


## Article

# Path Planning for Self-Collision Avoidance of Space Modular Self-Reconfigurable Satellites

Jiping An , Xinhong Li \*, Zhibin Zhang, Guohui Zhang, Wanxin Man, Gangxuan Hu and Junwei He

Department of Aerospace Science and Technology, Space Engineering University, Beijing 101416, China; ajp112233@alumni.sjtu.edu.cn (J.A.); zhangzhibinseu@163.com (Z.Z.); zhangguohui123@stu.xjtu.edu.cn (G.Z.); manwanxin@126.com (W.M.); hgx0609@163.com (G.H.); junwei.online@gmail.com (J.H.)

\* Correspondence: 13366159269@189.cn

**Abstract:** Space modular self-reconfigurable satellite (SMSRS) is a new type of satellite. The research on self-collision avoidance of SMSRS is important for its on-orbit safety but is not completely solved. This paper offers a new method for joint path planning for self-collision avoidance of SMSRS. Firstly, we establish the collision detection model for SMSRS based on forward kinematics and the spherical nonholonomic envelope to detect the collision occurring in SMSRS. Then, to achieve offline path planning in joint space, we proposed the self-collision avoidance strategy, which splices multiple C-spaces based on the pre-defined joint path into a binary map, and then transforms the binary map into a map with the dangerous potential field, and planning algorithms based on a map with the dangerous potential field is proposed to find the optimal collision-free path. The new method is applied to two cases and both find collision-free joint paths for SMSRS successfully, which demonstrates the feasibility of the method. In addition, this study bridges the gap in the study of self-collision avoidance of super-redundant self-reconfigurable satellites.

**Keywords:** self-collision avoidance; path planning; self-reconfigurable satellite; collision detection model



**Citation:** An, J.; Li, X.; Zhang, Z.; Zhang, G.; Man, W.; Hu, G.; He, J. Path Planning for Self-Collision Avoidance of Space Modular Self-Reconfigurable Satellites. *Aerospace* **2022**, *9*, 141. <https://doi.org/10.3390/aerospace9030141>

Academic Editor: Dario Modenini

Received: 19 February 2022

Accepted: 3 March 2022

Published: 5 March 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

With the large-scale exploitation of outer space and the increasing frequency of spacecraft and satellite launches, future space systems are expected to have low cost, rapid response, and multiple uses [1]. However, the traditional development mode cannot meet the needs of future space systems [2]. For example, traditional space systems are almost specially designed for missions, requiring subsystem-by-subsystem development and numerous iterations, which are slow, repetitive, and expensive [3]. In addition, the structures of these traditional spacecraft systems are essentially permanently fixed and designed to perform a single mission that cannot be reconfigured for other missions. Long development cycles and launch-window limitations also make it difficult for them to respond quickly to emergencies [3].

To overcome these limitations of existing space systems, many new concept satellites have been proposed, and the most representative ones are CubeSats [4,5]. The primary goal of CubeSats is to provide a small, lightweight, low-cost platform for academic and technology demonstration. A CubeSat is a 10 cm × 10 cm × 10 cm cube. It provides a standard for the design of picosatellites to reduce cost and development time, increase accessibility to space, and sustain frequent launches [6]. The spacecraft bus was designed to be versatile enough to accommodate different payloads that meet the mass, volume, and power constraints. Presently, the CubeSat Projects become attractive for wide fields [7,8], such as exoplanet detection [9,10], space weather [11], weather forecasting [12], and river-flow estimation [13].

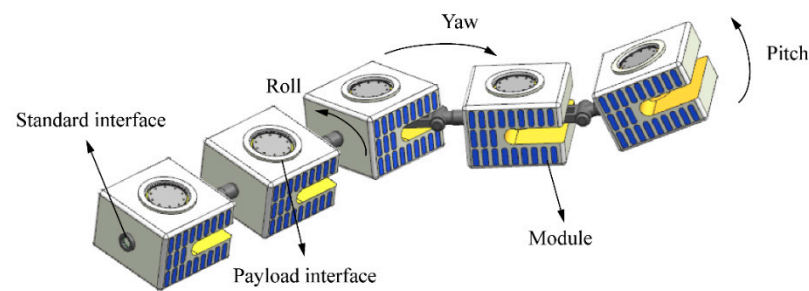
However, the utilization of a standard, low-cost satellite bus will reduce the reliability and decrease the lifetime of Cubesats [14,15]. A statistical study performed in 2013 on the

first 100 launched CubeSats [16] shows that the mission failures of CubeSats are on a high level [17]. In addition, the structure of CubeSat is fixed, and it carries limited payloads so that they can only perform a single space task.

Taking the design concept of CubeSat as the baseline, we proposed a new type of satellite with a reconfigurable structure and adjustable function, named Space Modular Self-Reconfigurable Satellite (SMSRS). SMSRS has the following features:

(1) Modular: SMSRS has a chain structure composed of functional modules and joints. Common satellite subsystems are integrated into unit modules and are called subsystem modules. The unit subsystem modules are installed with standard interfaces for payloads and joints. According to shapes and work positions, the payloads are designed as external structures carrying standard docking interfaces to the subsystem modules, or as unit payload modules. The modular design of SMSRS facilitates the structural folding and packaging before launch and the reconfiguration in space [18].

(2) Scalability: Payload and subsystem modules are functionally independent and physically separated from each other. The volumes and masses of module units are divided into several series from small to large, which can be selected according to the demand of payloads. All modules have standard interfaces, which are used to connect with modular joint modules for ground or in-orbit extension according to the type of space mission. The type and number of modules carried by SMSRS are configured to mission requirements. The model of the SMSRS with five modules is shown in Figure 1.



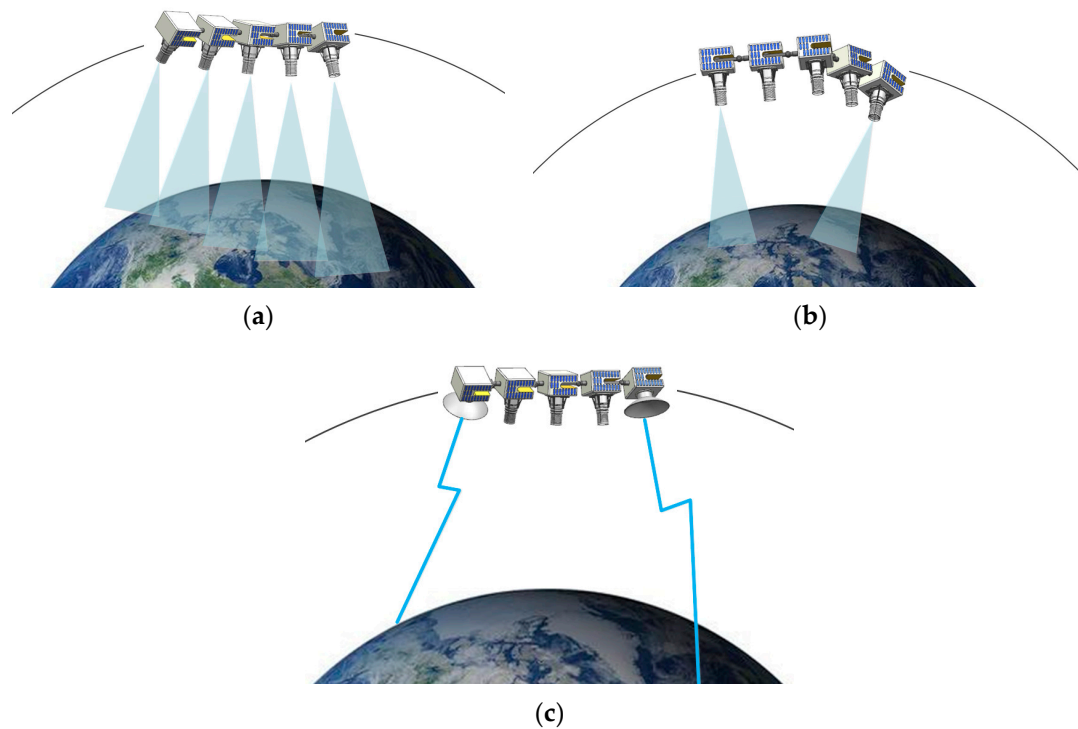
**Figure 1.** Model of SMSRS with five modules.

(3) Structural Reconfigurability: There are three mutually orthogonal joints between modules of SMSRS. The three joints form degrees of freedom (DOFs) between modules in relative rotation, roll, and yaw and enable the SMSRS to self-reconfigure adaptively to mission command.

(4) Risk resistance. The scalability and modularity together determine the risk resistance of SMSRS. Since modules are functionally independent, the failure of a local satellite module makes the satellite not completely fail, and the multiple payloads it carries can still be reprogrammed functionally. In addition, multiple joints between modules allow the modules to maintain some relative motion capability in the event of a single joint failure.

(5) Functional Adjustability: The types of payloads carried by SMSRS include optical cameras, SAR, communication payloads, etc. While carrying different payloads, SMSRS arranges and reorganizes these payloads in different space orientations by structural reconfiguration and therefore achieves the ability to accomplish multiple types of space missions. There are typical application scenarios for SMSRS:

- When SMSRS carries multiple optical cameras, it could change the spatial orientations of these cameras through joint motions. With this, these cameras could achieve an expanded imaging area by stitching together their field of view as shown in Figure 2a, or achieve the reconnaissance of multiple targets as shown in Figure 2b.
- When SMSRS carries multiple communication payloads and adjusts them towards different orientations, it can achieve multi-area communication to the earth, as shown in Figure 2c.



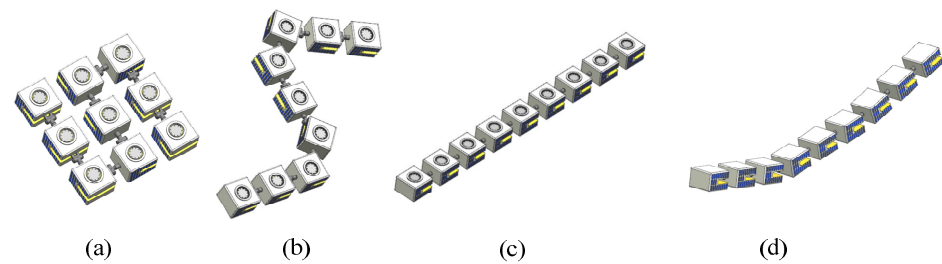
**Figure 2.** Application scenarios for SMSRS: (a) SMSRS carries multiple optical cameras and expands the imaging area by stitching together fields of view. (b) SMSRS reconnaissance of multiple targets by multiple optical cameras. (c) SMSRS carries multiple communication payloads to achieve multi-area communication to the earth.

The application scenarios of SMSRS are by no means limited to these, and there are larger expansions. Moreover, in recent years, a large number of small satellite launches have led to space traffic congestion and burdened space traffic management. At the same time, a large amount of space debris is generated, which seriously threatens the safety of satellites in orbit. The multi-functional feature of SMSRS, which allows a single satellite to perform multiple functions adaptively, can reduce the number of satellite launches. It reduces the pressure of space traffic management and the generation of space debris.

The target orbit of SMSRS is Low Earth Orbit (LEO). The advantage of standing high and seeing far in High Earth Orbit (HEO) can be compensated by the feature of SMSRS multi-payload synergy, e.g., the large field of view of HEO satellites can be achieved by using fields of view stitching of SMSRS. In addition, LEO launches can launch heavier mass satellites at a lower cost, providing margin for SMSRS in-orbit scalability.

When running on LEO, the satellite tracking is characterized by low transmission delay, low coverage, low link loss, and low power consumption. The low transmission delay and low loss are beneficial to the command transmission of the reconfiguration for SMSRS, and in solving the low coverage problem, as shown in Figure 2c, adjusting the orientation of the tracking system can be used to expand the tracking range.

As shown in Figure 3, SMSRS are folded and packed on the ground before launch. After getting into orbit, the folded configuration is gradually unfolded and changed from the initial configuration to the work configuration for performing the designed mission. When the SMSRS receives a new mission command, its mission planning system will plan the new mission configuration and send the joint control system command for reconfiguring to the new work configuration. Modules of SMSRS have solar cells attached to each surface to ensure battery recharging in different configurations.



**Figure 3.** Configurations of SMSRS from the folded state to unfolded state and work state. (a) Folded state, (b) Unfolding, (c) Unfold state, (d) Work state.

The function switching of SMSRS depends largely on the motion of the joints. However, from Figure 1, it can be seen that the modules of SMSRS are large in size but small in spacing between each other, and the motion of the joints allows the relative positions of these modules to change over a wide range so that collisions between modules can easily occur. When a collision occurs, it is likely to cause structural damage and instability of the system [19]. To avoid self-collisions between modules in whole reconfiguration progress is crucial for the on-orbit safety of the system [20]. In this paper, we will study how to achieve self-collision avoidance in the reconfiguration of SMSRS.

The remaining paper is organized as follows. Section 2 sorts out the related work on self-collision avoidance of self-configuring systems. In Section 3, the collision detection model of SMSRS is established. Section 4 elaborates on the self-collision strategy of SMSRS. In Section 5, the parameters of SMSRS and two self-collision avoidance tasks are set. Section 6 verifies the self-collision avoidance path planning method. Section 7 summarizes the work of this paper.

