Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 2508304, 17 pages
http://dx.doi.org/10.1155/2016/2508304

Research Article

Hierarchical Task Planning for Multiarm Robot with Multiconstraint

Yifan Wang, Hanxu Sun, Gang Chen, Qingxuan Jia, and Boyang Yu

School of Automation, Beijing University of Posts and Telecommunications, No. 10, Xitucheng Road, Haidian District, Beijing 100876, China

Correspondence should be addressed to Yifan Wang; wangyifan@bupt.edu.cn

Received 25 March 2016; Revised 12 July 2016; Accepted 27 July 2016

Academic Editor: Francisco Gordillo

Multiarm systems become the trends of space robots, for the on-orbit servicing missions are becoming more complex and various. A hierarchical task planning method with multiconstraint for multiarm space robot is presented in this paper. The process of task planning is separated into two hierarchies: mission profile analysis and task node planning. In mission profile analysis, several kinds of primitive tasks and operators are defined. Then, a complex task can be decomposed into a sequence of primitive tasks by using hierarchical task network (HTN) with those primitive tasks and operators. In task node planning, A* algorithm is improved to adapt the continuous motion of manipulator. Then, some of the primitive tasks which cannot be executed directly because of constraints are further decomposed into several task nodes by using improved A* algorithm. Finally, manipulators execute the task by moving from one node to another with a simple path plan algorithm. The feasibility and effectiveness of the proposed task planning method are verified by simulation.

1. Introduction

With the continuous exploration of space, the on-orbit tasks like construction, maintenance, and service of spacecraft become more and more complex [1–3]. The space environment has the characteristics of high vacuum, intense radiation, and alternate change between high temperature and low temperature, so it is highly risky for the astronaut EVA (Extra-Vehicular Activity). However, in order to ensure the normal maintenance of equipment in capsule, astronaut change of on-orbit guard will also consume a lot of resources. So we can utilize space manipulator to assist or replace astronaut to execute all kinds of tasks. It can not only protect life security of astronaut, but improve the work efficiency and save operation cost.

The on-orbit task has the characteristic of complex circumstance and various constraint and kinds, so it becomes development tendency that utilizes space manipulator to execute complex and precision tasks. In most cases, the space manipulator is in the circumstance which is small, complicated, and lacking in supplement, and energy supplement and maintenance cost a great deal, so it bears the constraint from itself and environment when it is running. On the other hand, because of the deficiency of onboard computer and reliability requirement of on-orbit control system, the space manipulator can carry some kind of fundamental path planning algorithm. In the face of flexible on-orbit tasks, the traditional path planning method for space manipulator cannot satisfy the complex requirement of on-orbit service task, even if the path planning method is with many optimal abilities. So it is necessary to introduce the task planning upon path planning of the manipulator, to make the manipulator get the capacity to analyze, disassemble, and plan complex tasks and the capacity of configuration and optimization of resources in task level. Task planning can make space manipulator not only deal with more complex and precision tasks, but macroscopically optimize and configure resources and can delay maturing of manipulator and maintain the reliability of space manipulator.

Many researchers have studied task planning theory. Fikes and Nilsson [4] succeed in developing a robotic path planning system STRIPS based on resolution principle. This method is applied to Shakey robotic system, and it is widely adopted by researchers. Erol et al. [5–7] summarize the

fundamentals and regulation of hierarchical task network (HTN) and give universal planning algorithm (UMCP).

For robot task planning problems, scholars carry out the related research work. Kaelbling and Lozano-Pérez [8] give an integrated hierarchical task and movement planning strategy, and design a depth-first recursive algorithm which traverses planning tree. The algorithm realizes the complicated logic of moving objects and clearing obstacles. But this algorithm only analyzes the upper logic of tasks and does not take all kinds of constraint in implementation into account. Lozano-Pérez and Kaelbling [9] give a task and movement planning method based on symbol search. At first, the method searches the feasible sequence of upper operation and then transforms local planning into a geometrical constraint satisfaction problem. It solves the decision problems about grasping position, setting position, and moving path under geometrical constraint. But the geometric constraint solver adopted by the algorithm needs too much pretreatment time, so the planning efficiency will reduce. Barry et al. [10] transform multibehaved planning problem into multimode planning problem and solve it by hierarchical algorithm to get different grasping position of the same object. But the planning process does not take the optimization of the performance parameters of the robot into account, so it does not satisfy space application requirements. Aiming at underwater robot, Honghao et al. [11] puts forward a practical task process model based on Petri net; it can be used for task modeling and management. But this model only can be appropriate for the presupposed typical task. Robot lacks independent ability in complicated circumstance. Zong et al. [12] proposed hierarchical planning method of SimMan combination task based on key state. Hierarchical reinforcement learning is adopted to sampling in state space, and the state with the most frequency is chosen as the key state which is used to decompose the compound task. But the algorithm only gives the action sequence and does not plan the detail action reasonably. So it cannot be used in robotic control system. Singh et al. [13] design task planning algorithm for planet rover which is based on D^* algorithm. The algorithm is mainly used for obstacle avoidance and navigation of rover, and it introduces multimodule decision-making mechanism to decide next action of rover. But it does not take geometric constraints of robot body into account.

