DISTRIBUTED FAST MOTION PLANNING FOR SPACECRAFT SWARMS IN CLUTTERED ENVIRONMENTS USING SPHERICAL EXPANSIONS AND SEQUENCE OF CONVEX OPTIMIZATION PROBLEMS

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

This paper presents a novel guidance algorithm for spacecraft swarms in an environment cluttered with many obstacles like a debris field or the asteroid belt. The objective of this algorithm is to reconfigure the swarm to a desired formation in a distributed manner while minimizing fuel and avoiding collisions among themselves and with the obstacles. The agents first use a spherical-expansion-based sampling algorithm to cooperatively explore the workspace and find paths to the desired terminal positions. Using a distributed assignment algorithm, the agents converge on an optimal assignment of the target locations in the desired formation. Then each agent generates a locally optimal trajectory from its current location to its terminal position by solving a sequence of convex optimization problems. As the agent moves along this trajectory, it receives the position of other agents and updates its trajectory to avoid collisions with other agents and the obstacles. Thus the swarm achieves the desired formation in a distributed manner while avoiding collisions. Moreover, this algorithm is computationally efficient, therefore it can be implemented onboard resource-constrained spacecraft. Simulations results show that the proposed distributed algorithm can be used by a spacecraft swarm to reconfigure a desired formation around an asteroid in a collision-free manner.

INTRODUCTION

Trajectory planning for multi-spacecraft formations and swarms, composed of hundreds to thousands of spacecraft, has been a major area of research over the past decade.[1–7] Although there have been significant advances in the development of swarm guidance algorithms for cooperative spacecraft, they cannot be directly applied to handle uncooperative obstacles. In this paper, we present a novel guidance algorithm for spacecraft swarms in an environment cluttered with many obstacles like a debris field or the asteroid belt. The objective of this algorithm is to reconfigure the swarm to

[*]Robotics Technologist, Jet Propulsion Laboratory (JPL), California Institute of Technology (Caltech), Pasadena, California, 91109, USA; Saptarshi.Bandyopadhyay@jpl.nasa.gov

[†]Graduate Student, Department of Aerospace, Caltech, Pasadena, California, 91125, USA; fbaldini@caltech.edu

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

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

[¶]Robotics Systems Engineer, JPL, Caltech, Pasadena, California, 91109, USA; Amir.Rahmani@jpl.nasa.gov

[‖]Robotics Systems Engineer, JPL, Caltech, Pasadena, California, 91109, USA; Jean-Pierre.de.la.Croix@jpl.nasa.gov

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

a desired formation in a distributed manner while minimizing fuel and avoiding collisions among themselves and with the obstacles.

In the robotics literaure, sampling-based motion planning algorithms have been used to plan trajectories through cluttered envionements.[8–11] Our prior work (Ref. 12) presents a sampling-optimization-based algorithm for planning the trajectory of a single spacecraft through a 3D cluttered environment. This novel spacecraft trajectory planning algorithm, dubbed the Spherical Expansion and Sequential Convex Programming (SE–SCP) algorithm, 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).[12] The SE–SCP algorithm first explores the 3D workspace using spherical expansions (as shown in Fig. 1) to generate a feasible path from the start position to the target position. Then the algorithm generates a fuel-optimal trajectory using a sequence of convex optimization problems (as shown in Fig. 2). This trajectory is locally optimal within its homotopy class and is globally optimal as the number of samples in the spherical expansion step tends to infinity.

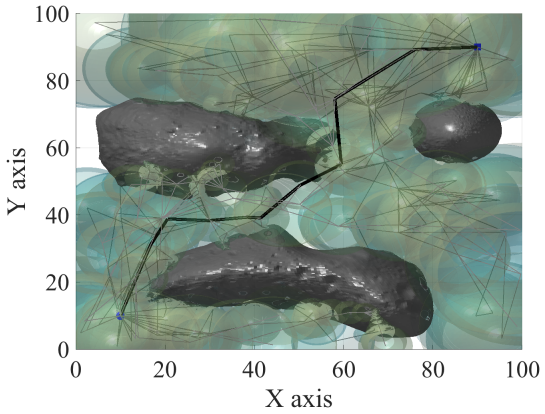

Figure 1. Spherical expansion is used to find a path from start to target position[12]

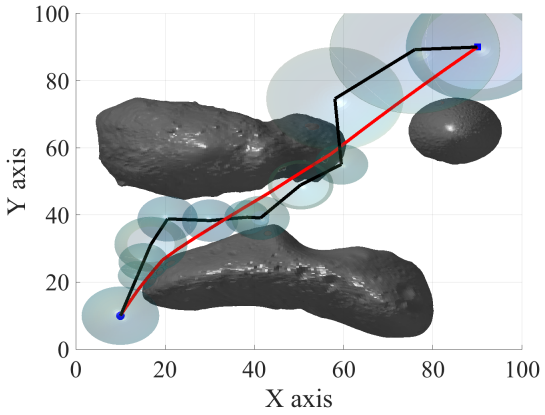

Figure 2. Sequence of convex optimization problems generates a locally optimal trajectory[12]

In this paper, we extend the SE–SCP algorithm for distributed trajectory planning of spacecraft swarms in cluttered environments. The main challenges that arise include: (i) The spacecraft or

agents need to explore the entire workspace in a cooperative manner because it might be impossible or highly inefficient to explore the entire workspace alone. (ii) The agents need to optimally assign their target positions among themselves. (iii) While traveling to their assigned target position, the agents need to avoid collisions with the obstacles and among themselves. These challenges are addressed in this paper.

