

Clustering-based uncorrelated track association

Louis Penaffiel

Aptima, Inc.

William Dupree

Aptima, Inc.

Thomas Gemmer

Aptima, Inc.

CONFERENCE PAPER

Abstract: In this work, we explore the application of unsupervised machine learning to Space Domain Awareness (SDA) by associating uncorrelated tracks (UCTs) to known resident space objects (RSOs). This is done by using a two-step, unsupervised machine learning model to filter the Space Catalog in order to find objects with similar orbital elements to the UCT. We present results in the Catalog in order to find objects with similar orbital elements to the UCT. We present results in the context of improving analyst workflows related to UCT association by automating our models and validating our approach on metrics of accuracy and processing time.

1. INTRODUCTION

Every year the boundaries of space exploration are pushed to new heights. From 2019 to 2028, approximately 10,000 satellites are slated to be launched into space. These, along with other rocket launches, will vastly increase the number of objects (including active satellites and debris) in space to track. This makes the demand for more capable and more automatic tools and analytics to analyze these objects a primary concern. It is necessary to keep track of what satellites are doing to know if a satellite is in danger of collision or is being threatened in some other way [1, 2]. A major task in that area is associating uncorrelated tracks (UCTs) to known resident space objects (RSOs). UCTs mainly arise when an object is poorly tracked (making it difficult for sensors to pick up) or an object has maneuvered (making its trajectory significantly different from the most recently known orbit). This task has mainly been performed manually by analysts, making it time and resource intensive.

In this study, we focus on automating the UCT correlation procedure through analyzing historical data to characterize the relationships between UCTs and cataloged objects, such as origin (e.g., debris or loss of custody) or threat (UCT will threaten a cataloged object). We apply this study in the context of increasing the effectiveness of space operators by creating a microservice called WhoAmI. WhoAmI performs the UCT association, then produces human-understandable alerts containing a rank-ordered list of the most probable RSOs that the UCT might be associated to. WhoAmI performs associations of UCTs to cataloged RSOs using a joint approach of machine learning combined with astrodynamics analysis. We use a two-step, unsupervised machine learning model to filter the Space Catalog in order to find objects with similar orbital elements to the UCT. The first step uses density-based clustering of the catalog in orbital element space to find cataloged objects in the astrodynamics vicinity of a UCT. The second step uses propagation models to perform a more traditional pass of filtering by directly analyzing the orbital paths of the filtered candidates. In addition to directly associating UCTs to a catalog, we use a similar approach to matching the UCTs to their most likely launches. This allows WhoAmI to automatically find origin information for debris objects so that they can be entered into the catalog as well as making it potentially useful in breakup analysis. The results of this process are then passed via alerts for space operators with visualizations of probability of association to act on. We validate our approach on metrics of accuracy (using historical tracking data and maneuvers as baselines) and processing time. In Section 2, we introduce the data sources we will be using and the automation of UCT association. In section 3, we introduce how we augment that automation with clustering and identifying most likely launches. In section 4, we perform model experiments, and in Section 5 we give our conclusion.

2. DATA SOURCES AND AUTOMATING UCT ANALYSIS

The main data source for this study will be from Space-Track due to their large catalog and history of two-line elements (TLEs) [3]. However, we have also set up data pipelines for data coming from the Unified Data Library

(UDL) [4]. The current number of actively tracked satellites in Space-Track is 21,815, and for every observation for each of these satellites, there is an associated TLE. Therefore, if we want to automate an observation of a target UCT TLE, then we can compare that TLE to the TLEs of the RSOs in Space-Track, which we will now refer to as the catalog.

Because a TLE is just an approximation of the actual orbit of satellites and the time of observations of that in the catalog vary, to fully compare orbits, we cannot only compare the TLEs. We would need to propagate the orbits from these TLEs and compare the distances and velocities of these propagated tracks to find the RSO with the least number of differences from the satellite at the same time point. This RSO will be our top candidate for what the UCT is associated to. The propagations in this study were done using the Python wrapper of the Orekit library, but as propagation is not the main focus, WhoAmI is created to be more robust to other propagation codes [5].

For one target TLE, this would be a computationally intensive task, primarily because of the propagation. In a 64 GB RAM Windows 10 laptop running the Python version of Orekit, iterating and propagating over the whole catalog of RSOs would take around 20 to 22 hours. Also considering that TLEs change with new observations, it is necessary to be able to perform this calculation in a frequent manner. We also want to bring up the questionable predictive power of TLEs in general when it comes to propagation, and this manner of UCT track correlation is more of a preliminary pass to generate human-readable alerts to analysts, who will take a deeper dive into the UCT. The main constraint of this automation is the number of RSOs that need propagation performed and the number we then compare against. We propose the use of a technique based on unsupervised machine learning methods called iterative gating, as a way to reduce the 21,815 satellites in the catalog to a smaller number of candidates.

