Automated Collision Risk Assessment and Mitigation

N. Maric[(1)], E. Arias[(2)], P. Gago[(2)], L. Porcelli[(2)], D. Saez[(2)], D. Escobar[(2)], A. Stanculescu[(3)], A. Solomon[(3)], G. Muntean[(3)], Hélène Ma[(4)], F. Letizia[(4)], Benjamin Bastida[(4)], Klaus Merz[(4)]

[1]GMV UK; OX110QG, Didcot, Oxfordshire, United Kingdom,
[2]GMV ES; 28760 Tres Cantos, Madrid, Spain
[3]GMV RO; 014476 Bucharest, Romania
[4]ESA; 64293 Darmstadt, Germany
nina.maric@gmvnsl.com
earias@gmv.com
pau.gago.padreny@gmv.com
lorenzo.porcelli@gmv.com
dsaez@gmv.com
descobar@gmv.com
andrei.stanculescu@gmv.com
gmuntean@gmv.com
helene.ma@ext.esa.int
francesca.letizia@esa.int
benjamin.bastida.virgili@esa.int
klaus.merz@esa.int

ABSTRACT

In the modern world, the space safety is of growing concern for all of the spacefaring nations. This is mainly due to two major factors: accelerated increasing number of artificial space objects formed from break-up events as well as the rapidly growing mega-constellations and launch rates which significantly increase the artificial objects population around Earth. Within ESA's Space Safety Programme, a cornerstone has been created focusing on automated collision avoidance to reduce operational costs, late commanding paths and operations concepts as well as means for coordination of operators and catalogue providers.

GMV is currently upgrading the Autonomous Collision Avoidance System (***AUTOCA***) developed under ARTES contract, within the scope of ESA's CREAM activity S2P-S1-CR-01 "Automated avoidance manoeuvre decisions and design". This system aims at its use for large fleets (e.g. large operators in GEO and future mega-constellations in LEO and MEO) and also orbit raising scenarios with full-electric satellites (e.g. orbit transfer from LEO to Upper-LEO or from LEO/GTO to GEO). It is important to mention that within the scope of this activity, additional prototypes will be integrated in the system. Within this paper the focus is placed on three prototypes related to the measurements and catalogue data: Data Fusion, Orbit Propagation and Artificial Intelligence/Machine Learning (AI/ML) techniques for Covariance Matrix and State Vector predictions.

1 DATA FUSION

Nowadays, it is quite common for satellite operators to receive CDMs from several sources, due to the increment in the amount of SSA providers. As a consequence, the following problems may arise:

- Lack of consistency in the states vectors and covariance matrices from different sources due to biases in one or several of the SSA providers.
- Lack of covariance realism between different sources due to problems in the covariance computation of one or several of the SSA providers

Some of these issues are better resolved on the SSA provide side (e.g., biases), whereas the other could be solved on the operator side (e.g., issues related to covariance realism).

Once these issues are solved, the problem of finding a common solution based on the form information from different sources still remains. One way to tackle this problem is to generate a combined solution for each object independently, in particular, of the chaser, as the target is most likely an object for which operational ephemerides have been provided to the SSA providers. In order to reach this combined solution, it is necessary to first synchronise the states of the object coming from different sources at the same epoch. Then, a new combined orbit should be generated together with its covariance at the synchronised epoch. With this information it is possible to re-evaluate the TCA of the conjunction event and obtain the new states and covariance of both objects at the new TCA.

Since the data provided in CDMs is assumed to be

Gaussian, a common approach for data fusion in Gaussian cases, such as the Covariance Intersection (CI) Method (see [1]), is analysed. This method is very useful when the cross-correlation of the data is unknown, which is the case here (see [2]). It is based on obtaining a convex combination of n estimates $(\vec{x}_i, \boldsymbol{C_i})$

$$\vec{x} = \sum_{i=1}^{n} \tilde{\omega}_i \, C C_i^{-1} \vec{x}_i \qquad (1)$$

$$C^{-1} = \sum_{i=1}^{n} \tilde{\omega}_i \, C_i^{-1} \qquad (2)$$

where the non-negative weighting coefficients $\tilde{\omega}_i$ are added in order to take into account the cross-correlation between the estimates, which is assumed unknown. These weighting coefficients also verify that

$$\sum_{i=1}^{n} \tilde{\omega}_i = 1 \qquad (3)$$

These coefficients are usually chosen to minimize either the trace or the determinant of $\boldsymbol{C}$. Taking into account the conclusions achieved in [3], [4] and [5], the determinant minimization criterion is used hereafter.