## 2. Related Work

The researches on avoiding self-collision are mainly for redundant robotic arms, and the contents mainly involve two aspects: self-collision detection and self-collision avoidance strategies. Collision detection focuses on establishing an approximate geometric model of the collision-prone structures of the research object to calculate the relative position between them [21] and to judge whether they will collide or have collided. There are two common methods to establish the collision detection model. The first method uses a large number of polygons [22,23] to model the geometrical shapes of collision-prone structures to fit them as closely as possible. This method is generally suitable for systems that have an irregular shape and higher requirements for collision detection accuracy, or are used to perform sophisticated tasks, but it requires higher computational costs. For systems with regular shapes or have low requirements for collision detection accuracy, using the sphere-based model to establish its approximate structural envelope is another method [20,21]. These approximate envelopes can be selected according to the shape of the system, including spheres [21,24], swept-spheres [25], cylinders [20], capsules [26], patch-based bounding volumes [27], etc. Although the second method could not accurately describe the shape of the collision-prone structures, it is computationally economical.

In the present research, self-collision avoidance strategies are usually divided into the online reactive approach [27,28] and offline path planning. Of the two, the online reactive approach does not predetermine the angle motion path but detects collisions in real time during joint movements. Online reactive approaches usually use the distance functions as “repulsive potential” functions and then converts them into joint velocities based on Jacobian Inverse Kinematics (IK) [26] and joint torques for torque control schemes [29], or regards them as a constraint in an optimization-based IK solver to adjust the joint motion path in real time [28,30], etc. To obtain smooth joint motion trajectories, some researchers have proposed a unified framework to detect collisions by using a series of sensors capable of estimating joint torques and accelerations in real time. However, the use of expensive hardware such as sensors always increases the cost [21]. Although the online reactive

approach has a shorter computation time, it is mostly applied to ground-based humanoid robots and is only applicable to systems with fewer DOFs because of its instability.

Offline path planning can be divided into two categories: path planning of end-effector in workspace [31,32] and path planning of joints in joint space [33,34]. As there is no end-effector of SMSRS, we do not discuss the path planning of the end-effector, but focus on the path planning of the joints in joint space. In contrast to the online method, offline path planning predetermines the motion path of joints and there would not be self-collision when joints move along the predetermined path. Compared with the online reactive approach, offline path planning is more stable and efficient, and the planned paths are guaranteed to be optimal, which are more suitable for space systems with higher requirements for stability. Moreover, for SMSRS, offline path planning will not make its joints detour in space, which is also of great significance for energy conservation. Therefore, offline path planning is the more suitable choice for the self-collision avoidance strategy of SMSRS.

Currently, offline path planning for collision avoidance of self-configuration systems such as space robotic arms is mostly focused on avoiding collision with fixed obstacles or target vehicles in the environment [20,35]. The methods to avoid collision with obstacles in the environment include using a random search algorithm to find a collision-free joint path in the Configuration Space (C-Space) of the robotic arm, and commonly used algorithms include the A\* algorithm [36] and the Rapid Exploration Random Tree (RRT) algorithm [37–39], and using optimization techniques [40], etc. In addition, the current offline path planning only needs to find collision-free joint paths to achieve the desired pose of the end effector, regardless of the pose of other intermediate structures [20]. The researches on self-avoidance are also mainly focused on systems with fewer DOFs. Therefore, current research cannot meet the requirements of self-collision path planning of SMSRS, mainly because it has the following newly derived difficulties:

- (1) SMSRS has a super-redundant structure, and its ultra-high DOFs make collisions between each module possible and easy;
- (2) The pose of each module of SMSRS is changed in real time during reconfiguration, for one module, all other modules are its dynamic obstacles—this also imposes a burden on collision detection; and
- (3) Different from the robots that only achieve the pose of the end effector without collision, SMSRS requires all modules to achieve desired poses without collision after path planning.

In this paper, considering the above difficulties and the actual space mission requirements, we proposed a new self-collision avoidance method for SMSRS, which consists of a collision detection model and a self-collision avoidance strategy. The collision detection model is established based on the Forward Kinematics (FK) and spherical nonholonomic envelope, which has no hardware cost and low computational cost. Among the offline and online strategies, considering the high stability requirements of the SMSRS and reducing additional hardware as much as possible, the offline path planning is selected to realize self-collision avoidance of SMSRS, and a self-collision avoidance path planning strategy based on pre-defined path and spliced C-space is proposed.

The main contributions of this study are as follows: (1) a collision detection model is developed using multiple discrete spherical, which can approximate envelop SMSRS using a few spheres and execute collision detection for SMSRS at a low computational cost; (2) a method is proposed for splicing multiple C-spaces into a binary map based on a pre-defined path, which transforms the path planning in dynamic C-space into the planning of time sequence of joints on the pre-defined path; and (3) a new collision-free path search algorithm based on the map with dangerous potential fields is proposed to find the safest path without self-collision for SMSRS.

### 3. Collision Detection of SMSRS

The collision detection model of SMSRS is used to detect whether each module collides with others, and the relative poses of modules are the basis of collision detection. SMSRS

consists of a series of rigid bodies and its modules and inter-module connection structures can also be regarded as links, and this structure forms a kinematic chain. We firstly ignore the specific shape of modules and treat them as links, and calculate their space poses by establishing an FK model in Section 3.1. On this basis, the collision detection model will be completed after assigning the shape characteristics of these links in Section 3.2.

### 3.1. Module Pose

In SMSRS, each moveable joint defines a joint variable, all joints form a joints vector  $\mathbf{q} = (\theta_1, \theta_2, \dots, \theta_{N_j})$  and the number of joints  $N_j$  defines DOFs of SMSRS. Calculating the poses of links using the given values of the joint variables and the link parameters is a forward problem, which is known as FK [41,42]. In FK, the poses of links can be represented by a homogeneous transformation matrix. For example, the homogeneous transformation matrix [43] of the  $i$ th link relative to the base coordinate system  $\Sigma_b$  is:

$${}^0T_i = \begin{bmatrix} n_{xi} & o_{xi} & a_{xi} & p_{xi} \\ n_{yi} & o_{yi} & a_{yi} & p_{yi} \\ n_{zi} & o_{zi} & a_{zi} & p_{zi} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} [ \mathbf{n}_i & \mathbf{o}_i & a_i ] & \mathbf{p}_i \\ \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i & \mathbf{p}_i \\ \mathbf{0} & 1 \end{bmatrix} \quad (1)$$

where,  $\mathbf{R}_i$  represents the attitude matrix and  $\mathbf{p}_i$  presents the position vector of the  $i$ th link. To obtain the homogeneous transformation matrix, we should establish the multi-link coordinate system according to certain rules, then select the parameters to describe relative relationships between coordinate systems. In this paper, we select Denavit–Hartenberg (D–H) method to build the homogeneous transformation matrix. According to the principles of the D–H method, we fix every link with a coordinate system as shown in Figure 4, the  $\Sigma_b$  is usually established at the center of mass of the attitude control module to facilitate the attitude control, and then the multi-link coordinate system of SMSRS is divided into  $a$  and  $b$  side by  $\Sigma_b$ . In some space systems, such as space dual-arm robots, the joint axis on both sides of  $\Sigma_b$  are always designed to be symmetrical in direction, so the D–H parameters on both sides are equal [44]. However, the three joints between the modules are orthogonal and are asymmetrical on both sides of  $\Sigma_b$ , which violates principles of the D–H method on establishing coordinate systems. In this case, we proposed the concept of virtual joint coordinate system  $Jv$  [18].  $Jv$  is a coordinate system established between the  $\Sigma_b$  and the first joint coordinate system on the  $b$  side. The origin of the  $Jv$  coincides with the origin of  $\Sigma_b$ . Then, the pose matrix of the link on the  $b$  side only needs to multiply the homogeneous transformation matrix from the  $Jv$  to  $\Sigma_b$  between the first joint coordinate system and  $\Sigma_b$ . Moreover,  $Jv$  does not affect the mass and size of SMSRS when its mass and size are set to 0. The insertion of  $Jv$  successfully solves the problem caused by the asymmetrical direction of joints on  $a$  and  $b$  sides.

Then, four parameters between adjacent coordinate systems: link length  $A$ , link twist  $\alpha$ , link offset  $d$ , and joint angle  $\theta$  are measured to calculate their relative relationship. In FK, any single translation or rotation can be expressed by a homogeneous transformation matrix, and a series of motions can be expressed by multiplying these homogeneous transformation matrices of single motions. The four selected parameters imply four-step motions from the  $(i - 1)$ th coordinate system to the  $i$ th coordinate system, so the homogeneous transformation matrix between them can be expressed as:

$$\begin{aligned}
 {}^{i-1}T_i &= Rot(x, \alpha_{i-1}) \cdot Tran(A_{i-1}, 0, 0) \cdot Rot(z, \theta_i) \cdot Tran(0, 0, d_i) \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & A_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & A_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned} \tag{2}$$

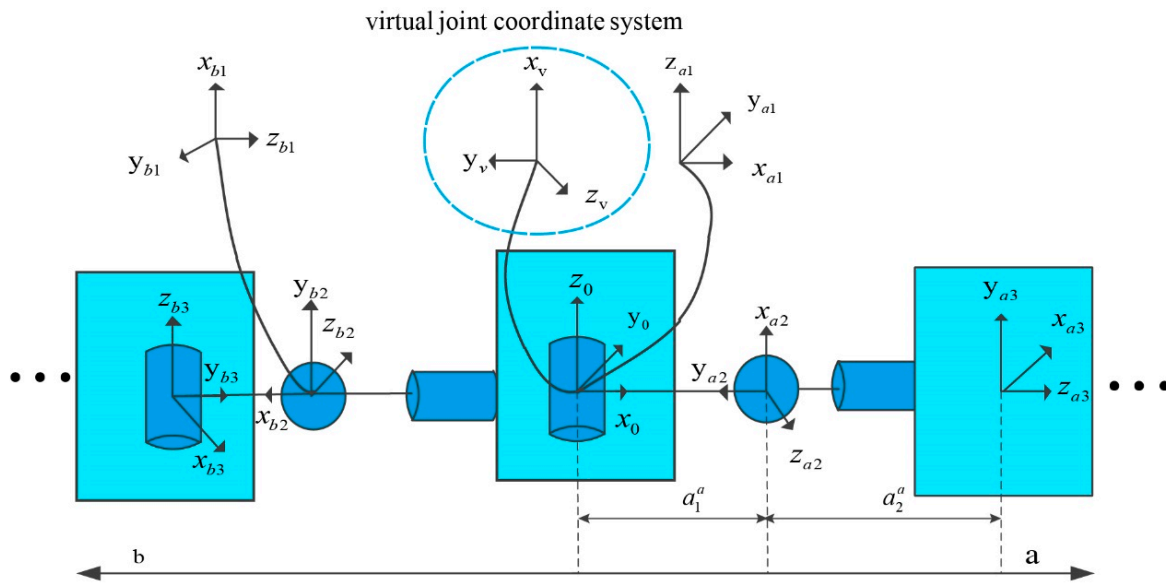


Figure 4. The link coordinate system of SMSRS.

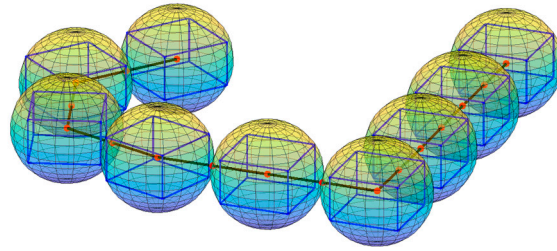
The relative poses of non-adjacent links can be obtained by the successive multiplications of adjacent link homogeneous transformation matrices [43]. Equation (3) shows the homogeneous transformation matrix of the  $i$ th link relative to  $\Sigma_b$ .

$${}^0T_i = {}^0T_1(\theta_1) {}^1T_2(\theta_2) {}^2T_3(\theta_3) \cdots {}^{i-1}T_i(\theta_i) = \prod_{k=1}^i {}^{k-1}T_k(\theta_k) \tag{3}$$

### 3.2. Self-Collision Detection Model

#### 3.2.1. Spherical Nonholonomic Envelope

A complete self-collision detection model needs to assign shape properties to links based on the obtained link poses. In this paper, the spherical shape is chosen to envelop the modules of SMSRS, which has the following advantages: (1) the single parameter can avoid a large amount of computational consumption and improve the computational speed. (2) The fixed shape of SMSRS modules can be completely enveloped by spheres, and the approximate nature of the spherical envelope makes the collision detection conservative [24], which is suitable for space systems where reliability and safety are vital. Since the volume of the module is much larger than the volume of the inter-module joint structure, we create a spherical envelope only for the module and ignore the joint structures. The spherical nonholonomic envelope could further improve computational efficiency. And the spherical nonholonomic envelope of SMSRS with nine modules is shown in Figure 5.

