Scalable Conjunction Processing using Spatiotemporally Indexed Ephemeris Data

Irene A. Budianto-Ho
Stellar Science Ltd Co, Albuquerque, NM 87110
Clay Alberty
Stellar Science Ltd Co, Albuquerque, NM 87110
Randall Scarberry
Stellar Science Ltd Co, Albuquerque, NM 87110
Stephen Johnson
Stellar Science Ltd Co, Albuquerque, NM 87110
Robert M. Sivilli
Air Force Research Laboratory, Kirtland AFB, NM 87117

ABSTRACT

The collision warnings produced by the Joint Space Operations Center (JSpOC) are of critical importance in protecting U.S. and allied spacecraft against destructive collisions and protecting the lives of astronauts during space flight. As the Space Surveillance Network (SSN) improves its sensor capabilities for tracking small and dim space objects, the number of tracked objects increases from thousands to hundreds of thousands of objects, while the number of potential conjunctions increases with the square of the number of tracked objects. Classical filtering techniques such as apogee and perigee filters have proven insufficient. Novel and orders of magnitude faster conjunction analysis algorithms are required to find conjunctions in a timely manner.

Stellar Science has developed innovative filtering techniques for satellite conjunction processing using spatiotemporally indexed ephemeris data that efficiently and accurately reduces the number of objects requiring high-fidelity and computationally-intensive conjunction analysis. Two such algorithms, one based on the *k*-d Tree pioneered in robotics applications and the other based on Spatial Hashing used in computer gaming and animation, use, at worst, an initial $O(N \log N)$ preprocessing pass (where N is the number of tracked objects) to build large $O(N)$ spatial data structures that substantially reduce the required number of $O(N^2)$ computations, substituting linear memory usage for quadratic processing time.

The filters have been implemented as Open Services Gateway initiative (OSGi) plug-ins for the Continuous Anomalous Orbital Situation Discriminator (CAOS-D) conjunction analysis architecture. We have demonstrated the effectiveness, efficiency, and scalability of the techniques using a catalog of 100,000 objects, an analysis window of one day, on a 64-core computer with 1TB shared memory. Each algorithm can process the full catalog in six minutes or less, almost a twenty-fold performance improvement over the baseline implementation running on the same machine. We will present an overview of the algorithms and results that demonstrate the scalability of our concepts.

1. INTRODUCTION

The United States has been concerned with Space Situational Awareness (SSA) since the launch of Sputnik 1 in 1957. At that time only a single satellite needed to be tracked, but over the years the number of tracked objects orbiting the earth has grown to over ten thousand, including operational spacecraft, uncontrolled dead satellites, spent rocket bodies, upper stages, and space debris. Each of those objects travels at extremely high velocities and represents a potential threat to the safety of government and commercial multi-billion dollar irreplaceable assets. Unforeseen collisions pose a real and significant danger to the safety of humans in space.

Today, the responsibility to detect, track, identify, and catalog all observable objects orbiting the Earth is assigned to the United States Strategic Command's (USSTRATCOM) Joint Space Operations Center (JSpOC). The space surveillance duties of JSpOC include: tasking the Space Surveillance Network (SSN), a worldwide network of radar and optical sensor systems; maintaining the Space Catalog, a comprehensive listing of space object numbers, types and ephemeris parameters; performing conjunction analysis (CA) to predict potential on-orbit collisions; and alerting operators, when necessary, to maneuver their spacecraft out of harm's way.

Catalog Filtering Needs – The next decades of spaceflight are certain to greatly multiply the population of objects orbiting the Earth, making collision prediction more computationally challenging. Furthermore, the number of objects being tracked and catalogued will likely increase when more advanced sensor systems capable of detecting smaller objects are deployed. The Space Catalog is projected to grow to contain over 100,000 objects in the near future with multiple updates per day to each object's ephemeris. Accurate and efficient filtering techniques are needed to reduce the number of object pairings that need to be put through the precise but computationally-expensive conjunction analysis algorithm.

Classical Catalog Filtering Techniques – Several time independent filtering techniques are based on simplified analyses of orbital elements [1]. The apogee/perigee filter, for example, eliminates object pairings that should never collide since the apogee of one object is much lower than the perigee of the other. The orbit path filter is another technique based on orbit geometry that eliminates pairings whose minimum distance between the orbital paths is larger than the specified threshold. The time filter considers the time intervals in which an object pairing is within the threshold distance of each other's orbital planes. If the time intervals do not overlap, the pair can be eliminated from further analysis. Other screening algorithms that have been proposed and implemented are time dependent [2] [3]. These algorithms break down the entire window of analysis into smaller time steps and use the Cartesian positions and velocities to compare the axial and/or total separation distance between the two objects during the time step to determine potential conjunction.