This paper is organized as follows. The problem statement and the multi-agent SE–SCP algorithm are described in Section 2 and 3 respectively. Numerical simulations are presented in Section 4 and the paper is concluded in Section 5.

PROBLEM STATEMENT

Let $\mathcal{X} \subset \mathbb{R}^3$ represent the 3D workspace in Local-Vertical-Local-Horizontal (LVLH) frame, as shown in Fig. 3. Let $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ represent the stationary obstacles in this workspace. The region where the swarm can maneuver freely is given by $\mathcal{X}_{\text{free}} = \mathcal{X}/\mathcal{X}_{\text{obs}}$. We assume that $\mathcal{X}_{\text{obs}}$ is known to each agent.

Let N agents belong to this swarm. The initial positions of these N agents are given by $X^i_{\text{init}} \in \mathcal{X}$ for all $i \in \{1, \dots, N\}$. Note that the agent index is denoted by the superscript. Similarly, the N terminal positions are given by $X^j_{\text{goal}} \in \mathcal{X}$ for all $j \in \{1, \dots, N\}$. The actual assignment of the agents to terminal positions will be performed later because the cost-to-go for each agent cannot be calculated beforehand on account of the obstacles. We assume that $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ and $X^j_{\text{goal}} \in \mathcal{X}$ for all $j \in \{1, \dots, N\}$ are known to each agent.

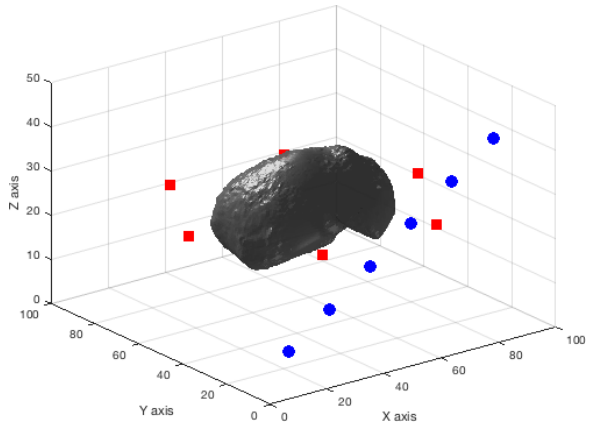

Figure 3. The 3D workspace $\mathcal{X}$, the obstacles $\mathcal{X}_{\text{obs}}$, the initial positions $X^i_{\text{init}}, \forall i \in \{1, \dots, N\}$ (in blue), and the terminal positions $X^j_{\text{goal}}, \forall j \in \{1, \dots, N\}$ (in red) are shown for $N = 6$ agents.

To avoid inter-agent collisions, each agent must maintain at least r_{col} distance with every other agent in the swarm. Moreover, let $r_{\max}$ represent the maximum distance that any agent can travel in any time instant. We assume that the initial and final positions satisfy this collision avoidance constraints, i.e.,

$$\left\| X^i_{\text{init}} - X^\ell_{\text{init}} \right\|_2 \geq r_{\text{col}} + r_{\max}\,, \qquad \forall i, \ell \in \{1, \dots, N\}, i \neq \ell \tag{1}$$

$$\left\| X^j_{\text{goal}} - X^\ell_{\text{goal}} \right\|_2 \geq r_{\text{col}} + r_{\max}\,, \qquad \forall j, \ell \in \{1, \dots, N\}, j \neq \ell \tag{2}$$

The objective of the Multi-Agent Spherical Expansion and Sequential Convex Programming (Multi-Agent SE–SCP) algorithm is to ensure that all the N agents reach the N terminal positions while avoiding collisions with the obstacles and among themselves.