3. CLUSTERING

The main algorithm for this iterative “gating” strategy is illustrated in Fig. 1 to clusters based on the variables from the TLEs. Essentially, it is:

1. Find clusters at higher levels.
2. Data is rescaled using standard scaling.
3. Drop outliers and re-cluster at lower levels.
4. Repeat steps 1-3 until the target is in its own cluster and does not cluster with anything. The last cluster it belongs to in the previous step is the group we keep for performing the propagation on.

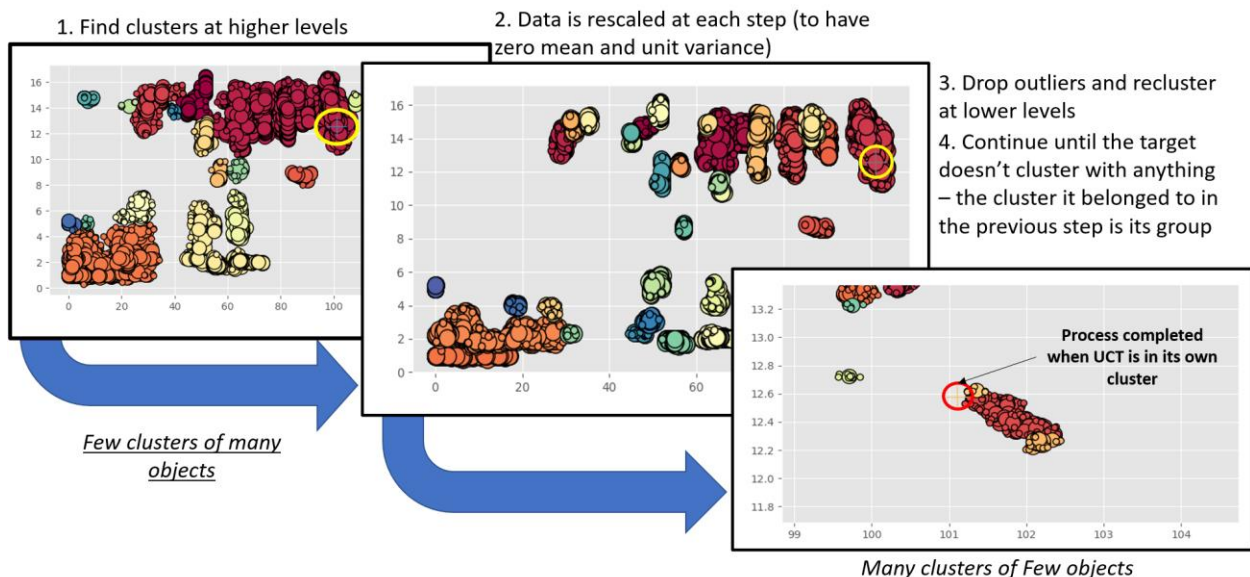


Fig. 1. Illustration of iterative gating algorithm. Essentially, for each iteration we perform a clustering and then remove the outliers. Once the target becomes an outlier, that is when the iteration ends and the cluster it belonged to in the previous iteration is the group we perform the propagation on.

Since we are basing this information on the TLEs, we investigate which TLE variables would be appropriate to use for this iterative gating algorithm. We chose seven TLE features because they are the most stable across time slices:

inclination, eccentricity, mean motion, semimajor axis, apogee, perigee, and period, as shown in Fig. 2. An example of an unstable feature is the pericenter argument in Fig. 3.