For multiarm system, Kim et al. [14, 15] put forward that if we want to use dual-arm robot to bin something, the system must combine obstacles avoidance algorithm, precision control, and compliant control together in planning process, but the algorithm will be highly complex. Thus, they come up with assisting dual-arm robot task planning to reach its goals by demonstration of kinematics. The method has high efficiency and security, but it needs too much artificial assist, so the universality of automation is very poor. Charoenseang et al. [16] use sensors to build the virtual operation environment on the basis of providing visual feedback and force feedback to manipulator; then the task planning is executed to reach the goals. Compared to teaching method, this method makes task planning of dual-arm coordination intelligent. Kim et al. [17, 18] use rapidly expanding random tree to generate assembly path and repeating capture path. These paths compose the plan result of dual-arm robot assembly task. Analogously, Vahrenkamp et al. [19] utilize the gradient descent method to random sample in reachable workspace which is calculated in advance and then use feasible inverse arithmetic which consumes lowly to get the limited joint angle and get task path by iteration. The method does not need specific parameter configuration, so it can plan independently. Aiming at mobile dual-arm robot, Takahama et al. [20] utilize the method which adds the task points that satisfy the constraint conditions in the straight line trajectory of end to generate task trajectory. But it is lacking in the high-precision collision detection algorithm, so it is not up to the precision operation tasks.

At present, the research at home and abroad related to motion planning of space manipulator mainly focuses on path planning algorithm and its optimization. Qi et al. [21] have studied obstacle avoidance path planning algorithm of space manipulator based on genetic algorithm. Chen et al. [22] put forward obstacle avoidance path planning algorithm of space manipulator based on C space. Yoo et al. [23] utilize laser and sonar sensor to finish obstacle avoidance task by adding specific points. The method does not use the information of joint angle, so the singular points are excluded, and it reduces the complexity of algorithm. The above-mentioned algorithms are all aimed at joint space planning; they cannot be used when there are strict requirements for the posture of end.

Based on the characteristic of on-orbit operation of space multimanipulator system, this paper studied the task planning method in complex circumstance with multiconstraint. First, the paper comes up with a double-layer task planning framework, and it separates the planning into complex task decomposition and action detail planning. Second, three necessary elements of HTN—primitive work, plan domain, and plan algorithm—are specially designed for space manipulator; then complex task can be decomposed into primitive tasks sequence based on HTN. In order to ensure the feasibility of the primitive task, the paper proposes a highly efficient precision collision detection algorithm based on point cloud model. Third, taking various constraints and optimization into consideration, A^* algorithm is improved to plan task nodes for primitive task. In order to ensure the feasibility of the task, collision avoidance is given special attention. A highly efficient precision collision detection algorithm based on point cloud model is proposed, for its modeling approach is very simple and easy for spaceborne computer to use. Finally, the task planning method is verified by simulation.

2. General Idea

Manipulator control system is divided into three levels: task planning, path planning, and motion control. Task planning, as the top level of manipulator control system, is responsible for the receiving, analyzing, and resolving of tasks. It should be able to divide a task into a series of actions which can be planned and executed by manipulator directly. Due to the diversity of ways to execute a task, resources planning is also a part of task planning. Resources would be optimized in the process of the whole task, and planning result is given in

the form of several task nodes and the resources allocation between two adjacent nodes.

In this paper, the task planning for space manipulator is divided into two stages.

(1) Planner accepts the task target and divides complex task into simple tasks (primitive task) considering the ability of the manipulator. Logic feasibility of the task is ensured at the same time. Target task and initial state, as the input of planner, are the information including target location of objects and manipulators, the initial location of objects and manipulators, and whether the manipulator caught an object and which object the manipulator caught. We use HTN to do the dividing work. Although HTN is an existing method, it provides only ideas of decomposing complex task and how the method works, and no details about how a task is described mathematically. Those details usually depend on specific situation that HTN applies on. For manipulator, it is a problem to distinguish "move from A to B" and "move from C to D," although they are all a type of MOVE task. The novelty of the HTN in this paper is

   (a) designing HTN plan domain for manipulator; in plan domain, some simple task (primitive task) that can be executed by manipulator directly is mathematically defined; so different tasks can be distinguished simply by their mask matrix (defined in Section 3);

   (b) proposing a mathematical method for describing the task executing effect; with the help of this method, HTN reasoning can be processed by computer, and planning process can work automatically.

(2) Each simple task is decomposed into several simple paths automatically using a heuristic search algorithm in order to ensure that the tasks can be performed. The input of the heuristic search algorithm is a primitive task which is output from HTN. Primitive task usually refers to moving from one position to another including initial configuration and target position of manipulator. Initial configuration and target position are the initial state and final state of a manipulator action and can be regarded as two points in its work space, while heuristic algorithm can elegantly solve the way-finding problem between two points. But, heuristic algorithm usually gives an irregular trajectory which is difficult for manipulator to execute. So, in this paper, an improved heuristic algorithm is proposed and is used to obtain a series of intermediate nodes (task node). Then, trajectories between any two adjacent nodes can be planned by an existing simple path plan algorithm for manipulator.