Filtering techniques based simply on orbital elements have performance issues that limit their scalability as the size of the space object catalog increases. To be accurate, they must compare mean values rather than osculating values, which require sampling multiple points in the object trajectory. Often the orbital parameters used in the filter – such as orbital plane and line of nodes – are not directly available from the ephemeris and require extra computation to obtain. The threshold values used in these analyses may need to be loosened to counter some of the periodic effects in the trajectory so as not to incorrectly eliminate an object pair from further analysis. In addition, some of these techniques involve computationally-costly iterative schemes. The most fundamental deficiency of existing techniques, however, lies in the fact that these filtering methods require comparisons between all object pairs, resulting in tests in the order of N^2, where N is the total number of objects in the Space Catalog. Innovative filtering methods that can handle both current and future growth in the Space Catalog must boost efficiency by breaking the N^2 limitation while retaining full accuracy and reliability.

2. CONTINUOUS ANOMALOUS ORBITAL SITUATION DISCRIMINATOR

The Continuous Anomalous Orbital Situation Discriminator (CAOS-D) software is developed by Intelligent Software Solutions for the Air Force Research Laboratory (AFRL) to provide an extensible net-centric framework for conjunction analysis and optimized SSN tasking based on streaming updates of the Space Catalog and of available SSN sensors. Implemented in Java on top of the Open Services Gateway initiative (OSGi) plug-in based architecture, CAOS-D is designed to easily integrate and evaluate various approaches to catalog filtering, conjunction point identification, and risk assessment using a "plug-and-play" concept.

The conjunction analysis processes in CAOS-D are divided into three stages of Propagation, Conjunction Analysis, and Post Processing/Risk Assessment (see Fig. 1). Each stage can be implemented by a single plug-in or a chain of plug-ins. Available propagation plug-ins can process Two Line Element (TLE) sets, Vector Covariance Matrix (VCM) data, and interpolate ephemerides. The conjunction analysis stage may consist of time-independent and time-dependent filters that can quickly and accurately eliminate object pairs from further consideration prior to the conjunction point finding phase, which uses more expensive root finders to locate the points of minimum approach. Finally, the post-processing/risk assessment stage computes probability of collision and generates enhanced conjunction point information. The test results presented in this paper do not include this post-processing stage.

CAOS-D Validation – Reference [4] describes a set of time dependent filters implemented in CAOS-D based on the Conjunction Sieve (CSieve) tool developed by the Aerospace Corporation and inspired by [2], and the Smart Sieve method developed by the European Space Agency [3]. Reference [5] shows these CSieve- and Smart Sieve-based CAOS-D filters to produce consistent results when compared against five industry-standard conjunction analysis screening tools.

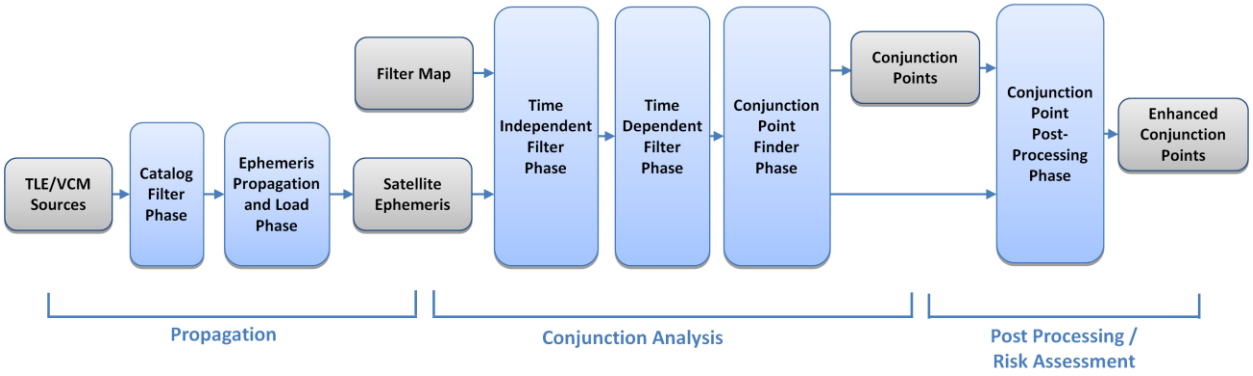

Fig. 1. Conjunction analysis phases and processes in CAOS-D.

