# TransforMARS: Fault-Tolerant Self-Reconfiguration for Arbitrarily Shaped Modular Aerial Robot Systems

Rui Huang, Zhiyu Gao, Siyu Tang, Jialin Zhang, Lei He, Ziqian Zhang, Lin Zhao

***Abstract*— Modular Aerial Robot Systems (MARS) consist of multiple drone modules that are physically bound together to form a single structure for flight. Exploiting structural redundancy, MARS can be reconfigured into different formations to mitigate unit or rotor failures and maintain stable flight. Prior work on MARS self-reconfiguration has solely focused on maximizing controllability margins to tolerate a single rotor or unit fault for rectangular-shaped MARS. We propose TransforMARS, a general fault-tolerant reconfiguration framework that transforms arbitrarily shaped MARS under multiple rotor and unit faults while ensuring continuous in-air stability. Specifically, we develop algorithms to first identify and construct minimum controllable assemblies containing faulty units. We then plan feasible disassembly-assembly sequences to transport MARS units or subassemblies to form target configuration. Our approach enables more flexible and practical feasible reconfiguration. We validate TransforMARS in challenging arbitrarily shaped MARS configurations, demonstrating substantial improvements over prior works in both the capacity of handling diverse configurations and the number of faults tolerated. The videos and source code of this work are available at the anonymous repository: `https://anonymous.4open.science/r/TransforMARS-1030/`**

## I. Introduction

Modular Aerial Robotic Systems (MARS) serve as flexible and adaptive agents in diverse flight tasks including searching, rescuing, and transportation. MARS can respond to dynamical environment changes through disassembly and reassembly. Research on MARS has developed mid-air docking [1]–[3], in-flight separation [4]–[7], self-reconfiguration algorithms [8]–[10], structural optimization [11]–[13], trajectory tracking [14], [15], fault-tolerance control [16], [17] and collision-free path planning [17] to improve robustness and flexibility. However, compared with ground modular robots [18]–[21], aerial platforms are more severely affected by faults, which can severely degrade control performance if not properly mitigated.

To address this challenge, prior studies [9], [10] have shown that self-reconfiguration can significantly improve trajectory-tracking performance after faults occur. Specifically, prior work [9] proposed a self-reconfiguration strategy to enhance fault-tolerant control under rotor-level failures, thereby maintaining mission continuity. The approach formulates a mixed-integer linear program that selects module placements to mitigate fault effects. However, the scheme has clear limitations. As illustrated in Fig.1(a), it primarily targets partial faults (e.g., a single failed rotor in a quadcopter) by attaching a functional module to the impaired unit (Step 2), and it does not handle complete unit failure. Moreover, it does not explicitly account for uncontrollable factors, and its performance depends on both the fidelity of the simulator's physics engine and the selected objective function for arranging the position of the faulty unit. Subsequent work [10] narrows this gap by deriving a control-constrained dynamic model of MARS and proposing a robust self-reconfiguration algorithm that maximizes the controllability margin (CM) at each intermediate reconfiguration step. In particular, it computes a minimally controllable subassembly (MCS) that encompasses the faulty unit to enable stable in-air

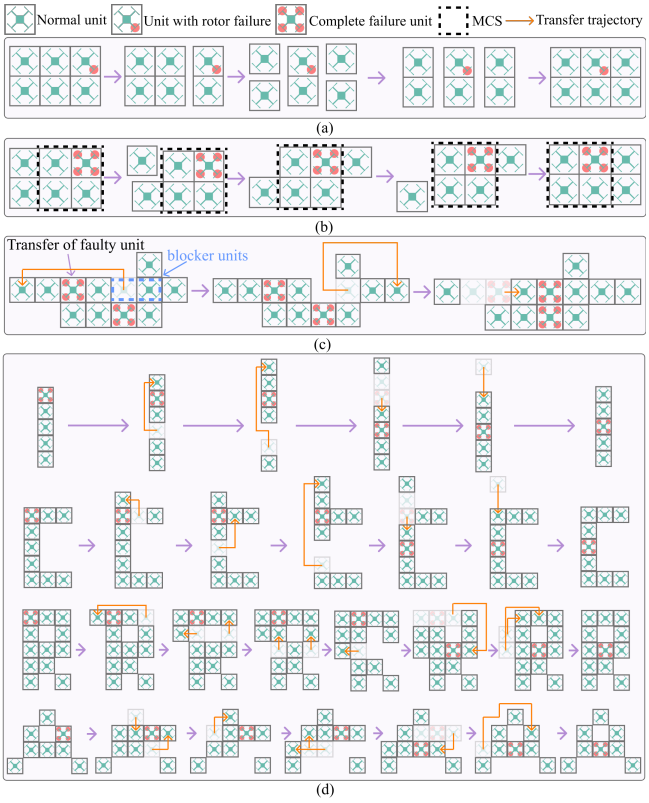


Fig. 1. Self-reconfiguration of $3 \times 2$ and arbitrary assemblies. (a) Prior work [9] attaches a functional unit adjacent to a faulty one, addressing only rotor-level failure. (b) Subsequent work [10] optimized controllability and extended the approach to single-unit faults, but remained limited to rectangular configurations. (c) Proposed TransforMARS accommodates multiple unit faults in arbitrary configurations; in the first two steps, normal units are relocated to create vacancies for the minimum controllable subassembly containing the faulty units. (d) TransforMARS reconfigures in ICRA letter-shaped formations, demonstrating flexibility under faulty units.

Rui Huang, Siyu Tang, Jialin Zhang, Lei He, Ziqian Zhang and Lin Zhao are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (Email: ruihuang@u.nus.edu, e1352616@u.nus.edu, jialinzhang@u.nus.edu, lei.he@nus.edu.sg, e1546905@u.nus.edu, elezhli@nus.edu.sg). Zhiyu Gao is with the Department of Mechanical Engineering, National University of Singapore, Singapore 117583 (Email: e1553948@u.nus.edu).

transport, and then plans control-feasible disassembly and assembly sequences for robust self-reconfiguration, illustrated in Fig. 1(b).

Nevertheless, [10] still exhibits two key limitations: (i) *limited generalization*–it assumes only a single faulty unit and confines the analysis to standard rectangular configurations, so it cannot scale to multiple faults or arbitrary configurations; and (ii) *lack of path planning*–it omits explicit collision-aware motion planning and conflict-free assembly during self-reconfiguration. Consequently, the routing of units fails to generalize across configurations.

