Trajectory Optimization for Satellite Reconfiguration Maneuvers with Position and Attitude Constraints

Ian Garcia and Jonathan P. How
Aerospace Controls Laboratory
Massachusetts Institute of Technology
{ianmga, jhow}@mit.edu

Abstract— **This paper presents an algorithm for designing spacecraft reconfiguration maneuvers, which is a difficult 6 DOF trajectory design problem with nonlinear attitude dynamics and non-convex constraints. In addition, some of the constraints couple the positions and attitudes of the spacecraft, which makes the problems high-dimensional. The essential feature of the design methodology is the separation into a simplified path planning problem obtaining a feasible solution, then improving it significantly by a smoothing operation. The first step is solved using a Rapidly-exploring Random Tree. The smoother then optimizes the trajectories by iteratively solving a linear program using a linearization of the cost function, dynamics, and constraints about the initial feasible solution. Examples are presented to demonstrate the validity of the approach for complex problems with four spacecraft.**

I. Introduction

Formation flying of multiple spacecraft is an attractive technology for many planned space science missions [2]. To achieve the full science objectives, these missions typically require a complex initialization maneuver and/or formation reconfiguration maneuvers to reorient the fleet. These maneuvers consist of moving and rotating a group of N spacecraft from an initial configuration to a desired target configuration, while satisfying an arbitrary number of constraints (e.g., see Figure 1). These constraints may consist of collision avoidance, restrictions on the region of the sky where certain spacecraft instruments can point (e.g., a sensitive instrument that cannot point at the Sun), or restrictions on pointing towards other spacecraft (e.g., requirements on maintaining inter-spacecraft communication links and having cold science instruments avoid high temperature components on other vehicles). It is also desirable to perform these maneuvers with the lowest possible fuel expenditure.

This problem is particularly difficult due to the nonlinearity of the attitude dynamics, the non-convexity of some of the constraints, and the coupling between the positions and attitudes of all spacecraft. The nonlinearity of the spacecraft attitude dynamics makes the attitude trajectory design problem difficult in itself. Although numerous solutions exist for this problem, this nonlinearity adds considerable difficulty to the general spacecraft reconfiguration problem. The non-convex constraints place this problem in a general class of path planning problems with obstacles that are known to be NP-hard and thus computationally difficult [3]. Finally, some of the pointing constraints introduce coupling between the positions and attitudes of the entire fleet. As a result, the trajectory design must be solved as a single $6N$ DOF problem instead of N separate 6 DOF problems.

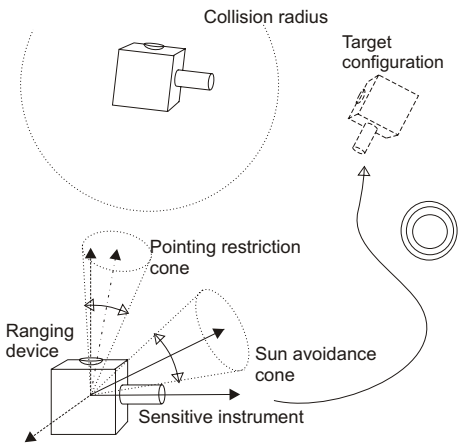

Fig. 1. The formation reconfiguration problem

A. Background

The spacecraft trajectory design problem for the unconstrained translation and attitude maneuvers has been the subject of extensive research with successful results. In contrast, the trajectory design problem with constraints is more difficult and has attracted attention more recently. Most of the solutions for this problem find either the translation or the attitude trajectories. Some of these solutions include potential functions methods [4], [5], geometric/heuristic methods [6], Mixed-Integer Linear Programming [7], and randomized algorithms [8], [9]. The trajectory design problem with combined translation and attitude has also been investigated. Some recent solutions are based on geometric/heuristic methods [10] and randomized algorithms [11]. However, the geometric and heuristic methods are problem specific and cannot be extended to solve the general reconfiguration problem. The randomized algorithms were used to solve a problem with a single spacecraft and no pointing constraints. The reconfiguration problem addressed in this paper is more general and on a larger scale than the problems considered before. For example, we design maneuvers for 4 spacecraft with 24 DOF. The following sections outline the solution technique developed to solve these problems, followed by several demonstrations.

