

VML 3.0 Reactive Sequencing Objects and Matrix Math Operations for Attitude Profiling

Dr. Christopher A. Grasso¹
Blue Sun Enterprises, Boulder, Colorado, 80302

and

Joseph E. Riedel²
Jet Propulsion Laboratory / California Institute of Technology, Pasadena, California, 91109

VML (Virtual Machine Language) has been used as the sequencing flight software on over a dozen JPL deep-space missions, most recently flying on GRAIL and JUNO. In conjunction with the NASA SBIR entitled "Reactive Rendezvous and Docking Sequencer", VML version 3.0 has been enhanced to include object-oriented element organization, built-in queuing operations, and sophisticated matrix / vector operations. These improvements allow VML scripts to easily perform much of the work that formerly would have required a great deal of expensive flight software development to realize. Autonomous turning and tracking makes considerable use of new VML features. Profiles generated by flight software are managed using object-oriented VML data constructs executed in discrete time by the VML flight software. VML vector and matrix operations provide the ability to calculate and supply quaternions to the attitude controller flight software which produces torque requests. Using VML-based attitude planning components eliminates flight software development effort, and reduces corresponding costs. In addition, the direct management of the quaternions allows turning and tracking to be tied in with sophisticated high-level VML state machines. These state machines provide autonomous management of spacecraft operations during critical tasks like a hypothetical Mars sample return rendezvous and docking. State machines created for autonomous science observations can also use this sort of attitude planning system, allowing heightened autonomy levels to reduce operations costs. VML state machines cannot be considered merely sequences - they are reactive logic constructs capable of autonomous decision making within a well-defined domain. The state machine approach enabled by VML 3.0 is progressing toward flight capability with a wide array of applicable mission activities.

I. Introduction

VML (Virtual Machine Language) has been used as the sequencing flight software on thirteen deep-space missions, most recently flying on GRAIL and JUNO, and slated for flight on OSIRIS-Rx. In conjunction with the NASA SBIR entitled Reactive Rendezvous and Docking Sequencer, VML version 3.0 has been enhanced to include object-oriented element organization, built-in queuing operations, and sophisticated matrix and vector operations. These improvements allow VML scripts to perform much of the work that formerly would have required a great deal of expensive flight software development to realize. Attitude profiling in VML is an example of a malleable guidance, navigation, and control (GNC) flight capability which can enhance safety by rejecting illegal attitude changes and lower implementation effort relative compared to equivalent flight software implementations.

Autonomous slewing and tracking makes considerable use of new VML features. Profiles generated by flight software are managed using object-oriented VML data constructs executed in discrete time by the VML flight software. VML vector and matrix operations provide the ability to calculate and supply quaternions to the attitude controller flight software which produces torque requests. Targeting the new spacecraft attitude involves

¹ Principal VML Engineer, Blue Sun Enterprises, 1942 Broadway Suite 314, Boulder, CO, 80302, Senior Member.

² Principal Engineer, Optical Navigation, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA, 91109.

determining the desired orientation by using vector operations on the spacecraft features to present to primary and secondary targets. Turning uses constraint-based slews which utilize trigonometric operations. Keep-out zones to prevent orienting sensitive elements of the spacecraft toward the sun are considered, and a beltway of known safe paths are utilized in order to guide the spacecraft from its current orientation to the orientation desired.

Using VML-based attitude planning components eliminates flight software development effort, and reduces corresponding costs. In addition, the direct management of the quaternions allows turning and tracking to be tied in with sophisticated high-level VML state machines. These state machines provide autonomous management of spacecraft operations during critical tasks like a Mars hypothetical sample return rendezvous and docking, including trajectory management and target acquisition. State machines created for autonomous science observations can also use this sort of attitude planning system, allowing heightened spacecraft autonomy to reduce operations costs.

VML state machines cannot be considered merely time-ordered sequences - they are reactive logic constructs capable of autonomous decision making within a well-defined domain. The state machine approach enabled by VML 3.0 is progressing toward flight capability with a wide array of applicable mission activities.

This paper outlines the basics of the attitude profiling problem in order to orient a spacecraft for power production, communications requirements, observations, and maneuvering. VML 3.0 flight capabilities are summarized that have evolved from flight missions like Mars Phoenix, and technical research like the Reactive Rendezvous and Docking Sequencer which is currently funded as a phase II NASA Small Business Innovative Research (SBIR) grant. A high-level outline of the profiling process is given, and examples of VML features used during the calculations is provided. Advantages of using the VML 3 scripting language over traditional flight software development are discussed, and implementation status of the VML attitude profiler is given.

II. Attitude profiling

A. Challenges

For a typical mission that has substantial target-relative Guidance, Navigation, and Control (GN&C) activities, the attitude resource of a spacecraft has many demanding customers. These include a power system which generally requires a power-positive solar-array attitude, propulsion events, navigation imaging, and science remote-sensing. If the mission includes a rendezvous, then the complex pointing associated with approach, rendezvous and docking will be required. Landing on a celestial body requires extended propulsion phases for large bodies, or discrete events for small ones, followed by attitude changes to achieve the landing position which will match the local surface normal – which on a small body is a rotating vector. Remote sensing, and especially mapping, may require complex sweeping attitudes to optimally develop visual and topographic maps of the surface, which often requires multiple passes to obtain different view geometries and varying time-of-day illumination angles. Flyby missions, such as of small bodies, require rapid turn-around of onboard-determined navigation solutions to maintain attitude lock on the target. Because of the singular nature of flyby geometries, such missions also frequently feature specialized additional constraints such as shield orientation (for a comet flyby) or a specific orientation for a large SEP array, to allow rapid slewing of the spacecraft body (around the panel axis) at close-approach. Navigation may require complex surveys of the celestial sphere for cruise navigation imaging, or targeting of specific longitude/latitude locations on a body surface for landmark acquisition. Science investigations may need to rapidly scan a gaseous limb of a planet or satellite with the high gain antenna to obtain radio-occultation observations from the Earth. Onboard science analysis software may be searching for changes on the surface of the target body, and request remote-sensing observations of opportunity. All of these examples have been planned and/or implemented on NASA deep space planetary missions over the last three decades.

To compound the many varied attitude demands of these missions – and often many of these are featured in a single mission – spacecraft attitude requirements must be achieved in the face of severe constraints. It has already been noted that a solar-powered spacecraft must generally remain power positive, which in the case of a thrusting SEP mission is a very stringent constraint. Spacecraft almost always carry instruments that cannot be pointed at the Sun, and many sensitive instruments cannot even look at a bright target such as the Earth or Moon (when illuminated) without a substantial subsequent cool-down period before again being usable. SEP engines have been known to be sensitive to direct Sun illumination as well, and often batteries or other electrical equipment has been found to be susceptible to excess illumination and associated heat. On the other hand, in-flight exigencies have

required forced sun-heating of otherwise low-temperature-craving instruments to relieve contamination condensation. Other engineering elements have been given sun-therapy as well, often temporarily over-riding otherwise stringent pointing constraints. High gain antennas have been precluded from Sun point on occasion, due to their focusing actions of radio waves as well as light, leading to concerns about melting of secondary mirrors and wave-guide apertures. Star trackers, though generally immune to damage by sun exposure, will almost always lose lock when exposed, and temporarily disable a critical part of the attitude control system, which in some instances might lead to a safe-hold condition. Sweeping a large body through the star-tracker field could cause the same problem.

Most of the above needs and constraints (excepting those involving onboard autonomy) could be satisfied by laborious hand construction of sequences on the ground for uplink and subsequent execution. However, this method, certainly the norm for missions of the past, is highly labor intensive, expensive and mission limiting. Missions utilizing highly autonomous GN&C, such as Stardust, Deep Impact and especially Deep Space 1 have shown the advantage of partial or complete automation of many of these processes and constraints. It is certain that future missions will feature increasing automation and the need to cope with increasingly complex spacecraft pointing needs. Thus the challenge becomes to perform at any time virtually any pointing needed by the spacecraft itself, its autonomy systems, the Engineering Team, and the Science Team. In addition to having this versatility, the system must be safe, must prevent user errors and detect self-faults, and must be easy for the ground and onboard autonomy systems to use: few things are more complex to specify than the immediate and on-going attitude of a spacecraft. Finally, the pointing system needs to be transparent in use, function, and operation, in order for the many stakeholders of spacecraft pointing to be assured that the desired, requested, or anticipated actions will in fact occur, and will occur safely.

