# Finding Structure Configurations for Flying Modular Robots

Bruno Gabrich[1], David Saldaña[2], and Mark Yim[1]

***Abstract*— Flying Modular Structures offer a versatile mechanism that can change the arrangement of constituent actuators according to task requirements. In this work, we extend a modular aerial platform that can expand its actuation capabilities depending on the configuration. Each module is composed of a quadrotor in a cage that can rigidly connect with other modules. The quadrotor is connected with the cage by a revolute joint that allows it to rotate with respect to the cage. Modules located in the structure are either parallel or perpendicular to one another. The task specification defines forces and moments needed during the execution. We propose two search methods to find a configuration that can satisfy the specification. The first approach consists of an exhaustive search that yields optimal structure configurations by exploring the whole search space. The second approach proposes a heuristic based on subgroup search, reducing the problem complexity from exponential to linear. We validate our proposed algorithms with several simulations. Our results show that the proposed heuristic is computationally efficient and finds a near-optimal configuration even for flying modular structures composed of a large number of modules.**

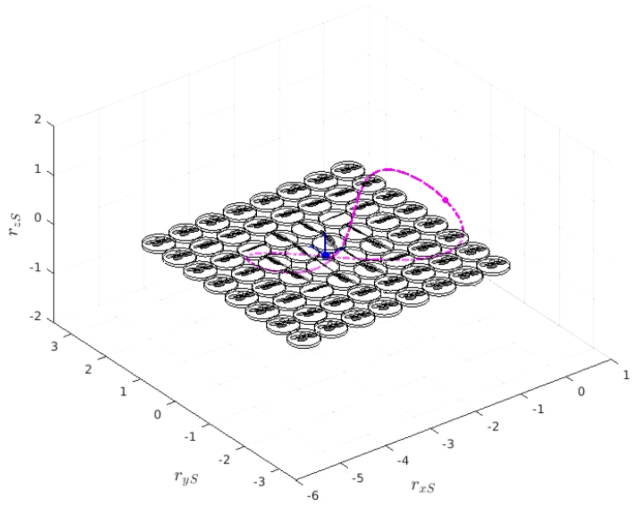

Fig. 1. A flying modular structure composed of 64 modules in a square configuration following a desired trajectory.

## I. Introduction

Flying modular robots have been rapidly explored in recent years due to the ease and accessibility of small and low-cost quadrotors. The use of quadrotors as the smallest system element enables the system to self-assemble in midair forming 2-D rigid structures [1], [2]. Before this, flying modular structures successfully performed self-assembly on the ground [3]. Different types of structures with different numbers of modules were successfully assembled. Because these structures have rotors aligned in parallel, they are only capable of controlling four degrees of freedom (i.e., as in traditional quadrotors) even though the conglomerate has many more than four motors.

The major advantage of flying modular robots over other cooperative flying systems is the ability to self-reconfigure in midair [1], [4] and adapt to varied tasks or in response to faulty modules in the system [5]. Scaling modular robots by increasing the number of modules implies an increase in the system complexity and cost of the platform, thus simplicity at the modular level is a desired feature. Modular robots lie at the intersection of versatility and robustness [6],[7],[8] due to self-reconfiguration. A system that allows modular robots to complete high-level tasks was developed in [9], by enabling modules to autonomously self-reconfigure. In [10], self-assembling boats formed floating structures. In [11] and [12], in-flight grasping capabilities were embedded to the flying modular system. This was the first investigation of non-rigid passive connections between flying modules enabling grasping of objects without a gripper or manipulator arm attachment. In [13], Zhao et al. explored a novel design for flying modular robots using modules composed of two degree of freedom joints and a dual-rotor gimbal, which allowed the flying vehicle to change the angles between modules in-flight.

One of the limitations of *ModQuad* [1] is the loss of yaw control as the number of modules scale up due to actuator saturation. The work developed in [14] investigated cases for which quadrotors could tilt and achieve unique roll attitude angles enabling a novel yaw actuation method that enlarges the number of modules that flies in cooperation in a line configuration. Furthermore, pointing thrust vectors at different orientations could lead to better utilization of the system's redundancies. In [15], [16], [17], [18] the authors explored platforms capable of thrust vector at different orientations resulting in fully-actuated 6 DOF (degree of freedom) systems.

In this work we introduce configuration search methods for arbitrary modular configurations utilizing the 1-DOF design presented in [14]. By allowing only 1-DOF at the modular level we simplify design, cost and avoid additional constraints imposed by spherical joints due to its usual range of motion limitation. By the utilization of flying modular robots one could design a structure capable to exert forces

[1] B. Gabrich, M. Yim are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA: {brunot, yim}@seas.upenn.edu.
[2] D. Saldaña is with the Autonomous and Intelligent Robotics Laboratory -AIRLab-, Lehigh University, Bethlehem, PA, USA: saldana@lehigh.edu.
The authors would like to thank Aaron Weinstein for the discussions, ideas, and the introduction to the simulation framework.

and moments with specific maximum magnitudes at specific directions. Given this forces and moments capabilities, these robots could be used to transport a payload, fly over a specific path or even execute non-prehensile manipulation of objects with proper structure configurations.

The contribution of this paper is twofold. *i)* We extend the system actuation capabilities of the work presented in [14] by allowing arbitrary configurations to be assembled. This extension allows the structure to be fully actuated for specific configurations. *ii)* The task specification defines forces and moments needed for execution. We propose two search methods to find a configuration that can satisfy this specification. One finds the optimal structure configuration with a high computational cost, and the other finds a near-optimal structure configuration through a computational efficient solution.

## II. System Model and Definitions

The building block of our platform is called module.

**Definition 1 (Module).** *refers to a flying vehicle with a cage capable of docking with other modules.*

Each module has a mass $m$ and an inertia tensor $\mathbf{I}$. Multiple modules can self-assemble to form a structure.

