 5th order Runge-Kutta method with adaptive step size control and dense output [5].

The propagation of the state vector over time intervals Δt can be expressed as:

$$\vec{y}(t_0 + \Delta t) = \text{Dormand-Prince}(\vec{y}_0, \Delta t)$$

This describes the numerical integration of the two-body differential equation using the Dormand-Prince method to update the position and velocity vectors over the specified time intervals. We use this method to evaluate the IOD predictions and explore its use as a physics-based loss function for training machine learning models.

3.3 Machine Learning Framework

The proposed machine learning framework leverages a neural network to learn the representation [7, 1] of measurements to orbital parameters. This section outlines the input features, model architecture, training process, loss functions, and the exploration of different output modes.

The input features to the neural network are derived from a sequence of observational data and include:

- **Time (t)**: The time at which the observation is made in seconds relative to the initial observation t_0.
- **Observer's Position ($\vec{r}_{\mathbf{obs}}$)**: The ECI position of the observer at time t.
- **Line of Sight Vector ($\vec{s}$)**: The ECI unit vector from the observer to the satellite at time t.

The neural network model is designed to predict the satellite's orbital elements at the initial observation time t_0. The architecture includes:

- **Input Layer**: Takes in the concatenated input features for each observation.
- **Hidden Layers**: A series of hidden layers with non-linear activation functions. We explore different architectures for the hidden layers, including fully connected (dense) layers and Long Short-Term Memory (LSTM) layers [9].
- **Output Layer**: Outputs the predicted orbital elements at time t_0.

We explore different output modes for the neural network:

- **Position and Velocity**: Outputting the ECI position and ECI velocity vectors of the satellite, providing a direct representation of its state.
- **Slant Range and Velocity**: Outputting the range to the satellite and its ECI velocity components, constraining the predicted position vector to the ECI LOS vector.
- **Modified Equinoctial Elements**: Outputting the satellite's predicted orbital elements in modified equinoctial form [17].

The training process involves supervised learning, where the neural network learns to minimize the difference between the predicted and true orbital elements. The steps include:

- **Dataset Preparation**: The dataset is split into training and validation sets, ensuring that the same satellite does not appear in both sets to avoid data leakage.
- **Dataset Normalization**: Input features are normalized to improve training stability and performance. Specifically:
  - **Position Normalization**: Each component of both the observer's position vector $\vec{r}_{\text{obs}}$ and the satellite's position vector $\vec{r}$ is scaled by the standard deviation σ_{pos} of all position components:

$$\vec{r}_{\text{obs, norm}} = \frac{\vec{r}_{\text{obs}}}{\sigma_{\text{pos}}}$$

$$\vec{r}_{\text{sat, norm}} = \frac{\vec{r}}{\sigma_{\text{pos}}}$$

  - **Velocity Normalization**: Each component of the velocity vector $\vec{v}$ is scaled by the standard deviation σ_{vel} of all velocity components:

$$\vec{v}_{\text{norm}} = \frac{\vec{v}}{\sigma_{\text{vel}}}$$

  - **Time Normalization**: The observation time t is scaled by the standard deviation σ_t of all time values:

$$t_{\text{norm}} = \frac{t}{\sigma_t}$$

- **Batch Training**: The model is trained in mini-batches for efficiency and stability.
- **Optimization**: The Adam optimizer is used with scheduled learning rate decay [14].
- **Early Stopping**: Training is monitored to avoid overfitting, with early stopping implemented based on validation performance.

We explore Euclidean-based loss and physics-based loss. The loss functions are described below.

The Euclidean loss measures the distance between the true and predicted vectors. For position vectors $\vec{r}$ and velocity vectors $\vec{v}$, the loss functions are defined as:

$$\mathscr{L}_{\text{position}} = \sqrt{\sum_{i=1}^{3} (r_{\text{true},i} - r_{\text{pred},i})^2}$$
$$\mathscr{L}_{\text{velocity}} = \sqrt{\sum_{i=1}^{3} (v_{\text{true},i} - v_{\text{pred},i})^2}$$

The combined loss function is the sum of the position and velocity losses:

$$\mathscr{L}_{\text{combined}} = \mathscr{L}_{\text{position}} + \mathscr{L}_{\text{velocity}}$$

The physics-based loss function incorporates the dynamics of satellite motion by evaluating the error both at the initial time t_0 and after propagating the state vectors forward to a later time t_1. This approach ensures that the predicted states adhere to physical laws of motion.

