Technical Notes

Discretized Quaternion Constrained Attitude Pathfinding

Henri C. Kjellberg*
Firefly Space Systems, Austin, Texas 78727
and
E. Glenn Lightsey†
Georgia Institute of Technology, Atlanta, Georgia 30332

DOI: 10.2514/1.G001063

Nomenclature

d_p	=	node-density parameter
$\boldsymbol{e}_v$	=	eigenvector
$f(n)$	=	future-path cost function
$g(n)$	=	past-path cost function at node n
n_t	=	total number of nodes on a given shell
p	=	pixel number
$\underline{\boldsymbol{q}}$	=	unit quaternion
$\boldsymbol{w}$	=	unit vector of constraint direction in external frame
$\boldsymbol{v}$	=	unit vector of constraint axis in body frame
Θ	=	half-cone angle describing size of the constraint
θ	=	angular distance between two quaternions
ϕ	=	eigenangle

I. Introduction

CONSTRAINED attitude pathfinding is the guidance task of finding a desired rotation trajectory for an object pointed toward an initial attitude to rotate it to a final attitude, while simultaneously satisfying path constraints. In the case of reorienting a spacecraft, which is considered in this Note, Fig. 1 shows some typical constraints. Keep-out constraints may be associated with avoidance criteria, such as not pointing a sensitive instrument within a certain angle of the sun, which might cause damage to the device. Keep-in constraints may be related to the fields of view of certain sensors, for example, to keep a sun sensor within an angle of the sun. Body rate constraints can be imposed by the saturation of an inertial measurement unit (IMU).

Microsatellites like SpaceDev's Trailblazer have used constrained attitude pathfinding to reduce the cost of the spacecraft bus by satisfying sensor constraints through attitude guidance [1]. The rise of low-cost multispacecraft missions, such as the Planet Labs' Flock-1, necessitates increasing the autonomous capability of the spacecraft to minimize direct human operator intervention [2].

Received 19 September 2014; revision received 22 June 2015; accepted for publication 25 August 2015; published online 11 November 2015.

*Guidance, Navigation, and Control Engineer; hck@utexas.edu.
†Professor, School of Aerospace Engineering; glenn.lightsey@gatech.edu. Fellow AIAA.

II. Existing Attitude Pathfinding Methods

Different approaches have been created to solve specific formulations of the constrained attitude pathfinding problem. The avoidance cone constrained attitude pointing problem has been explored by McInnes [3], Hablani [4], Frazzoli et al. [5,6], and Kim et al. [7,8]. Previous work by the authors provides a survey of these methods as well as a new approach for attitude pathfinding of a pointing vector using a discretized shell of unit radius [9].

A. Discretized Pointing Vector Pathfinding

In the prior method, discretized nodes on a pixelated unit shell are used to represent possible directions for the boresight vector of a sensitive instrument axis in the external reference frame. Keep-out and keep-in constraints are defined for a single vector in the spacecraft body frame, representing the body-fixed direction of the instrument boresight vector. Pointing constraints are associated with specific directions in the external frame, which must be excluded from or included in the acceptable solution path depending on the type of constraint. Discretization is achieved using a subroutine known as icosahedron [10]. Once the unit shell has been discretized, a trajectory is found using a graph pathfinding algorithm known as A^* [11].

B. Limitations of Prior Algorithms

The prior approach only considers a single body-fixed direction in obtaining a solution. Because the solution determines the trajectory of a single boresight vector, rotations about that axis are not specified. Third-axis attitude constraints, possibly imposed by additional sensors, must be incorporated by rotating about the primary sensor's boresight axis.

III. Discretized Quaternion Pathfinding

A new method titled discretized quaternion constrained attitude pathfinding has been created to generalize the algorithm and accommodate multiple constraints. The new method also uses the icosahedron shell discretization technique and the A^* pathfinding algorithm, but instead applies them to a quaternion representation of attitude. By changing the way that the problem is formulated, an arbitrary number of sensitive directions distributed throughout the body frame can be paired with an arbitrary number of constraints in the external frame.

