Fast Motion Planning for Agile Space Systems with Multiple Obstacles

Francesca Baldini*, Saptarshi Bandyopadhyay†, Rebecca Foust‡, Soon-Jo Chung§, Amir Rahmani¶, Jean-Pierre de la Croix‖, Alexandra Bacula,** Christian M. Chilan††, Fred Y. Hadaegh‡‡

In this paper, we develop a novel algorithm for spacecraft trajectory planning in an environment cluttered with many geometrically-fixed obstacles. The Spherical Expansion and Sequential Convex Programming (SE–SCP) algorithm first uses a spherical-expansion-based sampling algorithm to explore the workspace. Once a path is found from the start position to the goal position, the algorithm generates a locally optimal trajectory within the homotopy class using sequential convex programming. If the number of samples tends to infinity, then the SE–SCP trajectory converges to the globally optimal trajectory in the workspace. The SE–SCP algorithm is computationally efficient, therefore it can be used for real-time applications on resource-constrained systems. We also present results of numerical simulations and comparisons with existing algorithms.

I. Introduction

Spacecraft trajectory planning for formation flying has been a major area of research over the past decade.[1] There have been significant advances in the development of passive relative orbits for such missions[2] and formation flying algorithms for cooperative spacecraft.[3] In applications like navigation through a debris field or asteroid belt, the trajectory planning algorithm has to find fuel-optimal paths while avoiding collision with obstacles. Therefore, a key challenge is the development of computationally-efficient real-time algorithms that can handle uncooperative obstacles while guaranteeing some form of optimality. The main difficulty is that the existing spacecraft trajectory planning algorithms are unable to generate a path through uncooperative obstacles. On the other hand, sampling based algorithms have been successfully implemented for many robotic motion planning applications through stationary obstacle fields.[4] In this paper, we introduce a novel spacecraft trajectory planning algorithm, called the Spherical Expansion and Sequential Convex Programming (SE–SCP) algorithm, which is computationally efficient for real-time implementation on resource constrained systems and guarantees local optimality within the homotopy class (i.e., the class of local trajectories that can be reached from the original trajectory using continuous deformations).

I.A. Literature Survey

A number of probabilistic sampling based algorithms for robotic motion planning have been proposed in the recent years, namely probabilistic road maps (PRM[5] and PRM*[4]), rapidly exploring random trees

*Graduate Visiting Student, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign (UIUC), Urbana, Illinois, 61801, USA; baldini2@illinois.edu

†Postdoctoral Researcher, Jet Propulsion Laboratory (JPL), California Institute of Technology (Caltech), Pasadena, California, 91109, USA; Saptarshi.Bandyopadhyay@jpl.nasa.gov

‡Graduate Student, Department of Aerospace Engineering, UIUC, Urbana, Illinois, 61801, USA; foust3@illinois.edu

§Associate Professor, Department of Aerospace, California Institute of Technology, Pasadena, California, 91125, USA; sjchung@caltech.edu Senior Member, AIAA.

¶Research Scientist, JPL, Caltech, Pasadena, California, 91109, USA; Amir.Rahmani@jpl.nasa.gov

‖Research Scientist, JPL, Caltech, Pasadena, California, 91109, USA; jean-Pierre.de.la.Croix@jpl.nasa.gov

**Undergraduate Student, Department of Aerospace Engineering, UIUC, Urbana, Illinois, 61801, USA; anbacula@gmail.com

††Postdoctoral Research Associate, Department of Aerospace Engineering, UIUC, Urbana, Illinois, 61801, USA; chilan@illinois.edu

‡‡Senior Research Scientist and Technical Fellow, JPL, Caltech, Pasadena, California, 91109, USA; fred.y.hadaegh@jpl.nasa.gov. Fellow, AIAA.

(RRT,[6] RRT*[4]), and fast marching trees (FMT*[7]). Among these algorithms, PRM*, RRT*, and FMT* are asymptotically optimal; i.e., as the number of samples tends to infinity, the path is optimal for distance-based cost functions.

There are a number of issues with directly applying these algorithms for the spacecraft trajectory planning on computational-resource-constrained systems that require real-time algorithms. Firstly, these algorithms guarantee optimality only when the number of samples tend to infinity. Moreover, there are no optimality guarantees for a finite number of samples. Hence, this approach is not suitable for practical implementation on board spacecraft. In this paper, we develop the SE–SCP algorithm that guarantees local optimality within a homotopy class using a finite number of samples.

Furthermore, distance-based cost functions are usually used in these existing algorithms. These algorithms cannot be easily modified to consider other dynamics-based cost function. In the SE–SCP algorithm, we use the actual 6 degrees of freedom (DOF) dynamics of the spacecraft to generate the trajectory and minimize the actual cost (like fuel, control effort, or time of flight) incurred by the spacecraft to track the trajectory. Therefore, the main focus of this paper is to develop the SE–SCP algorithm that overcomes the above shortcomings.

The SE–SCP algorithm is discussed in detail in Section III. The key steps in the algorithm are:

- The algorithm first explores the workspace space using a probabilistic or deterministic sampling technique.
- Once a geometric path is found from the start position to the goal position, a trajectory that satisfies the dynamics constraints is generated along this path using sequential convex optimization. This trajectory is locally optimal within the homotopy class.
- On further sampling, if a better (i.e., less costly) geometric path is found, then the algorithm generates the new optimal trajectory that satisfies the dynamics constraints along the new path. The algorithm stores the best trajectory found until the current time instant.
- The algorithm can exit any time after the first path is found. The algorithm outputs the current best trajectory.
- If number of samples tends to infinity, then the algorithm gives the globally optimal trajectory.

This paper is organized as follows. We present the spacecraft dynamics in Appendix and the problem statement in Section II. The SE–SCP algorithm is described in detail in Section III. Numerical simulation results and comparison with other existing algorithms are shown in Section IV. This paper is concluded in Section V.

II. Problem Statement

In this section, we present the problem statement. Let $\mathcal{X} \in \mathbb{R}^n$ represent the workspace. Let $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ represent the obstacles in the workspace. Let δ denote the clearance that the spacecraft must maintain from any obstacle. Let $\mathcal{X}_{\text{unsafe}} \subset \mathcal{X}$ denote the unsafe region around an obstacle that the spacecraft cannot enter. Therefore, the region where the spacecraft can maneuver freely is given by $\mathcal{X}_{\text{free}} = \mathcal{X}/(\mathcal{X}_{\text{obs}} \cup \mathcal{X}_{\text{unsafe}})$.

The spacecraft's dynamics is given by the following nonlinear equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

where $\boldsymbol{f}$ is a smooth nonlinear function, $\boldsymbol{x}$ is the state vector of the spacecraft, $\boldsymbol{u} \in \mathcal{U}$ is the control input, and $\mathcal{U}$ is the feasible range of control inputs. The spacecraft dynamics in the relative frame that incorporate J_2 perturbations and atmospheric drag are given in the Appendix.

Let $\boldsymbol{\sigma}(t) \in \mathcal{X}_{\text{free}}$ represent a feasible trajectory for the spacecraft and $c(\boldsymbol{\sigma}(t))$ represent the cost incurred by the spacecraft to track this trajectory. Let $X_{\text{init}} \in \mathcal{X}_{\text{free}}$ and $X_{\text{goal}} \in \mathcal{X}_{\text{free}}$ represent the starting position and the goal position of the spacecraft at times t_0 and t_f respectively.