In order to avoid the numerical implementation effort required to solve the previous highly non-linear optimization problem, the approach followed in this analysis is the one proposed in [5], based on an improvement of the Fast Covariance Method (fast approximate solution of the previous optimization problem) proposed in [6]. In [5], it is proved that the loss of accuracy of the approximate solution with respect to the optimal solution would be marginal for high degrees of cross-correlation. In addition, the improved version of the Fast Covariance Intersection Method proposed in [5] aims to solve the insensitivity of the original method with respect to the relative orientation of the individual covariance matrix estimates $\boldsymbol{C_i}$ when computing the weighting coefficients $\tilde{\omega}_i$, which leads to a significant performance improvement.

According to [5], the weighting coefficients $\tilde{\omega}_i$ can be computed based on the individual information matrix estimates $\boldsymbol{I_i}$ $(\boldsymbol{I_i} = \boldsymbol{C_i^{-1}})$

$$\tilde{\omega}_i = \frac{\det(\boldsymbol{K}) - \det(\boldsymbol{K} - \boldsymbol{I_i}) + \det(\boldsymbol{I_i})}{n \det(\boldsymbol{K}) + \sum_{j=1}^{n} \left[\det(\boldsymbol{I_j}) - \det(\boldsymbol{K} - \boldsymbol{I_j})\right]} \qquad (4)$$

with

$$\boldsymbol{K} = \sum_{j=1}^{n} \boldsymbol{I_j} \qquad (5)$$

The suitability of this approach will be analysed using several states and covariance matrices estimates $(\vec{x}_i, \boldsymbol{C_i})$ for the same conjunction event included in CDMs coming from different SSA providers (e.g., CSpOC, CARA, etc…).

2 ENHANCED STATE TRANSITION MATRIX (STM) PROPAGATION

If the effect of the manoeuvre were to be numerically integrated, the computational cost of the optimisation process would rise dramatically. Therefore, the effect of the manoeuvre on the orbit of the spacecraft is evaluated through an enhanced propagation that combines the well-known STM and an analytical Keplerian propagation. The motivation behind this approach is to reduce the computational time required by a full numerical propagation, while also obtaining a higher precision than a linear STM propagation of the orbit. This model is still accurate for longer-time propagations and larger manoeuvres, where the STM approach fails, at a fraction of the computational cost of the full numerical propagation.

The enhanced propagation process has two different steps. The first one, consisting of a Keplerian propagation of the Collision Avoidance Manoeuvre (CAM) effect, is illustrated in Figure 1. The different orbits of the satellite of interest are presented, each obtained through a different propagation.

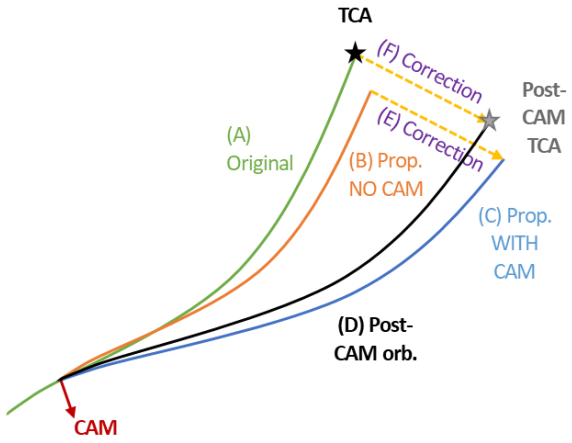

Figure 1- Graphic depiction of the orbit propagations.