The MITRE Corporation also conducted a study comparing various conjunction analysis tools, including CAOS-D. CAOS-D was reported to have identified 69 conjunctions not found by the industry-standard tool and missed 78 conjunctions, of which 49 involved an object that was not included in the CAOS-D run. The CAOS-D team resolved these issues by including the missing object, modifying the conversion tool provided by MITRE to round, rather than truncate, the times to the nearest millisecond, and using different post-filter and root-finder plug-ins to reduce the number of missed conjunctions at the boundaries of the analysis window. Following these changes, CAOS-D missed one conjunction event found by the industry-standard tool and found four additional conjunction events. These events have closest approach distance near the threshold distance ($R_{threshold}$) of 10 km and were attributed to numerical differences between the systems running the different programs. Another known difference between the industry-standard tool and CAOS-D is the treatment of docked objects. CAOS-D does not currently report docked objects as conjunction events.

CAOS-D Hybrid Filter Chain – Fig. 2 illustrates the CAOS-D hybrid filter chain, which testing has shown to be the most efficient combination of the built-in CAOS-D time dependent filters. The X-axis CSieve filter running in parallel with the Y-axis CSieve filter acts as the first phase of the filtering process. The Z-axis CSieve filter subsequently processes the intersection of the object pairs found by the X- and Y-axis CSieve filters. The remainder of the filter chain is run in series and requires more expensive operations to further eliminate object pairs from consideration. Descriptions of these filters can be found in [4]. The pDotV filter is equivalent to the RDotV filter described in the paper, but it performs additional checks at the beginning and end of the analysis window. The pDotV filter will report a conjunction if either the object pair is within the specified distance threshold and is moving apart at the beginning of the analysis window or the objects are within the specified distance threshold and moving closer together at the end of the analysis window.

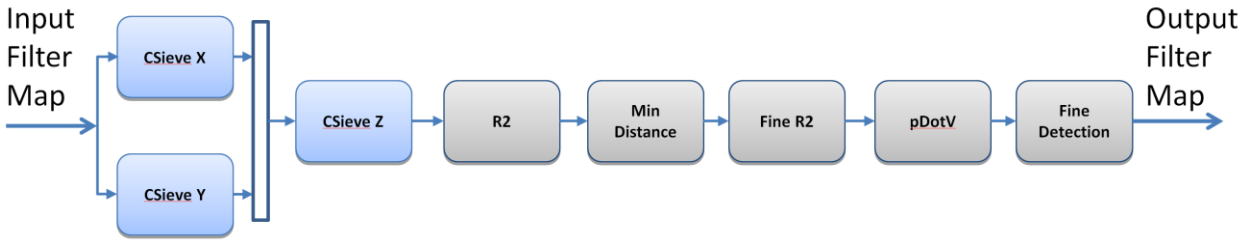

Fig. 2. CAOS-D hybrid filter chain is the most efficient time-dependent filter chain, using the single-axis CSieve-based filters followed by Smart Sieve-based and CSieve-based post-filters.

3. CATALOG FILTERING BASED ON SPATIOTEMPORALLY INDEXED DATA

Stellar Science has developed two primary time-dependent filtering techniques for satellite conjunction processing using spatiotemporally indexed data structures. Similar to the role of CAOS-D single-axis CSieve filters, the main objective is to quickly and effectively reduce the number of object pairings requiring high-fidelity and

computationally-intensive conjunction analysis. The first is the Extremum-Checking Spatial Partition Tree, or EC, based on the *k*-d tree concept pioneered in robotics and artificial intelligence applications [6]. The second is Cut Bounding Box (CBB) Spatial Hash based on the concept of spatial hashing, which has emerged from the computer graphics community as an efficient representation for spatial data [7]. Both filters have been implemented as OSGi plug-ins for CAOS-D and have been shown to be effective, efficient, and scalable as time-dependent filters. Our CAOS-D runs have further shown that, given the same input, substituting either EC or CBB for the single-axis CSieve-based filters generates the same output filter map as the hybrid filter chain. Validation tests are planned to check the performance of these filters against the industry-standard tool.