The optimal motion planning problem is to design a trajectory $\boldsymbol{\sigma}^*(t) \in \mathcal{X}_{\text{free}}$ from the start position to the goal position so that the cost incurred by the spacecraft to track this trajectory $c(\boldsymbol{\sigma}^*(t))$ is minimized.

This is written mathematically as follows:

$$\begin{aligned} &\underset{\boldsymbol{\sigma}(t),\,\boldsymbol{u}(t)}{\text{minimize}} \; c(\boldsymbol{\sigma}(t))\,, \\ \text{subject to} \quad &\boldsymbol{\sigma}(t_0) = X_{\text{init}}\,, \\ &\boldsymbol{\sigma}(t_f) = X_{\text{goal}}\,, \\ &\boldsymbol{\sigma}(t) \in \mathcal{X}_{\text{free}} \quad \forall t \in [t_0, t_f]\,, \\ &\dot{\boldsymbol{\sigma}}(t) = \boldsymbol{f}(\boldsymbol{\sigma}(t),\, \boldsymbol{u}(t)) \quad \forall t \in [t_0, t_f]\,, \\ &\boldsymbol{u}(t) \in \mathcal{U} \quad \forall t \in [t_0, t_f]\,. \end{aligned} \tag{2}$$

The globally optimal solution of Eq. (2) is the best trajectory for the spacecraft to take. But exploring the entire workspace and all the homotopy classes within it can take a large amount of time. Instead, it is better to find the best locally optimal solution within the homotopy classes that can be obtained and updated relatively fast till the present time step. Let us define $\mathcal{X}_{\text{homotopy}} \in \mathcal{X}_{\text{free}}$ as the set of all homotopy classes that have been discovered until the present time step. Therefore, the updated optimization problem is stated as follows:

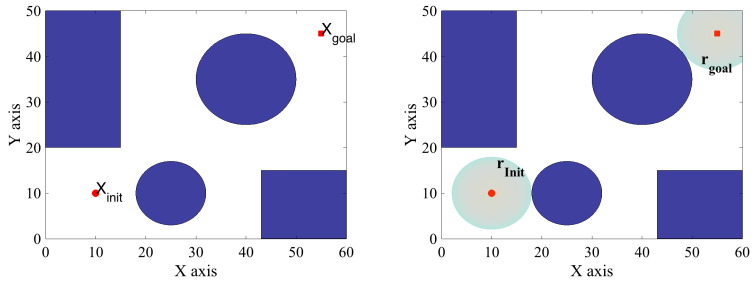
$$\begin{aligned} &\underset{\boldsymbol{\sigma}(t),\,\boldsymbol{u}(t)}{\text{minimize}} \; c(\boldsymbol{\sigma}(t))\,, \\ \text{subject to} \quad &\boldsymbol{\sigma}(t_0) = X_{\text{init}}\,, \\ &\boldsymbol{\sigma}(t_f) = X_{\text{goal}}\,, \\ &\boldsymbol{\sigma}(t) \in \mathcal{X}_{\text{homotopy}} \quad \forall t \in [t_0, t_f]\,, \\ &\dot{\boldsymbol{\sigma}}(t) = \boldsymbol{f}(\boldsymbol{\sigma}(t),\, \boldsymbol{u}(t)) \quad \forall t \in [t_0, t_f]\,, \\ &\boldsymbol{u}(t) \in \mathcal{U} \quad \forall t \in [t_0, t_f]\,. \end{aligned} \tag{3}$$

In this paper, we present the solution to both Eqs. (2) and (3) using our algorithm.

III. Spherical Expansion and Sequential Convex Programming Algorithm

In this section, we present the SE–SCP algorithm. The pseudocode for the SE–SCP algorithm is presented in Algorithm 1.

Let $\mathcal{V}$ be the set of vertices. In addition to storing the position of the vertex, $\mathcal{V}$ also stores the minimum distance of the vertex from the obstacles. Let $\mathcal{E}$ be the set of edges, where each element stores the starting vertex of the edge, the end vertex of the edge, the actual trajectory to travel from the start vertex to the end vertex, and the cost of the trajectory. Therefore, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed graph.



(1a) X_{init} and X_{goal} are shown (1b) r_{init} and r_{goal} are shown

Figure 1: Initialization step. The obstacles are marked in blue.

During initialization, the initial position X_{init} and the goal position X_{goal} are added to $\mathcal{V}$ along with their minimum distance from the obstacles. For X_{init}, the minimum distance from the obstacle r_{init} is obtained using the function `FindMinDfromObs` which returns the radius of the largest sphere centered on that point that will not intersect with any obstacle (see lines 1, 2, 9). During initialization, the cost for the path $P_{\text{init,goal,old}}$ and the trajectory $\boldsymbol{\sigma}_{\text{init,goal,old}}$ are set to infinity. The initialization step is shown in lines 1–4 and in Fig. 1.

Algorithm 1 SE–SCP Algorithm

1: $r_{\text{init}} \leftarrow \texttt{FindMinDfromObs}(X_{\text{init}}, \mathcal{X}_{\text{obs}})$
2: $r_{\text{goal}} \leftarrow \texttt{FindMinDfromObs}(X_{\text{goal}}, \mathcal{X}_{\text{obs}})$
3: $\mathcal{V} \leftarrow \{X_{\text{init}}[r_{\text{init}}], X_{\text{goal}}[r_{\text{goal}}]\}$, $\mathcal{E} \leftarrow \emptyset$
4: $c(P_{\text{init,goal,old}}) \leftarrow \infty$, $c(\boldsymbol{\sigma}_{\text{init,goal,old}}) \leftarrow \infty$
5: **for** $i = 1, \ldots, n$ **do**
6: $X_{\text{rand}} \leftarrow \texttt{GenerateSamples}$
7: $X_{\text{nearest}} \leftarrow \texttt{Nearest}(\mathcal{G} = (\mathcal{V}, \mathcal{E}), X_{\text{rand}})$
8: $X_{\text{new}} \leftarrow \texttt{Steer}(X_{\text{nearest}}, X_{\text{rand}}, r_{\text{nearest}})$
9: $r_{\text{new}} \leftarrow \texttt{FindMinDfromObs}(X_{\text{new}}, \mathcal{X}_{\text{obs}})$
10: $\boldsymbol{X}_{\text{near}} \leftarrow \texttt{NearIntesectedSpheres}(\mathcal{G} = (\mathcal{V}, \mathcal{E}), X_{\text{new}}, r_{\text{new}})$
11: $\mathcal{V} \leftarrow \mathcal{V} \cup X_{\text{new}}[r_{\text{new}}]$
12: **for all** $X_{\text{n}} \in \boldsymbol{X}_{\text{near}}$ **do**
13: $\boldsymbol{\sigma}_{\text{n,new}}, c(\boldsymbol{\sigma}_{\text{n,new}}), \boldsymbol{\sigma}_{\text{new,n}}, c(\boldsymbol{\sigma}_{\text{new,n}}) \leftarrow \texttt{SimpleTrajectory}(X_{\text{n}}, X_{\text{new}}, r_{\text{n}}, r_{\text{new}})$
14: $E \leftarrow E \cup \{(X_{\text{n}}, X_{\text{new}})[\boldsymbol{\sigma}_{\text{n,new}}, c(\boldsymbol{\sigma}_{\text{n,new}})], (X_{\text{new}}, X_{\text{n}})[\boldsymbol{\sigma}_{\text{n,new}}, c(\boldsymbol{\sigma}_{\text{n,new}})]\}$
15: **end for**
16: $P_{\text{init,goal,new}}, c(P_{\text{init,goal,new}}) \leftarrow \texttt{MinPath}(\mathcal{G} = (\mathcal{V}, \mathcal{E}), X_{\text{init}}, X_{\text{goal}})$
17: **if** $c(P_{\text{init,goal,new}}) < c(P_{\text{init,goal,old}})$ **then**
18: $\boldsymbol{\sigma}_{\text{init,goal,new}}, c(\boldsymbol{\sigma}_{\text{init,goal,new}}) \leftarrow \texttt{SCPOptimalTrajectory}(X_{\text{init}}, X_{\text{goal}}, P_{\text{init,goal,new}})$
19: **if** $c(\boldsymbol{\sigma}_{\text{init,goal,new}}) < c(\boldsymbol{\sigma}_{\text{init,goal,old}})$ **then**
20: $\boldsymbol{\sigma}_{\text{init,goal,old}} \leftarrow \boldsymbol{\sigma}_{\text{init,goal,new}}$, $c(\boldsymbol{\sigma}_{\text{init,goal,old}}) \leftarrow c(\boldsymbol{\sigma}_{\text{init,goal,new}})$
21: $P_{\text{init,goal,old}} \leftarrow P_{\text{init,goal,new}}$, $c(P_{\text{init,goal,old}}) \leftarrow c(P_{\text{init,goal,new}})$
22: **end if**
23: **end if**
24: **end for**

