Adaptive Flow Planning of Modular Spherical Robot Considering Static Gravity Stability

Haobo Luo and Tin Lun Lam

***Abstract*—Modular robots have a unique obstacle-crossing method, flow. Flow is realized by constantly changing the connection relationship between modules, namely reconfiguration. Existing flow planning methods do not consider the static stability in their adaptation to obstacles. This letter proposes a flow planning method with scalability, adaptability, and static gravity stability. The criterion of static gravity stability is always satisfied through the following two innovations. First, each target configuration in the flow process is designed to grasp obstacles like vines. Second, in motion planning, each module maintains contact with the obstacle or a fixed module to maximize the supporting polygon of the configuration. What's more, the simplified path output by the connection planning and the precise calculation based on the mesh model realize the scalability and adaptability of the flow planning method. In simulation, we evaluate the adaptability to various obstacles and the margin of static gravity stability.**

***Index Terms*—Cellular and modular robots; planning, scheduling and coordination; path planning for multiple mobile robots or agents**

I. Introduction

MODULAR robots can change the connection relationship between modules according to the requirements of the environment and tasks. The configuration of modular robots can be quadruped, snake, lizard, etc. The sequence of actions to transform the current configuration of the modular robot into the target configuration is called the reconfiguration process. Modular Self-Reconfigurable Robot (MSRR) performs better than traditional fixed robots in versatility and adaptability [1] [2]. MSRR can move in three ways: wheeled locomotion, footed locomotion and flow. For wheeled locomotion, some modules need to be equipped with wheels [3] or tracks [4]. Footed locomotion is achieved by changing the angle between the modules [5]. These two locomotion ways generally do not require disconnection or connection between modules. Flow is a locomotion method implemented by modular robots through multiple consecutive reconfiguration processes [6]. This letter focuses on the planning of flow. The flow process of MSRR simulates the adaptability of water to different surfaces, which embodies the unique advantage of MSRR in exploration.

MSRR can be divided into two types, chain-type and lattice-type. Chain-type MSRR includes FreeBot [7], M-TRAN III [8] and SuperBot [9]. Lattice-type MSRR includes Telecube [10], ATRON [11] and Vacuubes [12]. Since the hardware modules of different MSRR have diverse structures and drives, algorithm research is usually based on general kinematic models, such as Proteo [13], SlidingCube [14] and RollingSphere [15]. The shapes of Proteo and SlidingCube are rhombic dodecahedron and cube respectively, while RollingSphere is spherical. The connection of spherical modules can better adapt to uneven obstacle surfaces [15], so this letter focuses on hardware modules that can be abstracted as RollingSphere models, such as 3D Catoms [16], FireAnt-3D [17] and FreeBot [7].

The flow planning method is based on reconfiguration algorithms. The research directions of reconfiguration algorithms can be divided into search-based approaches, agent-based approaches, and control-based approaches [6]. Search-based approaches estimate the path distances between initial, current, and final configurations by some metrics such as the minimal number of steps [18] to search for the shortest path. Search-based approaches consume a lot of memory and computing time to traverse the entire configuration space, which grows exponentially as the number of modules increases [19]. In agent-based approaches, each module in MSRR is regarded as an agent that can observe the local environment and act independently. For example, Million Module March [14] inspired by reinforcement learning can direct a large-scale configuration composed of SlidingCube modules to flow over obstacles composed of cubes in a decentralized manner. Agent-based approaches have a certain degree of randomness and are difficult to debug on the hardware. Control-based approaches simulate the feedback control loop to navigate the sequence of configuration changes converging to the target one. In the gradient-based [20] method, the non-source module calculates the steepest descent gradient following the density of nearby attractors broadcast by the source module. Further, Luo *et al.* [15] proposed a reconfiguration algorithm that calculates large gradients through three-step minimization and a flow strategy that uses the connection of spherical RollingSphere modules to adapt to obstacles. However, the motion planning in [15] still relies on centralized calculation in discrete space, and its target configuration extracted from OctoMap is not accurate enough. More importantly, the basic actions in [15] do not consider the static gravity stability, so that many configurations may be turned over by gravity in practical applications.

This letter proposes a flow planning method superior to

Manuscript received: September, 9, 2021; Revised December, 11, 2021; Accepted January, 22, 2022. This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China (62073274), and the funding AC01202101103 from the Shenzhen Institute of Artificial Intelligence and Robotics for Society. *(Corresponding author: Tin Lun Lam)*

The authors are with School of Science and Engineering, the Chinese University of Hong Kong, Shenzhen, China and also with Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), the Chinese University of Hong Kong, Shenzhen, China. (e-mail: haoboluo@link.cuhk.edu.cn; tllam@cuhk.edu.cn)

Digital Object Identifier (DOI): see top of this page.

