

A POWERFUL, COST EFFECTIVE, WEB BASED ENGINEERING SOLUTION SUPPORTING CONJUNCTION DETECTION AND VISUAL ANALYSIS

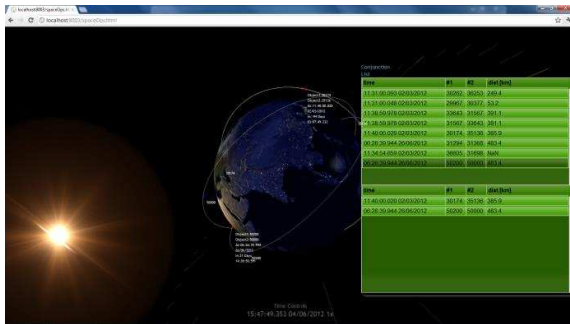
Daniel M. Novak⁽¹⁾, Davide Biamonti⁽²⁾, Jeremy Gross⁽²⁾, Martin Milnes⁽²⁾

⁽¹⁾ CGI, Rheinstrasse 95, 64295 Darmstadt, Germany, Email: daniel.m.novak@cgi.com

⁽²⁾ CGI, Rheinstrasse 95, 64295 Darmstadt, Germany, Email: {first name}.{last name}@cgi.com

ABSTRACT

An innovative and visually appealing tool is presented for efficient all-vs-all conjunction analysis on a large catalogue of objects. The conjunction detection uses a nearest neighbour search algorithm, based on spatial binning and identification of pairs of objects in adjacent bins. This results in the fastest all vs all filtering the authors are aware of. The tool is constructed on a server-client architecture, where the server broadcasts to the client the conjunction data and ephemerides, while the client supports the user interface through a modern browser, without plug-in. In order to make the tool flexible and maintainable, Java software technologies were used on the server side, including Spring, Camel, ActiveMQ and CometD. The user interface and visualisation are based on the latest web technologies: HTML5, WebGL, THREE.js. Importance has been given on the ergonomics and visual appeal of the software. In fact certain design concepts have been borrowed from the gaming industry.



1 INTRODUCTION

Debris orbiting the Earth have been recognised as a growing hazard to missions' safety and conjunction prediction is therefore becoming ever more important to ensure safe operations of spacecraft [1]. Tools and services covering that already exist [2][3][4], but they have limitations like late warning times, difficult maintenance due to out-dated software technologies, not

always ergonomic graphical interfaces or restrictive IPRs.

Moreover, the performance of current codes performing all-on-all analysis on large catalogues is limited by bottlenecks at the initial filtering for potentially colliding object pairs [5]. As the population of registered debris will increase the issue of computational efficiency will gain more attention. A tool to perform conjunction prediction is presented, with the capability of performing analysis between large sets very efficiently. Innovations have been implemented at multiple levels: within the underlying mathematical algorithms, the software architecture and the user interface.

The filtering algorithm implemented in the tool is based on a nearest neighbour search technique used in various fields like data compression and computational genomics. Building on our experience in the implementations of real-time operational scientific algorithms, we have performed detailed performance analyses of a range of alternative filtering strategies, to identify potential processing bottlenecks. This has led to the development of extremely efficient filtering techniques (including the use of spatial binning as a means of providing an effective first-level filter). A significant improvement in computing time has been achieved compared to existing tools like CRASS or Closeap.

A client-server architecture has been chosen for the tool, with the potential of parallelising the number crunching computations. The propagations, the filtering and the probability computations run on the server side, using updated catalogue data, in this instance TLE data. Usability and ease of access were identified as paramount to ensure that the scientific data can be understood and exploited by the majority of professional operators and amateurs. Latest web technologies, using graphical acceleration with HTML5 and WebGL and streaming, were adopted so that it can be run smoothly from a web browser, without the need to install a plug-in, while ergonomics and visualisation design concepts were borrowed from the gaming industry in order to optimize the interaction of the end-user with the functionalities offered by the tool and with the vast amount of data it relies on. Such a tool has big

potential, due to its performance, its ease to expand with more accurate models and functionalities, its ease to operate and maintain and its potential accessibility to the end user through any device that has a modern browser and a decent graphics card. The architecture of the tool allows easy plug-in of scientific models and data sources such as instruments, proving the tool's potential applicability to various space situational awareness applications.

The filtering algorithm is presented first, explaining how the conjunctions are detected and why the method is fast. Then the architecture and technology stack of the tool is presented. Finally the user interface and the underlying technologies are covered.