The green curve labelled by *A* represents the original orbit obtained from the operator of the satellite. As the dynamical model may differ from the one used by the operator, if the state vector at the CAM epoch is selected and propagated with no CAM effect, a different orbit will be obtained, even if a high-fidelity dynamic model is used. In the enhanced propagation method, a Keplerian propagation is done analytically, which avoids numerical integration. This is represented by the orange curve labelled as *B*. Next, the state vector at the CAM epoch with the CAM effect is propagated with the same analytical Keplerian propagator, thus obtaining the blue curve labelled as *C*. The difference between *B* and *C* is

the effect of the CAM – the correction labelled by *E* – assuming Keplerian dynamics and including linear and non-linear effects. This correction could be applied to the original orbit *A*, thus resulting in a new orbit *D*. This new orbit (black curve labelled as *D*) employs the operator's dynamical model but also takes into account the CAM effect through the Keplerian dynamics. In reality, the real effect of the CAM which is the difference between a true orbit *D* (using the same dynamical model as the operator to propagate the state vector with the CAM effect) and orbit *A* will result in a correction *F*, which is slightly different from *E*. However, this error is small for typical CAM manoeuvre sizes and propagation times needed.

The second step in the simplified propagation consists of propagation through an STM, in which the STM is computed based on the original ephemeris (set fixed) and a high-fidelity dynamic model. This STM hence captures with great accuracy the linear effect of the CAM considering the most relevant orbital perturbations (geopotential, 3rd body, drag). The STM is computed only once for each epoch in the ephemeris and then reused in each CAM computation, which leads to a small computational cost.

The Keplerian propagation of the CAM effect captures linear and non-linear effects of the central gravity, while the STM propagation captures all linear effects from central gravity and orbital perturbations. In order to retain an appropriate balance between computational cost and accuracy, the CAM effect can be computed as follows combining both approaches:

$$\begin{aligned} &\boldsymbol{x_D}(t) \\ &= \boldsymbol{x_A}(t) + \Phi(t, t_M) \cdot \boldsymbol{u} \\ &+ R_{TNW}^T R_{TNW,K}\left[\boldsymbol{x}_{B,K}(t, t_M, \boldsymbol{u}) - \boldsymbol{x}_{A,K}(t, t_M)\right. \\ &\left. - \Phi_{K,\boldsymbol{xv}}(t, t_M) \cdot \boldsymbol{u}\right] \end{aligned} \quad (6)$$

which adds to the original orbit $\boldsymbol{x_A}$, in the TNW frame, a correction is constructed as:

- $\left[\boldsymbol{x}_{B,K} - \boldsymbol{x}_{A,K}\right](t, t_M, \boldsymbol{u})$: the difference between a keplerian propagation of the initial state of orbit with ($\boldsymbol{x}_{B,K}$) and without ($\boldsymbol{x}_{A,K}$) manoeuvre. That is two independent propagations. As it is done analytically, it does not introduce a significant computational cost. This correction contains all linear and non-linear effects of the CAM propagation accounting for the central gravity.
- $[\Phi(t, t_M) - \Phi_K(t, t_M)]_{\boldsymbol{xv}} \cdot \boldsymbol{u}$: the effect of the manoeuvre is propagated through an STM, in which to the effect of the full perturbations linearized STM $\Phi(t, t_M)$, the effect of an STM considering only Keplerian motion $\Phi_K(t, t_M)$ is subtracted because it was already accounted for by the keplerian propagation. This correction contains all linear effects of the CAM propagation accounting for the orbital perturbations considered in the high-fidelity dynamic model.

To account for any misalignment between the Keplerian correction (propagations with and without manoeuvre and Keplerian STM) and the nominal no-CAM trajectory in Earth-centred frame, these corrections are applied in the local TNW frame of the original no-CAM trajectory, accounted for by the terms $R_{TNW}^T R_{TNW,K}$.

The results in Figure 2 illustrate the error commited against a full numerical propagation. If one considers an in-track burn of 0.1 m/s, and propagates the effect of such a manoeuvre for up to 4 days, it is possible to compare the performance of the enhanced STM propagation with respect to other algorithms:

- DM0: propagation through classical STM (which includes Keplerian and perturbations).
- DM1: enhanced propagation, without applying the corrections in the local TNW frame.
- DM2: enhanced propagation.
- DM3: considering only full Keplerian dynamics, without perturbations.