B. Enabling factors for a modern approach

With the flight of Deep Impact [13], deep space missions finally entered the “modern” world of reasonably capable onboard computer processing, with the first flight of a RAD-750 processor. Though plagued with “beta-version” problems with the board, the throughput of the RAD-750 was sufficient to allow autonomous onboard navigation (AutoNav) to impact the nucleus of comet Tempel-1, and image the impact zone with the flyby spacecraft. But the RAD-750 is now over a decade old: more computing power is in the offing, and will most certainly be applied to the next generation of planetary spacecraft. This fact will eliminate the heretofore absolute constraint of spacecraft flight software systems, that anything computational must be done in the most efficient way possible. Elimination of this constraint will allow for relatively low-rate computations – even if somewhat complex – to be performed in a much more malleable language, such as VML, even if it is an interpreted language. But there are other enablers for a modern approach to attitude profiling.

Since DS1, a number of missions have taken advantage of the AutoNav generalized solar system body and spacecraft ephemeris specification system called “Ephemeris Services” (ES) [9]. ES represents the positions of a body or spacecraft in time as a set of Chebychev polynomials. The ephemeris is broken up into a series of polynomials in time with one string of polynomials for inertial x, y, and z. The polynomials can be in any frame and centered on any body, but traditionally, Earth Mean Equator 2000 centered on the Sun during cruise and on the target body on approach, flyby or orbit operations is used. The polynomial segments can be of arbitrary length, and can be of arbitrary order. The type of object being represented will determine that combination, with planet ephemerides in general showing an optimum polynomial representation with long period polynomials of high-order (e.g. 21), whereas a highly propulsive spacecraft trajectory will be optimal with short-period low-order (e.g. 3) polynomials. The use of ES allows other onboard elements to have instant access to the positions of any celestial body, the host spacecraft, or other spacecraft. ES also allows the representation of desired target trajectories, as was done on DS1, or even pointing directions, as is done on Dawn currently.

A highly flexible and generic pointing specification capability was designed and implemented for the first time on the Cassini spacecraft [17][18] in the Ada language, and largely re-implemented on Deep Space 1 in the C language, with many very substantial enhancements to enable AutoNav operation[19]. These missions took a traditional flight software approach to attitude specification and profile, and as such were relatively brittle and non-malleable, except for that made possible by the rare – and generally labor intensive and traumatic – FSW updates. As very highly capable as both systems were, there were numerous occasions when both operations teams wished they had more or subtly different abilities, and much ground effort was expended to work around the absent capability.

The RRDS (Responsive Rendezvous and Docking Sequencer) system [12] being implemented as part of a NASA Phase II SBIR is primarily targeted toward a hypothetical Mars sample return mission, but is to a high degree generic to any rendezvous problem. Working with an onboard autonomous GN&C system, such as the DS1 or Deep Impact AutoNav system, RRDS will accomplish the rendezvous of a spacecraft with a target and the docking with (or capture of) that target. As VML 3.0 has developed over the last several years, an advanced experimental version of DI's AutoNav has also evolved, to take advantage of some of the features of VML 3.0, especially including state machines and a number of other important and enabling features. This new version of AutoNav is called AutoGNC [7][17][20] because of attitude guidance and control features including attitude estimation, control and primitive attitude profiling capability. Though primitive, these capabilities were sufficient to perform TRL-6 class system computer-based demonstrations of several very difficult mission scenarios, including "Touch and Go" on an asteroid, and lunar landing of a large crewed vehicle. These attitude profiling systems were readily implemented in VML, where the mathematics of the profiling operations took place in small C routines. For VML 3.0 however, all of the mathematics required for the attitude specification and profiling can take place in VML itself, and this is the design of the RRDS attitude profiler.

C. Attitude specification

The design of the VML 3.0-based attitude specification system will make use of a number of characteristics that distinguish it from VML 2.0 including:

- Native state-machine constructs
- Object-oriented sequencing constructs
- Vector and matrix algebra
- Conditional wait/detection on multiple logical states
- Arrays and data structures

At the core of the RRDS attitude specification and profiling system is the generic specification of targets and spacecraft features. Targets can be any solar system object, arbitrary positions in space, another spacecraft, or a vector – anything whose position can be specified as a Chebyshev polynomial in time. This would include a fixed celestial attitude (e.g., a star), which is the degenerate case of a 0th order single term polynomial (i.e., a constant). The names of the targets, and their ES identifiers are loaded into a VML database with a VML attitude specification utility. Any number of potential targets may be entered into the database. Spacecraft features are associated with various elements or instruments or faces of the spacecraft, and might include sensors, star trackers, solar array gimbal axes, main engines, actuator booms, spacecraft faces, and anything else of operational interest. The features are named appropriately (e.g., "solar-panel yoke"), and entered in the database, with a unit vector associated with the feature in spacecraft coordinates, using another VML attitude specification utility.

The first specification necessary is the primary pointing specification. This could be, for example, the narrow angle camera (NAC) toward the orbiting sample (OS) – something a sample return mission would need to do. This specification would cause the spacecraft to be oriented such that NAC camera boresight would be pointed to the OS, based on the ES-specified ephemeris of the OS, which in turn might have been computed by the ground and uplinked, or perhaps generated onboard by an autonomous navigation function.

The next specification necessary is the second constraint of attitude, as specifying only one pointing direction leaves a 2π radian ambiguity rotation about the primary pointing specification. There is always at least one other specification needed by any spacecraft besides the primary, and that is usually to provide power. In this case the secondary feature is "Panel-yoke", and the secondary target is "Sun", but there is another specification for the secondary required, and that is an orientation specification, in this case that is "normal". This accomplishes putting the gimbal axis of the solar array normal to the sun vector, allowing the gimbal to put the arrays flat on the sun to maximize power. Setting the orientation specification "toward" the sun would have attempted to place the panel yoke as near to pointing toward the sun as possible – in general a direct point would not be possible – but this orientation would be of little use for gathering energy.

With the primary and secondary specifications stated, the spacecraft will have an orientation to achieve when the specification is activated, with the NAC on the OS, and the panel yoke normal to sun (with the arrays automatically

turned to point to the sun). But there remains another ambiguity to resolve. The panel-yoke can be oriented normal to the sun line in two ways, 180 degrees apart. This is resolved in two possible ways in the RRDS attitude manager. One way is with an explicit statement of a tertiary target/feature/orientation like “sun”, “x-axis”, “away”, which would disambiguate by achieving an attitude that keeps the spacecraft X-axis away from the Sun: this would be useful if that spacecraft face hosted most of the temperature sensitive gear. A second way is with a specification of tertiary_constraint_mode = “automatic” which would allow the constraint manager to automatically make the decision (to be discussed below). See Figure 1.

