AAS 09-374

ASSESSING SATELLITE CONJUNCTIONS FOR THE ENTIRE SPACE CATALOG USING COTS MULTI-CORE PROCESSOR HARDWARE

Vincent T. Coppola,[*] Sylvain Dupont,[†] Kevin Ring,[‡] and Frank Stoner[§]

The recent collision of the Iridium 33 spacecraft with Cosmos 2251 debris has shown the importance of conducting conjunction assessments on a continual regular basis for operational spacecraft. We study the feasibility of assessing the entire space object catalog (i.e., all-on-all assessment) for 1 and 5 day analysis periods, using both low and high fidelity ephemerides, using COTS software and COTS multi-core processor hardware. We show that a catalog of 12,000 space objects (involving almost 72 million pairings) can be assessed within one hour and thus incorporated into an operational environment. The impact on the assessment of larger catalogs (e.g., 20K or 100K objects) will also be discussed.

INTRODUCTION

The number of near-Earth space objects being tracked in the US space catalog is well over 19,000 and continues to increase. As the recent collision of the Iridium 33 spacecraft and the Cosmos 2251 debris illustrates, identifying possible collisions between space objects is necessary for maintaining the viability of operational assets and for forecasting changes in the number of hazards present in the space environment. Prediction of these events in advance of 1 to 5 days is necessary to allow for further assessment and execution of any actions needed to mitigate risk to the affected spacecraft.

In general, the determination of the times when two space objects are close together is rather straight forward to understand, implement, and apply. However, hardware and software performance quickly becomes a consideration when attempting to perform this determination over an extended analysis period for every possible pair of objects from the entire space catalog. For a catalog of 12,000 objects, almost 72 million pairs must be evaluated; a catalog of 20,000 objects requires almost 200 million pairings; a catalog of 100,000 objects requires almost 5 billion pairings.

Software performance can by markedly improved by foregoing direct search methods on each and every pairing in favor of more selective algorithms which can quickly eliminate pairs of objects based on characteristics of orbital motion. Typically, a series of geometrical and temporal filters is used to quickly reduce the search space.[1,2,7] Robust root and extremum finding algorithms are then employed to isolate intervals of time when a pair of objects are within a specified

[*] Senior Astrodynamics Specialist, Analytical Graphics Inc., 220 Valley Creek Blvd, Exton, PA.
[†] Lead Software Architect, Analytical Graphics Inc., 220 Valley Creek Blvd, Exton, PA.
[‡] Lead Architect for AGI Components, Analytical Graphics Inc., 220 Valley Creek Blvd, Exton, PA.
[§] Senior Astrodynamics Specialist, Analytical Graphics Inc., 220 Valley Creek Blvd, Exton, PA.

proximity to one another and to determine precise times of closest approach. The use of filters greatly decreases the time necessary to determine all close approaches while maintaining accuracy.[7]

Improved performance is not limited to an improved algorithm, however. Because each pair of objects must be considered independently, the assessment of the entire catalog is very amenable to parallelization. The search space can be divided among multiple processing units to reduce the overall execution time required to obtain a solution. Historically, such processing requirements necessitated use of expensive or exotic hardware configurations.[3] Recently, multiple core/multiple processor personal computers have come into the commercial market which can perform the necessary computations. By marrying an improved algorithm with COTS multi-core/multi-processor hardware, the entire space catalog can be assessed in less than one hour---a time scale amenable for use operationally.

This paper presents results for the all-on-all conjunction assessment of ~12,000 objects over both a 1 day and a 5 day analysis period. Two methods for obtaining the state information for the space objects are considered. The first method utilizes analytical propagation of the orbit state from known elements at a given epoch (i.e., using SGP4 with TLEs). Analytical propagation reduces the computer memory footprint of each object at the expense of reduced accuracy of the ephemerides themselves. This has been the most common mechanism for assessing the entire catalog. The second method models the path of each space object using a table of ephemeris coupled with an interpolation method. The accuracy of the ephemeris can be made quite high using this scheme; however, the memory footprint per object is greatly increased.

Three different software implementations will be used. The first software implementation is a general purpose Commercial Off-The-Shelf (COTS) 32-bit application which has, as part of its feature set, the ability to perform conjunction analysis between space objects. The second software implementation builds on the first by splitting the computations into groups and running the first implementation for each group on different processors. Results are then assembled at the end. The third software implementation is a custom application designed specifically to perform conjunction analysis between space objects, but built using a set of COTS software components. It can run natively on 32 and 64-bit architectures. A 32-bit process on the Microsoft Windows operating system is limited to a 3 gigabyte address space. No matter how much RAM is physically installed in the system, the application will not be able to allocate more than 3 gigabytes. In contrast, an application running as a 64-bit process on a 64-bit version of Windows has no such address space limitation and can address terabytes (or more). The extra address space, combined with cheap RAM, enables an application that is written to run as a 64-bit process to be both faster and simpler than an otherwise equivalent 32-bit application.