To bridge the gap, we propose TransforMARS, a general fault-tolerant self-reconfiguration framework that also plans disassemble and reassemble motions with controllability guarantees. Unlike prior methods, our approach handles multiple faulty units and generalizes to arbitrary configurations. See Figs. 1(c) and (d) for examples. In particular, Steps 1–2 of Fig. 1(c) first relocate the normal units that may obstruct the trajectory of the faulty unit, which ensures that the faulty unit can be transferred to desired position without conflict in Step 3. More specifically, our contributions are summarized as follows:

1) We propose a general self-reconfiguration planner for arbitrarily shaped MARS with multiple faulty units. For each faulty unit, the method **first** constructs a controllable subassembly that includes the faulty unit for transfer. Unlike prior approaches which build the MCS by selecting normal units directly from the original configuration without relocating them (black and blue dashed boxes in Fig. 2), our method flexibly transfers normal units that are not directly connected to the faulty unit and reassembles them into a MCS identified in the target configuration. **Second**, to ensure every MCS can reach its designated position, we identify and remove potential obstructions ("blocker units") along their transfer trajectories (see the first two steps in Fig. 1(c) and Step 2 in Fig. 2(c), Ours). This stage is called the path-clearance strategy. Once the trajectories are unobstructed, the MCS units are moved to their target positions. **Finally**, to prevent blockages caused by an incorrect assembly order, such as earlier units obstructing later ones, we compute a *conflict-free assembly sequence*, ensuring that the remaining normal units are relocated to the vacant positions without interference.
2) Extensive experiments show that our method can maintain the controllability of intermediate subassemblies and reconfigure with substantially fewer assembly and disassembly steps compared with [9]. In contrast to [10], our algorithm extends self-reconfiguration from single-fault to multi-fault scenarios and supports arbitrary, irregular configurations. We further validate the approach on irregular shapes in CoppeliaSim [22] simulation and execute the planned sequences on real quadrotors using Crazyswarm [23], demonstrating its feasibility.

## II. Preliminaries

We assume that all units in MARS are homogeneous, each with a unit mass $m$. Let $n$ denote the total number of drone units. We denote the overall control input of MARS as $\boldsymbol{u}_f = [T\ \tau_x\ \tau_y\ \tau_z]^{\mathrm{T}}$, where $T$ is the collective thrust generated by all units, and $\tau_x$, $\tau_y$, and $\tau_z$ are the collective torques along the $x$-, $y$-, and $z$-axes, respectively. The control input $\boldsymbol{u}_f$ is bounded by the maximum limit $\boldsymbol{u}_{\max} = [T_{\max}\ \tau_{x,\max}\ \tau_{y,\max}\ \tau_{z,\max}]^{\mathrm{T}}$ and the lower bound $\boldsymbol{u}_0 = [0\ 0\ 0\ 0]^{\mathrm{T}}$. Accordingly, we define the feasible control input set as $\boldsymbol{\Omega} = \{\boldsymbol{u} \mid \boldsymbol{u}_0 \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}\}$, which includes all possible control inputs that MARS can provide. When all units function properly, the system can hover in the air. However, under certain failure scenarios, specific configurations may become uncontrollable if the remaining normal units cannot compensate for the lost thrust and torque contributed by faulty units. To quantify controllability, we adopt the CM defined in [10], whose formulation is given as follows. For set $\boldsymbol{\Omega}$ and an arbitrary point $\boldsymbol{\alpha}$, we first define the following function $\zeta$ as an indicator:

$$\zeta(\boldsymbol{\alpha}, \partial\boldsymbol{\Omega}) \triangleq \begin{cases} \min\limits_{\boldsymbol{\beta}\in\partial\boldsymbol{\Omega}} \|\boldsymbol{\alpha}-\boldsymbol{\beta}\|, & \text{if } \boldsymbol{\alpha}\in\boldsymbol{\Omega} \\ -\min\limits_{\boldsymbol{\beta}\in\partial\boldsymbol{\Omega}} \|\boldsymbol{\alpha}-\boldsymbol{\beta}\|, & \text{if } \boldsymbol{\alpha}\in\boldsymbol{\Omega}^{\mathrm{C}} \end{cases} \tag{1}$$

where $\partial\boldsymbol{\Omega}$ denotes the boundary of the set $\boldsymbol{\Omega}$, and $\boldsymbol{\Omega}^C$ its complement. In essence, $\zeta(\boldsymbol{\alpha}, \partial\boldsymbol{\Omega})$ represents the signed minimum distance from the point $\boldsymbol{\alpha}$ to the boundary $\partial\boldsymbol{\Omega}$, with the sign determined by the membership of $\boldsymbol{\alpha}$. Specifically, the distance is positive if $\boldsymbol{\alpha}$ lies inside $\boldsymbol{\Omega}$, zero if it lies on the boundary, and non-positive otherwise.

For our problem, $\boldsymbol{\Omega}$ is exactly the feasible control input set of MARS under failures. Let $\boldsymbol{g} = [nmg\ 0\ 0\ 0]^{\mathrm{T}}$ (with $g$ being the gravitational acceleration). If $\boldsymbol{g}$ is an *interior point* of $\boldsymbol{\Omega}$, then the CM takes a positive value given by

$$\zeta(\boldsymbol{g}, \partial\boldsymbol{\Omega}) = \min_{\boldsymbol{u}_f\in\partial\boldsymbol{\Omega}} \|\boldsymbol{g} - \boldsymbol{u}_f\| > 0, \tag{2}$$

which indicates that the available control authority is sufficient to counteract gravity and stabilize the orientation, allowing MARS to continue hovering even after a failure. In contrast, when $\zeta(\boldsymbol{g}, \partial\boldsymbol{\Omega}) < 0$, the system becomes uncontrollable. Moreover, a larger value of $\zeta(\boldsymbol{g}, \partial\boldsymbol{\Omega})$ corresponds to greater control authority, providing a straightforward and effective metric for evaluating the control performance of the system.

## III. Generalized Self-Reconfiguration of MARS

In this section, we present a generalized self-reconfiguration method applicable to arbitrarily shaped configurations with multiple faulty units. First, **Algorithm 1** determines the minimum controllable subassembly (MCS) for each faulty unit, ensuring that faulty units are not isolated during the reconfiguration process. Next, **Algorithm 2** plans path clearance and transfer for the MCS, along with a conflict-free assembly sequence that transfers all normal units to the target configuration while preventing blockage during the reconfiguration process.