Algorithm 1 Multi-Agent SE–SCP Algorithm for the i^{th} agent

$r^i_{\text{init}} \leftarrow \texttt{MinDistObs}(X^i_{\text{init}}, \mathcal{X}_{\text{obs}}), \quad \mathcal{V}^i \leftarrow \{X^i_{\text{init}}[r^i_{\text{init}}]\}$ ▷ ***Initialization step***
for $j = \{1, \dots, N\}$ **do**
  $r^j_{\text{goal}} \leftarrow \texttt{MinDistObs}(X^j_{\text{goal}}, \mathcal{X}_{\text{obs}}), \quad \mathcal{V}^i \leftarrow \mathcal{V}^i \cup \{X^j_{\text{goal}}[r^j_{\text{goal}}]\}$
end for
$\mathcal{E}^i \leftarrow \emptyset$, $F^i_{\text{reached}} \leftarrow 0$, $F^i_{\text{connected}} \leftarrow 0$, $X^i_{\text{term}} \leftarrow \emptyset$
while $F_{\text{reached}} \neq 1$ **do**
  $Y^\ell, X^\ell_{\text{new}}, F^\ell_{\text{connected}}, \forall \ell \in \{1, \dots, N\} \leftarrow \texttt{AllAgentCommunicate}$ ▷ ***Spherical Expansion step***
  $\hat{\mathcal{X}}^i_{\text{obs}} = \mathcal{X}_{\text{obs}}$
  for $\ell = \{1, \dots, N\}/\{i\}$ **do**
    $\hat{\mathcal{X}}^i_{\text{obs}} = \hat{\mathcal{X}}^i_{\text{obs}} \cup \texttt{GenerateSphere}(Y^\ell, r_{\text{col}} + r_{\max}), \quad \mathcal{V}^i \leftarrow \mathcal{V}^i \cup \{X^\ell_{\text{new}}[0]\}$
  end for
  $\mathcal{V}^i_{\text{new}} \leftarrow \emptyset$
  for all $X_{\text{v}}[r_{\text{v}}] \in \mathcal{V}^i$ **do**
    $r_{\text{v}} \leftarrow \texttt{MinDistObs}(X_{\text{v}}, \hat{\mathcal{X}}^i_{\text{obs}}), \quad \mathcal{V}^i_{\text{new}} \leftarrow \mathcal{V}^i_{\text{new}} \cup \{X_{\text{v}}[r_{\text{v}}]\}$
  end for
  $\mathcal{V}^i \leftarrow \mathcal{V}^i_{\text{new}}$
  $X_{\text{rand}} \leftarrow \texttt{GenerateSample}$
  $X_{\text{nearest}} \leftarrow \texttt{NearestNode}(\mathcal{V}^i, X_{\text{rand}})$
  $X^i_{\text{new}} \leftarrow \texttt{Steer}(X_{\text{rand}}, X_{\text{nearest}})$
  $r^i_{\text{new}} \leftarrow \texttt{MinDistObs}(X^i_{\text{new}}, \hat{\mathcal{X}}^i_{\text{obs}}), \quad \mathcal{V}^i \leftarrow \mathcal{V}^i \cup \{X^i_{\text{new}}[r^i_{\text{new}}]\}$
  $\mathcal{E}^i \leftarrow \emptyset$
  for all $X_{\text{v}}[r_{\text{v}}], X_{\text{w}}[r_{\text{w}}] \in \mathcal{V}^i$ and $X_{\text{v}} \neq X_{\text{w}}$ **do**
    if $\|X_{\text{v}} - X_{\text{w}}\|_2 \leq r_{\text{v}} + r_{\text{w}}$ **then**
      $c_{\text{v,w}} \leftarrow \texttt{EdgeCost}(X_{\text{v}}, X_{\text{w}}), \quad c_{\text{w,v}} \leftarrow \texttt{EdgeCost}(X_{\text{w}}, X_{\text{v}})$
      $\mathcal{E}^i \leftarrow \mathcal{E}^i \cup \{\overrightarrow{X_{\text{v}}\, X_{\text{w}}}[c_{\text{v,w}}]\} \cup \{\overrightarrow{X_{\text{w}}\, X_{\text{v}}}[c_{\text{w,v}}]\}$
    end if
  end for
  if $X^i_{\text{term}} = \emptyset$ **then** ▷ ***Sequential Convex Programming step***
    if $\sum_{\ell=1}^{N} F^\ell_{\text{connected}} = N^2$ **then**
      $X^i_{\text{term}} \leftarrow \texttt{DistributedAssignment}, \quad \mathcal{V}^i \leftarrow \mathcal{V}^i \cup \{X^i_{\text{term}}[0]\}$
    else
      $F^i_{\text{connected}} \leftarrow 0$
      for $j = \{1, \dots, N\}$ **do**
        $P^{i,j}, c_{P^{i,j}} \leftarrow \texttt{MinPath}(\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i), X^i_{\text{init}}, X^j_{\text{goal}})$
        if $c_{P^{i,j}} < \infty$ **then**
          $F^i_{\text{connected}} \leftarrow F^i_{\text{connected}} + 1$
        end if
      end for
    end if
  else
    if $X^i_{\text{term}} = Y^i$ **then**
      $F^i_{\text{reached}} \leftarrow 1, \quad \boldsymbol{x}^i \leftarrow \emptyset$
    else
      $F^i_{\text{reached}} \leftarrow 0$
      $P^i, c_{P^i} \leftarrow \texttt{MinPath}(\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i), Y^i, X^i_{\text{term}}), \quad (\boldsymbol{x}^i_1, \boldsymbol{u}^i_1, c_{\boldsymbol{x}^i_1}) \leftarrow \texttt{OptimalTraj}(P^i)$
      for $k = \{1, \dots, N_{SCP}\}$ **do**
        $P^i_k \leftarrow \texttt{GeneratePath}(\boldsymbol{x}^i_k), \quad (\boldsymbol{x}^i_{k+1}, \boldsymbol{u}^i_{k+1}, c_{\boldsymbol{x}^i_{j+1}}) \leftarrow \texttt{OptimalTraj}(P^i_k, \boldsymbol{x}^i_k, \boldsymbol{u}^i_k)$
      end for
      $\boldsymbol{x}^i \leftarrow \boldsymbol{x}^i_{N_{SCP}+1}$
    end if
  end if
  $Y^i \leftarrow \texttt{AgentMotion}(\boldsymbol{x}^i), \quad \mathcal{V}^i \leftarrow \mathcal{V}^i \cup \{Y^i[0]\}$
end while

MULTI-AGENT SPHERICAL EXPANSION AND SEQUENTIAL CONVEX PROGRAMMING ALGORITHM

The Multi-Agent SE–SCP algorithm's pseudocode for a single agent is presented in Algorithm 1. During the *Initialization step*, the necessary data structures are created and initialized. Then the *Spherical Expansion step* and the *Sequential Convex Programming step* are executed iteratively until the agent reaches its terminal position.