**Definition 2 (Structure).** *refers to the composition of $n$ modules docked together in the same plane.*

Modules in a structure are indexed by $i = 1, ..., n$. The structure frame $\mathcal{S}$ is located at the structure center of mass. Modules frame $\mathcal{M}_i$ and flying vehicle frame $\mathcal{Q}_i$ are located and the center of mass of each module. Later we define the world frame $\mathcal{W}$ and illustrate all frames in Fig. 2. The structure inertia tensor is denoted by $\mathbf{I}_S$. Furthermore $\mathbf{r}^*_S, \dot{\mathbf{r}}^*_S, \ddot{\mathbf{r}}^*_S$ denotes linear structure translation, velocity and acceleration in $\mathcal{W}$. The structure's angular velocity and acceleration in frame $\mathcal{S}$ are $\mathbf{\Omega}_S$ and $\dot{\mathbf{\Omega}}_S$ respectively. $\phi_i$, $\dot{\phi}_i$, $\ddot{\phi}_i$ indicate module's roll angle, velocity and acceleration in the same order in $\mathcal{M}_i$. The superscript $^*$ designates desired values for any entity. Considering each module as a rigid body, its pose and configuration space is in $SE(3)$. The configuration space of a flying structure composed of $n$ modules is $SE(3) \times (S^1)^n$.

**Definition 3 (Module Configuration).** *refers to parallel or perpendicular orientation of module $i$ x-axis relative to $\mathcal{S}$.*

**Definition 4 (Form Configuration $\mathscr{F}_c$).** *refers to a set of points in $\mathcal{S}$ that represents the position of $n$ docked modules.*

**Definition 5 (Structure Configuration).** *refers to the composition of module and form configuration.*

The structure orientation in $\mathcal{W}$ is represented by $\mathbf{R}^{\mathcal{W}}_{\mathcal{S}} \in SO(3)$, and $\mathbf{R}^{\mathcal{S}}_{\mathcal{M}_i} \in \{\mathbf{R}_{z,k\frac{\pi}{2}} : k = 0, 1\}$. $\mathbf{R}^{\mathcal{S}}_{\mathcal{M}_i}$ is module configuration dependent and represents the orientation of each module w.r.t. $\mathcal{S}$, where $\mathbf{R}_{z,k\frac{\pi}{2}}$ is a rotation matrix around the $z$-axis. $\mathbf{R}^{\mathcal{M}_i}_{\mathcal{Q}_i} \in \{\mathbf{R}_{x,\phi_i}\}$ is flight behavior dependent and represents the flying vehicle frame w.r.t. $\mathcal{M}_i$,

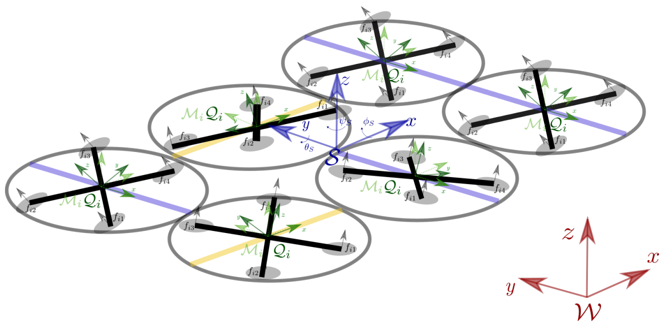

Fig. 2. A six modules structure configuration in a rectangular form configuration. Each module has its module configuration determined by its yaw orientation relative to $\mathcal{S}$.

whereas $\mathbf{R}_{x,\phi_i}$ is a rotation matrix around the $x$-axis. The translational dynamics is then defined by

$$nm\ddot{\mathbf{r}}_S = -nm\, g\, \mathbf{e}_3 + \mathbf{R}^{\mathcal{W}}_{\mathcal{S}} \sum_i \mathbf{R}^{\mathcal{S}}_{\mathcal{M}_i} \mathbf{R}^{\mathcal{M}_i}_{\mathcal{Q}_i} \mathbf{e}_3 \sum_j f_{ij},$$

where $g$ is the gravity constant, $\mathbf{e}_3 = [0, 0, 1]^\top$. $f_{ij}$ corresponds to the force produced by actuator $j = 1, ..., 4$ of module $i$, which in fact, are our system control inputs. Given these definitions the rotational dynamics is defined as follows

$$\mathbf{I}_S \dot{\mathbf{\Omega}}_S + \mathbf{\Omega}_S \times \mathbf{I}_S \mathbf{\Omega}_S = \sum_i \mathbf{r}^{\mathcal{S}}_i \times \mathbf{R}^{\mathcal{S}}_{\mathcal{M}_i} \mathbf{R}^{\mathcal{M}_i}_{\mathcal{Q}_i} \mathbf{e}_3 \sum_j f_{ij},$$

where $\mathbf{r}^{\mathcal{S}}_i = [r_{xi}, r_{yi}, 0]^\top$ is the $ith$ module position in $\mathcal{S}$. The structure inertia tensor can be approximated to $\mathbf{I}_S = n\mathbf{I} + m \sum_i \mathrm{Diag}[r^2_{iy}, r^2_{ix}, (r^2_{iy} + r^2_{ix})]$, which comes from the parallel axis theorem.

## III. Flying Modular Structure Control

Before introducing our configuration optimization methods (in Section IV), we define a $SE(3)$ controller for arbitrary structure configurations; initiating from a desired pose and arriving at the quadrotor actuator level. Our approach consists of four steps. First, a desired pose trajectory is used to compute structure desired forces and desired moments. Second, the desired structure behavior is distributed for all modules. Third, thrust and desired roll is computed. Third, the actuator distribution is outlined and control inputs are obtained.