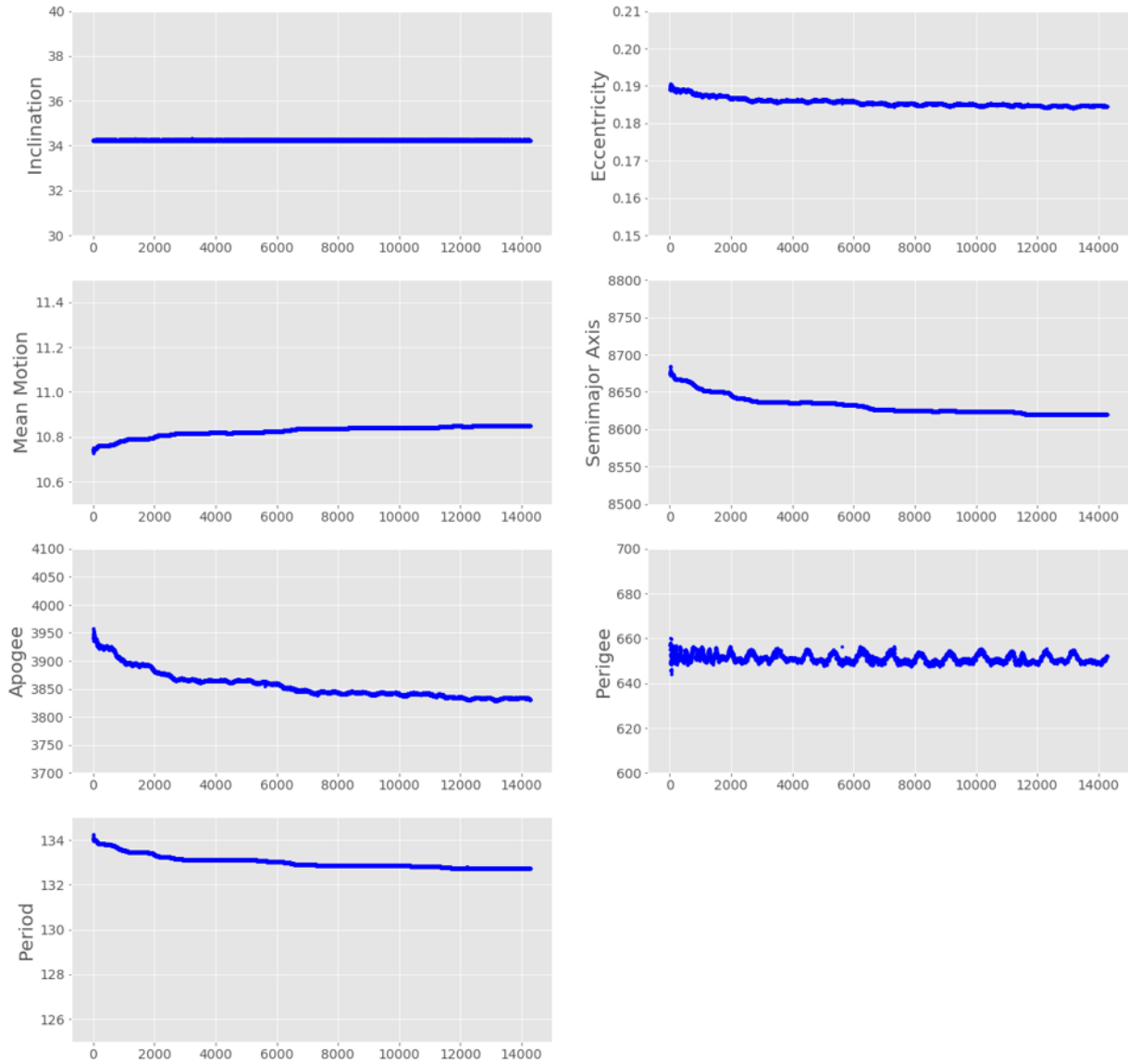


Fig. 2. Vanguard 1 (SATNO 00005) ‘stable’ orbital parameters from the TLEs. We show observations across 51 years (1960-2021), displayed in temporal order from left to right. The features are inclination, eccentricity, mean motion, semimajor axis, apogee, perigee, and period, respectively.

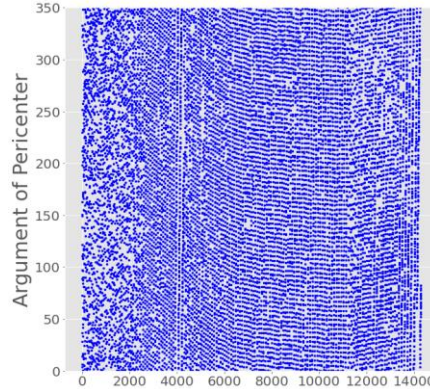


Fig. 3. Example of an ‘unstable’ orbital parameter (argument of pericenter) from TLEs. We show observations of Vanguard 1 satellite across 51 years (1960-2021), displayed in temporal order from left to right.

3.1 DBSCAN

The original method used to drive this clustering approach is Density Based Spatial Clustering of Applications with Noise (DBSCAN) [6]. DBSCAN clusters together points that are close to each other based on two parameters: the minimum number of points to form a cluster (minPts), and the distance that specifies how close the points should be to be considered a part of a cluster, ϵ . This distance used for ϵ is based on the distance metric used. For our purposes we use the Euclidean distance metric $d(p, q) = (\sum_i (q_i - p_i)^2)^{1/2}$, where the sum is over all possible i s, features.

With these two parameters, the core algorithm of DBSCAN is as follows [7]:

1. Find the points in the ϵ neighborhood of every point and identify the core points which have number of neighbors greater than minPts.
2. Connect the components of core points on the neighbor graph and ignore all non-core points.
3. Assign each non-core point to a nearby cluster if it is within the ϵ neighborhood of that cluster, otherwise it is noise.

Combining this with iterative gating, we essentially use DBSCAN to be able to identify outliers and remove them. For our use case, the parameters may need to be changed for each pass through the iterative gating algorithm. The main parameter issue is not minPts, but ϵ . This is because as the outliers get dropped, the already dense regions in the dataset would stay the same. We implement the elbow method to identify the proper ϵ at each iterative gate [8].

Furthermore, the ϵ needs to be an input that defines the maximum distance between two points. The method that we employ first calculates the distance to the nearest n points for each point, sorting and plotting the results. To calculate this, we use the kNearestNeighbors algorithm, which needs the number of neighbors, $n_neighbors$, to make that calculation [9]. Note that kNearestNeighbors is a classification algorithm based on the labels of neighbors. However, for our purpose, we only use it to make the distance calculations. We chose $n_neighbors$ to be $minPts+1$, unless the number of samples is less than $n_neighbors$. In that case, we make the $n_neighbors$ the number of samples. This second scenario will occur near the last iterations of the iterative gating algorithm.