Extremum-Checking Spatial Partition Tree – This methodology inserts each catalog object into a six-dimensional *k*-d tree based on its minimum and maximum positions during the time step, which is computed as described in [4] and is based on a method outlined in [2]. In searching for potential conjunctions with a given primary object during the time step, the tree is traversed to find all other objects whose minimum and maximum positions overlap along all three axes, satisfying:

$$r_{i_{\text{secondary},min}} \leq r_{i_{\text{primary},max}} + R_{threshold} \text{ and } r_{i_{\text{secondary},max}} \geq r_{i_{\text{primary},min}} - R_{threshold}, \text{ for } i = x, y, \text{ or } z.$$

A graphical illustration is found in Fig. 3, showing an example where the EC algorithm will correctly identify a potential conjunction between objects *A* and *B*, whose extreme positions overlap and trajectories cross during the time step. The EC criterion uses simple mathematics to minimize computational cost and is purposely conservative in accounting for the objects' motion during the time step to avoid missing potential conjunctions, or false negatives. However, EC can produce false positives as shown in Fig. 3, where the extreme positions of the two objects overlap, resulting in a misidentification of a potential conjunction between objects *B* and *C*. For a conjunction analysis method to be reliable, it must satisfy the hard constraint that it does not produce false negatives (i.e., incorrectly classifying an object pair to not be in danger of collision, but whose closest approach distance during the time step is smaller than the specified threshold). False positives, on the other hand, do not affect the filter's accuracy, but they increase the computational burden of the subsequent step in the chain, having to process more data. Minimizing the number of false positives is therefore essential to maximizing the efficiency of the entire conjunction analysis process.

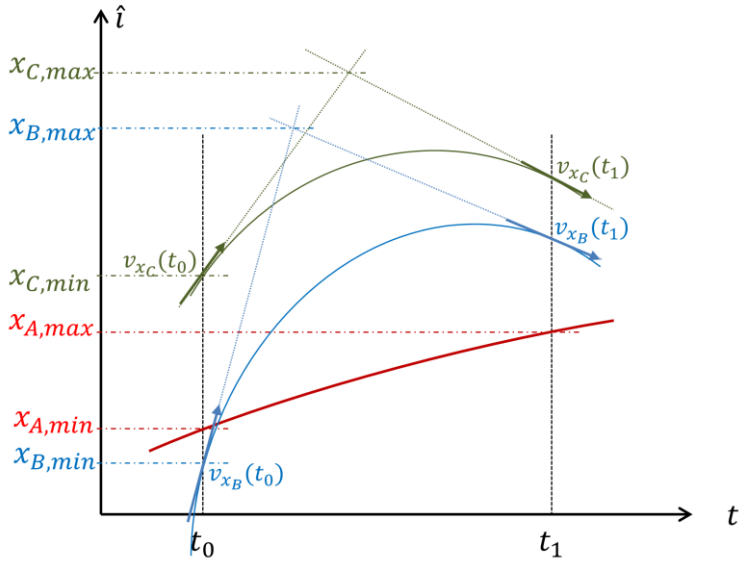

Fig. 3. Exaggerated view along the X-axis to illustrate Extremum-Checking criterion for correctly identifying conjunctions between objects *A* and *B* and for falsely identifying conjunctions between objects *B* and *C* (assuming negligible for $R_{threshold}$ simplification).

The efficiency of range searching operations on a *k*-d tree relies on its ability to quickly eliminate large regions of the search space. The search process is optimal given a balanced tree. Incrementally inserting objects into a spatial partition tree in a random order can result in an unbalanced tree whose branches have different depths. A badly-

balanced tree can result in at worst *O(N)* search behavior, while range searching a balanced tree requires *O(log N)* operations [8]. We implemented an option to sort the objects based on their extreme positions along one dimension, compute the median value along that dimension, and insert the object closest to that median value. The process continues, alternating dimensions, until all of the objects are added to the tree. A balanced tree can be built in *O(N log N)* time [9], while incremental insertion costs *O(log N)*, but the gain in the search time should outweigh the additional overhead from this sorting step. The tests presented in this paper are run using balanced *k*-d trees.