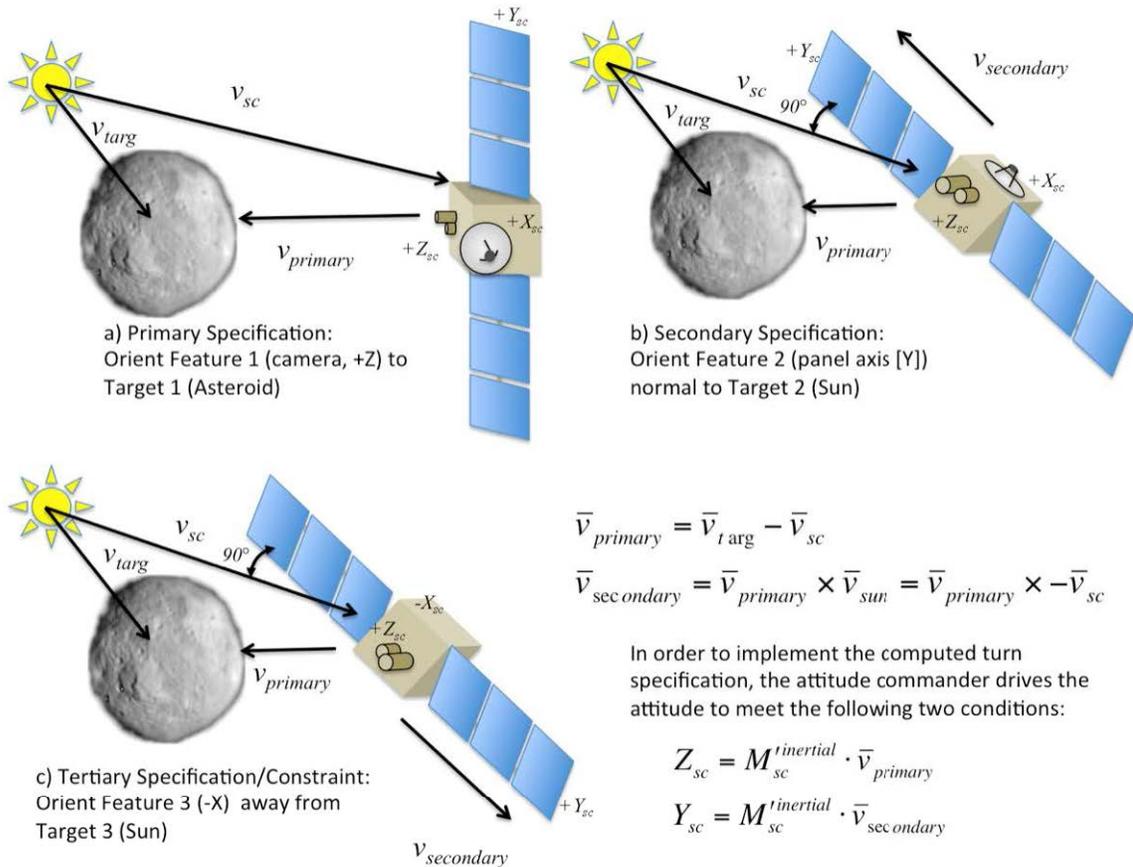


Figure 1: Step-wise definition of pointing attitude. 1) Specification of primary target/feature, leaving rotational ambiguity. 2) Specification of secondary target/feature with orientation, leaving binary ambiguity. 3) Binary disambiguation.

D. The RRDS VML-based attitude profiling system architecture and operation

Having in hand a generalized means of specifying the pointing of the spacecraft in almost all circumstances, the architecture of the profiling system can now be described. Figure 2 shows the data flow and operational elements of the RRDS Profiler. In previous missions, all computational and logical elements were in compiled form: in recent history, this has meant mostly the C language. In the current RRDS design, key elements of the profiler are written in VML. These include the overall Attitude Profiler construct, the Turn Manager, and the Track Manager. These managers make direct calls to C-subroutines for additional information or services, such as to ES. As a request to implement an attitude specification is received, the Attitude Profiler state machine is set to the Turn state, and turn planning commences (to be described in detail below) wherein the path to the desired attitude is computed to avoid attitude constraints. The path is computed in discrete short (e.g., 10 second) intervals, and spline polynomials are fit to these intervals for the attitude quaternion, quaternion rate, and acceleration as a function of time. The splines in turn are passed to the attitude commander, which simply evaluates the polynomial plan for a real-time expression of the immediate attitude profile. When the planner, which operates in real-time synchrony with the actual spacecraft turn, completes the plan, the Attitude Profiler state is promoted to “Track,” but the data flow remains the same, with

~10 second intervals of spline polynomials periodically produced, and passed to the attitude commander. The principal distinction between “Turn” and “Track” is that the beltway turn path planner is not invoked in “Track”. Nevertheless, the attitude constraints are constantly monitored by the Track Manager (in much the same way as the turn-path planner observes them) and will halt tracking if the desired specification forces the spacecraft into a pointing constraint zone, triggering a fault state.

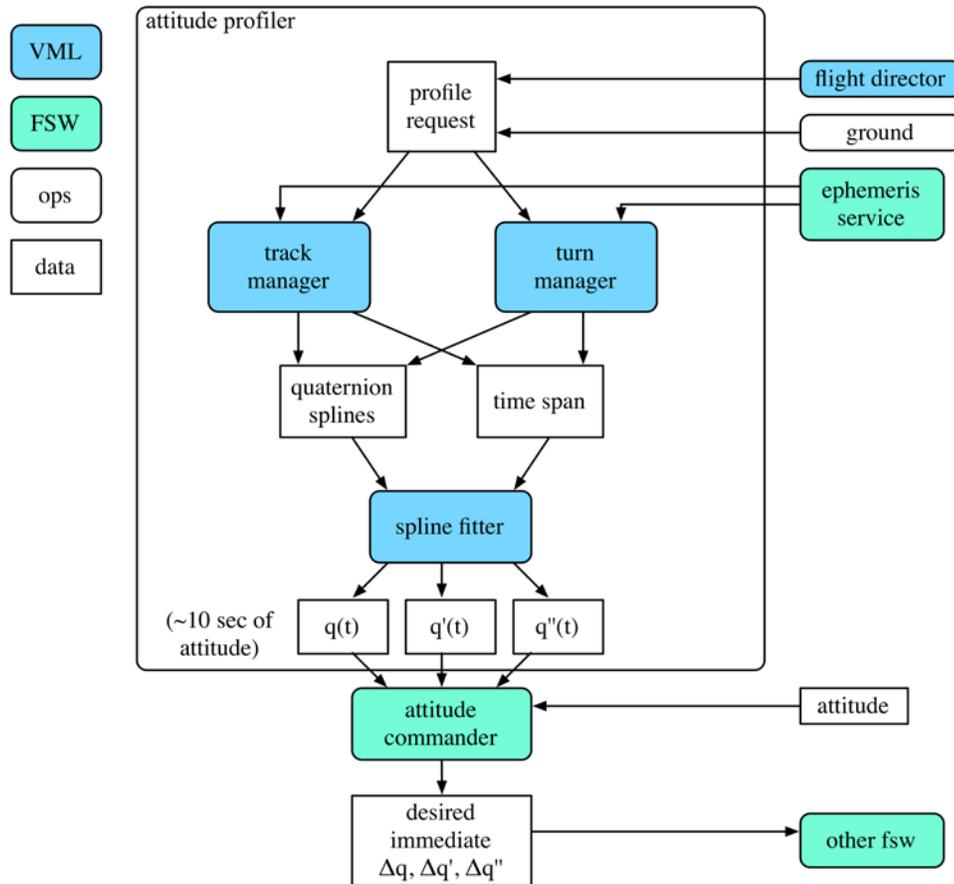


Figure 2: VML-based attitude profiler. Profiler implemented as VML components using the AutoGNC ephemeris service and providing outputs to flight software attitude commander.

The self-checking of constraints and flagging of violations raises a key advantage of using VML for the attitude profiling application, namely fault protection. With VML-based managers controlling the real time logical operations, it is very natural to embed the logical actions of fault handling in this naturally responsive and highly real-time language. Because of this, all fault detections and reactions within the profiler are laid out in the managers of the profiler itself, and fault states that cannot be resolved within the profiler are referred upward to the Flight Director for action, the Flight Director itself being another state-machine-based VML element.

Other advantages of the VML approach include the much more compact code formulation of real-time events in the endogenously real-time language. State machines, being part of the native constructs of VML 3.0 are also represented transparently. These two factors alone account for at least an order of magnitude reduction in lines of code vs. C-language coding of similar functions. With reduced code comes increased visibility of function. There are few states of the spacecraft more important than its attitude: the VML-based attitude profiler makes the access and action of the attitude profiling process transparent to all elements of the mission, and gives the mission planners and managers additional tools to accomplish necessary mission functions. Included in these tools are the ability to create arbitrary numbers of attitude specification abbreviations, as over 90% of a typical mission attitude specifications are repeated specifications, and an abbreviated attitude specification can result in very substantial operations savings.

Though block modularity of commands is nothing new, VML 3.0 offers the ability to spontaneously create shorthand abbreviations of attitude specifications with parameterizations using an uplinked sequence. For example, one such specification could be to point the NAC to a target, with panels on, but with the NAC boresight biased by a parameterized value. This specification could be created easily using the RRDS attitude profiler with simple mathematical modifications to the target inertial vector. In previous missions such a definition would have required a new FSW build.