### *A. Structure Desired Wrench*

Based on a geometric controller [19], a centralized structure pose control takes as inputs a pre-defined trajectory, $\mathscr{T} := (\mathbf{r}^*_S(t), \mathbf{R}^{\mathcal{W}*}_{\mathcal{S}}(t))$, that is utilized to compute structure forces and moments as follows

$$\mathbf{F}^{*\mathcal{S}}_S = nm\mathbf{R}^{\mathcal{S}}_{\mathcal{W}}(\ddot{\mathbf{r}}^*_S + \mathbf{K}_p(\mathbf{r}^*_S - \mathbf{r}_S) + \mathbf{K}_d(\dot{\mathbf{r}}^*_S - \dot{\mathbf{r}}_S) + g\mathbf{e}_3), \quad (1)$$

$$\mathbf{M}^*_S = \mathbf{I}_S(\mathbf{K}_R\mathbf{e}_R - \mathbf{K}_\Omega \mathbf{\Omega}_S) + \mathbf{\Omega}_S \times \mathbf{I}_S \mathbf{\Omega}_S, \quad (2)$$

where $\mathbf{K}_p$, $\mathbf{K}_d$, $\mathbf{K}_R$ and $\mathbf{K}_\Omega$ are positive gain matrices, and $\mathbf{e}_R$ is the orientation error that can be written as

$$\mathbf{e}_R = \frac{1}{2}(\mathbf{R}^{\mathcal{S}}_{\mathcal{W}} \mathbf{R}^{\mathcal{W}*}_{\mathcal{S}} - \mathbf{R}^{\mathcal{S}*}_{\mathcal{W}} \mathbf{R}^{\mathcal{W}}_{\mathcal{S}})^\vee,$$

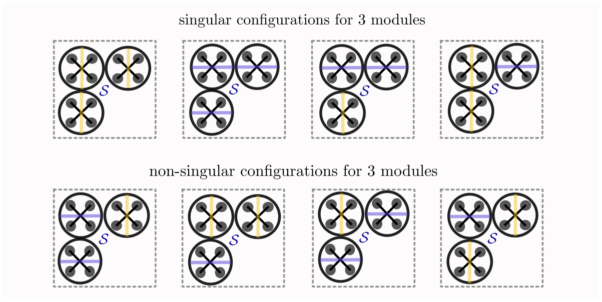
singular configurations for 3 modules
non-singular configurations for 3 modules
$\mathcal{S}$

Fig. 3. Singular and non-singular configurations for structures composed of 3 modules. The two singular configurations on the left cannot move laterally along the $x$ and $y$-axis respectively. The other two singular configurations on the right have respectively a $x-z$ and $y-z$ moments coupling.

where the operator $^\vee$ is the vee map from $SO(3)$ to $\mathbb{R}^3$. From (1) and (2), we obtain a desired structure wrench $\mathbf{W}^* = [\mathbf{F}_S^{*\mathcal{S}}, \mathbf{M}_S^*]^\top$.

### B. Module Force Distribution

The obtained wrench serves as a reference to compute desired force vectors for each module in the structure. Each module has a force $\mathbf{F}_i^{*\mathcal{M}_i}$ the effect of all module forces produces a wrench in the structure

$$\mathbf{W} = \mathbf{S}\mathbf{F}^{\mathcal{M}}, \tag{3}$$

where $\mathbf{F}^{*\mathcal{M}} = [\mathbf{F}_1^{*\mathcal{M}_1\top}, \dots, \mathbf{F}_n^{*\mathcal{M}_n\top}]^\top$ is the vector that compiles all module forces, and $\mathbf{S} \in \mathbb{R}^{6\times 3n}$ is the structure configuration matrix. $\mathbf{S}$ can be written as the product of three matrices

$$\mathbf{S} = \bar{\mathbf{P}}\bar{\mathbf{R}}\bar{\mathcal{I}}_c, \tag{4}$$

where $\bar{\mathbf{P}} \in \mathbb{R}^{6\times 3n}$ is interpreted as the position matrix, $\bar{\mathbf{R}} \in \mathbb{R}^{3n\times 3n}$ the orientation matrix and lastly $\bar{\mathcal{I}}_c \in \mathbb{R}^{3n\times 3n}$ a matrix that constraints modules forces to be produced on its $x$-axis. This constraint is a system feature shown in Fig. 2, meaning that $\mathbf{F}_i^{\mathcal{M}_i} = [0, F_{yi}, F_{zi}]^\top$. Therefore, (3) can be expanded as

$$\mathbf{W} = \begin{bmatrix} \mathcal{I} \dots \mathcal{I} \\ \hat{\mathbf{P}}_1 \dots \hat{\mathbf{P}}_n \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\mathcal{M}_1}^{\mathcal{S}} \dots \mathbf{0} \\ \ddots \\ \mathbf{0} \dots \mathbf{R}_{\mathcal{M}_n}^{\mathcal{S}} \end{bmatrix} \begin{bmatrix} \mathcal{I}_c \dots \mathbf{0} \\ \ddots \\ \mathbf{0} \dots \mathcal{I}_c \end{bmatrix} \mathbf{F}^{\mathcal{M}}, \tag{5}$$

where $\mathcal{I} = \text{Diag}[1,1,1]$, $\mathcal{I}_c = \text{Diag}[0,1,1]$, $\hat{\mathbf{P}}_i = [\mathbf{r}_i^{\mathcal{S}}]\times$ is the skew symmetric matrix of each module position in $\mathcal{S}$.

The structure wrench has fixed dimension whereas force vectors depend on the number of modules. For structures with more than three modules, the system can be redundant depending on module configuration; and therefore, an optimal way to control the structure wrench needs to be defined. In order to find a solution that minimizes module forces we use quadratic programming.

*Quadratic Programming for Force Distribution:* Solutions for $\mathbf{F}^{*\mathcal{M}}$ can be obtained through the formulation of the following quadratic program minimization, which takes into account feasible forces to each module. This optimization is

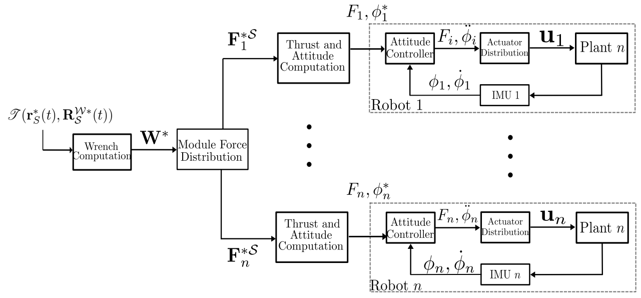

Fig. 4. Flying Modular Structure Control Diagram.

described as follows

$$\begin{aligned} \min_{\mathbf{F}^{*\mathcal{M}}} \quad & \frac{1}{2}\mathbf{F}^{*\mathcal{M}\top}\mathbf{F}^{*\mathcal{M}} \\ \text{s.t.} \quad & \left\|\mathbf{W}^* - \mathbf{S}\mathbf{F}^{*\mathcal{M}}\right\|_2^2 = 0, \\ & \mathbf{F}_{min}^{\mathcal{M}} \leq \mathbf{F}^{*\mathcal{M}} \leq \mathbf{F}_{max}^{\mathcal{M}}, \end{aligned} \tag{6}$$

where the vector operator $\leq$ compares element by element, $\mathbf{F}_{min}^{\mathcal{M}}$ represents lower bound constraints whereas the upper bound is represented by $\mathbf{F}_{max}^{\mathcal{M}}$. We solve this optimization problem using the *quadprog*-function from MATLAB.