The “elbow” comes into play after sorting and plotting these distances. The optimal value for ϵ is where that plot is at maximum curvature, for which an example is shown in Fig. 4. To automate this portion, we take advantage of the kneed Python package [10]. The kneed package performs the calculation of finding the elbow point, given a list of distances.

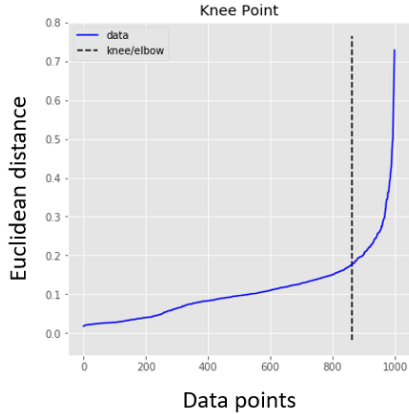


Fig. 4. Plot of distances for the catalog from Space-Track. These Euclidean distances are calculated by using the 7 features identified in Section 2, then scaled using Standard Scaling. The knee/elbow point was calculated using the kneed package and the ϵ is the Euclidean distance at the point where the data line and dotted line intersect.

Now implementing this elbow point, we refine the original iterative gating algorithm:

1. Find clusters at higher levels using DBSCAN by first scaling with Standard Scaling, and with ϵ found through the kneed package and $\text{minPts} = 10$.
2. Drop outliers. If there were no outliers, rescale ϵ by .8, meaning that the maximum distance between two points is decreased by a factor of .8.
3. Re-cluster at lower levels.
4. Repeat steps 1-3 until the target is in its own cluster and does not cluster with anything, or there is no elbow point found. The last cluster it belongs to in the previous step is the group we keep for performing the propagation on.

This factor of .8 was chosen ad hoc as a large enough factor to still be robust enough to minor differences while keeping the iterations significantly different. Subsequent tests may adjust that factor based on findings.

The initial implementation was based on an older version of scikit-learn package, which did not support multi-dimensional DBSCAN [11]. This led to performing DBSCAN on two-dimensional combinations of the features and aggregating the cluster results for each set. This was also computationally intensive. Therefore, when the new version of scikit-learn was released that allowed for multi-dimensional DBSCAN, we switched and saw improvements in speed and in clustering because the aggregation step for the two-dimensional DBSCAN combinations was inclusive, in the sense that as long as an RSO is in a cluster for any of those 7C2 possible combinations then that will survive for the next iteration in iterative gating. The clustering improvement is quantified by as follows: 2D Combination DBSCAN ($O(10^3)$) to Multi-dimensional DBSCAN with adaptive ϵ ($O(10^2)$), where the value in the parentheses is the order of magnitude of the result of the clustering, to what we call the trimmed catalog.

3.2 PCA + DBSCAN

One other augmentation we implemented was the use of Principal Component Analysis (PCA) to perform dimensional reduction on the feature space before performing DBSCAN [12]. The reasoning behind using PCA is that even after standard scaling there will be variance between the various features and that these features are highly correlated to each other. Looking at the number of principal components (dimensions) necessary to explain the 7 features, we find the number to be 1 (as shown in Fig. 6), by computing it for the values in the catalog.

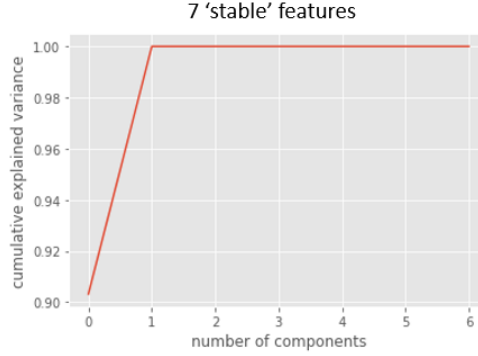


Fig. 6. Plot of the cumulative explained variance against the number of components after performing PCA on the 7 features of the satellites in the Space-Track catalog.

We then use the scikit-learn implementation of PCA with the addition of PCA, which gave another speed increase and clustering improvement because of it being one-dimensional DBSCAN. This clustering improvement is as follows: 2D Combination DBSCAN ($O(10^3)$), to Multi-dimensional DBSCAN ($O(10^2)$), to Multi-dimensional DBSCAN + PCA ($O(10)$), where the value in the parentheses is the order of magnitude of the result of the clustering, to what we call the trimmed catalog.