II. Solution Approach

The approach chosen to tackle the computational difficulty of this problem is to concentrate first on finding any feasible trajectory. The second part then focuses on improving the solution while maintaining feasibility. These two parts are called the *path planner* and the *smoother* in this paper. This separation is a natural approach that has been used many times in difficult path planning problems (and recently for spacecraft trajectory design [11]).

A. Path Planner

The problem of finding a feasible path between two points with an arbitrary number of constraints is a classical path planning problem that has been studied for many years. The difficulty of this problem makes it intractable to find a solution with guarantees for problems with more than a few (e.g., 3–6) dimensions. However, by relaxing the guaranteed completion, randomized path planning algorithms such as the Probabilistic Roadmaps (PRM) have been successful in solving larger problems [12]. Rapidly-exploring Random Trees (RRT), a more recent variant of these randomized planners, is used in this paper because it performed better than other algorithms in our experimental comparisons [1].

The path planning problem is posed without differential constraints. In general these additional constraints limit the expansion of the randomized search trees and increase the solution time. Instead, the direct trajectories between two points consist of rest-to-rest straight-line translations and eigen-axis rotations at constant rates. Since the spacecraft is at rest at each node of the search tree, a branch of the search tree can grow in any direction. The algorithm generates a trajectory consisting of a sequence of states, connected by these feasible "direct trajectories". This trajectory is then passed to the *smoother*.

B. Smoother

The *smoother* improves the cost of the trajectory previously found. This trajectory is represented by the states and control inputs sampled at fixed time-steps. The smoothing problem is posed as a nonlinear optimization with nonlinear constraints. The cost function, dynamics and constraints are then linearized, and the resulting equations are solved as a linear program. These steps are repeated iteratively until the cost associated with the trajectory cannot be improved.

After each iteration the solution may violate some of the constraints by a small amount, so there is an additional step to recover a feasible solution. This step uses knowledge specific to the trajectory design problem, and is the main difference between the smoother and a similar algorithm, the Sequential Linear Programming method (SLP) [13]. The recovery step is discussed further in Section III-C.

This algorithm is also different from other trajectory smoothing algorithms proposed in the literature because it improves the cost of the trajectory as a whole, it is deterministic, and always remains feasible [11], [14].

III. The Reconfiguration Algorithm

A. Problem formulation

The general reconfiguration problem consists on finding a trajectory, represented by the state and control inputs of N spacecraft, from time 0 to T. This state consists of

$$x_i(t) \in R(3), \quad \dot{x}_i(t) \in R(3) \tag{1}$$

$$q_i(t) \in SO(3), \quad \omega_i(t) \in R(3) \tag{2}$$

where $i \in 1 \dots N$ indicates the spacecraft. $x_i(t)$ is the position of its center, $\dot{x}_i(t)$ its velocity, and $\omega_i(t)$ its angular velocity, all measured with respect to a local inertially fixed frame. The attitude quaternion is $q_i(t)$. The input consists of

$$u_i(t) \in R(3) \text{ and } M_i(t) \in R(3) \tag{3}$$

where $u_i(t)$ are the forces and $M_i(t)$ the moments. Let $p_i(t)$ be a point of the trajectory of a single spacecraft at time t,

$$p_i(t) = \left[x_i(t), \dot{x}_i(t), u_i(t), q_i(t), \omega_i(t), M_i(t)\right], \tag{4}$$

for $i \in 1 \dots N$, and

$$p(t) = [\cdots, p_i(t), \cdots] \tag{5}$$

a point in the composite trajectories of all the spacecraft at time t.

For the deep space missions considered, the translational dynamics are approximated with a double integrator

$$\begin{bmatrix} \dot{x}_i(t) \\ \ddot{x}_i(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_i(t) \\ \dot{x}_i(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i(t) \tag{6}$$

and the attitude dynamics, in quaternion notation, are

$$\dot{q}_i(t) = \frac{1}{2}\Omega_i(t) q(t) \tag{7}$$

$$J\dot{\omega}_i(t) = -\omega_i(t) \times (J\omega_i(t)) + M_i(t) \tag{8}$$

where

$$\Omega_i(t) = \begin{bmatrix} 0 & w_{i,3}(t) & -w_{i,2}(t) & w_{i,1}(t) \\ -w_{i,3}(t) & 0 & w_{i,1}(t) & w_{i,2}(t) \\ w_{i,2}(t) & -w_{i,1}(t) & 0 & w_{i,3}(t) \\ -w_{i,1}(t) & -w_{i,2}(t) & -w_{i,3}(t) & 0 \end{bmatrix} \tag{9}$$