E. Example cases for the attitude profiling system

1. Tracking a celestial body while power positive

Shown in Figure 3, this is a typical science configuration of a spacecraft, allowing imaging of the target while getting maximum power on the panels. The NAC is directed toward the geometric center of the body. At the same time, the panel yoke of the spacecraft is rolled about the NAC axis, to the point that it is normal to the Sun vector. There are two such solutions, with the X-axis of the spacecraft either toward or away from the sun. In this case, the X-axis is away from the sun. Schematically, the entire specification is:

Primary attitude specification (Target/SC-feature): Europa/NAC

Secondary attitude specification (Target/SC-feature/orientation): Sun, Panel Yoke, normal

Tertiary attitude specification (Target/SC-feature/orientation): Sun, X-axis, away

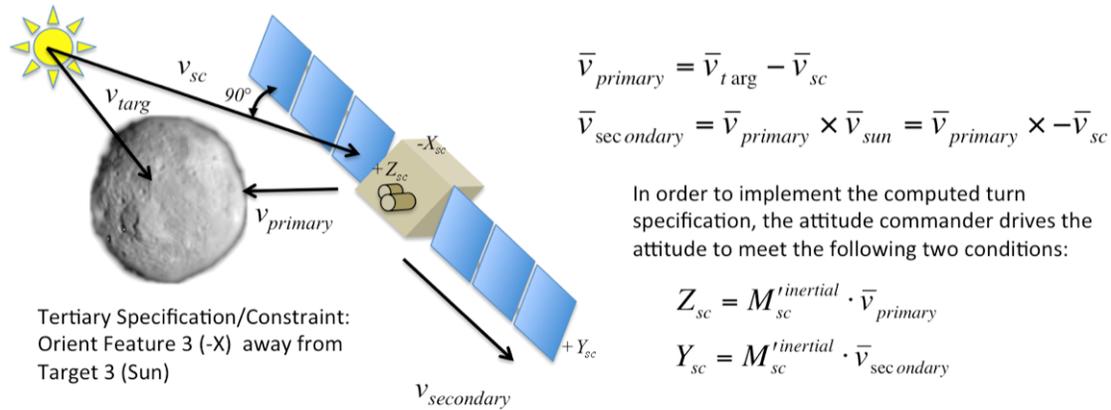


Figure 3: Tracking a celestial body with panels on Sun. Typical configuration for science observations.

2. Keeping a thrust vector on a pseudo-target while power positive

Shown in Figure 4, this is a configuration similar to that which the Dawn spacecraft is frequently commanded, but without the many advantages and conveniences that a VML-based profiling strategy would offer. The SEP engine is pointed in the direction of an ephemeris pseudo-target, while maintaining a power-positive state. The pseudo-target is a unit vector specified as any body would be specified in an ES ephemeris file while the panels are positioned as in the previous case, but here the secondary ambiguity is resolved by positioning the X-axis of the spacecraft toward the sun. Schematically, the specification is:

Primary attitude specification (Target/SC-feature): thrust_target_id/engine_thrust_vector

Secondary attitude specification (Target/SC-feature/orientation): sun, panel yoke, normal

Tertiary attitude specification (Target/SC-feature/orientation): sun, x-axis, toward

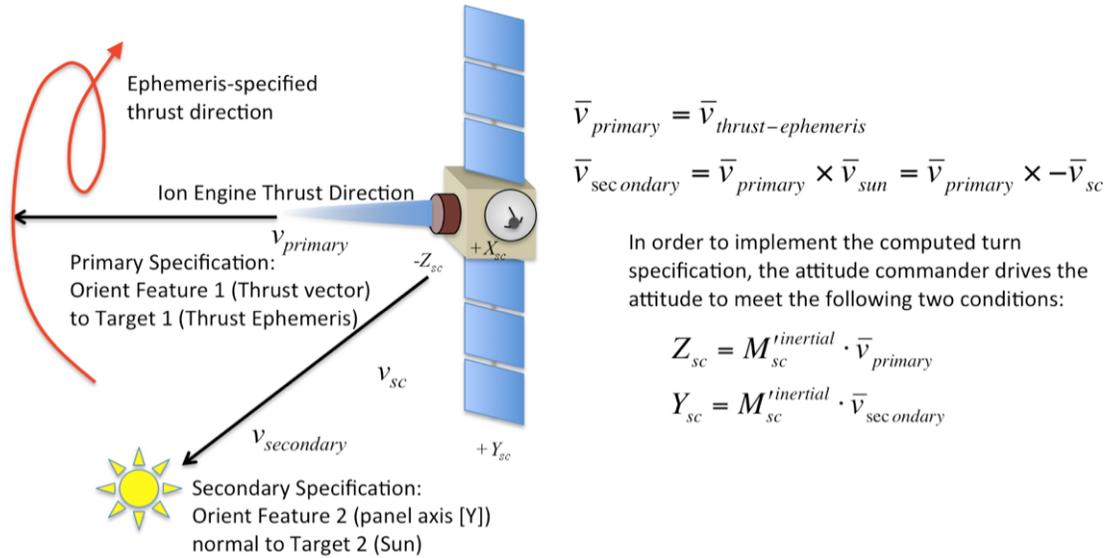


Figure 4: Thrust vector on pseudo-target, panels on Sun. Point the thrust vector of the ion engine at a changing calculated position in space while keeping the panels on the sun for power production.

3. Tracking an object during flyby, while remaining as power-positive as possible

Shown in Figure 5, this presents a complex case of a SEP powered spacecraft (with large gimbaled arrays), tracking an object in a high-speed flyby. The key geometrical problem is to guarantee that the spacecraft is able to slew about the panel yoke of the spacecraft, as rotating the spacecraft about other axes will be much more time-consuming due to the large mass-moments of those axes. Therefore a custom orientation is defined, with VML-based mathematics as shown in the figure. Schematically, the specification is:

Primary attitude specification (Target/SC-feature): asteroid/NAC

Secondary attitude specification (Target/SC-feature/orientation): custom, panel yoke, toward

Tertiary attitude specification N/A

In this case, no tertiary attitude specification is necessary, because by specifying the panel yoke toward the custom pointing direction (as formulated in the figure), there is no ambiguity.

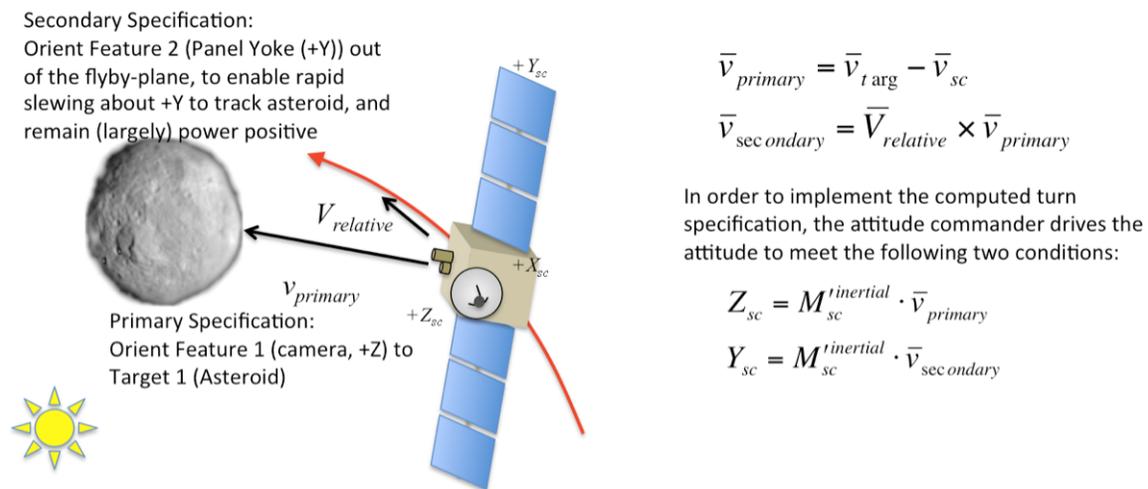


Figure 5: Track flyby body, panel yoke out of fly-by plane. Point an instrument package at a target and protect the array from potential debris strikes by pointing the panel yoke in a direction other than the flyby plane.

III. Spacecraft commanding using Virtual Machine Language

A. Attitude profile use

The attitude profiler is implemented as a series of VML constructs. These constructs manage data, perform calculations, access flight software services, and publish results in order to drive the spacecraft orientation. VML 3.0 is a commercialized product developed partly under NASA's SBIR program, chosen for its capabilities and heritage.

B. Spacecraft commanding

Commands are directives to the spacecraft, typically represented in a human-readable form and translated to a binary format. Commands cause the spacecraft to behave in some desirable way for the purposes of science collection, power management, thermal stabilization, propulsive maneuvers, pyrotechnic firing, and the like. Commands may originate from ground-based human operators, flight software elements and sequences.

Sequencing is the issuance of spacecraft commands from an on-board store which allows the spacecraft to perform in an automated fashion when no uplink is available, or when light speed delays obviate direct commanding from the ground. Virtual Machine Language (VML) [1][5] is an award-winning [6] standardized multi-mission language supplied by Blue Sun Enterprises which provides a structure from which spacecraft commands are issued. Commands in VML may be timed according to absolute (wall-clock) time and relative time, as well as in response to conditions on board the spacecraft using a technique known as *event-driven* sequencing.