Solutions for module forces are of the kind $\mathbf{F}_i^{\mathcal{M}_i} = [0, F_{yi}, F_{zi}]^\top$. Thus, each module force has two independent variables. Therefore, in order for rank($\mathbf{S}$) $\geq 6$ a condition of $n \geq 3$ needs to fulfilled. Furthermore, with the goal to guarantee full rank control of $\mathbf{S}$, both form configuration and module configuration are required. For instance, three modules in a line form configuration cannot be full rank. On the other hand, three modules that form a corner may or may not have full rank. Figure 3 shows eight of the same form configuration, but different module configurations, four that are singular and four not.

### C. Thrust and Decentralized Attitude Computation

In order to compute a modules' thrust and attitude, a similar approach described in [14] is used. The thrust of the $i$th flying vehicle can be computed using the following relation

$$F_i = \mathbf{F}_i^{*\mathcal{M}_i} \cdot \mathbf{R}_{\mathcal{Q}_i}^{\mathcal{M}_i}\mathbf{e_3}.$$

Later, $\mathbf{F}_i^{*\mathcal{M}_i}$ is used to compute a desired orientation of the $i$th flying vehicle, defined as $\mathbf{R}_{\mathcal{Q}_i}^{\mathcal{M}_i*} = \begin{bmatrix}\mathbf{b}_{i1}^* & \mathbf{b}_{i2}^* & \mathbf{b}_{i3}^*\end{bmatrix}$, through the following relations

$$\mathbf{b}_{i3}^* = \mathbf{F}_i^*/\|\mathbf{F}_i^*\|, \mathbf{b}_{i1} = [1,0,0]^\top, \mathbf{b}_{i2}^* = \mathbf{b}_{i3}^* \times \mathbf{b}_{i1}.$$

The roll angle is the only DOF to be individually set on each flying vehicle, thus $\phi_i^* = \arctan(b_{2iz}^*/b_{3iz}^*)$ is the desired roll angle computation. Unlikely previous work, we note that quadrotors only use two input commands: thrust $F_i$ and desired roll angle $\phi_i^*$.

### D. Attitude Controller and Actuator Distribution

In order to obtain actuator forces for each flying vehicle the angular accelerations are computed as $\ddot{\phi}_i = K_{p,\phi}(\phi_i^* - \phi_i) + K_{d,\dot{\phi}}(\dot{\phi}_i^* - \dot{\phi}_i)$, where $K_{p,\phi}$ and $K_{d,\dot{\phi}}$ are positive gains.

Therefore the $i$th thrust and $i$th moments around $x$ can be related to the $i$th control input as follows

$$\begin{bmatrix} F_i \\ M_{xi} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ y_{i1} & y_{i2} & y_{i3} & y_{i4} \end{bmatrix} \mathbf{u}_i, \tag{7}$$

where $M_{xi} = I_x \ddot{\phi}_i$ are flying vehicle moments around its $x$-axis and $\mathbf{u}_i = [f_{i1}, f_{i2}, f_{i3}, f_{i4}]^\top$. Applying a pseudo-inverse transformation to (7) we find the following control inputs

$$f_{ij} = \frac{F_i}{4} + \frac{M_{xi}}{4y_{ij}}$$

where $y_{ij}$ is the actuator j distance in $y$ from $\mathcal{Q}_i$. Because $y_{i1} = ... = y_{i4}$, the pseudo-inverse equally distributes actuator effort. Arriving at the actuator level distribution allows us now to control structures of flying modular robots along pre-defined trajectories. However, the question remains: what are the optimal structure configurations that are more suitable for a specific path.

## IV. Module Configuration Search

In this section, we propose to find optimal and near-optimal structure configurations for a given path and number of modules. More specifically we introduce a module configuration search that minimizes control effort along a pre-defined trajectory $\mathscr{T}$ and a pre-defined form configuration $\mathscr{F}_c(\mathbf{r}_1^{\mathcal{S}}, ..., \mathbf{r}_n^{\mathcal{S}})$. For a baseline, we developed an algorithm that finds optimal solutions by exploring all possible combinations in the search space. This *Exhaustive Search* checks all module configuration combinations, and has complexity exponential in $n$. In scenarios for which $n$ is large and multiple changes in the form configuration and trajectory are required, a faster method to compute module configurations is desirable. We propose a greedy approach that is near-optimal, but computationally efficient, called *SubGroup Search*. It first uses our Algorithm 2 to divide the structure into layers and create subgroups within each layer and according to the module's Euclidean distance to $\mathcal{S}$. Using this approach, the search applies to each subgroup instead of the whole structure search space, thus decreasing substantially the module configuration space to be explored.

For both *Exhaustive Search* and *SubGroup Search* methods the optimization problem can be defined as follows

$$\begin{aligned} \min_{\mathbf{k}} \quad & \int_0^{t_f} \|\mathbf{u}(t)\|_2 \, dt \\ \text{s.t.} \quad & \mathbf{k} \in \{0,1\}^n , \end{aligned}$$

where $\mathbf{k}$ embeds the binary module configuration constraints associated to all modules within the structure and $\mathbf{u} = [\mathbf{u}_1, \dots, \mathbf{u}_n]^\top$. As described in Section II and III each module is either perpendicular or parallel to one another, and these features are embedded in $\mathbf{k}$ for later computation of $\mathbf{R}_{\mathcal{M}_i}^{\mathcal{S}}(\mathbf{k}) \in \{\mathbf{R}_{z,k\frac{\pi}{2}} : k = 0, 1\}$ and consequently the structure matrix $\mathbf{S}(\mathbf{k})$. Therefore, both search methods outputs a vector $\mathbf{k}$ corresponding to modules configurations that minimizes control effort along a pre-defined trajectory.