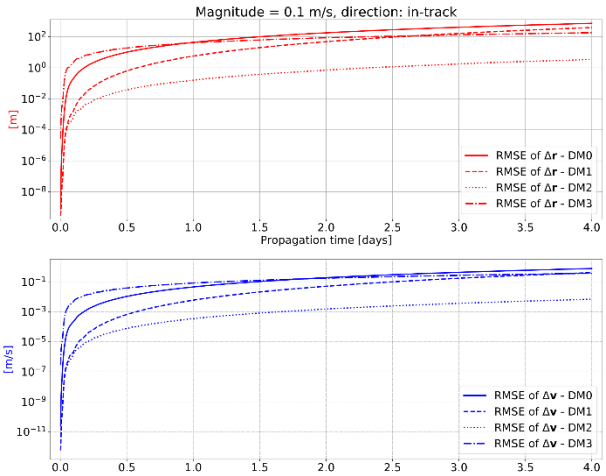

Figure 2- Test for propagation accuracy of the dynamical models (RMSE, in-track manoeuvre)

As it can be observed, DM2 outperforms all the other models, setting to a maximum error in the order of meters in position and 10^{-2} millimetres per second in velocity. All the other models (DM0, DM1 DM2 and DM3) reach errors over 100 m in position and 0.1 m/s in velocity.

3 AI/ML TECHNIQUES

Having a reliable prediction of the criticality of an event allows the human operator to make decisions in a more informed manner and can help with navigating operational constraints and lead to less costly manoeuvres. There are multiple approaches that can be employed in order to predict the criticality. An important

aspect related to the available data is that conjunction events are described by updates (in the form of Conjunction Data Messages – CDMs) in a sequential manner. In such a setting, models specifically tailored for sequential data can be employed.

3.1 Data Preparation

In order to train any machine learning model, a large dataset is needed. In this context, the dataset is represented by historical CDMs based on which the machine learning models will infer the general patterns, from the data, during the model training process in order to be able to accurately predict unseen data.

Once the input dataset is parsed, it will be stored within the internal database of the software and when the machine learning models are being trained, the data will be fetched and pre-processed in an attempt to offer relevant information to the models that will be trained.

The raw dataset consists of 219585 events and 2586718 JSPOC CDMs that are between two time intervals such as December 2014-November 2019 and November 2021-June 2022. The pre-processed dataset consists of a total of 151844 events and 910882 CDMs.

3.2 Data pre-processing

One of the most relevant steps of the machine learning training is represented by the data pre-processing step.

In this step, large datasets are being retrieved from the internal dataset, additional fields are computed and the dataset is filtered in order to remove duplicate data and outliers. Lastly, the model features are being scaled.

3.2.1 Computation of additional fields and dataset filtering

After assuring unified datatypes for each set of data and initial cleaning of missing values or misspelled entries, fields of the dataset are renamed and normalized to allow easier data handling. Additional useful fields are calculated, e.g. time to TCA for each CDM (based on CDM creation date) or time since last observation used for the Orbital Determination (OD) process. State vectors and covariance matrix elements for primary and secondary objects, as well as relative state vector may be registered in different references frames, thus all the relevant values are transformed to common inertial system EME2000, allowing proper and consistent computations. For a portion of CDMs the information of the probability of collision is missing, thus for consistency it is recalculated for all CDMs according to [7].

In order to identify the actual updates of orbital information of the secondary object, a set of filters are defined to discard the updates in which the secondary object information has not been updated, based on the information provided in the CDM. Therefore, events with a single update, covariance outliers, updates that have last observations start and end equal to the previous, updates with observation used or available equal to the previous CDM and updates with state vector distances within 10^{-6} compared to the previous CDMs are discarded.

In order to train the Machine Learning model, the dataset is divided in the following way: 80% for training, 10% for testing and 10% for validation.

3.2.2 Scaling the data

The range of feature values varies widely, thus a transformation is applied aiming proper scaling of data used as input for machine learning algorithm.

Quantile Transformer is used in order to transform features according to quantile information [8]. Its default behaviour is to map the original values to a uniform distribution; however, it is possible to obtain a normal distribution as well. The mapping is done by using the cumulative distribution function F of the feature and the quantile function of the desired distribution G as such:

$$G^{-1}(F(X)) \tag{7}$$

The formula is based on the fact that the cumulative distribution function of a random variable X is uniformly distributed on $[0, 1]$, and that $G^{-1}(U)$ has distribution G, where U is a random variable uniformly distributed in the range $[0, 1]$ as well. According to the documentation, the quantile transform, by performing a rank transform, smooths out "unusual" distributions and is more robust to outliers than regular scaling methods. Correlations and distances within/across features are, however, affected (distorted).