C. Features and components

The VML flight execution environment provides multiple threads of execution within one task context using a data-driven construct known as a *sequencing engine*. VML allows an extensive set of variable types, including integers, unsigned integers, double-precision floats, logicals, and strings. Arithmetic and trigonometric calculations, logical manipulations, and matrix operations are available for use. Conditionals may be used to make decisions based on local values at runtime. WHILE and FOR loops perform iteration. Sequences exist as named functions which can accept parameters and have local variables. Functions may be packaged together into a single file loaded onto an engine in order to associate runtime behavior or to provide libraries of commonly needed services.

The VML tool suite consists of an embedded VML Flight Component (VMLFC), a ground-based VML Compiler, and the Offline Virtual Machine (OLVM) program. This suite allows products to be generated, loaded, executed, and tested. The relationship of each of these VML tools is shown in Figure 6. A source file containing human-readable VML script is generated using a standard editor or a tool. The compiler translates a text file into a loadable binary file, translating commands and times using mission-specific tools and tracking valid global variables and symbolic constants for the mission. The file produced can then be loaded by the VMLFC.

A typical development process runs the compiled module under OLVM in order to test and validate the behavior of the code. OLVM is used to perform user-defined tests automatically by first capturing a user-guided session, then extracting user keystrokes from the human-readable session output and rerunning the test. This automates the testing process with very little investment of effort. OLVM can be widely deployed on Linux, Macintosh, and Sun platforms. Developers typically test products before taking them to the slower, less available, and more expensive real-time Software Test Lab.

D. Heritage

Virtual Machine Language development started in 1997. Five versions have been implemented so far. VML has been used or is in use on thirteen NASA flight missions to date, including Stardust [2], Genesis, Mars Odyssey,

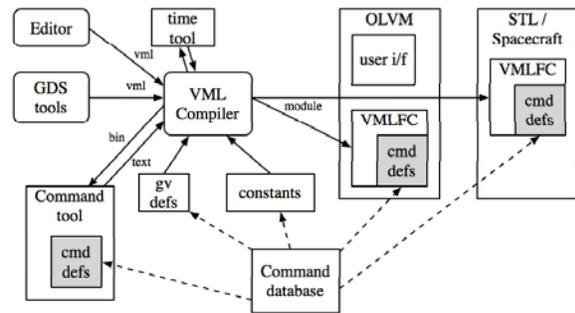


Figure 6. VML tool chain. Files of functions are created by using an editor or ground data system (GDS) tool to create human-readable VML script. The compiler translates this into a binary format usable within OLVM or a flight computer in a test lab or on the spacecraft.

Spitzer Space Telescope [3][4][8], MRO, Dawn, Phoenix, JUNO, GRAIL, and MAVEN. VML is slated for use on OSIRIS-Rx. VML 3.0 is in use on Kennedy Space Center's RESOLVE lunar regolith analysis demonstration.

IV. Beltway attitude profiling

Attitude profiling may be implemented relatively simply as a *beltway*, by having the spacecraft orientation follow as often as possible a fixed set of paths rather like the serpentine belt of a car's internal combustion engine. The beltway defines safe paths to use in order to avoid adopting unsafe or damaging orientations. The attitude profiler is responsible for entering and exiting the beltway, and prebuilding the beltway paths based on constraints. This section discusses the graphical representation of orientation constraints, a simple turn algorithm based on the concept of beltways, and the simple mathematical relationships used when implementing the algorithm

A. Graphical representation of constraints

Constraints prevent pointing elements of the spacecraft in potentially damaging directions. The most common constraint satisfaction problem in attitude profiling involves turning the spacecraft from its current orientation to a new orientation while avoiding pointing instruments and other light- and heat-sensitive components at the sun. These constraints can be represented in three-dimensional space as cones centered on the spacecraft center of mass. Having between two and ten keep-out zones is typical for a spacecraft.

A two-dimensional projection of these keep out zones onto the celestial sphere yields the field of view shown in Figure 7, with azimuth ranging from $-\pi$ to π , and elevation ranging from $-\pi/2$ to $\pi/2$. The target points represent the desired location of the sun at the start and end of the turn. The spacecraft is turned in a way that the sun moves between and among the keep-out zones without crossing into one. In this case, the straight-path turn would inappropriately allow solar impingement on zones k_2 , k_3 , and k_4 , requiring an alternate path. One simple, safe, and relatively efficient path to follow is highlighted in grey.

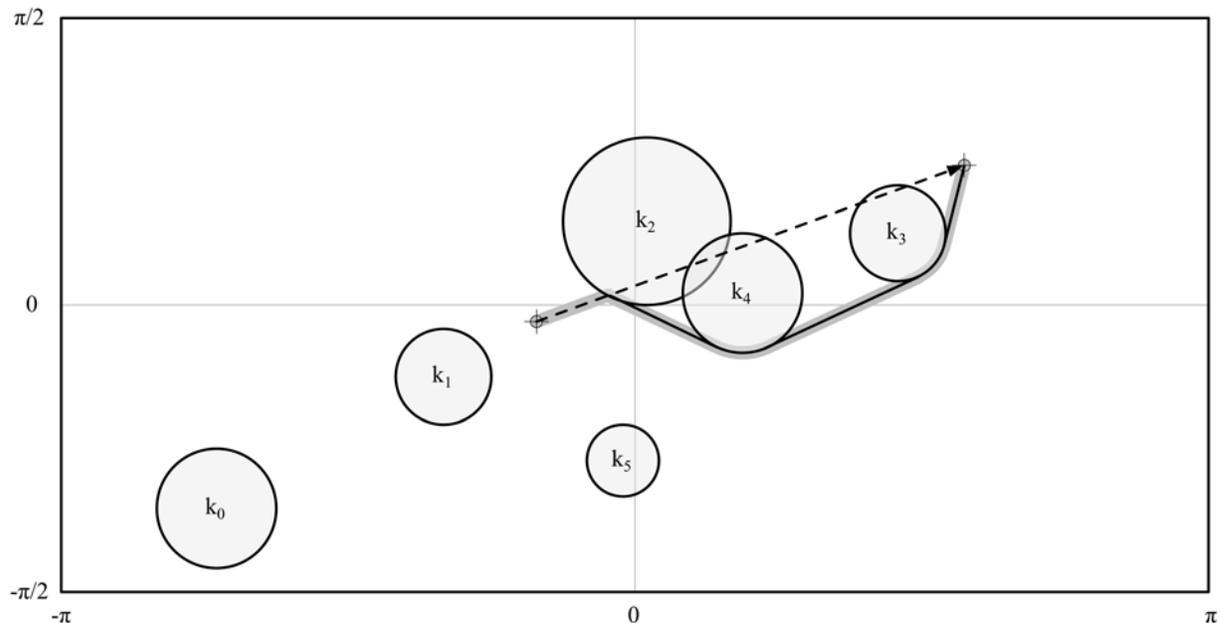


Figure 7. Constraint representation as field of view. Projection onto two-dimensional field of view in azimuth range $-\pi \dots \pi$ radians and elevation range $-\pi/2 \dots \pi/2$ radians.

In the current operational paradigm, the spacecraft operations team pre-checks endpoints for legality, then creates a turn profile on the ground and uplinks the result to the spacecraft for execution. Ground personnel use a variety of mathematical computations and analyses to optimize the path and verify its safety. The consequences of a mistake can range from minor to disastrous, including degradation of instrument functionality, loss of cryogen, and instrument destruction. When the ground plans a turn, the results are typically cross-checked for penetration into keep-out zones by the flight system. In such a case the flight system stops motion, safes the spacecraft, and requests ground intervention. Safe mode recovery is a costly and complex reaction to a bad turn request: it may result in

workweeks or work-months of personnel time to recover the spacecraft to normal operating procedure, and should be avoided.

One way to reduce the risk of turn errors is to use on-board calculation of the turns, and onboard constraint-based turn planning. Placing the turn process entirely onboard has the added advantage of simplifying the turn execution process by largely removing associated ground activities. In addition, on-board turn calculation allows larger on-board autonomy like on-board trajectory planning to occur using flight software elements like AutoGNC. Modern sequence languages like VML contain sufficient mathematical richness to implement algorithms directly in the sequencing language. Due to the lower cost of implementation and larger "bang for the buck" of sequences relative to flight software, a VML implementation can reduce cost and risk relative to flight software.