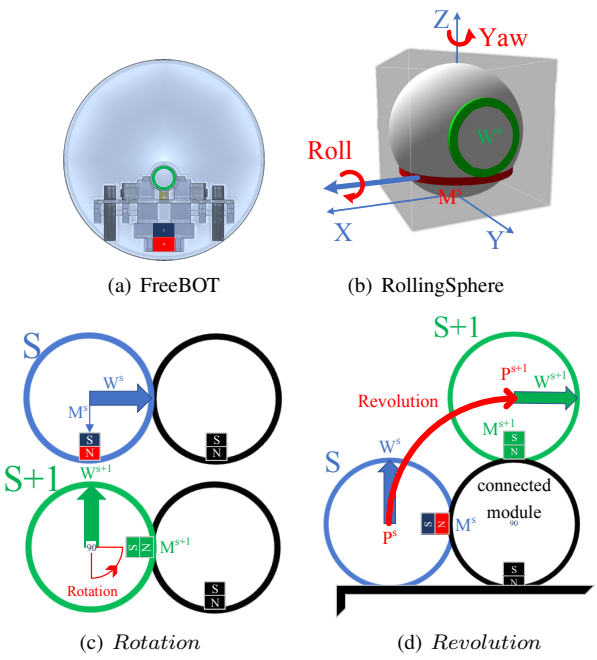
existing work in terms of scalability, adaptability, and static stability. First, we divide the reconfiguration planning into connection planning and motion planning. The connection planning outputs a simplified path to each module, which enables computationally intensive motion planning to be executed in a distributed and scalable manner. Second, our flow planning method uses calculations based on the mesh model instead of Octomap [15], which improves the adaptability of the output trajectory to rugged and complex obstacles. Third, we considered how to satisfy the criterion of static gravity stability during the entire flow process. If the vertical projection point of the Center of Gravity (CoG) is within the 2D convex hull of the vertical projection points of the 3D contact points between MSRR and the obstacle, MSRR can move stably without tipping. This is called the criterion of static gravity stability [21]. The above-mentioned 2D convex hull is also called the supporting polygon. In motion planning, each module maintains contact with the obstacle or a fixed module to expand the supporting polygon of the configuration. In the target configuration design, the branches that grow from the appropriate vacancies in the trunk path of the target configuration grab obstacles like the feet of vines. These two innovations make each configuration in the reconfiguration process statically stable, as proven in Section V.

The remaining content of this letter is organized as follows. Section II describes the abstract kinematic model and the definition of configuration. Section III introduces the flow planning method including target configuration design, connection planning and motion planning. The simulation results are shown in Section IV. Section V provides theoretical proof and discussion. Section VI finishes this letter by conclusions.

II. Problem Formulation

A. Basic Actions and Prerequisites

The method proposed in this letter applies to a kinematics model, RollingSphere [15]. The RollingSphere model has three basic actions, $\{Yaw, Rotation, Revolution\}$. To describe those actions, two perpendicular unit vectors, M^s and W^s, are defined to represent the direction of connection and the direction of connection change respectively in step s. Take FreeBOT as an example. FreeBOT consists of a spherical iron shell and an internal trolley that has differential wheels and a permanent magnet, as shown in Fig. 1(a). M^s of FreeBOT represents the direction of the permanent magnet attracting the spherical shell of another module, and W^s represents the advancing direction of the differential wheels of the internal trolley. In simulation, M^s and W^s are indicated by the normal vectors of the red circle and the green circle as shown in Fig. 1(b). The inner trolley of FreeBOT has two driving forces, as shown in Fig. 1(b), yaw and roll. With the Yaw force, the advancing direction W^s can be any unit vector on the plane perpendicular to M^s. With the roll force, the trolley can perform two basic actions, $\{Rotation, Revolution\}$. $Rotation$ means that the inner trolley rolls around the center of the module itself without changing the position of the module, as shown in Fig. 1(c). $Revolution$ refers to the position change of the entire module around the center of the connected module,



(a) FreeBOT (b) RollingSphere

(c) *Rotation* (d) *Revolution*

Fig. 1. The RollingSphere kinematics model. (a)FreeBOT; (b) Simulation in Gazebo; (c) *Rotation* action; (d) *Revolution* action.

as shown in Figs. 1(d). These basic actions have different dynamic gravity stability when executed by different hardware implementations of RollingSphere, involving the driving force of the module or the friction on the ground. In this letter, we focus on the static gravity stability and assume that the applied hardware module has fulfilled the four prerequisites: (1) 3D global positioning capability, (2) 3D local perception capability, (3) distributed identifier assignment capability [22], and (4) normal static friction coefficient.

B. Definition of configuration

The configuration of MSRR is composed of the positions of all modules and an adjacency matrix as shown in Eq. (1). In the adjacent matrix, the element $x_{uv} = 1$ means that the module M_u is connected to the module M_v, as shown by $M_u \rightarrow M_v$ in Eq. 2 where u and v represent two module IDs.

$$CFG = \{[X(u,:), P_u] \mid u = 1, \ldots, n\} \tag{1}$$

$$X : x_{uv} = \begin{cases} 1 & M_u \rightarrow M_v \\ 0 & \text{others} \end{cases} \quad u, v = 1, \cdots, n \tag{2}$$

For the sake of distinction, we call the member of the current configuration as module and the member of the target configuration as vacancy. The purpose of the reconfiguration planning is to fill all the vacancies in the target configuration by adjusting the 3D position of each module in the current configuration. A splicing action may be required between two consecutive reconfiguration processes, which varies according to different hardware implementations. For example, the splicing action applicable to FreeBOT is to uniformly reverse the connection sequence between modules on the entire path. For more details, please refer to [15].

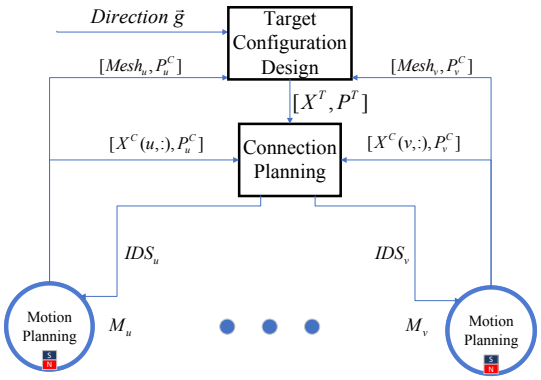


Fig. 2. The distributed framework of the flow planning method.

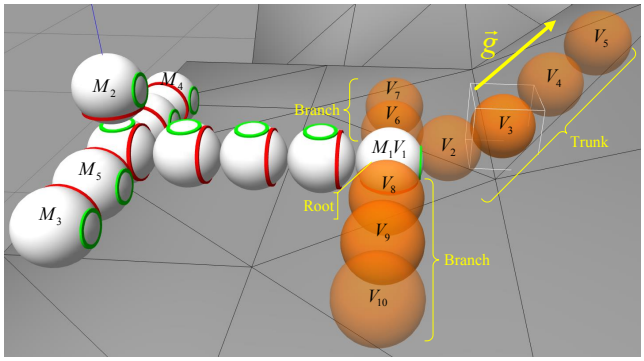


Fig. 3. An example of a reconfiguration process.