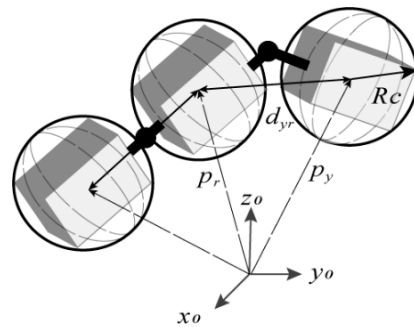


**Figure 5.** Spherical envelope of SMSRS with nine modules.

### 3.2.2. Self-Collision Detection Criteria

Figure 6 shows the schematic diagram of the spherical nonholonomic envelope of the SMSRS with three modules. The center of the spherical envelope locates at the center of the module body, and the length of the radius  $R_C$  is one half of the diagonal length of the module body. In this paper, the distance between the centers of two modules is used to determine whether modules collide. For SMSRS,  $\Psi$  is defined as the set of modules. Select two different modules  $y, r \in \Psi, (y \neq r)$ , their position vectors with respect to  $\Sigma_b$  are  $\mathbf{p}_y$  and  $\mathbf{p}_r$ ,  $d_{yr}$  denotes the distance between the centers of the two modules, and it equals to the 2-norm of the difference between  $\mathbf{p}_y$  and  $\mathbf{p}_r$ . The criteria for determining modules  $y, r$  have not collided is:

$$d_{yr} = \|\mathbf{p}_y - \mathbf{p}_r\| > 2R_C \quad (4)$$



**Figure 6.** Schematic diagram of the collision detection model.

For a certain configuration of SMSRS, only if  $d_{yr}$  of any two modules is larger than  $2R_C$ , it can be concluded that SMSRS has no collision under this configuration, and as long as there exist any two modules that collide, this configuration has collision.

## 4. Self-Collision Avoidance Strategy

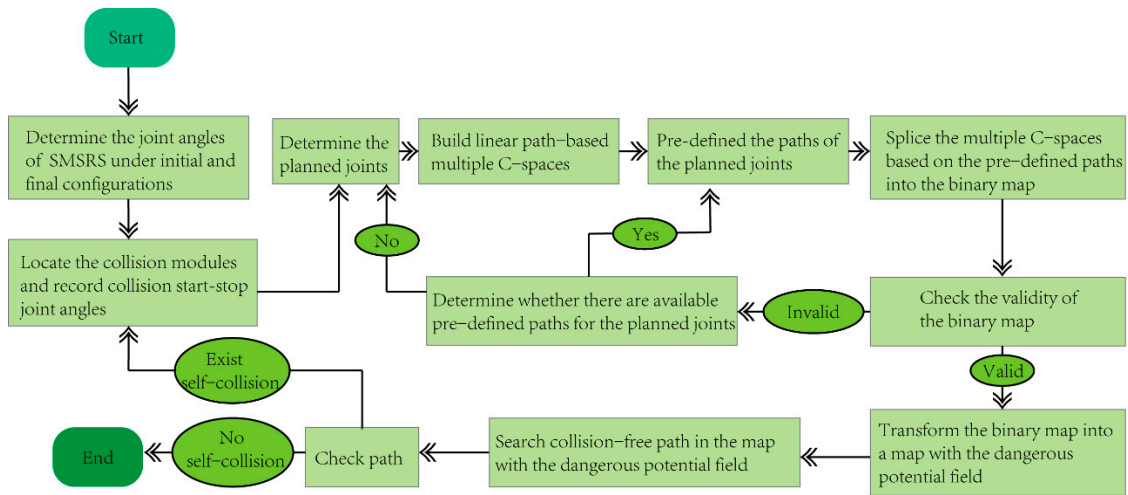
Collision avoidance is a fundamental problem in offline path planning, and a two-step framework is widely used in collision-free path planning for robotic arms with few DOFs and mobile robots, i.e., building the C-space and then searching for collision-free paths in the C-space. For the offline collision-free problem of SMSRS in time-varying configurations, we propose the spliced C-space based on pre-defined joint paths and the path planning algorithms based on digital maps with the dangerous potential field, which together consist of the self-collision avoidance strategy of SMSRS. It is implemented in several steps as follows to achieve the self-collision avoidance strategy of SMSRS:

- (1) Locate the collision module and record collision start-stop joint angles;
- (2) Determine the planned joints;
- (3) Build linear path-based multiple C-spaces;
- (4) Pre-defined the paths of the planned joints;
- (5) Splice the multiple C-spaces based on the pre-defined paths into the binary map, and check the validity of the binary map;



- (6) Transform the binary map into a map with the dangerous potential field;
- (7) Search collision-free path in the map with the dangerous potential field; and
- (8) Check paths.

The logical relationships of these steps are shown in Figure 7, where steps (1) to (2) are for initialization, (3) to (5) are for C-space construction, and (6) to (8) are for path planning algorithms based on digital maps with the dangerous potential field. The idea and execution process of each step will be described below.



**Figure 7.** Steps of self-collision avoidance strategy of SMSRS.

#### 4.1. Locate the Collision Module

The initial configuration and final configuration with corresponding joint angles are given by the mission planning system. The results of the path planning of SMSRS, on a self-collision-free basis, need to satisfy the pose requirements of each link, not only the desired poses of the end link while ignoring the poses of the other links. Therefore, the goal of the path planning of SMSRS is to find feasible paths for all joints so that the SMSRS can move from the initial configuration to the final configuration without collision. We set all joints of SMSRS to move in linear paths from initial angles to final angles, and locate the collision modules under their linear paths. If there are no additional optimization objectives, the linear path is the most economical path, which avoids the energy waste caused by joints moving in complex paths and could facilitate the subsequent joint trajectory optimization.

Under the linear paths of joints, the collision detection model in Section 3.2 is used to determine whether there is a collision between modules in real time, and once a module collision is detected, the two modules are marked as collision modules, and the collision start angles of joints  $q_{col\_start}$  are recorded. The final configuration of the SMSRS is collision-free, indicating the collision state will not last until the end, allowing the SMSRS to continue moving and recording the collision stop angles of joints  $q_{col\_end}$ .

#### 4.2. Determine the Planned Joints

Once the collision module and the  $q_{col\_start}$  and  $q_{col\_end}$  are clarified, there are two options for the selection of the joints to be planned: planning the path of all joints or planning the path of only some selected joints. In this paper, the paths of two joints are selected to plan to avoid the self-collision, which is based on the following considerations: (1) the path planning with few joints can reduce the planning complexity and computational cost, while a large number of stable and efficient path search algorithms for planes can be utilized, which also improves the visibility; (2) SMSRS has wide ranges of joint angles, and by reasonably configuring the two planned joints, the configurations of SMSRS can be substantially adjusted so that new collision-free paths can be found in the collections

of these configurations; and (3) each joint needs to return to the desired angle after path planning, and when multiple joint angles are planned at the same time, it is difficult to ensure that.

The principle of configuring the planned joints is the maximum success rate and maximum stability. The so-called maximum success rate means that the selected joint can find the collision-free path as successfully as possible after planning. Maximum stability means that the planned joint will not significantly change the configuration of the collision-free module, resulting in unnecessary collisions. Combining these two principles, we develop the following selection rules for the planned joints:

1. For  $i$ th joint, when its  $\theta_{i\_col\_start} \neq \theta_{i\_col\_end}$ , we call it motion joint, otherwise no motion joint—when there are more than two motion joints between two collision modules, the two adjacent motion joints closest to the end of SMSRS are selected as the planned joints;
2. When the selected joints cannot successfully avoid the self-collision after planning, another two adjacent motion joints between collision modules are selected from the end to the middle of SMSRS;
3. When the number of motion joints between collision modules is less than two, a no motion joint closest to the end collision module and a motion joint are selected as planned joints;
4. Under (3), if it is still not able to find self-collision-free paths, the motion joint and no motion joint are configured from the end to the middle of SMSRS; and
5. After selection, we denote the planned joints as  $\theta_1^p, \theta_2^p$ , their start collision angles as  $\theta_{1\_start}^p, \theta_{2\_start}^p$ , and end collision angles as  $\theta_{1\_end}^p, \theta_{2\_end}^p$ .

#### 4.3. Splice Multiple C-Spaces Based on a Pre-Defined Path

##### 4.3.1. Multiple C-Spaces

The concept of C-space originates from collision detection and avoidance of robotics [12,13], which uses the pose parameters of rigid objects in the workspace as its coordinate parameters, and the kinematic relations map the poses of target objects and obstacles from the workspace to the C-space.

C-space is often used for path planning of rigid objects in stationary obstacles. Therefore, C-space can be further divided into two subspaces: obstacle space and free space. The obstacle space refers to the space occupied by obstacles in the C-space, while the free space refers to the space not occupied by obstacles in C-space. Unlike the physical workspace, the dimensions of the C-space are defined as the DOFs of all control variables in the motion system [45]. A schematic of a two-dimensional C-space is shown in Figure 8. When the boundaries of the obstacle space and the free space in the C-space are determined, collision-free path planning is to search for a path in the C-space that connects the start point and end point and does not pass through the obstacles.

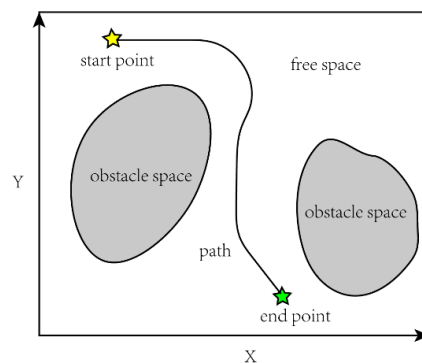


Figure 8. Schematic of a two-dimensional C-space.

For SMSRS, a configuration  $\mathbb{C}$  can be defined by the  $N_j$ -dimensional joint vector  $\mathbf{q}$ , with  $\mathbf{q}$  denoting the current state of each joint. At this point,  $\mathbb{C}$  can be designated as a point in the  $N_j$ -dimensional C-space of SMSRS, and the boundary of this convex C-space is defined by the upper bound  $\mathbf{u}_b$  and the lower bound  $\mathbf{l}_b$  of joint angles.

$$\begin{aligned} \mathbf{l}_b &= \left[ \theta_{1_{min}} \quad \theta_{2_{min}} \quad \theta_{3_{min}} \quad \cdots \quad \theta_{N_j_{min}} \right]^T \\ \mathbf{u}_b &= \left[ \theta_{1_{max}} \quad \theta_{2_{max}} \quad \theta_{3_{max}} \quad \cdots \quad \theta_{N_j_{max}} \right]^T \end{aligned} \tag{5}$$

However, in Section 4.2 we illustrate selecting only two joints for path planning, so there is not necessary to build an  $N_j$ -dimensional C-space of SMSRS, but only a two-dimensional C-space with the angles of planned joints as coordinate parameters. At a certain moment  $t$ , SMSRS has a configuration  $\mathbb{C}_t$ . If a collision occurs in SMSRS under  $\mathbb{C}_t$ , the point  $(\theta_1^p(t), \theta_2^p(t))$  belongs to the obstacle space of the C-space, and if no collision occurs, the point belongs to the free space. Then, the first task is to map the obstacles in the workspace space to C-space. In SMSRS, all non-planned joints will be used to build the obstacle space.

However, unlike the C-space with fixed obstacles, the C-space of SMSRS is with dynamic obstacles as modules are in motion with time, so it is impossible to use computational methods to calculate its obstacle space, this paper uses the random sampling method to determine the obstacle space of the C-space of SMSRS.

At a moment, we could fix the angles of non-planned joints and build the C-space of the planned joints by the random sampling method. While the angles of non-planned joints also need to move with time, we propose the concept of spliced C-space. We select multiple time points in the collision process of SMSRS uniformly, build the C-space at each time point, and splice them together into a spliced C-space. The operation procedures are:

(1) Moderate angle allowance is added into  $\mathbf{q}_{col\_start}$  and  $\mathbf{q}_{col\_end}$  to ensure the states of SMSRS before and after planning are self-collision-free. Assuming that the total time of the collision process is  $T$  and joint moves from the  $\mathbf{q}_{col\_start}$  to  $\mathbf{q}_{col\_end}$  linearly, the collision process is equated into  $N$  time sequences. The start moment of each time sequence is  $T_t, t = 1, 2 \cdots N$ , and the joint angles of SMSRS at  $T_t$  is  $\mathbf{q}_{T_t}$ . The determination principle of  $N$  is:  $N = \max\left(\left(\theta_1^p\right)_{max} - \left(\theta_1^p\right)_{min}, \left(\theta_2^p\right)_{max} - \left(\theta_2^p\right)_{min}\right)$  to keep it in a reasonable interval.

(2) At  $T_t$ , randomly sample  $(\theta_1^p(T_t), \theta_2^p(T_t))$  with a sampling number  $\vartheta$ , and the sample intervals are for  $\theta_1^p(T_t), \theta_2^p(T_t)$  are  $\left(\left(\theta_1^p\right)_{min}, \left(\theta_1^p\right)_{max}\right), \left(\left(\theta_2^p\right)_{min}, \left(\theta_2^p\right)_{max}\right)$ , respectively, put all the sampled points into the matrix  $\mathbb{C}_{T_t} \in \mathbb{R}^{\vartheta \times 2}$  in order.