Cut Bounding Box Spatial Hash – A spatial hash table can efficiently represent a fixed grid space partitioning by projecting the three dimensional space into a one dimensional hash table. A hash function converts a grid cell coordinate (a triplet of *x*, *y*, *z* position coordinates) into a hash value that indexes a bucket in a fixed-size hash table. The CBB Spatial Hash algorithm inserts each catalog object into buckets that correspond to grid cells traversed by the object during the time step. The hash table is memory efficient as it does not need to directly represent the entire spatial grid in memory, but only the grid cells that contain one or more objects. Potential conjunctions are determined by pairing objects that occupy the same bucket.

CBB begins by defining a bounding box containing each object's trajectory during the time step based on its computed extreme positions, which is described in [4] and is based on a method outlined in [2]. This bounding box is the entire pink region shown in Fig. 4, surrounding the object's trajectory depicted in blue. The size of the bounding box depends on the time step and the motion of the object, but is typically much larger than a spatial grid cell, depicted by the white lines in Fig. 4. An object in Low Earth Orbit (LEO) with a 60 s time step, for example, may require a bounding box with dimensions of over 400 km, whereas a spatial grid cell measures typically 50 or 100 km along each axis.

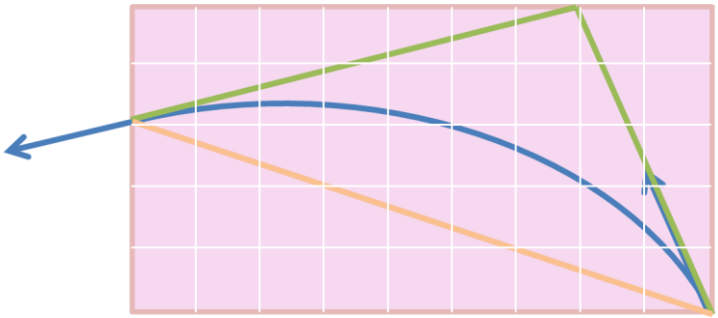

Fig. 4. Two-dimensional view of cuts to the Spatial Hashing Bounding Box.

To reduce the number of false positives, the algorithm further introduces cut planes to limit the spatial extent of the bounding box and only inserts the object into grid cells contained within these cut planes. The algorithm first defines the plane of travel, the plane containing the trajectory, using three points: the initial position, the final position, and the point of intersection between the projected velocity vectors. Five cut planes are computed using this plane of travel, along with the initial and final positions and velocities. The green lines represent two cut planes orthogonal to the plane of travel that contains the object's velocity vectors at the beginning and end of the time step. The orange line depicts a cut plane orthogonal to the plane of travel that contains the initial and final positions. The final two cut planes are parallel to the plane of travel, but shifted above and below by an offset distance.

The efficiency of the spatial hashing algorithm largely depends on the hash functions. The most popular spatial hashing algorithms take the form of either:

$$h_1(i,j,k) = i \times p_x + j \times p_y + k \times p_z \text{ or}$$

$$h_2(i,j,k) = (i \times p_x) \oplus (j \times p_y) \oplus (k \times p_z),$$

where p_x, p_y, and p_z are prime or pseudo-prime numbers and (i,j,k) is the integer coordinates or key used by the hash function to compute an index to an array of hash table bucket. (i,j,k) for a given position (x,y,z) is calculated using the floor function:

$$i = \left\lfloor \frac{x}{l} \right\rfloor, j = \left\lfloor \frac{y}{l} \right\rfloor, k = \left\lfloor \frac{z}{l} \right\rfloor,$$

where l is the cell dimension or length of the cubic bucket along each axis. The default value for l is the specified minimum distance threshold for determining potential conjunctions, $R_{threshold}$.

In our testing, h_1 outperforms h_2, resulting in fewer hash collisions, when used with prime numbers that are on the order of 10^7. Hash collision occurs when the hash function generates references to the same hash table bucket from two different keys, so two widely-separated points in space in two spatial grid cells with different (i, j, k) coordinates are assigned the same hash table bucket. Since the CBB algorithm pairs all objects contained in a bucket, hash collisions ultimately result in false positives. Smaller prime numbers allow faster processing of the hash computation, but can generate larger number of hash collisions, resulting in a correspondingly larger number of false positives and an overall loss of performance. Refer to [10] for more discussions on optimizing hash functions, hash table size, and cell size.

The Spatial Hash algorithm theoretically requires *O(N)* operations for both insertion and search, and thus, should outperform the Spatial Partition Tree for processing large catalogs. However, the insertion step involves more expensive computations to determine the bounding box, cut planes, and ultimately tens to hundreds of hash table buckets into which each object needs to be inserted. The search step performs much less expensive operations, iterating over all of the filled hash table buckets and pairing the objects contained within the same bucket.