III. Flow Planning

The proposed flow planning method consists of three parts: target configuration design, connection planning and motion planning. The distributed framework and the input of each part are shown in Fig. 2. Each module M_u can perform motion planning and perception locally according to the ID sequence of the modules to be passed, denoted as IDS_u. In the initialization phase of the reconfiguration process, each module M_u sends the local mesh model $Mesh_u$, its 3D position P_u^C and current connection relationship $X^C(u,:)$ to a global node for target configuration design and connection planning. The global node can run on any module and receive the overall goal direction $\vec{g}$ on the X-Y plane input by the user.

A. Target Configuration Design

A target configuration and a current configuration are shown in Fig. 3 as an example. On the right side of Fig. 3, the orange transparent spheres represent vacancies $V_v, v = 1, \cdots, n$. On the left side of Fig. 3, the white spheres with ribbons represent modules $M_u, u = 1, \cdots, n$. The $\vec{g}$ in Fig. 3 is a goal direction vector on the X-Y plane pointing to the destination of the entire MSRR. The design of the target configuration mainly includes the calculation of the next vacancy and the selection of bifurcation vacancies.

1) Calculate the Next Vacancy: The 3D positions of all modules in the current configuration are projected to $\vec{g}$ to select the farthest module, called the anchor module, which is matched with the root vacancy such as the $M_1 = V_1$ in

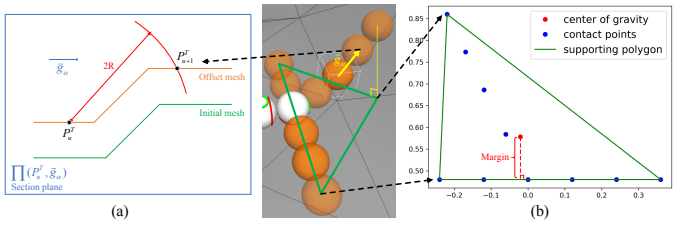


Fig. 4. Target configuration Design. (a) The calculation method of the next vacancy. (b) The supporting polygon of the target configuration.

Fig. 3. Recursively, V_1 triggers the calculation of the new vacancy V_2. The 3D position of the new vacancy P_{u+1}^T should be at a distance $2R$ from P_u^T and at a distance R from the contact triangle $\triangle(O_{u+1}^c, \vec{n}_{u+1}^c)$ in the mesh model of the environment. These two constraints are expressed as Eq. 3 and 4, where R represents the radius of each module.

$$\| P_{u+1}^T - P_u^T \| = 2R \tag{3}$$

$$\| (P_{u+1}^T - O_{u+1}^c) \cdot \vec{n}_{u+1}^c \| = R \tag{4}$$

The two constraints in Eq. 3 and 4 have not completely restricted the 3D position P_{u+1}^T. We add a constraint about $\vec{g}_\alpha$ to calculate candidate positions. $\vec{g}_\alpha$ represents the vector obtained by rotating $\vec{g}$ around the Z axis of the world coordinate system by an angle $\alpha \in [-180°, 180°)$. A candidate position is restricted to the plane where P_u^T and $\vec{g}_\alpha$ are located, denoted as $\prod(P_u^T, \vec{g}_\alpha)$. The constraint of Eq. 4 can be satisfied by offsetting the local mesh near P_u^T by a distance of R outward along the normal of vertices [23]. The constraint of Eq. 3 can be satisfied by solving the intersection of the offset mesh and the sphere $\bigodot(P_u^T, 2R)$ with P_u^T as the center and $2R$ as the radius. In order to meet the above three constraints simultaneously, we use the plane $\prod(P_u^T, \vec{g}_\alpha)$ to cut the sphere $\bigodot(P_u^T, 2R)$ and the offset mesh to find the intersection points, as shown in Fig. 4(a). From these intersection points, the point with the farthest projection to $\vec{g}_\alpha$ is chosen as a candidate position. For example, $\vec{g}_\alpha$ can has 13 values with $\alpha = i * 10°, i = [-6, 6], i \in \mathbb{Z}$, corresponding to 13 candidate positions. We define the supportability of a contact point within the contact triangle $\triangle(O^c, \vec{n}^c)$ as $\mathcal{S} = sign * \cos(\angle(\vec{n}^c, \vec{g}_\alpha))$, where $sign$ outputs $+1$ when MSRR climbs up, otherwise -1. Among all candidate positions, a position with the maximum $\mathcal{S}$ is selected as the 3D position of the next vacancy.

2) Select Bifurcation Vacancies: All the vacancies in the trunk path shown in Fig. 3 can be calculated by the method in Subsection III-A1. Next, we attach branches to the appropriate vacancies in the trunk path, just like the feet of a vine. If the contact points of the x consecutive vacancies in the trunk path of the target configuration have $\mathcal{S} < 0$, the two vacancies before and after these vacancies grow two branches with a total length of x based on the method in Subsection III-A1. The specific number of vacancies in the left or right branch is adjusted according to the supportability of the contact point in their respective directions. For example, the four consecutive vacancies, V_2, V_3, V_4 and V_5 in Fig. 3 are on a steep surface,

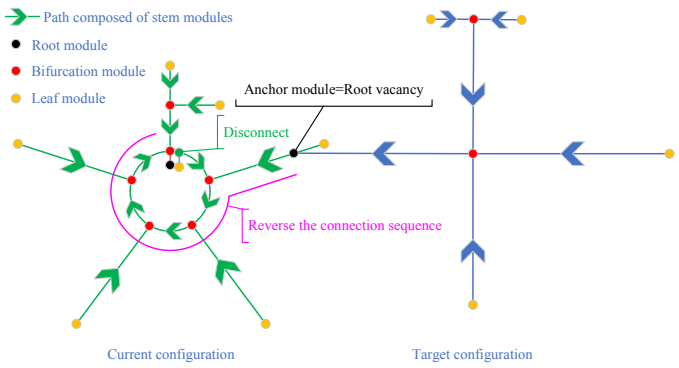


Fig. 5. An example of connection planning between two configurations.

and the vacancy V_1 on a flat surface in front of these four modules is selected as a bifurcation vacancy. As shown in Fig. 4(b), the two branches can expand the supporting polygon into a triangle, a quadrilateral, and so on. The distance between the vertical projection point of the CoG and one edge of the supporting polygon that is closest to the CoG is defined as the margin of static gravity stability, denoted as $\mathbb{M}$. The margin of static gravity stability is increased by the designed branches.