It is worth noting that this task plan method can be applied to any manipulator theoretically, although it is particularly designed for space manipulator. When it is applied to a robot, a task can be assigned as "tighten the bolt." Then, planner would decompose it into "move to spanner," "grab (spanner)," "move to bolt," and "tighten (bolt)" automatically. All those actions are part of predefined primitive tasks and can be executed by robot directly. However, there is usually no need to apply task planning to a manipulator which works on earth. On one hand, there is always a man nearby while the manipulator is working; the man who stands beside it could plan its task depending on the realities of situation and monitor any potential dangers on the manipulator. On the other hand, robot on earth (especially industrial robot) is always used to do very simple and repetitive works [24, 25]. It often works with a teach pendant or runs preprogrammed programs and has no need to do a task plan. However, space manipulator works in a harsh space environment. Space environment has the characteristics of difficult communication, lack of humans, and changing contents of task. So, it demands the autonomy of space manipulator, and task planning is necessary for space manipulator to do on-orbit servicing.

Figure 1 shows the double-layer framework for task planning.

3. Mission Profile Analysis Based on Hierarchical Task Network

3.1. Hierarchical Task Network and General Planning Algorithm. Hierarchical task network planning uses the idea of task decomposition which divides complex nonprimitive tasks into executable primitive tasks step by step. HTN abstracts the goal state of task into goal task, a layered structure of the task will be formed from complex to simple, and a primitive task which is indissoluble will be placed into the lowest level. HTN's programming domain consists of a list of operators of primitive tasks and a list of operators of compound tasks; when the goal task and the current state of environment are given, the feasible basic motion sequence will be calculated by the planner with the use of method of simplifying the task in planning domain.

There are three types of tasks in HTN [6]:

(1) Goal tasks, like goals in STRIPS, are properties we wish to make true in the world.

(2) Primitive tasks are tasks we can directly achieve by executing the corresponding action.

(3) Compound tasks denote desired changes that involve several goal tasks and primitive tasks.

Goal tasks and compound tasks are called nonprimitive tasks.

A primitive task, a goal task, and a compound task are syntactic construct of the form do($f(x_1, \ldots, x_k)$), $f \in F$; achieve(l), where l is a literal; perform$[t(x_1, \ldots, x_k)]$, $t \in T$, where $x_1, \ldots, x_k$ are task parameters.

Tasks are connected together via the use of task networks; task network is a combination of tasks and task constraints; the form is as follows:

$$\left((n_1 : \alpha_1), \ldots, (n_m : \alpha_m), \phi\right), \tag{1}$$

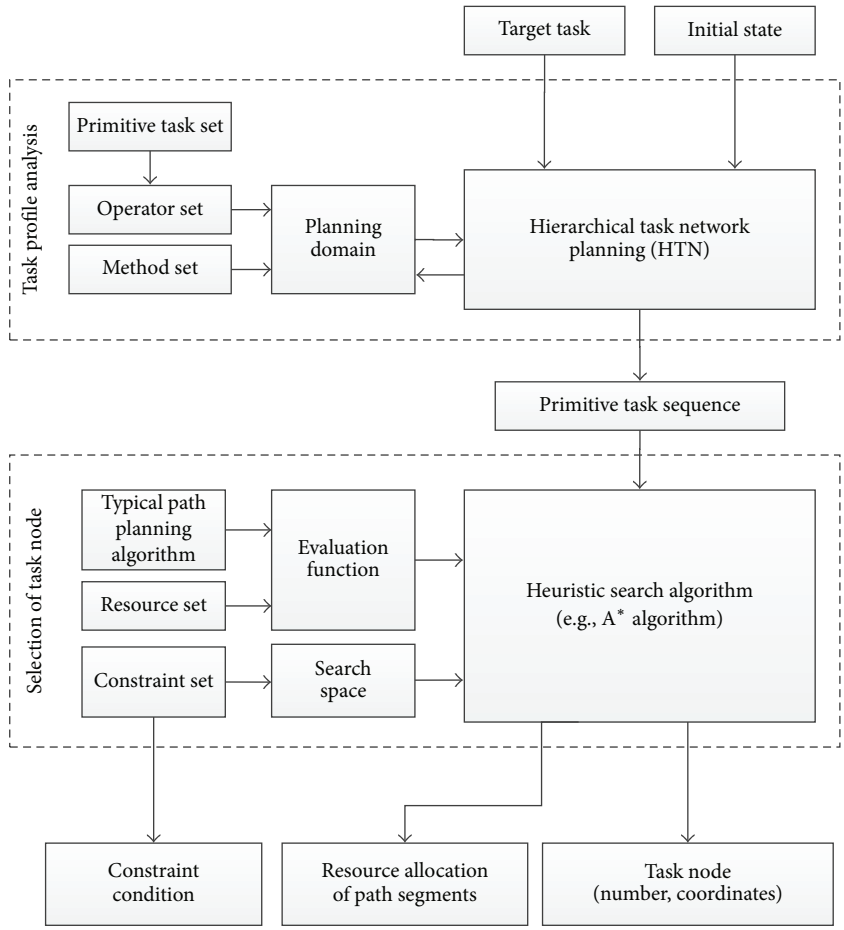


Figure 1: Framework of space manipulator task planning.

where α_i is a task with the label of n_i and ϕ is a boolean formula constructed from variable binding constraints such as $v = v'$ and $v = c$, temporal ordering constraints such as $n \prec n'$, and truth constraints such as (n, l), (l, n), and (n, l, n'), where $n, n' \in N$, $v, v' \in V$, l is a literal, and $c \in C$. $n \prec n'$ means that the task labeled with n must precede the one labeled with n'; (n, l), (l, n), and (n, l, n') mean that l must be true immediately after n, immediately before n, and between n and n', respectively.