3.3 CONFIDENCE SCORE

After clustering, we take the resulting group, then calculate a distance for each RSO in that group to the target UCT. This is done by

$$d_{sep} = \sum_{i \in features} \frac{|o_{targ}^i - o_{RSO}^i|}{\bar{o}_{RSO}^i}$$

where d_{sep} is the distance separation, $features$ is the feature set and $o_{targ/RSO}$ corresponds to the target object or RSO, respectively, o_{RSO}^i with the bar corresponding to the average [13]. The denominator corresponds to the mean value of that feature for all those remaining in the group of RSOs. The lower d_{sep} the better the ranking for that RSO. As this separation has a range of $[0, \infty)$, we perform some transformations to scale these values from 0 to 1, where the closer the value is to 1, the better the ranking. The current version of this transformation is

$$f(x) = \frac{\frac{1}{x}}{\frac{1}{x} + 2}$$

The $\frac{1}{x}$ portion makes it so that the closer x is to 0, the higher the transformed value will be. The denominator is used to scale it from 0 to 1. The value of 2 is used simply to flatten out the curve a bit more. If we want to make the curve sharper, we decrease the value of 2. Our system has a confidence value from 0 to 10, so we then simply multiply this value by 10. We then pass along the top 5 for the visualization. The reason we send the top 5 is because even though we greatly reduce the number of RSOs to perform propagation on, it may still take some time for the propagation to compute. Hence, we send the top 5 as a first pass as an alert to analysts before a more granular ranking is created.

3.4 MOST LIKELY LAUNCHES

In the event that we fail to associate a UCT to a space catalog object, i.e., low confidence scores, we try to associate the UCT to a launch. The rationale is that the UCT is likely a debris from a launch and is potentially a new object that can be added to the RSO. The way we do this is through taking the trimmed catalog from the result of the clustering and look at each candidate RSO's launch. Then for each launch, we calculate the launch separation similar to how we calculate the distance separation above

$$l_{sep} = \sum_{j \in launch} \sum_{i \in features} \frac{|l_{targ}^i - l_j^i|}{\bar{l}_j^i}$$

where l corresponds to o from the distance separation equation, but for launches, over the same feature set. This time we aggregate the separation by using each RSO that has that launch date value ($j \in launch$). However, from application of this launch separation metric, it is not enough to get the correct launch date. We developed another launch separation value that is weighted by the number of RSOs in the trimmed catalog because of the similar argument that in the debris use case, the orbital parameters are similar between debris, the target debris will be more likely clustered with debris from the same launch. This metric weighed launch separation is defined as

$$l_{weighted\ sep} = \frac{\sum_{i \in features} |l_{targ}^i - \bar{l}_j^i|}{N_{features} N_{cand}}$$

where this metric is the sum of the target satellite's values over the features subtracted by the mean over all RSOs in the same launch (j). The weighting comes by dividing by the number of features, $N_{features}$ which is always 7, and the number of candidates, N_{cand} in the trimmed catalog that match the launch date. This weighting will reduce the launch separation more for the launches that have more RSOs in the trimmed catalog. Lastly, we take the sum of these two metrics, with the most likely launch being the one with the least value.

4. EXPERIMENTS

The workflow of the WhoAmI algorithm is as follows. Given a target UCT TLE, we perform iterative gating of that TLE against the catalog of TLEs from Space-Track, resulting in a trimmed catalog which significantly reduce the number of RSO candidates. Furthermore, a top 5 candidate ranking is made based on the confidence scores from the iterative gating. Then, if those scores are high, we perform propagation of the target TLE against the candidates in the trimmed catalog. If the scores are low, we perform a launch association.

To illustrate this workflow, we perform two types of experiments: simulation and real data. In the simulation scenario, we simulate a "UCT" by labelling a known object as a UCT and then try to match an old TLE of that satellite to the most recent catalog. This first experiment also will handle an example of a debris. We also investigate different time slices to measure sensitivity to temporal decay. For the real data, we take a known UCT (80,000s range in Space-Track) and try to match it against the rest of the space catalog.

4.1 EXPERIMENT 1: SIMULATED UCT

For the first part of the simulated UCT experiment, we look at a 21-day old TLE of satellite number 40086, which is an Orbcomm satellite. Performing the iterative gating algorithm yields us with the top 5 candidates with the corresponding confidence score, shown in Fig. 7.

NORAD_CAT_ID	OBJECT_NAME	confidence
40091	ORBCOMM FM 103	8.272707
40086	ORBCOMM FM 109	7.829363
40087	ORBCOMM FM 107	7.136400
41188	ORBCOMM FM 117	5.724086
41179	ORBCOMM FM 114	5.614140

Fig. 7. Top 5 ranking of the candidates after performing iterative gating on a 21-day old TLE of satellite 40086.

Given this first pass, we then send an alert of these confidence scores, while in the background the propagation is running. The propagation correctly identified 40086 as the match to the simulated UCT, with the relative distances and velocities plotted at each time step, shown in Fig. 8. Note how in the clustering the correct RSO, 40086, was ranked second, but after running the propagation, the correct RSO was identified. Also note the vast range of the

relative distances and velocities. This has more to do with the propagation code we implemented as it was not a robust one.