B. Connection Planning

The adjacency matrix input to the connection planning can be used to draw the graph of the connection relationship of the configuration. Fig. 5 shows an example of the drawn graphs based on two adjacency matrices in another reconfiguration process. In Fig. 5, we define four types of modules: if in the adjacency matrix,

(1) Row u has no 1 $\rightarrow$ root module M_u;
(2) Column u has no 1 $\rightarrow$ leaf module M_u;
(3) Row u and Column u has 1 $\rightarrow$ stem module M_u;
(4) Column u has multiple 1's $\rightarrow$ bifurcation module M_u.

We define a module path in the configuration as the path composed of all intermediate modules that connect one module to another. It can be seen in Fig. 5 that the directions of all module paths eventually converge to a unique circuit of the current configuration or a unique root vacancy of the target configuration. One attribute of the RollingShere module is single out-degree which means that the module has only one active connection point, such as FreeBOT. Generally, we have theorem 1 as follows:

Theorem 1. *A connected component composed of single out-degree modules either contains one root module or one circuit.*

Theorem 1 can be proved by knowledge in [24]. Theorem 1 is explained as that the fully connected configuration composed of single out-degree modules, such as RollingSphere modules, can be transformed into a tree through at most one disconnection. As shown by the current configuration in Fig. 5, two arbitrary modules in the circuit are disconnected and transformed into a root module and a leaf module. If the root module of the current configuration is not the anchor module matched with the root vacancy, all intermediate modules in the module path between the root module and the anchor module uniformly reverse the connection sequence, as indicated in Fig. 5. Reversing the connection sequence is completed by the $Rotation$ of each module and is used to splice two consecutive reconfiguration processes [15].

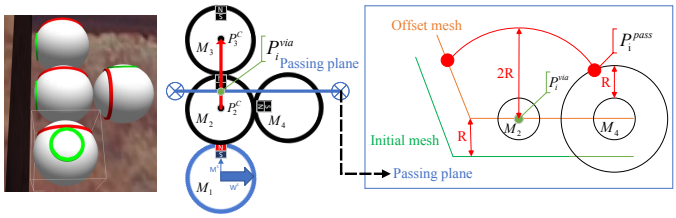
Algorithm 1 Connection planning

1: Disconnect the circuit in the current configuration
2: Reverse the connection sequence
3: **for** LVL in LevelGroupOrderIter() **do**
4: Calculate the List of Leaf Modules LLM
5: Bipartite matching
6: Ignore the matched leaf modules
7: **end for**
8: Update the IDs of the vacancies

The algorithm for connection planning is summarized in Algorithm 1. As introduced above, Line 1-2 in Algorithm 1 transform the current configuration into a tree-like configuration whose root module is matched with the root vacancy of the target configuration. The next step is to match the remaining vacancies in the target configuration with the remaining modules in the current configuration one by one. Line-3 of Algorithm 1 iterates over the tree of the target configuration applying level-order strategy [25]. The LVL in Line-3 represents a List of Vacancies for each Level. For example, the vacancies $LVL = [V_2, V_6, V_8]$ in Fig. 3 are at the same level. In Line-4 of Algorithm 1, all leaf modules in the current configuration are calculated, such as $LLM = [M_2, M_3, M_4]$ in Fig. 3. In Line-5 of Algorithm 1, the minimum weight full bipartite matching [26] between LVL and LLM is performed using the weight defined in Eq. 5. In Eq. 5, the module distance $d_{module}(M_u, V_v)$ represents the number of intermediate modules included in the module path. The Euclidean distance $d_{euc}(P_u^C, P_v^T)$ is standardized by the maximum Euclidean distance in the current iteration. For example, M_3 in Fig. 3 will be matched with V_8, even though M_3 has the same d_{module} with the other two vacancies in LVL. Once some modules are matched, they will be ignored in Line-6 of Algorithm 1 to expose their connected modules as leaf modules for the next iteration. These iterations of bipartite matching make the total path of all modules the shortest. After all iterations, the vacancies are sorted according to their level in the tree and their weights. The sorting result updates the IDs of the vacancies to indicate the priorities. Finally, the IDs of modules or vacancies in the path between each module M_u and its matching vacancy are output as the simplified path IDS_u for the motion planning of each module M_u.

$$\mathbb{W}(M_u, V_v) = d_{module}(M_u, V_v) + \frac{d_{euc}(P_u^C, P_v^T)}{d_{max}} \quad (5)$$

One attribute of the configuration of MSRR is connectivity [13] which means that any two modules in the configuration can be connected by a path composed of other modules. Our connection planning algorithm always maintains the connectivity of each successive configuration in the flow process.



(a) Top view (b) Passing plane

Fig. 6. An example of calculating passing points in motion planning.

C. Motion Planning

The current configuration contains two types of modules: fixed and free modules. Free modules refer to leaf modules that have not reached their target vacancy. Modules other than leaf modules and leaf modules that have reached the target vacancies are fixed. A free module can have two types of collisions, one with other free modules and one with fixed modules or obstacles. The first type of collision is avoided by using the updated ID of the matched target vacancy as a priority. Modules with lower priority will stop at the appropriate step when they foresee a collision, until the module with higher priority has left the collision range.