(3) At  $T_t$ , the angles of the planned joints at the corresponding positions in  $\mathbf{q}_{T_t}$  are replaced by each set of  $\theta_1^p(T_t), \theta_2^p(T_t)$  in  $\mathbb{C}_{T_t}$  to form a new joint angle sequence. The new joint angle sequence is saved to the matrix  $\mathbb{B}_{T_t} \in \mathbb{R}^{\vartheta \times N_j}$ .

(4) At  $T_t$ , each  $\mathbf{q}_{T_t}$  in  $\mathbb{B}_{T_t}$  is brought into the FK of SMSRS to calculate the module poses, and the self-collision detection model in Section 3.2 is used to judge whether there is a collision occurring in SMSRS under  $\mathbf{q}_{T_t}$ , if there is a collision occurring, the point  $(\theta_1^p(T_t), \theta_2^p(T_t))$  are counted into the obstacle space, otherwise, it is counted into the free space. Until each  $\mathbf{q}_{T_t}$  in  $\mathbb{B}_{T_t}$  are judged, the work to build C-space at  $T_t$  is finished. Figure 9 shows a schematic diagram of C-space built by random sampling with  $\vartheta = 40,000$ , from which we can see that as long as the sampling points are sufficient, the boundary of the obstacle space is clear.

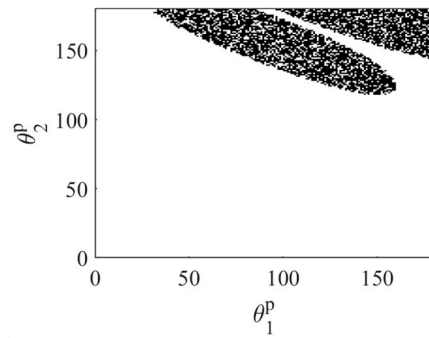


Figure 9. Two-dimensional C-space based on random sampling.

(5) Follow steps 2 to 4 to build the C-space at each moment until obtaining the  $N$ -multiplicity two-dimensional C-space.

### 4.3.2. Pre-Defined Path

To find self-collision-free joint paths from the  $N$ -multiplicity C-spaces, the primary is to splice the  $N$ -multiplicity C-spaces according to certain principles. In this paper, we propose a method for splicing  $N$ -multiplicity C-spaces based on a predefined path of planned joints. The idea is to predefine a joint path and splice  $N$ -multiplicity C-spaces into a binary map along the predefined path. In the binary map, searching for a collision-free path is essentially the planning of the time sequence of the pre-defined path. Thus, the path planning of joints in  $N$ -multiplicity C-spaces is transformed into the planning of its time sequence on a pre-defined path in spliced C-space. Firstly, we propose a method to pre-define the path of the planned joint from  $\theta_{1\_start}^p, \theta_{2\_start}^p$  to  $\theta_{1\_end}^p, \theta_{2\_end}^p$ , which is shown in Figure 10.

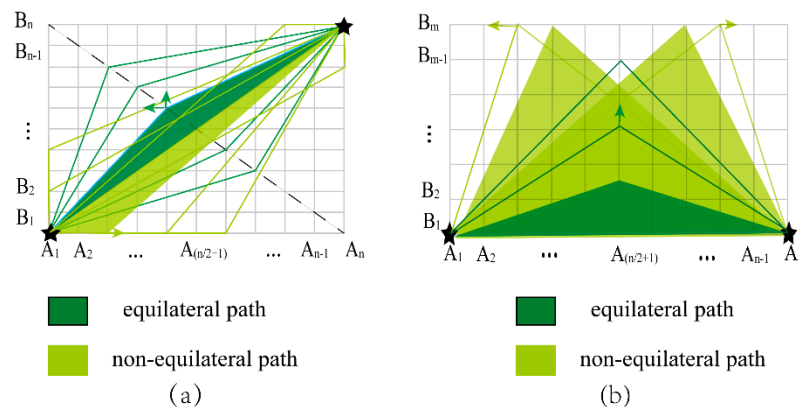


Figure 10. Schematic diagram of pre-defined three-angle paths: (a) the pre-defined path for two planned joints that are both motion joints, (b) the pre-defined path method when one of the planned joints is no motion joint.

Figure 10a shows the pre-defined path for two planned joints that are both motion joints. Firstly, the angles of planned joints from  $(\theta_{1\_start}^p, \theta_{2\_start}^p)$  to  $(\theta_{1\_end}^p, \theta_{2\_end}^p)$  are divided into  $n$  equal intervals,  $n = \max\|\theta_{1\_end}^p - \theta_{1\_start}^p, \theta_{2\_end}^p - \theta_{2\_start}^p\|$ . The  $(A_1, B_1)$  point is the start point of the path, representing the start collision angle of the planned joint  $(\theta_{1\_start}^p, \theta_{2\_start}^p)$ .  $(A_n, B_n)$  is the end of the path, representing the end collision angle of the planned joint  $(\theta_{1\_end}^p, \theta_{2\_end}^p)$ . The default pre-defined path is linear:  $(A_1, B_1) \rightarrow (Ver_x, Ver_y) \rightarrow (A_n, B_n)$ , and it forms a triangle with  $(A_1, B_1) \rightarrow (A_n, B_n)$ .  $(Ver_x, Ver_y)$  is the apex of the obtuse angle, and it has the following three kinds of moving rules:

(1) Equilateral path: along with the  $(A_n, B_1) \rightarrow (A_1, B_n)$  line,  $Ver_x$  move from the  $A_{(n/2+1)}$  to both sides of  $A_{(n/2+1)}$ ,  $Ver_y$  move from  $B_{(n/2+1)}$  to both sides of  $B_{(n/2+1)}$ .

(2) Non-equilateral path:  $Ver_y = B_n$ ,  $Ver_x$  move from  $A_n$  to  $A_1$ ; or  $Ver_x = A_n$ ,  $Ver_x$  move from  $B_n$  to  $B_1$ .

(3) Non-equilateral path:  $Ver_y = B_1$ ,  $Ver_x$  move from  $A_1$  to  $A_n$ ; or  $Ver_x = A_1$ ,  $Ver_x$  move from  $B_1$  to  $B_n$ .

Figure 10b shows the pre-defined path method when one of the planned joints is no motion joint. Firstly, the angles of motion joints are divided into  $n$  equal intervals from  $\theta_{1\_start}^p$  to  $\theta_{1\_end}^p$ , and  $\theta_{2\_end}^p = \theta_{2\_start}^p$ ,  $n = \|\theta_{1\_end}^p - \theta_{1\_start}^p\|$ . The  $(A_1, B_1)$  point is the start point of the path, representing the start collision angle of the planned joint  $(\theta_{1\_start}^p, \theta_{2\_start}^p)$ .  $(A_n, B_1)$  is the end of the path, representing the end collision angle of the planned joint  $(\theta_{1\_end}^p, \theta_{2\_end}^p)$ . The side length of both X and Y directions of a grid are equal. The default pre-defined path is linear:  $(A_1, B_1) \rightarrow (Ver_x, Ver_y) \rightarrow (A_n, B_1)$ , and it forms an equilateral triangle with  $(A_1, B_1) \rightarrow (A_n, B_1)$ .  $(Ver_x, Ver_y)$  has two kinds of moving rules:

(1) Equilateral path:  $(Ver_x, Ver_y)$  move along  $(A_{(n/2+1)}, B_2) \rightarrow (A_{(n/2+1)}, B_m)$ .  $B_m \in (\theta_{2\_min}^p, \theta_{2\_max}^p)$ .

(2) Non-equilateral path:  $Ver_x$  move from  $A_{(n/2+1)}$  to both sides of  $A_{(n/2+1)}$  and  $Ver_y$  move from  $B_1$  to  $B_m$ .  $B_m \in (\theta_{2\_min}^p, \theta_{2\_max}^p)$ .

We restrict  $(Ver_x, Ver_y)$  can only be located on the grid points in Figure 10, it generates a new pre-defined path for each step forward on the grid. In Figure 10a, if there is no collision-free path in the spliced C-space generated by the current pre-defined path, a new path needs to be pre-defined according to rules (1)~(3) successively, and the pre-defined path needs to be guaranteed the obtuse angle is greater than 100 degrees; In Figure 10b, if all pre-defined paths in rule (1) are proved to be invalid, a new path shall be pre-defined according to rule (2), and the value of the two acute angles of the path should be less than 80 degrees. As shown in Figure 7, if all the compliant pre-defined paths of the currently planned joints are proved invalid, the progress needs to return to the step of determining planned joints.

#### 4.3.3. Splice Multiple C-Space

Finding the time sequence for the planned joints on the pre-defined path with no collisions is our inspiration for splicing  $N$ -multiplicity C-spaces. Equate the pre-defined path into  $S$  segments. To ensure the length of the path segment is equal to the C-space grid,  $S = \max(A_n - A_1), (B_n - B_1)$ . Create an  $N \times S$  dimensional matrix  $\mathbb{Z}$ , calculate the point of joint angles  $(\theta_{1\_j}^p, \theta_{2\_j}^p)$ ,  $j \in 1, 2 \dots S$  of the start point of each segment on the predefined path. For  $j$ th point, judge whether it is in free space or obstacle space of the  $i$ th multiplicity C-space, and, if it is in the free space,  $\mathbb{Z}(i, j) = 0$ ; otherwise  $\mathbb{Z}(i, j) = 1$ . After finishing the judgments, the  $N$ -multiplicity C-spaces based on the pre-defined paths are spliced into matrix  $\mathbb{Z}$ , which can be regarded as a two-dimensional digital map composed of the numbers 1 and 0 and is named as the binary map.

The  $N$  rows of the binary map present the  $N$ -multiplicity C-spaces and the  $S$  columns present the  $S$  segments of whole movement time along the pre-defined path. If we could find an all-0 path from the top-left to the bottom-right in the binary map, it means that the planned joints could cross the  $N$ -multiplicity C-spaces along the pre-defined path collision-freely under a specific time sequence. A binary map is valid if there exists an all-0 path from the top-left to the bottom-right; otherwise, the map is invalid, and a pre-defined path needs to be reselected. We propose an all-0 path search algorithm based on the total path length to determine whether the binary map is valid.

In Figure 11, the binary map is plotted into a map with grids. Every grid has two values, the top value represents its binary value, and the bottom value represents

the binary sum of all the grids on the optimal path pushed to the current grid from the start point, denoted by  $\bar{U}$ .

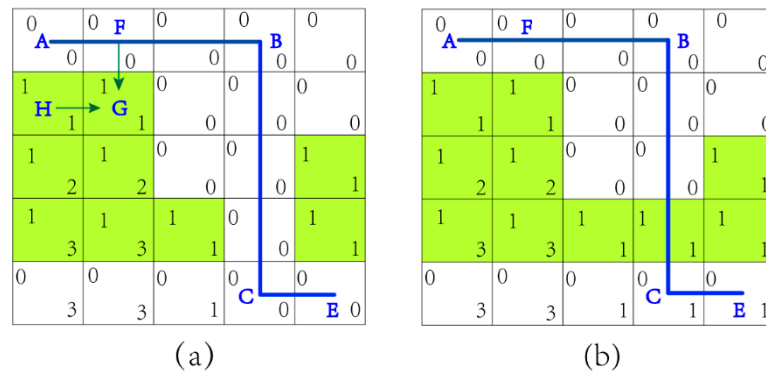


Figure 11. Valid binary map and invalid binary map: (a) valid binary map, (b) invalid binary map.

We specify the path to a grid can only be extended from its upper grid and left grid, e.g., the path to grid G can only be extended from grid F or H, and  $\bar{U} = 1$  for  $A \rightarrow H$ ,  $\bar{U} = 0$  for  $A \rightarrow F$ . Therefore, the path from grid F is the more optimal path for grid G. Based on this method, the steps of the all-0 path search algorithm based on the total path length are:

- (1) In a binary map, set the top-left grid as the start point and the bottom-right grid as the end point, if one of their binary values is not 0 or none is 0, the binary map is invalid and the pre-defined path should be adjusted to redraw the binary map.
- (2) The path to a grid can only be extended from its upper grid and left grid. Based on this principle, we first calculate the  $\bar{U}$  of grids in the first row and first column from the start point down and right, respectively. Then, in left to right and top to bottom order, we continue to calculate  $\bar{U}$  of each grid until reaches the end point.
- (3) Judge whether  $\bar{U}$  of the end point is 0, if  $\bar{U} = 0$ , as shown in Figure 11a, it proves that there exists an all-0 path from the start point to the endpoint. If  $\bar{U} \neq 0$ , as shown in Figure 11b, it proves that there exists no all-0 path from the start point to the end point, the map is invalid.

#### 4.4. Path Planning Algorithms

##### 4.4.1. Dangerous Potential Fields

As shown in Figure 12a, a valid binary map may have several collision-free paths from the start point to the end point, to select the optimal path from them, a collision-free path planning algorithm based on a Map with Dangerous Potential Fields (MDPF) is proposed.