**Algorithm 1:** ExhaustiveSearch

```
Input: form configuration $\mathscr{F}_c$, trajectory $\mathscr{T}$
Output: Optimal module configurations $\frac{\pi}{2}\mathbf{k}_{min}$
1  $e_{min} \leftarrow \infty$
2  $\mathbb{K} \leftarrow \{\text{BinaryVector}(q,n) | \, q = 1, ..., 2^n\}$
3  foreach $\mathbf{k} \in \mathbb{K}$ do
4      $\mathbf{S} \leftarrow \text{StructureConfiguration}(\mathscr{F}_c, \frac{\pi}{2}\mathbf{k})$
5      $\mathbf{u}(t) \leftarrow \text{SimulateTrajectory}(\mathbf{S}, \mathscr{T})$
6      $e \leftarrow \int_0^{t_f} \|\mathbf{u}(t)\|_2 \, dt$
7      if $e < e_{min}$ then
8          $e_{min} \leftarrow e$
9          $\mathbf{k}_{min} \leftarrow \mathbf{k}$
10 return $\frac{\pi}{2}\mathbf{k}_{min}$
```

**Algorithm 2:** CreateSubGroups

```
Input: form configuration $\mathscr{F}_c$
Output: SubGroups $\mathbb{G}$
1  $\mathbb{G} \leftarrow \emptyset$
2  $\mathbb{A} \leftarrow \{x_i \, | \, i = 1, ..., n\} \cup \{y_i \, | \, i = 1, ..., n\}$
3  $\mathbb{A} \leftarrow \text{SortUnique}(\mathbb{A})$
4  foreach $\ell \in \mathbb{A}$ do
5      $\mathbb{L}_\ell \leftarrow \{i | \, (x_i = \ell \wedge y \leq \ell) \vee (y_i = \ell \wedge y < \ell), i = 1...n\}$
6      $\mathbb{D} \leftarrow \{\|\mathbf{r}_i\| \, | \, i \in \mathbb{L}_\ell\}$
7      $\mathbb{D} \leftarrow \text{SortUnique}(\mathbb{D})$
8      foreach $d \in \mathbb{D}$ do
9          $\mathbb{G}_d \leftarrow \{i \, | \, \|\mathbf{r}_i\|_2 = d, \, i \in \mathbb{L}_\ell\}$
10         $\mathbb{G} \leftarrow \mathbb{G} \cup \{\mathbb{G}_d\}$
11 return $\mathbb{G}$
```

**Algorithm 3:** SubGroupSearch

```
Input: SubGroups $\mathbb{G}$, trajectory $\mathscr{T}$
Output: module configurations $\frac{\pi}{2}\mathbf{k}_{min}$
1  $e_{min} \leftarrow \infty$
2  $\mathbf{k} \leftarrow \mathbf{0}$
3  $\mathbf{k}_{min} \leftarrow \mathbf{k}$
4  foreach $\mathbb{G}_d \in \mathbb{G}$ do
5      $n_d \leftarrow |\mathbb{G}_d|$
6      for $k = 1$ to $2^{n_d}$ do
7          $\mathbf{k}[\mathbb{G}_d] \leftarrow \text{BinaryVector}(q, n_d)$
8          $\mathbf{S} \leftarrow \text{StructureConfiguration}(\mathbb{G}, \frac{\pi}{2}\mathbf{k})$
9          $\mathbf{u}(t) \leftarrow \text{SimulateTrajectory}(\mathbf{S}, \mathscr{T})$
10         $e \leftarrow \int_0^{t_f} \|\mathbf{u}(t)\|_2 \, dt$
11         if $e < e_{min}$ then
12             $e_{min} \leftarrow e$
13             $\mathbf{k}_{min} \leftarrow \mathbf{k}$
14 return $\frac{\pi}{2}\mathbf{k}_{min}$
```

### A. Exhaustive Search

Given a form configuration $\mathscr{F}_c$ and a specific trajectory $\mathscr{T}$, an approach that searches all possible module configurations

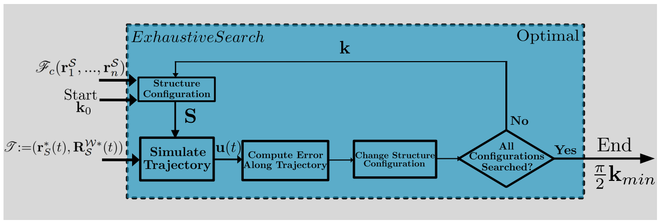
ExhaustiveSearch
Optimal
k
$\mathscr{F}_c(\mathbf{r}_1^S, ..., \mathbf{r}_n^S)$
Start
$\mathbf{k}_0$
Structure Configuration
S
No
$\mathscr{T} := (\mathbf{r}_S^*(t), \mathbf{R}_S^{W*}(t))$
Simulate Trajectory
u(t)
Compute Error Along Trajectory
Change Structure Configuration
All Configurations Searched?
Yes
End
$\frac{\pi}{2}\mathbf{k}_{min}$

Fig. 5. Exhaustive Search Flow Diagram. Search used for structures composed of small number of modules and yields optimal modules configurations.

is described in Algorithm 1 and illustrated in Fig. 5. This approach generates all possible binary combinations of $\mathbf{k}$. For each combination, a structure matrix $\mathbf{S}$ is computed and a simulation along $\mathscr{T}$ is performed. The *Exhaustive Search* returns optimal module configurations that minimizes control effort along the whole path.

Searching for all possible combinations is only feasible for a small number of modules. The number of configurations the algorithm has to check is $\mathcal{O}(2^n)$. Simulating each configuration for structures composed of a large number of modules would be computationally intractable.

## B. *CreateSubGroups Algorithm*

For the *SubGroup Search* method we first propose an algorithm that receives a form configuration $\mathscr{F}_c$ as an input, and outputs the subgroups $\mathbb{G}$. The motivation for creating subgroups comes with the goal to reduce the search space.