The second type of collision is avoided by setting the passing points of the trajectory. Fig. 6 shows how to calculate the passing points that are in contact with obstacles or fixed modules without colliding. In the top view shown in Fig. 6(a), M_1 is a free module. M_2 is the connected module of M_1. M_3 is the next module to be connected by M_1 according to the connection planning result. M_4 is a fixed module that has reached its target vacancy. First, we take 10 points uniformly on the 3D line segment between P_2^C and P_3^C, denoted as $P_i^{via}, i = 1, \cdots, 10$. The plane crossing P_i^{via} and with $(P_3^C - P_2^C)$ as the normal vector is called the passing plane, as shown in Fig 6. The passing plane is used to cut obstacles and fixed modules for their profile. Second, the profiles of the mesh model and the fixed modules such as M_4 are offset outward by a radius R. Third, a passing point P_i^{pass} is selected from the intersection points between the offset profiles of mesh or fixed modules and the circle with P_i^{via} as the center and $2R$ as the radius, as shown in Fig. 6(b).

The trajectory between two consecutive passing points consists of three basic actions. The orientation of M_u changes according to Eq. 6, where the $from\_rotvec$ function calculates the incremental rotation matrix based on the rotation axis $Axis$ and the rotation angle $\Delta\Theta$. For the Yaw action, $Axis$ is the M^s of M_u, and $\Delta\Theta$ is the angle between the 3D vector $(P_i^{pass} - P_u^C)$ and the plane where W^s and M^s of M_u are located. The role of Yaw is to adjust the W^s of M_u to point to the next passing point. For the $Rotation$ action, $Axis$ and $\Delta\Theta$ change according to the collision angle and the position of the next module in the simplified path output by the connection planning. The role of $Rotation$ is to adjust the M^s of M_u to point to the next module to connect. For the $Revolution$ action, $Axis$ is the cross product of the two direction vectors M^s and W^s of M_u, and $\Delta\Theta = 1°$ is user-defined. In addition to the orientation change, the translation change of $Revolution$ is calculated based on Eq. 7, where P_{center}^s represents the 3D position of the currently connected module.

$$\begin{aligned} \Delta R &= from\_rotvec(Axis, \Delta\Theta) \\ R^{s+1} &= \Delta R \times R^s \end{aligned} \tag{6}$$

$$P^{s+1} = (\Delta R) \times (P^s - P_{center}^s) + P_{center}^s \tag{7}$$

When a module moves along the trajectory controlled by the passing points calculated above, it can just touch the mesh model or other fixed modules without colliding. This feature of the trajectory enables the module to maintain contact with the obstacle or a fixed module. The more contact points, the larger the supporting polygon of the configuration. For example, humans can increase the area of the supporting polygon by touching their hands with the ground. We explained this intuition in detail in the proof in Section V.

IV. Simulation

This section introduces the simulation results when the proposed flow planning method is used to control the spherical MSRR crossing five representative obstacles. MSRR shows adaptability and static stability in these processes.

A. Settings

Gazebo is used to demonstrate the trajectories output by the flow planning method. Each module of MSRR in Gazebo is controlled by a ROS node. Each ROS node runs motion planning code and exchanges information with a master node that performs target configuration design and connection planning according to the distributed framework described in Fig. 2. The motion planning code contains functions for collision detection and obstacle avoidance. As a result, Gazebo's built-in collision detection and gravity can be canceled to focus on planning. In other words, each module in Gazebo perfectly follows the trajectory output by the flow planning method. The mesh model of the obstacle is considered to be perfectly reconstructed [27].

In this letter, one step refers to the $Revolution$ of a single module by an angle $\Delta\Theta = 1°$ or the $Rotation$ of a single module by an angle $\Delta\Theta = [60°, 180°)$. The action of $Rotation$ does not change the 3D position of the module and can be executed much faster than the action of $Revolution$. Therefore, the action of $Rotation$ is considered to be finished in one step. The action of Yaw is considered to be completed instantly, not counted as one step.

B. Evaluation Index

The evaluation indicators of the experiment include the margin of static gravity stability $\mathbb{M}$ introduced in Subsection III-A2 and the adaptability calculated by Eq. 8. Eq. 8 evaluates the adaptability of the configuration to obstacles in each step of the trajectory. In Eq. 8, $(P_u^C - O_u) \cdot \vec{n}_u$ represents the distance between the 3D position P_u^C of the module M_u and its nearest triangle $\triangle(O_u, \vec{n}_u)$ in the mesh model of the obstacle.

TABLE I
ADAPTABILITY TO FIVE OBSTACLES

Obstacle (difficulty)	Total steps		Average adaptability, $\overline{\sigma}$	
	Ours	Baseline	Ours	Baseline
Rocky Hill (100%)	9242	/	6.546	/
Stair (23.4%)	11411	10937	3.922	15.439
Fence (45.9%)	2624	2468	4.587	16.791
Stone (56.8%)	2374	2194	4.249	20.863
Cobbles (89.2%)	2475	2201	5.213	33.854

Under perfect adaptation, this distance should be equal to the radius R of the module. This distance is calculated by trimesh [28] and normalized by the diameter $2R$ of the module. The calculation is repeated for n modules to get the σ for a configuration.

$$\sigma = \frac{1}{n}\sum_{u=0}^{n} \parallel \frac{(P_u^C - O_u)\cdot \vec{n}_u - R}{2R} \parallel \tag{8}$$