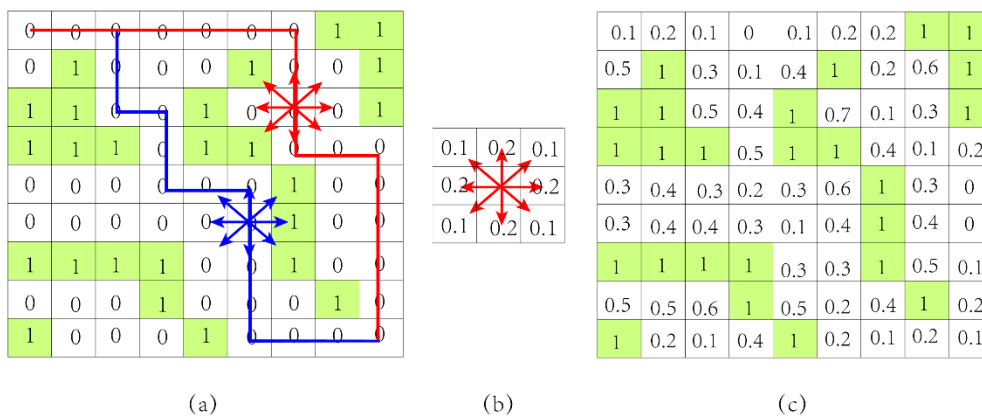


Figure 12. Conversion of the binary digital map to map with the dangerous potential field: (a) a valid binary map with several collision-free paths, (b) effect weights of the dangerous potential field to the center grid, (c) map with the dangerous potential field.

The dangerous potential field of the map is used to evaluate the possibility of potential collisions of its grids. As shown in Figure 12b, a grid is centered on the eight surrounding grids, and the values  $\ell_i$  in the surrounding grids represent their effect weights to the dangerous potential field of the center grid. The dangerous potential field strength of the center grid is  $\mathbb{Q} = \sum_{i=0}^8 \ell_i \mathcal{L}_i$ , where  $\mathcal{L}_i$  is the binary value of its eight surrounding grids, and for the four oblique grids,  $\ell_i = 0.1$ , for the other grids,  $\ell_i = 0.2$ . Calculating the dangerous potential field strength of all grids in the binary map, we convert the binary map into a digital map with the dangerous potential field, as shown in Figure 12c. We expect to find an optimal path in the map with the dangerous potential field by the MDPF algorithm. The optimal path has the smallest total value of the dangerous potential field of all grids on the path, that is, the safest path with the lowest collision possibility.

#### 4.4.2. MDPF Algorithm

Same as the all-0 path search algorithm in Section 4.3.3, the MDPF algorithm sets the path start point at the top-left grid and the end point at the bottom-right grid, and  $\bar{U}$  of all grids are calculated in the same order. The difference is that the value at the top of the grid represents the value of  $\mathbb{Q}$ , and the bottom value  $\bar{U}$  represents the sum of the  $\mathbb{Q}$  of all grids on the optimal path from the start point to it. In addition, the all-0 path search algorithm uses  $\bar{U}$  to determine whether a binary map is valid, but the MDPF algorithm aims to find the safest path in the map with the dangerous potential field.

Since  $\bar{U}$  of all grids are calculated in the forward direction, the optimal path of one grid from the start point can be retracted based on the  $\bar{U}$  value of other grids. For example, in Figure 13, the path to grid G can only extend from grid C or grid D. Then we compare the  $\bar{U}$  of grid C and grid D, it can be found that  $\bar{U} = 0.6$  for A→C and  $\bar{U} = 1.1$  for A→D, therefore, the path to grid G is extended from grid C. According to the principle that the sub-path of the optimal path must also be optimal, the sub-path of the current path back one step is also the optimal path. Therefore, the mechanism for the MDPF algorithm is to find the optimal path by backtracking  $\bar{U}$  from the end point to the start end in the map with the dangerous potential field.

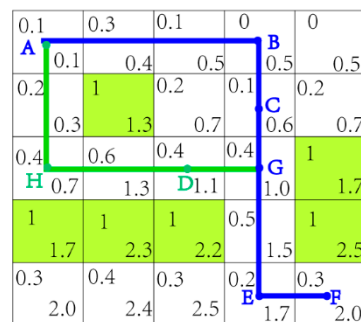


Figure 13. Path planning in the map with the dangerous potential field.

From the end point, we compare the  $\bar{U}$  values of its upper grid and left grid, and mark the grid with smaller  $\bar{U}$  as the target grid, then compare the  $\bar{U}$  values of the upper grid and left grid of the target grid. Following this step, the target grid can be backtracked to the start grid and all the target grids form the optimal path. The MDPF algorithm achieves the planning of the optimal path without traversing every path.

In the MDPF algorithm, for the current grid, if the  $\bar{U}$  value of its upper grid is the same as its left grid, different optimal paths will be obtained when setting different backtracking directions, including priority to the left; priority to the upper; priority to the upper left; and priority to the previous direction, etc.

### 4.4.3. Check Path

The information contained in the optimal path obtained by the MDPF algorithm is the time sequence for the planned joints to cross the  $N$ -multiplicity  $C$ -spaces on the pre-defined path, and it can be translated into joint paths. Then, it should be stitched with the linear joint paths of the planned joints together to form the complete joint paths of SMSRS from the initial configuration to the final configuration. Check whether SMSRS has self-collision under the complete joint paths, if there is no collision, it proves that the path is successfully planned, and if there still exists collision, start the next joint path planning program from locating the collision module. Until the complete joint paths are verified collision-free, the path planning ends.

## 5. Settings

### 5.1. Parameter Setting of SMSRS

In this paper, SMSRS with nine modules and 25 DOFs is selected as the path planning object, and  $\Sigma_b$  locates in its middle module, so there are four modules on both the  $a$  and  $b$  sides of  $\Sigma_b$ . The DOFs of  $a$  side are 12, and, the DOFs of the  $b$  side are 13 due to the existence of virtual joints. The body diagonal length of the module is 0.23 m. The D–H parameters of SMSRS are shown in Table 1.

**Table 1.** D–H parameters of SMSRS.

D–H Parameters of $a$ Side					D–H Parameters of $b$ Side				
Link	$\theta_i^a(deg)$	$d_i^a(m)$	$\alpha_{i-1}^a(deg)$	$A_{i-1}^a(m)$	Link	$\theta_i^b(deg)$	$d_i^b(m)$	$\alpha_{i-1}^b(deg)$	$A_{i-1}^b(m)$
1	0	0	0	0	v <sup>1</sup> -0	90	0	90	0
2	90	0	90	0.198	1-v	0	−0.243	90	0
3	−90	0.243	−90	0	2	−90	0	90	0
4	−90	0	−90	0	3	90	0	−90	0.198
5	90	0	90	0.198	4	−90	−0.243	−90	0
6	−90	0.243	90	0	5	−90	0	90	0
7	−90	0	−90	0	6	90	0	−90	0.198
8	90	0	90	0.198	7	−90	−0.243	−90	0
9	−90	0.243	90	0	8	−90	0	90	0
10	−90	0	−90	0	9	90	0	−90	0.198
11	90	0	90	0.198	10	−90	−0.243	−90	0
12	−90	0.243	90	0	11	−90	0	90	0
-	-	-	-	-	12	90	0	−90	0.198

The lower and upper limits of the joint angle are set as:

$$\begin{aligned} \mathbf{I}_b &= \begin{bmatrix} -90 & -90 & -90 & \dots & -90 \end{bmatrix}^T \\ \mathbf{u}_b &= \begin{bmatrix} 90 & 90 & 90 & \dots & 90 \end{bmatrix}^T \end{aligned} \tag{6}$$

### 5.2. Setting of Cases

This paper selects two cases to illustrate the application of the self-collision avoidance method for SMSRS in detail and verifies the feasibility of this method.

#### 5.2.1. Case 1

In case 1, the SMSRS moves from the initial configuration in Figure 14 to the final configuration in Figure 15.  $q_{start_a}$ ,  $q_{start_b}$  are, respectively, the initial angles of joints on



both  $a$  and  $b$  sides of  $\Sigma_b$ , and  $q_{end_a}$ ,  $q_{end_b}$  are, respectively, the final angles of joints on both  $a$  and  $b$  sides of  $\Sigma_b$ .

$$\begin{aligned}
 q_{start_a} &= [0, 0, 0, 90, 0, 0, 0, 0, 0, 90, 0, 0] \\
 q_{start_b} &= [0, 0, 0, -90, 0, -90, 0, 0, -30, 0, 0, 0] \\
 q_{end_a} &= [0, 0, 0, -90, 0, -90, 0, 0, -30, 0, 0, 0] \\
 q_{end_b} &= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
 \end{aligned}
 \tag{7}$$

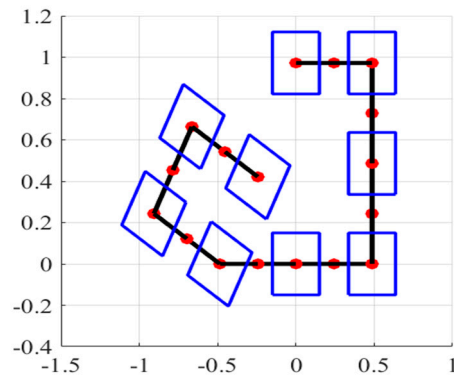


Figure 14. Initial configuration of SMSRS in case 1.

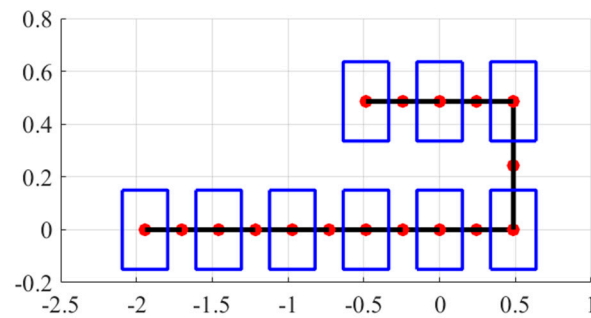


Figure 15. Final configuration of SMSRS in case 1.

As shown in Figure 16, modules will collide when joints of the SMSRS moves in linear joint paths.

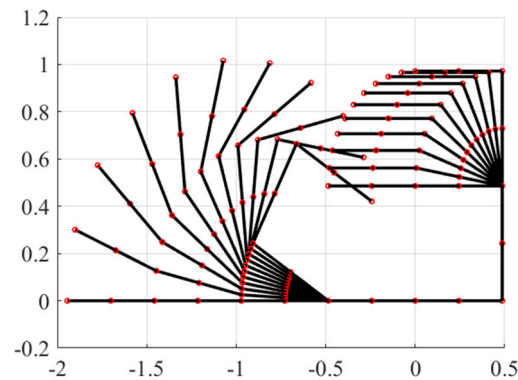


Figure 16. SMSRS reconfigure in linear joint paths in case 1.

### 5.2.2. Case 2

In case 2, the SMSRS moves from the initial configuration in Figure 17 to the final configuration in Figure 18.  $q_{start_a}$ ,  $q_{start_b}$  are, respectively, the initial angles of joints on

both  $a$  and  $b$  sides of  $\Sigma_b$ , and  $q_{end_a}$ ,  $q_{end_b}$  are, respectively, the final angles of joints on both  $a$  and  $b$  sides of  $\Sigma_b$ .

$$\begin{aligned}
 q_{start_a} &= [0, 0, 0, 90, 0, 90, 90, 0, 0, 90, 30, 0] \\
 q_{start_b} &= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \\
 q_{end_a} &= [0, 0, 0, 90, 0, -90, 90, 0, 0, 90, 0, 0] \\
 q_{end_b} &= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
 \end{aligned} \tag{8}$$

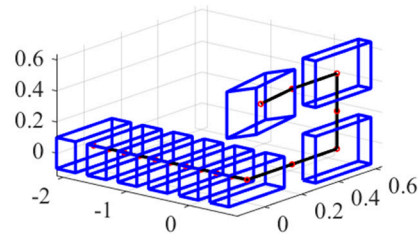


Figure 17. Initial configuration of SMSRS in case 2.

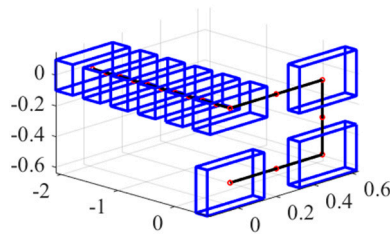


Figure 18. Final configuration of SMSRS in case 2.

As shown in Figure 19, modules will collide when joints of the SMSRS moves in linear paths.

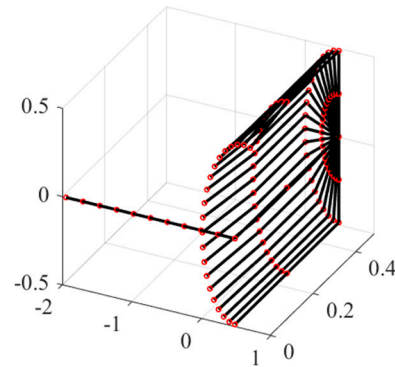


Figure 19. SMSRS reconfigure in linear joint paths in case 2.