A planning domain $\mathscr{D}$ is a pair [7]

$$\mathscr{D} = \langle \mathrm{Op}, \mathrm{Me} \rangle, \tag{2}$$

where Op is a set of operators (one for each primitive task) and Me is a set of methods.

An operator tells the effect of a primitive task and is a syntactic construct of the form

$$\mathrm{Op} \coloneqq \left(f\left(v_1, \ldots, v_k\right), q_1, \ldots, q_n, l_1, \ldots, l_m\right), \tag{3}$$

where f is a primitive task symbol, v_i is a variable of f, q_i denote the preconditions that must be satisfied before executing primitive task f, and l_i denote the primitive task's effects (also called postconditions).

A primary task can be completed by performing the corresponding action. A nonprimary task is completed by telling the planner how to accomplish it using a method. A method is a construct of the form

$$\mathrm{Me} \coloneqq (\alpha, d), \tag{4}$$

where α is a nonprimitive task and d is a task network. It says that one way to achieve α is to perform the tasks specified in the network d.

A planning problem is a triple.

$$P = \langle d, I, \mathscr{D} \rangle, \tag{5}$$

where, $\mathscr{D}$ is a planning domain, I is the initial state, and d is the task network.

A plan strategy which could be used in any field is proposed in [7]. Planning process starts from task network d and performs the following steps repeatedly until nonprimitive task no longer exists:

(1) Find a nonprimitive task in d and find a method $m = (t, d')$ in M, wherein t and u are consistent.

(2) Simplify u to generate a new d; namely, use the task in d' to replace u, and merge constraints of d and d'.

(3) Once there is no nonprimitive task in d, find an instance σ that could satisfy all constraints. σ is a successful planning result of the primitive task.

In a given task network, the interaction between tasks is inevitable, some of which are beneficial, while others are harmful. For example, the result of a task will destroy the precondition of another task, or results and preconditions of the two tasks will conflict with each other. There is no general method to deal with conflicts; conflicts are resolved by adding a new task in task network in this paper.

3.2. Task Profile Analysis of Space Manipulator. Although a universal task planning program was given by HTN, all components of the planner should be designed for the specific application of space manipulator, for HTN's operation is too abstract. A method for characterizing the state of scene in which space manipulator works is proposed in this section. Meanwhile, comparative method and state transfer operation for scene states are also given. Then, primitive task set and its operator for space manipulator are defined in this section. Task profile analysis for space manipulator based on HTN is realized by using these instancing planning elements.

3.2.1. Characterization of Scene State. The size of objects is not considered during task profile analysis. Only the logical relations between objects which are operated by manipulator are analyzed. So, the objects in the scene are abstracted into some operation interface vectors. When the end position of space manipulator is consistent with object interface vector, the object could be operated and the movement of the object would be abstracted into the change of interface vector. Suppose there are k objects and a manipulator in the scene; then the state of scene is indicated by S:

$$S = \begin{bmatrix} s_0 & s_1 & \cdots & s_k \end{bmatrix} = \begin{bmatrix} \Theta_0 & \Theta_1 & \cdots & \Theta_k \\ g_0 & g_1 & \cdots & g_k \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & \cdots & x_k \\ y_0 & y_1 & \cdots & y_k \\ z_0 & z_1 & \cdots & z_k \\ \alpha_0 & \alpha_1 & \cdots & \alpha_k \\ \beta_0 & \beta_1 & \cdots & \beta_k \\ \gamma_0 & \gamma_1 & \cdots & \gamma_k \\ g_0 & g_1 & \cdots & g_k \end{bmatrix}, \tag{6}$$

where s_0 represents the end effector of manipulator, $s_1 \sim s_k$ represent objects, and $\Theta = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix}^T$ is a coordinate. For manipulator, Θ indicates the coordinate of end position; for the object in the environment, it indicates operation interface vector which is usually defined on capture position of the object; g indicates whether the manipulator carries an object or not, and $g = \begin{bmatrix} g_0 & g_1 & \cdots & g_k \end{bmatrix}$ only have two values:

(a) $g = 0$: there is no-load on the manipulator;

(b) $g_0 = 1$: there is only one item $g_i = 1$ among $g_1 \sim g_k$; other items are all zero; it is used to indicate that the ith object is being caught by manipulator.

State mask is defined in order to help the calculation between state matrixes.

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1k} \\ m_{21} & m_{22} & \cdots & m_{2k} \\ m_{31} & m_{32} & \cdots & m_{3k} \\ m_{41} & m_{42} & \cdots & m_{4k} \\ m_{51} & m_{52} & \cdots & m_{5k} \\ m_{61} & m_{62} & \cdots & m_{6k} \\ m_{71} & m_{72} & \cdots & m_{7k} \end{bmatrix}, \tag{7}$$

where $m_{ij} \in \{0, 1\}$.