and J is the inertia matrix, which is considered to be the same for all spacecraft for simplicity. The collision avoidance constraints are

$$\|x_i(t) - x_j(t)\| \geq R \tag{10}$$

for $i, j \in 1 \dots N$, $i \neq j$, and R is the minimum distance allowed between the centers of spacecraft. The *absolute stay outside* pointing constraints are given by

$$z_k^T y_k \leq \cos\theta_k \tag{11}$$

where $k \in 1 \dots N_c$ identifies a constraint. This condition ensures that the spacecraft vector y_k remains at an angle greater than $\theta_k \in [0, \pi]$ from the inertial vector z_k. The vector y_k is the representation in the inertial coordinate

frame of the body vector y_{kB}. The transformation of coordinates is given by

$$y_k = y_{kB} - 2(q_{i0}^T q_{i0})y_{kB} + 2(q_{i0}^T y_{kB})q_{i0} - 2q_{i4}(y_{kB} \times q_{i0}) \tag{12}$$

where $q_{i0}(t)$ and $q_{i4}(t)$ are defined as

$$q_i(t) = [q_{i1}(t), q_{i2}(t), q_{i3}(t), q_{i4}(t)]^T = [q_{i0}(t), q_{i4}(t)]^T$$

The *absolute stay inside* pointing constraints only differ by the sign of the equation

$$z_k^T y_k \geq \cos\theta_k \tag{13}$$

The inter-spacecraft *relative stay outside* pointing constraints are given by

$$\hat{x}_{ij}^T y_k \leq \cos\theta_k \tag{14}$$

where y_k and θ_k are as above, and $\hat{x}_{ij}(t) = (x_j(t) - x_i(t))/(\|x_j(t) - x_i(t)\|)$ is the unit vector that points from spacecraft i to spacecraft j. Similarly, the *relative stay inside* pointing constraints are

$$\hat{x}_{ij}^T y_k \geq \cos\theta_k \tag{15}$$

For this problem, the cost to minimize is

$$J = \sum_{i=1}^{N} \int_0^T |u_i(t)| + |M_i(t)|\, dt \tag{16}$$

B. The planner

The randomized path planning algorithm consists of LaValle's randomized dense tree, in particular the bidirectional variant as described in [15]. The algorithm is reproduced here, but interested readers are encouraged to consult references [1] or [15].

```
RDT-BALANCED-BIDIRECTIONAL(p_i, p_f)
 1  T_a.init(p_i); T_b.init(p_f)
 2  for j ← 1 to K
 3       do p_n ← NEAREST(T_a, α(j))
 4          p_s ← STOPPING-CONFIGURATION(p_n, α(j))
 5          if p_s ≠ p_n
 6            then T_a.add-vertex(p_s)
 7                 T_a.add-edge(p_n, p_s)
 8                 p'_n ← NEAREST(T_b, p_s)
 9                 p'_s ← STOPPING-CONFIGURATION(p'_n, p_s)
10                 if p'_s ≠ p'_n
11                   then T_b.add-vertex(p'_s)
12                        T_b.add-edge(p'_n, p'_s)
13                 if p'_s = p_s
14                   then return Solution
15          if |T_b| > |T_a|
16            then SWAP(T_a, T_b)
17  return Failure
```

In this algorithm, T_a and T_b represent trees with a composite trajectory point p at each node (Eq. 5). The points p at the nodes are considered at rest, so the only relevant information in these points are positions and attitudes. The two trees T_a and T_b start from the initial and final points of the desired trajectory. At each iteration $\alpha(i)$ generates a random point, and NEAREST$(T_a, \alpha(i))$ finds the point in the tree T_a with the minimum distance to this point $\alpha(i)$. The distance is defined as

$$\text{distance}(p_1, p_2) = \sum_{i=1}^{N} \|x_{1,i} - x_{2,i}\| + K_a \angle(q_{1,i}, q_{2,i})$$

where $\angle(q_{1,i}, q_{2,i})$ is the angle of an eigen-axis rotation between attitude $q_{1,i}$ and $q_{2,i}$ for spacecraft i, and K_a is a weight that relates translation distance and rotation angle. In the scenarios presented later, $K_a = 6$.