After initialization, the SE–SCP algorithm consists of two steps, namely the *Spherical Expansion step* (lines 6–15) and the *Sequential Convex Optimization step* (lines 16–23), that are executed iteratively run until the algorithm is stopped. We now discuss these steps in detail.

III.A. Spherical Expansion Step

During this step, the workspace is explored using a sampling technique as shown in lines 6–15. The objective of this step is to generate a graph which connects the initial position X_{init} to the goal position X_{goal}.

III.A.1. Generate Samples

The SE–SCP algorithm can use both random or quasi-random (deterministic) sampling. Since the irregularity of the random samples is not a main driver in the motion planning applications, quasi-random sampling is preferred because of its advantages like faster convergence and ease of generation.[8,9] It should be noted that the choice of sampling affects the performance and the analysis of the SE–SCP algorithm.

For a given sampling choice, the function `GenerateSamples` in line 6 returns a random variable $X_{\text{rand}} \in \mathcal{X}$. If random sampling is chosen, then X_{rand} is drawn from a uniform distribution in $\mathcal{X}$ such that the sequence of random points are independent and identically distributed (i.i.d.). Instead, if quasi-random sampling is chosen, then X_{rand} is generated using the Halton sequence. Halton sequence deterministically produce samples with low discrepancy so that a good coverage is obtained over the workspace with a limited number of samples.

III.A.2. Generate New Point using Spherical Expansion

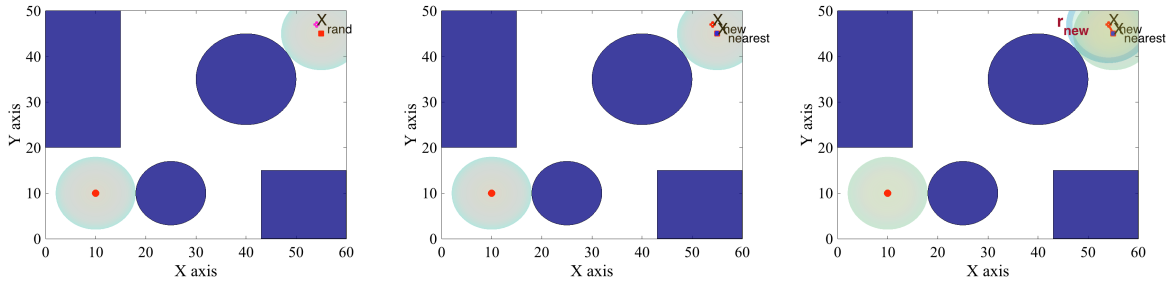
Given the sample X_{rand}, the function `Nearest` in line 7 returns the vertex X_{nearest} in $\mathcal{V}$ that is nearest to X_{rand}:

$$\texttt{Nearest}(\mathcal{G} = (\mathcal{V}, \mathcal{E}), X_{\text{rand}}) := \underset{X \in \mathcal{V}}{\operatorname{argmin}} \|X - X_{\text{rand}}\|_2 \tag{4}$$

Note that the minimum distance from the obstacles r_{nearest} for X_{nearest} is already stored in $\mathcal{V}$.

The function `Steer` in line 8 generates the point X_{new} using either of the two following steps:

- If X_{rand} is serendipitously inside X_{nearest}'s sphere (i.e., $\|X_{\text{nearest}} - X_{\text{rand}}\|_2 \leq r_{\text{nearest}}$) as shown in Fig. 2, then the function `Steer` returns the point $X_{\text{new}} = X_{\text{rand}}$.

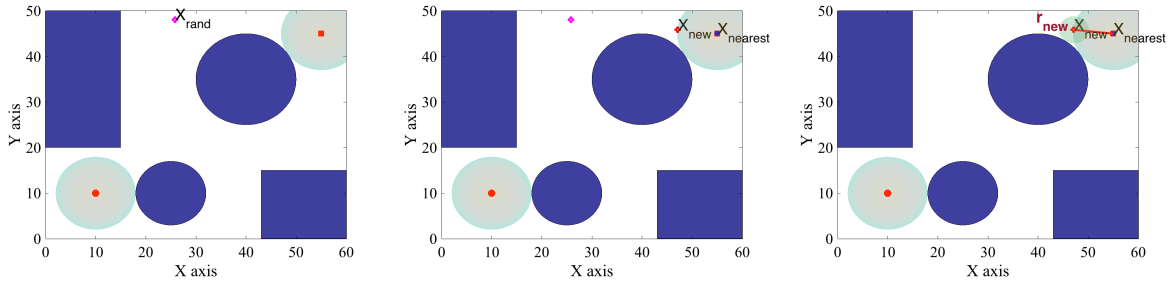


(2a) X_{rand} is generated (2b) X_{new} is generated (2c) Edges and r_{new} are generated

Figure 2: `Steer` function, where X_{rand} is inside X_{nearest}'s sphere

- Otherwise, X_{rand} is outside X_{nearest}'s sphere as shown in Fig. 3. Then the function `Steer` returns the point X_{new} that is on the surface of X_{nearest}'s sphere and closest to X_{rand}:

$$\texttt{Steer}(X_{\text{nearest}}, X_{\text{rand}}, r_{\text{nearest}}) := \underset{\|X - X_{\text{nearest}}\|_2 = r_{\text{nearest}}}{\operatorname{argmin}} \|X - X_{\text{rand}}\|_2 \tag{5}$$



(3a) X_{rand} is generated (3b) X_{new} is generated (3c) Edges and r_{new} are generated

Figure 3: `Steer` function, where X_{rand} is outside X_{nearest}'s sphere

The minimum distance from obstacles r_{new} for the point X_{new} is computed using the function `FindMinDfromObs` in line 9. In contrast with the classical steering function in RRT*,[4] where the radius of the sphere is fixed and represents the step-size of the RRT* algorithm, the radius of the sphere in the SE–SCP algorithm is variable and the average step-size adapts with the density of obstacles. Therefore, the SE–SCP algorithm generally finds a feasible path faster than other sampling-based algorithms.