Both Spatial Partition Tree and Spatial Hash algorithms theoretically require *O(N)* memory to store the entire catalog. However, while the Spatial Partition Tree stores each object into exactly one leaf node for each time step, the Spatial Hash may need to fill tens or hundreds of buckets per object, requiring tens to hundreds times more memory. Thus, memory usage is more critical for the Spatial Hashing algorithm. We added parameters to allow users to control initial memory allocation for the hash table, along with a load factor that determines when the table needs to be resized and memory needs to be reallocated.

4. AUXILIARY FILTERING STEPS

A key feature of the CAOS-D architecture is the ability to easily chain filter modules together in series or in parallel in order to investigate and ultimately select the best multi-stage filtering approach. However, the loading and unloading of plug-ins costs a non-negligible amount of overhead, especially if the different modules perform the same expensive calculations. Furthermore, the CAOS-D Application Programming Interface (API) provides restrictions to the type of input and output data to be passed between plug-ins. As a result, Stellar Science developed pre- and post-processing steps to further increase the efficiencies of our spatiotemporal filters and eliminate the need for redundant computations, without requiring customized modification to the published API.

Orbit Sorting Pre-Filter – We developed an orbit sorting algorithm that can be run prior to our spatial index filters. The algorithm classifies each object into one or more overlapping orbital regimes based on the radial distance of the object's minimum and maximum positions during the time step. Currently, the orbital regimes are defined as LEO contains the space below 3000 km altitude, while Medium Earth Orbit (MEO) spans from 2900 km to 35100 km altitude, and Geosynchronous Earth Orbit (GEO) contains the space above 35000 km altitude. The overlap distance is user-configurable and is necessary to prevent false negatives.

When used with EC, this orbit pre-sorting algorithm effectively divides the catalog into three separate and more compact *k*-d trees to represent the orbit regimes. Theoretically – assuming the ideal case where no objects are found in the overlap regions and the objects are equally distributed into the three regimes – the insertion step into each balanced tree costs *O*(*N/3 log N/3*) for an approximate total reduction of 10% for *N* = 100,000, while searching each tree for conjunctions with each object requires *O*(*log N/3*) for another 10% speed-up. This improved efficiency needs to compensate for the additional overhead involved in checking for orbital regimes.

The orbit pre-sort is not expected to provide substantive improvement when used with the spatial hash method, which would need to maintain three separate hash tables to represent the orbit regimes. The hashing algorithm spends the majority of its time in the insertion step, while the search step is the most expensive for EC. If an object

is classified into two orbit regimes, for example, CBB has the additional overhead of inserting the object into two tables.

Internal Post-Filter – CAOS-D Smart Sieve and CSieve-based post-filters have been shown to be accurate and effective in using more rigid computations to further eliminate conjunction points. We implemented these five post-filters internally to take advantage of data that is already computed and stored by the spatiotemporal filters or one of the preceding post-filters in the chain. Validation tests show that our internal versions produce the same results as the original CAOS-D filters run externally in series, but at up to 15% improved efficiencies.

5. RESULTS

The results presented in this section are executed on a 64-core Linux machine with 1TB shared memory. The same root-finder is used at the end of the conjunction analysis stage for all tests. The time required by the conjunction analysis stage, which includes the time dependent filter chain and root-finder, is provided to enable independent evaluation of the efficiency of the entire conjunction analysis process without the added variability due to the propagation and risk assessment stages.

The results shown in Table 1 use a standard VCM catalog, containing approximately 16,000 objects. The all-on-all conjunction analyses look for object pairs that have minimum approach distance of 10 km over a window of seven days. For reference, the propagation stage for generating ephemeris points every 60 seconds over the seven day window typically takes between 30 to 40 minutes.

Table 1. Test results processing ~16K VCM catalog for seven-day analysis window with 60 s time steps and 10 km threshold distance.