Continuing with the algorithm, STOPPING-CONFIGURATION$(p_n, \alpha(i))$ finds the last valid configuration in the "direct motion" from p_0 to p. In our implementation, a *direct motion* is a rest-to-rest straight line translation and eigen-axis rotation of each spacecraft

$$\left.\begin{array}{r} \text{trans}(x_{1,i}, x_{2,i}; t) = x_{1,i} + t(x_{2,i} - x_{1,i}) \\ \text{rot}(q_{1,i}, q_{2,i}; t) = q_{1,i} \cdot (q_{1,i}^{-1} \cdot q_{2,i})^t \end{array}\right\} \quad \begin{array}{c} \forall i \\ t \in [0,1] \end{array} \tag{17}$$

where rot is a quaternion interpolation.

If a new node is successfully found by the direct motion, then a branch to this node is added to the tree, and a similar attempt is made to connect the opposite tree to the new node. If the attempt succeeds the algorithm stops and returns a good trajectory, otherwise it continues.

The output of the algorithms consists of a sequence of points from the initial point p_i to the desired target point p_f. At these points the spacecraft are at rest, their states are described by position and attitude values, and there is a direct motion to the next point that is guaranteed to satisfy all the constraints.

C. The smoother

The resulting trajectory from the planner is then smoothed. However, this trajectory must be described first as full state and input pairs sampled at fixed time-steps. In general these samples will not coincide with the points of the trajectory from the planner, and the spacecraft will not necessarily be at rest at these points in time. In this section the state notation $p(t)$ has been replaced by $p(k)$ which stands for $p(k\Delta T)$, where ΔT is the time-step. The complete trajectory is represented by the sequence of points $p(k), k \in 0 \ldots \lceil T/\Delta T \rceil$. For these points

$$p(k+1) = h(p(k)) \tag{18}$$

where $h(p(k))$ is the propagation for time ΔT of the states $p(k)$ for constant inputs $u(k)$ and $M(k)$. To ensure consistency between all elements of $p(k)$ (states and inputs), $\dot{x}(k)$, $u(k)$, $w(k)$ and $M(k)$ must be found that satisfy Eq. 18. This is accomplished using a discrete approximation for the short time interval propagation. The discrete equations for translational dynamics are

$$\begin{bmatrix} x_i(k+1) \\ \dot{x}_i(k+1) \end{bmatrix} = \begin{bmatrix} I & \Delta T I \\ 0 & I \end{bmatrix} \begin{bmatrix} x_i(k) \\ \dot{x}_i(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta T I \end{bmatrix} u_i(k) \tag{19}$$

and for the attitude dynamics

$$\omega_i(k+1) = \omega_i(k) - \Delta T J^{-1}\omega_i(k)\times(J\omega_i(k)) + \Delta T J^{-1} M_i(k) \tag{20}$$

and

$$q(k+1) = \left[I + \frac{\Delta T}{2}\Omega_i(k)\right] q(k) \tag{21}$$

where $\Omega_i(k)$ is just the discrete form of Eq. 9. Thus to obtain the full $p(k)$ from x and q

$$\dot{x}_i(k) = \frac{x_i(k+1) - x_i(k)}{\Delta T} \tag{22}$$

$$u_i(k) = \frac{\dot{x}_i(k+1) - \dot{x}_i(k)}{\Delta T} \tag{23}$$

$$M_i(k) = J\frac{\omega_i(k+1) - \omega_i(k)}{\Delta T} + \omega_i(k) \times (J\omega_i(k)) \tag{24}$$

and $\omega_i(k) = [\tilde{\omega}_{i1}(k), \tilde{\omega}_{i2}(k), \tilde{\omega}_{i3}(k)]^T$, from

$$\tilde{\omega}_i(k) = 2Q_i(k)^{-1}\frac{q_i(k+1) - q_i(k)}{\Delta T} \tag{25}$$

where

$$Q_i(k) = \begin{bmatrix} q_{i,4}(k) & -q_{i,3}(k) & q_{i,2}(k) & q_{i,1}(k) \\ q_{i,3}(k) & q_{i,4}(k) & -q_{i,1}(k) & q_{i,2}(k) \\ -q_{i,2}(k) & q_{i,1}(k) & q_{i,4}(k) & q_{i,3}(k) \\ -q_{i,1}(k) & -q_{i,2}(k) & -q_{i,3}(k) & q_{i,4}(k) \end{bmatrix}$$