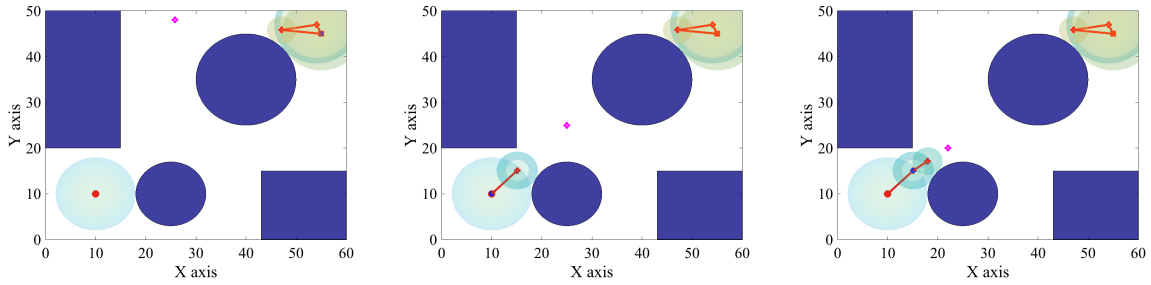
The function `NearIntesectedSpheres`$(\mathcal{G} = (\mathcal{V}, \mathcal{E}), X_{\text{new}}, r_{\text{new}})$ in line 10 generates a list $\boldsymbol{X}_{\text{near}}$ of all the vertices whose spheres intersect with the sphere centered at the point X_{new} with radius r_{new}, i.e.,:

$$\boldsymbol{X}_{\text{near}} := \{X_{\text{n}} \in \mathcal{V} \,:\, \|X_{\text{n}} - X_{\text{new}}\|_2 \leq r_{\text{n}} + r_{\text{new}}\} \tag{6}$$

Therefore, a feasible collision-free path exists between each vertex in $\boldsymbol{X}_{\text{near}}$ and X_{new}. Clearly, it follows from our construction of X_{new} that $X_{\text{nearest}} \in \boldsymbol{X}_{\text{near}}$. The point X_{new} is added to the set of vertices $\mathcal{V}$ along with the minimum distance to obstacle r_{new}, as shown in line 11.

III.A.3. Generate Graph's Edges

We now generate the edges of the graph that connect the new vertex X_{new} with the existing vertices in $\boldsymbol{X}_{\text{near}}$. For each vextex $X_{\text{n}} \in \boldsymbol{X}_{\text{near}}$, the function `SimpleTrajectory`$(X_{\text{n}}, X_{\text{new}}, r_{\text{n}}, r_{\text{new}})$ in line 13 is used to generate the two trajectories between X_{n} and X_{new}.



(4a) Iteration 2 (4b) Iteration 3 (4c) Iteration 4

(4d) Iteration 5 (4e) Iteration 6 (4f) Iteration 7

Figure 4: Multiple iterations of the spherical expansion step

The first trajectory from X_{n} to X_{new} is given by $\boldsymbol{\sigma}_{\text{n,new}}$ and its cost is given by $c(\boldsymbol{\sigma}_{\text{n,new}})$. Similarly, the second trajectory from X_{new} to X_{n} is given by $\boldsymbol{\sigma}_{\text{new,n}}$ and costs $c(\boldsymbol{\sigma}_{\text{new,n}})$. These edges are added to the set of edges $\mathcal{E}$, where the corresponding trajectory and cost of trajectory is also stored.

Although sequential convex programming or other optimization based techniques can be used to generate these trajectories, in this paper we connect the vertices using straight lines and set the Eucledian distance as the cost of the trajectory. This simplification gives good results with significant savings in computation load.

In Fig. 4, multiple iterations of the spherical expansion step are shown. At each step, a new vertex is added to the set of vertices and multiple edges are added to the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Note that the graph grows from both the start and goal positions.

III.B. Sequential Convex Optimization Step

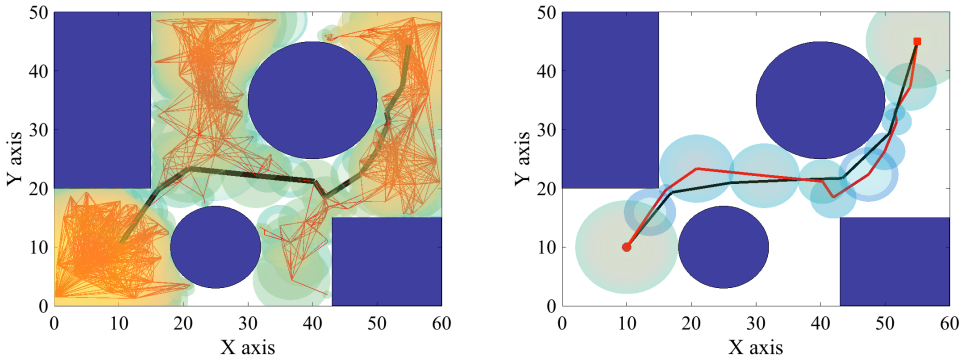
During this step, the shortest paths in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from the initial position X_{init} to the goal position X_{goal} are found, and then the locally optimal trajectory is found using sequential convex programming (SCP).

III.B.1. Generate Shortest Path

The function `MinPath`$(\mathcal{G} = (\mathcal{V}, \mathcal{E}), X_{\text{init}}, X_{\text{goal}})$ in line 16 finds the minimum cost path in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from X_{init} to X_{goal}. Graph search algorithms like Dijkstra's algorithm can be used for this step. The function `MinPath` outputs the optimal path $P_{\text{init,goal,new}}$ and the cost for traversing this path $c(P_{\text{init,goal,new}})$. If no path exists, then $c(P_{\text{init,goal,new}})$ is set to infinity. For example, the shortest path from the start to goal position is shown in Fig. 5.

III.B.2. Generate Optimal Trajectory

If the cost of the shortest path $c(P_{\text{init,goal,new}})$ generated during the present time instant is less than that stored previously in $c(P_{\text{init,goal,old}})$, then the optimal trajectory is found using the function `SCPOptimalTrajectory`$(X_{\text{init}}, X_{\text{goal}}, P_{\text{init,goal,new}})$ in line 18.