The true and predicted state vectors $\vec{y}_{\text{true},0}$ and $\vec{y}_{\text{pred},0}$ are propagated forward using the Dormand-Prince solver to obtain $\vec{y}_{\text{true},1}$ and $\vec{y}_{\text{pred},1}$, respectively.

The combined loss at both t_0 and t_1 is calculated as:

$$\mathscr{L}_0 = \mathscr{L}_{\text{combined}}(\vec{y}_{\text{true},0}, \vec{y}_{\text{pred},0})$$
$$\mathscr{L}_1 = \mathscr{L}_{\text{combined}}(\vec{y}_{\text{true},1}, \vec{y}_{\text{pred},1})$$

The final physics-based loss is the average of these two losses:

$$\mathscr{L}_{\text{propagated}} = \frac{\mathscr{L}_0 + \mathscr{L}_1}{2}$$

This loss function ensures that the predictions fit the observed data and respect the physical constraints of orbital dynamics.

3.4 Evaluation Metrics

The performance of IOD models is primarily evaluated using the included angle between the predicted and true orbital elements from the observer's initial position $\vec{r}_{\text{obs}}$. This metric assesses the accuracy of the orbital elements from the observer's perspective, which is crucial for a follow-up observation. The evaluation metrics are defined as follows:

To evaluate the propagation accuracy from the perspective of the observer, we propagate the state vectors forward by a time interval Δt using the Dormand-Prince method:

$$\begin{aligned}\vec{y}_{\text{true},\Delta t} &= \text{Dormand-Prince}(\vec{y}_{\text{true}}, \Delta t)\\ \vec{y}_{\text{pred},\Delta t} &= \text{Dormand-Prince}(\vec{y}_{\text{pred}}, \Delta t)\end{aligned}$$

To calculate the included angle between the true and predicted position vectors, we define the position vectors relative to the observer's position $\vec{r}_{\text{obs}}$ at t_0:

$$\begin{aligned}\vec{r}_{\text{true, rel}} &= \vec{r}_{\text{true}} - \vec{r}_{\text{obs}}\\ \vec{r}_{\text{pred, rel}} &= \vec{r}_{\text{pred}} - \vec{r}_{\text{obs}}\end{aligned}$$

The position vectors after propagation relative to the observer are:

$$\begin{aligned}\vec{r}_{\text{true, rel},\Delta t} &= \vec{r}_{\text{true},\Delta t} - \vec{r}_{\text{obs}}\\ \vec{r}_{\text{pred, rel},\Delta t} &= \vec{r}_{\text{pred},\Delta t} - \vec{r}_{\text{obs}}\end{aligned}$$

The included angles between the true and predicted position vectors at the initial and propagated times are calculated as follows. The dot product gives the cosine of the angle, and the norm of the cross product gives the sine:

$$\begin{aligned}\cos(\alpha) &= \frac{\vec{r}_{\text{true}} \cdot \vec{r}_{\text{pred}}}{|\vec{r}_{\text{true}}||\vec{r}_{\text{pred}}|}\\ \sin(\alpha) &= \frac{|\vec{r}_{\text{true}} \times \vec{r}_{\text{pred}}|}{|\vec{r}_{\text{true}}||\vec{r}_{\text{pred}}|}\end{aligned}$$

The included angle α can be computed using the arc-tangent function:

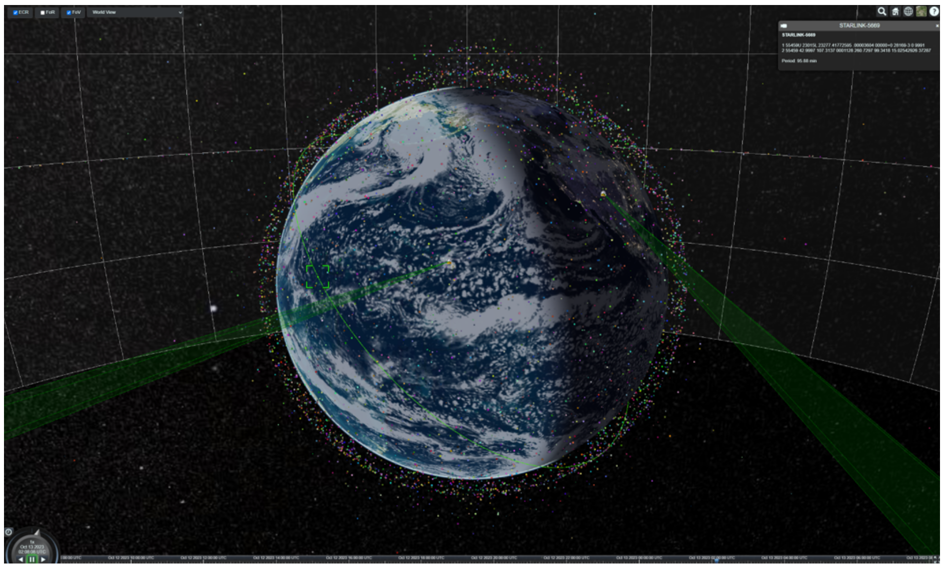
$$\alpha = \arctan 2(\sin(\alpha), \cos(\alpha))$$