### 5.3. Setting of Algorithm

For case 1 and case 2,  $N = 180$ , and set the number of samples to  $\vartheta = 40,000$ . All algorithms are implemented in MATLAB 2012a and run on the same machine with an Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz and 8 G memory.

## 6. Simulation Results and Analysis

### 6.1. Case 1

#### 6.1.1. Locate Collide Modules and Determine Planned Joints

Case 1 sets the joints moving from the initial angles to the final angles in linear paths. According to the collision detection model in Section 3 and the method for locating collision modules in Section 4.1, the two terminal modules of SMSRS are located as the collision

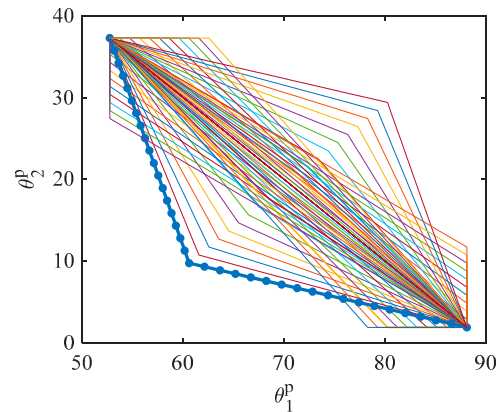
modules by real-time detection, and the collision time sequence is (71, 364) when the total time of collision process is divided into 1000 segments.

According to the selection principle of the planned joints in Section 4.2,  $\theta_1^p$  is set as the tenth joint on  $a$  side and  $\theta_2^p$  is set as the seventh joint on  $a$  side for case 1.  $(\theta_{1\_start}^p, \theta_{2\_start}^p) = (86.31, 3.65)$ ,  $(\theta_{1\_end}^p, \theta_{2\_end}^p) = (52.74, 37.26)$ ,  $(\theta_{1\_min}^p, \theta_{1\_max}^p) = (-90, 90)$ ,  $(\theta_{2\_min}^p, \theta_{2\_max}^p) = (-90, 90)$ .

### 6.1.2. Spliced C-Space

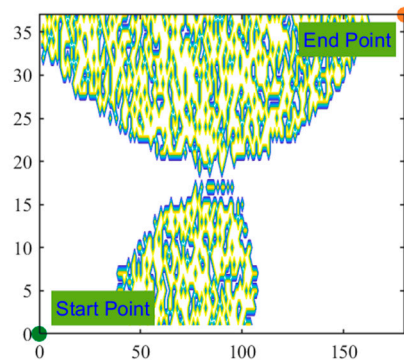
Follow the steps in Section 4.3.1 to create the 180-multiplicity C-spaces. When both of the planned joints are motion joints, their paths are pre-defined as shown in Figure 10a. For each pre-defined path, we judge whether the binary map generated by splicing C-space under this pre-defined path is valid, and, if it is not, we re-select the pre-defined path according to the principles in Section 4.3.2.

After multiple iterations shown in Figure 20, one pre-defined path is found as the needed pre-defined path because the binary map generated by splicing 180-multiplicity C-spaces under it is valid, and it is marked obviously in Figure 20.



**Figure 20.** All iterated pre-defined paths and final pre-defined path in case 1.

Under the final pre-defined path, we splice 180-multiplicity C-spaces following steps in Section 4.3.3 and obtain the binary map shown in Figure 21. In the binary map, (0, 0) is the start point of the path, and (180, 37) is the end point, and there exist accessible paths from the start point to the end point.

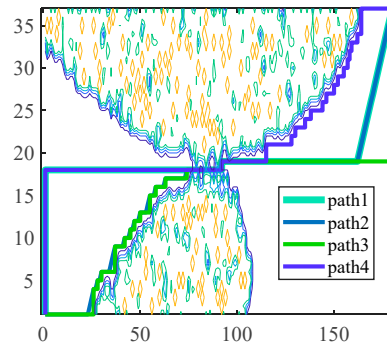


**Figure 21.** Binary map of case 1.

### 6.1.3. Path Planning

The map with the dangerous potential field can be obtained by calculating the dangerous field strength of all grids in the binary map. In the map with the dangerous potential

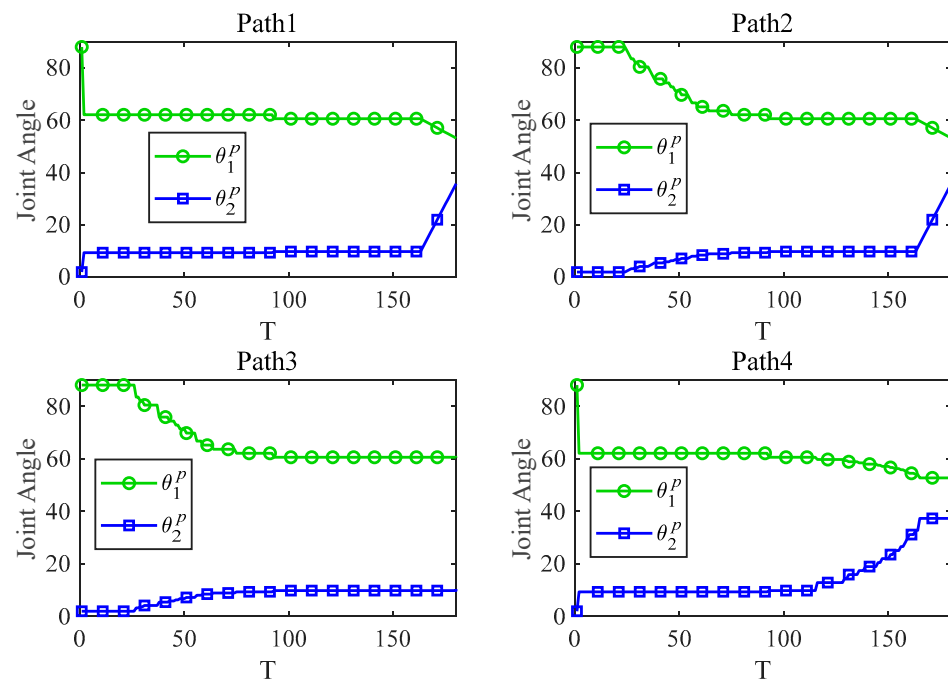
field, the MDPF algorithm could for search the collision-free path. Different paths will be obtained when setting different priority backtracking directions in the MDPF algorithm. Figure 22 shows the different collision-free paths founded by the MDPF algorithm. Among them, the priority backtracking directions set for path 1 is the previous direction, for path 2 is the bottom-left, for path 3 is the below, and for path 4 is the left.



**Figure 22.** Collision-free paths founded by the MDPF algorithm in the map with the dangerous potential field for case 1.

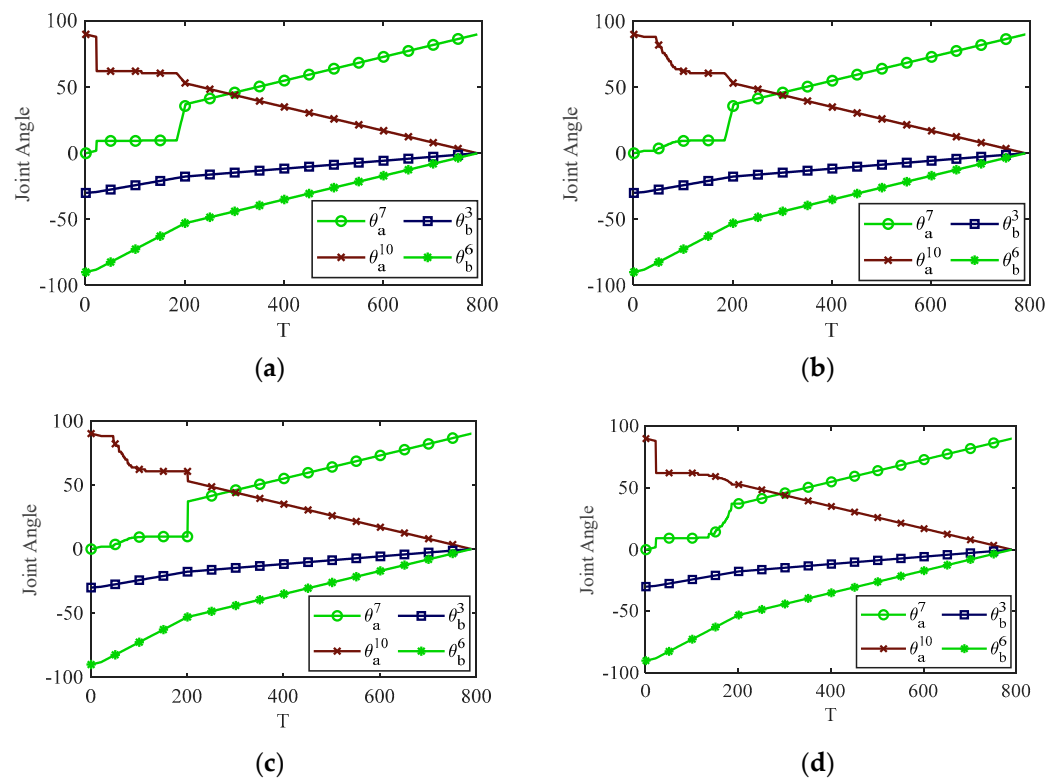
#### 6.1.4. Check Path

The collision-free path in the map with the dangerous potential field contains the information of the time sequence of the planned joint from the first to the Nth multiplicity C-space based on the pre-defined path. we translate the collision-free path into the angle curves of planned joints in Figure 23.

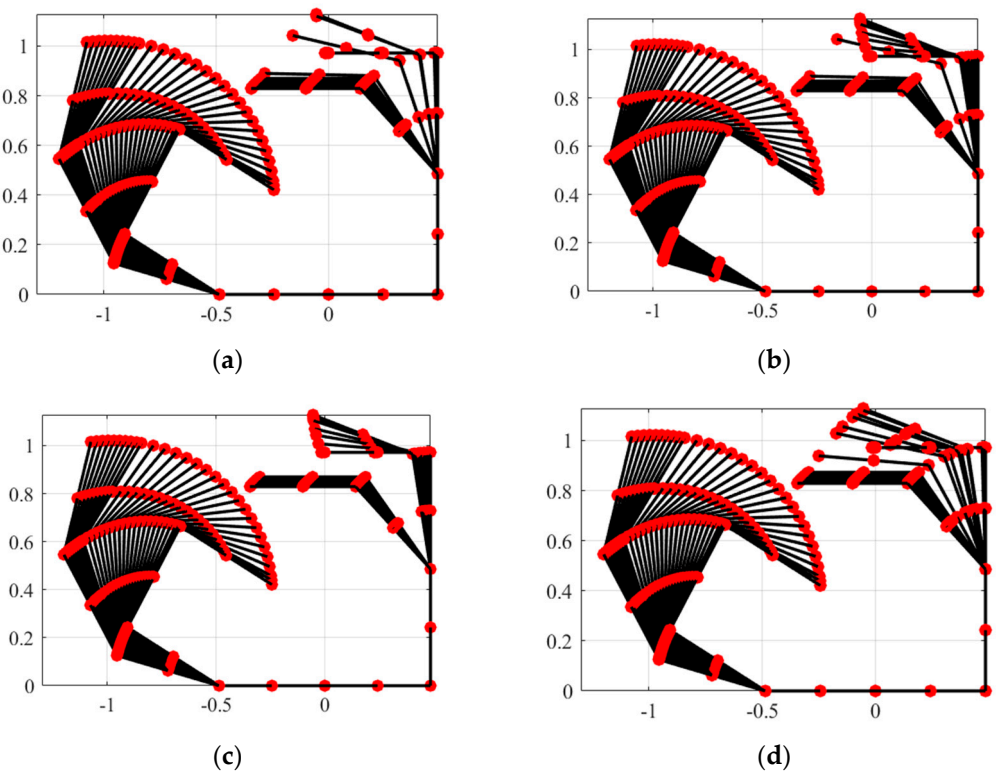


**Figure 23.** Angle curves of planned joints under four collision-free paths.

Maintaining the linear curves of joints when no self-collision occurs and inserting planned angle curves in Figure 21 into them, complete curves of all motion joints of the SMSRS are shown in Figure 24. We analyze the reconfiguration processes of SMSRS under these angle curves, and they are shown in Figure 25. Under different angle curves, the reconfiguration processes of SMSRS are different, but they can successfully avoid the self-collision in common.



**Figure 24.** Paths of all motion joints of the SMSRS under four collision-free paths: (a) path 1, (b) path 2, (c) path 3 and (d) path 4.



**Figure 25.** Reconfiguration processes of SMSRS under four collision-free paths: (a) path 1, (b) path 2, (c) path 3 and (d) path 4.