The discrete dynamics (Eqs. 19–21) can then be represented as the equalities

$$p(k+1) - h(p(k)) = 0, k \in 0 \ldots (\lceil T/\Delta T \rceil - 1) \tag{26}$$

with pointing and collision avoidance constraints

$$g_n(p(k)) \leq 0, k \in 0 \ldots \lceil T/\Delta T \rceil, n \in 1 \ldots N_c \tag{27}$$

An iteration of the smoothing algorithm consists of finding a perturbation of the trajectory that improves the cost while maintaining the feasibility of the trajectory. The update is achieved by using the first-order Taylor approximation of the constraints and the cost

$$\begin{aligned} & p(k+1) + dp(k+1) - h\left(p(k) + dp(k)\right) \\ & \approx p(k+1) - h(p(k)) + dp(k+1) - \nabla h(p(k))^T dp(k) \\ & = 0, \end{aligned} \tag{28}$$

and

$$g_n(p(k) + dp(k)) \approx g_n(p(k)) + \nabla g_n(p(k))^T dp(k) \leq 0 \tag{29}$$

for $\|dp(k)\| \leq \epsilon \ll 1$. Note that these are *linear constraints* in the variables $dp(k)$ since h, g_n and $p(k)$ are known in advance.

The discrete form of the cost function is

$$J = \Delta T \sum_{k=0}^{\lceil T/\Delta T \rceil} \sum_{i=1}^{N} |u_i(k)| + |M_i(k)| \tag{30}$$

which can be rewritten as

$$J = \Delta T \sum_{k=0}^{\lceil T/\Delta T \rceil} \sum_{i=1}^{N} |u_i(k) + du_i(k)| + |M_i(k) + dM_i(k)| \tag{31}$$

which in a linear minimization is equivalent to

$$J = \Delta T \sum_{k=0}^{\lceil T/\Delta T \rceil} \sum_{i=1}^{N} a_i(k) + b_i(k) \tag{32}$$

subject to

$$|u_i(k) + du_i(k)| \leq a_i(k), \ \forall i, k \tag{33}$$

$$|M_i(k) + dM_i(k)| \leq b_i(k), \ \forall i, k \tag{34}$$

The step to find the perturbation is naturally formulated as a linear program because it is the minimization of a linear function subject to linear equality and inequality constraints (Line 2 in the SMOOTHER-STEP). Here the trajectory is improved as a whole and the process continues in a deterministic fashion. The algorithm is as follows:

SMOOTHER(p)
1 **for** $j = 1$ **to** M
2 **do** SMOOTHER-STEP(p)
3 **return** p

SMOOTHER-STEP(p)
1 **for** $i = 1$ **to** N
2 **do** solve linear program:

$$\begin{aligned} & \min \Delta T \textstyle\sum_{k=0}^{\lceil T/\Delta T \rceil} a_i(k) + b_i(k) \\ & \text{subject to} \\ & \forall k \begin{cases} \begin{bmatrix} I & -\nabla h(p(k))^T \end{bmatrix} \begin{bmatrix} dp(k+1) \\ dp(k) \end{bmatrix} = 0 \\ \nabla g_n(p(k))^T dp(k) \leq -g_n(p(k)) \\ |u(k) + du(k)| \leq a(k) \\ |M(k) + dM(k)| \leq b(k) \\ |dp(k)| \leq \epsilon \end{cases} \end{aligned}$$

▷ end of linear program
3 $p(k) \leftarrow p(k) + dp(k), \forall k$
4 $q(k) \leftarrow \frac{q(k)}{\|q(k)\|}, \forall k$
5 $(\dot{x}(k), \ u(k)) \leftarrow$ inverse of $(x(k), \ x(k+1)), \forall k$, from equations (22) and (23)
6 $(\omega(k), \ M(k)) \leftarrow$ inverse of $(q(k), \ q(k+1)], \forall k$, from equations (26) and (24)
▷ end of for
7 **return** p

Due to the linear approximations, in general the updated solution in step 3 may violate the constraints by a small amount (less than O(ϵ), where $0 < \epsilon \ll 1$). Therefore the solution is repaired in lines 4 to 6 to recover the consistency with the constraints of the problem, particularly the unit norm quaternion constraint and the dynamics from Eqs. 19–21. The inequality constraints are not explicitly repaired. Any violation of these constraints is negligible and does not affect the convergence of the algorithm. To guarantee that the final solution meets the constraints, a small margin is added to the constraints before running the algorithm. For example, the angles and collision radius are increased by a small percentage over the actual values.

