Autonomous In-Orbit Satellite Assembly from a Modular Heterogeneous Swarm

Rebecca C. Foust[a,b,*], E. Sorina Lupu[b], Yashwanth K. Nakka[b], Soon-Jo Chung[b,c], Fred Y. Hadaegh[c]

[a] *University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA*
[b] *California Institute of Technology, Pasadena, CA, 91125, USA*
[c] *Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA*

Abstract

This paper presents a decentralized, distributed guidance and control scheme to combine a heterogeneous swarm of component satellites into a large satellite structure. The component satellites for the heterogeneous swarm are chosen to promote flexibility in final shape inspired by crystal structures and Islamic tile art. After the ideal fundamental building blocks are selected, basic nanosatellite-class satellite designs are made to assist in simulations involving attitude control. The Swarm Orbital Construction Algorithm (SOCA) is a guidance and control algorithm to allow for the limited type heterogeneity and docking ability required for in-orbit assembly. The algorithm consists of two parts, a distributed auction which uses barrier functions to ensure the proper agent selection for each target, and a trajectory generation portion which leverages model predictive control and sequential convex programming to achieve optimal collision-free trajectories to the desired target point even with nonlinear system dynamics. The optimization constraints use a boundary layer to determine whether the collision avoidance or the docking constraints should be applied. The algorithm was tested in a simulated perturbed 6-DOF spacecraft dynamic environment for planar and out-of-plane final structures and on two robotic platforms, including a swarm of frictionless spacecraft simulation robots.

Keywords: self-assembly, optimal guidance and control, satellite, docking,

*Corresponding Author
Email addresses: foust3@illinois.edu (Rebecca C. Foust), eslupu@caltech.edu (E. Sorina Lupu), ynakka@caltech.edu (Yashwanth K. Nakka), sjchung@caltech.edu (Soon-Jo Chung), fred.y.hadaegh@jpl.nasa.gov (Fred Y. Hadaegh)

optimal assignment, swarm systems

Nomenclature

$\mathcal{A}_{\text{set}}^{j}$	=	Assembly set of agent j
$\mathcal{B}$	=	Barrier function for number of docks
$\mathcal{B}_{\text{ang}}$	=	Barrier function for angle of docks
C	=	Auction cost
$\mathcal{D}_j$	=	Dock set of agent j
δt	=	Time step [s]
$\mathbf{f}$	=	Discrete-time nonlinear dynamics
$\mathcal{F}$	=	Cost integrand function
g_i	=	Convex inequality constraints
J	=	Trajectory generation cost function
k	=	Discrete time step
$\mathcal{L}_{i,j}$	=	SCPn inequality constraint constant
n	=	Maximum number of docks ports on the current agent
N	=	Number of agents
$N_T(\mathbf{x}_f)$	=	Number of docks required at target position $\mathbf{x}_f$
$\mathcal{N}_j$	=	Set of neighbor agents
$\mathbf{p}^i$	=	Auction bid of agent i
$\mathbf{P}$	=	Ellipsoidal error profile of the nominal trajectory
$\mathbf{r}$	=	Position elements of the state [m]
R_{bl}	=	Boundary layer radius [m]
R_{col}	=	Collision avoidance radius [m]
R_{comm}	=	Communication radius [m]
R_{dock}	=	Dock radius [m]
$\mathbf{x}_{j,0}$	=	Initial state constraint
$\mathbf{x}_k$	=	State trajectory at the k-th time step
$\overline{\mathbf{x}}_{n,k}$	=	Nominal, nonlinear state trajectory at the k-th time step
$\mathcal{X}_f$	=	Set of terminal positions [m]
t	=	Time [s]
t_0	=	Initial time [s]
t_f	=	Final time [s]
T	=	Final discrete time step [s]
θ	=	Dock angle allowed by agent type [degrees]
$\Theta_T(\mathbf{x}_f)$	=	Dock angle required at terminal position [degrees]
$\mathbf{u}_k$	=	Control input trajectory at the k-th time step [m/s^2]
$\overline{\mathbf{u}}_k$	=	Nominal control input trajectory at the k-th time step [m/s^2]

$U_{\max}$ = Maximum control input [m/s^2]
$V_{\max}$ = Maximum speed [m/s]
w = SCPn iteration

1. Introduction

Design and construction of large space systems is often constrained by factors like launch vehicle fairing size, available payload mass, or ability to withstand launch loading. Satellites constructed in space would not experience these design constraints, allowing for lighter, more capable satellites. Start to finish construction in orbit is not yet possible, but improvements can still be made through recent advances in swarm spacecraft guidance and control [18, 21, 22] and autonomous rendezvous and docking [10]. By leveraging the above swarm guidance and control algorithms, a large space structure can be constructed from a swarm of component satellites. The advantages of such a mission are clear: increased reliability due to redundancy, increased flexibility, ability to reconfigure for future missions, and ability to self-repair [43]. Applications for these missions range from the small scale, where the components are microsatellites building a support structure for a distributed telescope or a solar sail, to the large scale, where components are habitat modules building a space colony. The mission concept is illustrated in Figure 1. The steps are as follows:

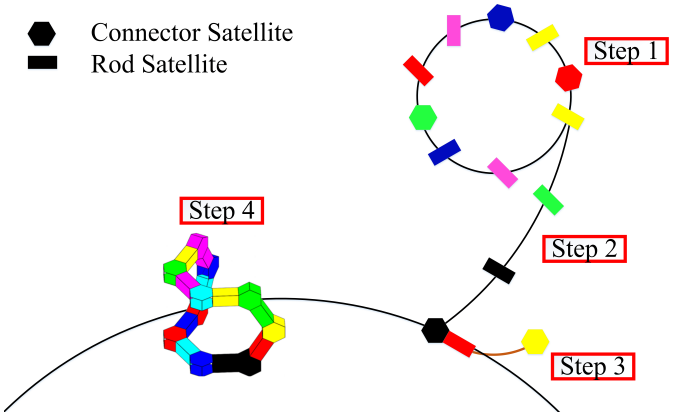


Figure 1: Outline of Mission Steps

Step 1 The components enter into loose formation to stay close to other components until they are used (e.g. see collision-free J_2 invariant passive relative orbits in [21]).

Step 2 The components determine their desired final position in the assembly and move to take the position using SOCA.

Step 3 Along the path to the final position, components assigned to neighboring positions dock and proceed combined.

Step 4 Finally, a complete structure is made once all components have reached their final destination.

1.1. Related Work

To reach the full potential of this swarm, it is imperative that the proposed guidance and control algorithm allows each agent in the swarm to act independently, without global knowledge of the swarm. Centralized algorithms are disadvantageous because they require all-to-all or all-to-one communication, which is difficult in large spacecraft swarms, either highly taxes the communication systems or introduces a single-point failure. This requirement means the algorithm must be decentralized, so each agent decides its own trajectory based on information from the neighboring agents with which it can communicate. Also, to increase the flexibility in potential final structures, it is beneficial to use multiple types of agents in the construction swarm, so the guidance and control algorithm must be able to handle heterogeneous agents. Finally, the proposed algorithm must control both position and attitude of the spacecraft since docking is required.

In the literature, many examples of decentralized swarm guidance schemes exist, but the swarms are typically homogeneous [35, 38, 39, 40, 44]. The heterogeneous swarm guidance schemes typically use centralized algorithms. A similar modular swarm construction mission was demonstrated using a homogeneous swarm of rectangular robots constructed in a brick pattern [31]. Though this demonstration involved a homogeneous swarm with a planar construction and centralized guidance and assignment, the assembly scheme docks along the way to the final location, similar to the present paper. Another team created a satellite assembly guidance and control scheme for a homogenous swarm to a predefined final formation using a glideslope algorithm to guide each satellite to a dock relative to other satellite, with collision avoidance and particular constraints on relative velocity at certain waypoints along the trajectory [27]. The approach is suboptimal; it uses linearized dynamics and neglects perturbations and relative attitude but succeeds in building the formation at a low fuel cost [27].

The field of robotic self-assembly has many interesting and innovative mechanisms. The robotic systems that are applied to space assembly are

typically multi-use robots with multiple end-effectors like the MoleCubes [45], which have a useful reconfiguration technique where the cubes rotate along a diagonal axis to switch the location of two faces. This actuation type could be very useful in the in-space construction scheme we have defined. Another interesting actuation type with space applications is used by the MIT M-Blocks. The M-Blocks are cubes that have magnetic edges and an internal flywheel which allows them to pop up and latch on to make various configurations [30]. Self-assembling robotics applied to space applications is limited. The Transformable Robotic Infrastructure-Generating Object Network (Trigon) system uses robotic self-assembly for in space construction to facilitate human planetary missions [19]. The Trigon system is multi-use and can build structures from rovers to habitats using a "kit-of-parts", a set of Trigon parts. Each Trigon part is essentially a face with actuators along the edges that can interact to self-assemble by moving parts along the structure [19].

In orbit, some methods propose a free-flying tether robot which can dock with components to combine them into an overall structure [9, 14, 33]. One study proposes a sensing and optimal control strategy for a swarm of homogeneous spacecraft with plume impingement constraints assuming negligible orbital perturbations [36]. Assembler agents (robots which put components together) are more popular, like the antenna construction in [32], which generates target position constraints from antenna design, but neglects orbital dynamics in favor of flexible body dynamics. Assembler agent style assembly can be seen in many proposed missions, detailed further in [29], along with additive manufacturing schemes.

PolyBots [42] self-assemble using two types of agents with hermaphroditic docking ports. The two agent types are similar to our design, a node and a segment. Though the system architecture is similar to our concept, PolyBots are mainly for surface operations and can be connected to form an arbitrary robot. The flexibility of the segment agent allows the PolyBot chains to be used for locomotion and manipulation.

This paper details a decentralized, distributed assignment and guidance scheme for a heterogeneous swarm, called the Swarm Orbital Construction Algorithm (SOCA) for six Degrees-of-Freedom (DOF) spacecraft dynamics. Hence, this paper also elucidates results of experimental validation using Caltech's Multi-Spacecraft Testbed for Autonomy Research (M-STAR) [25]. In prior work, the Swarm Assignment and Trajectory Optimization (SATO) algorithm was used to solve a target assignment and collision-free path planning problem by implementing a decentralized auction algorithm with a trajectory planner which implemented model predictive control using se-

quential convex programming (MPC-SCP) [22, 23]. The two algorithms are run sequentially over the course of the SATO algorithm so that the initial assignments and trajectories can be updated as agent connectivity changes or collision avoidance is needed. All agents are assumed to know of the set of target locations, and have a limited communication radius. SATO was designed for a homogeneous swarm targeting to a disconnected, free-flying formation and did not incorporate docking or assembly of any kind. While SATO performs admirably for this type of homogeneous swarm formation building, it must be altered for the proposed heterogeneous construction swarm. In addition to the heterogeneity logic, the collision avoidance logic in MPC-SCP must be carefully relaxed to allow docking agents to come within the collision avoidance radius.

The main contributions of this paper are the derivation and experimental validation of the in-orbit assembly guidance algorithm which makes it suitable for limited type heterogeneity in the swarm and allows for docking satellites while avoiding undesired collisions. The heterogeneity is handled in the target assignment phase, where a barrier function is added to the optimization cost to make improper assignments prohibitively expensive. The barrier functions do not encode which type of agent must be assigned to each spot, but rather what docking characteristics the agent must fit to function in that target location. This prevents agents from going to positions where they cannot satisfy the requirements. MPC-SCP is used to generate the collision-free trajectories, with a boundary layer added to relax collision constraints on agents targeting neighboring positions to allow the agents to dock before reaching the target.

This paper will go over the design of the Swarm Orbital Construction Algorithm (SOCA), first the heterogeneous auction algorithm in Section 2 and then the trajectory generation algorithm in Section 3. Next, simulation results from the improved SOCA algorithm are presented in Section 4. Finally, the experimental validation of the algorithm on both wheeled robots and spacecraft simulator robots is presented in Section 5, with the MPC-SCP trajectories followed by tracking controllers. The result of this paper a coherent and robust algorithm for in-orbit construction using a decentralized algorithm to guide and control a heterogeneous, docking swarm of satellites.

2. Heterogeneous Target Assignment

2.1. Heterogeneous Docking Components

When designing the construction swarm, a lot of thought was put into the geometry of the agents and the resulting final constructions possible using those geometries. The desire for flexibility in final configuration led to the examination of Islamic tile art and the geometry of crystals for inspiration. Islamic art like that shown in Figure 2 relies strongly on geometric shapes and symmetry, creating complicated and elegant periodic or aperiodic patterns [1, 2]. These patterns can be helpful in the investigation of large space structures for several reasons. Mainly, the vast array of complicated geometric patterns can be used as models for layouts of potential space structures because they are sufficiently complex to suit a variety of mission types and the basic geometric shapes can be constructed using standard satellite buses. The focus on symmetry is beneficial for space structures as well because symmetric structures are more likely to be controllable. By inspection of the rigid body Euler equations, it is clear that the attitude dynamics of asymmetric satellites are significantly more entwined than axisymmetric satellites, and in the proposed scheme it is highly possible that the center of gravity of an asymmetric assembly is outside the assembly, therefore the control points available may not be able to stabilize the assembly. When evaluating potential large space structures, controllability is a grave concern.