Initialization Step

The i^{th} agent's Multi-Agent SE–SCP algorithm intends to generates a directed graph $\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i)$ in the safe region $\mathcal{X}_{\text{free}}$. Each node in the set of nodes $\mathcal{V}^i$ stores the position of the node and the minimum distance of that node from any obstacle (both in $\mathcal{X}_{\text{obs}}$ and other agents). For the node X^i_{init}, the minimum distance r^i_{init} from the obstacle $\mathcal{X}_{\text{obs}}$ is obtained using the function `MinDistObs`$(X_{\text{init}}, \mathcal{X}_{\text{obs}})$, which takes in the position of the node and the obstacles in the workspace and returns the radius of the largest sphere centered on that node which does not intersect with any obstacle. Similarly, the minimum distances r^j_{goal} from the obstacle $\mathcal{X}_{\text{obs}}$ is obtained for all the terminal positions X^j_{goal} for all $j \in \{1, \ldots, N\}$. Then the set of nodes $\mathcal{V}^i$ is initialized with the nodes $X^i_{\text{init}}[r^i_{\text{init}}]$ and $X^j_{\text{goal}}[r^j_{\text{goal}}]$ for all $j \in \{1, \ldots, N\}$.

Each element in the set of edges $\mathcal{E}^i$ stores the edge's starting and ending nodes and the cost of traversing that edge. The set of edges $\mathcal{E}^i$ is initialized with the empty set. The flag F^i_{reached}, which denotes if the agent has reached its terminal position, is set to zero. The flag $F^i_{\text{connected}}$, which denotes the number of terminal positions that the agent is connected to, is also set to zero. The assigned terminal position X^i_{term} of the i^{th} agent is set to an empty set.

Spherical Expansion Step

During this step, the workspace is explored using the sampling technique shown in lines 7–27 in Algorithm 1. The objective of this step is to populate the graph $\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i)$ so that paths from X^i_{init} to X^j_{goal} for all $j \in \{1, \ldots, N\}$ can be found.

Let $Y^\ell \in \mathcal{X}$ for all $\ell \in \{1, \ldots, N\}$ represent the current position of each agent. The agents exchange their position Y^ℓ, their new nodes X^ℓ_{new}, and their flag $F^\ell_{\text{connected}}$ using the function `AllAgentCommunicate`, which relies on inter-agent communication and a strongly-connected communication network topology. During the first iteration, no new nodes are communicated. Communicating only the new nodes, as opposed to complete trajectories or other features of the obstacles, allows the agents to collaboratively explore the workspace in a computationally efficient manner.

The lines 8–11 create a new obstacle set $\hat{\mathcal{X}}^i_{\text{obs}}$ where the original obstacle set $\mathcal{X}_{\text{obs}}$ is augmented with spheres of radius $(r_{\text{col}} + r_{\max})$ centered on the position of all the other agents. Thus $\mathcal{X}^i_{\text{free}} = \mathcal{X} / \hat{\mathcal{X}}^i_{\text{obs}}$ represents the region where the i^{th} agent can maneuver freely.

The new nodes from other agents $X^\ell_{\text{new}}, \forall \ell \in \{1, \ldots, N\}$ are also added to $\mathcal{V}^i$ during lines 8–11. The lines 12–16 are used to update the radius of the nodes in $\mathcal{V}^i$ with the new obstacles set $\hat{\mathcal{X}}^i_{\text{obs}}$.

The Multi-Agent SE–SCP algorithm can use both random or quasi-random (deterministic) sampling since the randomness of the samples is not crucial for motion planning applications.[13] For a given sampling choice, the function `GenerateSample` returns a random sample $X_{\text{rand}} \in \mathcal{X}$. Next, the function `NearestNode`$(\mathcal{V}^i, X_{\text{rand}})$ takes in the current set of nodes $\mathcal{V}^i$ and the given sample

Figure 4. Multiple iterations of the Spherical Expansion step are shown. All agents (in magenta) are located at their starting positions.

X_{rand} and returns the node X_{nearest} that is nearest to X_{rand}. Note that the minimum distance from the obstacles r_{nearest} for the node X_{nearest} is already stored in $\mathcal{V}^i$.

The function $\texttt{Steer}(X_{\text{rand}}, X_{\text{nearest}})$ generates the new point X^i_{new} according to either of the following two cases: (i) If X_{rand} is serendipitously inside X_{nearest}'s sphere, then new point $X^i_{\text{new}} = X_{\text{rand}}$. (ii) Otherwise, the new point X^i_{new} is on the surface of X_{nearest}'s sphere and closest to the sample X_{rand}. In contrast with the classical steering function in the literature,[8] where the step-

Figure 5. Multiple iterations of the Spherical Expansion step and the Sequential Convex Programming step are shown. All agents (in magenta) are moving to their terminal positions.

size of the algorithm is fixed, the radius of the sphere in the Multi-Agent SE–SCP algorithm is variable and adapts with the density of obstacles. Therefore, the Multi-Agent SE–SCP algorithm generally finds a feasible path faster than other sampling-based algorithms. The minimum distance from obstacles r^i_{new} for the point X^i_{new} is computed using the function `MinDistObs`. The new node $X^i_{\text{new}}[r^i_{\text{new}}]$ is added to the set of nodes $\mathcal{V}^i$.