In summary, our approach consists of the following steps:

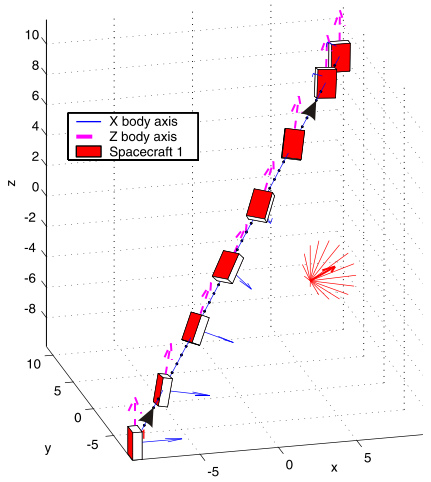

Fig. 2. Example: Simple maneuver. Simple translation and rotation with sun avoidance constraint.

FIND-RECONFIGURATION(p_i, p_f)
1 $s =$ RDT-BALANCED-BIDIRECTIONAL(p_i, p_f)
2 **if** $s \neq Failure$
3 **then** discretize s
4 SMOOTHER(s)
5 **return** s

IV. EXAMPLES

This section presents examples of varying complexity. They demonstrate that for simple problems the algorithm generates the expected results, and for harder problems it generates reasonable trajectories. In the figures that illustrate these examples the trajectories are represented by a solid line, each dot represents a time step, and arrows show the direction of movement. The spacecraft are typically shown along the trajectories after every fifth time step. The vectors shown are the X, Y, and Z body axes. The plot axes correspond to the axes of the local inertially fixed frame. Also, the examples that include a sun avoidance (stay-out) constraint show a "red umbrella", with the "handle" representing the vector pointing *toward* the sun, and a cone of rays representing the angle covered by the constraint.

These experiments were run on a Fujitsu T3000 with an Intel Centrino processor at 1.4 GHz and Windows XP, and the algorithms were programmed in C++ and compiled with Microsoft Visual C++ 7.1. The linear solver used was the GLPK library. The computation times for the *planner* ranged from below 1 second for single spacecraft problems like the simple maneuver, to 30-40 minutes for highly constrained problems with 4 spacecraft. The number of iterations of the main loop in RDT-BALANCED-BIDIRECTIONAL ranged from 1 to 10 for simple examples, to between 200 and 500 for the difficult ones. The cut-off in our experiments was 500 iterations, after which the algorithm returned a failure. The computation times for the *smoother* were from below 1 second to 3 seconds, using the interior point solver.

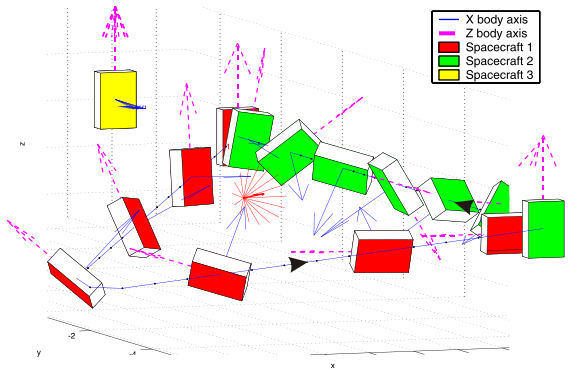

Fig. 3. Example: coupled maneuver. Shown in detail. Spacecraft 1 and 2 switch places while pointing at each other. Spacecraft 3 points at 1 and 2. They also avoid pointing at the sun.