Performance and consistency of the determination will be presented using 2 hardware configurations: (i) a single 32-bit 4Gb RAM dual-core COTS personal computer; and (ii) a single 64-bit 32Gb RAM 4-core dual processor COTS personal computer. Each will be running on a commonly available Windows operating system.

THE SPACE OBJECT CATALOG

One of USSTRATCOM's missions is the tracking and cataloging of space objects. Some of the catalog is made available to the public while some information is restricted. As of Feb 2009, the public catalog consisted of about 12,000 space objects. Our testing will use the public catalog from 11 Feb 2009 that consisted of 11,970 space objects.

The conjunction assessment for the entire space catalog requires conjunction determinations for each unique pairing amongst all the objects. The number of unique pairings can be computed

as N*(N-1)/2 where N is the size of the catalog. For N > 10,000, the linear term in the formula contributes less than 0.01% so N^2/2 provides the correct order estimate for the pairings to consider. For 12,000 objects, this would be about 72 million pairings that must be considered.*

While each pairing must be considered, it is not necessary to compute close approaches for every pairing. We are interested in performing conjunction assessment to determine when objects are close to one another—objects that are never close to each other are of no interest. Thus, there's no computation needed for the assessment of a LEO object with a GEO object; once the regimes for the two objects have been determined, no further consideration for this pairing is necessary.

Space Object Ephemerides

The public catalog consists of TLEs (two-line elements) that contain state information for use with SGP4[4,5], an analytical propagator---one TLE per space object. The resulting ephemerides are not often high fidelity ephemerides (i.e., they differ from truth by more than tens of meters). In fact, the quality of the ephemerides produced using TLEs is not known. It is commonly thought that the largest difference from truth will be on the order of a few kilometers, yet there are cases where the difference is much worse (30-100s of km). In such cases, the large differences are not a fault of the SGP4 modeling capability; instead, they represent other issues (e.g. misidentifying measurements, maneuvering spacecraft)[6].

Given the nature of the TLE-based ephemerides, it's difficult to rely on their use for conjunction assessment purposes unless a large threshold of interest is used. Ideally, a catalog consisting of high fidelity ephemerides (or lacking that, state information and force modeling parameters for performing a numerical integration to produce high fidelity ephemerides) would be available for use. While USSTRATCOM has such a catalog, it is currently not available to the public.

Our interest is the processing of conjunctions assessments for the entire space catalog. The best conjunction assessment is made using the highest fidelity ephemeris data available. Assessments using TLEs can be suspect because of the lower quality ephemeris. In contrast, a table of ephemeris is the most natural mechanism for representing high fidelity data. Ephemeris tables inject a new consideration into the processing methodology because of the large memory footprint required for their use.

Of course, while we want to use ephemeris tables, we have no access to that data from USSTRATCOM. We are also not able to generate our own space catalog because the space observation measurements, (upon which TLEs and those ephemeris tables are generated) are not made available to the public either. Lacking a high fidelity ephemerides source, we cannot assess the accuracy of our analysis compared with truth data. However, we can assess the impact of using ephemeris tables on the solution methodology and expected computational performance, as compared to using TLEs.

Thus, we need only a space object catalog consisting of a table of ephemeris for each space object—whether those ephemerides are high fidelity or not is immaterial for our purposes. The simplest choice for generating these ephemerides, then, is simply to use SGP4 with each of the TLEs of the public catalog to create an ephemeris table for each object. The processing then needs to deal with the ephemeris tables directly, rather than using any TLE information whatsoever for an object. While the end results will not speak to accuracy, they will speak to perform-

* The actual number of pairings for 12,000 objects is 71,994,000; the actual number for 11,970 objects is 71,634,465.

ance so that organizations that do have access to the USSTRATCOM ephemeris data have a basis for understanding the operational implication of assessing the catalog using high fidelity data.

We will store each ephemeris table as its own ephemeris file and perform the conjunction assessment using the ephemeris files. An advantage of our approach is that we can compare directly the use of TLEs versus the use of ephemeris tables on performance (i.e., time to complete the task) and consistency (i.e., obtaining the same results).