A. Discretization of Attitude Quaternion

Discretization of the attitude quaternion is a preliminary step that is conducted before the execution of the pathfinding algorithm. The discretized quaternion guidance algorithm uses the attitude quaternion represented as

$$\underline{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{e}_v \sin(\phi/2) \\ \cos(\phi/2) \end{bmatrix} \tag{1}$$

Here, $\boldsymbol{e}_v$ represents the eigenvector and ϕ is the eigenangle. The discretization is achieved by creating a series of concentric shells of varying radius. The normalized vector from the origin to a node on any given shell is $\boldsymbol{e}_v$; the length of the vector is ϕ. Every node in the discretization represents a full three-axis attitude quaternion.

The goal of the discretization scheme is to evenly distribute nodes throughout the entire attitude space such that the eigenangle for a rotation between any two neighboring nodes is approximately constant. The icosahedron subroutine developed by Tegmark distributes

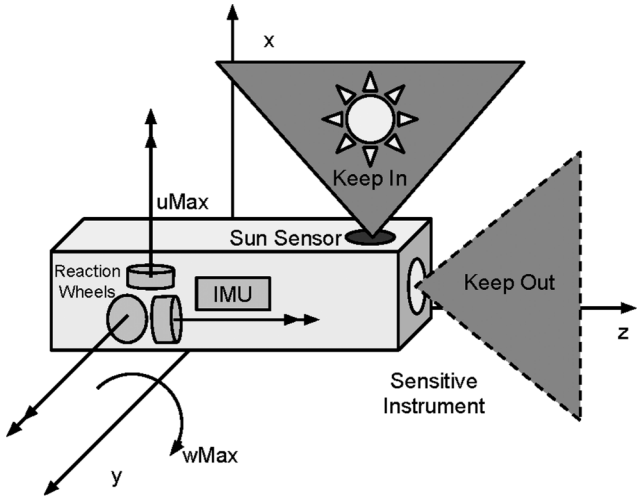

Fig. 1 Common spacecraft sensor and actuator attitude constraints.

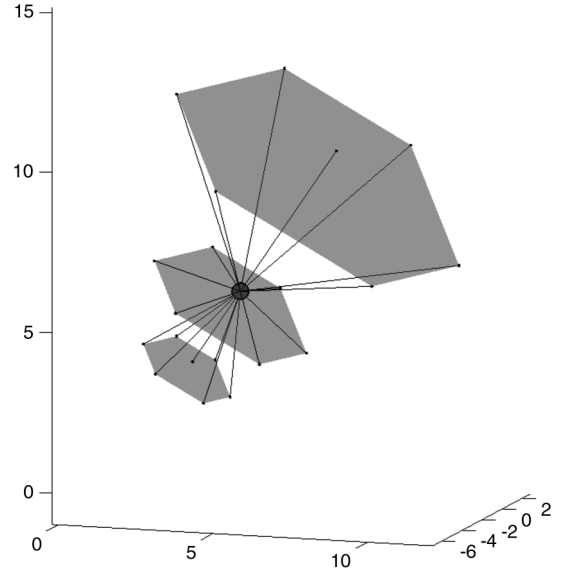

Fig. 2 Node at center indicated by large gray circle has 20 neighboring nodes shown as points connected with a solid line.

nodes approximately evenly across each shell. Full details of the icosahedron subroutine are found in [10].

The total number of nodes on a given shell can be controlled by a node-density parameter $d_p \in \mathbb{N} \geq 2$. The total number of nodes on a shell for a given node-density parameter is

$$n_t = 40d_p(d_p - 1) + 12 \tag{2}$$

Each node has a total of up to 20 neighboring nodes. Figure 2 shows the typical neighbor distribution of a node. The neighbors include the six neighbors on the same shell, seven nodes on the shell layer above, and seven nodes on the shell layer below.

The angular distance θ between two neighboring nodes with quaternions specified by $\underline{\boldsymbol{q}}_a$ and $\underline{\boldsymbol{q}}_b$ is given by

$$\theta = \arccos[2(\underline{\boldsymbol{q}}_a \cdot \underline{\boldsymbol{q}}_b)^2 - 1] \tag{3}$$

The rotation angle between a given node and each of the six neighboring nodes on the same shell, as well as the nodes directly above and below, is kept approximately equal by appropriately discretizing the quaternion parameters $\boldsymbol{e}_v$ and ϕ. The angular distance between any two neighboring nodes that reside on the same shell is dependent on the eigenangle of the shell ϕ. The shells that are closer to the origin, with small eigenangles, yield smaller angular distances for