2 FILTERING ALGORITHMS

This section describes the algorithm to identify pairs of objects that come close to each other in a certain time window. The term close can potentially be defined by the user in different forms: miss distance below a threshold, presence of another object inside a pizza box or an ellipsoid around the target object.

2.1 Initial filtering in existing conjunction prediction tools

The role of the conjunction detection functionality is to identify close approaches between pairs of objects in the catalogue. The challenge is to limit the work load due to comparisons based on large amounts of data.

An assessment has been made of information available in the public domain relating to various approaches adopted for filtering in the all-vs-all case for conjunction prediction. This included identification of the filtering approach used and the overall performance, and a comparison with alternative schemes considered and prototyped by CGI. The previous approaches that were looked at include SOCRATES from CSSI [2], Closeap from GMV [3] and CRASS from ESA. Tab. 1 summarises the findings and the properties of the filters of the tools.

It should be noted that CRASS was not intended originally to be used for all-vs-all conjunction detection, so its performance illustrates that CRASS's method is not well adapted to the problem. The latest version of Closeap may also show some performance improvements.

2.2 Nearest neighbor search

Building on our experience in the implementations of real-time operational scientific algorithms, we have performed detailed performance analyses of a range of alternative filtering strategies, to identify potential processing bottlenecks. Our analysis of the bottlenecks

in the conjunction analysis processing (using gprof on the software running under Linux) highlighted that the majority of the processing time was originally taken up performing the initial comparison of pairs of objects (looking for close conjunctions). Clearly, this is of the order of N^2 , where N is the number of objects (but can be halved by eliminating comparing object A and B and then comparing B and A), however, with 13,000 objects, this gives around 170 million 3D position pair comparisons, which requires substantial processing power. The aim of the initial filtering is to reduce the number of necessary comparisons as quickly as possible, so that minimal processing power is expended on impossible pairs of objects. We also addressed alternative techniques used in other areas of computing relating to "nearest neighbour searches". A novel technique is proposed here to filter out objects efficiently such that conjunctions are identified very fast.

Table 1. Summary of existing filtering methods.

System	Filtering Methods	Performance
SOCRATES	Out of date TLE Apogee/perigee Orbit path Time	2,700 payloads vs 8,500 objects = 5 hours on 2 GHz Pentium 4 for 7 days prediction
Closeap (GMV)	Out of date TLE Remove decaying/decayed objects Apogee/perigee filter –on pairs of objects Then 3 consecutive filters on pair relative position/velocity: - Smart sieve (a series of filters based on very simple astrodynamics principles) - Fine conjunction - Safety ellipsoid	8,000 objects all-vs-all: 4.7 min using 1CPU @2.66 GHz for 1 day prediction
CRASS (ESA)	"smart sieve", based on maximum velocity & acceleration	8,000 objects all-vs-all: 121.1 min using 1CPU @2.66 GHz for 1 day prediction

The identification of close approaches is actually an example of a well-established technique. In fact nearest neighbor search, also known as proximity search, similarity search or closest point search, is an optimization problem for finding closest points in metric spaces. Examples are:

- Naïve approach, i.e search all points with successive spatial filtering
- Space partitioning, e.g. kd-tree
- Locality Sensitive Hashing, i.e. putting data in “buckets”/bins

CGI has explored all three approaches – whilst the spatial partitioning approach, using kd-trees, looked promising, in CGI’s prototype it did not perform as fast as the spatial binning approach, due to the overhead in constructing the spatial index for each object. CGI retained the local sensitive hashing.

2.3 The spatial binning algorithm

Our conclusions from the performed studies, incorporated into our conjunction prediction tool, are for the following sequence of filtering steps:

- 1) Perform initial orbit propagation of all objects using a relatively coarse time step. This can be of the order of few minutes.
- 2) Use simple interpolation for a small number of intermediate points. This reduces the maximum size needed for binning. Indeed, in one minute, an object in LEO can cover up to 500 km. A time step of one minute was adopted, so the position of all objects are known at each minute.
- 3) For each time step, apply 3D spatial binning. Here bins are cubes of 500km side. The binning is as follows:
 - 3.a) Divide space into cubic bins and assign each object to a bin. The propagation and spatial binning concept is illustrated in Fig. 1.
 - 3.b) Loop over all objects and find the objects in the same bin and the 8 adjacent bins. Searching in the adjacent bins is necessary to cover the cases where the initial object is close to the wall of its bin.
- 4) Determine closest-approach approximations based on positions at two subsequent time steps, by closest separation of pairs of vectors $[\mathbf{r}_1(t_i), \mathbf{r}_1(t_{i+1})]$ and $[\mathbf{r}_2(t_i), \mathbf{r}_2(t_{i+1})]$. This is necessary because the point of closest approach is most probably between two time steps. A rudimentary calculation gives that for time steps of 1 minute the maximum distance errors are roughly 100 m assuming circular orbits at higher than 200 km altitude.
- 6) One more filtering can be applied on the set of neighboring objects to find those that satisfy a stricter criterion, e.g. relative distance below 250 meters. Other

criteria may also be applied, for example finding objects inside a pizza box or an ellipse around the initial object.