Ephemeris Interpolation

While SGP4 can be used to compute an ephemeris value at any requested time, an ephemeris table must be interpolated to find values at times off its grid. Some conjunction assessment strategies avoid interpolation entirely by requiring all ephemerides for all objects to have the same time grid.[3] This complicates the preparation of the ephemerides and forces all objects to use the smallest time step needed for analysis, bloating the required memory footprint. Our approach instead relies on interpolation of the ephemeris to produce an accurate ephemeris sample when off the time grid. This can be accomplished through a judicious choice of time step when creating each ephemeris table (where the choice of time step for each object can be made independently of the knowledge of any other object).

It is important that each ephemeris table be created so that interpolation of the table differs little from the value that would have been computed using SGP4 with the TLE directly. Typically, 90 points per orbit is needed for adequate interpolation of nearly circular orbits. For eccentric orbits, it is critical to have enough samples near perigee to do accurate interpolation. A general rule is to use a step size that would produce 90 points per orbit for a circular orbit at the same perigee as the eccentric orbit. We also set a maximum step of 300 sec. Thus, LEOs generally use 60 sec steps while GEOs typically use 300 sec steps.

Each ephemeris table contains a listing of time and 3 Cartesian values each of position and velocity for a total of 7 values per time step. Each value is represented in the computer as a double that contains 8 bytes; hence, there are 56 bytes per time step.

Assessing Larger Catalogs

The number of unique pairings of objects that must be considered scales as N^2/2. The memory footprint, however, scales linearly with N. Pairing and memory estimates for larger catalog sizes are given in Table 1.

Table 1. Computation size as a function of catalog size N.

Catalog Size N	Pairings	Memory Footprint	
		1-day Ephemeris (Gigabytes)	5-day Ephemeris (Gigabytes)
12,000	72 million	0.87	4.3
20,000	200 million	1.4	7.2
50,000	1.25 billion	3.6	18
100,000	5 billion	7.2	36

The 20,000 object catalog is 1.66 times larger than the 12,000 object catalog and yet requires 2.78 times as many pairings to consider. The 100,000 object catalog is 8.33 times the size of the 12,000 object catalog but requires 69.4 times as many pairings.

If the distribution of objects in the 12,000 object catalog over different orbit regimes is the same as in the larger catalogs, then one could use the number of pairings as a simple estimate of the time needed to compute the larger catalogs. Thus, the 100,000 object catalog would require 69 times as long as the 12,000 object catalog. However, it is expected that many more LEO space objects will exist in the larger catalogs. LEO objects take more computation time to assess than other objects, so the simple estimate is probably on the low side---the computation time should be expected to be even longer.

CONJUNCTION ASSESSMENT ALGORITHM

Different organizations use different metrics for assessing conjunctions. Range, radial separation, maximum probability, and true probability are commonly used. In this paper, a pairing of objects will be declared to have a conjunction whenever the range between them is less than some specified threshold. Our focus is the assessment of the entire catalog, not on weighing the relative merits of different metrics. While the other metrics may have merits, their computation is not much more complicated than computing the range at the time of closest approach and thus has little impact on the ability to assess the entire catalog.

Detecting Conjunctions

In most cases, conjunctions occur over very short time intervals, usually measured in seconds, while the assessment is performed over long time intervals, typically days. A simplistic approach to finding these small time intervals within the larger analysis intervals is to sample the range between the objects over the larger analysis interval using a small time step and detect which samples were below the specified threshold. To detect conjunctions whose duration was at least 1 second long, the approach would take 86,400 samples over each day. This simplistic approach using fine time step sampling does indeed work but is computationally expensive. Since the conjunction durations are often small, most all the samples produce a range outside the threshold and are not of value.

Better approaches use far less samples while still detecting the conjunction intervals. Our approach combines two strategies: (i) using a series of geometric filters to quickly determine times when a pairing cannot possibly have a conjunction; and (ii) searching for conjunctions using coarse time stepping coupled with numerical root finding techniques to reduce the number of samples while precisely determining conjunction times.

Conjunction Filters

The idea of conjunctions filters goes back at least as far as Hoots et al[1]. Before any computationally intensive search for conjunctions is performed, objects are first passed through a series of tests to winnow the large analysis interval into a set of much smaller time intervals that need to be investigated by a search algorithm.

The first filter is the Apogee/Perigee filter. The apogee and perigee for each object is determined over the analysis interval. The radius value for an object will lie within the range from perigee to apogee over the entire analysis interval. If the two range intervals do not come within the specified threshold, then the objects themselves can never be closer than the threshold, and there can be no conjunctions. This filter quickly determines that LEOs and GEOs will never have a conjunction and no further processing is done.