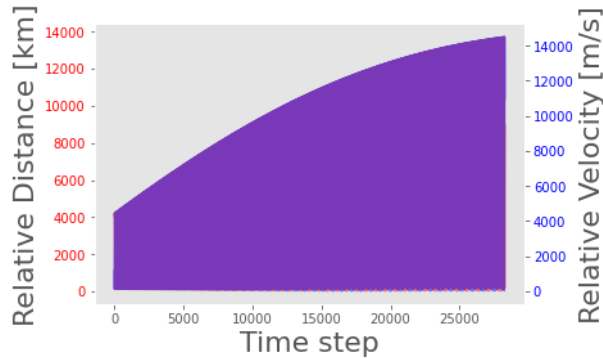


Fig. 8. Relative distances and velocities comparing satellite 40086’s most recent TLE to a 21-day old TLE. The red values correspond to the difference in distances, while the blue line corresponds to the difference in velocities of the propagated TLEs, for each time step of propagation. The key takeaway is that at some time step, there is a minimum relative distance and minimum relative velocity for each candidate, and this candidate had the global minimum across all candidates.

The second part of Experiment 1 is to perform this track correlation with a simulated UCT from a debris. We perform this calculation on a 13-day old TLE of RSO 34038, which is a COSMOS 2251 Debris. The top 5 ranking from the clustering portion and the correctly identified relative distances and velocities from the propagation portion are seen in Fig. 9., which shows everything matches.

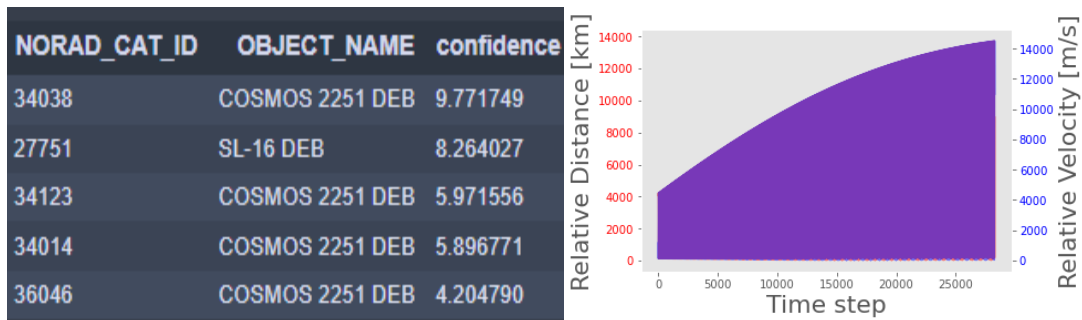


Fig. 9. **Left)** Top 5 ranking of the candidates after performing iterative gating on a 13-day old TLE of RSO 34038. **Right)** Relative distances and velocities comparing satellite 34038’s most recent TLE to a 13-day old TLE.

Results are then compiled on 100 attempted matches of varying TLE ages to better understand the temporal decay in the predictive power and approximate nature of TLEs and is shown in Fig. 10. Because propagating for a longer period of time in older TLEs will also lead to a longer propagation of errors, performing propagation on these older TLEs will be less likely to find matches, hence the full match—indicating both clustering and propagation results contained the correct target—is less. However, the reason why the clustering match—filter match—is still performing relatively better at older TLEs is mainly because of the stability of the 7 features chosen to perform the clustering on.

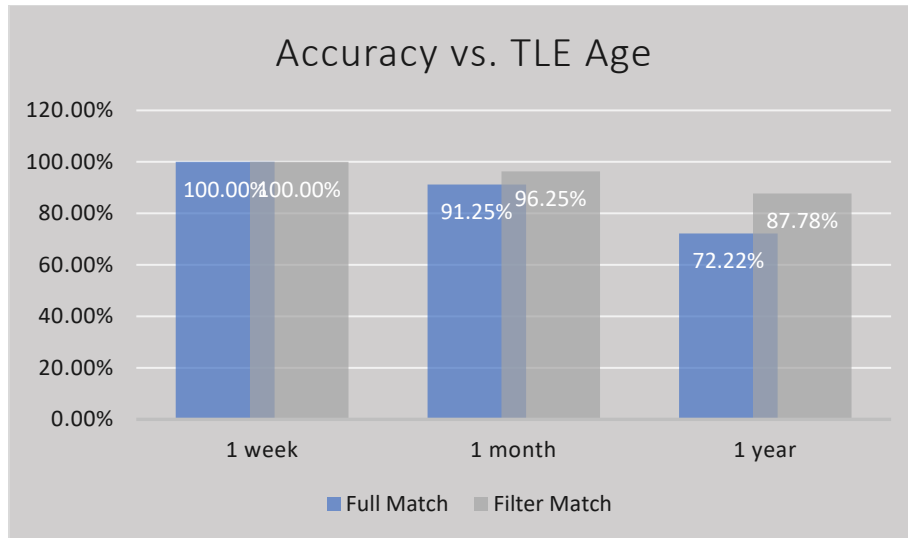


Fig. 10. Accuracy against TLE age for 100 attempted matches. Full match indicates both the clustering and propagation results contained the correct target, whereas filter match only indicates that the clustering trimmed catalog contained the target, but propagation results did not confirm.

4.2 EXPERIMENT 2: REAL UCT

We perform the same workflow on analyst objects from the 18th Space Control Squadron (SPCS) preliminary launch association hypothesis to test out the accuracy of WhoAmI on real UCTs. The results from the iterative gating on the most recent observation of UCT 81614 is shown in Fig. 11. Note that prior to clustering, we remove the UCT from the catalog, because the catalog already contains the most recent observation of that UCT.