3.3 Machine Learning for Covariance prediction

Prediction of the covariance elements of the secondary satellite by means of machine learning techniques represents one of the main goals of the study. These fields are further used to in order to compute the Probability of Collision (PoC), on which the CAM decisions are based.

The covariance prediction as part of the conjunction assessment analysis is a key feature that allows satellites' operators to anticipate the expected collision risk, avoiding the planning of CAM strategies that are not really needed or with a lower cost in terms of delta V when they are really needed.

The three main diagonal positional elements (Along-Track, Cross-Track and Radial) of the covariance matrix in RTN reference frame, from the secondary object, are selected for the prediction. Within the system, it is possible to train a machine learning model in order to predict data at four different time steps, such as:

- 24 hours after the last available update.
- 48 hours after the last available update.
- 72 hours before the time of closest approach.
- At the time of closest approach.

The machine learning models are composed of a Long Short Term Memory (LSTM), an Attention Mechanism and a fully connected layer.

The machine learning models take as input numerous features from all CDMs within an event in order to perform a prediction.

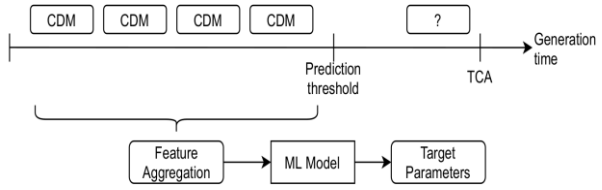

Figure 3 – Event Aggregation Setup

Long Short Term Memory

Recurrent Neural Networks (RNN) such as LSTM allow both training and making inferences using sequences of arbitrary length. In the case of conjunction events, the input can be represented by the sequence of available CDMs. The basic unit of training is represented by an event (a sequence of data from the corresponding CDMs). Therefore, even though the events are analysed individually, the model learns to generalise, as it is exposed to a training set containing multiple events.

Attention Mechanism

In order to increase the performance given by the LSTM model, an attention mechanism was proposed. The core idea behind the attention mechanism is that it allows the LSTM model to selectively focus on valuable parts of the evolution of an event and, therefore, learn the association between them.

Given the implied task of forecasting values based on several timeframes, an attention mechanism was proposed for testing to assess its performance impact on the several types of predictions. The mechanism has been proposed due its ability to solve the biggest problem in sequence to sequence tasks, which consists in decoding a variable length sequence based on a single context vector. It enables the utilisation of all hidden time steps of the input sequence during the decoding process.

There are two types of attention mechanisms proposed to be tested in the current section. The first one is proposed in [9] which is formally called additive attention, the second being an attention mechanism proposed in [10]. The main difference between them is how they score similarities between the current decoder input and encoder outputs.

Within the current study it was opted to make use of the Bahdanau attention mechanism.

3.4 Machine Learning for State Vector Prediction

The approach through which the machine learning model is developed is similar to the one proposed in [11] in the sense that it does not predict directly the State Vector (SV), but rather the error between an already existing propagation at TCA and the true SV, at the same considered time. The error predicted by the model is then added to the initial propagation, therefore obtaining the improved SV prediction.

In contrast to the approach used within the covariance prediction models, the SV prediction model takes as input features data from (and related to) only one individual CDM at a time in order to make a prediction. This approach is advantageous because the model can be operated from the early stages of an event, only needing data from individual CDMs to obtain the desired SV prediction. This means that it is not required to wait for a certain amount of CDMs to be received or for a certain time threshold (2 days before TCA, for example) to pass, but rather the model can operate with minimal data, at any stage in the event.

The model architecture used for the SV prediction is a Deep Neural Network (DNN), consisting of multiple hidden layers of artificial neurons, of different sizes. One important aspect is the fact that the model improves when trying to predict one component of the SV propagation error at a time, which is due to the fact that different model architectures work better for the different components of the SV. Therefore, three different machine learning models are implemented, one for each position component of the SV (X, Y and Z) of the secondary object.

A machine learning model based on artificial neural networks, like the DNN, learns from the training dataset through the backpropagation algorithm. In short, the algorithm starts by feeding through the neural network a sample of the training set and calculates, in turn, the output from each neuron of each layer. Then, the algorithm calculates the error between the output of the network and the target value (or values) it is trying to predict and calculates the contribution of each neuron on the last layer to the error obtained. The next step is to calculate how much of these error contributions are due to each neuron in the layer immediately preceding the last layer, and so on, until the algorithm arrives at the input values. The gradient of the function defining the error is calculated for each neuron in the network, thus determining the "direction" in which the weights of each artificial neuron need to be changed. As part of an optimization algorithm, the size of the step taken in this proper “direction” is called the learning rate and it is a configurable hyperparameter.