The second filter is the Orbit Path filter. When the two space objects are orbital, one can associate an elliptic orbit path with each object using Keplerian elements for each object evaluated at some time during the analysis interval.* One can then compute the minimum distance between the ellipses in 3D space using a numerical minimization algorithm. If the minimum distance is larger than the specified threshold, then the objects cannot have a conjunction at any time and no further processing is done.

The third and last filter is the Time filter. Building upon the geometry of the Path filter, information about where each object is in its elliptical motion is now incorporated. For each object, time intervals are determined wherein the object is within the specified threshold of the other object's orbital plane.† These sets of intervals for each satellite of the pair are then intersected, producing time intervals in which each object is within the specified threshold of the other's orbital plane. If no intervals survive because of the phasing the two objects' motion, then there can be no conjunctions and no further processing is performed. However, if time intervals do exist, then any conjunctions must occur within these intervals. Only these intervals need to then be searched for conjunctions. Typically, these intervals are small and represent a very small portion of the original analysis interval.

The actual implementation of the filters is more complicated than described by Hoots et al.[1] Woodburn et al.[7] document some deficiencies in the original implementations and described improvements that avoid incorrect filtering. One simple improvement noted is the addition of extra padding beyond the threshold value itself when determining whether to filter out an object from further consideration. Moreover, the improvements adopt a 'trust but verify' approach to filtering where assumptions made by the filters are checked for their validity before accepting their conclusion.

The software being used in this paper incorporates improvements to the Apogee/Perigee and Time filters. The Orbit Path filter, however, will not be used in this paper to generate results.

Event Detection

Once a set of intervals to investigate has been determined from the filters, each must be searched to determine whether conjunctions exist. The search routine uses coarse sampling coupled with event detection that builds upon a robust numerical root finding routine and numerical extremization routine. Samples are taken not for the purpose of identifying a conjunction but rather to identify the trend in the value of range over time. They do not require sampling to occur only on grid points of an ephemeris table but rather rely on interpolation of the table to produce accurate sampling.

The event detection routine monitors when the samples cross the range threshold. It also investigates whenever the trend changes between increasing and decreasing (and vice-versa), invoking the extremization routine to isolate the range at its minimum. The local extreme value is then used to detect times when the range value has crossed the specified threshold. Once a crossing has been detected, numerical root finding is used to accurately and precisely compute the time at which a crossing occurred. Crossing times are then assembled into conjunction intervals.

---

* In rare instances an object may be hyperbolic; in few instances objects may be sub-orbital. In such situations neither the orbit path nor time filtering is applied: the entire analysis interval must be searched for conjunctions.

† In cases where the relative inclination is small (i.e., in nearly coplanar orbits), the time filter is not applied: the entire analysis interval must be searched for conjunctions.

This strategy achieves fine time accuracy with coarse time step sampling: it is computational fast and yet achieves all required accuracy.

METHODOLOGY: ASSESSING THE ENTIRE CATALOG

Assessing the entire space catalog amounts to performing an investigation for each unique paring of objects in the catalog – resulting in millions of independent assessments. The computation is very amenable to parallelization. The assessment can easily be split up into pieces and distributed to different processors on a group of machines. The entire assessment can be performed as quickly as the hardware resources will allow.

The only data required to perform the computation is an ephemeris for each object. If TLEs are being used, then all TLEs can be loaded into memory quite readily and then SGP4 can be used to compute ephemeris at any requested time. If ephemeris files are being used, however, then these files must be read from a hard disk into memory. One should avoid reading from the hard disk as much as possible because it is slow. A 64-bit Windows application has the ability to simply read all the ephemerides into memory, up to the hardware's capacity, and avoid reading ephemeris files more than once. 32-bit Windows applications, however, are memory constrained---reading all files into memory only works for smaller problems.

If multiple machines are being used, then networking speed may affect performance depending on the amount of information being passed. Serving out ephemeris over a network from one machine can affect performance depending on how often files are being requested during computations. We found it beneficial for each machine to have its own local copy of ephemeris files rather than relying on a single server.

TEST CONFIGURATION

Hardware

We tested on several COTS hardware configurations, available from Dell. *CATMachine* is a Dell Precision T3400 Intel Core 2 Duo 3Ghz E6850 32-bit machine with 4 Gigabytes RAM and 2 processors running Windows Vista SP1. This is a high-end box suitable for an engineer's office and costs around $2,000. *MadCATMachine* is a Dell Precision T5400 Intel 2x4 Core XEON X5450 3 Ghz 64-bit machine with 32 Gigabytes RAM and 8 processors running Windows XP SP2. This is a high-end compute server and costs around $3,500. It has 10,000 RPM hard disk drives for even faster disk access than the standard 7200 RPM drives.