To ensure a successful follow-up observation, α must be smaller than the sensor's field of view; otherwise, a search pattern is required, complicating operations and reducing available sensor time.

4. EXPERIMENTAL SETUP

4.1 Simulation Environment

The simulation environment for our study was implemented using the open-source SatSimJS library [3]. SatSimJS can simulate EO observations of satellites and has a visualization component, as shown in Figure 2 based on CesiumJS [2].



STARLINK-5669

```
1 55459U 23015L 23277.41772565 .00003604 00000+0 28169-3 0 9991
2 55459 42.9997 107.3137 0001128 260.7297 99.3418 15.02542926 37287
```

Period: 95.88 min

Fig. 2: Simulation environment is based on the open-source SatSimJS library. Observation datasets are generated from specified TLE catalog and user specified observatories.

Below is a description of the simulation setup and process:

1. **Loading Observatories and Satellites:**
   - Load observatories from an input file with parameters: name and location (latitude, longitude, altitude).
   - Load satellites from an input file containing TLE data and categorize them into orbital regimes. The criteria for each orbital regime are as follows:
     - **GEO**: Satellites with a mean motion close to 1 revolution per day (0.95 to 1.05) and low eccentricity (less than 0.02).
     - **MEO**: Satellites with orbital periods between 128 and 1000 minutes and eccentricity less than 0.25.
     - **LEO**: Satellites with a mean motion greater than 11.25 revolutions per day and eccentricity below 0.25.
     - **HEO**: Satellites with an eccentricity greater than 0.25.
2. **Simulation Execution:**
   - Run the simulation in a loop from start time to end time, with each iteration representing a time step defined by the delta step parameter.
   - Update the universe to the current simulation time at each iteration.
   - Select a random observatory to track a random satellite in its field of regard. The next satellite picked is based on the orbital regime to balance the dataset between GEO, MEO, LEO, and HEO.
   - For each satellite, collect five sequential observational data separated by a random frame rate. Data collected includes satellite name, time, observer ECI position, ECI line-of-sight vector, satellite ECI position and velocity, classical orbital elements [16], and modified equinoctial orbital elements [17].

4.2 Training and Testing Data Description

Table 1 summarizes the parameters for the training dataset. The TLE catalogs were sourced from CelesTrak, a web-based service that provides orbital data and analytical tools for the space community [12].

Source	Start (Date Pulled)	Sim End Time	Obs	Sites	LOS Err	Frame Time (s)
CelesTrak	2023-10-04T00:00	2023-10-10T00:00	3,000,000	3	None	$X \sim \mathscr{U}(1.0, 5.0)$
CelesTrak	2023-10-26T00:00	2023-11-01T00:00	3,000,000	3	None	$X \sim \mathscr{U}(1.0, 5.0)$
CelesTrak	2024-01-24T00:00	2024-01-30T00:00	3,000,000	3	None	$X \sim \mathscr{U}(1.0, 5.0)$
CelesTrak	2024-04-30T00:00	2024-05-06T00:00	3,000,000	3	None	$X \sim \mathscr{U}(1.0, 5.0)$
CelesTrak	2024-06-03T00:00	2024-06-09T00:00	3,000,000	3	None	$X \sim \mathscr{U}(1.0, 5.0)$

Table 1: Data Sources for Training and Validation Datasets. These data sources were combined to create a training dataset of 15,000,000 observations or 5,000,000 sequential collections.

The training data is split into training and validation sets based on satellite name to ensure unique satellites in each set. This facilitates an unbiased evaluation of the model's performance on unseen data. This is achieved by dividing the unique satellite names into training (95%) and validation (5%) sets.