Islamic tile artists created quasi- and aperiodic tiling patterns centuries before western mathematicians like Roger Penrose formalized them [3]. Penrose tilings, or 5-way symmetric tilings do not have the translational symmetry of most tile patterns which makes them aperiodic [20, 28]. This means the tilings are not simple tessellations, but are complex and non-repeating patterns. Unfortunately these fascinating Penrose tile designs require non-convex geometries which are less desirable for satellite structural designs. Tessellation, which makes use of translational symmetry, is also helpful in increasing the packing efficiency of the rocket, components that tessellate will pack more densely into the rocket, enabling more agents to be brought to orbit in each launch. Crystals found in nature typically have a periodic geometric layout much like the above mentioned art. For example, the mineral beryl typically has a hexagonal prism shape in macro-scale. It also has a hexagonal void created at the nano-scale, created by the ring of silicon-oxygen bonds [8].

The chosen hexagonal prism connectors with rectangular prism rods can be combined to create a complex planar structure with a minimal degree

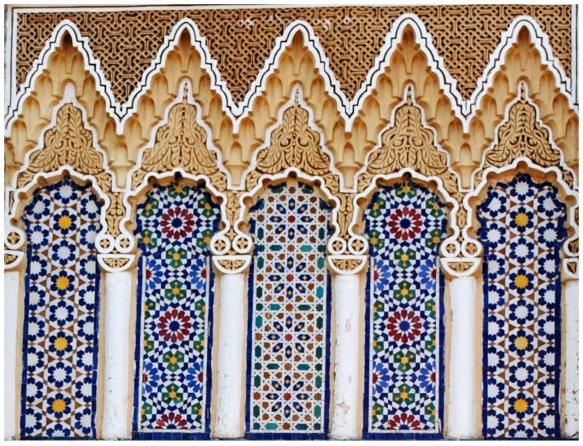


Figure 2: Geometric motivation for this work was found in art and nature, through Islamic tile art, like the Royal Palace Gates in Fes, Morocco

of heterogeneity. Hexagonal and rectangular agent types can be used to create the following shapes: parallelogram, equilateral triangle, hexagon, any combination thereof, as shown in Figure 3. Any one of these shapes can tessellate to cover the whole plane, but it is also possible to combine the shapes to cover the plane in more useful ways tailored to particular missions. These shapes can be combined using semiregular or uniform tiling in methods similar to that of Islamic tile art and the work of M.C. Escher, where fixed geometric shapes are tessellated and decorated to create art [7]. The patterns and shapes can all be made at any scale, by adding more rectangular rod agents between the hexagonal connector agents.

The concept of tessellation and tiling is much more complicated in three dimensions, with only three regular geometries capable of filling 3-space, the cube, tetrahedron and octahedron [41]. Instead of complete 3-space tessellation, several planes of different tilings can be connected by out of plane agents to create 3D configurations. When designing these configurations, it would also be beneficial to examine space grid structures, an architectural feature akin to complicated trusses where patterns of struts combine in three dimensions to act as a single unit [6]. The ability of this system to create three dimensional configurations relies heavily on the docking orientations allowed by the chosen docking port. In this paper we will assume the docking system allows docking at a set of relative angles, 0° and 90°. This makes the generation of 3D shapes possible, though limiting the out of plane dock angles limits the possible final shapes.

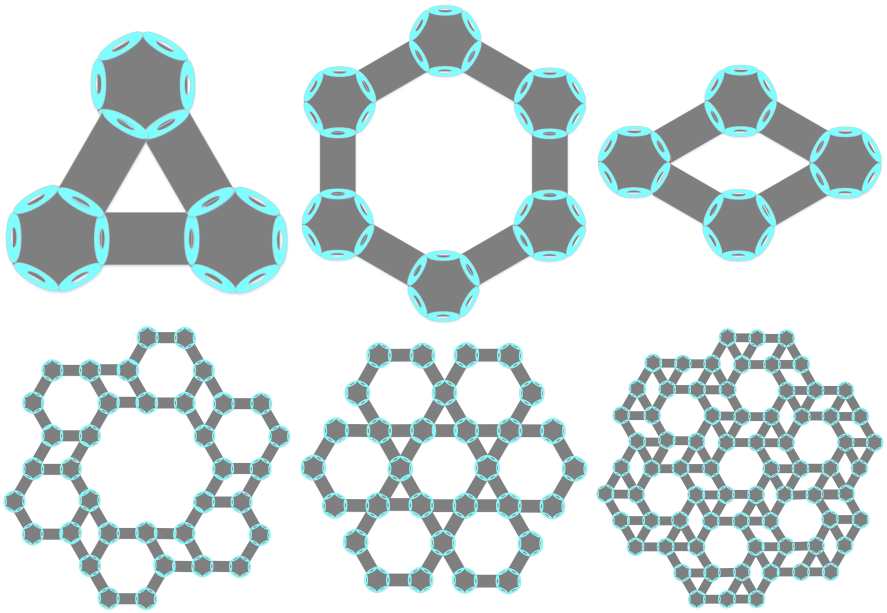


Figure 3: Examples of potential shapes and tiling patterns possible with the proposed mission, using only rectangular rod and hexagonal connector agents

2.2. Assignment with Conflict Resolution for Heterogeneous Agents

The target assignment algorithm, called Distributed Auction Algorithm for Docking (DAA-D), assumes each agent in the swarm knows the location of all the possible targets. The targets are provided by ground crew, along with the requisite information about docking constraints. Since the algorithm is decentralized and distributed, all bidding information is communicated only to agents within the communication radius. This communication radius limitation prevents the algorithm from reaching the centralized, optimal solution when the problem begins with the graph disconnected, but allows it to more accurately approximate the real systems. Each agent calculates its cost to each target using some cost function like the distance between the two points or the cost of a minimum-fuel optimal trajectory, shown in Problem 1. The agent then uses this cost to bid for type-appropriate target locations with the agents within its communication network as shown below in Problem 2. The auction is designed to allow all bids to propagate completely through the communication network. As the agents move to the targets, the communication graph becomes connected which ensures that over time the optimal assignment will be reached [23].

Problem 1 (Auction Cost for Docking).

$$C(\mathbf{x}_{j,0}, \mathbf{x}_{j,f}) = \min_{\mathbf{u}_j} \sum_{k=k_0}^{T-1} \|\mathbf{u}_j[k]\|_1 \Delta t \quad \text{subject to} \tag{1}$$

$$\mathbf{x}_j[k+1] = A_j[k]\mathbf{x}_j[k] + B_j[k]\mathbf{u}_j[k] + z_j[k], \quad k = k_0, \ldots, T-1, j = 1, \ldots, N \tag{2}$$

$$\|\mathbf{u}_j[k]\|_\infty \leq U_{\max} \quad k = k_0, \ldots, T-1, \quad j = 1, \ldots, N \tag{3}$$

$$\|H\mathbf{x}_j[k]\|_2 \leq V_{\max} \quad H = [0_{3\times 3} \quad I_{3\times 3}], \quad k = k_0, \ldots, T, j = 1, \ldots, N \tag{4}$$

$$\mathbf{x}_j[0] = \mathbf{x}_{j,0} \tag{5}$$

$$\mathbf{x}_j[T] = \mathcal{X}_f(j) \tag{6}$$

where $\mathbf{x}$ is the state, $\mathbf{u}$ is the control, $A(\bar{\mathbf{x}}_{n,k}) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_k}\right|_{(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k)}$, $B(\bar{\mathbf{u}}_k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}_k}\right|_{(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k)}$, $z(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k) = \mathbf{f}(\bar{\mathbf{x}}_{n,k}, \bar{\mathbf{u}}_k) - A(\bar{\mathbf{x}}_{n,k})\bar{\mathbf{x}}_{n,k} - B(\bar{\mathbf{u}}_k)\bar{\mathbf{u}}_k$, $\mathbf{f}$ is the nonlinear dynamics, and $\mathcal{X}_f$ is the set of terminal positions. The cost function used here and in the trajectory generation section was chosen to create spacecraft fuel optimal paths, though the vector norm chosen depends on the spacecraft thruster configuration [22]. DAA-D has to be designed to ensure that each agent type is assigned to an appropriate location. For the current agent definitions, the agent types have different numbers and locations of docking ports but the same radius. This means that potential target locations can be differentiated by the quantity and the angle of docks required at each location. Algorithmically, this involves changing the cost function used in the auction to make improper assignments prohibitively expensive. This is achieved through the use of barrier functions. The target information known by every agent must now indicate the location of the target and how many docks must be performed at that location.

A barrier function can be used to prevent agents from successfully bidding on targeting locations they cannot accommodate. An example barrier function is:

$$\mathcal{B}(n, N_T(\mathbf{x}_f)) = \begin{cases} -\log(a(n, N_T(\mathbf{x}_f)) & n \geq N_T(\mathbf{x}_f) \\ \text{Inf} & n < N_T(\mathbf{x}_f) \end{cases} \tag{7}$$

with $N_T(\mathbf{x}_f)$ is the number of docks required at a target, n is the maximum number of docks an agent can perform based on its type (6 for connectors, 2 for rods) and $a(n, N_T)$ is a function of the docking ports available and required, like the examples investigated below. The barrier function is chosen

to give $\mathcal{B}(n, N_T(\mathbf{x}_f))$ infinite value when the number of docks at a target exceeds the number of docks the agent can perform. It is also possible to use a sigmoid function to change the performance of the assignment by changing the function $a(n, N_T(\mathbf{x}_f))$.

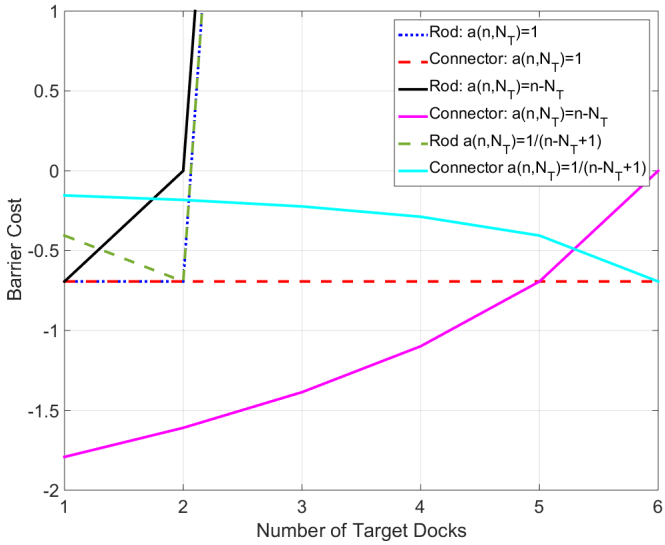


Figure 4: Possible barrier functions to use in assignment

Barrier functions made using three different $a(n, N_T(\mathbf{x}_f))$ are shown in Figure 4. Using $a(n, N_T(\mathbf{x}_f)) = 1$ gives a simple sorting of agents, where the cost for rod agents is prohibitively high at positions that require more docking ports than they have. If $a(n, N_T(\mathbf{x}_f)) = n - N_T(\mathbf{x}_f)$ is used, the barrier function makes the cost lower if fewer docking ports are used, where N_T is low. Changing $a(n, N_T(\mathbf{x}_f))$ to some decreasing positive function of $n - N_T(\mathbf{x}_f)$ like $1/(n - N_T(\mathbf{x}_f) + 1)$ makes the cost lower for positions where they can use the most docking ports, where N_T is the highest allowed. For all of these barrier functions, the cost of assigning to locations requiring more docking ports than the agent has are made prohibitively high. If no barrier function is used then the problem is the same as the standard auction from SATO [23], and improper assignments are allowed to occur. The chosen barrier function uses $a(n, N_T(\mathbf{x}_f)) = n - N_T(\mathbf{x}_f)$ to dissuade agents from using all of their docking ports. Since the barrier function only strongly affects completely unsuitable targets, the optimality of the auction is not affected. Of course, the function must be well chosen such that the

influence of the barrier on suitable targets is not large enough to negatively impact the optimality.

The barrier function used above would not be sufficient to properly assign all input configurations. A potential improper assignment is illustrated in Figure 5, where the number of docks required is possible for the rod agent, but the angle required is not possible. This is an example of an assignment with an underutilized connector, with one or two docks required. To be able to accurately handle these cases, it is necessary to augment the barrier function to include the angle between the docks so that all improper assignments are avoided. This requires that the angle of the docks are encoded in the desired final configuration that all agents have access to.