We also tested on a computer farm involving 5 dual-core Dell 32-bit machines with 2 Gigabytes RAM networked together with one machine serving as an ephemeris server. While we have preliminary results showing superior performance over *CATMachine*, the performance of *MadCATMachine* was so superior we did not continue testing with the computer farm. We fully expect that adding more hardware resources can improve performance, but our goal was to show that the entire catalog can be assessed on readily available hardware and software and the two machines satisfy these requirements.

Software

We used 3 different software solutions to perform the conjunction assessment: STKEngine, ParallelCAT, and FastCAT.

STK Engine. STK Engine (version 8.1.3) is a member of Analytical Graphics' STK product suite that incorporates STK's computational capabilities without its user interface. While it can support both 2D and 3D graphics, neither graphics windows were used. STK Engine is a single-

threaded 32-bit application that has the ability to perform conjunction assessment using STK/CAT for one group of objects against another group of objects, using either TLEs or ephemeris files, upon purchase of a Conjunction Analysis Extension license. It uses conjunction filters, efficient trend sampling, and event detection to find conjunctions. The standard cost including one year of support is around $19,250.

STK/CAT was designed to perform conjunction assessment of a group of primary objects against a group of secondary objects. Normally, an object would not appear in both lists. However, when assessing the entire catalog, it is natural to have the same object appear in both lists. STK/CAT computes correctly in that case too; however, it treats the object as two separate entities. The consequence of this is that the ephemeris for the object is loaded twice; hence, the memory footprint for assessing the catalog is doubled.

Being a 32-bit application, STK Engine is limited by Windows to 3 Gigabytes RAM. [Normally, the limit is 2 Gigabytes RAM unless special operating system settings are enabled.] This memory constraint precluded the ability to assess the entire catalog using ephemeris files. Moreover, STK Engine uses the resources of only 1 processor because it is not multi-threaded, leaving the other processor available for other activities (or idle).

ParallelCAT. ParallelCAT is a 32-bit application server program that was written to perform the catalog assessment by partitioning it into groups of a size that can be computed using multiple instances of the STK Engine operating in parallel. The server organizes the computations into groups, instantiates STK Engine on different processors (on 1 or more machines), requests each STK Engine to compute a group of the computation, and then assembles the results. While written as a .NET application, it simply acts as a sophisticated batch file running a set of groups simultaneously. ParallelCAT can make use of every processor on the machine simultaneously to achieve the fastest computation possible using STK Engine as the compute engine.

The object catalog can be divided into P groups of nearly equal size M. We then perform the assessment on each unique pair of groups and combine the results together. This is represented pictorially in Figure 1 where the catalog has been divided into 5 groups (A, B, C, D and E). Only the lower triangle is computed since each pairing located in the upper triangle is already represented by a pairing in the lower triangle.

Heterogeneous pairings (B-A, C-E, D-B, etc.) are represented by squares: they require M^2 assessments. Homogeneous pairings (A-A, B-B, etc.) are represented as triangles: they require only M*(M-1)/2 assessments (i.e., the number of unique pairings). The total number of 'square' tasks is P*(P-1)/2 (which for P = 5 is 10) and the number of 'triangle' tasks is P; the total number of tasks is P*(P+1)/2. The triangle tasks are expected to require roughly half the time of a 'square' task.

The most efficient computation strategy is to use all available processors 100% of the time, with each ending at the same time. However, it is not known ahead of time how long any task may take. Some tasks will inevitably contain more conjunctions than others and thus require longer to process. Some load balancing can be done up front using orbit class, though we did not implement such a strategy. The more tasks being performed the more the up front costs, associated with the starting of a task, contribute to the total time. Too few tasks, however, and the memory requirement may exceed the limitations set by the 32-bit operating system and thus overwhelm STK Engine.