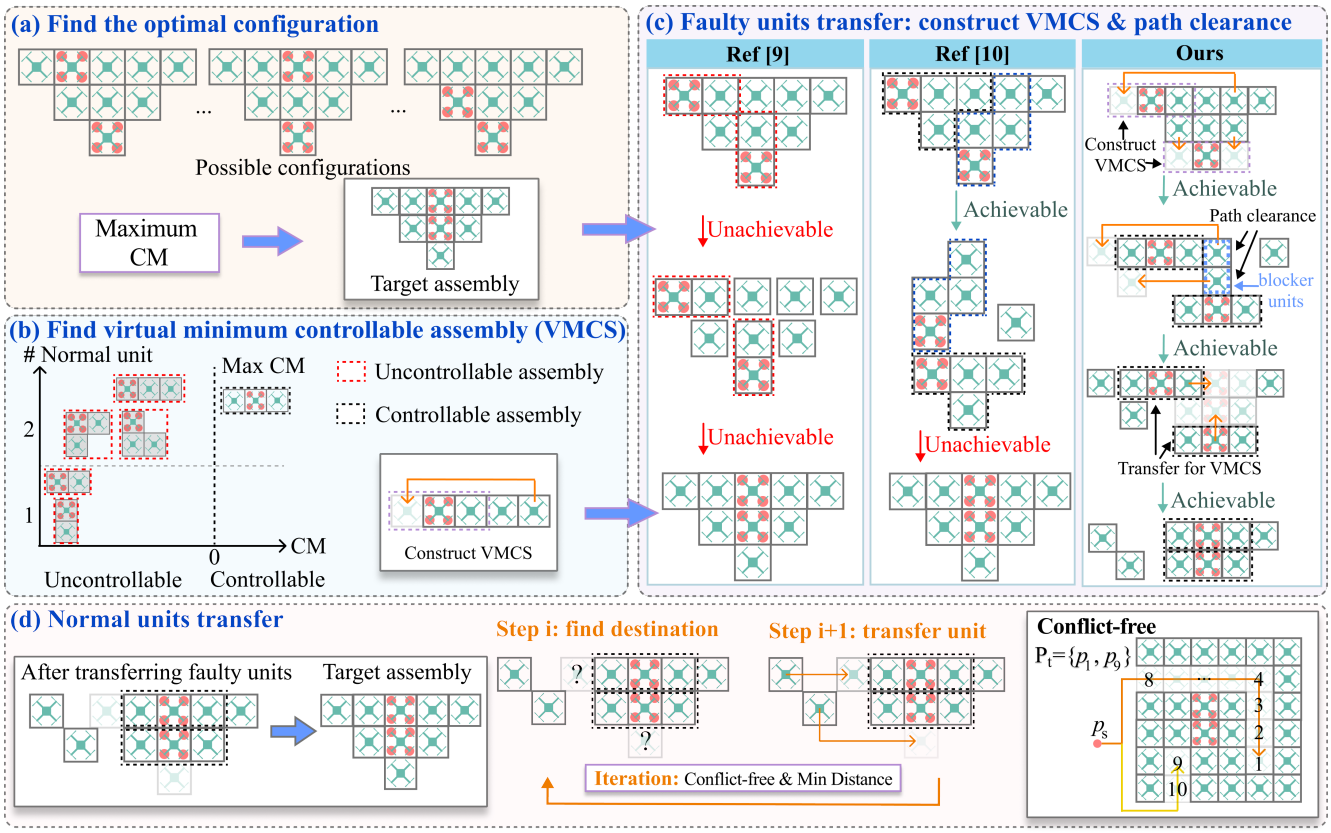


Fig. 2. Framework of TransforMARS. (a) The configuration with the maximum CM value is computed as the target assembly. (b) At each step, a normal unit is connected to the faulty unit until the CM becomes positive, and the VMCS is chosen as the configuration with the maximum CM under the same normal unit. (c) The VMCS is then constructed by relocating suitable normal units, followed by path clearance to ensure all VMCS units can reach their designated positions. (d) The remaining vacant positions are sequentially filled by selecting non-conflicting destinations and transferring the nearest units, repeated until reconfiguration is complete.

### A. Definition

Let $p_i = (x_i, y_i)$ denote the coordinate of the $i$-th unit in the MARS $\mathcal{P}$, and let $n_f$ be the number of faulty units, with $\mathcal{F}$ denoting the set of faulty units. During the self-reconfiguration process, the single assembled MARS can be decomposed into $n_p$ subassemblies. We denote their collection by

$$\mathcal{P} = \{p_1, p_2, \ldots, p_n\}, \tag{3a}$$

$$\pi_0\big(G[\mathcal{P}]\big) = \{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_{n_p}\}, \tag{3b}$$

$$\mathcal{S}_i = \bigcup_j \{p_{i,j}\}, i \in \{1, 2, \cdots, n_p\}, \tag{3c}$$

where, $G[\mathcal{P}]$ denotes the grid graph induced by the set of occupied cells $\mathcal{P}$, where edges represent adjacency (i.e., 4-neighbor connectivity). The operator $\pi_0(\cdot)$ returns the set of connected components of a graph. Thus, $\pi_0(G[\mathcal{P}]) = \{\mathcal{S}_1, \cdots, \mathcal{S}_{n_p}\}$ is the partition of $\mathcal{P}$ into $n_p$ disjoint subassemblies, each $\mathcal{S}_i$ consisting of units $p_{i,j}$. We denote by $\mathcal{D}_i$ the subset of faulty units in $\mathcal{S}_i$. Accordingly, a MARS as a whole or any of its subassemblies can be fully described by the pair $(\mathcal{P}, \mathcal{F})$ or $(\mathcal{S}_i, \mathcal{D}_i)$, respectively.

For subassemblies that contain at least one faulty unit, their CMs are computed using (1) and the smallest value among them is employed to measure the system-level CM. Specifically, suppose $n_d$ subassemblies contain faulty units; the overall CM of MARS is defined as

$$CM_{(\mathcal{P},\mathcal{F})} = \min_{i=1,2,\cdots,n_d} \zeta(\mathbf{g}_i, \partial\boldsymbol{\Omega}_i)\,, \tag{4}$$

where $\boldsymbol{\Omega}_i$ is the feasible control input set of the $i$-th faulty subassembly, $\mathbf{g}_i$ is its total weight. The indicator $\zeta(\mathbf{g}_i, \partial\boldsymbol{\Omega}_i)$ denotes its CM value as computed in (1). In the following sections, we use $CM_{(\mathcal{P},\mathcal{F})}$ to evaluate MARS during reconfigurations.