2.2.3 Results

The filtering technique described above has been implemented and tested on an all-vs-all scenario. Here the conjunction criterion was a relative miss distance below 250 meters.

Orekit [6] is used as the astrodynamics toolbox in the software. Orekit provides various classes related to orbits, reference frames, time scales, propagation, Earth Orbit Parameters and leap seconds. It is based on the Apache Commons Math mathematical library. Orekit is already used by ESA and CNES, and it allows integrating physical models of high accuracy. Based on Java, the toolkit is extremely modular and easy to expand.

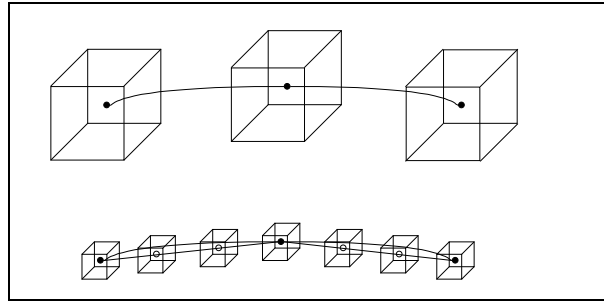


Figure 1. Spatial binning of (x,y,z) data for each time step. Linear interpolations on the position of the debris are done for time steps of higher resolutions.

A modular design allows using different kinds of catalogues. The tool was tested on NORAD TLE catalogues distributed by Celestrak [7] which has 13,000 entries. Orekit provides an SGP4 propagator.

Using the above technique, our prototype was able to process 13,000 objects in an all-vs-all scenario in 3 minutes using a single core of a 1.2 GHz Core 2 Duo CPU (or in just 8 seconds using all threads of a 4CPUDual-core machine running at 2.66 GHz) for 1 day (24 hours) prediction. The criterion for retaining a conjunction is a miss distance below 250 meters.

Tab. 2 shows how the number of pairs to be assessed is reduced with each filtering method.

Tab. 3 shows the relative performance of alternative first-pass filtering methods, based on 13,000x13,000 objects and 1 day predictions. As the prototype is being developed in time, the same catalogue has always been used for benchmarking purposes.

Table 2. Initial pair-reductions with alternative filtering methods

Initial Filtering Method	Number of 3D pair comparisons
None: $N(N-1)$	170 million
Avoid repeated pair checks, i.e. A vs B = B vs A: $N(N-1)/2$	85 million
Use Max Perigee & Min Apogee	29 million
DX primary filter subsequently followed by DY, DZ and DR filters	15 million
Spatial Binning	2 million

Table 3. Comparison of performance of alternative conjunction prediction approaches, for 1 day horizon

Conjunction Solution	Number of Objects	CPU speed [GHz]	Normalised Processing Time for 13,000 vs 13,000 (for published number of objects) [min]
SOCRATES	2,700 vs 8,500	2	316 (43)
CRASS	8,000 vs 8,000	2.6?	320 (121)
Closeap	8,000 vs 8,000	2.66	12.4 (4.7)
CGI (dx/dy/dz filter)	13,000 vs 13,000	1.2	20
CGI (ditto plus min I/O)	13,000 vs 13,000	1.2	8
CGI (ditto plus spatial filtering)	13,000 vs 13,000	1.2	5.1
CGI (ditto plus temporal interpolation of time steps)	13,000 vs 13,000	1.2	3.0
CGI (ditto on faster machine)	13,000 vs 13,000	2.66	1.2
CGI (parallelised) on 8 threads	13,000 vs 13,000	2.66	0.15

3 THE CLIENT-SERVER ARCHITECTURE AND TECH STACK

The software is split in a client-server architecture. The server side updates the catalogue, computes future conjunctions and broadcasts them to whoever is listening to the messages. The client side displays the data related to the broadcasted conjunction and supports a rich visualisation through a browser, without plug-in and tapping into the graphical processing capabilities of the client's machine.