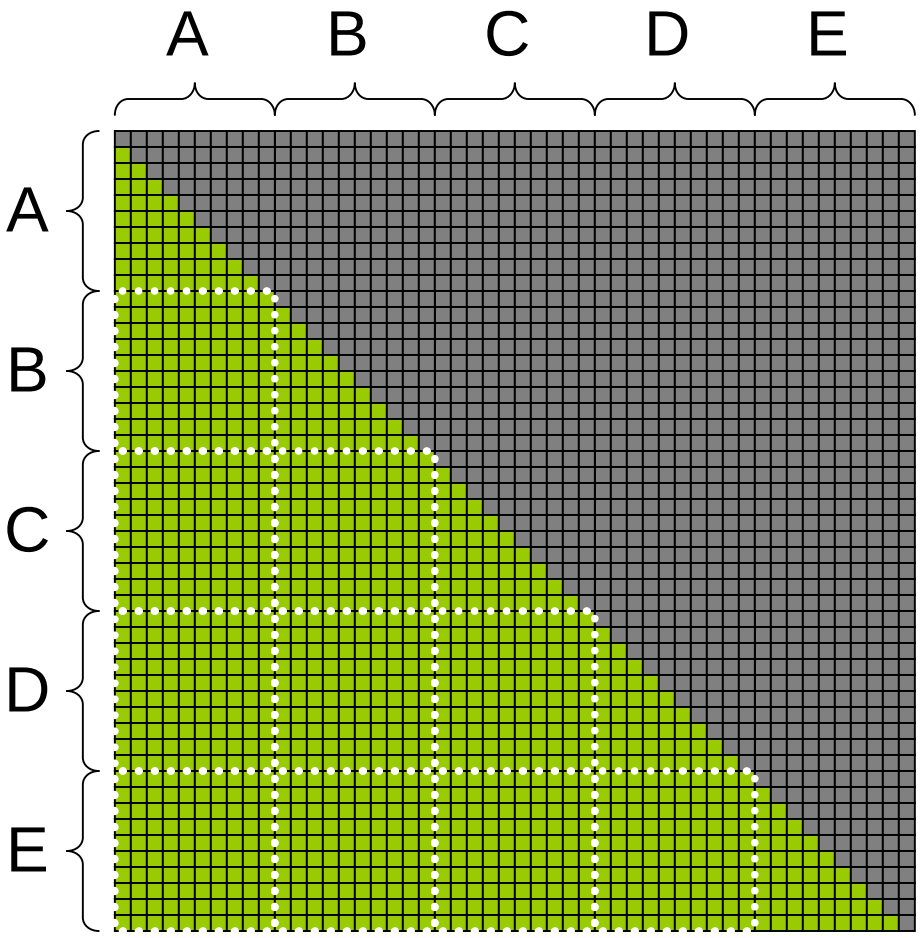

Figure 1. Graphical depiction of the partition strategy. The assessment is done only for the pairs represented in the lower triangle since every pairing in the upper triangle is already represented in the lower triangle.

When the number of tasks is neither too few, nor too many, the driving force for the total computation time is how many processors are being used near the end of the computation when no new tasks are being requested. When only a small number of the available processors are being used near the end, then it may have been possible to rearrange the tasking to more effectively use the processing resources. We found that the best strategy was to assign 'square' tasks first and perform the quicker 'triangle' tasks at the end. This helps to use all available processors longer.

FastCAT. FastCAT is a 64-bit single-machine application written using AGI Components. It is a multi-threaded application that makes use of all processors available. Because the *MadCAT-Machine* has enough memory, all ephemerides are loaded into memory simultaneously.

The FastCAT application also uses the Apogee/Perigee and Time filters, smart sampling, and event detection to precisely find conjunction times. The filter implementations are not directly supported by AGI Components (as of the date of this writing). However, the algorithms neces-

sary for processing TLEs, SGP4 propagation, root and extremum finding, element set conversion, etc., are available. This made implementation of the filters relatively straight forward.

FastCAT takes the most direct approach to assessing the entire catalog. The application loops through every pairing, assigning its computation to a thread and collects together the results as each thread completes. It can also show a 3D graphical display of its progress to satisfy an operator's curiosity, though it lengthens the computation time somewhat. The standard cost for the software developer kit with 1 runtime license is $6,650.

RESULTS

The test data consisted of the public catalog from 11 Feb 2009 involving 11,970 objects. The analysis start time was 12 Feb 2009 05:00:00 UTC and durations of 1 and 5 days were investigated. Both the Apogeee/Perigee filter and the Time filters were used, with extra padding of 30 km; the Orbit Path filter was not used. The range threshold was set at 5 km. The total number of conjunctions reported for the 1-day analysis is 5338 and for the 5-day analysis is 26933.

Figure 2 shows the distribution of the minimum range at conjunction for the 1-day analysis results. The closest conjunction was at 84 meters. The distribution shown in Figure 2 is typical for the catalog---the test data is not special in any way.