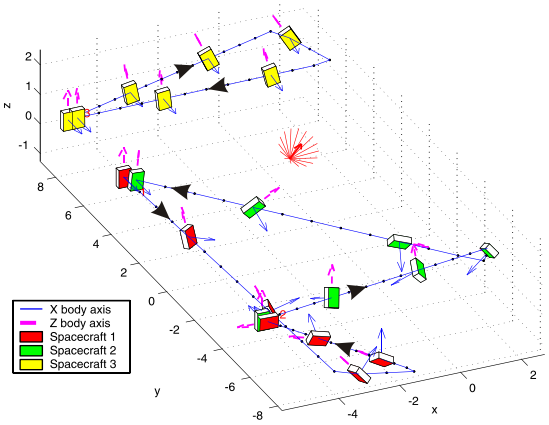

Fig. 4. Same as Figure 3, solution before the smoothing

A. Example: Simple maneuver

Figure 2 shows the final trajectory for a simple problem: move a spacecraft from $[-9,-9,-9]^T$ to $[9,9,9]^T$ and rotate it 90° about the inertial Z-axis, while avoiding pointing at the sun. The unit vector pointing at the sun is represented in the figure by the vector in the direction of $[1,1,0]^T/\sqrt{2}$ surrounded by a 40^o angle cone. The sensitive instrument points in the direction of the body X axis (solid blue in the figure) and it must stay out of the cone. The plot shows that the solver designs a smooth trajectory that skirts this constraint. As expected, the translation path is minimal: a simple straight line.

B. Example: Coupled Maneuver

Figures 3 and 4 show a slightly more complex example with three spacecraft that demonstrate the interaction between the states of the spacecraft and the coupling constraints. Spacecraft 1 and 2 are initially at positions $[-5,5,0]^T$ and $[-5,-5,0]^T$. Spacecraft 1 must turn 180^o around the Z axis and 90^o around the X axis. Spacecraft 2 only has to turn 180^o around the Z axis. Both must also point their body X axis (solid blue) at the other spacecraft to within 30^o. Spacecraft 3 must end at the same starting position of $[-5,9,0]^T$, and point its body X axis at both spacecraft 1 and 2 to within 15^o angle. The vehicles must

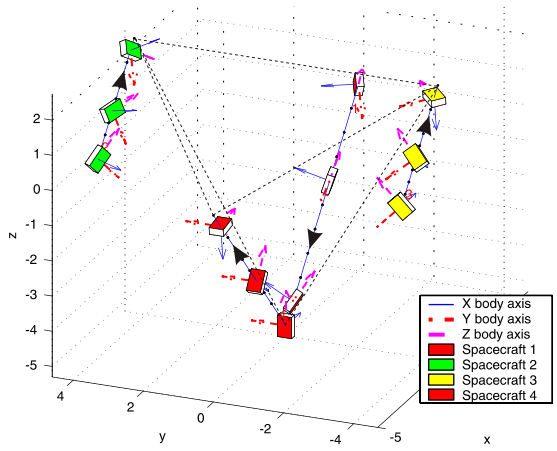

Fig. 5. Rotate tetrahedral configuration 90^o degrees around Y axis. Pointing constraints remain as before.

remain 3.5 units apart to avoid colliding.

The final trajectory in Figure 3 shows that spacecraft 2 translates straight from the start to finish while spacecraft 1 moves away just enough to satisfy the collision avoidance constraint. Notice that for spacecraft 1 to point the body X axis at spacecraft 2 while avoiding pointing it at the Sun, it has to move off the X-Y plane. This shows the coupling between relative and absolute pointing constraints, and collision avoidance. Also, when spacecraft 1 moves off the initial alignment, spacecraft 3 rotates slightly to keep both spacecraft 1 and 2 inside its specified cone.

The trajectory of figure 3 is based on the trajectory shown in Figure 4, which is shown here for comparison. Figure 4 shows the trajectory produced by the planner, before the smoothing procedure. As expected, the trajectory is feasible, but clearly suboptimal. All the spacecraft move and rotate away from the target positions, then return to them, which is particularly evident for spacecraft 3. The difference between the figures shows the large changes possible by the smoother given a feasible initial solution.

C. Example: Four vehicles

Figure 5 shows the solution to a complex example with four spacecraft. The constraints in this example are as follows: (a) Spacecraft 1, 2 and 3 must point their body Y axis (dashed red) toward spacecraft 4; and (b) they must also point their body X axis (solid blue) to each other in a ring (spacecraft 1 must point to spacecraft 2, 2 to 3, and 3 to 1). These 6 constraints place tight restrictions on the possible movements of the spacecraft which must be closely coordinated. The attitude of spacecraft 4 is not constrained.