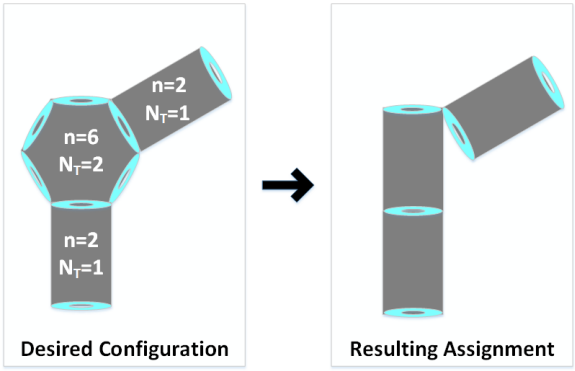


Figure 5: Bad Assignment: the desired configuration on the left has an underutilized connector, with only two docks required. This allows the initial barrier function to mis-assign a rod agent to the connector location, resulting in a disconnected structure.

The angle barrier function becomes more complicated in 3D configurations, and the choice of barrier function depends on the geometries of the chosen swarm. For planar configurations, the angle barrier function can act on the angle between docking agents, some multiple of 60° for connectors, or 180° for rods

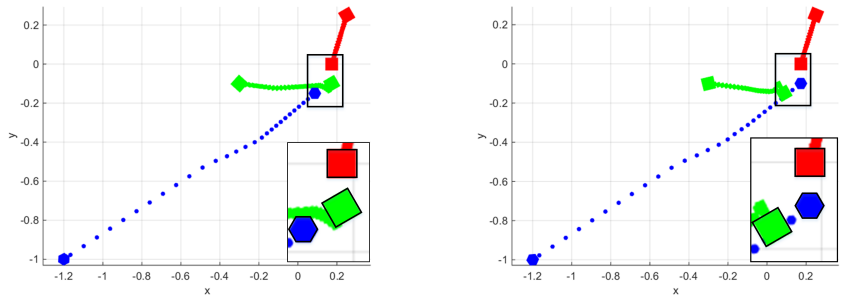
$$\mathcal{B}_{\text{ang}}(\theta, \Theta_T(\mathbf{x}_f)) = -\log(s(\theta, \Theta_T(\mathbf{x}_f)) + 1) \tag{8}$$

for some sigmoid $s(\theta, \Theta_T(\mathbf{x}_f))$ like

$$s(\theta, \Theta_T(\mathbf{x}_f)) = \begin{cases} \theta - \Theta_T(\mathbf{x}_f) & \theta \geq \Theta_T(\mathbf{x}_f) \\ -1 & \theta < \Theta_T(\mathbf{x}_f) \end{cases} \tag{9}$$

where θ is the dock angle allowed by the agent type, either 60° for connectors

or 180 ° for rods, and $\Theta(\mathbf{x}_f)$ is the angle required for docking at terminal position $\mathbf{x}_f$. As above, the function s can be replaced to more closely tailor the sorting to the mission at hand. In 3D configurations, the connector geometry we have chosen is the same as 2D, but we opt to utilize the all of the dock orientations allowed by the hermaphroditic docking port. The barrier function for the planar case can also be used for this configuration since the out of plane angle does not affect the choice between the rod and the connector. Adding this barrier function and providing the required dock angles successfully eliminates this problem, as shown in Figure 6. Without the angle barrier function, the agents can assign to improper target locations. The addition of the angle barrier function prevents rods from assigning to terminal positions which require a connector because of the dock angle.



(a) Without angle barrier function **(b) With angle barrier function**

Figure 6: 3-Agent example showing the efficacy of the angle barrier function in preventing improper assignments (not to scale)

To apply SOCA to a swarm with a higher degree of heterogeneity, the logic in the above barrier functions would need to be altered to properly sort between the characteristics of the new agent types.

Problem 2 (Updated Assignment with Docking Barrier Functions).

$$\min_{\mathbf{x}_{j,f},\ j=1\ldots N} \sum_{j=1}^{N} \left[C(\mathbf{x}_{j,0}, \mathbf{x}_{j,f}) + \mathcal{B}(n_j, N_T(\mathbf{x}_{j,f})) + \mathcal{B}_{\text{ang}}(\theta_j, \Theta_T(\mathbf{x}_{j,f})) \right] \quad (10)$$

subject to the following constraints:

$$\mathbf{x}_{j,f} \in \mathcal{X}_f, \quad \mathbf{x}_{j,f} \neq \mathbf{x}_{i,f}, \quad \forall j = 1 \ldots N, \quad \forall i \neq j$$

where $\mathcal{B}(n_j, N_T(\mathbf{x}_{j,f}))$ is the docking port barrier function and $\mathcal{B}_{\text{ang}}(\theta_j, \Theta_T(\mathbf{x}_{j,f}))$ is the docking angle barrier function presented above in Eqs. (7) and (8).

The algorithm to solve the optimal assignment problem is presented in Algorithm 1. See [23] for an optimality proof of this auction algorithm. The ϵ added to the bids in Line 21 prevents any two agents from having exactly the same bid, preventing gridlock.

Algorithm 1 Distributed Auction Algorithm for Docking (DAA-D)

1: $\mathcal{X}_f$ = terminal positions in desired shape
2: $\mathbf{c}^i(s)$ = cost of agent i choosing target s
3: $\mathcal{B}^i(s)$ = docking barrier function for agent i choosing target s
4: $\mathcal{B}^i_{\text{ang}}(s)$ = angle barrier function for agent i choosing target s
5: m^i = # of targets available for agent i to bid on
6: $\mathbf{p}^i = \mathbf{0}_{1\times m^i}$
7: $\mathbf{p}^i_{\text{old}} = -\mathbf{1}_{1\times m^i}$
8: $j^i = 1$
9: $count^i = 0$
10: **for all** i (*run in parallel*) **do**
11: **while** $count^i < 2D_{\text{net}}$ **do**
12: **if** $|\mathbf{p}^i(j^i)| > \mathbf{p}^i_{\text{old}}(j^i)$ (*i is outbid*) **then**
13: $m^i = \max\left(m^i, |\{s|\mathbf{p}^i(s) \neq 0\}|\right)$
14: **if** $|\{s|\mathbf{p}^i(s) > 0\}| = m^i$ **then**
15: $m^i = |\{s|\mathbf{p}^i(s) > 0\}| + 1$
16: $\mathbf{p}^i(1:m^i) = -\left(|\mathbf{p}^i(1:m^i)| + \epsilon\right)$
17: **end if**
18: $v^i = \min_{s=1\ldots m^i}\left(\mathbf{c}^i(s) + \mathcal{B}^i(s) + \mathcal{B}^i_{\text{ang}}(s) + |\mathbf{p}^i(s)|\right)$
19: $j^i = \arg\min_{s=1\ldots m^i}\left(\mathbf{c}^i(s) + \mathcal{B}^i(s) + \mathcal{B}^i_{\text{ang}}(s) + |\mathbf{p}^i(s)|\right)$
20: $w^i = \min_{s=1\ldots m^i, s\neq j^i}\left(\mathbf{c}^i(s) + \mathcal{B}^i(s) + \mathcal{B}^i_{\text{ang}}(s) + |\mathbf{p}^i(s)|\right)$
21: $\gamma^i = w^i - v^i + \epsilon$
22: $\mathbf{p}^i(j^i) = |\mathbf{p}^i(j^i)| + \gamma^i$
23: $count^i = 0$
24: **else if** $\mathbf{p}^i \neq \mathbf{p}^i_{\text{old}}$ (*another agent is outbid*) **then**
25: $m^i = \max\left(m^i, |\{s|\mathbf{p}^i(s) \neq 0\}|\right)$
26: $count^i = 0$
27: **else**
28: $count^i = count^i + 1$
29: **end if**
30: $\mathbf{p}^i_{\text{old}} = \mathbf{p}^i$
31: Communicate $\mathbf{p}^i$ to all agents in $\mathcal{N}_{[i]}$

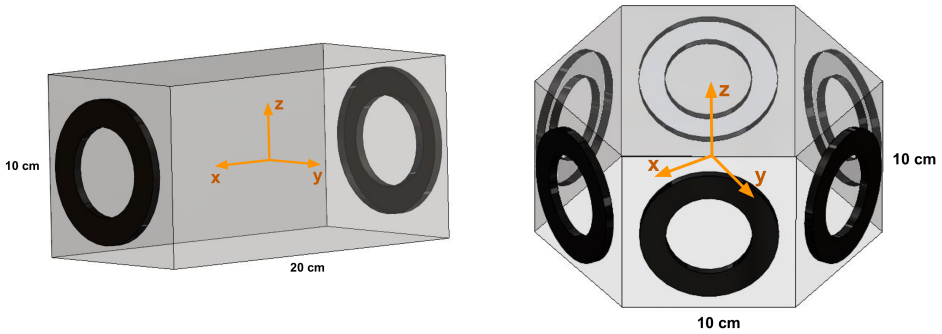
32: **for** $s = 1 \dots m^i$ **do**
33: $\mathbf{p}^i(s) = \min_{q \in \arg\max_{q \in \mathcal{N}_{[i]}} (|\mathbf{p}^q(s)|)} (\mathbf{p}^q(s))$
34: **end for**
35: **end while**
36: *Optional:* $m^i = |\{j | \mathbf{p}^i(j) \neq 0\}|$
37: *Optional:* Go back to line 6 and rerun with new m^i
38: $\mathbf{x}_{i,f} = \mathcal{X}_f(j^i)$
39: **end for**

In this algorithm, each agent calculates its bid, which is the augmented auction cost in Eq. (10), then communicates it with its neighbors within the connected communication graph. Then, the agents with the lowest bids claim their targets, and outbid agents generate a new bid by adding some increment to avoid gridlock in Line 16. This continues until all bids have the chance to fully propagate through the neighbors.

2.3. Shape Parameters

The rod agent is a rectangular prism with two docking ports located on the ends. The connector agent is a regular hexagonal prism with six docking ports along the sides. In order to execute this algorithm while incorporating attitude control, assumptions about mass properties and systems engineering configurations for the agent types must be made. The mass and volume advantages of the swarm will be most effective if the agents are kept small, in the nanosatellite class. The rod agent can use a standard 2U CubeSat bus, with a mass of 2.6 kg and principal inertia parameters $(44.5, 111.3, 111.3)\text{kgcm}^2$. Each side of the regular hexagonal prism must be the same as the face of a CubeSat to allow docking. The connector agent has a mass of 4 kg and principal inertia parameters $(116.7, 116.7, 166.7)\text{kgcm}^2$. The body frame definitions and the docking port locations for the two agent types are shown in Figure 7.

The docking ports combine features of existing magnetic docking ports [24, 26]. The docking ports are designed to be hermaphroditic and electromagnet-based with a rigidizing component so the electromagnets can be turned off. For more detail see [13].



(a) Rod Agent

(b) Connector Agent

Figure 7: Definitions of the Rod and Connector agent types, with docking ports shown in black

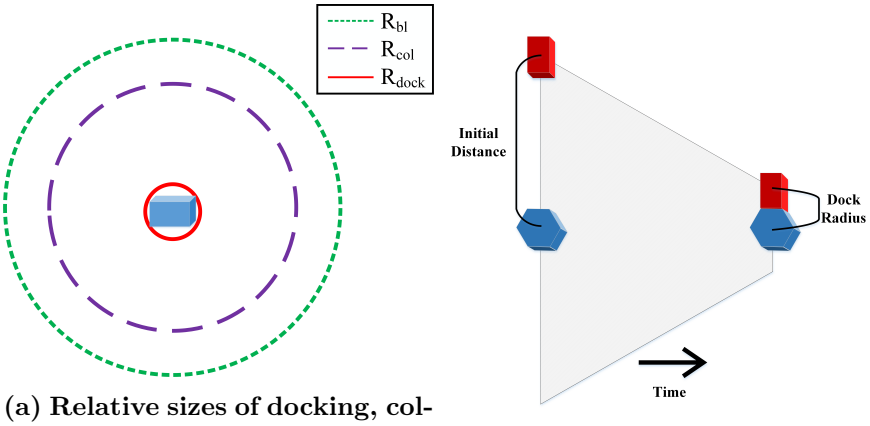
3. SOCA Problem Statements and Algorithms

3.1. Optimal SOCA Trajectory Generation

The next modification was to the trajectory generation algorithm. MPC-SCP is used to create the optimal, collision-free trajectories to the targets selected by DAA-D. Initially, a nominal trajectory is generated without considering collision avoidance. Then each agent solves the MPC-SCP problem with collision avoidance on a limited time horizon, with the knowledge of the nominal trajectories of the agents within its communication radius. The collision avoidance constraint is not convex, but by approximating the other agents' collision avoidance spheres as hyperplanes orthogonal to the surface, convexity can be obtained.