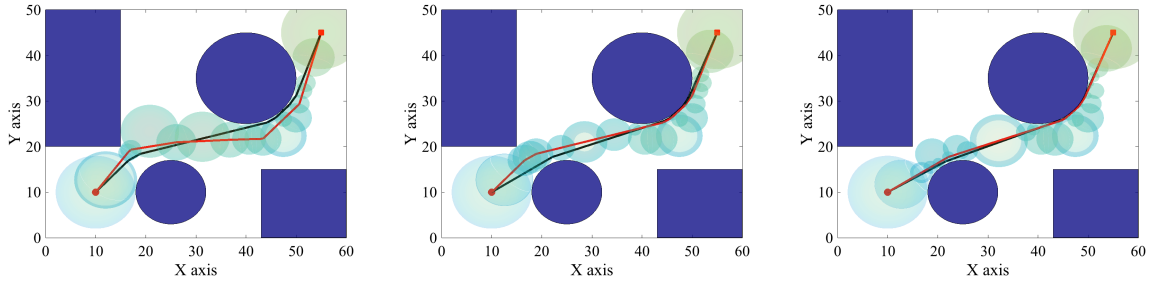
(5a) Shortest path on the graph (5b) Optimal trajectory through this Path

Figure 5: Generating Shortest Path and Trajectory Optimization over this Path

The function `SCPOptimalTrajectory` uses SCP to compute the optimal trajectory from X_{init} to X_{goal} that passes through the spheres corresponding to the points $\{X_0, X_1, \ldots, X_{\text{T}-1}\} \in P_{\text{init,goal,new}}$ as follows:

$$
\begin{aligned}
\underset{\boldsymbol{\sigma(k)},\, \boldsymbol{u(k)}}{\text{minimize}} \quad & \sum_{k=0}^{T-1} \|u(k)\|_2 \,, && && (7)\\
\text{subject to} \quad & \boldsymbol{\sigma}(0) = X_{\text{init}} \,, && \\
& \boldsymbol{\sigma}(T) = X_{\text{goal}} \,, && \\
& \boldsymbol{\sigma}(k+1) = A(k)\boldsymbol{\sigma}(k) + B(k)u(k) \,, && \forall k \in 0, \ldots, T-1 \,, \\
& \|\boldsymbol{\sigma}(k) - X_{\text{k}}\|_2 \leq r_{\text{k}} \,, && \forall k \in 0, \ldots, T \,, \\
& \|u(k)\|_2 \leq U_{\max} \,, && \forall k \in 0, \ldots, T \,.
\end{aligned}
$$

where the time $[t_0, t_f]$ is decomposed into multiple discrete steps $\{0, \ldots, T\}$ and the continuous time optimization problem in (3) is represented as a discrete optimization problem. The SCP optimization problem (7) generates the optimal trajectory that passes through the spheres in the shortest path as shown in Fig. 5.



(6a) Iteration 1 (6b) Iteration 2 (6c) Iteration 3

Figure 6: Iterative Trajectory Optimization using the new Optimal Trajectory

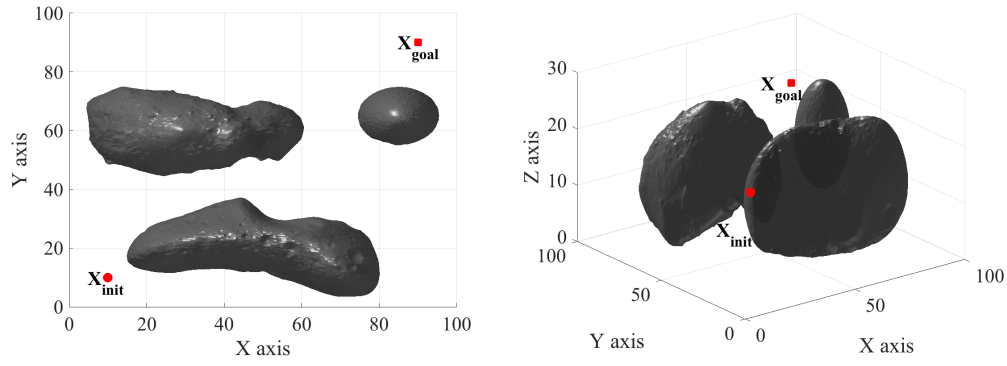
The optimal trajectory generated by the SCP optimization problem (7) can be further optimized by re-running (7) using the new optimal trajectory. Also, note that this sequential run can be used update the linear dynamics to better approximate the original nonlinear dynamics (see Ref. 10). This iterative optimization of the trajectory is shown in Fig. 6, where the red lines show the previous optimal trajectory and the black lines show the solution of the SCP optimization problem (7). Thus the final optimal trajectory converges to the locally optimal trajectory within the homotopy class.

IV. Numerical Simulations

In this section, we first show that the SE–SCP algorithm can be used by a spacecraft to navigate a debris field. We then present comparisons with existing algorithms like RRT* and PRM*.

IV.A. Spacecraft in Debris Field

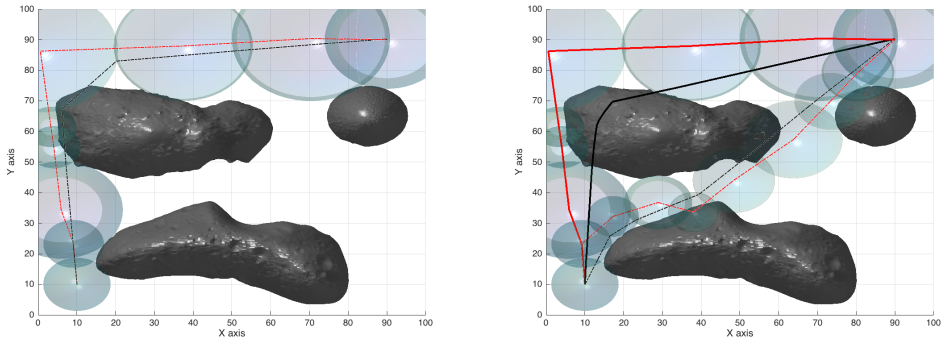
In this subsection, we show that the spacecraft can use the SE–SCP algorithm to move through a cluttered environment like a debris field. The debris environment, the start position, and the goal position are shown in Fig. 7. The spacecraft dynamics are discussed in Appendix. The objective is to find a trajectory from X_{init} to X_{goal} so that the fuel consumed is minimized.



(7a) (7b)

Figure 7: Asteroid environment

Fig. 8 shows two trajectories in two different homotopy classes that are generated by the spherical expansion step. Each of these trajectories are further refined using the SCP step, as shown in Fig. 9. The solution of the SE–SCP algorithm after a finite number of samples is the best locally optimal trajectory shown in Fig. 9. thus the SE–SCP algorithm is suitable for spacecraft applications.