C. Results

Table I summarizes the total steps and the average adaptability of each step's configuration when the flow planning method is applied to five obstacle environments with different relative difficulties. The five obstacles are Rocky Hill, Stair, Fence, Stone, and Cobbles. The difficulty of the obstacle is defined as the relative percentage of the number of triangles with an area less than πR^2 in the mesh model. The baseline in Table I refers to the flow planning method proposed in [15] based on OctoMap. The baseline method emphasizes the rapidity of the flow process but does not consider the static gravity stability of each configuration in the flow process. This lack of consideration makes the baseline method fail in the face of complex and rugged obstacles, which means that MSRR will frequently tip over. For example, the baseline method fails in the face of RockyHill. The flow planning method proposed in this letter can ensure that each configuration in the flow process meets the criterion of static gravity stability, thereby avoiding the overturning of MSRR. As for more comparisons, the proposed method uses about 10% more steps to climb over the same obstacle in exchange for static gravity stability, and improves the adaptability by about 15% using the mesh model, as shown in Table I. Fig. 7 shows the flow processes of MSRR crossing three obstacles, Rocky Hill in Row-1, Stair in Row-2 and Fence in Row-3. The larger obstacles, Rocky Hill and Stair, take three and four reconfiguration processes respectively, while the other three obstacles only take one reconfiguration process.

Fig. 8(a)(b)(c) show the trajectories of a particular reconfiguration process in the flow process of crossing three obstacles, Rocky Hill, Stair and Fence. It can be seen that the trajectories adapt to the obstacles like vines entwined together. Please refer to the appendix video to recognize the adaptability of the trajectory output by the flow planning method. Fig. 8(d)(e) show how MSRR adapts to the other two obstacles, Stone and Cobbles, during the flow process. Fig. 8(f)(g)(h)(i)(j) show the margin of static gravity stability of the configuration at each step in the five flow processes of crossing five obstacles. It can be seen that all the margins $\mathbb{M}$ are larger than zero, and the trend of $\mathbb{M}$ falls as the steepness of the local triangles in the mesh model of the obstacle increases. For example, when MSRR crosses a fence, the configuration riding on the fence, as shown in the third picture of Row-3 in Fig. 7, has a small margin $\mathbb{M}$, so a trough is produced in Fig. 8(h). Similarly, when MSRR crosses the rocky hill and the stair, there are corresponding 5 and 4 troughs in Fig. 8(f)(g). For example, every 90-degree corner in the stair makes the $\mathbb{M}$ of the configuration smaller.

In summary, the flow planning method proposed in this letter can always satisfy the criterion of static gravity stability and better adapt to obstacles.

V. ANALYSIS AND DISCUSSION

This section summarizes and proves the three sufficient conditions used in the flow planning method to satisfy the criterion of static gravity stability, and discusses the assumptions and limitations of the method.

Theorem 2. *The sufficient conditions for the configurations in the reconfiguration process to satisfy the criterion of static gravity stability are:*

(1) *MSRR maintains connectivity;*
(2) *Each free module M_u maintains contact with the obstacle or a fixed module M_v whose $v \notin IDS_u$;*
(3) *If the contact points of the x consecutive vacancies in the trunk path of the target configuration have $\mathcal{S} < 0$, the two vacancies before and after these vacancies grow two branches with a total length of x.*

Proof. In the following, $P_u^{\perp}$ represents the vertical projection point of the center of the sphere of M_u, and $C_j^{\perp}$ represents the vertical projection point of the j-th contact point between the configuration and the obstacle.

Because of condition (1), the vertical projection point of the CoG of the configuration is $P_G^{\perp} = (\frac{1}{n}\sum_{u=1}^{n} x_u, \frac{1}{n}\sum_{u=1}^{n} y_u) = \frac{1}{n}\sum_{u=1}^{n} P_u^{\perp}$. This equation shows that $P_G^{\perp}$ is a convex combination of points in the set $X_P^{\perp} = \{P_u^{\perp} \mid u = 1, \cdots, n\}$. Therefore, $P_G^{\perp}$ must be in the convex hull of the point set $X_P^{\perp}$. If the convex hull of all contact points $X_C^{\perp} = \{C_j^{\perp} \mid j = 1, \cdots, \infty\}$ contains the convex hull of $X_P^{\perp}$, the criterion of static gravity stability is sufficiently satisfied, as written in Eq. 9.

$$P_G^{\perp} \in Hull(X_P^{\perp}) \subset Hull(X_C^{\perp}) \tag{9}$$

Condition (3) makes the unstable vacancies in the trunk path of the target configuration supported. Taking Fig. 9 as an example, V_2 is supported by V_1 and V_1 will grow two branches similar to Fig. 3. In Fig. 9, the contact point C_1 whose $\mathcal{S} > 0$ ensures that the convex hull of $X_C^{\perp} = \{C_1^{\perp}, C_2^{\perp}, C_3^{\perp}\}$ contains the convex hull of $X_P^{\perp} = \{P_1^{\perp}, P_2^{\perp}\}$. By analogy, the designed target configuration satisfies Eq. 9.

Condition (2) makes the configuration of each step in the reconfiguration process statically stable. When the free module M_u maintains contact with the obstacle, M_u proceeds along the trunk path or the two branches designed in condition (3). Therefore, the change of $P_G^{\perp}$ of the configuration is limited to