The testing dataset is generated using the parameters outlined below in Table 2.

Source	Start (Date Pulled)	Sim End Time	Obs	Sites	LOS Err	Frame Time (s)
CelesTrak	2024-08-01T00:00	2024-08-08T00:00	500,000	3	5 arcsec	$X \in \{1.0, 1.5, \ldots, 5.0\}$

Table 2: Data Sources for Testing Datasets. The TLE catalog used for testing is at least 60 days newer than the ones used for the training dataset. Random normal noise with a standard deviation of 5 arc seconds is added to simulate observational errors, providing a basis for evaluating all IOD methods. The test dataset contained a total of 100,000 sequential collections.

5. RESULTS AND DISCUSSION

5.1 Model Performance and Comparison with Traditional Methods

We experimented with multiple hidden layer configurations to explore various neural network architectures. We tested combinations of fully connected (dense) and Long Short-Term Memory (LSTM) layers to capture temporal and spatial dependencies in the observational data. Our trials included architectures with up to six hidden layers, varying the number of units in each layer to find the optimal configuration. We experimented with shallow networks, using only dense layers and deeper networks incorporating LSTM layers, to determine the best balance between model complexity and performance.

After experimentation, we found that a network architecture comprising two LSTM layers followed by a dense layer yielded effective performance. Specifically, the architecture was as follows:

- **Input Layer**: 5x7 matrix representing concatenated input features from each observation
- **First Hidden Layer**: LSTM layer with 320 units
- **Second Hidden Layer**: LSTM layer with 320 units
- **Third Hidden Layer**: Dense layer with 320 units
- **Output Layer**: Outputs 6 numbers representing the predicted position and velocity vectors

In addition to exploring various hidden layer configurations, we also investigated different output modes for the neural network. Specifically, we tested models that output modified equinoctial orbital elements, slant range/velocity components (SV), and position/velocity components (RV). Our findings indicated that the models could not effectively learn modified equinoctial orbital elements. However, both SV and RV outputs showed the capacity to learn when trained with Euclidean loss and physics-based loss functions.

We processed the test dataset with the trained neural network. For comparison, we also processed the test dataset with the Gauss, Gooding, and Laplace methods using the first, middle, and last observations out of the five available. The MACHINA method utilized only the first and last observations. Gooding requires an initial range guess for the first and last measurements, for which we used the arbitrary guess of two times the sum of the observer positions, as suggested in the algorithm [8].

Tables 3, 4, 5, 6, and 7 summarize the performance metrics across various orbital regimes (GEO, MEO, LEO, HEO). These tables present the included angle between the two-body propagated positions of the true and predicted orbits from the observer after 5 minutes. The detailed description of this metric can be found in the Evaluation Metric section. Each model's performance is evaluated using three statistical measures: the first quartile (Q1), the median (Q2), and the third quartile (Q3) of their error distributions. Additionally, models are ranked based on their Q3 performance. Figures 3, 4, 5, 6, and 7 illustrate the distribution of angle errors for each model, highlighting performance differences across all samples.

Model	**Q1**	**Median (Q2)**	**Q3**	**Rank**
machina	0.000 799	0.004 567	0.058 257	1
learned (rv, physics)	0.045 739	0.181 420	0.547 207	2
learned (rv, euclidean)	0.071 523	0.325 306	0.870 854	3
learned (sv, euclidean)	0.016 073	0.153 507	0.890 841	4
gooding	0.009 783	0.696 497	19.842 115	5
gauss	0.075 602	0.342 840	179.351 078	6
laplace	0.078 146	0.342 934	179.351 149	7

Table 3: Overall performance metrics for different models. Included angle error in degrees after 5 minutes.

Model	**Q1**	**Median (Q2)**	**Q3**	**Rank**
machina	0.000 385	0.000 695	0.001 221	1
gooding	0.001 862	0.003 357	0.006 167	2
learned (sv, euclidean)	0.001 133	0.002 136	0.008 994	3
learned (rv, physics)	0.012 104	0.020 711	0.036 554	4
learned (rv, euclidean)	0.012 804	0.023 343	0.048 938	5
gauss	0.223 191	2.804 440	179.568 528	6
laplace	0.223 192	2.804 468	179.568 528	7

Table 4: Performance metrics for GEO orbital regime. Included angle error in degrees after 5 minutes.