Figure 6 illustrates Algorithm 2 which creates subgroups for structure configurations. Initially, a set of distances to the $x$ and $y$-axis for all modules are created, then sorted from large to small and duplicate elements are removed in Lines 2-3. Those distances define the structure layers. For each distance to the axes, a set of modules that belong to layer $\mathbb{L}$ are created in Lines 4-5. Then each layer is divided in subgroups that share same Euclidean distance in $\mathcal{S}$. In this way, each subgroup $\cup\mathbb{G}_d$ has modules that belong to the same layer and have equal Euclidean distance to the center of mass.

## C. *SubGroup Search*

Given a trajectory $\mathscr{T}$ and subgroups $\mathbb{G}$ from Algorithm 2, we propose Algorithm 3 to find a near-optimal configuration, but reducing the search space. Using subgroups within a structure allow us to reduce the search space by only exploring all possible configurations within each subgroup (Fig. 7). The reasoning behind this approach comes from the *Modules Force Distribution* described in Section III-B. The *quadratic-programming* optimization results in modules located further in frame $\mathcal{S}$ to have larger impact in the structure behavior during flight. This characteristic is imposed by the *quadratic-programming* approach, since the minimization of the objective function in (6) yields force vectors of bigger magnitudes to modules located far from the structure center of mass. Therefore, we can conclude that these modules, in fact, produce bigger moments to the conglomerate compared to modules located closer to the center.

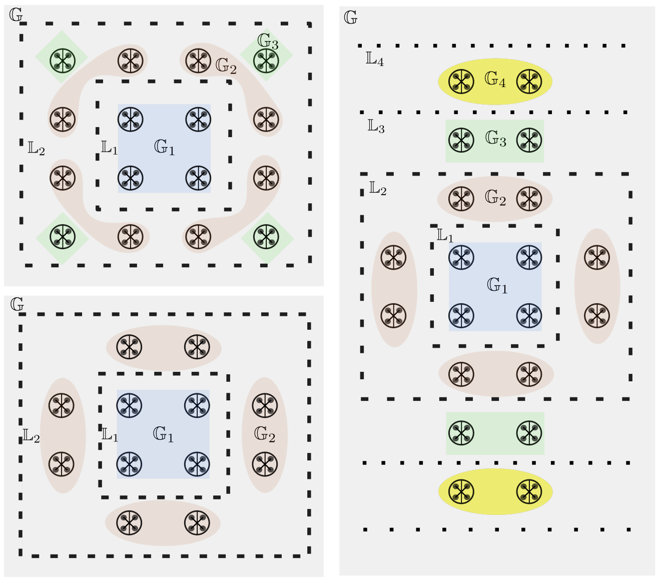

Fig. 6. Top view of three distinct expanded structure configurations and its subgroups. Two subgroups $\mathbb{G} = \mathbb{G}_1 \cup \mathbb{G}_2$ are created for a symmetric cross form configuration composed of 12 modules. Three subgroups $\mathbb{G} = \mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{G}_3$ are created for a square form configuration composed of 20 modules. Four subgroups $\mathbb{G} = \mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{G}_3 \cup \mathbb{G}_4$ are created for an asymmetrical cross form configuration composed of 20 modules.

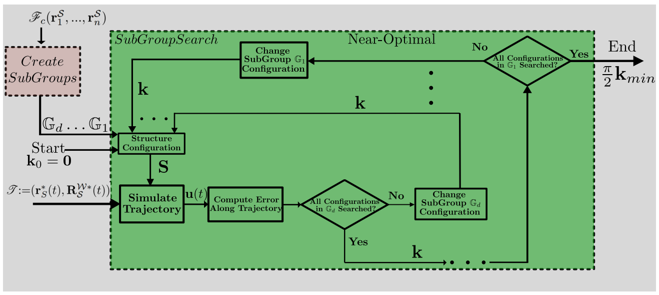
$\mathscr{F}_c(\mathbf{r}_1^S, ..., \mathbf{r}_n^S)$
Create SubGroups
SubGroupSearch
Near-Optimal
Change SubGroup $\mathbb{G}_1$ Configuration
No
All Configurations in $\mathbb{G}_1$ Searched?
Yes
End
$\frac{\pi}{2}\mathbf{k}_{min}$
k
k
$\mathbb{G}_d \dots \mathbb{G}_1$
Start
$\mathbf{k}_0 = \mathbf{0}$
Structure Configuration
S
$\mathscr{T} := (\mathbf{r}_S^*(t), \mathbf{R}_S^{W*}(t))$
Simulate Trajectory
u(t)
Compute Error Along Trajectory
All Configurations in $\mathbb{G}_d$ Searched?
No
Change SubGroup $\mathbb{G}_d$ Configuration
Yes
k

Fig. 7. SubGroup Search Flow Diagram. Search used for structures of large number of modules and yields near-optimal modules configurations.

We want to take advantage of modules with highest performance impact and with this in mind we choose to optimize the structure configuration in dependent sequential steps by using subgroups. Because we are not searching the whole structure configuration space anymore, we need to define an initial search point for Algorithm 3 to start. The specific singular configuration $\mathbf{k} = \mathbf{0}$, is set to be our initial structure configuration in the search, once generically singular configurations are poor solutions for an arbitrary problem. The initial search point is set in Line 2, then search initiates at subgroup $\mathbb{G}_d$ which is located at the structure edges. Then all possible module configurations ($\mathbf{k}[\mathbb{G}_d]$) within that subgroup are computed. It is important to notice that module configurations from other subgroups are untouched and the method keeps the configuration that minimized control effort by only exploring subgroup $\mathbb{G}_d$. Because subgroup $\mathbb{G}_d$ is located at the structure edge, it is then the subgroup that contains modules that could contribute the most for the structure moments control. This is why subgroup $\mathbb{G}_d$ is chosen to be optimized at first, once the idea is to obtain the maximum in terms of performance from its modules. Next, the algorithm moves to subgroup $\mathbb{G}_{d-1}$ and exploration of all module configurations within that subgroup takes place. We would like to highlight that at this level the optimization of subgroup $\mathbb{G}_{d-1}$, uses as an additional information the optimized configuration obtained for subgroup $\mathbb{G}_d$ as well as the remaining configuration states of the untouched subgroups. In other other words, we can say that for each subgroup optimization we need to know the full structure configuration, even though search takes place locally within each subgroup. With this reasoning, the algorithm keeps moving inwards within the structure, optimizing in a sequential manner each subgroup, until subgroup $\mathbb{G}_1$ is reached.