### B. Find Optimal Configuration and VMCS

We first determine the *optimal configuration* $(\mathcal{P}^*, \mathcal{F}^*)$, defined as the configuration that maximizes the CM among all possible arrangements with the same number of faulty and normal units as illustrated in Fig. 2(a). Prior work [9] transfers faulty units by attaching only a single functional unit, which yielded a negative CM and is practically infeasible (Fig. 2(c)-**Ref. [9]**). In general, each faulty unit must be bolstered by multiple normal units to form a flyable subassembly, termed the *Minimum Controllable Subassembly* (MCS) [10] (Fig. 2(c)-**Ref. [10]**). However, [10] identifies the MCS only from the units already attached to the faulty unit, which is conservative. In contrast, we consider all available normal units in the MARS when constructing the MCS, including those not directly attached to the faulty unit (see Fig. 2(c)-**Ours**, for example). We refer to this more flexible and general construction as the *Virtual Minimum*

**Algorithm 1** VMCS Identification Algorithm

**Input:** $\mathcal{P}, \mathcal{F}, k = 1, CM^* = -\inf$
**Output:** VMCS $\mathcal{P}^\dagger, \mathcal{F}^\dagger$
**while** $CM^* < 0$ **do**
  $\mathbf{C}_k \leftarrow$ the set of all possible rigidly connected assemblies $\{\mathcal{P}', \mathcal{F}'\}$ with $k$ normal units and the faulty unit.
  $CM^* \leftarrow \max\{CM_{(\mathcal{P}',\mathcal{F}')}$ for $\forall(\mathcal{P}', \mathcal{F}') \in \mathbf{C}_k\}$
  $k \leftarrow k + 1$
**end while**
**Return** $\mathcal{P}^\dagger, \mathcal{F}^\dagger$

*Controllable Subassembly* (**VMCS**). The detailed procedure is presented in Algorithm 1, and an example calculation is illustrated in Fig. 2(b).

Specifically, if a faulty unit is already connected with sufficient normal units to form a VMCS ($\mathcal{P}^\dagger, \mathcal{F}^\dagger$), no additional disassembly or assembly is required. Otherwise, when a VMCS cannot be found for a faulty unit in the original configuration, i.e., vacant position $p^*$ exists, we detach a normal unit $p_{i,j}$ and reassemble it with the faulty one, as shown in Fig. 2(b) and in the Fig. 2(c)-**Ours**. To find the optimal normal unit to be detached, we formulate the following disassembly optimization problem that jointly considers the CM and the path length:

$$\min_{p_{i,j} \notin \mathcal{F}} \; c_1 \Delta_{CM}^2 - c_2 L(p_{i,j}, p^*), \tag{5a}$$

$$\Delta_{CM} = CM_{(\mathcal{P}_\mathcal{X}, \mathcal{F}_\mathcal{X})} - CM_{(\mathcal{P}^*, \mathcal{F}^*)}, \tag{5b}$$

where $(\mathcal{P}_\mathcal{X}, \mathcal{F}_\mathcal{X})$ denotes the MARS configuration after detaching unit $p_{i,j}$, $c_1$ and $c_2$ are weight parameters balancing the CM deviation and the path length, respectively. Here, $L(p_{i,j}, p^*)$ denotes the path length computed using the $A^*$ algorithm for moving the $j$-th unit in the $i$-th subassembly from its current position $p_{i,j}$ to the destination $p^*$.

### C. Faulty Units Transfer: Path Clearance and Transfer for VMCS

Some normal units may obstruct the transfer of VMCS, as illustrated in the Fig. 2(c)-**Ours**. In such cases, the first two steps relocate these normal units to alternative positions, enabling the VMCS to reach its designated destination. We define such interfering units as *blocker units* if they (i) occupy a target position assigned to a VMCS or (ii) lie along a VMCS's transfer trajectory. To ensure successful VMCS transfer, these blocker units must temporarily vacate their current positions. Instead of relocating blocker units to arbitrary or manually specified positions as in [9], we move them to a set of temporary *waiting positions* $\mathcal{W} = \{w_1, w_2, \ldots, w_q\}$, selected from the vacant positions in the target configuration $\mathcal{P}^*$ that are not part of any VMCS transfer path. Formally, $\mathcal{W} = \{w \mid w \in \mathcal{P}^* \setminus ((\mathcal{P} \cap \mathcal{P}^*) \cup T)\}$, where $T$ denotes the set of transfer path coordinates. This strategy, which is termed the *path-clearance relocation rule*, reduces redundant moves by ensuring that vacated units remain close to their final target positions. As detailed in **Algorithm 2**, Lines 14–25, each blocker unit is first moved to its nearest waiting position $w^* \in \mathcal{W}$ (Line 21). Once these paths are cleared, the VMCS units proceed to their designated targets.

**Algorithm 2** TransforMARS Algorithm

1: **Input:** $\mathcal{P}, \mathcal{F}$
2: **Output**: $\mathcal{P}, \mathcal{F}$
3: $\mathcal{P}^*, \mathcal{F}^* \leftarrow$ Compute optimal configuration for $(\mathcal{P}, \mathcal{F})$ [10]
4: $\mathcal{P}^\dagger, \mathcal{F}^\dagger \leftarrow$ Compute VMCS for all faulty units in $\mathcal{F}$
5: **for** $p_f \in \mathcal{F}$ **do** ▷ Find VMCS
6:  **if** cannot find adjacent normal units to form VMCS **then**
7:  **for** all vacant position $p^*$ in VMCS **do**
8:  $p_{i,j} =$ optimization$(\mathcal{P}, \mathcal{F}, \mathcal{P}^*, \mathcal{F}^*, p^*)$ (5a)
9:  transfer $p_{i,j}$ to $p^*$ using trajectory generated by $A^*$
10:  update $\mathcal{P}, \mathcal{F}$
11:  **end for**
12:  **end if**
13: **end for**
14: **for** $i$-th VMCS **do** ▷ Path clearance for VMCS
15:  Compute trajectory $T_i$ of transfering $i$-th VMCS s.t. faulty unit reaching position $p_f \in \mathcal{F}^*$
16: **end for**
17: $T \leftarrow T_1 \cup T_2 \cup \cdots \cup T_{n_f}$
18: **for** blocker unit $j$ staying in trajectory $T$ **do**
19:  waiting position set $\mathcal{W} = \{w \mid w \in \mathcal{P}^* \setminus (\mathcal{P} \cap \mathcal{P}^* \cup T)\}$
20:  $w^* = \arg\min_{w \in \mathcal{W}} L(w, p_j)$
21:  transfer $p_j$ to $w^*$ using trajectory generated by $A^*$
22:  update $\mathcal{P}, \mathcal{F}$
23: **end for**
24: transfer all VMCS s.t. faulty unit reaching position $p_f \in \mathcal{F}^*$
25: update $\mathcal{P}, \mathcal{F}$
26: **while** $\mathcal{P}^* \setminus (\mathcal{P} \cap \mathcal{P}^*) \neq \varnothing$ **do** ▷ Conflict-free destination
27:  compute virtual start point $p_s$
28:  $\mathcal{P}_t = \mathcal{P}^* \setminus (\mathcal{P} \cap \mathcal{P}^*)$
29:  **for** $p_t \in \mathcal{P}_t$ **do**
30:  $T \leftarrow$trajectory coordinates set from $p_s$ to $p_t$ using $A^*$
31:  **while** $\exists p' \in \mathcal{P}_t \cap T$ and $p' \neq p_t$ **do**
32:  $\mathcal{P}_t = \mathcal{P}_t \setminus \{p'\}$
33:  **end while**
34:  **end for**
35:  transfer units with shortest path to $\mathcal{P}_t$ using (6)
36:  update $\mathcal{P}, \mathcal{F}$
37: **end while**
38: **Return** $\mathcal{P}, \mathcal{F}$

### D. Normal Units Transfer: Conflict-Free Assembly Sequence

Previous work [10] plans the disassembly and assembly sequence by always selecting the option that maximizes the CM at each step, but its applicability is limited to standard rectangular configurations. In fact, once the CM exceeds a fixed threshold, the control authority of the entire system is sufficient for the designed controller. We introduce a more flexible strategy that can be applied to arbitrary configurations. This process is repeated until $\mathcal{P} = \mathcal{P}^*$.