Model	**Q1**	**Median (Q2)**	**Q3**	**Rank**
machina	0.000 496	0.000 930	0.001 671	1
learned (sv, euclidean)	0.034 978	0.075 256	0.164 281	2
learned (rv, physics)	0.078 219	0.141 061	0.239 033	3
learned (rv, euclidean)	0.151 083	0.265 737	0.448 652	4
gooding	0.789 348	1.059 728	1.445 280	5
gauss	0.120 187	0.562 585	179.315 648	6
laplace	0.120 188	0.562 586	179.315 650	7

Table 5: Performance metrics for MEO orbital regime. Included angle error in degrees after 5 minutes.

Model	Q1	Median (Q2)	Q3	Rank
gauss	0.015 138	0.037 917	0.119 823	1
machina	0.040 313	0.068 614	0.121 968	2
laplace	0.022 152	0.045 638	0.122 667	3
learned (rv, physics)	0.306 464	0.525 788	0.896 561	4
learned (rv, euclidean)	0.456 110	0.793 672	1.372 615	5
learned (sv, euclidean)	0.612 527	1.193 540	2.257 056	6
gooding	40.507 565	62.112 801	91.875 814	7

Table 6: Performance metrics for LEO orbital regime. Included angle error in degrees after 5 minutes.

Model	Q1	Median (Q2)	Q3	Rank
machina	0.021 589	0.048 561	0.072 504	1
gooding	0.015 884	0.060 793	0.127 504	2
learned (rv, physics)	0.144 544	0.453 800	1.102 110	3
learned (sv, euclidean)	0.151 433	0.577 001	1.103 708	4
learned (rv, euclidean)	0.279 987	0.780 580	1.542 498	5
gauss	0.234 966	2.515 207	179.512 860	6
laplace	0.234 966	2.515 623	179.512 974	7

Table 7: Performance metrics for HEO orbital regime. Included angle error in degrees after 5 minutes.

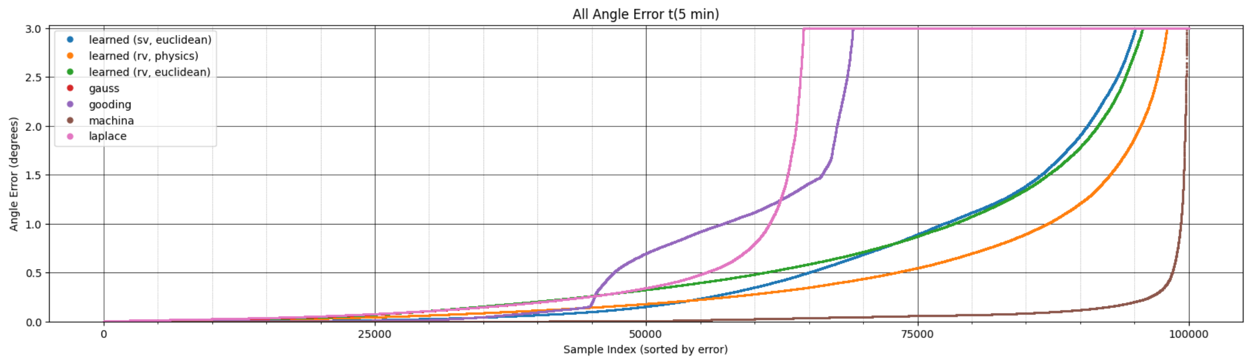


Fig. 3: Scatter plot of angle error after 5 minutes for all observations. The plot shows the included angle error in degrees, sorted by error for each sample index.