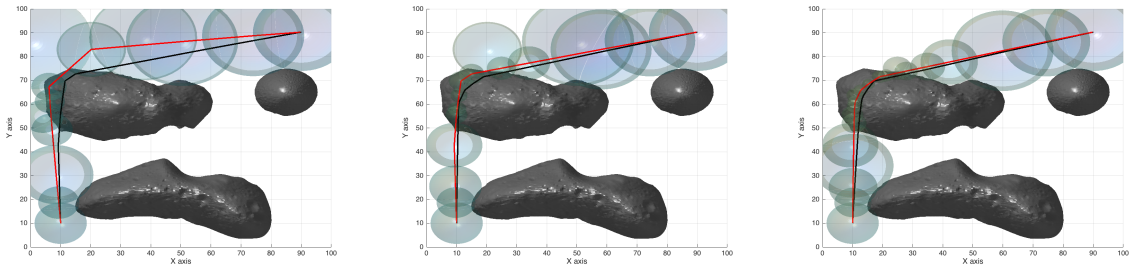
(8a) (8b)

Figure 8: Paths in different homotopy classes found by the spherical expansion step

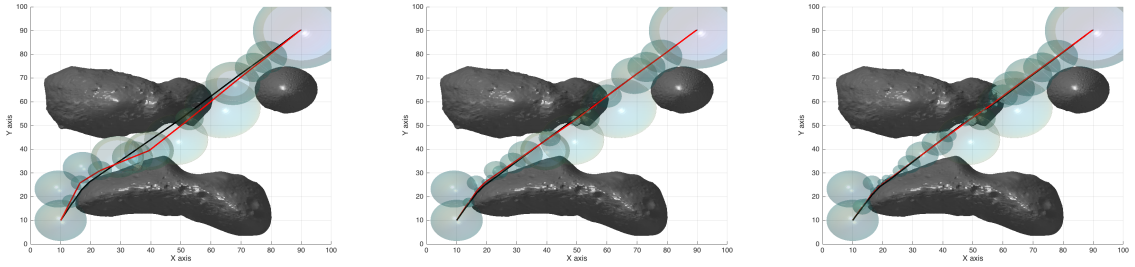
IV.B. Comparison with RRT* and PRM*

In this subsection, we compare the SE–SCP algorithm with the RRT* and PRM* algorithms.[4] These algorithms cannot directly incorporate the spacecraft dynamics. Therefore, we use the single integrator dynamics which is given by:

$$\dot{\boldsymbol{x}} = \boldsymbol{u}\,. \tag{8}$$



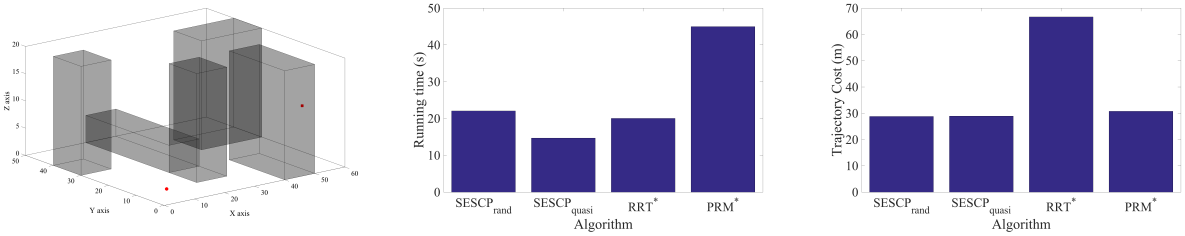
(9a) (9b) (9c)



(9d) (9e) (9f)

Figure 9: Trajectories are further optimized in the SCP step

The 3D benchmark environments are shown in Figs. 10–14. Each of the algorithms are executed multiple times and the histogram plots show the average computational runtime and trajectory cost. The SE–SCP's results with both random and quasi-random sampling are shown.



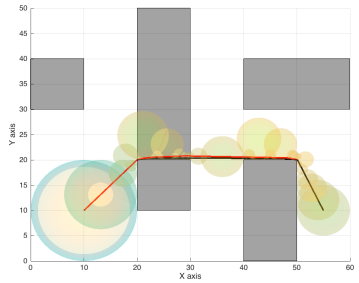
(10a) Benchmark Environment (10b) Running Time (10c) Trajectory Cost

Figure 10: Benchmark comparison room 1. The simulation is run with 500 samples.

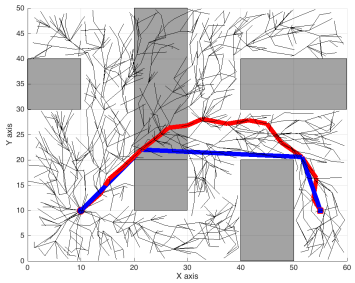
We conclude from these comparisons that the SE–SCP algorithm outperforms the the RRT* and PRM* algorithms. The main reasons for the superior performance of the SE–SCP algorithm are as follows:

- The spherical expansion step in the SE–SCP algorithm adapts with the density of the obstacles in the environment. Since there is no pre-defined step-length in the SE–SCP algorithm, the algorithm can build larger spheres when there are no nearby obstacles. This is extremely helpful in quickly covering the workspace and possibly generating a simple path from the start to the goal position.
- The SCP step generates the locally optimal trajectory within the homotopy class that satisfies actually dynamics, even if the original path is not close to the local optimal. This is useful in reducing the number of samples and computational time necessary to find the optimal path, while reducing the trajectory cost.

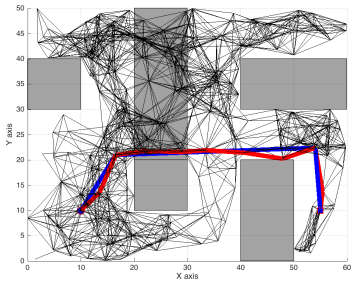
Therefore, the SE–SCP algorithm is suitable for generating optimal paths through cluttered environments while obeying the spacecraft dynamics.



(11a) SESCP Optimized trajectory

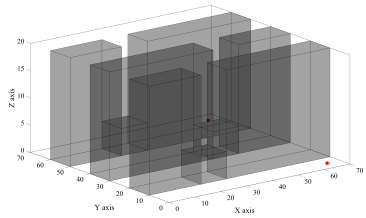


(11b) RRT* trajectory

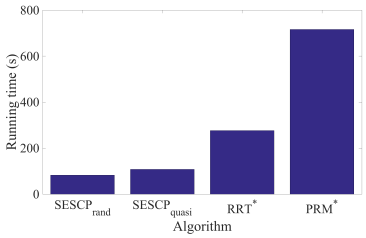


(11c) PRM* trajectory

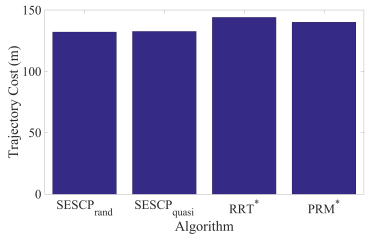
Figure 11: Trajectories in benchmark comparison room 1.



(12a) Benchmark Environment

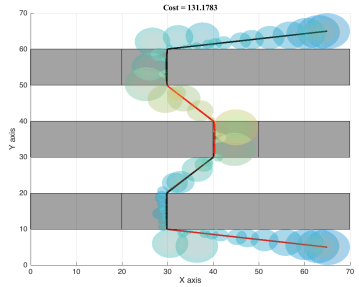


(12b) Running Time

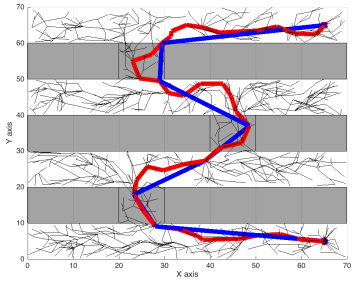


(12c) Trajectory Cost

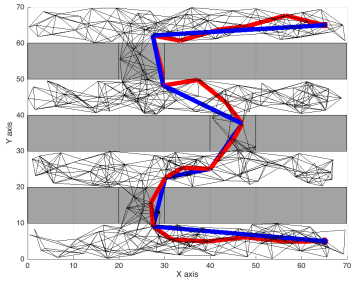
Figure 12: Benchmark comparison room 2. The simulation is run with 2000 samples.