Finally, the lines 21–27 are used to generate the new edge set $\mathcal{E}^i$. There exists a feasible collision-free path between each vertex in $\boldsymbol{X}_\mathrm{v}$ and X_w because their spheres intersect. We generate the edge that connect these nodes using the function $\texttt{EdgeCost}(X_\mathrm{v}, X_\mathrm{w})$, which takes in the two nodes and outputs the cost of traversing the directed edge $c_{\mathrm{v,w}}$ from X_v to X_w and it depends on the given convex objective function, such that the trajectory always remains inside the union of the two spheres. Then the directed edge $\overrightarrow{X_\mathrm{v}\, X_\mathrm{w}}[c_{\mathrm{v,w}}]$ is added to the set of edges $\mathcal{E}^i$. Similarly, the cost for traversing the directed edge $c_{\mathrm{w,v}}$ from X_w to X_v is generated and this new directed edge $\overrightarrow{X_\mathrm{w}\, X_\mathrm{v}}[c_{\mathrm{w,v}}]$ is added to $\mathcal{E}^i$. These costs are used for evaluating the function `MinPath` described below.

For the problem setup shown in Fig. 3, multiple iterations of only the Spherical Expansion step are shown in Fig. 4. Note that each agent is able to generate a dense graph within 30 iterations because each agent also uses the nodes generated by other agents.

Sequential Convex Programming Step

During this step in lines 28–52 in Algorithm 1, each agent first determines its terminal position and then moves towards that terminal position by generating locally optimal trajectories.

If the terminal position X^i_term is not yet assigned (line 28), then the agent first checks if all the agents are connected to all the terminal positions. If this is the case, then the agent execute a distributed assignment algorithm to converge on a suitable assignment of terminal positions (line 30). A number of distributed assignment algorithms using linear programming,[14] auction algorithm,[15] and variable-target-number distributed auction algorithm[4] exist in the literature, therefore they are not covered in this paper. Otherwise the i^th agent counts the number of terminal positions it is connected to in lines 32–38. The function $\texttt{MinPath}(\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i), X^i_\mathrm{init}, X^j_\mathrm{goal})$ takes in the current graph, the initial and goal positions, and returns the minimum-cost path $P^{i,j}$ along with the cost of that path $c_{P^{i,j}}$. The path $P^{i,j} = \{X_1[r_1], X_2[r_2], \ldots, X_\mathrm{m}[r_\mathrm{m}]\}$ is a sequence of m nodes with corresponding radii, where $X_1 = X^i_\mathrm{init}$ and $X_\mathrm{m} = X^j_\mathrm{goal}$. The cost of the path $c_{P^{i,j}}$ is the sum of the edges along that path. Graph search algorithms like Dijkstra's algorithm can be used for this step. If no path exists, then $c_{P^{i,j}}$ is set to infinity.

Problem 1: *Discrete-time Convex Optimal Motion Planning Problem*

$$\underset{\substack{\boldsymbol{x}[k],\,\forall k\in\{0,\ldots,T\}\\ \boldsymbol{u}[k],\,\forall k\in\{0,\ldots,T-1\}}}{\text{minimize}} \quad \sum_{k=0}^{T-1} c\left(\boldsymbol{u}[k]\right)\Delta\,, \tag{3}$$

$$\text{subject to } \boldsymbol{p}[0] = Y^i\,, \tag{4}$$

$$\boldsymbol{p}[T] = X^i_\mathrm{term}\,, \tag{5}$$

$$\|\boldsymbol{p}[2\ell] - X_\ell\|_2 \le r_\ell\,, \qquad \forall \ell \in \{1,\ldots,n-1\}\,, \tag{6}$$

$$\|\boldsymbol{p}[2\ell] - X_{\ell+1}\|_2 \le r_{\ell+1}\,, \qquad \forall \ell \in \{1,\ldots,n-1\}\,, \tag{7}$$

$$\|\boldsymbol{p}[2\ell+1] - X_{\ell+1}\|_2 \le r_{\ell+1}\,, \qquad \forall \ell \in \{1,\ldots,n-2\}\,, \tag{8}$$

$$\boldsymbol{u}[k] \in \mathcal{U}\,, \qquad \forall k \in \{0,\ldots,T-1\}\,, \tag{9}$$

$$\boldsymbol{x}[k+1] = F[k]\,\boldsymbol{x}[k] + G[k]\,\boldsymbol{u}[k] + H[k]\,, \qquad \forall k \in \{0,\ldots,T-1\}\,. \tag{10}$$

If the terminal position X^i_term is assigned (line 40), then the agent checks if it has already reached the terminal position, and sets the flag F_reached accordingly. Once a path from the current position

Y^i to the terminal position X^i_{term} is found, the function $\texttt{OptimalTraj}(P^i)$ takes in this new path P^i and returns the optimal trajectory $\boldsymbol{x}^i_1$, $\boldsymbol{u}^i_1$ and the cost of traversing this trajectory $c_{\boldsymbol{x}^i_1}$ by solving **Problem 1**(3)–(10). Here Δ is the time step of the algorithm. See Appendix for a discussion on linearization and discretization of spacecraft dynamics. All even points are the intersection of two spheres and all odd points are inside one sphere. This process is executed N_{SCP} times so that a locally optimal trajectory $\boldsymbol{x}^i$ is generated.