The maneuver starts with a tetrahedral formation with spacecraft 1, 2 and 3 in the X-Y plane pointing to spacecraft 4 below. The formation then rotates 90° about the inertial Y axis. The final solution consists of simple straight line translations for all 4 spacecraft, with minimal rotations toward the desired final attitudes, consistent with an optimal maneuver. The constraints are always met.

V. Conclusion

Designing spacecraft reconfiguration maneuvers is challenging because it includes nonlinear attitude dynamics, difficult non-convex constraints, and high dimensionality ($6N$ DOF) due to coupling of the multiple spacecraft states in the constraints. This paper presented a method that can solve for reconfigurations for up to 4 spacecraft. The essential feature of this method is the separation into a simplified path planning problem without differential constraints to obtain a feasible solution, which is then improved by a smoothing operation. The first step is solved using Rapidly-exploring Random Trees [1]. The smoother consists of an optimization by iteratively solving a linear program using a linearization of the cost function, dynamics, and constraints about the initial feasible solution. The examples demonstrated the validity of the approach and also showed that the algorithm can solve problems with four spacecraft with several complex pointing restrictions.

Acknowledgements

Research funded under NASA Grant NAG5-10440.

References

[1] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *International Journal of Robotics Research,* Vol. 20, No. 5, p. 378–400, May 2001.

[2] J. Leitner, F. Bauer, D. Folta, M. Moreau, R. Carpenter, and J. How, "Distributed Spacecraft Systems Develop New GPS Capabilities," in *GPS World: Formation Flight in Space* Feb. 2002.

[3] J. H. Reif, "Complexity of the Mover's Problem and Generalizations," *20th Annual IEEE Symposium on Foundations of Computer Science,* San Juan, Puerto Rico, October 1979, p. 421–427.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robotics Research*, (5), no. 1, p. 90–98, 1986.

[5] C. R. McInnes, "Large Angle Slew Maneuvers with Autonomous Sun Vector Avoidance," AIAA *JGCD*, Vol. 17, No. 4, p. 875–877.

[6] H. Hablani, "Attitude Commands Avoiding Bright Objects and Maintaining Communication with Ground Station," AIAA *Journal of Guidance, Control, and Dynamics,* Vol. 22, No. 6, p. 759–767.

[7] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft Trajectory Planning With Collision and Plume Avoidance Using Mixed-Integer Linear Programming," AIAA *Journal of Guidance, Control, and Dynamics,* Vol. 25, No. 4, p. 755–765, Aug. 2002.

[8] E. Frazzoli, "Quasi-Random Algorithms for Real-Time Spacecraft Motion Planning and Coordination" *International Astronautical Congress*, Houston, TX, 2002.

[9] E. Frazzoli, M. Dahleh, E. Feron, R. Kornfeld, "A Randomized Attitude Slew Planning Algorithm For Autonomous Spacecraft," *AIAA Guidance, Navigation and Control Conf.*, AIAA 2001–4155.

[10] D. P. Scharf, and S. R. Ploen, F. Y. Hadaegh, J. A. Keim, and L. H. Phan, "Guaranteed Initialization of Distributed Spacecraft Formations," *AIAA Guidance, Navigation and Control Conference*, AIAA 2003–5590, August 2003.

[11] J. Phillips, L. E. Kavraki, and N. Bedrosian , "Probabilistic Optimization Applied to Spacecraft Rendezvous and Docking," In *13th American Astronomical Society/AIAA - Space Flight Mechanics Meeting,*, Puerto Rico, February 2003.

[12] L. E. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," IEEE *Trans on Robotics and Auto.,* (12)4, p. 566–580, 1996.

[13] R. Fletcher, "Practical Methods of Optimization," John Wiley and Sons, 1987.

[14] C. Sultan, S. Seereeram, R. Mehra, and F. Hadaegh, "Energy Optimal Reconfiguration for Large-Scale Formation Flying," In *Proceedings of the American Control Conference,* p. 2986–2991, July 2004.

[15] S. M. LaValle, "Planning Algorithms," [Online] http://msl.cs.uiuc.edu /planning/, 2004.