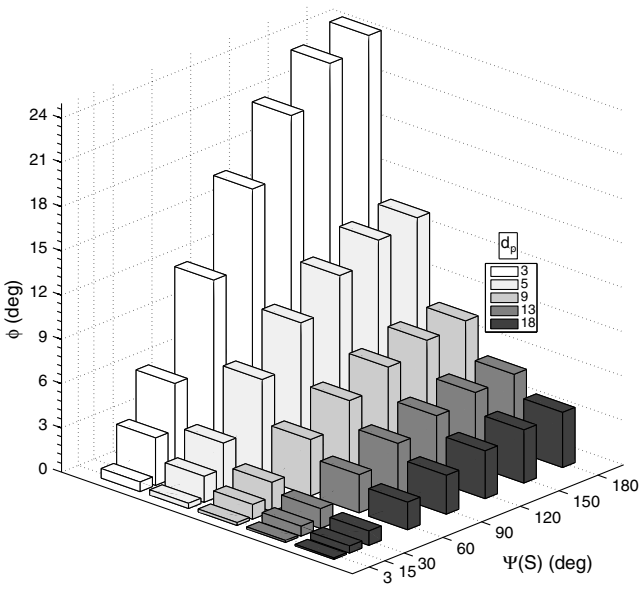

Fig. 3 Increasing node-density parameter d_p, for a given shell, decreases angular distance $\Psi(S)$.

neighboring nodes on the same shell. Therefore, it is desirable to discretize the shells that are closest to the origin more coarsely and the shells that are farther away from the origin more finely so that overall angular distance between neighboring nodes is kept approximately constant.

Let all the quaternions on a given shell be members of the set S:

$$\underline{\boldsymbol{q}}_i \in S = \{\underline{\boldsymbol{q}}_1, \underline{\boldsymbol{q}}_2, \ldots, \underline{\boldsymbol{q}}_n\} \tag{4}$$

All quaternions on S have the same eigenangle. Let the set N have all the neighboring nodes of the quaternion $\underline{\boldsymbol{q}}_i$ on the same shell. The cost function $\Psi(S)$ is formed to find the maximum angle between any quaternion in S and its closest neighbor $\underline{\boldsymbol{q}}_j$ [12]:

$$\Psi(S) = \max_{\underline{\boldsymbol{q}}_i \in S}\left(\min_{\underline{\boldsymbol{q}}_j \in N}\{\arccos[2(\underline{\boldsymbol{q}}_i \cdot \underline{\boldsymbol{q}}_j)^2 - 1]\}\right) \tag{5}$$

The distance between the closest neighbor on a shell above or below a given node is nearly constant because the eigenangle is chosen for each shell and the eigenvectors for the neighboring nodes on separate shells are approximately collinear. The eigenvectors are not perfectly collinear for shells above or below that are discretized with different node-density parameters d_p, however, the effect on total distance is small. The primary focus is to make sure the neighboring nodes on the same shell maintain a consistent distance throughout the full range of eigenangles. Therefore, the goal is to specify a value for d_p that matches $\Psi(S)$ as closely as required to the desired angular distance θ between the neighboring nodes.

Figure 3 shows the relationship between node-density parameter d_p, eigenangle ϕ, and the resulting angular distance $\Psi(S)$ between any two neighboring nodes on the same shell.

The level of discretization that is required for the guidance algorithm is determined by the desired angular resolution of the resulting attitude path as given by θ. To reduce the computational effort needed to achieve a solution, the shell discretization is set as coarsely as possible while achieving the desired level of angular resolution. For the example shown in this work, the desired node distance is set to approximately 3 deg. Therefore, a total of 60 shells are formed with the maximum eigenangle $\phi = 180$ deg. Figure 4 shows the resultant node-density parameters d_p versus shell numbers N to produce an approximately constant $\theta = 3$ deg angular separation.

B. Pathfinding Approach

One general purpose pathfinding algorithm is known as A^*. The block diagram in Fig. 5 summarizes the A^* pathfinding algorithm.