(1) Judging Whether State Satisfies a Condition. It is necessary to check whether the current state of scene satisfies the precondition (postcondition) of a task before (after) executing the task. Only specific several objects and their properties are concerned during the checking of state, and others are not taken into consideration. Suppose S is the current state of the scene. State S' and mask M describe the condition that needs to be satisfied by S. If the following equation is set up:

$$S' \ \& \ M = S \ \& \ M, \tag{8}$$

then the current state of scene satisfies the condition described by S' and M. Mask M is on both sides of the formula which are consistent with each other.

Operator & is an AND operation for matrix: if elements in M are 0, then the elements on the corresponding position of S are set to 0; if elements in M are 1, then the corresponding elements of S do not change.

(2) State Transfer. When an operator has an effect on the scene, the current scene will make the following change:

$$S_{\text{new}} = S_{\text{old}} \ \& \ \overline{M} + S' \ \& \ M, \tag{9}$$

where S_{new} is the scene state after conversion, S_{old} is the scene state before conversion, M is mask that indicates operator's effect, $\overline{M}$ is the inverse of M, and S' is the state matrix that indicates the effect of the operator.

3.2.2. Typical Primitive Tasks and Their Operators. Having taken the characteristics of various kinds of space manipulator in-orbit missions into consideration, three basic primitive tasks are defined to constitute the primitive task set: move, capture, and release. Complex tasks can be formed by these primitive tasks. Their corresponding operators contain both the precondition and postcondition of the task.

(1) MOVE. move(v) indicates that the end of the manipulator moves from the current position w to a target position v. Only the position of the manipulator in the state matrix is changed by MOVE tasks, as well as the location of the object caught by manipulator.

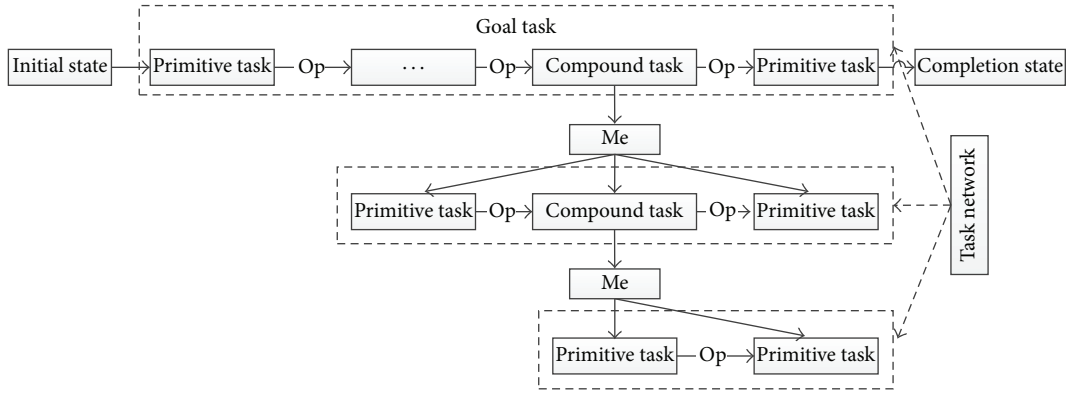


Figure 2: Structure of hierarchical network for task profile analysis.

Precondition:

$$S' = \begin{bmatrix} w & \cdots & w & \cdots \\ x & \cdots & x & \cdots \end{bmatrix},$$
$$M = \begin{bmatrix} x & \cdots & x & \cdots \\ x & \cdots & x & \cdots \end{bmatrix}. \quad (10)$$

Postcondition:

$$S' = \begin{bmatrix} v & \cdots & v & \cdots \\ x & \cdots & x & \cdots \end{bmatrix},$$
$$M = \begin{bmatrix} 1 & \cdots & x & \cdots \\ 0 & \cdots & 0 & \cdots \end{bmatrix}, \quad (11)$$

where, $x \in \{0, 1\}$.

Since unloaded and loaded manipulator both could execute MOVE task, the explicit form of the operator of the MOVE task is determined by the element x in the first column of the last line of the initial state matrix of the manipulator.

(2) CAPTURE. capture(u) indicates that the uth object in the environment is captured by manipulator. The position of the object is v_u. The end of manipulator and the object should be at the same position before capturing.

Precondition:

$$S' = \begin{bmatrix} v_u & \cdots & v_u & \cdots \\ 0 & \cdots & 0 & \cdots \end{bmatrix},$$
$$M = \begin{bmatrix} 1 & \cdots & 1_u & \cdots \\ 1 & \cdots & 1_u & \cdots \end{bmatrix}. \quad (12)$$

Postcondition:

$$S' = \begin{bmatrix} v_u & \cdots & v_u & \cdots \\ 1 & \cdots & 1 & \cdots \end{bmatrix},$$
$$M = \begin{bmatrix} 0 & \cdots & 0_u & \cdots \\ 1 & \cdots & 1_u & \cdots \end{bmatrix}. \quad (13)$$

(3) RELEASE. release(u) indicates that the uth object in the environment is released by manipulator. The position of the end of the manipulator is v_u. The object should have been caught by manipulator before releasing.