B. Beltway turn algorithm

The algorithm for determining the turn path in Figure 7 is simple, yet efficient. The destination endpoint for the sun within the field of view (provided by the ground, or by other on-board autonomy elements) is first checked for validity, and the requested turn is refused if the destination falls with a keep-out zone. Next, the set of relevant zones to consider is determined by finding the zones intersected by a straight-line path to the final orientation. If no zones are intersected, a straight-line path is followed. Otherwise, the algorithm calculates the intersection of the straight-line transit from the current position with the first keep-out zone circumference or tangent intersected by that path, creating the first segment of movement with the intersection as its endpoint. Once it reaches this intersection, the spacecraft follows the circular zone boundaries and pre-calculated tangents between keep-out zones which form a beltway to follow around the zones. The path exits the pre-calculated tangents to reach the final orientation.

The algorithm has several advantages over a three-dimensional solution. First, it is simple enough to be implemented onboard without undue processing. Second, most of the elements of any movement are common, thereby simplifying test: the beltway provides a common set of paths, all of which can be tested. Only the initial path segment from the current position to the required beltway and the exit from the beltway to the final destination are unique, and easily derived as straight lines from a point to a tangent point on a circle. Finally, it is relatively easy to visualize how the algorithm behaves, and to cross-check its behavior during spacecraft integration and operations.

C. Mathematical operations for finding intersections and tangents

Since the size and location of the keep out zones is constant, the entire set of beltways can be pre-calculated for later use during turning. Each potential beltway relationship between keep out zones is shown in Figure 8 (i) and (ii), where intersecting circles have only two possible outer tangents and non-intersecting circles have a set of two inner and two outer tangents. For n keep out zones, the worst-case number of possible relationships between each set of two circles, assuming no circles intersect, is $(n-1)^2$. For the mission with six keep-out zones shown in Figure 7, 98 tangents between circles must be calculated. During runtime, relationships solving for the intersection of a line segment and a circle shown in (iii) and two line segments shown in (iv) are also required.

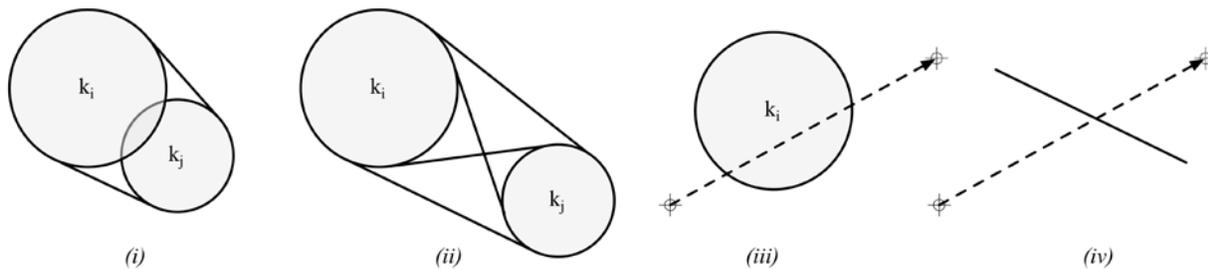
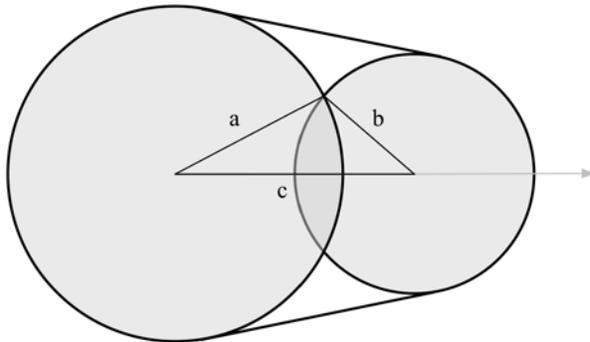


Figure 8: Geometric relationships appearing in beltway attitude profiling. (i) Intersecting circle, outer tangents. (ii) Nonintersecting circles, inner / outer tangents. (iii) Intersecting segment and circle. (iv) Intersecting segments.

The mathematical operations for finding the circle tangents shown in Figure 8 involve changing coordinate systems from Cartesian to a polar system centered on the left circle, rotating the coordinate system so that the right circle lays on the x axis, applying a set of algebraic equations to solve for the tangent line segments in the simpler frame, rotating these solutions back to their original orientation, and translating them back to the standard frame of reference and back into Cartesian form. The derivation of the solution in the simplified frame is omitted for brevity,

but requires nothing more than basic algebraic and trigonometric manipulations. A sample of the mathematical formulas used in the problem is given below.

1. Intersection of two circles in simplified frame

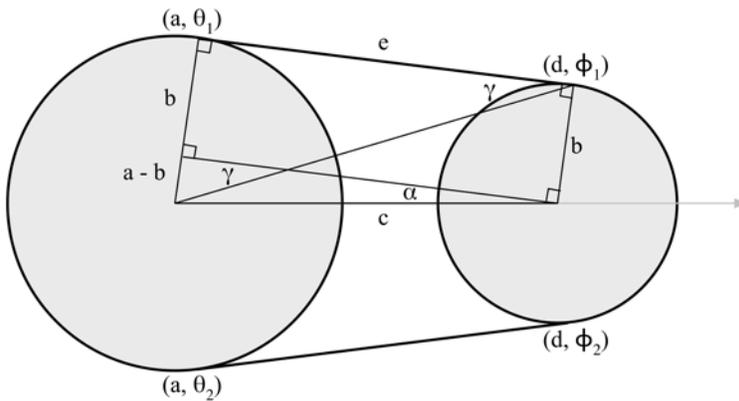


if $c > a + b$, 0 intersection points

if $c = a + b$, 1 intersection point
 $\theta = 0$

if $c < a + b$, 2 intersections points
 $\theta_1 = \cos^{-1}((b^2 - a^2 - c^2) / 2a)$
 $\theta_2 = \cos^{-1}((b^2 - a^2 - c^2) / 2a)$

2. Outer tangents between two circles in simplified frame



$$e = \sqrt{a^2 - (a-b)^2}$$

$$e = \sqrt{a^2 + b^2 - c^2}$$

$$e = \sin^{-1}\left(\frac{a-b}{c}\right)$$

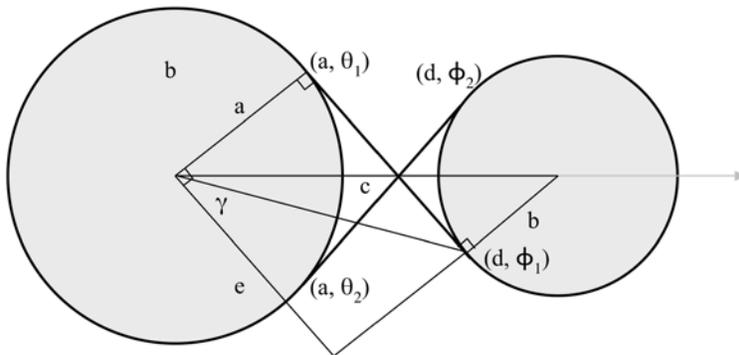
$$e = \tan^{-1}\left(\frac{a-b}{c}\right)$$

$$e_1 = \frac{a-b}{c}$$

$$e_2 = \frac{a-b}{c} + \frac{b}{c}$$

$$e_1 = -e_2$$

3. Inner tangents between two circles in simplified frame



$$e = \sqrt{a^2 - (a+b)^2}$$

$$e = \sqrt{a^2 + b^2 - c^2}$$

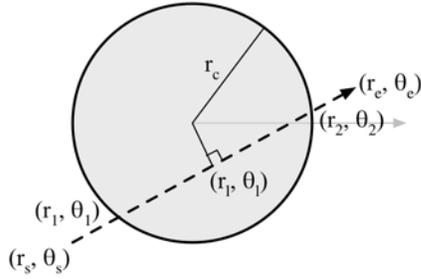
$$e_1 = \sin^{-1}\left(\frac{a}{c}\right)$$

$$e_2 = \sin^{-1}\left(\frac{b}{c}\right)$$

$$e_1 = e_2 - \frac{a-b}{c}$$

$$e_1 = -e_2$$

4. Intersection of line segment with circle in simplified frame



$$\begin{aligned} \alpha &= \tan^{-1} \left(\frac{y_1 - y_2}{x_1 - x_2} \right) \\ \alpha_1 &= \alpha \cos \frac{r_1 - r_2}{r_c} \\ \alpha_2 &= \alpha + \cos^{-1} \left(\frac{r_1}{r_c} \right) \\ \alpha_3 &= \frac{\alpha}{\cos \frac{r_1 - r_2}{r_c}} \\ \alpha_4 &= \alpha - \cos^{-1} \left(\frac{r_1}{r_c} \right) \\ \alpha_5 &= \frac{\alpha}{\cos \frac{r_1 - r_2}{r_c}} \end{aligned}$$

V. VML implementation of beltway attitude profiling

VML supplies the data structuring capabilities and matrix, trigonometric, arithmetic, and logical operations necessary to implement the beltway attitude profiler described above. The advantages to using VML are twofold: effort for developing VML constructs are considerably lower than for similar flight software, and the operational implementation allows standard command, file, and telemetry services to be used to verify behavior. This section lays out the high level operations of such a system, VML support for needed data structures and mathematical operators, sample VML statements implementing portions of the mathematical manipulations, and an evaluation of the efficiency of the algorithm based on the number of operations required.