NORAD_CAT_ID	OBJECT_NAME	confidence
18540	DELTA 1 DEB	6.109781
21677	DELTA 1 DEB	4.320949
21290	DELTA 1 DEB	4.297704
10760	DELTA 1 DEB	4.272656
21341	DELTA 1 DEB	4.184617

Fig. 11. Top 5 ranking of the candidates after performing iterative gating on the most recent TLE of UCT 81614.

Recall that the confidence score is a value from 0-10, so the results from the clustering are relatively poor confidence. Hence, we opt to use the launch association to find what the most likely launch is, the results of which are shown in Fig. 12. The launch association was able to correctly identify with the hypothesis from the Space-Track data of launch 1976-077.

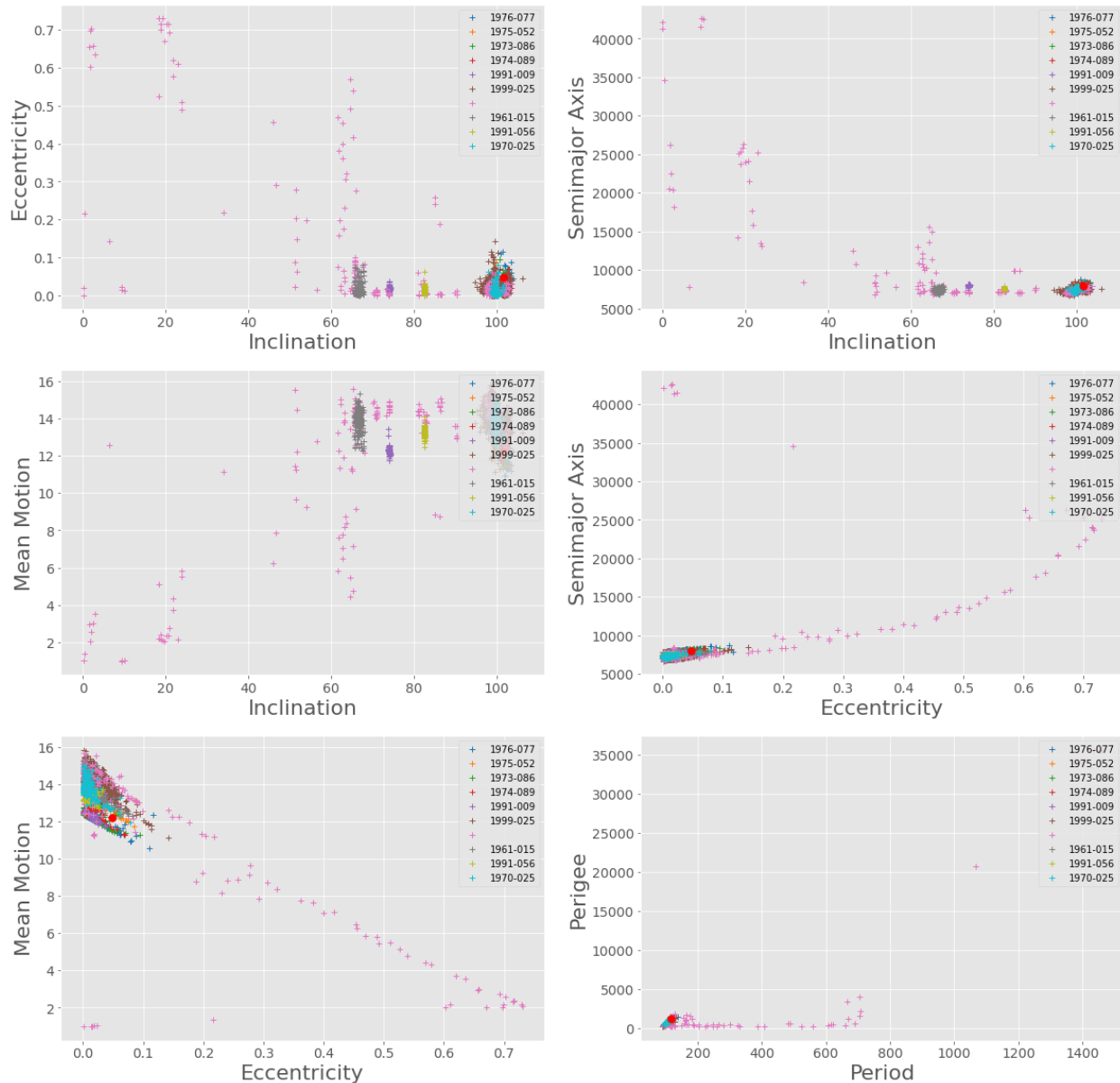


Fig. 12. The plots of various combinations of the 7 features for satellites of various launches. The red dot corresponds to the target UCT. The legend is ranked by the launch association, meaning that the first launch in the legend (1976-077) is the most likely launch association of this UCT, matching the hypothesis from 18 SPCS.