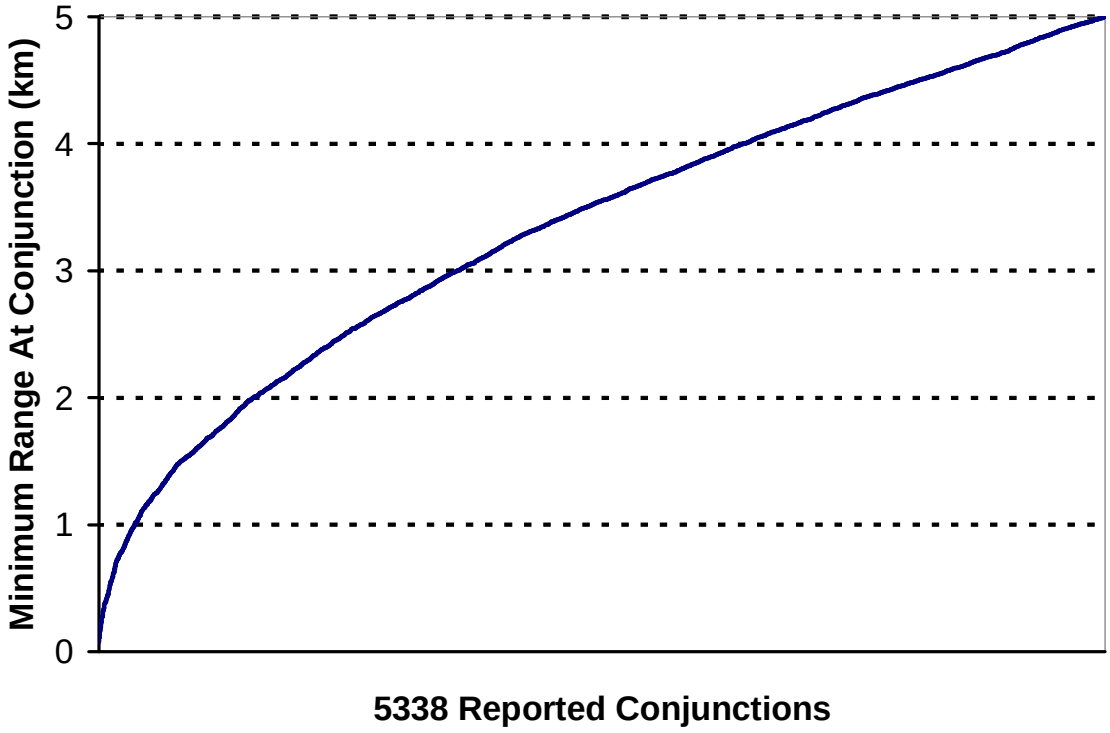

Figure 2. Distribution of minimum range at conjunction for all 5338 reported conjunctions.

STK Engine

The first tests used TLEs with SGP4 to generate ephemeris rather than ephemeris files. The use of filters was exercised to assess the impact on performance and accuracy. Table 2 shows that the computation speed is greatly enhanced when using the filters and that the use of the filters does not degrade the accuracy of the results.

Table 2. Computation time for STK Engine running on 1 processor on CATMachine, for a one day analysis using TLEs. Each analysis found the same 5,338 conjunctions.

Apo/Peri Filter	Time Filter	Compute Time (mins)
N	N	1716
Y	N	747
Y	Y	101

ParallelCAT

CATMachine. Table 3 shows an improvement in computation speed when using 2 processors rather than just 1, even when using ephemeris files rather than TLEs. Both the Apogee/Perigee and Time filters were used. The results for each of the 3 runs were identical and differed from the runs using TLEs by round-off noise.* The results demonstrate that the computation using ephemeris files is accurate and that ParallelCAT's group strategy works correctly as well. Still, on the CATMachine computer, it takes a little over an hour to assess the entire catalog for one day using ephemeris files.

Table 3. Computation time for ParallelCAT running on 2 processors on CATMachine, for a one day analysis using ascii ephemeris files.

Group size	Groups	Tasks	Compute Time (mins)
4000	3	6	64.8
2000	6	21	65.3
1500	8	36	71.5

MadCATMachine. Table 4 shows the results for both the 1 day and 5 day analyses, using 8 processors on MadCATMachine. The 1-day results match the baseline results to within round-off noise. Since a significant amount of time is needed for loading ephemeris files, we also investigated using ascii and binary formatted ephemeris files. The use of the binary format was a substantial improvement because of the manner in which STK Engine loads files.

* Output results for each conjunction include the identification numbers for each object, the start time and stop time of the conjunction, the time of closest approach, and the minimum range. Times were rounded to the nearest milli-second; the range was rounded to 0.1 meters.

Table 4. Computation time for ParallelCAT running on 8 processors on MadCATMachine, Both the Apogee/Perigee and Time filters were used.