(13a) SESCP Optimized trajectory

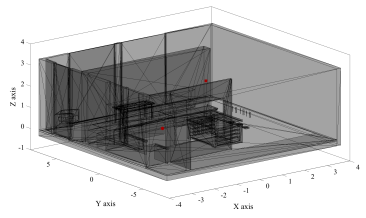


(13b) RRT* trajectory

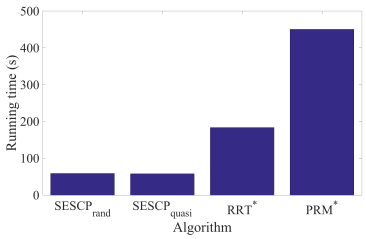


(13c) PRM* trajectory

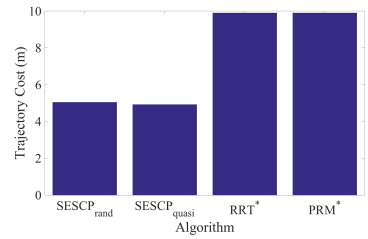
Figure 13: Trajectories in benchmark comparison room 2.



(14a) ROS Kitchen environment

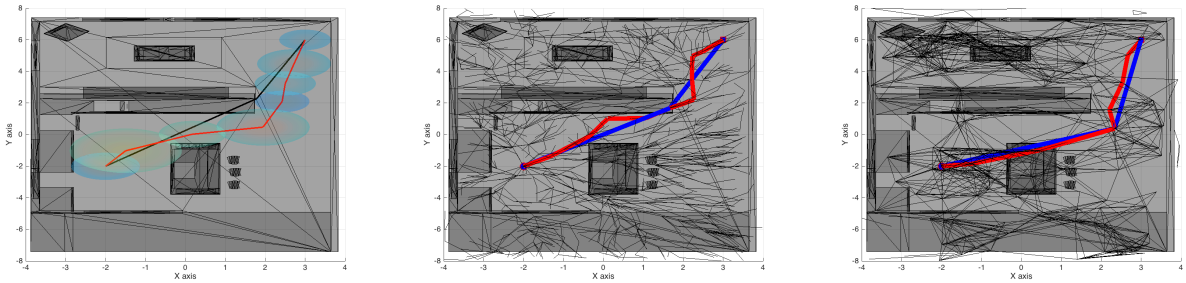


(14b) Running Time



(14c) Trajectory Cost

Figure 14: Benchmark comparison ROS kitchen environment. The simulation is run with 1000 samples.



(15a) SESCP Optimized trajectory (15b) RRT* trajectory (15c) PRM* trajectory

Figure 15: Trajectories in benchmark comparison ROS kitchen environment.

V. Conclusions

In this paper we presented a novel spacecraft trajectory planning algorithm through uncooperative cluttered environments that uses the spacecraft dynamics and minimizes the fuel consumed by the spacecraft. Our SE–SCP algorithm mainly has two steps. During the spherical expansion step, the algorithm explores the workspace using a random or quasi-random sampling technique. This step generates coarse geometric paths from the start position to the goal position. During the SCP step, the locally optimal trajectories are generated along these paths using SCP optimization. Therefore, as newer geometric paths are discovered by the spherical expansion step, better trajectories are generated by the SCP step. Hence the algorithm guarantees locally optimal trajectories using a finite number of samples and a globally optimal trajectory as the number of samples tends to infinity.

Using numerical simulations, we have shown that the SE–SCP algorithm outperforms the RRT* and PRM* algorithms in terms of the computational run time and the cost of the final trajectory. This is because the spherical expansion step in the SE–SCP algorithm adapts with the density of the obstacles in the environment, hence the workspace can be quickly covered with spheres. Moreover, the SCP step generates a locally optimal trajectory, which reduces the number of samples and computational time necessary to find the optimal path while reducing the trajectory cost. Finally, we show that the SE–SCP algorithm generates a locally optimal trajectory for the spacecraft to navigate the debris field. Therefore, the SE–SCP algorithm is suitable for spacecraft trajectory planning through uncooperative cluttered environments. Future work will focus on generating trajectories through non-stationary obstacles.

Appendix: Spacecraft Dynamics in LVLH Frame

In this section, we assume that the spacecraft is in Earth orbit near a real/virtual chief. We present the spacecraft dynamics with respect to this chief spacecraft. The spacecraft has three translational degrees of freedom and three control inputs. We are neglecting the attitude kinematics and dynamics of the spacecraft here.

Let us assume that there exists a real/virtual chief spacecraft in the Earth Centered Inertial (ECI) coordinate system, as shown in Fig. 16. The origin of ECI frame is located at the center of Earth, the $\hat{X}$ axis points towards the vernal equinox and the $\hat{Z}$ axis points towards the north pole. The Local Vertical Local Horizontal (LVLH) coordinate system, also shown in Fig. 16, is centered at the chief spacecraft and is designed to model the relative motion of all other deputy spacecraft with respect to this chief spacecraft.

The $\hat{x}$ direction is always aligned with the position vector from the center of Earth to the chief spacecraft and points away from the Earth, the $\hat{z}$ direction is aligned with the angular momentum vector, and the $\hat{y}$ direction completes the right–handed coordinate system.

The relative position and velocity of the spacecraft are expressed by $\vec{l} = [x \quad y \quad z]^T$ and $\dot{\vec{l}} = [\dot{x} \quad \dot{y} \quad \dot{z}]$ respectively. The orbital elements of the chief æ $= [r, v_x, h, i, \Omega, \theta]^T$ in the ECI frame are defined using hybrid orbital elements which include: geocentric distance r, radial velocity v_x, angular momentum h, inclination i, right ascension of the ascending node Ω and argument of latitude θ.

The J_2 disturbance will cause the chief orbit to drift relative to the Keplerian orbit. Spacecraft with

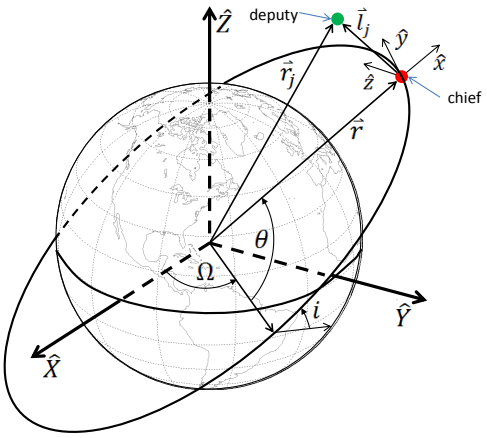


Figure 16: Earth Centered Inertial $(\hat{X}, \hat{Y}, \hat{Z})$ and LVLH $(\hat{x}, \hat{y}, \hat{z})$ frames (Reproduced with permission from Ref. 2)

different orbital periods will rapidly drift apart.

Since the spacecraft is initialized at a position which is within a few kilometers of the chief orbit, initial condition are derived using the Keplerian dynamics.