Algorithm	Pre-Filters	Post-Filters	Conjunctions Passed by Time Dependent Filter Phase	Conjunctions Passed by Root-Finder	CPU Time for Conjunction Analysis Stage (s)
Hybrid	None	External	1,092,891	276,487	513.396
Extremum-Checking Spatial Partition Tree	None	External	1,092,891	276,487	180.034
Extremum-Checking Spatial Partition Tree	Orbit-Sorting	External	1,092,891	276,487	172.65
Extremum-Checking Spatial Partition Tree	None	Internal	1,092,891	276,487	139.413
Extremum-Checking Spatial Partition Tree	Orbit-Sorting	Internal	1,092,891	276,487	155.857
Cut Bounding Box Spatial Hashing	None	External	1,103,887	276,487	363.600
Cut Bounding Box Spatial Hashing	Orbit-Sorting	External	1,103,887	276,487	402.159
Cut Bounding Box Spatial Hashing	None	Internal	1,103,887	276,487	308.906
Cut Bounding Box Spatial Hashing	Orbit-Sorting	Internal	1,103,887	276,487	367.079

All of the filter methods resulted in the same conjunction points found by the root-finder. The three methods effectively eliminated over 99% of all potential object pairings from requiring more expensive analysis. For this data set, EC is the most efficient among the three methods, capable of completing the conjunction analysis stage in a little over two minutes. The Spatial Hash method generated 1% more false positives than the other two methods, but still completed the analysis in approximately 60% of the time required by the baseline hybrid filter chain. For this

particular test, the orbit-sorting pre-filter provides minimal advantage when used with EC, and actually slows down the Spatial Hash method. The internal post-filters resulted in 15% to 20% speed-up of the conjunction analysis phase without introducing false negatives.

To demonstrate scalability of both spatiotemporal filters, we used a fictitious catalog developed by the MITRE Corporation consisting of pre-propagated ephemeris of approximately 100,000 objects. We randomly extracted subsets to create smaller catalogs of approximately 100, 1000, and 10,000 objects. Conjunction analysis is performed on each catalog over a one-day analysis window with 60 second time steps and 10 km threshold distance. For reference, the propagation stage involves loading the ephemeris points for the full catalog of 100,000 objects, which required less than 5 minutes.

Using orbit-sorting with EC to divide the catalog into three regimes provides a significant gain in computation time for this data set. Table 2 presents the results of the hybrid filter chain, EC with internal post-filters and orbit pre-sort, and CBB with internal post-filters without orbit pre-sort tested on the four catalogs. All three methods produced the same conjunction points. Again, all three methods effectively reduced the number of object pairings requiring further analysis by over 99%. Both EC and CBB scale well with catalog size. EC remains the most efficient method for the tested catalogs. However, Fig. 5 shows the Spatial Hash method to be more scalable than the Spatial Partition Tree algorithm, slowing down at a more gradual rate, possibly outperforming EC for much larger catalogs.

Table 2. Test results for one-day analysis window with 60 s time steps and 10 km threshold distance.

Algorithm	Pre-Filters	Catalog Size	Conjunctions Passed by Time Dependent Filter Phase	Conjunctions Passed by Root-Finder	CPU Time for Conjunction Analysis Stage (s)
Hybrid	None	100	4	0	0.487
Extremum-Checking Spatial Partition Tree	Orbit-Sorting	100	4	0	0.429
Cut Bounding Box Spatial Hashing	None	100	4	0	1.973
Hybrid	None	1,000	713	203	3.404
Extremum-Checking Spatial Partition Tree	Orbit-Sorting	1,000	713	203	0.730
Cut Bounding Box Spatial Hashing	None	1,000	720	203	4.497
Hybrid	None	10,000	80,836	20,602	34.671
Extremum-Checking Spatial Partition Tree	Orbit-Sorting	10,000	80,836	20,602	9.605
Cut Bounding Box Spatial Hashing	None	10,000	81,621	20,602	27.914
Hybrid	None	100,000	7,886,981	1,998,361	5899.688
Extremum-Checking Spatial Partition Tree	Orbit-Sorting	100,000	7,886,981	1,998,361	252.112
Cut Bounding Box Spatial Hashing	None	100,000	7,962,925	1,998,361	364.532