The technology stack used is the foundation of the scalability and flexibility of the tool. Only technologies with commercial friendly licenses were considered, in particular the Apache license.

Reusing established industry technology and best practices is the key to keep the software simple and focus only on the actual problem, i.e. algorithms, without delving with time-consuming and hard-to-maintain bespoke software solutions. It is also a statement that space applications have indeed different requirements than other domains, but they can nevertheless be met by already existing proven software and approaches.

Below are the powerful technologies that have been used for easy integration and messaging inside the software.

3.1 Spring



Spring is the most popular application development framework for enterprise Java™. Millions of developers use Spring to create high performing,

easily testable, reusable code without any lock-in to particular vendors or solutions. Spring enables us to focus on the business problem rather than the plumbing that connects components and systems.

The Spring framework features an incredibly powerful and flexible collection of technologies ranging from Data Access to Security to modern web applications (AJAX, REST). It is also Cloud Ready, entailing that Spring applications are supported on all popular cloud platforms like Cloud Foundry, Google App Engine and Amazon EC2.

3.2 Camel




Apache Camel is a versatile open-source integration framework based on known Enterprise Integration Patterns (EIP) that focuses on making

integration easier and more accessible to developers. It does this by providing:


- concrete implementations of all the widely used EIPs
- connectivity to a great variety of transports and APIs
- easy to use Domain Specific Languages (DSLs) to wire EIPs and transports together

3.3 ActiveMQ

 Apache ActiveMQ is an open source message broker providing Enterprise Features like clustering, multiple message stores, and ability to use any database as a JMS persistence provider besides VM, cache, and journal persistency.

ActiveMQ is used as enterprise service bus with Camel, to distribute messages in a distributed environment.

3.4 CometD

 CometD is a scalable HTTP-based event routing bus that uses an Ajax Push technology pattern known as Comet. CometD is a Dojo Foundation project to provide implementations of the Bayeux protocol in javascript, java, perl, python and other languages.

3.5 Putting it all together

Using Spring assemblies routes are adjusted, adding new class through configuration. Camel integration patterns and components allow decoupling the code and massively reduce the code base. With ActiveMQ messages can be easily distributed in a secure, persistent and complete way. Finally with Using CometD we can stream data to web interfaces. Fig. 2 illustrates how each component interacts with each other and how the technologies orchestrate that.

4 VISUALISATION

Embracing bleeding edge web technologies gives the freedom to experiment with new and innovative ways of user interface and data visualisation.

For the visualization side of the tool CGI decided to target modern browsers using JavaScript, HTML5, WebGL, and THREE.js. These technologies offer a number of advantages for our purposes, including easy distribution of the product, lack of additional software licenses, and a decrease in development time in comparison with building a traditional desktop application. Additionally there is no installation for end users which is beneficial for operators, but also could

allow subsections of the application to be served to the public for PR purposes, including education.

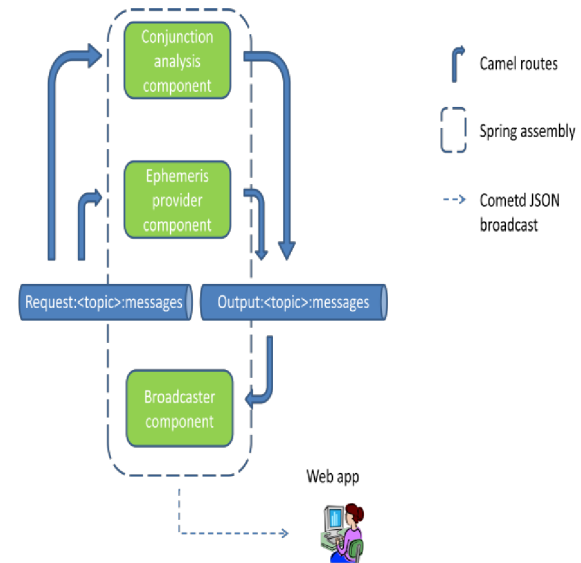



Figure 2. Technologies involved in the conjunction tool

4.1 Technology Dependency Overview:

 As mentioned, HTML5, WebGL and THREE.js are used. HTML5 is the newest HTML standard and allows a number of new features to run directly in the browser with no additional plug-ins. It introduces many innovations, in particular canvas and audio tags as well as offline storage. The most important feature, for our purposes, is the support that HTML5 provides for accessing WebGL.

WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 3D graphics within any compatible web browser without the use of plug-ins. The graphics are rendered in real time on traditional video hardware as is used in modern video games. WebGL programs consist of control code written in JavaScript and shader code that is executed on a computer's Graphics Processing Unit (GPU). WebGL is based on OpenGL ES 2.0 and it uses the HTML5 canvas element.