A. High level operation

Operationally, attitude profiling of the spacecraft shown in Figure 2 involves receiving some description of a turn, creating a set of spacecraft states based on this turn description, and feeding that set of states to a controller. The controller translates the states into a desired immediate state of position, velocity, and acceleration which is fed to a thrust and torque allocator, distributing the request across actuators like reaction wheels and thrusters. This allocation in turn drives the reaction control software, causing the spacecraft to read the resulting change via the star tracker, inertial measurement unit, and/or other instrumentation. An attitude estimator feeds the resulting approximation of the spacecraft orientation back to the attitude controller, which continues the turn.

The attitude profiling process lives within the overall attitude control process as the tracking manager (for maintain orientation) and the turn manager (for changing orientation targets). The managers are coded as VML components running on sequencing engines, accessing low-level flight software capable of providing data-driven services like ephemeris tracking.

By implementing the attitude profiler in VML, it becomes a simple matter to allow turn requests to come from a variety of sources in addition to the ground, including onboard autonomy elements like a trajectory management system or a flight director. Turn representations become high-level descriptions that are translated on board into turns for consumption by later stages in the attitude control process.

B. Data organization using collections

Certain elements of the attitude profiler data are sufficiently complex that structuring of related elements helps to organize the code, thereby making it simpler and easier to understand, develop, and maintain. VML provides a construct called a collection, essentially a heterogeneous array that can be used in a manner similar to an array or a data structure in C.

The quaternions representing the spacecraft position, velocity, and acceleration, and the changes thereto, are natural applications for collections. Spacecraft vectors appearing in Figure 4 have corresponding data declarations as three-vectors shown in Figure 9 a. A single quaternion would be declared as a 4x1 vector of double precision floating point values, following a standard {x, y, z, magnitude} layout with a magnitude of 1 and {x, y, z} constituting a unit vector. An entire array of quaternions containing values spanning an interval of time is given in Figure 9 b. In addition, start and stop indices track which span of values are currently in use.

```

declare gvVPrimary := vector(3)
declare gvVSecondary := vector(3)
declare gvVSun := vector(3)
declare gvVSpacecraft := vector(3)
declare gvThrustEphem := vector(3)

declare gvQ := double(10, 4, 1)
declare gvQdot := double(10, 4, 1)
declare gvQdotdot := double(10, 4, 1)
declare gvQStart := 0
declare gvQStop := 0

```

(a)

(b)

Figure 9: Declarations for global variables visible to the flight software and to the ground. (a) Vectors representing the primary and secondary pointing vectors, position of the sun, and orientation of the spacecraft. (b) Position, velocity, and acceleration quaternions covering one second intervals for ten seconds, and a set of indices indicating the range of elements containing valid values in those arrays.

C. VML statements for operations

The full set of matrix, arithmetic and trigonometric operations available in VML is given below. From these operators and functions, powerful spacecraft elements requiring sophisticated computation can be derived. The table is organized by the kind of operation. In addition to operations common to mathematical problems, logical, string, and bitwise operations and functions are also listed. The attitude profiler makes use of primarily matrix, arithmetic / trigonometric, logical, and comparison operations.

Matrix	Arithmetic / Trig	Comparison	Logical	Bitwise	String
+	+	=	and	and	concat
-	-	!=	or	or	split_left
*(matrix, scalar)	*	<	xor	xor	split_right
/	/	<=	not	invert	length
^	^	>		shift_left	
abs	abs	>=		shift_right	
adjugate	modulo				
cofactor	sin				
cross	cos				
determinant	tan				
dot	asin				
identity	acos				
invert	atan				
minor	atan2				
set_value					
transpose					

Figure 10: Operations available in VML. Organized by data type, showing operators and functions available for inclusion into sequencing products. Note that square root is implemented by the power (^) operator with an exponent value of 1/2.

D. Sample VML statements for cross product derivation and tangent calculation

The basic equations given in sections II C and IV C are directly supported by VML matrix operation statements, trigonometric operations, and arithmetic operations. A sample portion of code from the turn manager which calculates the primary and secondary pointing axes for Figure 4 (the thrust vector case) is given in Figure 11. The specification of the mathematical relationships concisely maps to VML constructs which execute at runtime. By writing into sequencing global variables, values are automatically telemetered to the ground for monitoring and verification, yielding considerable insight into the behavior of the system.

```

gvVPrimary := gvVThrustEphem ;updated by flight software service ES
gvVSecondary := gvVPrimary cross gvVSun

```

Figure 11: Vector operations for pointing axes with thrust vector on pseudo-target and panels on sun. Flight software updates the thrust vector ephemeris and sun ephemeris by writing the results into global variables. Matrix operations supply the cross product

A sample portion of code from a function for calculating inner circle tangents is given in Figure 12 in order to demonstrate the correspondence of these functions to the mathematical representations in IV C. The names of local variables differ from the more succinct mathematical names in order to enhance the clarity of the VML representation. The calculation accepts two inputs defining the radius of the two circles in the simplified frame of reference, and the offset of the second circle from the first along the axis in the simplified frame. Elements of the calculation block defining outputs and internal local variables have been omitted for brevity. Note the use of the power function with an exponent of 0.5 as a substitute for the square root function shown in C. Partial calculation products like `len_sq` are used when a term appears in more than one location in order to reduce computational effort. A constant value for π is defined in a common location in order to unify the value among all users. Ellipses (...) indicate omitted statements.

```

calculation calcInnerCircleTangents
  input rc1      ;radius of circle 1, a in diagram
  input rc2      ;radius of circle 2, b in diagram
  input offset2  ;offset of circle 2 from origin, c in diagram
  ...
body
  len_sq := (offset2 ^ 2) - ((rc1 + rc2) ^ 2)
  len := len_sq ^ 0.5

  d := (len_sq + rc1 ^ 2)

  thetal := asin(len / offset2)
  gamma := asin(rc1 / len)
  theta2 := thetal - gamma - pi / 2
  ...

```

Figure 12: Partial set of VML statements for calculating inner circle tangents

E. Single plan turn manager vs. periodic replan

A state machine diagram of the current version of the turn manager being developed is shown in Figure 13 (a), and uses a one-pass approach, planning an entire turn with all constraints, executing that turn as a series of segments to be followed, and falling into tracking mode to stay aligned with the target of the turn. It assumes turns of a short enough duration that any updates to the target are handled by tracking after the turn is complete.

The next version of the algorithm is shown in Figure 13 (b). It will use a periodic replan during the execution of the turn. This will allow tracking of targets that may be moving quickly relative to the spacecraft, for instance surface features on a body during a flyby. Note that this capability requires only very minor additions to the original single plan version, and shows up on the state diagram as the addition of one transition from turning back to planning. All of the single-plan code will be reused without change. This demonstrates the highly factored nature of the VML solution for attitude profiling. This change can be implemented without any changes to the underlying flight software, and could be applied in flight by simply changing a file containing the turn manager sequence elements and reloading an engine with the new file.

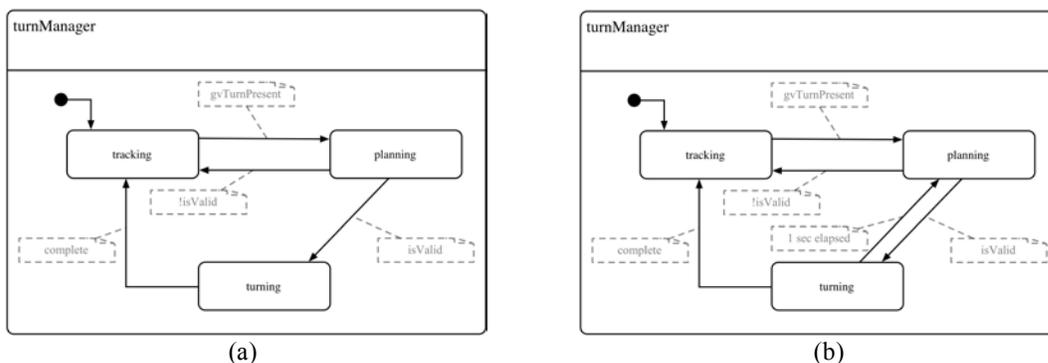


Figure 13: Turn manager. (a) Full plan, execution, and track. (b) Periodic replan during turn.