Precondition:

$$S' = \begin{bmatrix} v_u & \cdots & v_u & \cdots \\ 1 & \cdots & 1 & \cdots \end{bmatrix},$$
$$M = \begin{bmatrix} 1 & \cdots & 1_u & \cdots \\ 1 & \cdots & 1_u & \cdots \end{bmatrix}. \quad (14)$$

Postcondition:

$$S' = \begin{bmatrix} v_u & \cdots & v_u & \cdots \\ 0 & \cdots & 0 & \cdots \end{bmatrix},$$
$$M = \begin{bmatrix} 0 & \cdots & 0_u & \cdots \\ 1 & \cdots & 1_u & \cdots \end{bmatrix}. \quad (15)$$

For a multiarm system which consists of m manipulators, the manipulator is represented by the first m columns of the state matrix and its mask, and the object in the environment is represented by remaining columns.

3.2.3. Simplification of Space Manipulator Network Task. Hierarchical task network for space manipulator is established by using the primitive tasks and operators defined above according to HTN theory. The core of HTN planning algorithm is simplifying task and resolving conflicts between tasks. Since the manipulator executes tasks serially, its task planning proceeds sequentially. A strategy that simplifies the tasks and deals with conflicts sequentially is proposed in this paper. The structure of hierarchical network of the planning problem is shown in Figure 2.

The following steps give an explicit procedure of task profile analysis for space manipulator, and its flow diagram is shown in Figure 3.

Step 1. Extract task from the task network orderly.

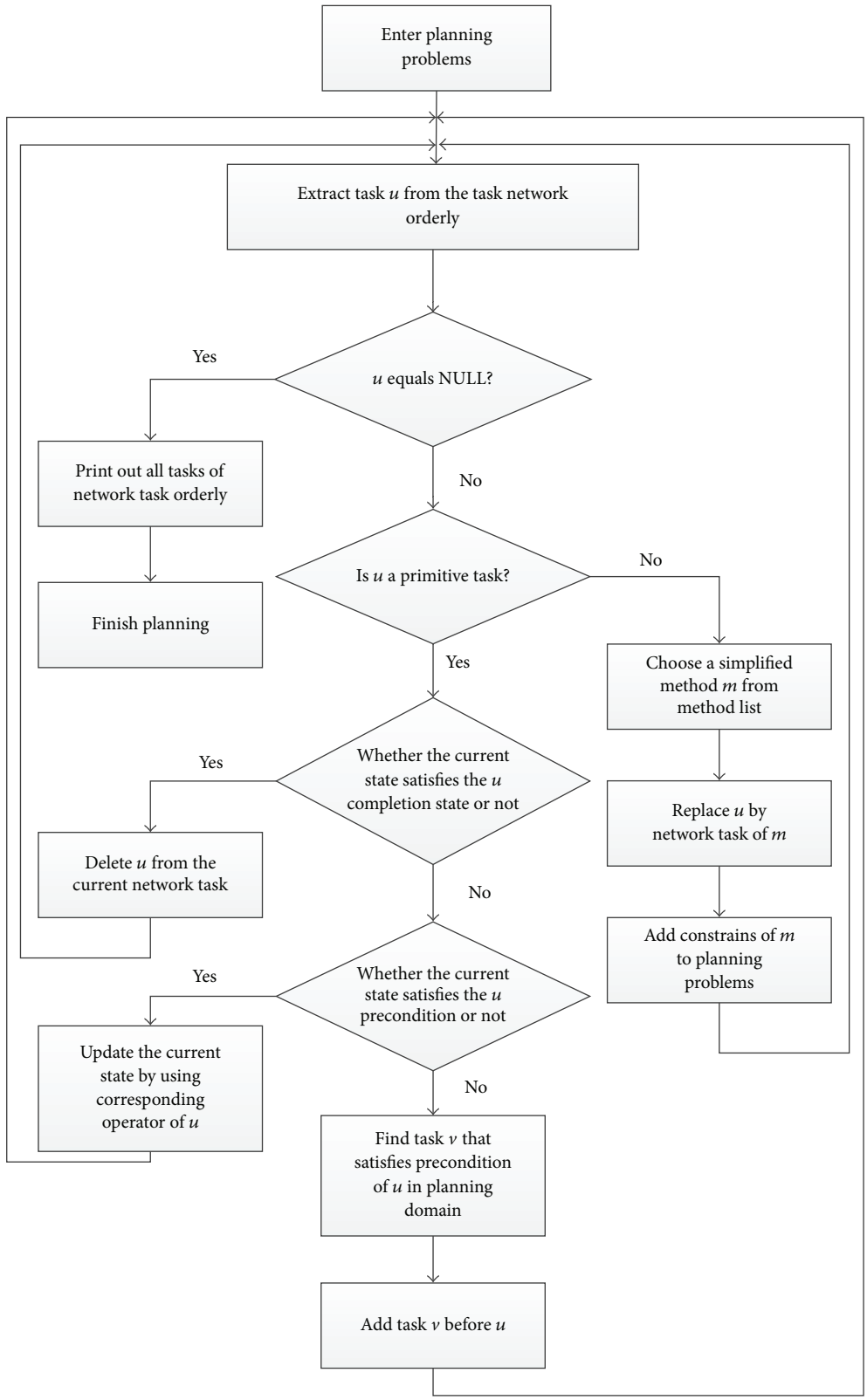


Figure 3: Simplification process of space manipulator network task.