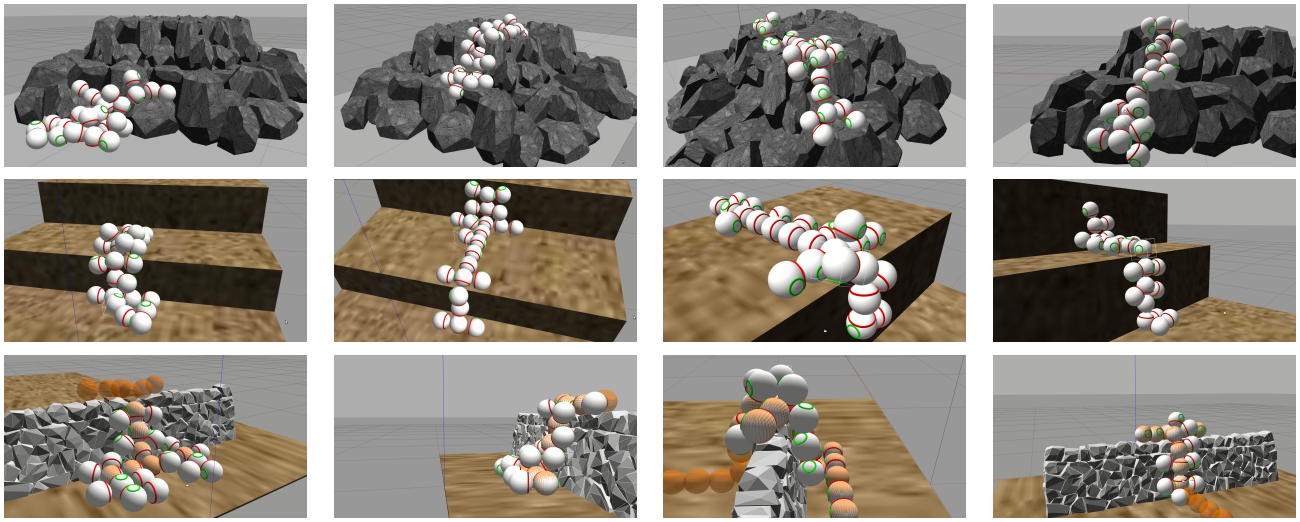
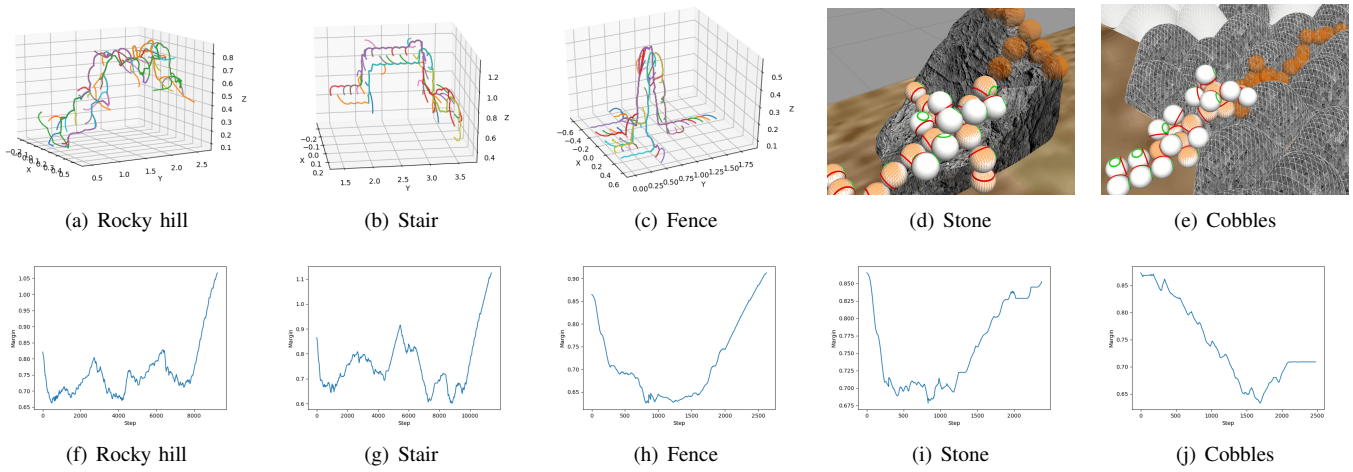


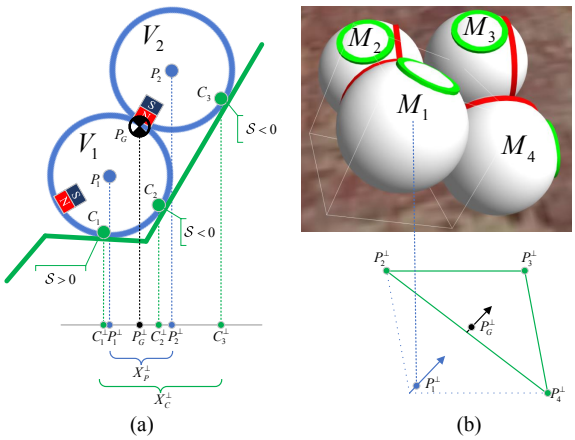
Fig. 7. Screenshots of the flow processes of crossing various obstacles by MSRR. Row-1: Rocky hill. Row-2: Stair. Row-3: Fence.



(a) Rocky hill (b) Stair (c) Fence (d) Stone (e) Cobbles

(f) Rocky hill (g) Stair (h) Fence (i) Stone (j) Cobbles

Fig. 8. The trajectories of crossing three obstacles and the margin of static gravity stability of the configuration of each step in five flow processes.



(a) (b)

Fig. 9. Two examples for the proof of Theorem 2.

the interior of the supporting polygon of the designed target configuration of the previous reconfiguration process. When the free module M_u maintains contact with a fixed module M_v whose $v \notin IDS_u$, these two modules and their respective connected modules locally form a parallelogram as shown in Fig. 9 (b). When M_u is continually being lifted and kept in contact with M_v, the vertical projection of the contact points of the sub-configuration composed of these four modules changes from a parallelogram to a triangle, and the $P_G^\perp$ of this sub-configuration gradually moves to the inside of the triangle. □

The method proposed in this letter still has assumptions and limitations to be studied before being applicable outdoors. The hypothetical positioning and perception are not perfect in an outdoor environment. In our previous work, the 3D position of each module is obtained by fusing the information of IMU, UWB [29] and magnetic sensor array [30]. The local environmental information is reconstructed by multi-robot SLAM [31] and depth estimation [32]. Our future work will focus on dynamic gravity stability which studies the physical interaction between the module and the environment, such as pressure and friction, and will also explore repair methods to deal with failures or damages of some modules. In addition to these two future studies, the third limitation to

be tackled is the global clock used for target configuration design and connection planning in the initialization phase of the reconfiguration process.