*1) Conflict-Free Destination for Next Unit:* During reassembling the remaining normal units, we initialize their corresponding set of *target positions* as $\mathcal{P}_t := \mathcal{P}^* \setminus (\mathcal{P} \cap \mathcal{P}^*)$. To achieve efficient re-assembly, we need to identify an assembly sequence that guarantees accessibility, meaning that no earlier-placed unit obstructs the movement of subsequent units.

Specifically, for each step, a virtual start point $p_s$ is chosen near $\mathcal{P}^*$ but outside $\mathcal{P} \cup \mathcal{P}^*$. An $A^*$ path is then computed

from $p_s$ to each candidate target position. If a candidate target lies along the $A^*$ path to another target, as illustrated in the conflict-free stage of Fig. 2(d) (Units 2–8 on the orange path and Unit 10 on the yellow path), it is discarded to avoid blockage. The remaining target positions in the set $\mathcal{P}_t$ (see **Algorithm 2**, Line 26–33) are thus guaranteed not to obstruct the paths of subsequent transfers.

*2) Reconfiguration Planner:* After determining the destination in $\mathcal{P}_t$, we next identify the unit to be relocated. For each candidate unit in set $\mathcal{P}_c := \mathcal{P} \backslash (\mathcal{P} \cap \mathcal{P}^*)$, we compute the $A^*$ path length from its current position to the destination. The unit with the minimum path length is then selected and transferred to the destination.

When $|\mathcal{P}_t| > 1$, the normal units may be transferred to different destinations. To assign normal units $p_{c,j} \in \mathcal{P}_c$ to target positions $p_{t,i} \in \mathcal{P}_t$ such that the total transfer distance is minimized, we formulate a binary assignment optimization problem:

$$\begin{aligned} \min_{x_{ij}} \quad & \sum_{i=1}^{|\mathcal{P}_t|} \sum_{j=1}^{|\mathcal{P}_c|} x_{ij} \cdot L(p_{c,j}, p_{t,i}) \\ \text{s.t.} \quad & \sum_{j=1}^{|\mathcal{P}_c|} x_{ij} = 1, \quad \forall i = 1, \ldots, |\mathcal{P}_t| \\ & \sum_{i=1}^{|\mathcal{P}_t|} x_{ij} \leq 1, \quad \forall j = 1, \ldots, |\mathcal{P}_c| \\ & x_{ij} \in \{0, 1\} \end{aligned} \tag{6}$$

Here, $x_{ij}$ is a binary variable indicating whether candidate unit $p_{c,j}$ is assigned to target position $p_{t,i}$, constraint $\sum_{j=1}^{|\mathcal{P}_c|} x_{ij} = 1$ indicates that each target $p_{t,i}$ is assigned exactly one unit, constraint $\sum_{i=1}^{|\mathcal{P}_t|} x_{ij} \leq 1$ indicates that each normal unit $p_{c,j}$ is scheduled at most once, and $L(p_{c,j}, p_{t,i})$ denotes the transfer distance between them.

## IV. Evaluation and Real-World Experiments

We employ a high-fidelity quadrotor model in CoppeliaSim [22] for simulation, following the setup in [10]. The experiments evaluate the controllability of arbitrarily shaped MARS with multiple faulty units that cannot be addressed by [9], [10], analyzing the CM as well as disassembly and assembly steps under failures. In particular, we compare the proposed method against [9], [10] on standard $M \times N$ configurations.

### A. Flexible Self-Reconfiguration

*1) Standard $M \times N$ Configuration with Multiple Faulty Units:* Unlike previous work [10] that considers only single-fault cases in the $3 \times 2$ configuration, we evaluate this configuration with two faulty units. Fig.3(a) illustrates the reconfiguration sequence and corresponding CM analysis. The algorithm completes the sequence in four disassembly and assembly steps (2–5), achieving a minimum CM of 1.3736, which is sufficient for stable control. The dynamic reconfiguration simulation results, obtained using the fault-tolerant controller in [17], are shown in Fig.3(b). In Step 2, after identifying the VMCS with **Algorithm 1**, normal units detach and relocate as shown in Step 2–3, ensuring that all faulty units can be transferred via VMCS. Subsequently, Algorithm 2 transfers the two VMCS to their final positions in the optimal configuration (Steps 4–5).

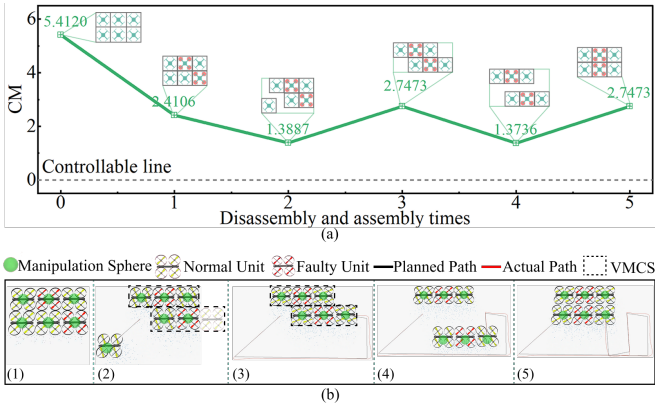


Fig. 3. Self-reconfiguration process of a $3 \times 2$ assembly. (a) Controllability margin versus disassembly and assembly step with corresponding configurations shown. (b) Dynamical simulation of the self-reconfiguration sequence in CoppeliaSim.