The agent traverses for one time instant along this locally optimal trajectory $\boldsymbol{x}^i$ and the function $\texttt{AgentMotion}(\boldsymbol{x}^i)$ gives the new current location of the i^{th} agent. Thus the Multi-Agent SE–SCP algorithm ensures that the i^{th} agent reaches its terminal position.

A few steps of the Multi-Agent SE–SCP algorithm are show in Fig. 5. Note that each agent updates its optimal trajectory based on its current location and the location of other agents while avoiding collisions with other agents and the obstacles.

NUMERICAL SIMULATIONS

In this section, we use the Multi-Agent SE–SCP algorithm to reconfigure a swarm of 48 agents into the desired formation shown in Fig. 6. The terminal positions form a spherical formation around the central asteroid in the debris field.

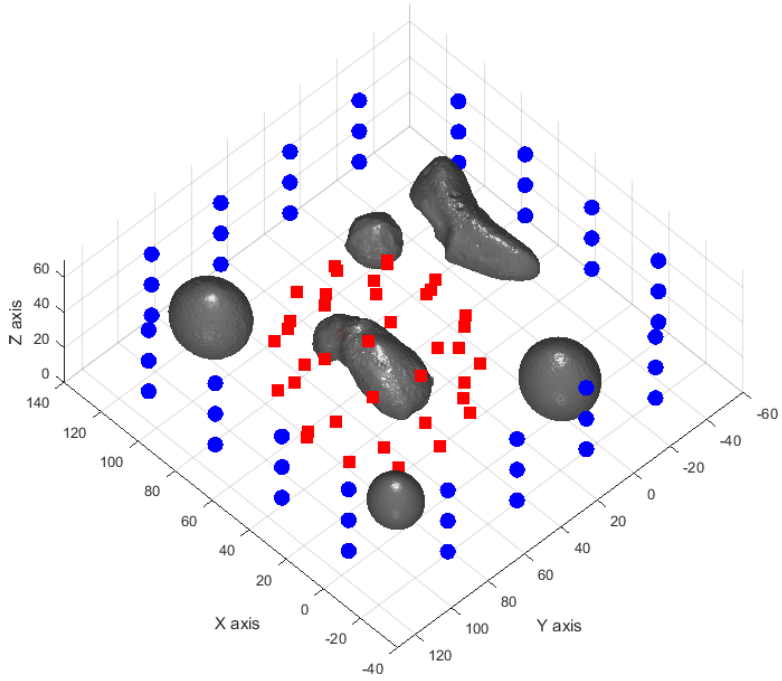

Figure 6. The 3D workspace $\mathcal{X}$, the obstacles $\mathcal{X}_{\text{obs}}$, the initial positions X^i_{init} (in blue), and the terminal positions X^j_{goal} (in red) are shown for $N = 48$ agents.

Multiple iterations of the Multi-Agent SE–SCP algorithm for agents 12, 24, 36, and 48 are shown in Fig. 7 and 8. During the 25 iterations of the Spherical Expansion step, each agent is able to generate a dense graph of the workspace. During the remaining 30 iterations, each agent travels to its chosen terminal position while avoiding collisions with other agents and the obstacles.