In regards to what the model is actually predicting, which is the difference between the propagated SV in the

current CDM at TCA and the true SV at TCA, it is important to note how the true value is computed. Since in the dataset there is no indication of what the true SV at TCA actually is, the final CDM in every event has been considered to be the true SV at TCA. However, in almost all cases, the TCA in the final CDM of an event does not coincide with the TCA in the other CDMs in the event and it varies slightly. For this matter, a "synchronisation" process has been performed, through which the SV in the final CDM is propagated to the TCA of each of the CDMs in the event. After this process, the difference in the SV components between the newly propagated SV (from the last CDM in the event) and the SV in the current CDM is calculated, giving the target parameters of the model. A description of the synchronisation process is given in the below image

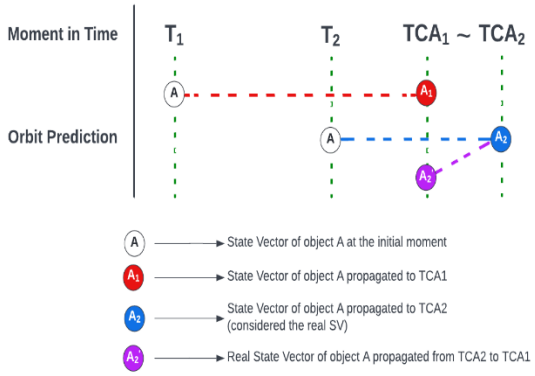

Figure 4 – State Vector Target Synchronization Process

3.5 Configurable parameters

In order to perform the training of a machine learning model, there are some parameters that need to be configured first. Within this study, the following hyperparameters were studied: "batch size", "learning rate", "number of hidden layers", "number of artificial neurons in each layer". In order to obtain the hyperparameter values, a "random search" technique was used which consists in providing multiple options or an interval range of values for each hyperparameter. The next step would be to train a configurable amount of n models. Once the training has been completed, the model that obtained the highest performance will be stored, along with the hyperparameter values.

For the other parameters such as the optimizer and the loss function, the ADAM optimizer [12] was selected and the MSE loss function was used for covariance prediction and the L1 loss function was used for the state vector prediction.

3.6 PRELIMINARY ANALYSIS RESULTS

Within this chapter the preliminary results are presented for the covariance prediction and for the state vector prediction models. Furthermore, details of the metrics used in order to evaluate the performance of the models are described as well.

3.6.1 Metric used for Covariance Prediction FoM

For evaluating model results, a metric called Figure of Merit (FoM) was defined, which is designed to represent objectively the model performance on a given validation dataset.

$$FoM = 1 - avg\begin{pmatrix} med(rel_{err_{t1}}), med(rel_{err_{t2}}), \\ \dots, med(rel\_err_{tn}) \end{pmatrix} \quad (8)$$

The range of some values even after filtering outliers is large. The logic behind this metric is that a robust metric that would provide representative information with respect to the model performance is needed, while weighting the outliers less.

For each target label that the model is trained to predict, the relative error of the predicted value with respect to the target value is computed. The median is selected from the distribution formed by the relative errors for a specific target throughout all the events. The median was chosen instead of the average because the median represents the middle score for a set of data arranged in order of magnitude, thus being less affected by outliers or skewed data. The medians are averaged, which gives us an average "median" relative error.

Finally, the result value is subtracted from 1 in order to normalize the results. This step ensures a value that is human readable. The upper bound value is 1, which indicates no error within the model prediction.

3.6.2 Covariance Prediction Results

In order to find the Machine Learning models with the highest performance, large empirical tests were performed on the LSTM with attention mechanism architecture.

The tables and figures presented in this section illustrate the results in terms of FoM for all covariance prediction cases:

- 24 hours after the last available update
- 48 hours after the last available update
- 72 hours to the time of closest approach
- Time of closest approach.

24 hours after the last available update.