We also like to report cases where the multi-dimensional DBSCAN + PCA clusters too well for these analyst objects. They cluster too well in the sense that there are very few candidates in the trimmed catalog, meaning there are fewer launches to test against. An example of this is seen from using the most recent TLE of UCT 81816, which also had relatively poor confidence scores from iterative gating, hence the reason we look at the launch association. The launch association ranking is shown for both the multi-dimensional DBSCAN + PCA and just the multi-dimensional DBSCAN in Fig. 13.

1999-025	7	1977-121	48
1981-053	3	1961-015	116
2018-081	1	1984-069	20
1991-006	1	1982-055	61
1975-028	1	1981-021	16
1990-081	1	1976-103	4
1974-089	1	2021-043	4
		1971-015	8
		1982-099	3
		2020-066	3

Fig. 13. Launch association ranking from the trimmed catalog after performing the iterative gating algorithm on the most recent TLE of UCT 81816. The value on the right of the table is the number of candidate RSOs from that launch appearing in the trimmed catalog. The ranking on the left corresponds to the multi-dimensional DBSCAN + PCA clustering, while the ranking on the right corresponds to only the multi-dimensional DBSCAN clustering.

Because the clustering overperformed, there were no candidate RSOs of the correct launch appearing in the trimmed catalog. However, if we perform a less strict clustering (only multi-dimensional DBSCAN), then those candidates appear in the trimmed catalog, allowing us to perform a more accurate launch association. Further investigation is necessary to identify which clustering algorithm to use for these UCT TLEs, but these results are promising in the use case of launch association.

5. CONCLUSIONS

In this analysis, we have explained a novel method in combining orbital propagation analysis with unsupervised machine learning algorithms to perform UCT association. We investigated the use case of automating UCT association to assist analysts in identifying RSO candidates in an efficient manner. The algorithms and techniques were used to perform the iterative gating to significantly reduce the number of candidates from the whole catalog, to perform the computationally intensive propagation analysis on, were detailed. We performed experiments by simulating UCTs by using older TLEs of known RSOs, and analyzed the temporal decay of TLE and the accuracy associated with our clustering and propagation algorithms. Lastly, we assessed WhoAmI's effectiveness on actual UCTs and made use of the launch association capability to better handle poor results from clustering. Further studies need to be made to get a better handle on which of the clustering algorithms to use when performing launch association.

While this paper showed more of the details of the algorithm, the production version of the WhoAmI algorithm is a microservice that can be used to perform these calculations. WhoAmI is able to ingest various types of data, ranging from TLEs to raw observations, like those found in the UDL, by performing various orbit determination techniques based on the type of data received. This capability of WhoAmI, along with UCT association and launch association, enables operators to perform UCT resolution tasks more effectively by automating the process of filtering the number of RSOs to consider more confidently by prioritizing UCTs with the highest threat assessment, and more adaptively by reducing reliance on a single data type.

6. FUNDING

The work shared here was supported by AFWERX, Air Force Research Laboratory (Space Vehicle Directorate), and Space and Missile Center.

7. REFERENCES

- [1] C. Shabarekh, J. Kent-Bryant, G. Keselman, and A. Mitidis, "A Novel Method for Satellite Maneuver Prediction," in *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, 2016
- [2] T. Gemmer and C. Shabarekh, "Leveraging Non-Traditional Sources (NTS) for Space Situational Awareness (SSA) Analytics," in *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, 2018
- [3] "Space-Track." <https://space-track.org/> (accessed 2021).
- [4] "Unified Data Library." <https://unifieddatalibrary.com/> (accessed 2021).
- [5] Maisonobe, Luc & Pommier, Véronique & Parraud, Pascal. (2010). OREKIT: AN OPEN SOURCE LIBRARY FOR OPERATIONAL FLIGHT DYNAMICS APPLICATIONS.
- [6] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. (eds.). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press. pp. 226–231.
- [7] Schubert, Erich; Sander, Jörg; Ester, Martin; Kriegel, Hans Peter; Xu, Xiaowei (July 2017). "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". *ACM Trans. Database Syst.* **42** (3): 19:1–19:21. doi:10.1145/3068335. ISSN 0362-5915. S2CID 5156876
- [8] Nadia Rahmah and Imas Sukaesih Sitanggang *2016 IOP Conf. Ser.: Earth Environ. Sci.* 31 012012
- [9] Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (PDF) (Report). USAF School of Aviation Medicine, Randolph Field, Texas.
- [10] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior," *2011 31st International Conference on Distributed Computing Systems Workshops*, 2011, pp. 166-171, doi: 10.1109/ICDCSW.2011.20.
- [11] Scikit-learn: Machine Learning in Python, Pedregosa et al., *JMLR* 12, pp. 2825-2830, 2011.
- [12] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". *Philosophical Magazine.* **2** (11): 559–572. doi:10.1080/14786440109462720.
- [13] Reades, Jon & Calabrese, Francesco & Sevtsuk, Andres & Ratti, Carlo. (2007). Cellular Census: Explorations in Urban Data Collection. *Pervasive Computing, IEEE.* 6. 30-38. 10.1109/MPRV.2007.53.