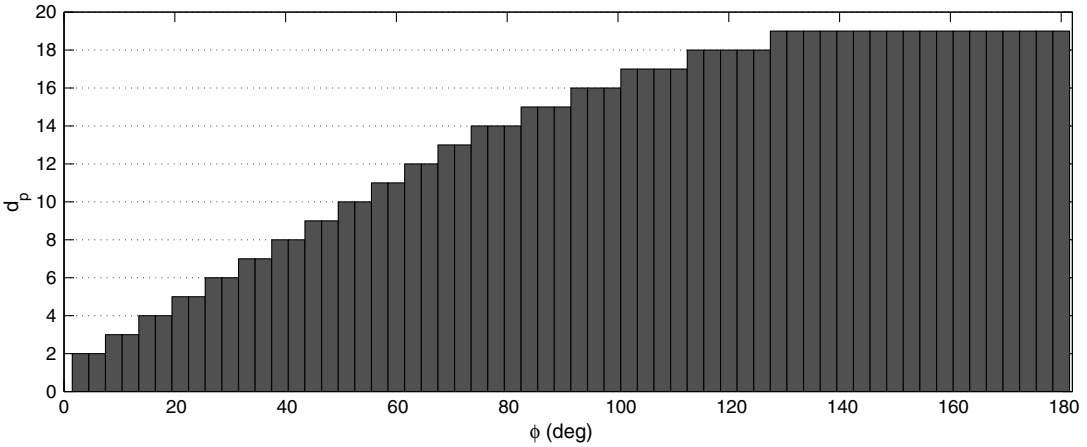

Fig. 4 For an eigenangle ϕ, the node-density parameter d_p is found.

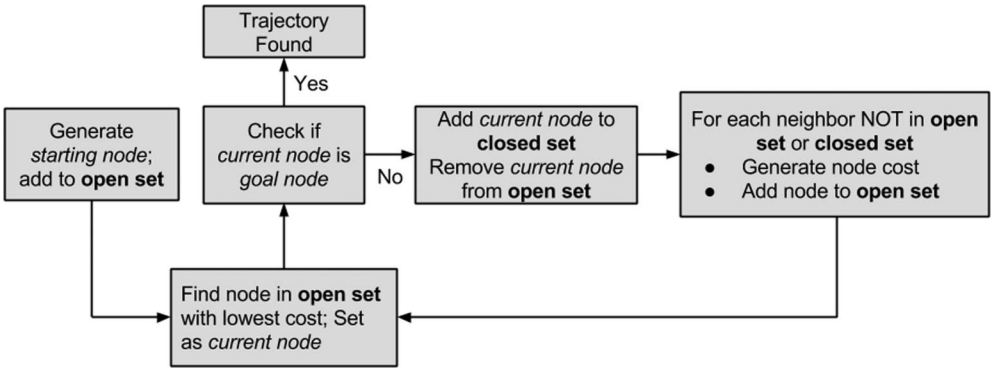

Fig. 5 Pathfinding algorithm A^*.

The algorithm is based on the concept of minimizing a total path cost that is the sum of node costs for each node in the path (details can be found in [11]).

There are two costs associated with the nodes used in finding the minimum total path cost. These are the past-path cost function and the future-path cost function. The past-path cost function, denoted as $g(x)$, estimates the total distance required to go from the initial attitude to a considered node. If the node is within a constraint, the cost is set arbitrarily high. The future-path cost function, denoted as $f(n)$, is an estimate of the remaining cost to get from a considered node to the goal node. The future-path cost causes the algorithm to favor nodes that reduce the distance to the goal node. The total cost for each considered node is the sum of the future-path and past-path cost functions. There are three steps to evaluating the cost for each considered node.

1. Get Node Identification from Neighbor List and Form Quaternion

Each node is identified by two parameters, the eigenangle ϕ that indicates shell radius and an integer known as the pixel number p:

$$\text{Node ID} := \begin{cases} p \\ \phi \end{cases} \tag{6}$$

A precomputed mapping of ϕ to d_p (as shown in Fig. 4, for example) provides the desired angular distance between neighboring nodes throughout the quaternion space. The icosahedron subroutine is used to compute the eigenvector $\boldsymbol{e}_v$ with d_p and p as inputs. After forming a starting node and goal node from their respective quaternions through the icosahedron subroutine, each subsequent node is identified by looking up the node identification (ID) from a precomputed list of neighbors.