The differential equation describing the motion of the chief orbit are as follows:

$$\dot{r} = v_x \tag{9}$$

$$\dot{v_x} = -\frac{\mu}{r^2} + \frac{h^2}{r^3} - \frac{k_{J_2}}{r^4}(1 - 3\sin^2 i \sin^2\theta) - C\|\vec{V_a}\|v_x \tag{10}$$

$$\dot{h} = -\frac{k_{J_2}\sin^2 i \sin 2\theta}{r^3} - C\|\vec{V_a}\|(h - \omega_e r^2 \cos i) \tag{11}$$

$$\dot{\Omega} = -\frac{2k_{J_2}\cos i \sin^2\theta}{hr^3} - \frac{C\|\vec{V_a}\|\omega_e r^2 \sin 2\theta}{2h} \tag{12}$$

$$\dot{i} = -\frac{2k_{J_2}\sin 2i \sin 2\theta}{2hr^3} - \frac{C\|\vec{V_a}\|\omega_e r^2 \sin i \cos^2\theta}{h} \tag{13}$$

$$\dot{\theta} = \frac{h}{r^2} + -\frac{2k_{J_2}\cos^2 i \sin^2\theta}{hr^3} - \frac{C\|\vec{V_a}\|\omega_e r^2 \cos i \sin 2\theta}{2h} \tag{14}$$

where ω_e is the rotation vector of the Earth, $k_{J2} = \frac{3}{2}J_2\mu R_e^2, 2.633\times10^10[km^5/s^2]$, $\mu = 398600.4418[km^3/s^2]$ is the gravitational constant , R_e is the radius of the Earth, $\vec{V_a}$ is the velocity of the chief spacecraft relative to the atmosphere, $C = \frac{1}{2}C_d\frac{A}{m}\rho$ is a constant used to simplify the drag expression.

The equations of motion for spacecraft are given by

$$\ddot{l} = -2S(\omega)\dot{l} - g(l, æ) + u \tag{15}$$

where the the matrix $S(\omega) \in \mathbb{R}^{3\times3}$ is defined as $S(\omega)\dot{l} = \omega \times \dot{l}$ and the function $g(l, æ) \in \mathbb{R}^3$ is expressed as follows:

$$g(l,æ) = \begin{bmatrix} \eta^2 - \omega_z & -\alpha_z & \omega_x\omega_z \\ \alpha_z & \eta^2 - \omega_z^2 - \omega_x^2 & -\alpha_x \\ \omega_x\omega_z & \alpha_x & \eta^2 - \omega^2 \end{bmatrix} l + (\zeta - \zeta_c) \begin{pmatrix} \sin i \sin\theta \\ \sin i \cos\theta \\ \cos i \end{pmatrix} + \begin{pmatrix} r^2(\eta - \eta_c \\ 0 \\ 0 \end{pmatrix} + u_{chief} \tag{16}$$

where

$$\zeta_c = \frac{2k_{J2}\sin i \sin\theta}{r^4} \tag{17}$$

$$\zeta = \frac{2k_{j_2} r_{sZ}}{r_s^5} \tag{18}$$

$$\eta_c^2 = \frac{\mu}{r^3} + \frac{k_{J2}}{r^5} - \frac{5k_{J_2}\sin^2 i \sin^2\theta}{r^5} \tag{19}$$

$$\eta^2 = \frac{\mu}{r_s^3} + \frac{k_{J2}}{r_s^5} - \frac{5k_{J_2}\sin^2 i \sin^2\theta}{r_{sZ}^5} \tag{20}$$

$$r = \sqrt{(r+x)^2 + y^2 + z^2} \tag{21}$$

$$r_{sZ} = (r+x)\sin i \sin\theta + y \sin i \cos\theta + z\cos i \tag{22}$$

$$\omega_x = -\frac{k_{J_2}\sin 2i \sin\theta}{hr^3} + \frac{r}{h}u_n \tag{23}$$

$$\omega_z = \frac{h}{r^2} \tag{24}$$

$$a_x = \dot{\omega}_x \tag{25}$$

$$a_z = \dot{\omega}_z \tag{26}$$

and (a_x, a_y, a_z) are the angular acceleration of coordinate system about $(x \quad y \quad z)$ axes, r is the geocentric distance, r_{sZ} is the distance from spacecraft to equator and ω_z and ω_z are the rotation rate of coordinate system about x-axis and z-axis respectively.

Eq. 15 can be rewritten as follows:

$$\dot{x} = f(x,æ) + Bu \tag{27}$$

where $B = [0_{3\times3} \quad I_{3\times3}]$. Linearizing Eq. (27) yields

$$\dot{x} = A(\overline{x},æ)x + Bu \tag{28}$$

where $\overline{x}$ is the nominal trajectory about which equations are linearized and

$$A(\overline{x},æ) = \begin{pmatrix} 0_{3\times3} & I_3 \\ -\frac{\partial g}{\partial x}|_{\overline{x}} & -2S(\omega) \end{pmatrix} \tag{29}$$

Acknowledgment

This work was supported by the Jet Propulsion Laboratory's Research and Technology Development (R&TD) program. Part of the research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1]Bandyopadhyay, S., Foust, R., Subramanian, G. P., Chung, S.-J., and Hadaegh, F. Y., "Review of Formation Flying and Constellation Missions using Nanosatellites," *J. Spacecraft and Rockets*, Vol. 53, No. 3, 2016, pp. 567–578, to appear.

[2]Morgan, D., Chung, S.-J., Blackmore, L., Acikmese, B., Bayard, D., and Hadaegh, F. Y., "Swarm-Keeping Strategies for Spacecraft under J2 and Atmospheric Drag Perturbations," *J. Guid. Control Dyn.*, Vol. 35, No. 5, 2012, pp. 1492 – 1506.

[3]Morgan, D., Subramanian, G. P., Chung, S.-J., and Hadaegh, F. Y., "Swarm Assignment and Trajectory Optimization Using Variable-Swarm, Distributed Auction Assignment and Sequential Convex Programming," *Int. J. Robotics Research*, Vol. 35, No. 10, 2016, pp. 1261–1285.

[4]Karaman, S. and Frazzoli, E., "Sampling-based algorithms for optimal motion planning," *Int. J. Robotics Research*, Vol. 30, No. 7, 2011, pp. 846–894.

[5]Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, 1996, pp. 566–580.

[6]LaValle, S. M. and Kuffner, J. J., "Randomized kinodynamic planning," *Int. J. Robotics Research*, Vol. 20, No. 5, 2001, pp. 378–400.

[7]Janson, L., Schmerling, E., Clark, A., and Pavone, M., "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robotics Research*, Vol. 34, No. 7, 2015, pp. 883–921.

[8]LaValle, S. M., Branicky, M. S., and Lindemann, S. R., "On the relationship between classical grid search and probabilistic roadmaps," *Int. J. Robotics Research*, Vol. 23, No. 7-8, 2004, pp. 673–692.

[9]Lindemann, S. R., Yershova, A., and LaValle, S. M., *Incremental Grid Sampling Strategies in Robotics*, Springer, 2004.

[10]Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *J. Guid. Control Dyn.*, Vol. 37, No. 6, 2014, pp. 1725–1740.