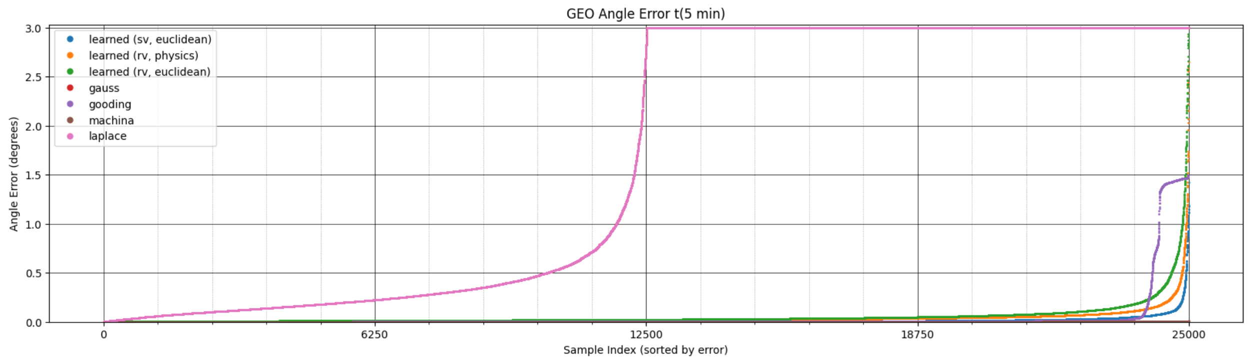


Fig. 4: Scatter plot of angle error after 5 minutes for GEO observations. The plot shows the included angle error in degrees, sorted by error for each sample index.

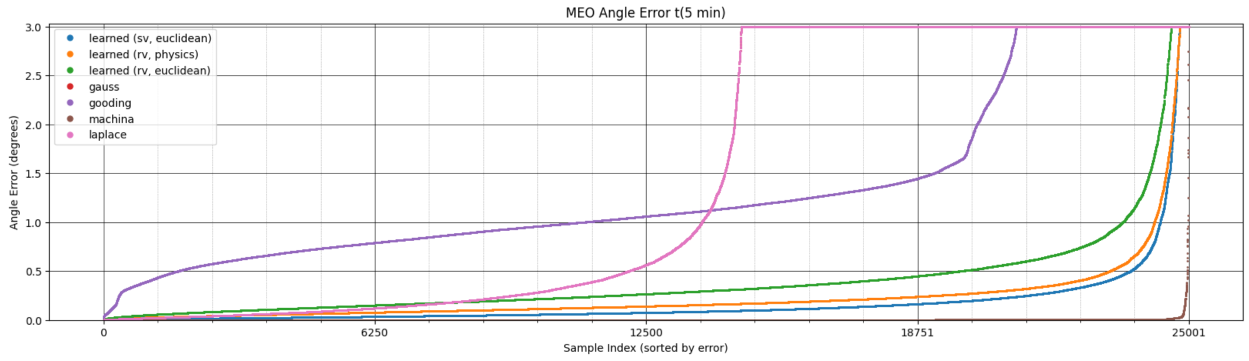


Fig. 5: Scatter plot of angle error after 5 minutes for MEO observations. The plot shows the included angle error in degrees, sorted by error for each sample index.

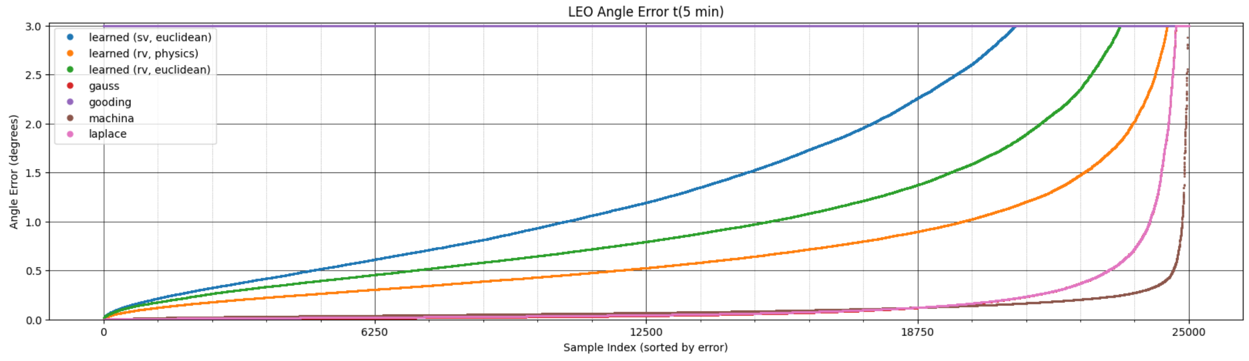


Fig. 6: Scatter plot of angle error after 5 minutes for LEO observations. The plot shows the included angle error in degrees, sorted by error for each sample index.