Step 2. If current task is a nonprimitive task, find a corresponding Me in planning domain and replace this nonprimitive task. In order to find an appropriate method, the mask of current task is compared with the masks of all tasks in the method set; if the item which has a 1 in the mask of current task also has a 1 in the mask of method, then this method could be used to simplify this task. Go to Step 1.

Step 3. Compare the current state and the completion state of this task, if current state satisfies the effect of the task, then this task does not need to be executed and is deleted from the task network.

Step 4. If the current state satisfies preconditions of this task, then update the current state by using the operator of this task. Go to Step 1.

Step 5. If the current state does not satisfy the preconditions of this task, then find a task that could satisfy the preconditions in planning domain and add this new task in front of the current task.

Step 6. If the preconditions of the current task could not be satisfied, then return to planning failure.

When the planner starts work, its plan domain is predefined for space manipulator by using the method in Sections 3.2.1 and 3.2.2. Then, the decomposition process follows the steps given in Figure 2. For example, primitive tasks are defined according to Section 3.2.2, and goal task is to move an object U from point A to point B. Then planner would decompose the task into a primitive task sequence of "MOVE to A - CAPTURE U - MOVE to B - RELEASE U." Section 5 shows the example of the decomposition process in detail.

4. Task Node Planning Based on the Improved A* Algorithm

The instantiated HTN planning can be directly applied in many fields, such as logistics transportation and disaster treatment. These benefit from the autonomous capability of the persons who execute the bottom layer of task network. However, the primitive tasks which are planned by HTN could not always be executed by the kind of executor like manipulator which has no autonomous capability. Thus, the task planner of manipulator needs to decompose the primitive task into several path sections which are determined by task nodes and can be planned by a typical trajectory planning algorithm. Then, the manipulator completes the primitive task by using the combination of some simple trajectories which connect the adjacent task nodes.

4.1. Manipulator Collision Detection Based on Point Cloud Model. Task profile analysis makes the task transform from complex to simple, but it is difficult for a primitive task to obtain feasible trajectory by using traditional path planning algorithm because of various constraints. Obstacle avoidance is the only necessary constraint for all tasks; thus it should be payed special attention. In order to satisfy the requirements of precise operation in space, a new high-precision collision detection algorithm for manipulator is proposed in this paper. Then, the task node planning method is achieved based on the collision detection.

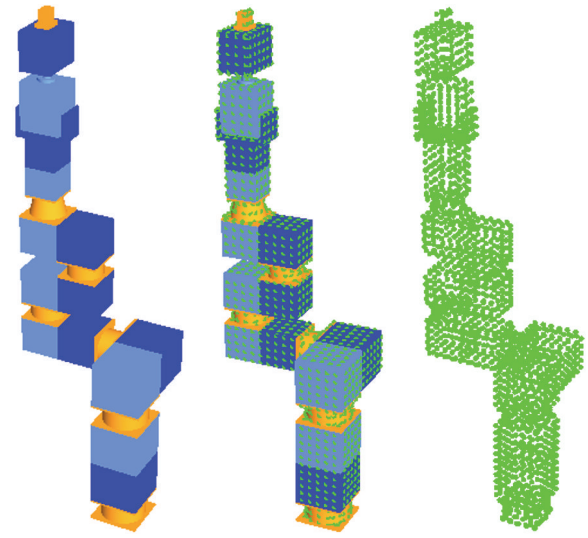


Figure 4: Example of point cloud model of object.

4.1.1. Point Cloud Model of Scene. In order to detect the collision between different parts of the manipulator accurately, a mathematical method that can describe the shape of the whole manipulator and the environment is needed. Basic geometry envelope is widely used in the shape modeling of manipulator collision detection at present. However, it is obvious that if the surface of the object is not regular enough, the basic geometry envelope will cause the model distortion. This will lead to the loss of part of the manipulator's workspace, and will result in the failure of many delicate tasks due to the imprecise collision detection. In addition, this method requires a lot of work of envelope of models, analyzing the shape of model artificially and selecting the approximate geometry. It is entirely unrealistic in space application. In this paper, a method of collision detection for manipulator based on point cloud modeling is proposed. In this method, dense scattered points are used to model for irregular object, then the AABB (Axis Aligned Bounding Box) algorithm is used to achieve the manipulator collision detection.

Point cloud model is defined as dense scattered points which distribute on the surface of the object. All the points in the model are defined in the coordinate system of the object itself. When the object is moving, the rotation matrix of the object can be used to transfer every point of the point cloud model into the inertial coordinate system.

After scanning all the objects in the scene by using Intersect Visitor tool of OSG (OpenSceneGraph), the point cloud model of the whole scene $M = \{M_1, M_2, \ldots, M_N\}$ is obtained; example is shown as in Figure 4, where $M_i \in R^{n_i \times 3}$ is the ith model's point cloud matrix. M_i is composed of n_i 3-dimensional row vectors, which represent the n_i points on the surface of ith object.