Because the agents need to dock for construction, the collision avoidance constraint must be suspended for agents that are attempting to dock. This is achieved using a boundary layer around the collision avoidance radius. The relative sizes of the radii are illustrated in Figure 8a. When an agent approaches within the boundary layer, the main agent checks to see if the approaching agent is targeted for a location neighboring the main agent's target. If it is, the agent follows the docking cone constraint. Otherwise, the collision avoidance constraint holds.

In order to allow the agents some flexibility in docking, the docking constraint requires the agent to maintain a shrinking distance to the agent to be docked. This means that over time the main agent will stay within



(a) Relative sizes of docking, collision avoidance and boundary layer radii

(b) Docking Cone

Figure 8: Description of docking radii and cone. In (a), the relative sizes of the docking, collision avoidance and boundary layer radii. Once another agent comes within the boundary layer radius, the main agent decides whether to dock or avoid it. If the agents are docking, the docking cone constraint shown in (b) goes into effect. The initial distance is the distance when the docking constraint is first applied, when the red agent comes within the boundary layer of the blue agent or vice versa.

a cone defined by the other agent. The cone radius begins as the initial separation and ends as the agent docking radius, as seen in Figure 8b. This docking cone forces the agents to come together by the final time. This way, the agents are allowed to dock before they reach the target if it is beneficial to their trajectories, or they can wait until the final position to dock.

3.1.1. Avoiding Collision with Docked Agents

This implementation requires that agents avoid docking with an approaching agent that is not in its docking set, even if that agent is already docked with an agent in its docking set. This causes agents to avoid docking until the final time step to avoid collision with the other agent. Imagine an agent approaching a docked pair, one of which it will need to dock with. Due to the collision avoidance radius of the other agent in the docked pair, the dock would be avoided until the final time step. This is fixed by simply adjusting the collision avoidance radius of agents in this position. Each agent, j, knows the set of agents it intends to dock with based on the assigned terminal positions. As agent j docks with another agent i, agent i is added to the assembly set of agent j, $\mathcal{A}^j_{\text{set}}$ and vice versa. This set is communicated

to other agents in j's dock set, which then reduce the collision avoidance radius for those agents to double the docking radius. This does not affect the convexity of the collision avoidance constraint because the radius changes between time steps. To illustrate this problem and the solution, Figure 9 shows the agents smoothly travelling to an equilateral triangle final configuration. The blue circles in the figure represent the collision avoidance radius of each agent. In this final configuration, the collision avoidance radii overlap. Without the adjusted collision avoidance for agents in the assembly set, the yellow, magenta, and cyan agents would avoid each other causing irregularities in the trajectories. The magenta agent can be seen avoiding the cyan agent by the adjusted smaller collision avoidance radius as it passes near the end of the trajectory.

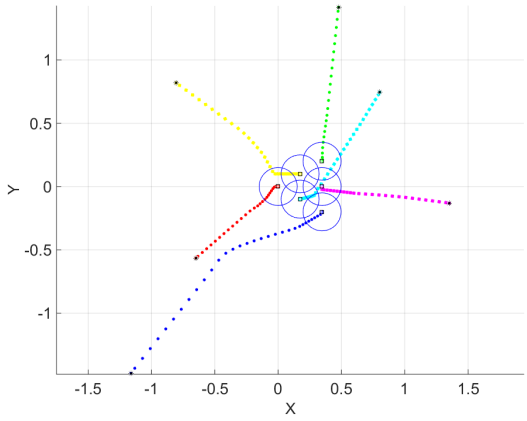


Figure 9: With the use of the assembly set condition, six agents can easily form a planar equilateral triangle without the collision avoidance issue. This example is performed using planar position and attitude double integrator dynamics for clarity.

3.1.2. Second-Order Cone Programming

The collision avoidance constraint introduces a lot of error due to the estimation of the other agents' trajectories. One way to make the algorithm robust to that error is to use second order cone programming to frame the constraint [5]. This method uses ellipsoidal error profiles around the estimated trajectory.

Lemma 1 *Let the nominal trajectories* $\bar{\mathbf{x}}$ *of each agent* i *approaching agent* j *be distorted by an ellipsoidal error* P*, which is the same for all agents*

and defined as the square root of the measurement covariance. Under these conditions the collision avoidance constraint becomes:

$$\begin{aligned}-(\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k])^T\mathbf{r}_j[k] \leq &- R_{ij}(\|\tilde{\mathbf{r}}_j[k]-\tilde{\mathbf{r}}_i[k]\|_2+2\|P\|_2)+\|P^T\tilde{\mathbf{r}}_i[k]\|_2 \\ &- \|\tilde{\mathbf{r}}_j[k]^TP\|_2+(\tilde{\mathbf{r}}_i[k]-\tilde{\mathbf{r}}_j[k])^T\tilde{\mathbf{r}}_i[k]\end{aligned} \tag{11}$$

where R_{ij} is R_{col} or $2R_{\text{dock}}$ if agent i is in the assembly set of j.

Proof. The collision avoidance constraint is defined as:

$$(\bar{\mathbf{x}}_j[k]-\bar{\mathbf{x}}_i[k])^TG^TG(\mathbf{x}_j[k]-\bar{\mathbf{x}}_i[k]) \geq R_{ij}\|G(\bar{\mathbf{x}}_j[k]-\bar{\mathbf{x}}_i[k])\|_2 \tag{12}$$

for each agent j with neighbor agents i, where the nominal trajectories are denoted with bars and G is used to select the position elements of the state. Since all but $\mathbf{x}_j[k]$ is constant and we can define $\mathbf{r}[k]=G\mathbf{x}[k]$, this can be expressed as a linear inequality, $\mathbf{a}_j\mathbf{r}_j \leq \mathbf{b}_j$, with the constants given as:

$$\mathbf{a}_j=(\bar{\mathbf{r}}_i[k]-\bar{\mathbf{r}}_j[k])^T \tag{13}$$

$$\mathbf{b}_j=-R_{ij}\|\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k]\|_2-(\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k])^T\bar{\mathbf{r}}_i[k] \tag{14}$$

We then introduce error ellipsoids into this linear inequality to increase robustness, with the ellipsoids defined using:

$$\sup\{\mathbf{a}_j\mathbf{x}_j|\mathbf{a}_j\in\varepsilon_j\} \leq \inf \mathbf{b}_j \tag{15}$$

This implies the Second Order Cone Constraint [5]:

$$\bar{\mathbf{a}}_jx_j+\|P_j\mathbf{x}_j\|_2 \leq \inf\{b_j\} \tag{16}$$

Then for any error direction $\mathbf{q}$:

$$\mathbf{a}_j=\{\bar{\mathbf{a}}_j+P_j\mathbf{q}|\|\mathbf{q}\|_2\leq 1\} \tag{17}$$

$$\mathbf{a}_j-\bar{\mathbf{a}}_j=P_j\mathbf{q} \tag{18}$$

$$(\mathbf{a}_j-\bar{\mathbf{a}}_j)^TP_j^{-T}P_j^{-1}(\mathbf{a}_j-\bar{\mathbf{a}}_j)=\mathbf{q}^T\mathbf{q}\leq 1 \tag{19}$$

Assume $\|\mathbf{q}\|\leq 1$, $\bar{\mathbf{r}}_j[k]=\tilde{\mathbf{r}}_j[k]+P_j\mathbf{q}_j$, $\bar{\mathbf{r}}_i[k]=\tilde{\mathbf{r}}_i[k]+P_i\mathbf{q}_i$, where $\tilde{\mathbf{r}}$ denotes the actual nominal trajectory. Using these assumptions and plugging Eq.

(13) into the left hand side of Eq. (15), we get:

$$\begin{aligned}\sup_{\|q_{j,i}\|_2\leq 1}(\mathbf{a}_j\mathbf{r}_j[k]) &= (\bar{\mathbf{r}}_i[k]-\bar{\mathbf{r}}_j[k])^T\mathbf{r}_j[k] + \sup_{\|q_{1,2}\|_2\leq 1}(\mathbf{q}_j^T P_j^T\mathbf{r}_j[k] - \mathbf{q}_i^T P_i^T\mathbf{r}_j[k]) \\ &= (\bar{\mathbf{r}}_i[k]-\bar{\mathbf{r}}_j[k])^T\mathbf{r}_j[k] + 2\|P_j^T\mathbf{r}_j[k]\|_2\end{aligned} \tag{20}$$

Substituting the assumed error ellipse constraints and assuming equal error profiles for both agents

$$\begin{aligned}\mathbf{b}_j &= -R_{ij}\|\tilde{\mathbf{r}}_j[k]-\tilde{\mathbf{r}}_i[k]+P_j\mathbf{q}_j-P_i\mathbf{q}_i\|_2 \\ &\quad -(\tilde{\mathbf{r}}_j[k]+P_j\mathbf{q}_j-\tilde{\mathbf{r}}_i[k]-P_i\mathbf{q}_i)^T(\tilde{\mathbf{r}}_i[k]+P_i\mathbf{q}_i) \\ &\leq -R_{ij}(\|\tilde{\mathbf{r}}_j[k]-\tilde{\mathbf{r}}_i[k]\|_2+2\|P\|_2)+\|P^T\tilde{\mathbf{r}}_i[k]\|_2-\|\tilde{\mathbf{r}}_j[k]^TP\|_2 \\ &\quad +(\tilde{\mathbf{r}}_i[k]-\tilde{\mathbf{r}}_j[k])^T\tilde{\mathbf{r}}_i[k]\end{aligned} \tag{21}$$

Through manipulation, we arrive at the new collision avoidance constraint:

$$\begin{aligned}-(\bar{\mathbf{r}}_j[k]-\bar{\mathbf{r}}_i[k])^T\mathbf{r}_j[k] \leq &- R_{ij}(\|\tilde{\mathbf{r}}_j[k]-\tilde{\mathbf{r}}_i[k]\|_2+2\|P\|_2)+\|P^T\tilde{\mathbf{r}}_i[k]\|_2 \\ &- \|\tilde{\mathbf{r}}_j[k]^TP\|_2+(\tilde{\mathbf{r}}_i[k]-\tilde{\mathbf{r}}_j[k])^T\tilde{\mathbf{r}}_i[k]\end{aligned} \tag{22}$$

□

The ellipsoidal error profile P allows the error in the nominal trajectory to be fitted to the on-board sensors more realistically. This is particularly important with the swarm configuration since small satellite sensors are less capable. Using the same profile for all agents is appropriate since most of the error would come from sensing, which is the same for all agents.

3.2. MPC-SCP Problem Statement

The updated SOCA trajectory generation problems are:

Problem 3 (Updated Trajectory Generation).

$$\min_{\mathbf{u}_j}\sum_{k=k_0}^{T-1}\|\mathbf{u}_j[k]\|_1\Delta t \tag{23}$$

subject to the following constraints:

(a) the dynamics, state, and control constraints

$$\mathbf{x}_j[k+1] = A_j[k]\mathbf{x}_j[k]+B_j[k]\mathbf{u}_j[k]+z_j[k], \quad k = k_0,\dots,T-1, \quad j = 1,\dots,N \tag{24}$$

$$\|\mathbf{u}_j[k] - \bar{\mathbf{u}}_j[k]\| \leq (\beta)^{w-1}\mathcal{T}_0, \quad \forall k = k_0,\dots,T-1 \tag{25}$$

$$\|H\mathbf{x}_j[k]\|_2 \leq V_{\max} \quad H = [0_{3\times 3} \quad I_{3\times 3}], \quad k = k_0,\dots,T, \quad j = 1,\dots,N \tag{26}$$

(b) the initial and terminal conditions obtained from Problem 2

$$\mathbf{x}_j[0] = \mathbf{x}_{j,0}, \quad \mathbf{x}_j[T] = \mathbf{x}_{j,f}, \quad j = 1,\dots,N \tag{27}$$

(c) the new collision avoidance constraint

$$\begin{aligned}-(\bar{\mathbf{r}}_j[k] - \bar{\mathbf{r}}_i[k])^T\mathbf{r}_j[k] \leq &- R_{ij}(\|\tilde{\mathbf{r}}_j[k] - \tilde{\mathbf{r}}_i[k]\|_2 + 2\|P\|_2) + \|P^T\tilde{\mathbf{r}}_i[k]\|_2 \\ &- \|\tilde{\mathbf{r}}_j[k]^T P\|_2 + (\tilde{\mathbf{r}}_i[k] - \tilde{\mathbf{r}}_j[k])^T\tilde{\mathbf{r}}_i[k]\end{aligned} \tag{28}$$

$$G = [I_{3\times 3} \quad 0_{3\times 3}], \quad k = k_{bl},\dots,\min\{k_0+T_H, T\}, \quad i \in \mathcal{N}_{[j]} \cap \mathcal{P}_j \setminus \mathcal{D}_j$$

where $\mathcal{N}_{[j]} = \{i|\ \|\mathbf{x}_j[k_0] - \mathbf{x}_i[k_0]\|_2 \leq R_{\text{comm}}\}$, R_{comm} is the communication radius of each agent, $\mathcal{D}_j$ is the set of agents that are assigned to dock with agent j and $\mathcal{P}_j$ is the set of agents that have a higher priority than j, P is the ellipsoidal error profile of the nominal trajectory, and R_{ij} is R_{col} or $2R_{\text{dock}}$ if agent i is in the assembly set of j.