Analysis duration (days)	Group Size	Tasks	Ephemeris Type	Compute Time (mins)
1	3000	10	ASCII	19.5
1	2000	21	ASCII	17.8
1	1000	78	ASCII	17.9
1	1500	36	ASCII	17.3
1	1500	36	Binary	13.2
5	1500	36	ASCII	72.8
5	1500	36	Binary	51.5

Since MadCATMachine has 8 processors, a group size of 4000 is not very effective because it uses only 6 tasks and thus 6 processors. The group size of 3000 is also not very efficient because after 8 tasks are complete there is only 2 to do leaving 6 processors not in use. Group sizes of 2000 and 1000 do better but still leave the possibility of idle processors. The group size of 1500, however, uses 36 tasks which divide evenly onto the 8 processors: this achieved the best performance. Note that the up front costs associated with performing a task (e.g., reading ephemeris files) can be significant, explaining why the times do not scale linearly.

Of note is that the entire space catalog of nearly 12,000 objects, modeled using binary ephemeris files, can be assessed for a 5-day duration in less than 1 hour on 1 inexpensive commercially available machine. Moreover, after setting the input parameters (e.g., start time, duration, group size, name of results file), no further user involvement is required: the entire activity could be run as an automated process.

FastCAT

Table 5 shows the results for both the 1 day and 5 day analyses, using 8 processors on the MadCATMachine. Both the 1-day and 5-day results match the ParallelCAT results to within 2 milliseconds. Since loading of ephemeris into memory occurs up front only one time, it is not as significant a contributor to the overall computation time: the use of binary ephemerides is expected to diminish the computation time by just a few percent.

Table 5. Computation time for FastCAT running on 8 processors on MadCATMachine, Both the Apogee/Perigee and Time filters were used.

Analysis duration (days)	Ephemeris Type	Compute Time (mins)
1	ASCII	12.2
5	ASCII	44.4

CONCLUSION

We show that a catalog of 12,000 space objects (involving almost 72 million pairings) can be assessed within one hour and thus incorporated into an operational environment. Moreover, the computational hardware necessary to achieve this level of performance is available commercially for about ~$3.5K. Certain considerations must be given to the software implementation in order to attain desired response time.

Certainly, the use of geometric and temporal filters greatly reduces the processing time over that resulting from direct search alone without degrading the accuracy of the results. The use of tabular ephemeris has additional advantages in that it may be of higher accuracy and may account for alterations to the orbit due to orbit maintenance, momentum unloading, etc.

For a long analysis time interval, the memory footprint required by the ephemerides may exceed the memory made available by the operating system. This can be addressed by partitioning the object catalog and distributing each group to a separate processor. Assuming that the objects in the catalog are not ordered with respect to orbital characteristics, choosing a group size which results in a number of tasks that is a multiple of the number of processors provides a simple partitioning scheme that is effective for load balancing across the available processors. We found that the strategy of analyzing the 'square' tasks first and the quicker 'triangle' tasks at the end helps to use all available processors longer.

Alternatively, an application running as a 64-bit process on a 64-bit version of Windows can address terabytes of memory and can readily load all ephemerides into memory. Since partitioning the catalog is not necessary, this type of implementation can be both faster and simpler than an otherwise equivalent 32-bit application.

REFERENCES

[1] Hoots, F.R., Crawford, L.L., and Roehrich, R.L., "An Analytical Method to Determine Future Close Approaches Between Satellites," *Advances in the Astronautical Scienc*es, Vol. 54, Pt. 1, 1984, pp. 281-298.

[2] Woodburn, J. and Dichmann, D. "Determination of Close Approaches for Constellations of Satellites," Paper No. C-5, IAF International Workshop on Mission Design and Implementation of Satellite Constellations, Toulouse, France, November 1997.

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

[4] Hoots, F.R. and Roehrich, R.L., "Spacetrack Report #3: Models and Propagation of the NORAD Element Sets", U.S. Air Force Aerospace Defense Command, Colorado Springs, CO, 1980.

[5] Vallado, D.A., Crawford, P., Hujsak, R., Kelso, T.S., "Revisiting Spacetrack Report #3", AIAA 2006-6753, 2006 AIAA/AAS Astrodynamics Specialists Conference, Keystone, CO, August 2006.

[6] Kelso, T.S., Vallado, D.A., Chan, J. Buckwalter, B., "Improved Conjunction Analysis via Collaborative Space Situational Awareness", 2008 AMOS Conference, Maui, Hawaii, September 2008.

[7] Woodburn, J., Coppola, V., Stoner, F., "A Description of Filters for Minimizing the Time Required for Orbital Conjunction Computations", AAS 09-372, AAS/AIAA Astrodynamics Conference, Pittsburgh, PA, August 2009