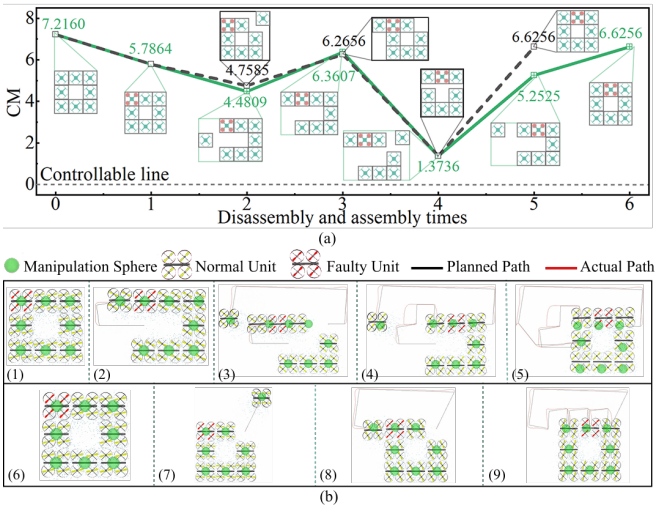


Fig. 4. Self-reconfiguration process of a $3 \times 3$ hollow assembly. (a) CM versus disassembly and assembly times. A higher $c_1$ weight (black dashed line) yields a higher average CM than the lower $c_1$ setting (green line). The black dashed line also results in a shorter reconfiguration sequence, as path clearance is skipped (Step 4 in green line) due to the relocation of normal unit (No. 3, top right). (b) Dynamic simulation in CoppeliaSim: (1)–(5) show the sequence for the green line, and (6)–(9) for the black dashed line.

*2) Hollow Configuration:* We further evaluate a hollow configuration. Fig. 4(a) illustrates two self-reconfiguration sequences of a $3 \times 3$ hollow configuration with one faulty unit (No. 1, top left, marked in red) under different weight parameter settings, while Fig. 4(b) shows the corresponding simulations. Specifically, we set $(c_1, c_2) = (2, -0.1)$ for the green line and $(c_1, c_2) = (4, -0.1)$ for the black dashed line in (5a). For small-scale configurations, assigning a higher weight to the CM term increases the average CM value, whereas the path-length weight has little impact due to short paths at each step. Thus, to ensure controllability, prioritizing the CM weight is essential. Notably, the disassembly and assembly times of the black dashed sequence are fewer than

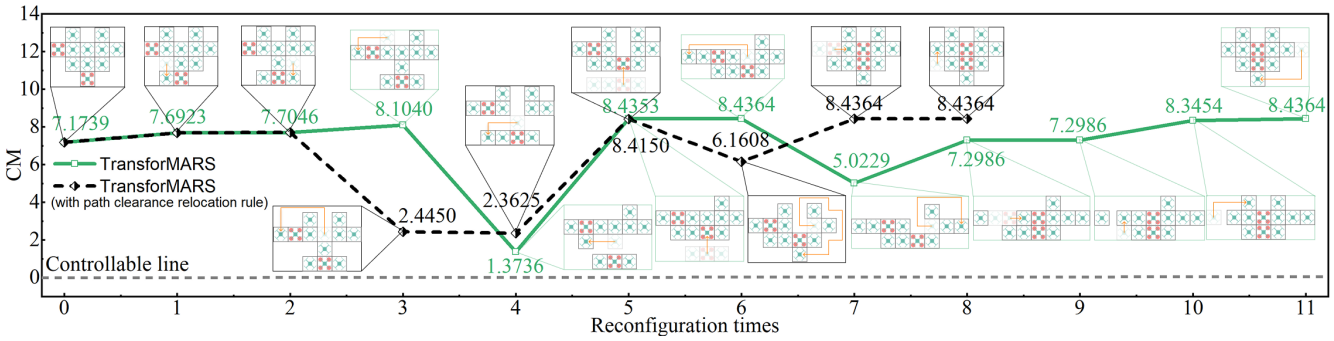


Fig. 5. Self-reconfiguration sequence of an 11-unit heart-shaped assembly. The black dashed line and green line correspond to the reconfiguration methods with and without the path clearance relocation rule, respectively. With the relocation rule, the number of reconfiguration steps is reduced, demonstrating its effectiveness in improving self-reconfiguration efficiency.

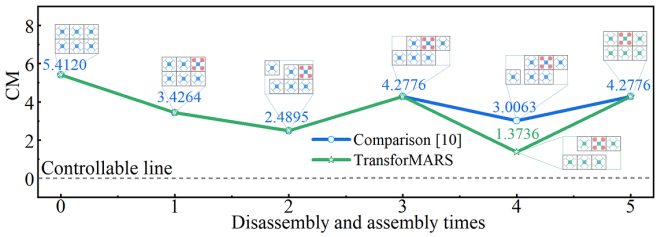


Fig. 6. Comparison of disassembly and assembly steps and CM for a failure at position 3 in a $3 \times 2$ assembly. TransforMARS (green) requires no extra steps compared with [10] (blue). Although the CM decreases at step 4, the system remains controllable.

those of the green one, since unit (No. 3, top right) serves as the best candidate to form a VMCS, thereby avoiding extra disassembly and assembly steps (see black dashed line, Step 3) in path clearance.

*3) Arbitrarily Shaped Configuration with Multiple Faulty Units:* Unlike [10], which addresses only standard rectangular configurations, we further evaluate arbitrarily shaped assemblies, such as an 11-unit heart-shaped configuration in Fig. 5. The green line denotes transferring units to a waiting position within the same row, while the black dashed line applies the *path clearance relocation rule* to move units directly to vacant coordinates in the optimal configuration. With this strategy, the reconfiguration sequence is reduced by three steps (from 11 to 8), equivalent to six disassembly-assembly operations, while also avoiding long detours: without it, the total path length increases by 35.29% (from 34 to 46 unit distances). **These results highlight the effectiveness of the path clearance relocation rule in reducing both path length (energy consumption) and reconfiguration times, thereby improving efficiency and safety.**

### *B. Comparison with the Baseline Method*

Table I compares existing reconfiguration methods [9], [10] with the proposed TransforMARS. [9] addresses rotor-level failures and can handle multiple rotor faults in arbitrary configurations, but it overlooks the controllability of intermediate subassemblies by simply connecting a faulty unit to a single normal one. [10] extends reconfiguration to both rotor- and unit-level failures with explicit controllability analysis, yet it is restricted to single-fault cases in standard rectangular configurations. In contrast, the proposed TransforMARS supports multiple faults at both rotor and unit levels, applies to arbitrary configurations, and maintains subassembly controllability, thereby overcoming the limitations of prior approaches. In the following subsection, we compare the reconfiguration sequences of the baseline methods with those of TransforMARS, and calculate the CM value of MARS in each step.

*1) $3 \times 2$ Configuration:* We first compare our method with previous approach [10] on standard configurations. As shown in Fig. 6, the self-reconfiguration sequences generated by [10] (blue line) and our proposed TransforMARS (green line) are illustrated. Compared with [10], our method achieves the target configuration (Step 5) with the same number of disassembly and assembly steps, while the average CM decreases by only 11.52% due to the transfer of VMCS. **This demonstrates that, unlike [10], which is restricted to limited configurations, TransforMARS not only solves standard $M \times N$ configuration problems with minor CM loss but also generalizes to a broader range of scenarios, including arbitrary shapes such as hollow and heart configurations.**

*2) $3 \times 3$ Configuration:* To further demonstrate the robustness of our method, we compare the CM values at the intermediate stages of different approaches in a $3 \times 3$ self-reconfiguration scenario with one faulty unit. In Fig. 7, the self-reconfiguration sequences generated by different approaches are plotted, along with the line charts representing their intermediate CM values. Solid lines present scenarios with a single rotor failure, while dashed lines denote complete unit failure cases. For CM, our proposed method (green) achieves higher values, with an average improvement of up to 78.03% compared to [9] (purple). The average CM decreases by only 20.50% relative to [10] (blue). In terms of disassembly and assembly, our approach requires 16.67% more steps than [10], yet still reduces the number of steps by 36.36% compared to [9].