	LSTM	LSTM with Attention
FoM score	0.826	**0.828**

Table 1 - 24h after last CDM covariance evolution model

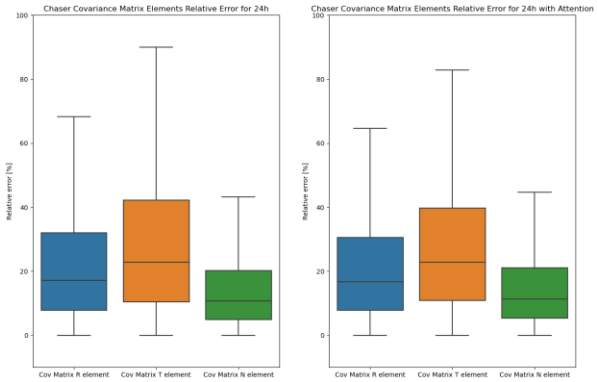

Figure 5 – Prediction of covariance at 24h LSTM vs attention model

From the error spread figure, a mild improvement can be seen in the spread of the relative error for the T element, which represents the hardest element to predict, as well as mild improvements in the R and N elements.

48 hours after the last available update.

	LSTM	LSTM with Attention
FoM score	0.805	**0.806**

Table 2 - 48h after last CDM covariance evolution model

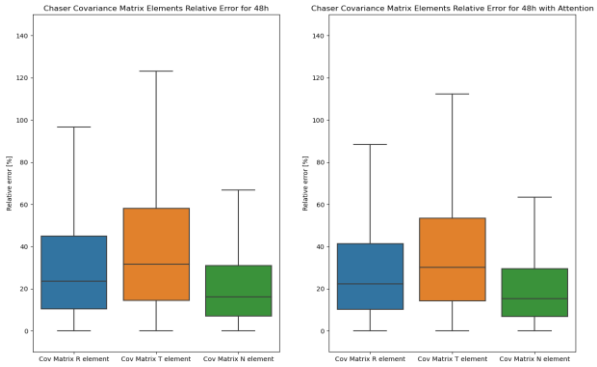

Figure 6 – Prediction of covariance at 48h LSTM vs attention model

In this prediction case, the spread of the T element relative error in the Bahdanau attention is higher but the median is lower. The same pattern can be observed for the R and N element's relative error.

This is visible in the table of results, where the Badhanau attention obtained a slightly higher FoM.

72 hours to the time of closest approach.

	LSTM	LSTM with Attention
FoM score	0.852	**0.861**

Table 3 - 72h before TCA covariance model

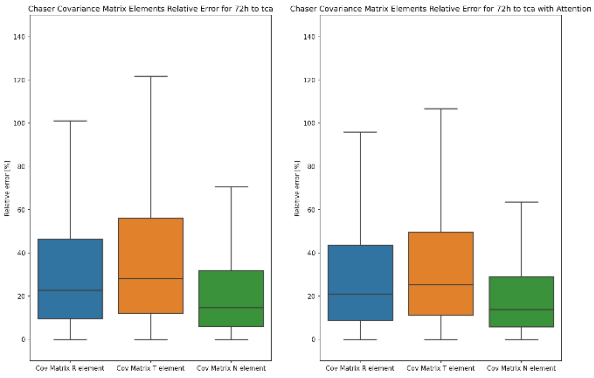

Figure 7 – Prediction of covariance 72h before TCA LSTM vs attention model

In the 72 hours to TCA case, both the spread and median of the relative error for all elements offer an improvement.

This is visible in the table of results, where the Badhanau attention obtained a slightly higher FoM.

Time of closest approach

	LSTM	LSTM with Attention
FoM score	0.513	**0.702**

Table 4 - Covariance at TCA model

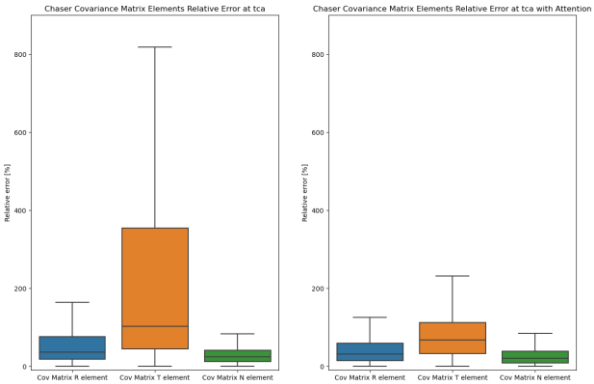

Figure 8 – Prediction of covariance at TCA LSTM vs Attention model

It is expected that the attention mechanism to achieve the highest performance increase when predicting at a far point in time such as predicting the covariance elements at the time of the closest approach.