(d) the docking condition
if $\|G(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k]\|_2 \leq R_{\text{bl}}$:

$$\|G(\mathbf{x}_j[k] - \bar{\mathbf{x}}_i[k])\|_2 \leq R_{\text{cone}}(k), \tag{29}$$

$$k_{bl} = \arg\min_{k_0\leq k\leq k_0+T_H}\{\|G(\bar{\mathbf{x}}_j[k] - \bar{\mathbf{x}}_i[k]\|_2 - R_{\text{bl}}\} \tag{30}$$

$$R_{\text{cone}}(k) = R_{\text{dock}} + \frac{\|G(\mathbf{x}_j[k_{bl}] - \bar{\mathbf{x}}_i[k_{bl}])\|_2 - R_{\text{dock}}}{T - k_{bl}}(T-k),$$

$$k = k_{bl},\dots,\min\{k_0+T_H, T\},\ i \in \mathcal{N}_{[j]} \cap \mathcal{P}_j \cap \mathcal{D}_j$$

After the problem above is solved using MPC-SCP, the nonlinear correction step is applied by numerically integrating the resultant optimal control trajectories $\mathbf{u_j}[\mathbf{k}]\forall j, \forall k$:

$$\begin{aligned}\overline{\mathbf{x}}_{n,j}[k+1] &= \mathbf{f}_j[k](\overline{\mathbf{x}}_{n,j}[k], \mathbf{u}_j[k]), \quad k = k_0,\dots,T-1, \\ \overline{\mathbf{x}}_{n,j}[k_0] &= \mathbf{x}_j[k_0] = \mathbf{x}_0\end{aligned} \tag{31}$$

where $\mathbf{f}_j[k]$ is the Runge-Kutta integration of the state with the given control.

3.3. MPC-SCP with Nonlinear Dynamics Correction

Though MPC-SCP trajectory generation converged on a solution, the solution did not necessarily follow the actual spacecraft dynamics. This was due to the linearization done in the optimization loop. The losses due to linearization accumulated over the trajectory and could become substantial over long duration simulations. The nonlinear correction step was added to the MPC-SCP portion of SOCA to reduce these errors.

For a discretized nonconvex problem with nonlinear dynamics and convex or convexified constraints like spacecraft guidance with collision avoidance, the solution can be approximated using sequential convex programming. This is done by linearizing the dynamics and generating a solution iteratively using the previous solution as a nominal trajectory until the optimization converges on a solution, like in MPC-SCP. For SCP with a nonlinear correction, the nominal trajectory for the next SCP iteration is taken as the numerically integrated nonlinear dynamics using the initial conditions and the current (w-th) SCP-generated control trajectory:

$$\begin{aligned} \mathbf{x}^w_{n,j}[k+1] &= \mathbf{f}_j[k](\mathbf{x}^w_{n,j}[k], \mathbf{u}^w_j[k]), \quad k = k_0, \ldots, T-1, \\ \text{and} \quad \mathbf{x}^w_{n,j}[k_0] &= \mathbf{x}^w_j[k_0] = \mathbf{x}_0 \end{aligned} \tag{32}$$

where $f_j[k]$ is the nonlinear dynamics, w is the SCP iteration, and the subscript n indicates a nominal, nonlinear trajectory. This SCP optimization process along with nonlinear dynamic correction step Eq. (32) is repeated until the sequence of trajectories converges. To ensure convergence and optimality, the problem's inequality constraints must be modified so that the corrected solution will be provably feasible to the nonconvex problem, generalized to Problem 4.

Problem 4 (Non-Convex Program (NCP) with Nonlinear Dynamics).

$$\min_{\mathbf{u}[k_0:T-1]} \quad \sum_{k=k_0}^{T-1} \mathcal{F}_u(\mathbf{u}_j[k]) v_j[k] \quad \text{subject to} \tag{33}$$

$$\mathbf{x}_j[k+1] - \mathbf{f}(\mathbf{x}_j[k], \mathbf{u}_j[k]) = \mathbf{0}, \quad \mathbf{x}_j[k_0] = \mathbf{x}_0, \quad k = k_0, \ldots, T-1 \tag{34}$$

$$g_i(\mathbf{x}_j[k], \mathbf{u}_j[k]) \leq 0, \quad i = 1, \ldots, p, \quad k = k_0, \ldots, T \tag{35}$$

where $\mathbf{f}$ represents the discretized nonlinear dynamics, g_i are the p convex inequality constraints on the problem, and $v_j[k]$ denotes the quadrature

weight of numerical integration (e.g., $v_j[k] = \Delta t = t_j[k+1] - t_j[k]$ for the Euler method).

The nonlinear-corrected version of SCP (Problem 3) will be referred to as SCPn, with the inequality constraints represented as g_i. Proofs of convergence to the KKT point of the nonconvex problem can be found in [11, 12]. For now, we show that the optimal solutions of SCPn will result in a cost that decreases. A more thorough, updated handling of the theory behind this method can be found in [11].

Problem 5 ((w)-th Sequential Convex Program: SCPn$^{(w)}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$).
Given $\bar{\mathbf{x}}_n[k_0:T] = \mathbf{x}_n^{(w-1)}[k_0:T]$, $\bar{\mathbf{u}}[k_0:T-1] = \mathbf{u}^{(w-1)}[k_0:T-1]$:

$$\underset{\mathbf{u}[k_0:T-1]}{\text{minimize}} \quad \sum_{k=k_0}^{T-1} \mathcal{F}_u(\mathbf{u}[k])v[k] \quad \text{subject to} \tag{36}$$

$$\mathbf{x}[k+1] - A(\bar{\mathbf{x}}_n[k])\mathbf{x}[k] - B(\bar{\mathbf{u}}[k])\mathbf{u}[k] - z(\bar{\mathbf{x}}_n[k], \bar{\mathbf{u}}[k]) = \mathbf{0}, \mathbf{x}_{k_0} = \mathbf{x}_0 \tag{37}$$

$$g_i(\mathbf{x}[k], \mathbf{u}[k]) + \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j}\|\mathbf{u}_j - \bar{\mathbf{u}}_j\| \leq 0, \quad i = 1, \ldots, p \tag{38}$$

$$\|\mathbf{u}[k] - \bar{\mathbf{u}}[k]\| \leq (\beta)^{w-1}\mathcal{T}_0, \tag{39}$$

The Lipschitz and convex functions $g_i(\mathbf{x}[k], \mathbf{u}[k])$, $i = 1, \ldots p$ are from Eq. (35), and a positive constant $\mathcal{L}_{i,j}$ is defined in Eq. (41) in Theorem 2 below.

Theorem 2 (Decreasing Cost over Optimal SCPn Sequence). *If there exists a unique feasible solution* $(\mathbf{x}_n^{(w)}[k_0:T], \mathbf{u}^{(w)}[k_0:T-1])$ *to the original nonconvex problem (Problem 4) for some* w*, then*

$$J(\mathbf{u}^{(w+1)}[k_0:T-1]) \leq J(\mathbf{u}^{(w)}[k_0:T-1]) \tag{40}$$

where $J(\mathbf{u})$ *is the cost function Eq. (23) of Problems 3 and 4. under the following condition for each* $g_i(\mathbf{x}[k], \mathbf{u}[k])$ *in Problem 3*

$$\mathcal{L}_{i,j} = 2\|\overline{B}[k]\|(\|\overline{A}[k]\|)^{k-j-1} \sup_{(\mathbf{x}[k],\mathbf{u}[k])\in\mathcal{D}} \left\| \frac{\partial g_i(\mathbf{x}[k], \mathbf{u}[k])}{\partial \mathbf{x}[k]} \right\| \tag{41}$$

where $\mathcal{D} := \{(\mathbf{x} \in \mathbb{R}^n, u \in \mathbb{R}^m) \,|\, \text{constraints of Problem 4}\}$ *is the convex domain defined by the feasible set of Problem 4.*

The nominal trajectory $(\mathbf{x}_n^{(w+1)}[k_0:T], \mathbf{u}^{(w+1)}[k_0:T-1])$ *obtained from SCPn is also a feasible solution to the nonconvex problem, meaning that Eq. (40) holds for all subsequent* w*.*

Proof. See Appendix. □

The $\mathcal{L}_{i,j}$ term added to the inequality constraints restricts the feasible set and thus can make finding solutions more difficult. This is particularly true in a system with highly nonlinear dynamics as the bound on the inequality constraint is tightened to ensure the corrected solution remains feasible. In a multi-agent context with collision avoidance as an inequality constraint, this bound will result in more conservative trajectories. If the agents are sufficiently dense, this may cause problems in execution. To ameliorate this issue, we point readers to the Modified SCPn algorithm presented in [11, 12], which provides the same theoretical guarantees but with less restriction of the feasible set and faster execution.

3.4. Optimal Trajectory Generation Algorithm

The algorithm to solve Problem 3 for the optimal trajectories allowing both docking and collision avoidance is shown below in Algorithm 2. Note that the nonlinear correction step 11 occurs after each iteration of MPC-SCP.

Algorithm 2 Guidance and Control using Sequential Convex Programming

1: $\mathcal{K} := \{1, \ldots, N\}$ Set of all agents
2: $w := 1$
3: $\bar{\mathbf{x}}_j[k] := \mathbf{0}_{6\times 1},\ \forall j, k$
4: $\mathbf{x}_{j,0}[k] :=$ the solution to Problem 3 (Trajectory Generation) with $\mathcal{P}_j = \emptyset,\ \forall j, k$
5: $\bar{\mathbf{x}}_j[k] := \mathbf{x}_j^0[k],\ \forall j, k$
6: Communicate $\bar{\mathbf{x}}_j[k]$ to all neighboring agents ($i \in \mathcal{N}_{[j]}$)
7: **while** $\mathcal{K} \neq \emptyset$ **do**
8: **for all** $j \in \mathcal{K}$ (*run in parallel*) **do**
9: $\mathbf{x}_{j,w}^{nom}[k] :=$the solution to Problem 3 (Trajectory Generation), $\forall k$
10: **end for**
11: $\mathbf{x}_{j,w}[k] := \mathbf{f}_j[k](\mathbf{x}_{j,w}^{nom}[k], \mathbf{u}_w^{nom}[k])\quad \forall k$ (Nonlinear Correction)
12: **for all** j (*run in parallel*) **do**
13: $\bar{\mathbf{x}}_j[k] := \mathbf{x}_{j,w}[k],\ \forall k$
14: Communicate $\bar{\mathbf{x}}_j[k]$ to all neighboring agents ($i \in \mathcal{N}_{[j]}$)
15: **if** $\|\mathbf{x}_{j,w}[k] - \mathbf{x}_{j,w-1}[k]\|_\infty < \epsilon_{\text{SCP}}\ \forall k$ and $\|G(\mathbf{x}_{j,w}[k] - \mathbf{x}_{i,w}[k])\|_2 > R_{\text{col}}\ \forall k \geq k_{bl}, \forall i \in \mathcal{N}_{[j]} \cap \mathcal{P}_j \setminus \mathcal{D}_j$ **then**
16: Remove j from $\mathcal{K}$
17: **end if**

18: **end for**
19: $w := w + 1$
20: **end while**

A model predictive control implementation of SCP (Algorithm 2) can be used to implement the DAA-D and SCPn methods in real time in order to simultaneously solve the SOCA problems. MPC uses a receding horizon to update the optimal target assignments for docking (Algorithm 1) and current trajectories obtained via communication with neighbors and on-board sensors. In this algorithm, each agent finds its optimal trajectory using the nominal trajectories received from other agents, then corrects the linearization error. This corrected trajectory is the new nominal trajectory, which is then communicated with the neighbors again. If the stopping condition is met, the agent is done optimizing, otherwise, it continues iterating through SCP.

SOCA is described in Algorithm 3. In this algorithm, the each agent determines its cost to each target, then runs Algorithm 1. Then, using the terminal position from the auction, Algorithm 2 is run to generate the trajectory. This continues in a loop until the final time is reached. If the communication graph grows, the cyclical nature of the algorithm allows the assignments to change to accommodate.