However, for both $3 \times 2$ and $3 \times 3$ configurations with one faulty unit, previous work [10] mainly focuses on CM and exhibits limited generalization, as it only addresses the standard $M \times N$ case. In fact, as long as CM $> 0$, the MARS remains controllable. For example, in Step 4 of Fig. 7, the CM reaches 1.3736, which is sufficient for stable control. **In contrast, our approach extends to a wide**

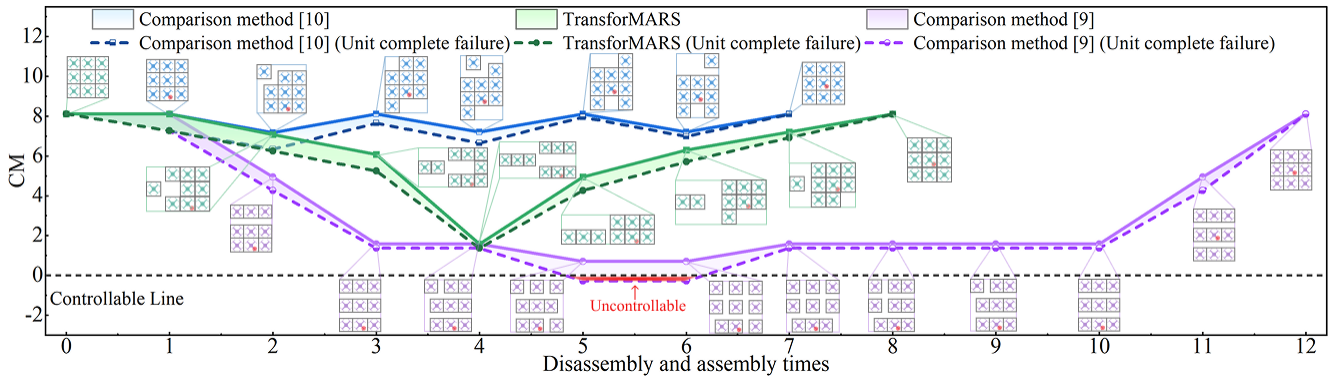


Fig. 7. Comparison of disassembly and assembly times and CM for failure at position 8 in a 3×3 assembly. The solid line represents single rotor failure and dashed line represents unit complete failure. Compared with [9], TransforMARS achieves substantially higher CM with fewer steps. Relative to [10] (blue), the average CM decreases by only 20.50%, and the number of steps increases by only 16.67%.

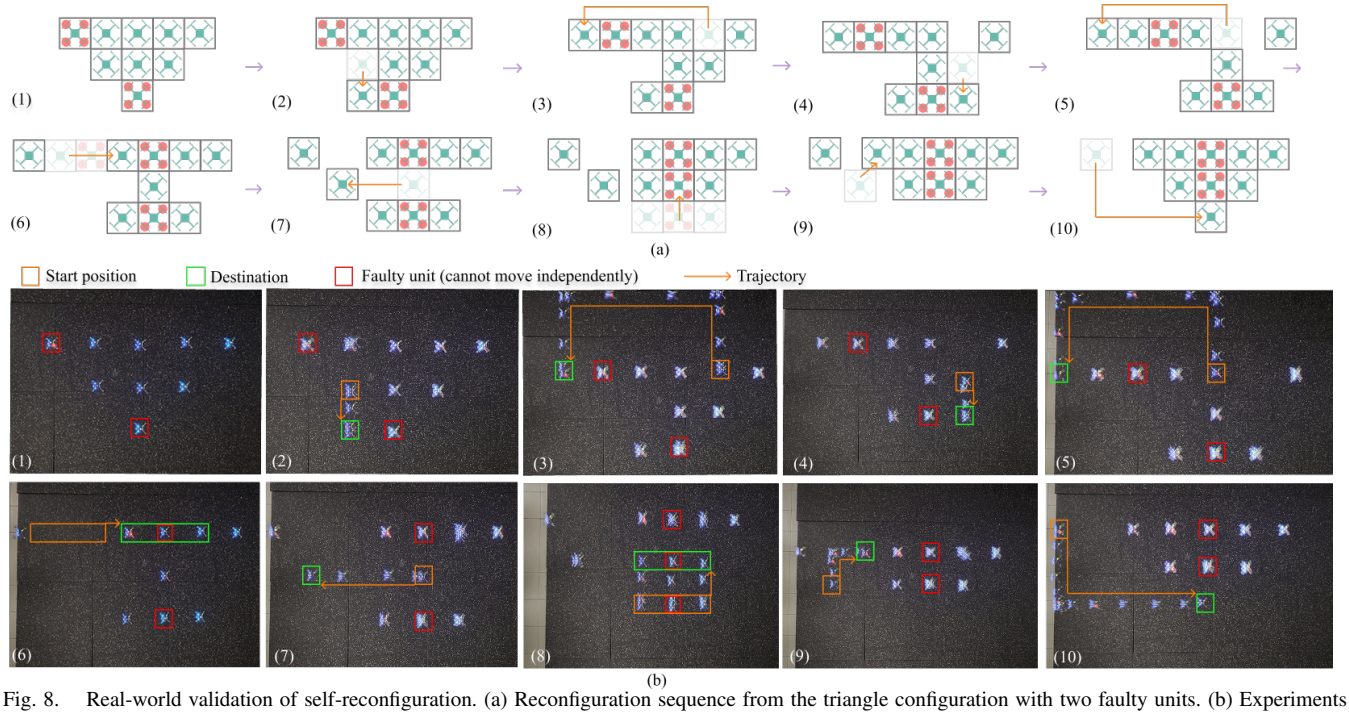


Fig. 8. Real-world validation of self-reconfiguration. (a) Reconfiguration sequence from the triangle configuration with two faulty units. (b) Experiments with 9 Crazyflie drones showing feasible trajectories during the self-reconfiguration process. A video and additional demonstrations are included in the supplementary video submitted with this paper.