2. Evaluate Constraints

The constraint evaluation uses the quaternion representation of attitude [7,8]. The attitude constraint is tested using

$$\underline{\boldsymbol{q}}^T(n) C_i(\boldsymbol{w}_i, \boldsymbol{v}_i, \Theta_i) \underline{\boldsymbol{q}}(n) \leq 0 \tag{7}$$

Here, n represents the node ID being evaluated, and $\underline{\boldsymbol{q}}$ is the quaternion attitude based on n. A constraint definition matrix C_i is created for each of the i body and external reference constraint vector pairs; $\boldsymbol{v}_i$ is the unit vector of the constraint axis expressed in the body-fixed reference frame, $\boldsymbol{w}_i$ is the unit vector of the constraint direction expressed in the external frame; and Θ_i is the half-cone angle describing the size of the constraint. C_i is determined by

$$C_i(\boldsymbol{w}_i, \boldsymbol{v}_i, \Theta_i) = \begin{bmatrix} A_i & b_i \\ b_i^T & d_i \end{bmatrix} \tag{8}$$

$$A_i = \boldsymbol{v}_i \boldsymbol{w}_i^T + \boldsymbol{w}_i \boldsymbol{v}_i^T - (\boldsymbol{v}_i^T \boldsymbol{w}_i + \cos\Theta_i) I_{3\times 3} \tag{9}$$

$$\boldsymbol{b}_i = \boldsymbol{w}_i \times \boldsymbol{v}_i \tag{10}$$

$$\boldsymbol{d}_i = \boldsymbol{v}_i^T \boldsymbol{w}_i - \cos\Theta_i \tag{11}$$

If Eq. (7) is true for the quaternion being tested, then the quaternion $\underline{\boldsymbol{q}}$ is outside the cone specified by $\boldsymbol{w}_i$, $\boldsymbol{v}_i$, Θ_i.

3. Compute Node Costs

The past-path cost function is formed by first evaluating all the keep-in and keep-out constraints:

Table 1 Simulation constraint settings

Body frame	External frame	Half-angle size, deg	Type
$v_{ssx} = [1, 0, 0]$	$w_{sun} = [-0.0995, 0.9950, 0.0000]$	70	Keep in
$v_{ssy} = [0, 1, 0]$	$w_{sun} = [-0.0995, 0.9950, 0.0000]$	70	Keep in
$v_{st} = [0, 0, 1]$	$w_{sun} = [-0.0995, 0.9950, 0.0000]$	30	Keep out
$v_{st} = [0, 0, 1]$	$w_{faint} = [0.0711, 0.0711, 0.9949]$	15	Keep out
$v_{st} = [0, 0, 1]$	$w_{Earth} = [-0.7071, -0.7071, 0.0000]$	85	Keep out

$$c_i^{\text{in}} = \begin{cases} 1, & \underline{\boldsymbol{q}}^T(n) C_i(\boldsymbol{w}_i, \boldsymbol{v}_i, \Theta_i) \underline{\boldsymbol{q}}(n) \le 0, \\ 0, & \underline{\boldsymbol{q}}^T(n) C_i(\boldsymbol{w}_i, \boldsymbol{v}_i, \Theta_i) \underline{\boldsymbol{q}}(n) > 0 \end{cases} \tag{12}$$

$$c_i^{\text{out}} = \begin{cases} 1, & \underline{\boldsymbol{q}}^T(n) C_i(\boldsymbol{w}_i, \boldsymbol{v}_i, \Theta_i) \underline{\boldsymbol{q}}(n) > 0, \\ 0, & \underline{\boldsymbol{q}}^T(n) C_i(\boldsymbol{w}_i, \boldsymbol{v}_i, \Theta_i) \underline{\boldsymbol{q}}(n) \le 0 \end{cases} \tag{13}$$

where c_i^{in} represents the ith keep-in constraint and c_i^{out} represents the ith keep-out constraint. If a constraint is violated, the equation produces a one. If the constraint is not violated, the result is zero. The past-path cost function is then computed as

$$g(n) = g(n-1) + \kappa_i \sum_{i=1}^{k_{\text{in}}} c_i^{\text{in}} + \psi_i \sum_{j=1}^{k_{\text{out}}} c_i^{\text{out}} + \Gamma \tag{14}$$

Here, $g(n-1)$ represents the past-path cost from the parent node to this node. Therefore, the past-path cost accumulates as the trajectory progresses forward. Each constraint that is violated adds to the cost of the node. The constants κ_i and ψ_i are set such that each violated cost adds significantly to the total cost. The residual cost Γ is the average distance between nodes.