*Number of configurations:* In Algorithm 3, the number of configurations to check is the most important factor to analyze as simulation in Line 9 is the most computationally expensive step. The for-loop in Line 6 is a critical point that can lead to an exponential number of configurations to check. However, the size of $\mathbb{G}_d$ is guaranteed to be always smaller or equal to eight by applying Algorithm 2. Since $\mathbb{G}_d \subset \mathbb{L}_\ell$, the number of elements in $\mathbb{G}_d$ is constrained by the layer with the largest number of modules. The worst case scenario is a square structure of $v \times v$ modules, its last layer has a square shape with $4(v-1)$ modules. In this layer, Line 9 restricts $\mathbb{G}_d$ to hold only modules that share the exact same Euclidean distance to $\mathcal{S}$. Because of the discretized nature of the grid, alignment of the modules and configurations are chosen with only two lines of symmetry, there can be at most eight modules in $\mathbb{G}_d$. Finally, the main foreach-loop in Line 5 is called at most $n$ times. All the elements in $\mathbb{G}$ are disjoint sets of modules, therefore, the number of groups in $\mathbb{G}$ has to be smaller or equal than $n$. This leads us to conclude that the number of configurations to be checked and simulated are $\mathcal{O}(n)$. Fig. 8 shows the number of configurations to be checked by the *Exhaustive Search* algorithm and the *SubGroup Search* algorithm. The baseline does not depend on the form configuration of the structure since it checks every possible combination. In contrast, the *SubGroup Search* is form configuration dependent to define subgroups on the layers. For two scalable structures, a square and a line, we can see that the square grows at a higher rate than the line configuration, but both cases are linear with respect to the number of modules $n$.

## V. Simulation

We evaluated the scalability and performance of our proposed methods in a developed MATLAB simulator. Based on the given $\mathscr{F}_c$ and $\mathscr{T}$, we execute our algorithms to find optimal and near-optimal configurations. Our control model assumes that quadrotors can generate instantaneous forces along the module $y$-axis. Our simulator includes quadrotors inertia around its $x$-axis for the system dynamics and our controller arrives at the actuator level of the flying vehicle. This makes the simulator more realistic and also shows that for small flying vehicles our controller assumption applies.

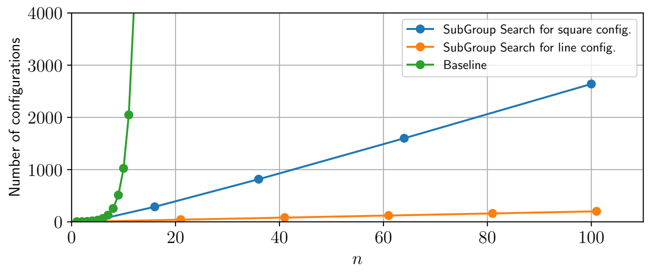

Fig. 8. Number of configurations to be evaluated by the Exhaustive Search (Baseline) algorithm and the SubGroup Search algorithm.

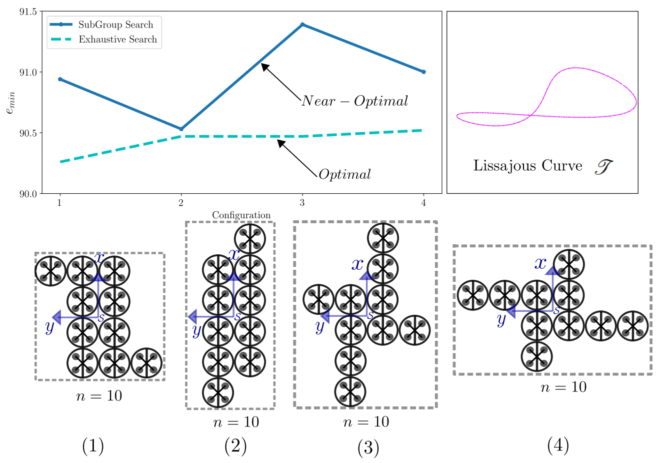

Fig. 9. Exhaustive Search $e_{min}$ compared to the SubGroup Search $e_{min}$ applied to distinct form configurations of $n = 10$, and for a given Lissajous curve $\mathscr{T}$. Four distinct structure configurations composed of 10 modules are tested using both search methods.

Figure 9 shows a performance comparison between the *Exhaustive Search* and *SubGroup Search*. The plot shows the accumulated error along a trajectory illustrating performances of both searches for distinct form configurations composed by the same number of modules. The plot comparison is made for structures of 10 modules. For $n \geq 10$ the *Exhaustive Search* starts to become impractical. The number of configurations to be checked by this method is $2^{10}$ for $n = 10$ whereas the *SubGroup Search* reduces it to $2^4 + 2^4 + 2^2$. For the *SubGroup Search* the number of configuration scales linearly with $n$, although we cannot say that the algorithm time complexity scales at the same rate. In Algorithm 3, the necessity to recompute a large configuration matrix $\mathbf{S} \in \mathbb{R}^{6\times 3n}$ and its pseudo-inverse at each time step causes the algorithm time complexity to grow at a faster rate even though the search space reduction is preserved.

With the *SubGroup Search* producing near-optimal results, we can then simulate larger number of modules. A structure composed of 36 modules was tested along a Lissajous curve trajectory. For this trajectory $\mathbf{R}_{\mathcal{S}}^{\mathcal{W}*}(t)$ is always an identity matrix, therefore the desirable structure orientation along the whole path is parallel to the $xy$ plane. Fig. 10 shows the structure flight behavior for those conditions. Very small deviations for the orientation can be verified.