Algorithm 3 Swarm Orbital Construction Algorithm (SOCA)

1: $k_0 = 0$
2: **while** $k_0 \leq T$ **do**
3: **for all** $i = 1, \ldots, N$ (*parallel*) **do**
4: **for all** $j = 1, \ldots, M$ **do**
5: Solve Problem 1 using SCP (Algorithm 2)
6: $\mathbf{c}^i(j)$ = cost of optimal solution to Problem 1
7: **end for**
8: **end for**
9: Solve Problem 2 using DAA-D (Algorithm 1)
10: $\mathbf{x}_{j,f}$ = solution to Problem 2, $\forall j$
11: **if** # of bids has changed **then**
12: $k_0 = 0$
13: **end if**

Solve Problem 3 using SCP (Algorithm 2)
$\mathbf{u}_j[k]$ = control solution to Problem 3, $\forall j$, $k = k_0 \dots k_0 + T_H - 1$
Apply $\mathbf{u}_j[k]$ for $k = k_0 \dots k_0 + T_H - 1$
Update k_0 and $\mathbf{x}_{j,k_0}$ to current time
end while

4. Simulation of 3D Spacecraft Dynamics with Attitude

The algorithm with the above modifications was implemented in MATLAB using CVX, a MATLAB-based convex optimization engine running the SDPT3 solver [16, 17, 37].

Simulations were performed using high-fidelity relative orbit dynamics [21] with J_2 perturbations with a virtual chief in a 500 km, 45° inclination orbit. All spacecraft are assumed to have a communication radius of 500 m. The attitude dynamics used are Euler's rotational equation. The J_2 perturbed orbital dynamics of each agent j are described in the LVLH frame by the following equations [21]:

$$\ddot{x}_j = 2\dot{y}_j\omega_z - x_j(\eta_j^2 - w_z^2) + y_j\alpha_z - z_j\omega_x\omega_z - (\zeta_j - \zeta)\sin(i)\sin(\theta) - r(\eta_j^2 - \eta^2) \tag{42}$$

$$\begin{aligned}\ddot{y}_j =& 2\dot{x}_j\omega_z + 2\dot{z}_j\omega_x - x_j\alpha_z - y_j(\eta_j^2 - w_z^2 - \omega_x^2) + z_j\alpha_x \\ &- (\zeta_j - \zeta)\sin(i)\cos(\theta) + x_j\omega_z - z_j\omega_x)\end{aligned} \tag{43}$$

$$\ddot{z}_j = 2\dot{y}_j\omega_x - x_j\omega_x\omega_z - y_j\alpha_x - z_j(\eta_j^2 - w_x^2 - (\zeta_j - \zeta)\cos(i)) \tag{44}$$

with parameters as defined in [21]. These equations are then linearized with respect to a nominal orbit found for each agent j for use in MPC-SCP. The attitude dynamics use Euler's rotational equation and attitude kinematics:

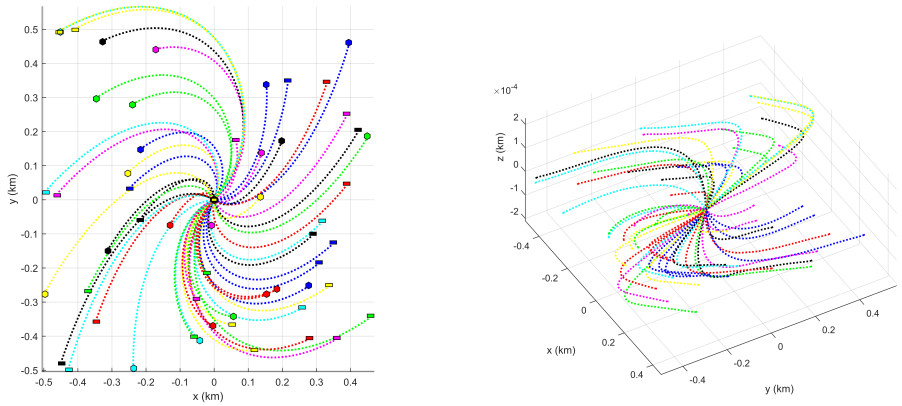
$$\mathbf{J}\dot{\omega} + \omega \times (\mathbf{J} \cdot \omega) = \tau_{\text{ext}} \tag{45}$$

$$\dot{\theta} = \mathbf{Z}(\theta)\omega \tag{46}$$

where $\mathbf{J}$ is the inertia matrix, ω is the angular velocity, τ_{ext} is the external torque vector acting on the body, θ is the vector of 3-2-1 Euler angles, and $\mathbf{Z}(\theta)$ is the corresponding kinematic transformation matrix for these Euler angles [4, 34]. These equations are linearized with respect to a nominal attitude trajectory.

4.1. Simulation Results

The first simulation uses 24 connectors and 30 rods targeted to a planar flower shape, though the trajectories to achieve the arrangement are three dimensional. In the figures presenting the results, rod agents are represented by rectangular prisms and connector agents are represented by hexagonal prisms, rotated to the trajectory orientation. Initial positions are enlarged to show orientation. The agents begin in J_2 invariant relative orbits, which greatly reduce the energy required to maintain the orbit and would likely be used for agents awaiting docking. The agents have an initial separation of up to 1.5 kilometer and final separation of twenty centimeters.



(a) 2D View of Trajectories **(b) 3D View of Trajectories**

Figure 10: 54 agents (30 rods, 24 connectors) combine from up to 1.5 km apart to make a planar hexagon with a 20 cm separation. Note the out of plane motion is all within 0.2 meters.

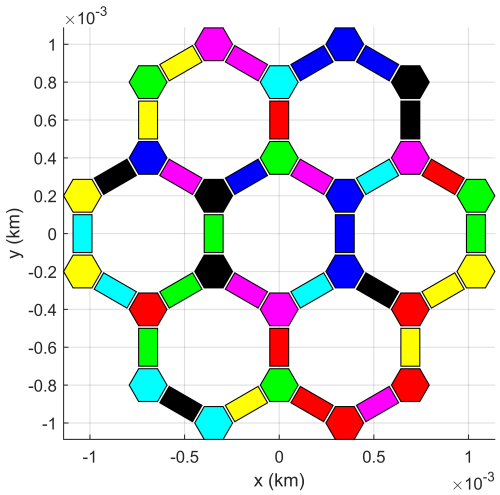
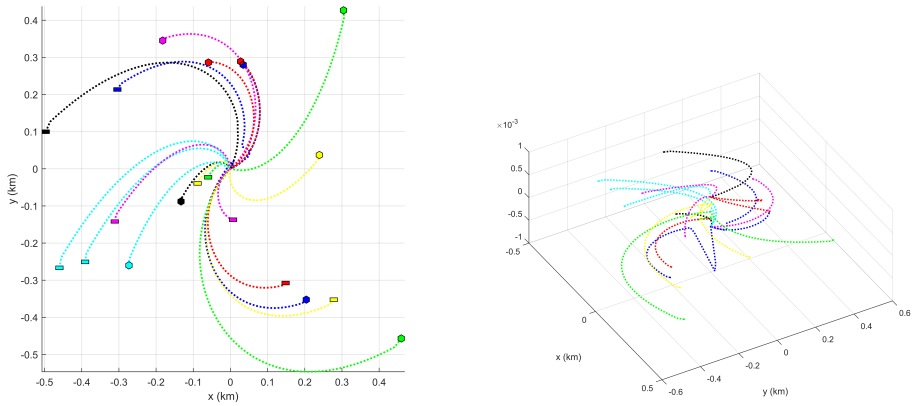


Figure 11: Zoomed in view of the final time step of all 54 agents in the final flower shape

Figure 10a shows the 2D view of the overall trajectories of the agents over the duration of the simulation. The 3D trajectories are shown in Figure 10b. Though the agents are allowed to move in three dimensions, the out of plane motion is minimal (<20 cm) due to the expense of out of plane motion and the planar nature of the target configuration. The scale difference is too large in these figures to show the agents achieve the target configuration since the final separation is so small. Figure 11 shows the position and orientation of all agents at the final time step. The agents reach the desired terminal configuration.

The second simulation uses 10 connectors and 10 rods targeted to a three dimensional folded hexagon shape. Figure 12a shows the 2D view of the overall trajectories of the agents over the duration of the simulation. The 3D trajectories are shown in Figure 12b. The out of plane motion in this simulation is larger than the previous simulation, about one meter. This is still very small, just above the 65 centimeters required of the target shape. Figure 13 shows the position and orientation of all agents at the final time step. Again, all agents reach the desired terminal configuration.



(a) 2D View of Trajectories (b) 3D View of Trajectories

Figure 12: 20 agents (10 rods, 10 connectors) combine from up to 1.5 km apart to make a folded hexagon with a 20 cm separation. Note the out of plane motion is still small, but up to one meter

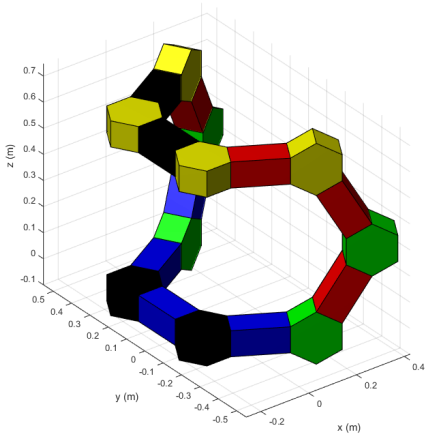


Figure 13: Zoomed in view of the final time step of all 20 agents in the 3D folded hexagon shape

5. 3DOF Experiment on Wheeled Robots and M-STAR Spacecraft Simulators

Experimental validation of the algorithm was performed first on wheeled robots in a motion capture environment then on spacecraft simulators in the flat floor facility at Caltech. The SOCA algorithm can generate optimal collision-free trajectories from any set of initial conditions within the workspace to create the desired shape at the specified final time, but in these experiments the algorithm is run offline so the robots are initialized in the same configuration each run so that the safety of the generated trajectory can be evaluated prior to running the robots.

5.1. Omni-Wheeled Robot Experimental Validation

Initial experiments were performed using six NEXUS 3-Wheeled Compact Omni-Directional Arduino Compatible Mobile Robots (shown in Figure 14). The robots are controlled using Arduinos with Digi XBee communication devices. The SOCA is run on a separate computer, which uses ROS to access the motion capture system, run the algorithm, and send commands to the robots. This computer then communicates commands to the robots through the XBee serial wireless modules attached to each of the robots. SOCA generates trajectories which require motion capture feedback to follow, so each robot is given its current position and orientation, along with the trajectory waypoint. Onboard the robot, a PD controller is used to generate the desired wheel rotations per minute (RPM), then map those values to the actual robot wheel commands, a set of three Pulse Width Modulation (PWM) values. A six-robot system was tested in a 2-meter by 2-meter motion capture space, where each robot read in its trajectory, given by a list of waypoints provided by an offline system SOCA algorithm.

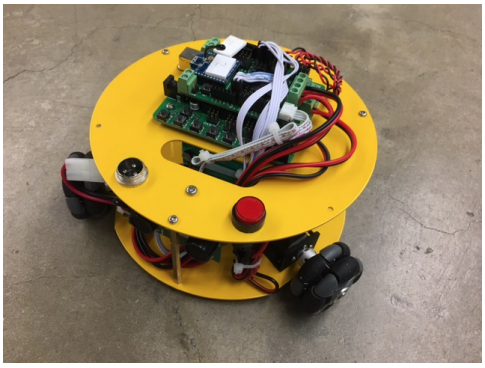


Figure 14: Omni-Directional 3-wheeled robot used to experimentally validate SOCA

Multiple preliminary tests were performed to characterize the relationship between the input PWM signal and the motor RPM. Inconsistencies were observed in the RPM at a constant PWM even on smooth surfaces. This problem was ameliorated by closing the loop onboard the robots using the motor encoders, which gives better control over the speed and path of the robot. The encoders are read at each control loop (about 0.1s) and the PWM is corrected based on the encoder readings. For more details see [15]. The experimental results show SOCA performing assignment and trajectory generation for 6 agents in planar final configuration with realistic 3DOF trajectories.