TABLE I

COMPARISON OF PREVIOUS METHODS

| Method | Fault | # Faulty Units | Configuration | Safety Guarantee |
|---|---|---|---|---|
| Ref. [9] | rotor | multiple rotors | **arbitrary** | ✗ |
| Ref. [10] | **rotor & unit** | single | rectangular | ✓ |
| **Ours** | **rotor & unit** | **multiple** | **arbitrary** | ✓ |

**range of configurations (e.g., hollow structures in Fig. 4, arbitrary assemblies in Fig. 5, and multi-error units) and further incorporates accessible path planning for unit movement, as demonstrated in the previous section. This offers greater practical significance than methods that only emphasize CM.**

### C. Real-World Experiment

We conduct real-world experiments to validate the proposed TransforMARS, a self-reconfiguration framework with integrated path planning. The testbed consists of nine Crazyflie micro-quadrotors serving as MARS units. A faulty unit cannot move independently and is transported by the VMCS determined by our planner. In contrast to previous approaches that only considered the reconfiguration sequence, where normal units were placed at manually designed waiting positions [9] or followed pre-defined trajectories [10], our method jointly plans both the accessible self-reconfiguration sequence and feasible trajectories for arbitrary configurations. Fig. 8(a) illustrates the reconfiguration sequence and trajectories from a triangle configuration with two faulty units, and Fig. 8(b) shows real-world experiments where Crazyflie drones follow planned trajectories to achieve successful self-reconfiguration.

Specifically, Steps 2–4 in Fig. 8 illustrate the construction of the VMCS. Steps 6 and 8 show the transfer of the VMCS to its final position in the target configuration, while Steps 5 and 7 demonstrate path clearance for the VMCS. Finally, Steps 9 and 10 depict the accessible disassembly and assembly sequences in which normal units complete the target configuration.

It is important to emphasize that our experiments focus exclusively on the trajectory execution and coordination phase of self-reconfiguration. In comparison, full self-reconfiguration, which involves both physical docking and separation, presents greater challenges due to the requirement of continuous and repeated operations. These operations remain inherently unstable, as existing docking methods [1], [2] and separation mechanisms [4] already exhibit noticeable instability after just a single docking or separation cycle. Moreover, full self-reconfiguration must repeatedly perform such unstable procedures until the system reaches its target configuration, making the process considerably more complex and error-prone than executing a single docking or separation action.

## V. Conclusion and Future Work

To address rotor- and unit-level failures in MARS, this paper presents a flexible self-reconfiguration framework that enhances fault-tolerant control. The proposed approach integrates self-reconfiguration with path planning for transporting faulty units, and supports reconfiguration into arbitrary target configurations. Specifically, our algorithms (i) identify the VMCS to enable safe transfer of faulty units, (ii) perform path clearance to guide the VMCS to their designated positions, and (iii) plan accessible disassembly and assembly sequences to avoid blockage at each stage. Compared with existing approaches, this work demonstrates the first general self-reconfiguration method that handles multiple faulty units in arbitrary configurations.

For future work, the existing docking [1], [2] and separation [4] mechanisms of MARS still face several challenges that can significantly impact self-reconfiguration performance. Until now, no successful real-world self-reconfiguration experiments have been reported. Therefore, we plan to conduct real-world experiments incorporating both disassembly and assembly functions to further validate the proposed framework.

## References


[1] D. Saldana, 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. Saldana, 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] J. Sugihara, T. Nishio, K. Nagato, M. Nakao, and M. Zhao, "Design, control, and motion strategy of trady: Tilted-rotor-equipped aerial robot with autonomous in-flight assembly and disassembly ability," *Advanced Intelligent Systems*, vol. 5, no. 10, p. 2300191, 2023.

[4] D. Saldana, 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] J. Zhang, F. Li, X. Lu, C. Zhang, Y. Xin, R. Zhao, and S. Lyu, "Design and control of rapid in-air reconfiguration for modular quadrotors with full controllable degrees of freedom," *IEEE Robotics and Automation Letters*, 2024.

[6] J. Sugihara, M. Zhao, T. Nishio, K. Okada, and M. Inaba, "Beatle—self-reconfigurable aerial robot: Design, control and experimental validation," *IEEE/ASME Transactions on Mechatronics*, 2024.

[7] S. Li, F. Liu, Y. Gao, J. Xiang, Z. Tu, and D. Li, "Airtwins: Modular bicopters capable of splitting from their combined quadcopter in midair," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 6068–6075, 2023.

[8] D. Saldana, B. Gabrich, M. Whitzer, A. Prorok, M. F. Campos, M. Yim, and V. Kumar, "A decentralized algorithm for assembling structures with modular robots," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2736–2743.

[9] 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.

[10] R. Huang, S. Tang, Z. Cai, and L. Zhao, "Robust self-reconfiguration for fault-tolerant control of modular aerial robot systems," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 12 614–12 620. [Online]. Available: https://arxiv.org/abs/2503.09376

[11] B. Gabrich, D. Saldaña, and M. Yim, "Finding structure configurations for flying modular robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6970–6976.

[12] J. Xu and D. Saldaña, "Finding optimal modular robots for aerial tasks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 922–11 928.

[13] Y. Su, Z. Jiao, Z. Zhang, J. Zhang, H. Li, M. Wang, and H. Liu, "Flight structure optimization of modular reconfigurable uavs," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 4556–4562.

[14] J. Xu, D. S. D'antonio, and D. Saldaña, "H-modquad: Modular multirotors with 4, 5, and 6 controllable dof," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 190–196.

[15] J. Xu, D. S. D'Antonio, and D. Saldaña, "Modular multirotors: From quadrotors to fully-actuated aerial vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 19 328–19 339, 2025.

[16] M. Li, K. Cui, and H. Koeppl, "A modular aerial system based on homogeneous quadrotors with fault-tolerant control," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 8408–8414.

[17] R. Huang, Z. Zhang, S. Tang, Z. Cai, and L. Zhao, "Robust fault-tolerant control and agile trajectory planning for modular aerial robotic systems," 2025. [Online]. Available: https://arxiv.org/abs/2503.09351

[18] H. Luo and T. L. Lam, "Auto-optimizing connection planning method for chain-type modular self-reconfiguration robots," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1353–1372, 2022.

[19] L. Zhao, Y. Jiang, M. Chen, K. Bekris, and D. Balkcom, "Modular shape-changing tensegrity-blocks enable self-assembling robotic structures," *Nature Communications*, vol. 16, no. 1, p. 5888, 2025.

[20] I. Yazidi, B. Piranda, M. Ouisse, and J. Bourgeois, "Efficient balance detection for modular robots," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 14 119–14 124.

[21] G. Sun, R. Zhou, Z. Ma, Y. Li, R. Groß, Z. Chen, and S. Zhao, "Mean-shift exploration in shape assembly of robot swarms," *Nature Communications*, vol. 14, no. 1, p. 3476, 2023.

[22] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013, www.coppeliarobotics.com.

[23] J. A. Preiss*, W. Hönig*, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3299–3304, software available at https://github.com/USC-ACTLab/crazyswarm. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989376