VI. Conclusions

This letter proposes a flow planning method with scalability, adaptability and static gravity stability. The flow planning method includes three parts: target configuration design, connection planning and motion planning. In target configuration design, appropriate vacancies on the trunk path of the configuration are attached branches. In motion planning, each module maintains contact with the obstacle or a fixed module. These two parts make configurations in the flow process satisfy the criterion of static gravity stability. Furthermore, the simplified path output by the connection planning and the precise calculation based on the mesh model achieve the scalability and adaptability of the flow planning method.

References

[1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield *et al.*, "The grand challenges of science robotics," *Science robotics*, vol. 3, no. 14, p. eaar7650, 2018.

[2] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[3] F. Hou, N. Ranasinghe, B. Salemi, and W.-M. Shen, "Wheeled locomotion for payload carrying with modular robot," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1331–1337.

[4] S. Kernbach, O. Scholz, K. Harada, S. Popesku, J. Liedke, H. Raja, W. Liu, F. Caparrelli, J. Jemai, J. Havlik *et al.*, "Multi-robot organisms: State of the art," *arXiv preprint arXiv:1108.5543*, 2011.

[5] V. Vonásek, M. Saska, L. Winkler, and L. Přeučil, "High-level motion planning for cpg-driven modular robots," *Robotics and Autonomous Systems*, vol. 68, pp. 116–128, 2015.

[6] H. Ahmadzadeh and E. Masehian, "Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization," *Artificial Intelligence*, vol. 223, pp. 27–64, 2015.

[7] G. Liang, H. Luo, M. Li, H. Qian, and T. L. Lam, "Freebot: A freeform modular self-reconfigurable robot with arbitrary connection point-design and implementation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6506–6513.

[8] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Distributed self-reconfiguration of m-tran iii modular robotic system," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 373–386, 2008.

[9] B. Salemi, M. Moll, and W.-M. Shen, "Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3636–3641.

[10] J. W. Suh, S. B. Homans, and M. Yim, "Telecubes: Mechanical design of a module for self-reconfigurable robotics," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 4. IEEE, 2002, pp. 4095–4101.

[11] M. W. Jorgensen, E. H. Ostergaard, and H. H. Lund, "Modular atron: Modules for a self-reconfigurable robot," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. Ieee, 2004, pp. 2068–2073.

[12] R. F. M. Garcia, J. D. Hiller, K. Stoy, and H. Lipson, "A vacuum-based bonding mechanism for modular robotics," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 876–890, 2011.

[13] M. Yim, Y. Zhang, J. Lamping, and E. Mao, "Distributed control for 3d metamorphosis," *Autonomous Robots*, vol. 10, no. 1, pp. 41–56, 2001.

[14] R. Fitch and Z. Butler, "Million module march: Scalable locomotion for large self-reconfiguring robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 331–343, 2008.

[15] H. Luo, M. Li, G. Liang, H. Qian, and T. L. Lam, "An obstacle-crossing strategy based on the fast self-reconfiguration for modular sphere robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3296–3303.

[16] P. Thalamy, B. Piranda, and J. Bourgeois, "Distributed Self-reconfiguration using a Deterministic Autonomous Scaffolding Structure," UBFC, Research Report 2843, 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02129729

[17] P. Swissler and M. Rubenstein, "Fireant3d: a 3d self-climbing robot towards non-latticed robotic self-assembly," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3340–3347.

[18] A. Pamecha, I. Ebert-Uphoff, and G. S. Chirikjian, "Useful metrics for modular robot motion planning," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 4, pp. 531–545, 1997.

[19] M. Eden *et al.*, "A two-dimensional growth process," in *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, vol. 4. University of California Press Berkeley, 1961, pp. 223–239.

[20] K. Støy, "Controlling self-reconfiguration using cellular automata and gradients," in *Proceedings of the 8th international conference on intelligent autonomous systems (IAS-8)*, 2004, pp. 693–702.

[21] L. Ting, R. Blickhan, and R. J. Full, "Dynamic and static stability in hexapedal runners." *The Journal of experimental biology*, vol. 197, no. 1, pp. 251–269, 1994.

[22] J. Assaker, A. Makhoul, J. Bourgeois, and J. Demerjian, "A unique identifier assignment method for distributed modular robots," in *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20). IEEE*, 2020.

[23] Blender, "Solidify modifier," [EB/OL], https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/solidify.html Accessed September 8, 2021.

[24] B. Bollobás, *Modern graph theory*. Springer Science & Business Media, 2013, vol. 184.

[25] Anytree, "Tree iteration," [EB/OL], https://anytree.readthedocs.io/en/latest/api/anytree.iterators.html Accessed September 8, 2021.

[26] R. M. Karp, "An algorithm to solve the m$\times$ n assignment problem in expected time o (mn log n)," *Networks*, vol. 10, no. 2, pp. 143–152, 1980.

[27] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[28] Dawson-Haggerty et al., "trimesh." [Online]. Available: https://trimsh.org/

[29] M. Li, G. Liang, H. Luo, H. Qian, and T. L. Lam, "Robot-to-robot relative pose estimation based on semidefinite relaxation optimization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4491–4498.

[30] Y. Tu, G. Liang, and T. L. Lam, "Graph convolutional network based configuration detection for freeform modular robot using magnetic sensor array," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4252–4258.

[31] X. Guo, J. Hu, J. Chen, F. Deng, and T. L. Lam, "Semantic histogram based graph matching for real-time multi-robot global localization in large scale environment," *IEEE Robotics and Automation Letters*, 2021.

[32] H. Zhang, T. Zhang, T. L. Lam, and S. Vijayakumar, "Posefusion2: Simultaneous background reconstruction and human shape recovery in real-time," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7631–7638.