From both the figure and the results table, the attention model outperformed the simple LSTM model by a considerable margin.

From the error spread figure, a significant difference in the spread of the relative error can be seen. Furthermore, one can notice a substantial difference in the median of the relative errors, especially in the hardest element to predict, the T element.

3.7 Metrics used for State Vector prediction

The mean squared error (MSE) estimates how close a regression model fits the data. It measures the average of the squares of the errors. That is, the average squared distance between the estimated, $\hat{y}$, and the real values, y. As it derives from the square of the Euclidean distance, it has always a positive value and the smaller the error, the better the predictive model is. The MSE incorporates the variance of the estimator (how widely spread the predictions are from one data sample to another) and its bias (how far off the average estimated values are from the actual values):

$$MSE(\hat{y}) = E_y\left[\left(\hat{y} - E_y[\hat{y}]\right)^2\right] + \left(E_y[\hat{y}] - \hat{y}\right)^2 \quad (9)$$

For an unbiased estimator, this metric represents the variance of the estimator. Assuming a vector y with n predictors the MSE is given:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|^2 \quad (10)$$

Taking the square root of the MSE yields the RMSE. This measure is always non-negative, and a zero value indicates a perfect fit to the data. For an unbiased estimator, the RMSE is the square root of the variance, also known as standard error. The error value is easier to interpret since it can be read with the same units as y.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|^2} \quad (11)$$

The Mean Absolute Error (MAE) is an arithmetic average of the absolute errors between the predictions and the true values:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \quad (12)$$

This metric uses the same scale as the data being measured and, therefore, cannot be used to make comparisons between variables with different measuring units.

The metrics used for estimating the performance of the SV machine learning models are the MAE and the RMSE.

3.8 State Vector prediction results

In order to determine the performance of the model, a baseline model is also implemented, for comparison purposes. Since the ML model is predicting the difference between the SV in any given CDM and the real SV at TCA, the baseline is considered to predict the same difference, always outputting a value of 0. The improved SV of the baseline is therefore the propagation given in the original CDM. In this case, if the ML model performs better than the baseline, consequently it represents an improvement with respect to the propagation given in the original CDM, which is the final goal of implementing the model.

Id.	State Vector	MAE [km]		RMSE [km]	
		DNN	Baseline	DNN	Baseline
1	X comp	0.754	1.113	3.022	3.951
2	Y comp	0.797	1.091	3.189	3.650
3	Z comp	0.462	0.718	2.097	3.114

From the above table it can be observed that the machine learning models are able to improve the SV position components in all cases by a relatively significant margin.

4 AKNOWLEDGEMENTS

This activity has been carried out under the ESA funded project CREAM#1.

5 References

[1] K. J. DeMars and J. S. McCabe, Multi-Sensor Data Fusion in Non-Gaussian Orbit Determination, 2014.

[2] Arambel, Rago and Mehra, Covariance Intersection Algorithm for Disturbed Spacecraft State Estimation, 2001.

[3] M. B. Hurley, An Information-Theoretic Justification for Covariance Intersection and its Generalization, 2001.

[4] M. Reinhardt, Analysis of Set-Theoretic and Stochastic Models for Fusion under Unknown Correlations, 2011.

[5] D. Franken and A. Hupper, Improved Fast Covariance Intersection for Distributed Data Fusion, 2005.

[6] W. Niehsen, Information Fusion based on Fast Covariance Intersection Filtering, 2002.

[7] M. R. Akella and K. T. Alfriend, "The Probability of Collision Between Space Objects," *Journal of Guidance, Control and Dynamics,* 2000.

[8] Q. Transformer, "Scikit-learn Documentation," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html.

[9] Bahdanau, Dzmitry, Cho, Kyunghyun and Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, 2014.

[10] M. T. Luong, H. Pham and C. D. Manning, Effective approaches to attention-based neural machine translation, 2015.

[11] H. Peng and X. Bai, Improving Orbit Prediction Accuracy through supervised Machine Learning, 2018.

[12] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, 2014.