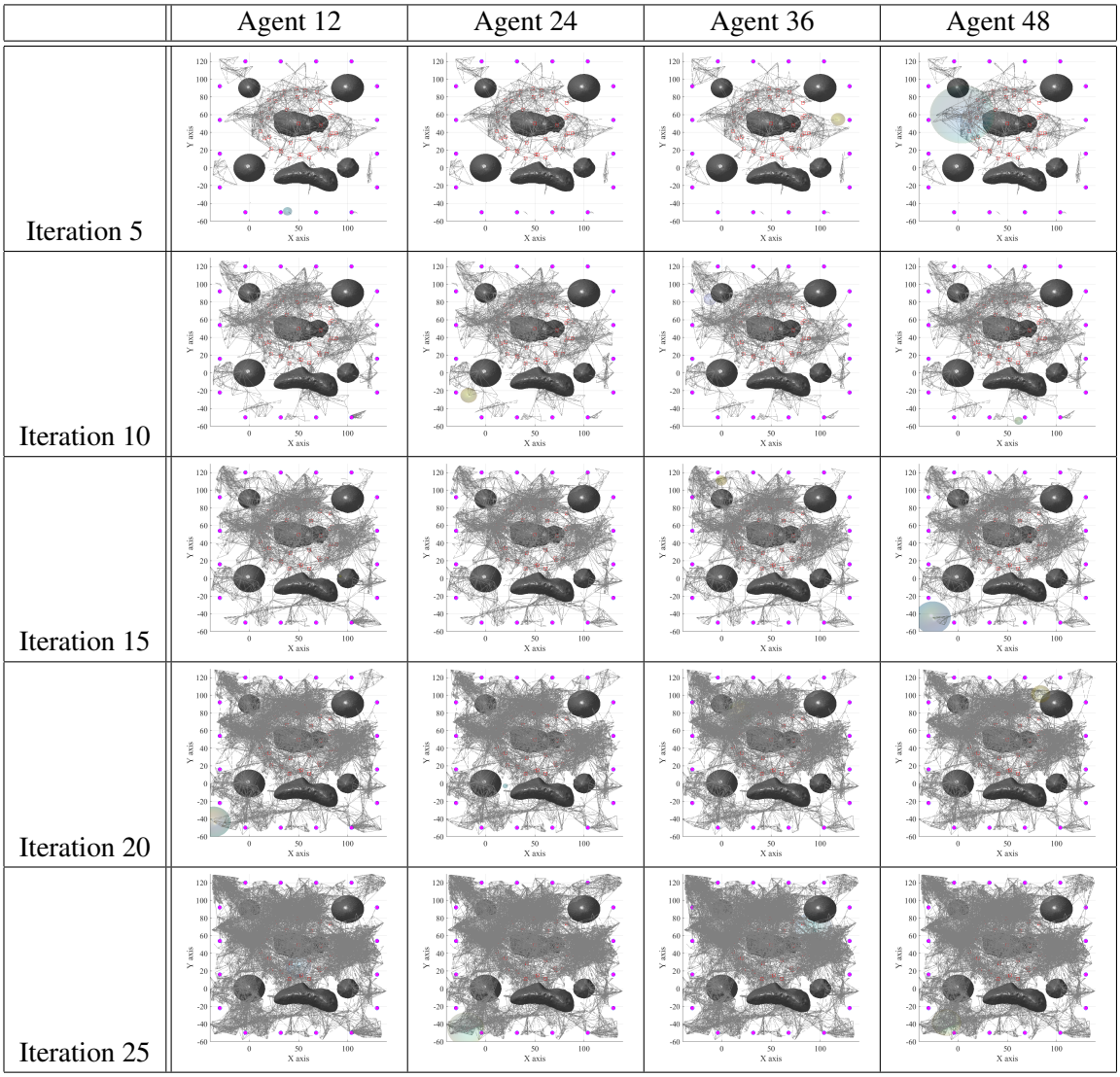

Figure 7. Multiple iterations of the the Spherical Expansion step for agents 12, 24, 36, and 48 are shown. All agents (in magenta) are located at their starting positions.

CONCLUSIONS

In this paper, we presented the Multi-Agent SE–SCP algorithm for motion planning of spacecraft swarm in cluttered environments. In the first step of our algorithm, the agents use a spherical-expansion-based sampling algorithm to cooperatively explore the workspace and map the obstacles in the environment. During the spherical expansion step, each agent stores the position of randomly generated nodes in the free space (the space that is free from obstacles) and the radius of the largest sphere that does not intersect with any obstacle. The agents exchange the positions of the nodes and their radii with their neighboring agents to generate a global view of the workspace while each agent has only explored a much smaller region. This step ensures that all the target positions are strongly connected in the global network of nodes.

Using a distributed assignment algorithm, the agents converge on an optimal assignment of the

Figure 8. Multiple iterations of the entire Multi-Agent SE–SCP algorithm for agents 12, 24, 36, and 48 are shown. The location of the agents are shown using magenta dots.

target locations in the desired formation. The agents use their global network of nodes to approximately determine their distance to each of the target locations. Then each agent generates a locally fuel-optimal trajectory from its current location to its target position using a sequence of convex optimization problems. As the agent moves along this trajectory, it detects the position of other agents and updates its trajectory to avoid collisions with other agents and the obstacles. Thus the swarm achieves the desired formation in a distributed manner while avoiding collisions.

The Multi-Agent SE–SCP algorithm is computationally efficient, therefore it can be implemented onboard resource-constrained spacecraft. Simulations results demonstrate the effectiveness of the proposed distributed algorithm for guidance of spacecraft swarms.

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. ©2017 California Institute of Technology. Government sponsorship acknowledged.

APPENDIX

Linearization and Discretization of Spacecraft Dynamics

Let τ_0 and τ_f represent the starting and final times respectively. Let $\boldsymbol{x}^i(\tau) \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}^i(\tau) \in \mathbb{R}^{n_u}$ represent the i^{th} agents's state vector and control input at time τ. The agent's state vector can be further decomposed into $\boldsymbol{x}^i(\tau) = \left(\boldsymbol{p}^i(\tau)^T, \dot{\boldsymbol{p}}^i(\tau)^T, \boldsymbol{\theta}^i(\tau)^T, \dot{\boldsymbol{\theta}}^i(\tau)^T\right)^T$, where $\boldsymbol{p}^i(\tau) \in \mathbb{R}^3$ and $\boldsymbol{\theta}^i(\tau)$ represent the vehicle's position and attitude vectors. The spacecraft dynamics are given by:

$$\dot{\boldsymbol{x}}^i(\tau) = \boldsymbol{f}\left(\boldsymbol{x}^i(\tau), \boldsymbol{u}^i(\tau)\right), \quad \forall \tau \in [\tau_0, \tau_f], \tag{11}$$