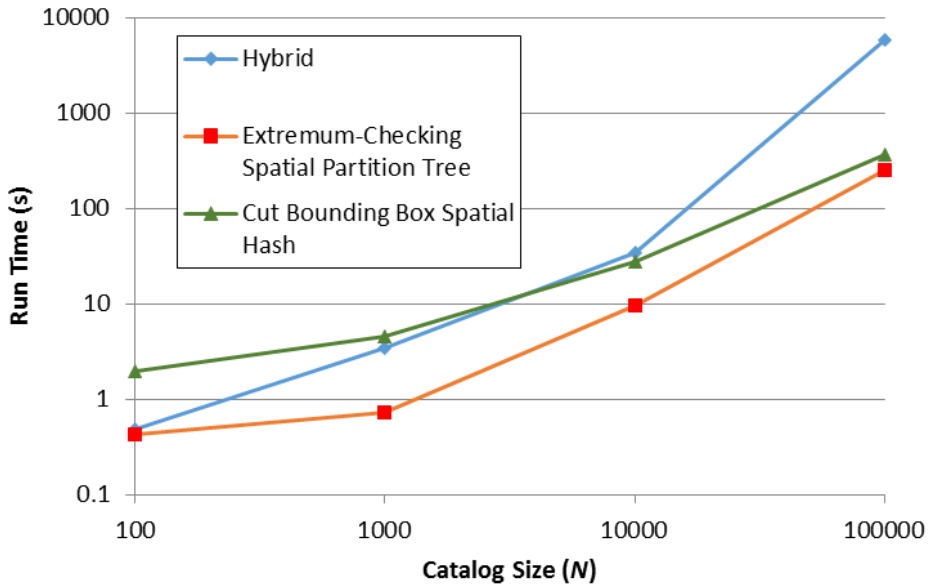

Fig. 5. Log-log plot of runtimes against the total number of objects in the catalog.

6. CONCLUSIONS

We introduced two primary filtering techniques we have developed for satellite conjunction processing using spatiotemporally indexed data structures. One is based on the *k*-d Tree pioneered in robotics applications and the other based on Spatial Hashing used in computer gaming and animation. We implemented these techniques as OSGi plug-ins for the CAOS-D conjunction analysis system. We also described pre- and post-filtering steps that have shown to further increase efficiency. We demonstrated that our filters can accurately produce the same conjunctions as the original CAOS-D hybrid filter chain, which has been validated against the industry-standard conjunction assessment tools, at up to twenty-fold increase in speed. Tests are planned to directly validate the performance of our filters against the industry-standard tools.

We demonstrated that, post-propagation, both filters are able to process a standard space catalog over a seven-day window in five minutes or less. Also post-propagation, both filters are capable of processing a large catalog of 100,000 objects to find conjunctions over a one-day analysis window in six minutes or less. We further demonstrated the scalability of our filter concepts, showing that these spatiotemporal filters are capable of handling both current and future growth in the Space Catalog to help JSpOC with their responsibilities to protect US and friendly assets, as well as protect astronauts during manned missions.

7. REFERENCES

[1] F. R. Hoots, L. L. Crawford and R. L. and Roehrich, "An Analytic Method to Determine Future Close Approaches Between Satellites," *Celestial Mechanics*, vol. 33, pp. 143-158, 1984.

[2] L. M. Healy, "Close Conjunction Detection on Parallel Computer," *Journal of Guidance, Control, and Dynamics*, Vols. 18, No. 4, pp. 824-829, July-August 1995.

[3] J. R. Alarcon-Rodriguez, F. M. Martinez-Fadrique and H. Klinkrad, "Collision Risk Assessment with a 'Smart Sieve' Method," in *Proceedings of Joint ESA-NASA Space-Flight Safety Conference*, 2002.

[4] B. Abernathy, S. Harvey, D. M. Surka and M. O'Connor, "The CAOS-D Architecture for Conjunction Analysis," in *Infotech@Aerospace 2011 Conference*, St. Louis, 2011.

[5] E. George and S. Harvey, "A Comparison of Satellite Conjunction Analysis Screening Tools," in *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, Wailea, 2011.

[6] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, pp. 509-517, 1975.

[7] E. J. Hastings, J. Mesit and R. K. Guha, "Optimization of Large-Scale, Real-Time Simulations by Spatial Hashing," in *Proceedings of the 2005 Summer Computer Simulation Conference*, 2005.

[8] F. P. Preparata and M. I. Shamos, Computational Geometry: An Introduction, New York: Springer-Verlag, 1985.

[9] A. W. Moore, "Efficient Memory-based Learning for Robot Control," Cambridge, UK, 1990.

[10] M. Teschner, B. Heidelberger, B. Muller, D. Pomeranets and M. Gross, "Optimized Spatial Hashing for Collision Detection of Deformable Objects," in *Proc. Vision, Modeling, Visualization '03*, Munich, 2003.