In this implementation, the future-path cost function is simply the eigenangle distance between this node's quaternion $\underline{\boldsymbol{q}}(n)$ and the goal node $\underline{\boldsymbol{q}}(g)$, which can be found using Eq. (3). Once the past- and future-path costs have been determined, the two costs are added together resulting in the total cost of traveling through this node $t(n)$:

$$t(n) = g(n) + f(n) \tag{15}$$

IV. Example Scenario

An example spacecraft reorientation scenario is presented with a set of complex but realistic attitude constraints.

A. Description of Constraints

The spacecraft has a star tracker with a field of view of 15 deg pointing along its z axis in the body frame. A pair of sun sensors point in the positive x axis and y axis, respectively. Each sun sensor has a cone half-angle field of view of 70 deg. The attitude constraints for the star tracker include 1) an Earth keep-out constraint of 85 deg, 2) a sun keep-out constraint of 30 deg, and 3) a faint constellation keep-out constraint of 15 deg.

Notionally, the star tracker would be unable to produce navigation solutions when the Earth or the sun is within its field of view due to detector saturation. Additionally, holding the sun within the field of view too long could damage the detector. The dim constellation constraint is applied because the star tracker is a low-cost device and is not guaranteed to work when viewing some external directions that do not have bright stars.

Additionally, the sun sensors must keep the sun within their 70 deg cone half-angle field of view. The sun sensor keep-in constraint can be applied in a manner that this constraint is violated only if neither sun sensor can see the sun; if a single sun sensor can see the sun, then the keep-in constraint is satisfied. The constraints are summarized in Table 1.

B. Result

The scenario begins in an orientation condition with all attitude constraints initially satisfied. The initial attitude quaternion is $\underline{\boldsymbol{q}}_i = [0.5990, 0.6272, 0.4252, 0.2588]^T$ and the final desired attitude quaternion is $\underline{\boldsymbol{q}}_f = [-0.1153, -0.1587, 0.1265, 0.9274]^T$, which represents an eigenaxis rotation of 164.28 deg.

The trajectory returned by the pathfinding algorithm is shown in Fig. 6. The initial directions of the vehicle's body x, y, and z axes in the external reference frame are shown as circles. The final directions for these axes are shown as crosses. For reference, the star tracker boresight is pointed along the vehicle's body z axis. In this case, the sun has risen across the Earth's horizon with a distance between the horizon and sun that is large enough to allow the star tracker (vehicle z axis) to transit in between the Earth and the sun so that both

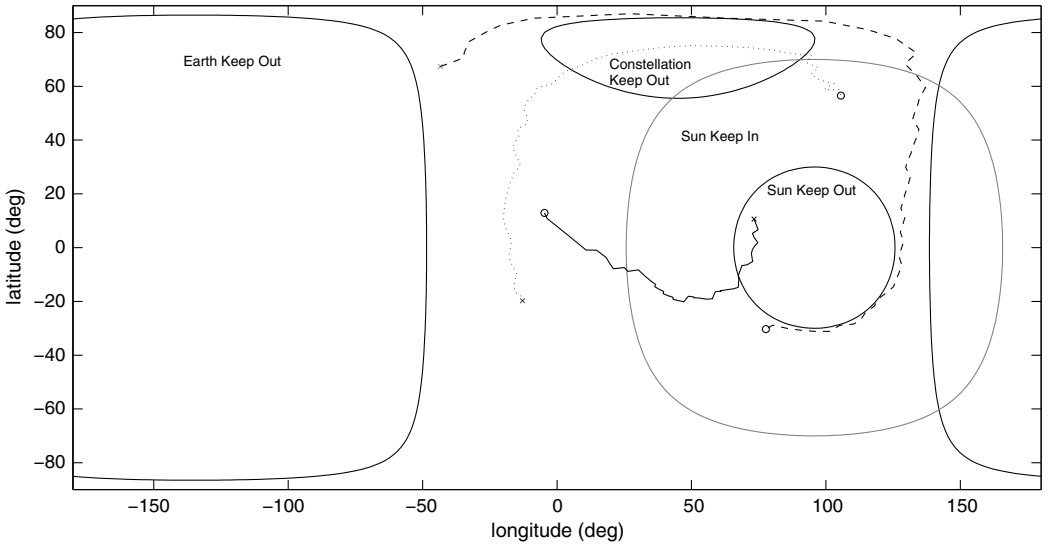

Fig. 6 Spacecraft reorientation maneuver example scenario. Dotted line represents direction of vehicle body x axis. Solid line represents y axis. Dashed line represents z axis.