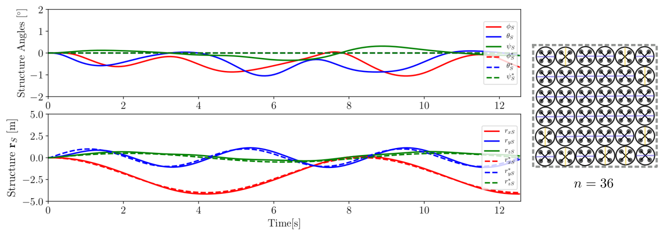

Fig. 10. 36 modules in a square form configuration performing a flight along a Lissajous curve $\mathscr{T}$. The Subgroup Search algorithm was utilized to generate the illustrated modules configuration for the given structure.

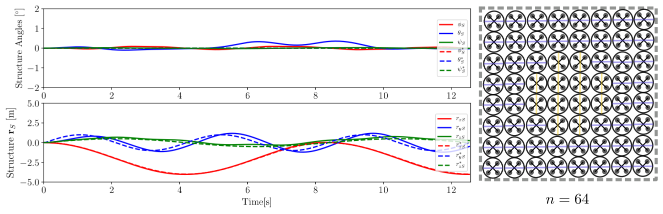

Fig. 11. 64 modules in a square form configuration performing a flight along a Lissajous curve $\mathscr{T}$. The Subgroup Search algorithm was utilized to generate the illustrated modules configuration for the given structure.

Along the same lines the desired position is tracked in a satisfactory manner. It is important to note that whenever our algorithm cannot find feasible forces for a desired wrench along a trajectory, the simulator stops and switches to the next possible modules configuration. This feature speeds up our algorithm in the process of finding near-optimal structure configurations. Similarly, simulations for 64 modules (Fig. 11) in a square form configuration demonstrates the dynamic response of the structure with the proposed $SE(3)$ controller. The dynamic response for 36 and 64 modules in a square form configuration not only demonstrates the validity of the controller approach proposed, in fact it also shows that the modules configuration generated by the *SubGroup Search* are controllable and yields satisfactory solutions.

Modules configurations can then be computed in a faster manner even for flying structures composed of a large number of modules.

## VI. Conclusion and Future Work

In this work a configuration generalization of the system presented in [14] is introduced. A flying structure controller that takes as inputs trajectories in $SE(3)$ was presented along two configuration optimization search methods. We demonstrated through simulation results that the *Exhaustive Search* yields optimal module configurations utilizing a costly search method whereas the *SubGroup Search* yields near-optimal results though the configuration space to be explored was substantially reduced. The results obtained from the proposed search method shows an important contribution for computing configurations for flying modular structures composed of a large number of modules.

In future work, we aim to extend this study to the form configuration level of the structure. Optimization of both module and form configuration would allow us to design flying modular robots capable to perform a diverse range of tasks by self-reconfiguration. Furthermore, we plan to add constraints to our controller to handle scenarios in which forces that keeps modules docked together are limited.

## References

[1] D. Saldaña, B. Gabrich, G. Li, M. Yim, and V. Kumar, “Modquad: The flying modular structure that self-assembles in midair,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 691–698.

[2] G. Li, B. Gabrich, D. Saldaña, J. Das, V. Kumar, and M. Yim, “Modquad-vi: A vision-based self-assembling modular quadrotor,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 346–352.

[3] R. Oung and R. D’Andrea, “The distributed flight array,” *Mechatronics*, vol. 21, no. 6, pp. 908–917, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.mechatronics.2010.08.003

[4] D. Saldaña, P. M. Gupta, and V. Kumar, “Design and control of aerial modules for inflight self-disassembly,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3410–3417, 2019.

[5] N. Gandhi, D. Saldana, V. Kumar, and L. T. X. Phan, “Self-reconfiguration in response to faults in modular aerial systems,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2522–2529, 2020.

[6] M. Yim, D. G. Duff, and K. D. Roufas, “Polybot: a modular reconfigurable robot,” in *ICRA*, 2000, pp. 514–520.

[7] K. Stoy, D. Brandt, D. J. Christensen, and D. Brandt, *Self-reconfigurable robots: an introduction*. MIT press Cambridge, 2010.

[8] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, and E. Klavins, “Modular Self-reconfigurable Robot Systems: Challenges and Opportunities for the Future,” {*IEEE*} *Robotics \& Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[9] J. Daudelin, G. Jing, T. Tosun, M. Yim, H. Kress-Gazit, and M. Campbell, “An integrated system for perception-driven autonomy with modular robots,” *Science Robotics*, vol. 3, no. 23, 2018.

[10] I. O’Hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, J. Seo, M. Turpin, V. Kumar, and M. Yim, “Self-assembly of a swarm of autonomous boats into floating structures,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1234–1240, 2014.

[11] B. Gabrich, D. Saldaña, V. Kumar, and M. Yim, “A flying gripper based on cuboid modular robots,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7024–7030.

[12] M. Zhao, K. Kawasaki, X. Chen, Y. Kakiuchi, K. Okada, and M. Inaba, “Transformable multirotor with two-dimensional multilinks: Modeling, control, and whole-body aerial manipulation,” in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 515–524.

[13] M. Zhao, T. Anzai, F. Shi, X. Chen, K. Okada, and M. Inaba, “Design, modeling, and control of an aerial robot dragon: A dual-rotor-embedded multilink robot with the ability of multi-degree-of-freedom aerial transformation,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1176–1183, 2018.

[14] B. Gabrich, G. Li, and M. Yim, “Modquad-dof: A novel yaw actuation for modular quadrotors,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8267–8273.

[15] M. Ryll, H. H. Bülthoff, and P. R. Giordano, “Modeling and control of a quadrotor uav with tilting propellers,” in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4606–4613.

[16] M. Ryll, D. Bicego, and A. Franchi, “Modeling and control of fast-hex: A fully-actuated by synchronized-tilting hexarotor,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1689–1694.

[17] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi, “6d physical interaction with a fully actuated aerial robot,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5190–5195.

[18] H.-N. Nguyen, S. Park, J. Park, and D. Lee, “A novel robotic platform for aerial manipulation using quadrotors as rotating thrust generators,” *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 353–369, 2018.

[19] T. Lee, M. Leoky, and N. McClamroch, “Geometric tracking control of a quadrotor UAV on SE(3),” *49th IEEE Conference on Decision and Control (CDC), 2010.*, pp. 5420–5425, 2010.