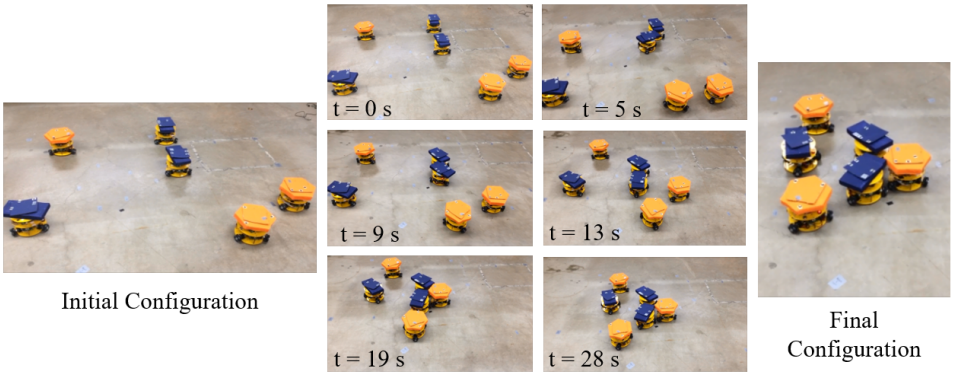


Figure 15: Time lapse of 6 omni-directional robots for SOCA experiment

The off-board control computer ran the SOCA algorithm given the starting points of each of the six robots and determined the optimal trajectory

with collision avoidance. The starting positions of the robots and the time lapse of the full test is shown in Fig 15. The algorithm worked very well however the onboard controller on the robots could not track the trajectories sufficiently well to enable docking. The tracking error of the robots is approximately 10 cm, which is sufficient to eliminate the collision avoidance effects of SOCA. This causes the two rectangular agents to get stuck at t=5-10sec and the bottom rectangular agent to miss its final orientation. To within the tracking error, all of the robots follow the trajectories well. Further investigation is needed to conclusively determine the source of the error but the culprit is most likely the low-level motor controller which has very inconsistent performance due to the noisy wheel encoders. The intended trajectory is plotted in Figure 16 with the actual trajectories achieved by the robots. For the most part, each of the robots achieves the correct direction of travel, but the tracking error prevents the trajectories from lining up perfectly. The wiggle seen in blue at the bottom of the figure is likely cause by this error in wheel actuation. The feedback keeps pulling the robot back towards the desired trajectory but the wheel errors keep causing deviations. The actuation errors in this ground robot system were a motivating factor in the development of the spacecraft simulator facility.

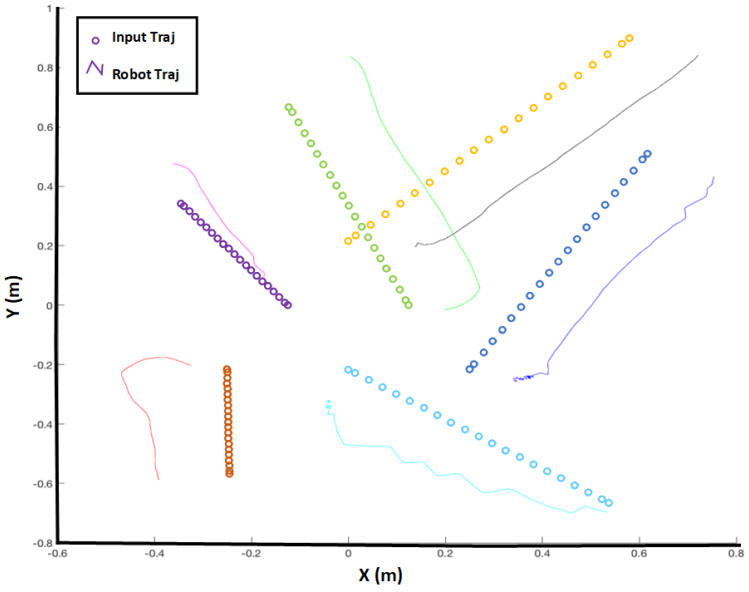


Figure 16: Actual vs desired robot trajectories

5.2. Spacecraft Simulator Facility Overview

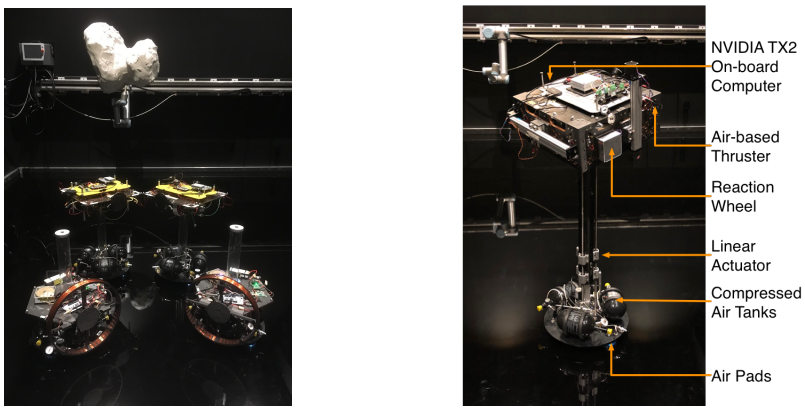


Figure 17: Caltech's Spacecraft Simulator Facility and Four M-STARs with the Two Docking Systems (left) and M-STAR diagram (right)

The Caltech Aerospace Robotics and Control Lab's Spacecraft Simulator Facility is composed of an ultra-precise epoxy flat floor shown in Figure 17, a clean room, and five M-STAR robots. The flat floor facility is the largest such facility at any university. It has two 7DOF robot arms mounted on linear actuators along the back and the side. The space is fully covered by 14 motion capture cameras placed around the exterior to track the position and orientation of each M-STAR.

The M-STARs, shown in Figure 17, use flat and spherical air bearings to achieve 5DOF frictionless motion and a linear actuator to achieve kinematic motion in the gravity direction. The M-STARs are equipped with thrusters, reaction wheels, electromagnets, and docking ports as actuators. Due to the design of the facility and the M-STARs, friction between the floor and the linear air bearings is negligibly small. The M-STARs are designed to be modular, capable of transforming from a 6DOF platform to a 3DOF and everything in between. For the purposes of this experiment, the M-STARs were in two 3DOF configurations with different docking port locations. Eight of the 16 onboard thrusters were used for 3DOF position and attitude control. For more details on the facility and the simulators, see [25].

5.3. In-Orbit Construction Experiment on the M-STARs

For this experiment, the four M-STARS made a T shape in the center of a 4 meter square. Three of the four agents were identical, with two docking

ports placed on opposite sides of the square upper stage. The fourth M-STAR had four ports, one on each side of the upper stage. These specifications were input into SOCA to generate optimal, collision-free construction trajectories. Trajectories were computed offline then fed to the M-STARs, which followed them using the onboard control scheme described in [13].

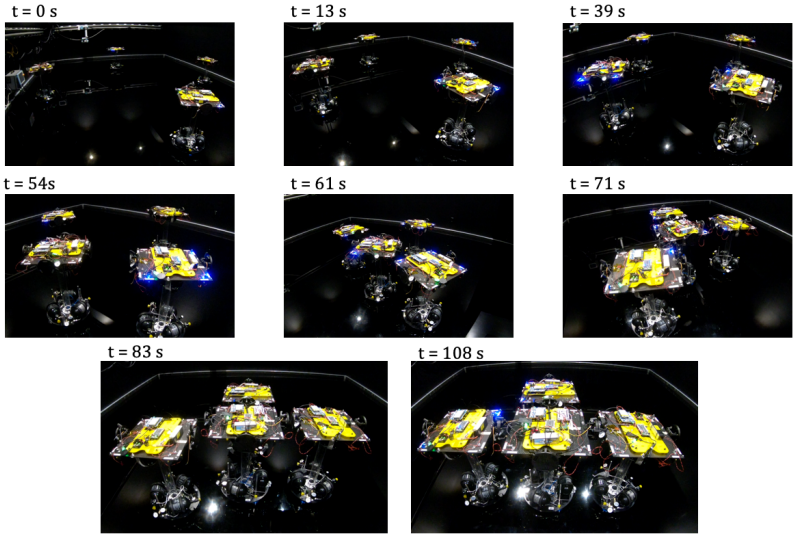


Figure 18: Time Lapse of 4 M-STARs Following SOCA Optimal Trajectories in Spacecraft Simulator Facility

Due to the aggressive nature of the optimal trajectories, the simulators come very close. A sample experimental trajectory is seen in Figure 19, and the discrepancy between the command and actual trajectory is shown in Figure 20. Stills from the experiment are shown in Figure 18. The biggest issue is when spacecraft 4 encounters some friction on the floor at 55 s, causing the actual trajectory to deviate from the desired. A video of the experiment can be found at `https://youtu.be/62cngDR1k-E`.

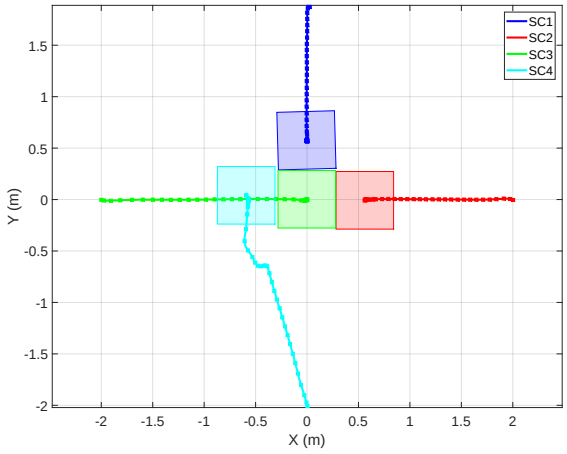


Figure 19: Full Motion Captured Trajectories of 4 M-STARs Following SOCA Optimal Trajectories

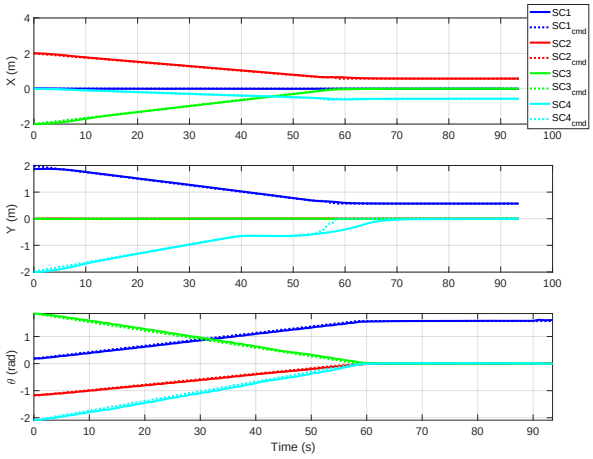


Figure 20: Experimental performance of M-STARS as compared to command

In general, the runtime of this algorithm in MATLAB simulations increases with additional agents because all agents are run on a single machine in these simulations. When these algorithms are run on separate computers onboard the simulators, the computation is expected to be sufficiently fast to adapt to changes in the scenario as they arise.

6. Conclusion

A distributed optimal guidance algorithm has been presented to enable self-assembly of a heterogeneous swarm of component satellites with limited communication radii. The agent types chosen can create a diverse set of final docked configurations which can cover the plane and build out-of-plane. This enhances prior work in the field because it is both distributed and heterogeneous, can function in a complex dynamic environment, and accounts for relative attitude dynamics. The SOCA algorithm handles the limited-type heterogeneity by adding barrier functions in the target assignment algorithm. The barrier functions make the cost of an improper assignment prohibitive by assessing dock metrics like number and angle of docks required at that target position and inflating the cost when those qualities are not matched by the agent type. The algorithm also creates collision-free trajectories while still allowing agents to dock by adding a boundary layer outside of the collision avoidance radius wherein the agent must choose to avoid or dock with an approaching agent. The algorithm was also made robust to uncertainty in the nominal trajectories of neighbors, and improved the handling of nonlinear dynamics to make the trajectories commanded by SOCA realistic and achievable.

The simulation results show SOCA performing assignment and trajectory generation for 20-54 agents in two and three dimensional final configurations with realistic trajectories. Preliminary experimental validation was performed on-board six wheeled robots, but the precision of the wheeled robot motion was insufficient. Four spacecraft simulators were used to successfully experimentally validate the optimal construction algorithm. The proposed scheme is useful for missions ranging from sparse aperture interferometric telescope construction to space colony or station construction. This algorithm can also be used for a higher degree of heterogeneity without substantially altering the algorithm, as discussed in the auction algorithm section.

Appendix

Proof of Theorem 2

Proof. Given that for some w, $(\mathbf{x}_n^{(w)}[k_0:T], \mathbf{u}^{(w)}[k_0:T-1])$ is a feasible solution of the NCP, Problem 4, we know that the constraints of the NCP are satisfied. Since $(\mathbf{x}_n^{(w)}[k_0:T], \mathbf{u}^{(w)}[k_0:T-1])$ is also used to linearize and discretize the dynamics in SCPn$^{(w+1)}$, it must satisfy Eq. (24) and thus is feasible to SCPn$^{(w+1)}$.

Since the cost of an optimal solution must be less than or equal to a feasible solution we arrive at:

$$J(\mathbf{u}^{(w+1)}[k_0 : T-1]) \leq J(\mathbf{u}^{(w)}[k_0 : T-1]) \tag{47}$$

Furthermore, this optimal input solution ($\mathbf{u}^{(w+1)}[k_0 : T-1]$) of SCPn$^{(w+1)}$ is used to generate a new nominal trajectory using Eq. (32). This new nominal, ($\mathbf{x}_n^{(w+1)}[k_0 : T-1], \mathbf{u}^{(w+1)}[k_0 : T-1]$) must also be feasible to Problem 4 if Eq. (41) holds.