F. Further enhancements

Once the basic turn manager is executing, a series of improvements will be considered for implementation. Among the options are:

- Some means of wrapping around the boundary to find a shorter route
- Allowing transitory crossing of some kinds of keep-out zones so long as vehicle does not stop, which corresponds to a thermal-only constraint rather than an exposure constraint
- Generalize the keep-out zones to handle hazards other than solar exposure

G. Progress on implementation

As of time of writing, the following elements of the attitude profiler have been undertaken. A progress indicator for each is given.

- AutoGNC Ephemeris Services: complete, tested, and available
- Geometric analysis of beltway mathematical relationships between zones: complete
- Library of routines for performing needed calculations for beltway: written and undergoing testing
- Routines for determining pointing from specification: initial implementation undergoing testing
- Extension of VML to include class for target specification: under development
- Algorithm for entering and exiting beltway: under development
- Spline fitter: under development
- State machine graphical specification of turn manager: complete
- Implementation of turn manager in VML: under development
- Integrated testing with simulated system: not started

VI. Conclusions

Attitude profiling for spacecraft is time consuming and operationally intensive when done by ground-based specialists. Recent developments in processing power and flight software improvements have opened the door to solutions that are fast enough for on-board execution, transparent in operation, and relatively low effort to implement. VML sequencing extensions supporting matrix mathematics, state machines, and object-oriented programming are being used to implement attitude profiling for the Reactive Rendezvous Docking Sequencer in a way that will allow onboard autonomy to be taken to a new level, without the need for custom flight software development and its associated cost and schedule risk. This new operations paradigm for on-board attitude profiling will allow operators to specify high-level activities to turn the spacecraft, thereby saving effort, reducing personnel requirements, and lowering cost and risk.

Acknowledgments

The work described in this paper was carried out by Blue Sun Enterprises, Inc., under an agreement with the National Aeronautics and Space Administration, and administered by the Office of Chief Technologist as a Small Business Innovation Research grant.

References

Reports, Theses, and Individual Papers

¹Grasso, C. A., Lock, P. d., "VML Sequencing: Growing Capabilities over Multiple Missions", AIAA Space Operations Conference Proceedings, April 2008.

²Grasso, C. A., "The Fully Programmable Spacecraft: Procedural Sequencing for JPL Deep Space Missions Using VML (Virtual Machine Language)", IEEE Aerospace Applications Conference Proceedings, March 2002.

³Grasso, C. A., "Techniques for Simplifying Operations Using VML (Virtual Machine Language) Sequencing on Mars Odyssey and SIRTf", IEEE Aerospace Applications Conference Proceedings, March 2003.

⁴Peer, S. and Grasso, C. A., "Spitzer Space Telescope Use of Virtual Machine Language", IEEE Aerospace Conference Proceedings, December 2004.

⁵Grasso, C. A., "Virtual Machine Language (VML)", NPO 40365, JPL Commercial Programs Office, Innovative Technology Asset Management Group, Docket Date: 12-May-2003.

⁶Grasso, C. A., “Virtual Machine Language (VML) NASA Board Award”, NASA Inventions and Contributions Board, NASA Technical Report 40365, Award Date: September 7, 2006.

⁷Riedel, J. A., et. al., “AutoNav Mark 3: Engineering the Next Generation of Autonomous Onboard Navigation and Guidance”, AIAA Guidance, Navigation, and Control Conference, August 2006.

⁸Chapel, J. et. al., “Aerobraking Safing Approach for 2001 Mars Odyssey”, American Astronautics Society Guidance and Control Conference, Feb 2002.

⁹Riedel, J.E., Bhaskaran, S., et. al., “Navigation for the New Millennium: Autonomous Navigation for Deep Space-1,” Proceedings of the 12th International Symposium on Flight Dynamics, Darmstadt, Germany, June 1997

¹⁰Grover, M., Cichy, D., Dasai, P.N., “Overview of the Phoenix Entry, Descent and Landing System Architecture,” AIAA Paper AIAA 2006-7218, AIAA/AAS Astrodynamics Specialist Conference; Honolulu, HI, 18-21 August 2008.

¹¹Garcia, M., Fujii, K., “Mission Design Overview for the Phoenix Mars Scout Mission,” AAS Paper 07-247, AIAA/AAS Space Flight Mechanics Meeting; Sedona, AZ, 28 January -01 February 2007.

¹²Grasso, C. A., Riedel, J. E., Vaughn, A.T., “Reactive Sequencing for Autonomous Navigation Evolving from Phoenix Entry, Descent, and Landing”, AIAA Space Operations Conference Proceedings, April 2010.

¹³Kubitschek, D., Mastrodomos, N., et. al., “Deep Impact Autonomous Navigation: The Trials of Targeting the Unknown,” AAS 06-081, 29th Annual AAS Guidance and Control Conference, Breckenridge, Co., Feb. 4-8, 2006.

¹⁴Grasso, C. A., “Formal Methods for Design, Development, and Runtime: Runtime Verification of Distributed Reactive Systems Using DR-VIA and RTV with extended TTM/RTTL Notation.” Doctoral Thesis, University of Colorado, 1996.

¹⁵Balaram, J. et. al., “DSENDS - A High-Fidelity Dynamics and Spacecraft Simulator for Entry, Descent and Surface Landing”, IEEE Aerospace Conference, October 2001.

¹⁶D'Amario, L. A., Bollamn, W. E., et. al., “Mars Orbit Rendezvous Strategy for the Mars 2003/2005 Sample Return Mission”, Jet Propulsion Laboratory, California Institute of Technology, document 092407 May 2008.

¹⁷Riedel, J.E., et. al., “Optical Navigation Plan and Strategy for the Lunar Lander Altair; OpNav for Lunar and other Crewed and Robotic Exploration Applications”, AIAA-2010-7719, AIAA GN&C Conference, Aug. 2010, Toronto Canada

¹⁸R. Gaskell, “Landmark Navigation and Target Characterization in a Simulated Itokawa Encounter,” AAS/AIAA Astrodynamics Specialists Conference, Jet Propulsion Laboratory, Pasadena, CA, August 2005.

¹⁹R. Gaskell, “Small Body Simulations for Navigation Approach and Landing,” AIAA Space 2005, American Institute of Aeronautics and Astronautics, Long Beach, CA, August 2005.

²⁰Riedel, J. E., et. al, “Configuring the Deep Impact AutoNav System for Lunar, Comet and Mars Landing”, AIAA-2008-6940; AIAA/AAS Astrodynamics Specialist Conference; Honolulu, HI, 18-21 August 2008.

²¹Riedel, J.E., Bhaskaran, et. al., “Using Autonomous Navigation for Interplanetary Missions: The Validation of Deep Space 1 AutoNav,” IAA Paper L-0807, Fourth IAA International Conference on Low-Cost Planetary Missions, Laurel, Maryland, May 2000.

²²Bhaskaran, S., J. E. Riedel, B. Kennedy, T. C. Wang, “Navigation of the Deep Space 1 Spacecraft at Borrelly,” AIAA paper 2002-4815, AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Monterey, CA, August 5-8, 2002.

²³Bhaskaran, S., Mastrodomos, N., Riedel, J., Synnott, S., “Optical Navigation for the Stardust Wild 2 Encounter,” 18th International Symposium of Space Flight Dynamics, October 11-15 2004, Munich Germany.

²⁴Rasmussen, R.D., Singh, G., et al, “Behavioral model pointing on Cassini using target vectors,” Proceedings of SPIE, vol. 2803, p. 271, 1996.

²⁵Wong, E., Breckenridge, W., “An Attitude Control Design for the Cassini Spacecraft,” AIAA-95-3274, 1995.

²⁶Lisman, S., Chang, D., Singh, G., Fred H., Hadaegh, “Autonomous Guidance And Control Of A Solar Electric Propulsion Spacecraft”, AIAA GN&C Specialist Conference, New Orleans, LA, 17 August, 1997

Related web sites

Blue Sun Enterprises VML Website <http://www.bluesunenterprises.com>