## 6.2. Case 2

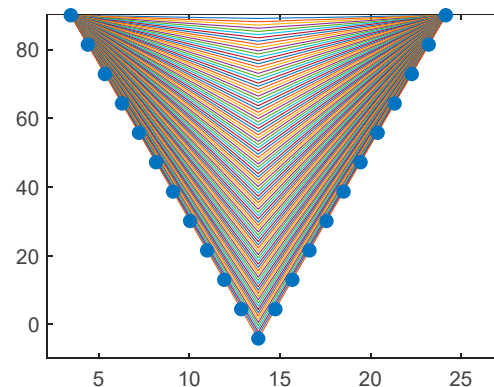
### 6.2.1. Locate Collide Modules and Determine Planned Joints

Case 2 sets the joints moving from the initial angles to the final angles in linear paths. According to the collision detection model in Section 3 and the method for locating collision modules in Section 4.1, the first and fifth modules on *a* side of SMSRS are located as the collision modules by real-time detection, and the collision time sequence is (245, 835) when the total time of collision process is divided into 1000 segments.

### 6.2.2. Spliced C-Space

According to the principles in Section 4.2 to select planned joints, they are set as two motion joints:  $\theta_1^p$  is the eleventh joint on *a* side and  $\theta_2^p$  is the sixth joint on *a* side. And  $(\theta_{1\_start}^p, \theta_{2\_start}^p) = (24.15, 54.9)$ ,  $(\theta_{1\_end}^p, \theta_{2\_end}^p) = (3.45, -69.3)$ ,  $((\theta_1^p)_{min}, (\theta_1^p)_{max}) = (-90, 90)$ ,  $((\theta_2^p)_{min}, (\theta_2^p)_{max}) = (-90, 90)$ . The 180-multiplicity C-spaces are built following the steps in Section 4.3.1. However, after pre-defining paths according to the principles in Figure 10a, the binary map generated by splicing C-space under each pre-defined path of current planned joints is invalid, it proves the planned joints cannot avoid self-collision of SMSRS after path planning.

In this case, we reselect the planned joint and set  $\theta_1^p$  as the eleventh joint on *a* side and  $\theta_2^p$  as the tenth joint on *a* side. At this point, the 180-multiplicity C-spaces are re-built using the random sampling method following steps in Section 4.3.1. After multiple iterations shown in Figure 26, one pre-defined path is found as the needed pre-defined path because the binary map generated by splicing 180-multiplicity C-spaces under it is valid, and it is marked obviously in Figure 26.

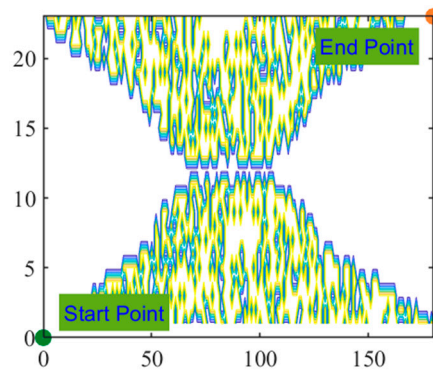


**Figure 26.** All iterated pre-defined paths and the final pre-defined path in case 2.

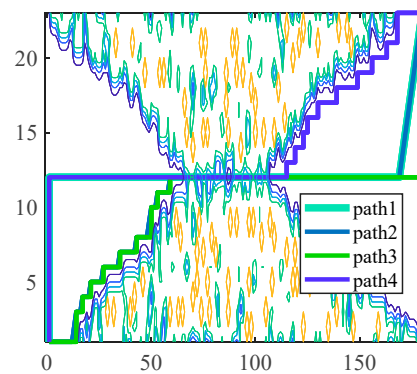
Under the final pre-defined path, splice 180-multiplicity C-spaces following steps in Section 4.3.3, and the binary map is shown in Figure 27. In the binary map, (0, 0) is the start point of the path, and (180, 37) is the end point, and there exist accessible paths from the start point to the end point.

### 6.2.3. Path Planning

The map with the dangerous potential field can be obtained by calculating the dangerous field strength of all grids in the binary map. In the map with the dangerous potential field, the MDPF algorithm could search for the collision-free path. Different paths will be obtained when setting different priority backtracking directions in the MDPF algorithm. Figure 28 shows the different collision-free paths founded by the MDPF algorithm. Among them, the priority backtracking directions for path 1 is the previous direction, for path 2 is the bottom-left, for path 3 is the below, and for path 4 is the left.



**Figure 27.** Binary map of case 2.



**Figure 28.** Collision-free paths founded by the MDPF algorithm in the map with a dangerous potential field for case 2.

#### 6.2.4. Check Path

The collision-free path in the map with the dangerous potential field contains the information of the time sequence of the planned joint from the first to the  $N$ th multiplicity C-space based on the pre-defined path. we translate the collision-free paths into the angle curves of planned joints in Figure 29.

Maintaining the linear curves of joints when no self-collision occurs and inserting planned angle curves in Figure 29 into them, complete curves of all motion joints of the SMSRS are shown in Figure 30. We analyze the reconfiguration processes of SMSRS under these angle curves, and they are shown in Figure 31. Under different angle curves, the reconfiguration processes of SMSRS are different, but they can successfully avoid the self-collision in common.

We observe that some segments of angle curves of planned joints change very sharply, such as the curve of  $\theta_a^{10}$  in Figure 28, which will lead to rapid changes of joint speeds and accelerations, even beyond their limits. However, the curves only reflect the path of joint angle and with no correspondence to time. Therefore, there are several ways to solve the problem:

1. If there is no limitation on the configuration time, we could extend the movement time of segments with sharp changes on the angle curves until the joint speeds and accelerations are within limitations;
2. Further improve the collision-free path planning algorithm by adding optimization objectives and constraints to find smoother joint paths; and
3. Select key points in the angel curves and deep plan the paths between these points to satisfy the joint velocity and acceleration constraints.

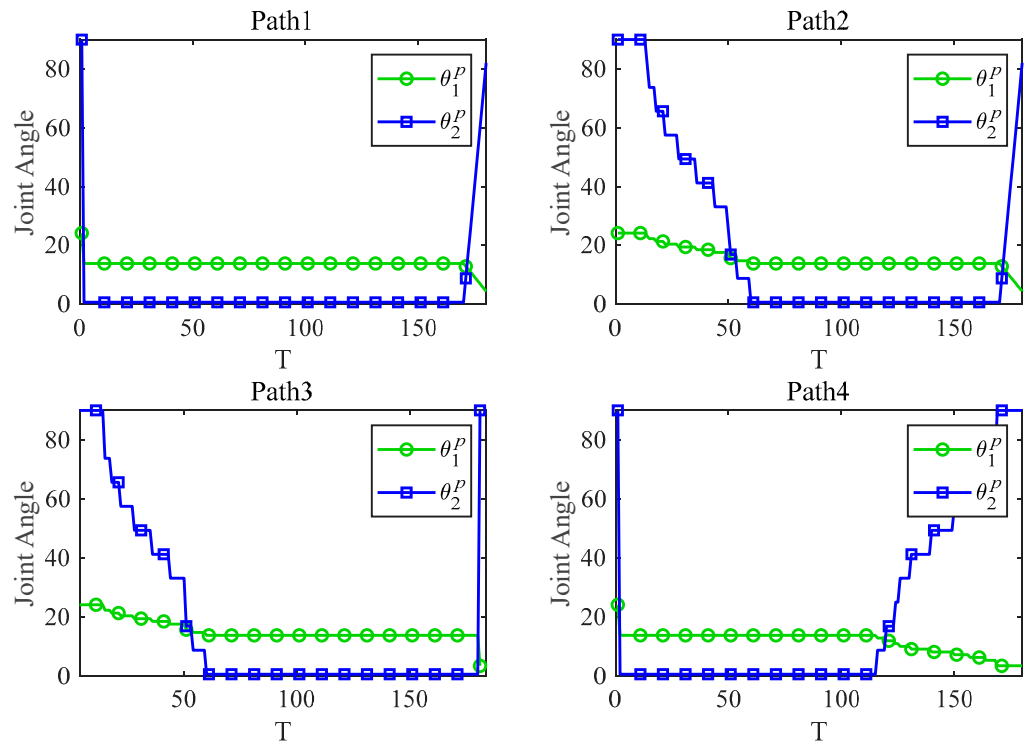


Figure 29. Angle curves of planned joints under four collision-free paths.

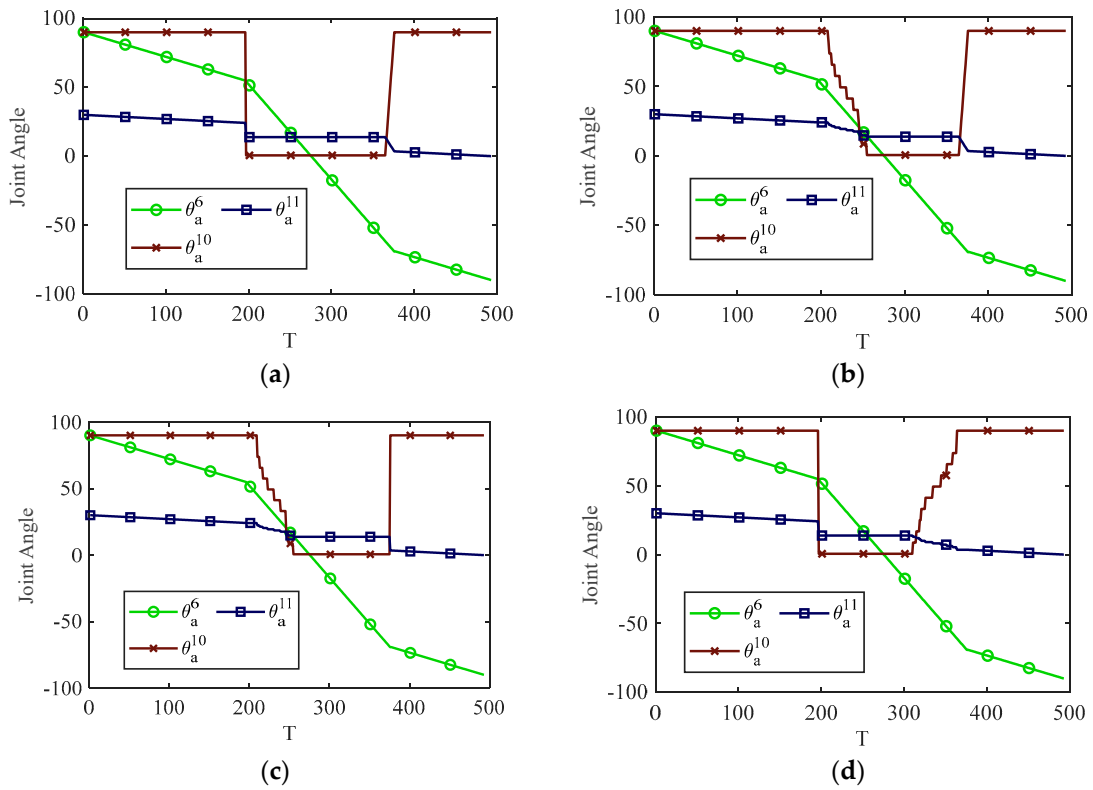
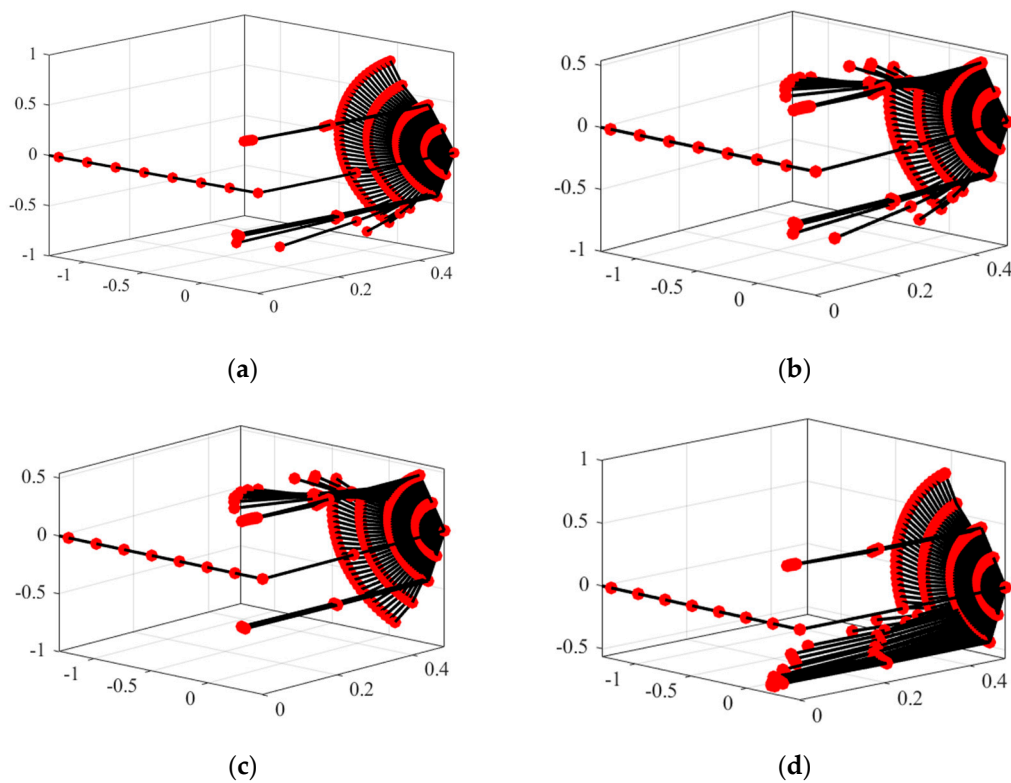


Figure 30. Paths of all motion joints of the SMSRS under four collision-free paths: (a) path 1, (b) path 2, (c) path 3 and (d) path 4.