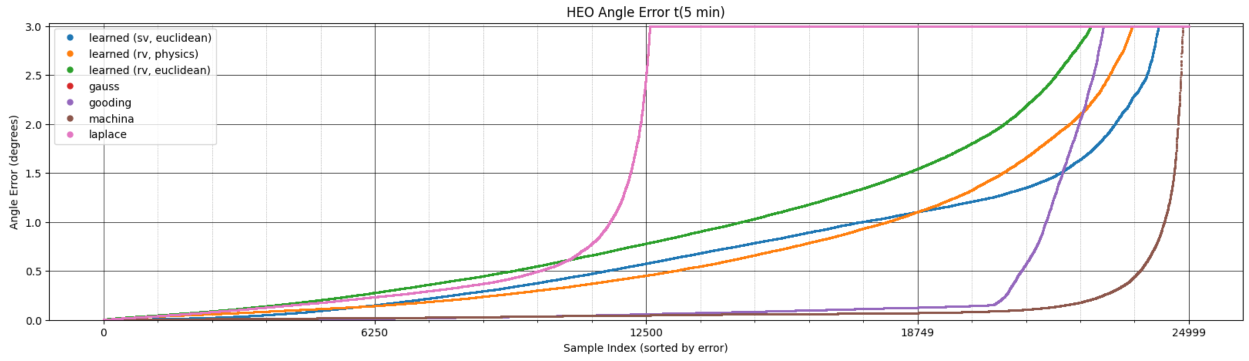


Fig. 7: Scatter plot of angle error after 5 minutes for HEO observations. The plot shows the included angle error in degrees, sorted by error for each sample index.

5.2 Discussion

Our experiments reveal key insights into using neural networks for IOD. A major insight is the demonstrated capability and feasibility of neural networks to learn and model this domain effectively.

When evaluated against the Gauss and Laplace methods, our neural network models showed significant improvements for observations with the shortest angular separation. The Gooding method, while effective, requires an accurate initial guess of the slant range. Using an arbitrary guess of two times the sum of the observer positions [8], as suggested in Gooding's algorithm, performs well for GEO but poorly for LEO.

The MACHINA method, however, outperformed all models evaluated in this study. This suggests that while neural network approaches hold promise, the MACHINA method remains the most effective solution for short-arc angles-only IOD tasks.

6. CONCLUSION

In this study, we investigated the application of neural networks for Initial Orbit Determination (IOD) using short-arc angles-only observational data. Through experimentation with various neural network architectures and output modes, we identified a configuration comprising two Long Short-Term Memory (LSTM) layers followed by a dense layer. We found that models trained to output Earth-centered inertial (ECI) orbital state vectors could learn effectively, with better results, when using a physics-based loss function for predicting position and velocity vectors. Importantly, our findings reveal that learned methods are not as constrained by data quality as traditional methods are. This highlights the potential of neural network-based approaches to achieve more robust and accurate orbit determinations.

7. ACKNOWLEDGEMENTS

The views expressed are those of the authors and do not reflect the official policy or position of the US Space Force, Department of Defense, or the US Government.

REFERENCES

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006.

[2] CesiumJS Developers. Cesiumjs: An open-source javascript library for world-class 3d globes and maps, 2023.

[3] SatSimJS Developers. Satsimjs: An open-source library for simulating electro-optical observations of satellites, 2023.

[4] TensorFlow Probability Developers. Tensorflow probability: A library for probabilistic reasoning and statistical analysis, 2023.

[5] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6:19–26, 1980.

[6] Justin R Fletcher and Jonathan E Kadan. Autonomous interactive agents for global telescope network orchestration. *SPIE Software and Cyberinfrastructure for Astronomy VIII*, 13101:990–999, 2024.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[8] R. H. Gooding. A new procedure for the solution of the classical problem of minimal orbit determination from three lines of sight. *Celestial Mechanics and Dynamical Astronomy*, 66(3):387–423, 1997.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] Felix R. Hoots and Ronald L. Roehrich. Models for propagation of norad element sets. *Spacetrack Report No. 3*, 1980.

[11] Dario Izzo. Revisiting lambert's problem. *Celestial Mechanics and Dynamical Astronomy*, 121(1):1–15, 2015.

[12] T.S. Kelso. Celestrak: Current norad two-line element sets. http://www.celestrak.com, 2024. Accessed: 2024-08-07.

[13] Paul Kervin, Vicki Soo Hoo, Daron Nishimoto, and Dennis Liang. Rapidly deployable raven-class systems ssa support in the field. Technical report, Air Force Research Laboratory, 2009.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015.

[15] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.

[16] David A. Vallado. *Fundamentals of Astrodynamics and Applications*. Springer, 4th edition, 2013.

[17] M. J. H. Walker, B. Ireland, and J. Owens. A set of modified equinoctial orbital elements. *Celestial Mechanics*, 36:409–419, 1985.