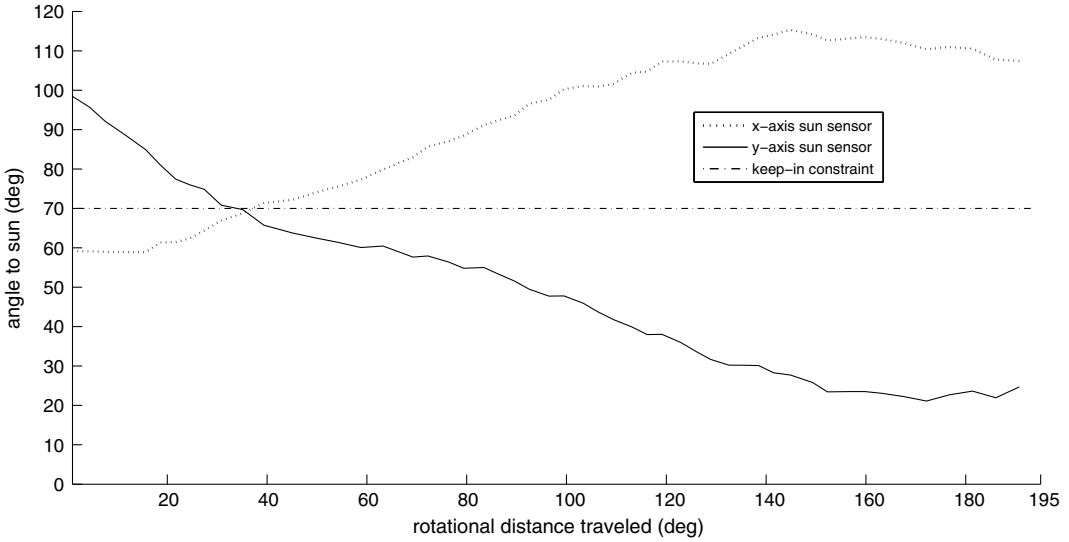

Fig. 7 Angles between each sun sensor and sun vector throughout maneuver. Sun sensors swap responsibility for viewing sun to satisfy keep-in constraint.

constraints are met. The faint constellation constraint is located in the northern hemisphere near the pole.

Note that the sun sensor on the vehicle x axis (dotted line) holds its position within the field of the view of the sun, loitering near the edge of the keep-in constraint until the sun sensor on the y axis (solid line) enters the keep-in constraint. Once the y-axis sun sensor has entered the region, the x-axis sun sensor is free to move out to allow the reorientation maneuver to be completed. Figure 7 shows this transition in detail by displaying the angle between the sun vector and the sun sensor boresights over time. Meanwhile, the star tracker navigates through the channel between the sun and the Earth's horizon as it moves toward the pole. Once at the pole, the star tracker is able to slew around the constellation keep-out constraint and to the final commanded orientation. Every constraint is satisfied throughout the entire maneuver.

The trajectory is globally optimal in distance within the context of the discretized graph [11]. The attitude trajectory does not take into account the actuation capabilities of the vehicle nor does it take into account the dynamics of the external constraints.

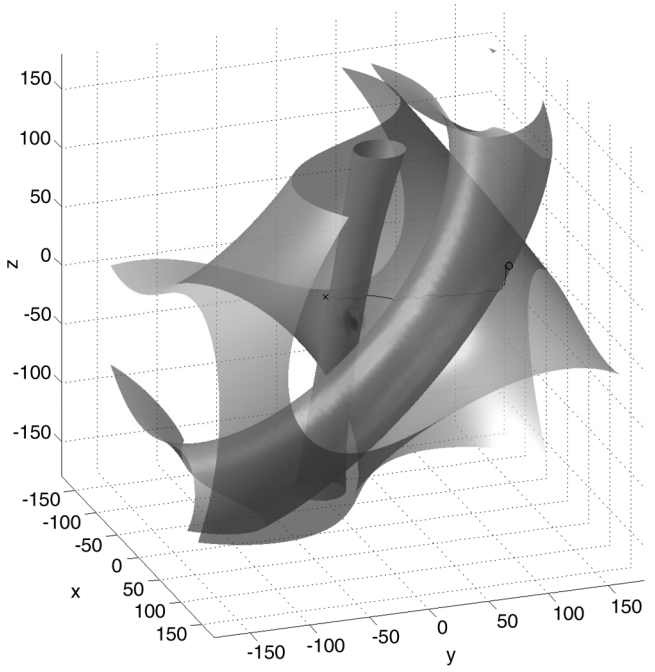

Fig. 8 Three keep-out constraints shown in three-dimensional view of quaternion space. Initial attitude is given by "o" and final attitude by "x."