**Figure 31.** Reconfiguration processes of SMSRS under four collision-free paths. (a) path 1, (b) path 2, (c) path 3 and (d) path 4.

## 7. Conclusions

SMSRS uses structural reconfiguration to achieve multi-mission capabilities. The research on self-collision avoidance in the reconfiguration process of SMSRS is particularly important for its safety, but it is still vacant because of the original and super-redundant structure. Based on the analysis of the characteristics of SMSRS and the summary of existing studies on the self-collision avoidance problem, we propose a method for self-collision avoidance of SMSRS, which includes a suitable collision detection model and an innovative self-collision avoidance strategy.

The collision detection model based on forward kinematics and the spherical nonholonomic envelope is capable of collision detection at a low computational cost for SMSRS. By comparison, offline path planning is the applicable self-collision avoidance strategy for SMSRS. The idea of splicing the multiple C-spaces based on pre-defined paths into a binary map solves the difficulty in joint path planning resulting from the dynamic change of module position, it successfully transforms the path search in joint space into time sequence planning based on the pre-defined path for planned joints. It provides a basis for collision-free path planning. Converting the binary map into a map with dangerous potential fields and searching for the safest path using the MDPF algorithm was proved can avoid self-collisions successfully. The planned joint angle curve lays the foundation for subsequent in-depth path planning and trajectory planning.

The proposed method of self-collision avoidance meets the needs of SMSRS and proved to be effective, which fills the gap of offline path planning research for self-collision avoidance of new self-configuration satellites to a certain extent. There are also implications for offline path planning of other self-configuration systems. Future works will focus on simplifying the steps of the method and the in-depth path planning and trajectory optimization for SMSRS with single or multiple optimization objectives.

**Author Contributions:** Conceptualization, J.A. and X.L.; Data curation, J.A.; Formal analysis, Z.Z. and J.A.; Funding acquisition, X.L.; Investigation, J.A., W.M. and G.H.; Methodology, J.A. and G.Z.; Project administration, X.L.; Supervision, X.L. and J.H.; Writing—original draft, J.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Defense Science and Technology Innovation Zone of China, grant number 00205501.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Flores-Abad, A.; Ma, O.; Pham, K.; Ulrich, S. A review of space robotics technologies for on-orbit servicing. *Prog. Aerosp. Sci.* **2014**, *68*, 1–26. [[CrossRef](#)]
- Zhang, Y.; Wang, W.; Sun, J.; Chang, H.; Huang, P. A Self-Reconfiguration Planning Strategy for Cellular Satellites. *Access* **2019**, *7*, 4516–4528. [[CrossRef](#)]
- Henry, C.A. An Introduction to the Design of the Cassini Spacecraft. *Space Sci. Rev.* **2002**, *104*, 129–153. [[CrossRef](#)]
- Waydo, S.; Henry, D.; Campbell, M. CubeSat design for LEO-based Earth science missions. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 9–16 March 2002.
- Poghosyan, A.; Golkar, A. CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions. *Prog. Aerosp. Sci.* **2017**, *88*, 59–83. [[CrossRef](#)]
- Cialone, G.; Gianfermo, A.; Di Cecco, A.; Mari, S.; Cassisi, S.; Santoni, F.; Piergentili, F. A concept mission for the Stellar Population and Evolution with Cubesats (SPEC). *Adv. Space Res.* **2019**, *63*, 800–811. [[CrossRef](#)]
- Santoni, F.; Gugliermetti, L.; Piras, G.; Pascale, S.D.; Pannico, A.; Piergentili, F.; Marzioli, P.; Frezza, L.; Amadio, D.; Gianfermo, A.; et al. GreenCube: Microgreens cultivation and growth monitoring on-board a 3U CubeSat. In Proceedings of the 2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace), Virtual Conference, 22–24 June 2020.
- Budianu, A.; Meijerink, A.; Bentum, M.J. Swarm-to-Earth communication in OLFAR. *Acta Astronaut.* **2015**, *107*, 14–19. [[CrossRef](#)]
- Iuzzolino, M.; Accardo, D.; Rufino, G.; Oliva, E.; Tozzi, A.; Schipani, P. A Cubesat Payload for Exoplanet Detection. *Sensors* **2017**, *17*, 493. [[CrossRef](#)]
- Kohout, T.; Näsilä, A.; Tikka, T.; Granvik, M.; Kestilä, A.; Penttilä, A.; Kuhno, J.; Muinonen, K.; Viherkanto, K.; Kallio, E. Feasibility of asteroid exploration using CubeSats—ASPECT case study. *Adv. Space Res.* **2018**, *62*, 2239–2244. [[CrossRef](#)]
- Viscio, M.A.; Viola, N.; Corpino, S.; Stesina, F.; Fineschi, S.; Fumentì, F.; Circi, C. Interplanetary CubeSats system for space weather evaluations and technology demonstration. *Acta Astronaut.* **2014**, *104*, 516–525. [[CrossRef](#)]
- Laskar, M.R.; Bhattacharjee, R.; Giri, M.S.; Bhattacharya, P. Weather Forecasting Using Arduino Based Cube-Sat. *Procedia Comput. Sci.* **2016**, *89*, 320–323. [[CrossRef](#)]
- Junqueira, A.M.; Mao, F.; Mendes, T.S.G.; Simões, S.J.C.; Balestieri, J.A.P.; Hannah, D.M. Estimation of river flow using CubeSats remote sensing. *Sci. Total Environ.* **2021**, *788*, 147762. [[CrossRef](#)] [[PubMed](#)]
- Villela, T.; Costa, C.A.; Brandão, A.M.; Bueno, F.T.; Leonardi, R. Towards the Thousandth CubeSat: A Statistical Overview. *Int. J. Aerosp. Eng.* **2019**, *2019*, 5063145. [[CrossRef](#)]
- Durmaz, B.; Demirkaya, B.Ö. Reliability considerations for design of space systems. In Proceedings of the 5th International Conference on Recent Advances in Space Technologies—RAST2011, Istanbul, Turkey, 9–11 June 2011.
- Bouwmeester, J.; Menicucci, A.; Gill, E.K.A. Improving CubeSat reliability: Subsystem redundancy or improved testing? *Reliab. Eng. Syst. Saf.* **2022**, *220*, 108288. [[CrossRef](#)]
- Menchinelli, A.; Ingiosi, F.; Pamphili, L.; Marzioli, P.; Patriarca, R.; Costantino, F.; Piergentili, F. A Reliability Engineering Approach for Managing Risks in CubeSats. *Aerospace* **2018**, *5*, 121. [[CrossRef](#)]
- An, J.; Li, X.; Zhang, Z.; Man, W.; Zhang, G. Joint Trajectory Planning of Space Modular Reconfigurable Satellites Based on Kinematic Model. *Int. J. Aerosp. Eng.* **2020**, *2020*, 8872788. [[CrossRef](#)]
- Liu, Y.; Yu, C.; Sheng, J.; Zhang, T. Self-collision Avoidance Trajectory Planning and Robust Control of a Dual-arm Space Robot. *Int. J. Control Autom. Syst* **2018**, *16*, 2896–2905. [[CrossRef](#)]
- Xie, Y.; Zhang, Z.; Wu, X.; Shi, Z.; Chen, Y.; Wu, B.; Mantey, K.A. Obstacle Avoidance and Path Planning for Multi-Joint Manipulator in a Space Robot. *Access* **2020**, *8*, 3511–3526. [[CrossRef](#)]
- Lei, M.; Wang, T.; Yao, C.; Liu, H.; Wang, Z.; Deng, Y. Real-Time Kinematics-Based Self-Collision Avoidance Algorithm for Dual-Arm Robots. *Appl. Sci.* **2020**, *10*, 5893. [[CrossRef](#)]
- Okada, K.; Inaba, M.; Inoue, H. Real-time and Precise Self Collision Detection System for Humanoid Robots. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005.
- Okada, K.; Inaba, M. A Hybrid Approach to Practical Self Collision Detection System of Humanoid Robot. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006.

24. Zhao, L.; Zhao, J.; Liu, H. Solving the Inverse Kinematics Problem of Multiple Redundant Manipulators with Collision Avoidance in Dynamic Environments. *J. Intell. Robot. Syst.* **2021**, *101*, 30. [[CrossRef](#)]
25. Cascio, J.; Karpenko, M.; Gong, Q.; Sekhavat, P.; Ross, I.M. Smooth proximity computation for collision-free optimal control of multiple robotic manipulators. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009.
26. Wang, W.; Zhu, M.; Wang, X.; He, S.; He, J.; Xu, Z. An improved artificial potential field method of trajectory planning and obstacle avoidance for redundant manipulators. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 172988141879956. [[CrossRef](#)]
27. Koptev, M.; Figueroa, N.; Billard, A. Real-Time Self-Collision Avoidance in Joint Space for Humanoid Robots. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1240–1247. [[CrossRef](#)]
28. Stasse, O.; Escande, A.; Mansard, N.; Miossec, S.; Evrard, P.; Kheddar, A. Real-time (self)-collision avoidance task on a hrp-2 humanoid robot. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008.
29. Dietrich, A.; Wimbock, T.; Taubig, H.; Albu-Schaffer, A.; Hirzinger, G. Extensions to reactive self-collision avoidance for torque and position controlled humanoids. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011.
30. Quiroz-Omana, J.J.; Adorno, B.V. Whole-Body Control With (Self) Collision Avoidance Using Vector Field Inequalities. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4048–4053. [[CrossRef](#)]
31. Yahya, S.; Moghavvemi, M.; Mohamed, H.A.F. Geometrical approach of planar hyper-redundant manipulators: Inverse kinematics, path planning and workspace. *Simul. Model. Pract. Theory* **2011**, *19*, 406–422. [[CrossRef](#)]
32. Dash, A.K.; Chen, I.M.; Yeo, S.H.; Yang, G. Workspace generation and planning singularity-free path for parallel manipulators. *Mech. Mach. Theory* **2005**, *40*, 776–805. [[CrossRef](#)]
33. Stilman, M. Global Manipulation Planning in Robot Joint Space With Task Constraints. *IEEE Trans. Robot.* **2010**, *26*, 576–584. [[CrossRef](#)]
34. Delgado, R.; Choi, B.W. Practical high curvature path planning algorithm in joint space. *Electron. Lett.* **2015**, *51*, 469–471. [[CrossRef](#)]
35. Rybus, T.; Wojtunik, M.; Basmadji, F.L. Optimal collision-free path planning of a free-floating space robot using spline-based trajectories. *Acta Astronaut.* **2022**, *190*, 395–408. [[CrossRef](#)]
36. Gao, X.; Jia, Q.; Sun, H.; Chen, G. Research on Path Planning for 7-DOF Space Manipulator to Avoid Obstacle Based on A Algorithm. *Sens. Lett.* **2011**, *9*, 1515–1519. [[CrossRef](#)]
37. Serrantola, W.G.; Grassi, V. Trajectory Planning for a Dual-Arm Planar Free-Floating Manipulator using RRTControl. In Proceedings of the 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2–6 December 2019.
38. Wei, K.; Ren, B. A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm. *Sensors* **2018**, *18*, 571. [[CrossRef](#)]
39. Gong, L.; Zhang, Y.; Cheng, J. Coordinated Path Planning Based on RRT Algorithm for Robot. *Appl. Mech. Mater.* **2014**, *494–495*, 1003–1007. [[CrossRef](#)]
40. Liu, S.; Zhang, Q.; Zhou, D. Obstacle Avoidance Path Planning of Space Manipulator Based on Improved Artificial Potential Field Method. *J. Inst. Eng. India Ser. C* **2014**, *95*, 31–39. [[CrossRef](#)]
41. Tong, Y.; Liu, J.; Liu, Y.; Yuan, Y. Analytical inverse kinematic computation for 7-DOF redundant sliding manipulators. *Mech. Mach. Theory* **2021**, *155*, 104006. [[CrossRef](#)]
42. Yiyang, L.; Xi, J.; Hongfei, B.; Zhining, W.; Liangliang, S. A General Robot Inverse Kinematics Solution Method Based on Improved PSO Algorithm. *IEEE Access* **2021**, *9*, 32341–32350.
43. Lopez-Franco, C.; Hernandez-Barragan, J.; Alanis, A.Y.; Arana-Daniel, N. A soft computing approach for inverse kinematics of robot manipulators. *Eng. Appl. Artif. Intel.* **2018**, *74*, 104–120. [[CrossRef](#)]
44. Zhang, Q.; Wang, L.; Zhou, D.S. Trajectory Planning of 7-DOF Space Manipulator for Minimizing Base Disturbance. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 10. [[CrossRef](#)]
45. Xie, Y.; Zhou, R.; Yang, Y. Improved Distorted Configuration Space Path Planning and Its Application to Robot Manipulators. *Sensors* **2020**, *20*, 6060. [[CrossRef](#)]