Combining the first-order convexity conditions of Eq. (38) for each g_i with the constraints themselves results in

$$\begin{aligned} g_i(\mathbf{x}_n^{(w+1)}[k], \mathbf{u}^{(w+1)}[k]) \leq \left.\frac{\partial g_i}{\partial \mathbf{x}[k]}\right|_{(\mathbf{x}_n^{(w+1)}[k], \mathbf{u}^{(w+1)}[k])} (\mathbf{x}_n^{(w+1)}[k] - \mathbf{x}^{(w+1)}[k]) \\ - \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j} \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \end{aligned} \tag{48}$$

To show ($\mathbf{x}_n^{(w+1)}[k_0 : T], \mathbf{u}^{(w+1)}[k_0 : T-1]$) is feasible to the NCP, we need to prove $g_i(\mathbf{x}_n^{(w+1)}[k], \mathbf{u}^{(w+1)}[k]) \leq \mathbf{0}$, whose sufficient condition can be given as

$$\left\| \frac{\partial g_i}{\partial \mathbf{x}[k]} \right\| (\|\mathbf{x}_n^{(w+1)}[k] - \mathbf{x}_n^{(w)}[k]\| + \|\mathbf{x}^{(w+1)}[k] - \mathbf{x}_n^{(w)}[k]\|) \leq \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j} \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \tag{49}$$

Plugging in the initial condition to Eq. (24) for SCPn$^{(w+1)}$ becomes

$$\begin{aligned} \mathbf{x}^{(w+1)}[k] - \mathbf{x}_n^{(w)}[k] = \sum_{j=k_0}^{k-2} \left(\prod_{i=j+1}^{k-1} A(\mathbf{x}_n^{(w)}[k+j-i]) \right) B(\mathbf{u}^{(w)}[j])(\mathbf{u}^{(w+1)}[j] - \\ \mathbf{u}^{(w)}[j]) + B(\mathbf{u}^{(w)}[k-1])(\mathbf{u}^{(w+1)}[k-1] - \mathbf{u}^{(w)}[k-1]) \end{aligned} \tag{50}$$

Assuming $\mathbf{f}(\mathbf{x}_n^{(w)}[k], \mathbf{u}^{(w)}[k])$ is Lipschitz over $\mathcal{D}$:

$$\begin{aligned} \|\mathbf{x}_n^{(w+1)}[k+1] - \mathbf{x}_n^{(w)}[k+1]\| &= \|\mathbf{f}(\mathbf{x}_n^{(w+1)}[k], \mathbf{u}^{(w+1)}[k]) - \mathbf{f}(\mathbf{x}_n^{(w)}[k], \mathbf{u}^{(w)}[k])\| \\ &\leq \|\overline{A}\| \|\mathbf{x}_n^{(w+1)}[k] - \mathbf{x}_n^{(w)}[k]\| + \|\overline{B}\| \|\mathbf{u}^{(w+1)}[k] - \mathbf{u}^{(w)}[k]\| \end{aligned} \tag{51}$$

where $\|\overline{A}\| = \sup_{(\mathbf{x}[k], \mathbf{u}[k]) \in \mathcal{D}} \|A(\mathbf{x}[k])\|$, and $\|\overline{B}\| = \sup_{(\mathbf{x}[k], \mathbf{u}[k]) \in \mathcal{D}} \|B(\mathbf{u}[k])\|$.

This can be expressed as a function of $\|\mathbf{u}^{(w+1)}[k] - \mathbf{u}^{(w)}[k]\|$ using $\mathbf{x}_n^{(w+1)}[k_0] = \mathbf{x}_n^{(w)}[k_0]$ as follows

$$\|\mathbf{x}_n^{(w+1)}[k] - \mathbf{x}_n^{(w)}[k]\| \leq \sum_{j=k_0}^{k-1} (\|\overline{A}\|)^{k-j-1} \|\overline{B}\| \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| \tag{52}$$

Applying a property of submultiplicativity of norms to Eq. (50) shows that both $\|\mathbf{x}_n^{(w+1)}[k] - \mathbf{x}_n^{(w)}[k]\|$ and $\|\mathbf{x}^{(w+1)}[k] - \mathbf{x}_n^{(w)}[k]\|$ possess the same upper-bound given in Eq. (52). Hence, substituting Eq. (52) into Eq. (49) shows that Eq. (49) is satisfied by

$$\sum_{j=k_0}^{k-1} 2 \left\| \frac{\partial g_i}{\partial \mathbf{x}[k]} \right\| (\|\overline{A}\|)^{k-j-1} \|\overline{B}\| \|\mathbf{u}_j^{(w+1)} - \mathbf{u}_j^{(w)}\| = \sum_{j=k_0}^{k-1} \mathcal{L}_{i,j} \|\mathbf{u}^{(w+1)}[j] - \mathbf{u}_j^{(w)}\| \tag{53}$$

Consequently, the condition of $\mathcal{L}_{i,j}$ Eq. (41) is established. Since the nonlinear dynamics constraint Eq. (34) is already satisfied, we conclude that if $(\mathbf{x}_n^{(w)}[k_0 : T], \mathbf{u}^{(w)}[k_0 : T-1])$ is a feasible solution to the NCP, when SCPn is applied, the optimal solution to SCPn is also a feasible solution to the NCP.

□

Acknowledgments

This work was supported by a NASA Space Technology Research Fellowship. Government sponsorship is acknowledged. This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA.

References

[1] Syed Jan Abas and Ahmed Salman. Geometric and group-theoretic methods for computer graphic studies of Islamic symmetric patterns. In *Computer Graphics Forum*, volume 11, pages 43–53, 1992.

[2] Ahmad Aljamali and Ebad Banissi. Normalization and exploration design method of Islamic geometric patterns. In *International Conference on Geometric Modelling and Graphics, London, U.K.*, pages 42–48, 2003.

[3] Phillip Ball. Islamic tiles reveal sophisticated maths. http://www.nature.com/news/2007/070219/full/news070219-9.html. Accessed: 2017-02-28.

[4] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y. Hadaegh. Nonlinear attitude control of spacecraft with a large captured object. *Journal of Guidance, Control, and Dynamics*, 39(4):754–769, 2016.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, U.K., 2004.

[6] John Chilton. *Space grid structures*. Taylor & Francis, 2007.

[7] Robert Coolman. Tessellation: The geometry of tiles, honeycombs and M.C. Escher. http://www.livescience.com/50027-tessellation-tiling.html. Accessed: 2017-3-17.

[8] Steven Dutch. Structure of beryl and cordierite. https://www.uwgb.edu/dutchs/Petrology/Beryl-CordStruc.htm. Accessed: 2017-2-28.

[9] Jacob Everist, Kasra Mogharei, Harshit Suri, Nadeesha Ranasinghe, Berok Khoshnevis, Peter Will, and Wei-Min Shen. A system for inspace assembly. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan*, volume 3, pages 2356–2361, 2004.

[10] Wigbert Fehse. *Automated rendezvous and docking of spacecraft*, volume 16. Cambridge University Press, 2003.

[11] Rebecca Foust, Soon-Jo Chung, and Fred Y. Hadaegh. Optimal guidance and control with nonlinear dynamics using sequential convex programming (to appear). *Journal of Guidance Control and Dynamics*, 2019.

[12] Rebecca Foust, Soon-Jo Chung, and Fred Y. Hadaegh. Solving optimal control with nonlinear dynamics using Sequential Convex Programming. In *AIAA Guidance, Navigation, and Control Conference, San Diego, California*, page 0652, 2019.

[13] Rebecca C. Foust, E. Sorina Lupu, Yashwanth K. Nakka, Soon-Jo Chung, and Fred Y. Hadaegh. Ultra-soft electromagnetic docking with applications to in-orbit assembly. In *International Astronautical Congress, Bremen, Germany*, 2018.

[14] Rebecca C. Foust, Yashwanth K. Nakka, Ayush Saxena, Soon-Jo Chung, and Fred Y. Hadaegh. Automated rendezvous and docking using tethered formation flight. In *9th International Workshop on Satellite Constellations and Formation Flying*, 2017.

[15] Rebecca C. Foust, Michelle Zhao, Suzanne Oliver, Soon-Jo Chung, and Fred Y. Hadaegh. Distributed control of an evolving satellite assembly during in-orbit construction. In *International Astronautical Congress*, 2017.

[16] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[17] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[18] Fred Y. Hadaegh, Soon-Jo Chung, and Harish M Manohara. On development of 100-gram-class spacecraft for swarm applications. *IEEE Systems Journal*, 10(2):673–684, 2016.

[19] A. Scott Howe and Ian Gibson. Trigon robotic pairs. In *AIAA Space*, 2006.

[20] Peter J. Lu and Paul J. Steinhardt. Decagonal and quasi-crystalline tilings in medieval Islamic architecture. *Science*, 315(5815):1106–1110, 2007.

[21] Daniel Morgan, Soon-Jo Chung, Lars Blackmore, Behcet Acikmese, David Bayard, and Fred Y. Hadaegh. Swarm-keeping strategies for spacecraft under J_2 and atmospheric drag perturbations. *Journal of Guidance, Control, and Dynamics*, 35(5):1492–1506, 2012.

[22] Daniel Morgan, Soon-Jo Chung, and Fred Y. Hadaegh. Model predictive control of swarms of spacecraft using sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 37(6):1725–1740, 2014.

[23] Daniel Morgan, Giri P. Subramanian, Soon-Jo Chung, and Fred Y. Hadaegh. Swarm assignment and trajectory optimization using

variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, 35(10):1261–1285, 2016.

[24] Zoltán Nagy, Jake J. Abbott, and Bradley J. Nelson. The magnetic self-aligning hermaphroditic connector a scalable approach for modular microrobots. In *2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1–6, 2007.

[25] Yashwanth K. Nakka, Rebecca C. Foust, E. Sorina Lupu, David B. Elliott, Irene S. Crowell, Soon-Jo Chung, and Fred Y. Hadaegh. Six degree-of-freedom spacecraft dynamics simulator for formation control research. In *2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird, UT*, 2018.

[26] Hai D. Nguyen Nathan Howard. Magnetic capture docking system. https://patents.google.com/patent/US7815149B1/en. Accessed: 2019-07-26.

[27] Mohamed Okasha, Chandeok Park, and Sang-Young Park. Guidance and control for satellite in-orbit-self-assembly proximity operations. *Aerospace Science and Technology*, 41:289–302, 2015.

[28] Roger Penrose. Pentaplexity. *Eureka*, 39:16–32, 1978.

[29] Máximo A. Roa, Korbinian Nottensteiner, Armin Wedler, and Gerhard Grunwald. Robotic technologies for in-space assembly operations. In *Proc. 14th Symp. Adv. Space Technol. Robot. Autom.*, 2017.

[30] John W. Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295, 2013.

[31] David Saldana, Bruno Gabrich, Michael Whitzer, Amanda Prorok, Mario F.M. Campos, Mark Yim, and Vijay Kumar. A decentralized algorithm for assembling structures with modular robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2736–2743, 2017.

[32] Yuchen She, Shuang Li, Bin Du, and Kai Cao. On-orbit assembly mission planning considering topological constraint and attitude disturbance. *Acta Astronautica*, 152:692–704, 2018.

[33] Wei-Min Shen, Peter Will, and Berok Khoshnevis. Self-assembly in space via self-reconfigurable robots. In *IEEE International Conference on Robotics and Automation, Taipei, Taiwan*, volume 2, pages 2516–2521, 2003.

[34] Malcolm D. Shuster. A survey of attitude representations. *Navigation*, 8(9):439–517, 1993.

[35] Yoonchang Sung, Ashish Kumar Budhiraja, Ryan K. Williams, and Pratap Tokekar. Distributed simultaneous action and target assignment for multi-robot multi-target tracking. In *International conference on robotics and automation (ICRA), Brisbane, Australia*, pages 1–9, 2018.

[36] Chiara Toglia, Fred Kennedy, and Steven Dubowsky. Cooperative control of modular space robots. *Autonomous Robots*, 31(2-3):209–221, 2011.

[37] K. C. Toh, M. J. Todd, and R. H. Tutuncu. Sdpt3 — a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[38] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory planning and assignment in multirobot systems. In *Algorithmic Foundations of Robotics X*, pages 175–190. Springer, 2013.

[39] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014.

[40] Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, 37(4):401–415, 2014.

[41] Eric W. Weisstein. Space-filling polyhedra. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Space-FillingPolyhedron.html. Accessed: 2016-12-16.

[42] Mark Yim, Kimon Roufas, David Duff, Ying Zhang, Craig Eldershaw, and Sam Homans. Modular reconfigurable robots in space applications. *Autonomous Robots*, 14(2-3):225–237, 2003.

[43] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S. Chirikjian. Modular self-

reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

[44] Jingjin Yu, Soon-Jo Chung, and Petros G. Voulgaris. Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies. *IEEE Transactions on Automatic Control*, 60(2):327–341, 2015.

[45] Victor Zykov, Andrew Chan, and Hod Lipson. Molecubes: An open-source modular robotics kit. In *International Conference on Intelligent Robots and Systems, Self-Reconfigurable Robotics Workshop*, pages 3–6, 2007.