C. Quaternion Perspective

Viewed in the quaternion space, the scenario produces three keep-out constraint tubes for the sun, Earth, and faint constellation keep-out constraints. Figure 8 shows these three constraints and the trajectory that satisfies each. Each axis represents the x, y, and z components of the quaternion's eigenvector scaled by the eigenangle (in degrees). Note that the keep-in constraints are hidden for clarity.

The large-diameter curved tube in the back of the plot represents the Earth constraint. The narrow tube going from the top to the bottom near the origin of the plot represents the faint constellation keep-out constraint. The medium-sized tube curving through the right side of the plot represents the sun keep-out constraint. The tubes are slightly transparent to show the trajectory passing behind the sun constraint. Here, the quaternion trajectory is seen to pass through the narrow gap in between the sun constraint and the Earth constraint.

V. Conclusions

The new constrained attitude pathfinding algorithm has been generalized by using a quaternion formulation of the attitude pathfinding problem. The limitations of the previous implementation have been removed, allowing for any number of body-fixed attitude constraints and directions to be supported by a single algorithm. The new approach is also capable of satisfying other types of constraints, such as a constraint that is met as long as a single sensor in a group of sensors is able to view a specified external direction.

Acknowledgments

This work was supported by a NASA Office of the Chief Technologist's Space Technology Research Fellowship grant (NNX11AN26H).

References

[1] Koenig, J. D., "A Novel Attitude Guidance Algorithm for Exclusion Zone Avoidance," *IEEE Aerospace Conference*, Vol. 38, IEEE Publ., Piscataway, NJ, March 2009, pp. 1–10.

[2] Niles, L., "Largest Flock of Earth-Imaging Satellites Launch into Orbit from Space Station," NASA, 2014, http://www.nasa.gov/mission_pages/station/research/news/flock_1/#.VgrPOuxVhBc [retrieved 22 March 2014].

[3] McInnes, C. R., "Large Angle Slew Maneuvers with Autonomous Sun Vector Avoidance," *Journal of Guidance, Control, and Dynamics*,

Vol. 17, No. 11, 1994, pp. 875–877.
doi:10.2514/3.21283

[4] Hablani, H. B., "Attitude Commands Avoiding Bright Objects and Maintaining Communication with Ground Station," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 6, 1999, pp. 759–767.
doi:10.2514/2.4469

[5] Frazzoli, E., Dahleh, M. A., Feron, E., and Kornfeld, R., "A Randomized Attitude Slew Planning Algorithm for Autonomous Spacecraft," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2001-4155, 2001.

[6] Frazzoli, E., Dahleh, M. A., Feron, E., and Kornfeld, R., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129.
doi:10.2514/2.4856

[7] Kim, Y., Mesbahi, M., Singh, G., and Hadaegh, F. Y., "On the Constrained Attitude Control Problem," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2004-5129, Aug. 2004.

[8] Kim, Y., Mesbahi, M., Singh, G., and Hadaegh, F. Y., "On the Convex Parameterization of Constrained Spacecraft Reorientation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 46, No. 3, July 2010, pp. 1097–1109.
doi:10.1109/TAES.2010.5545176

[9] Kjellberg, H. C., and Lightsey, E. G., "Discretized Constrained Attitude Pathfinding and Control for Satellites," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 5, 2013, pp. 1301–1309.
doi:10.2514/1.60189

[10] Tegmark, M., "An Icosahedron-Based Method for Pixelizing the Celestial Sphere," *Astrophysical Journal Letters*, Vol. 470, No. 2, Oct. 1996, pp. 81–84.
doi:10.1086/310310

[11] Hart, P. E., Nilsson, N. J., and Raphael, B., "Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107.
doi:10.1109/TSSC.1968.300136

[12] Lovisolo, L., and da Silva, E., "Uniform Distribution of Points on a Hyper-Sphere with Applications to Vector Bit-Plane Encoding," *IEEE Proceedings on Vision, Image and Signal Processing*, Vol. 148, No. 3, June 2001, pp. 187–193.
doi:10.1049/ip-vis:20010361