We now transform this continuous-time dynamics into a discrete-time dynamics so that it can be efficiently solved using sequential convex programming. Let $t[k]$, $\forall k \in \{0, \ldots, T\}$ represent the discrete time instants, where T is the number of discrete time steps (i.e., $t[0] = \tau_0$ and $t[T] = \tau_f$). Let Δ represents the time step size (i.e., $\Delta = t[k+1] - t[k]$). Therefore, $t[k] = \tau_0 + k\Delta$ for all $k \geq 1$. The state variables are discretized using a zero-order hold approach such that for all $k \in \{0, \ldots, T-1\}$:

$$\boldsymbol{u}(\tau) = \boldsymbol{u}[k], \quad \tau \in [t[k], t[k+1]), \tag{12}$$

$$\boldsymbol{x}(\tau) = \boldsymbol{x}[k], \quad \tau \in [t[k], t[k+1]), \tag{13}$$

$$\therefore \boldsymbol{p}(\tau) = \boldsymbol{p}[k], \, \boldsymbol{\theta}(\tau) = \boldsymbol{\theta}[k].$$

In order to write the spacecraft's dynamics equations (11) in a discrete-time form, this equation is first linearized around a point $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$, which need not be an equilibrium point, as follows:

$$\dot{\boldsymbol{x}}(\tau) = \boldsymbol{f}(\boldsymbol{x}^\star, \boldsymbol{u}^\star) + A(\boldsymbol{x}^\star, \boldsymbol{u}^\star, \tau) \cdot (\boldsymbol{x}(\tau) - \boldsymbol{x}^\star) + B(\boldsymbol{x}^\star, \boldsymbol{u}^\star, \tau) \cdot (\boldsymbol{u}(\tau) - \boldsymbol{u}^\star), \tag{14}$$

$$\text{where} \qquad A(\boldsymbol{x}^\star, \boldsymbol{u}^\star, \tau) = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))}{\partial \boldsymbol{x}(\tau)}\right|_{(\boldsymbol{x}^\star, \boldsymbol{u}^\star)}, \tag{15}$$

$$B(\boldsymbol{x}^\star, \boldsymbol{u}^\star, \tau) = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))}{\partial \boldsymbol{u}(\tau)}\right|_{(\boldsymbol{x}^\star, \boldsymbol{u}^\star)}. \tag{16}$$

Note that the matrices $A(\boldsymbol{x}^\star, \boldsymbol{u}^\star, \tau)$ and $B(\boldsymbol{x}^\star, \boldsymbol{u}^\star, \tau)$ in (15)–(16) are functions of the point $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ and time τ. Equation (14) is then discretized using a zero-order hold approach as fol-

lows:

$$\boldsymbol{x}[k+1] = F[k]\,\boldsymbol{x}[k] + G[k]\,\boldsymbol{u}[k] + H[k]\,, \tag{17}$$

$$\text{where } F[k] = e^{A(\boldsymbol{x}^\star,\boldsymbol{u}^\star,t_k)\Delta}\,, \tag{18}$$

$$G[k] = \int_{t[k]}^{t[k+1]} e^{A(\boldsymbol{x}^\star,\boldsymbol{u}^\star,\tau)\cdot(t[k+1]-\tau)} B\left(\boldsymbol{x}^\star,\boldsymbol{u}^\star,\tau\right) d\tau\,, \tag{19}$$

$$H[k] = \int_{t[k]}^{t[k+1]} e^{A(\boldsymbol{x}^\star,\boldsymbol{u}^\star,\tau)\cdot(t[k+1]-\tau)} \left(\boldsymbol{f}\left(\boldsymbol{x}^\star,\boldsymbol{u}^\star\right) - A\left(\boldsymbol{x}^\star,\boldsymbol{u}^\star,\tau\right)\boldsymbol{x}^\star - B\left(\boldsymbol{x}^\star,\boldsymbol{u}^\star,\tau\right)\boldsymbol{u}^\star\right) d\tau\,. \tag{20}$$

These are the transformed discrete-time dynamics equations used in **Problem 1**(3)–(10).

REFERENCES

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

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

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

[4] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm Assignment and Trajectory Optimization Using Variable-Swarm, Distributed Auction Assignment and Sequential Convex Programming," *Int. J. Robotics Research*, Vol. 35, 2016, pp. 1261–1285.

[5] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Trans. Robotics*, 2017. to appear.

[6] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "A Probabilistic Eulerian Approach for Motion Planning of a Large-Scale Swarm of Robots," *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Daejeon, South Korea, Oct. 2016.

[7] D. Morgan, G. P. Subramanian, S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic guidance of distributed systems using sequential convex programming," *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Chicago, IL, Sept. 2014, pp. 3850–3857.

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

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

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

[11] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "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.

[12] F. Baldini, S. Bandyopadhyay, R. Foust, S.-J. Chung, A. Rahmani, J.-P. d. l. Croix, A. Bacula, C. M. Chilan, and F. Y. Hadaegh, "Fast Motion Planning for Agile Space Systems with Multiple Obstacles," *AIAA/AAS Astrodynamics Specialist Conference*, Long Beach, CA, 2016.

[13] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *arXiv preprint arXiv:1505.00023*, 2015.

[14] D. Richert and J. Cortés, "Robust distributed linear programming," *IEEE Trans. Autom. Control*, Vol. 60, No. 10, 2015, pp. 2567–2582.

[15] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A Distributed Auction Algorithm for the Assignment Problem," *IEEE Conf. Decision Control*, Cancun, Mexico, Dec. 2008, pp. 1212–1217.