On top of this, the THREE.js library provides additional higher level functionality to ease the creation, management, and rendering of 3D scenes via WebGL.

4.2 The visuals

The Earth is textured with image data (provided to the public by NASA) which has been tiled and in some cases reprocessed by our own tool. At the time of writing, the targeted visual features include the following – although not all are fully implemented yet:

- 1) Time scrubbing the animations of all assets to any valid period of time.
- 2) Highlighting potential collision paths and locations, as well as the objects expected to be involved.
- 3) Streaming in higher resolution image data after initial load and caching said data between sessions. This will give immediate real time responsiveness right away even for first time access, but still provide gradually increasing visual quality over time which can be automatically maintained and immediately reclaimed on subsequent use sessions.
- 4) A number of hardware shaders (GPU programs) to increase visual quality and realism including Atmospheric shaders, night side glow post processing for the city lights, and more.
- 5) Realistic 3D models for all owned space born assets, including the ability to see not only where they are located, but also the correct angles, equipment status and orientation, and more.
- 6) Visualization of allowed arc ranges, as well as current visibility cones for radar and ground stations.

5 CONCLUSION

A modern conjunction analysis tool has been implemented by CGI. It is state of the art at many levels: a novel filtering algorithm accelerates dramatically the detection of conjunctions, while modern software technologies allow easy integration and modular structure of the tool. Emphasis has been put on appealing and ergonomic visuals and user interfaces.

CGI has experience in parallelisation of complex scientific algorithms, and has recently undertaken work using General Purpose Graphics Processing Units (GPGPUs) to assess the feasibility of using these for massive parallelisation, to further improving performance. The SSA conjunction prediction task is something which would be highly amenable to parallelisation, if further speed-ups are required.

Our aim with the visualization is to prove that it is possible and easy to be able to see everything in our sky, and our solar system, as well as Earth based ground and radar station coverage, in real time or at any point in the recorded past or projected future. This will allow for a number of possible applications including visualizing potential collision avoidance warnings, mission simulations, potential burns or visibility coverage, and more.

In theory there is no reason why the user interface could not be expanded over time to function as a browser for observing the state of, and for commanding, all controlled/known space assets. Imagine clicking on a satellite or ground station, or choosing it from a menu, and having a 3D model of that craft come up from which all parameters and equipment states can be viewed, or any commands can be issued as easily as playing a real time strategy video game, all this with real data.

Theoretically it would also be easy to visualize and analyse the NEO population and possible conjunctions by changing the Earth in the models to the Sun and the derbis database to a NEO database such as the one from NEODys.

CGI has implemented the routines to compute probabilities of conjunction using Alfriend's method but has not tested it so far extensively. CGI has also been looking at other methods that use fewer simplifying assumptions [8][9].

Such a tool has big potential, due to its performance, its ease to expand with richer features and more accurate physical models, its ease to operate and maintain and its potential accessibility to the end user through any device with a browser.

6 REFERENCES

1. Klinkrad, H.: "*Space Debris*", Wiley Online Library, 2006
2. Kelso, T. S., Alfano, S.: "*Satellite Orbital Conjunction Reports Assessing Threatening Encounters in Space (SOCRATES)*", 15th AAS/AIAA Space Flight Mechanics Conference, Copper Mountain, Colorado, USA, 2005
3. Escobar, D., Agueda, A., Martin, L. and Martinez, F.M.: "*Efficient ALL vs. ALL collision risk analyses*", Advanced Maui Optical and Space Surveillance Technologies Conference, Vol. 1, p. 32, 2011
4. Hall, R., Alfano, S. and Ocampo, A.: "*Advances in Satellite Conjunction Analysis*", Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference, Wailea, Maui, Hawaii, 2010
5. Woodburn, J., Coppola, V. and Stoner, F.: "*A Description of Filters for Minimizing the Time Required for Orbital Conjunction Computations*", Advances in the Astronautical Sciences, Univelt Inc, Vol. 135, No. 2, pp. 1157-1173, 2009
6. www.orekit.org
7. www.celestrak.com

8. Alfano, S.: "*Satellite Collision Probability Enhancements*", Journal of Guidance, Control and Dynamics, Vol. 29, No. 3, May-June 2006
9. Patera, R. P.: "*Satellite Collision Probability for Nonlinear Relative Motion*", Journal of Guidance, Control and Dynamics, Vol. 26, No. 5, September-October 2003