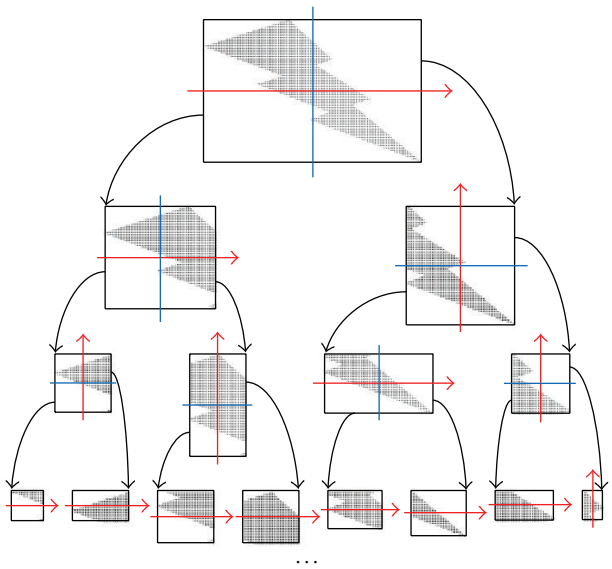


Figure 5: Binary tree of model point cloud.

4.1.2. Build Model's AABB Tree. After obtaining the scene point cloud model, the collision detection of the scene can be done by testing all intersections between M_i and M_j $(i \neq j)$. Intersection test between M_i and M_j $(i \neq j)$ is operated as follows: all points in the M_i and M_j are converted to the same coordinate system (inertial system, e.g.), and whether distance between a pair of points $m_{i,u}$ and $m_{j,v}$ is less than a preset safety threshold β is determined, where points $m_{i,u}$ and $m_{j,v}$ separately belong to M_i and M_j. If a pair of points $m_{i,u}$ and $m_{j,v}$ can be found in the scene, then the ith model and the jth model in the scene collide with each other. Besides, it is considered that the collision point exists in the middle of the connecting line between $m_{i,u}$ and $m_{j,v}$, because the value of β is very small.

Traversing all of the points between two point clouds is the simplest intersection test method, but it will consume large amount of computing resources. In this paper, the points of point cloud model are organized with clusters. AABBs of clusters are used to build AABB tree, and then AABB algorithm is used to detect the collision between any objects in the scene. A cluster is defined as a set of points which are near to each other in the point cloud model. The entire points of a cloud model are C_{root} which is called the root of cluster. Each cluster is divided into left and right child clusters C_l and C_r recursively from the root cluster until only one point is contained in a cluster. By following the steps above, the point cloud of a model can be organized with a binary tree which is shown in Figure 5.

The method of cluster division affects the executing efficiency of the collision detection. In this paper, a cluster is divided into two child clusters according to the principle of dividing from the middle of the longest axis of the model, so that the center of the two child clusters could be far away; thereby the unintersected clusters can be quickly eliminated. In order to determine the longest axis of the model, the maximum and minimum points on X, Y, Z directions are obtained by traversing the model points.

$$\begin{aligned} &P_{X\max}\left(x_{X\max}, y_{X\max}, z_{X\max}\right), \\ &P_{X\min}\left(x_{X\min}, y_{X\min}, z_{X\min}\right), \\ &P_{Y\max}\left(x_{Y\max}, y_{Y\max}, z_{Y\max}\right), \\ &P_{Y\min}\left(x_{Y\min}, y_{Y\min}, z_{Y\min}\right), \\ &P_{Z\max}\left(x_{Z\max}, y_{Z\max}, z_{Z\max}\right), \\ &P_{Z\min}\left(x_{Z\min}, y_{Z\min}, z_{Z\min}\right). \end{aligned} \tag{16}$$

Three axial dimensional sizes of the model can be calculated:

$$\begin{aligned} A_X &= \left|x_{X\max} - x_{X\min}\right|, \\ A_Y &= \left|y_{Y\max} - y_{Y\min}\right|, \\ A_Z &= \left|z_{Z\max} - z_{Z\min}\right|. \end{aligned} \tag{17}$$

Then, the model's longest axis W and its length A_W are obtained.

$$A_W = \max\left\{A_X, A_Y, A_Z\right\}, \tag{18}$$

where W is one of X, Y, Z.

AABB of the model is divided into two parts along the perpendicular bisection plane of axis W; the points belong to one of the two parts of point cloud model forming a new child cluster. A more intuitive way to express the segmentation method is that the vertical plane is made to the W axis at midpoint of $P_{W\max}$ and $P_{W\min}$; each side belongs to a new child cluster. Figure 5 shows an example for a two-dimensional point cloud model which split into binary trees. The red line represents the longest axis of the point cloud cluster W. The blue line shows the line by which the points in the cluster are divided into two child clusters. The blue line would be a plane in 3D point cloud. The partition criterion for the left and right child clusters of model is as follows:

$$\begin{aligned} C_l : &\left\{P\left(x, y, z\right) \mid w \in \left\{x, y, z\right\},\ w \right. \\ &\left. < \frac{\left(w_{W\max} + w_{W\min}\right)}{2}\right\}, \\ C_r : &\left\{P\left(x, y, z\right) \mid w \in \left\{x, y, z\right\},\ w \right. \\ &\left. > \frac{\left(w_{W\max} + w_{W\min}\right)}{2}\right\}, \end{aligned} \tag{19}$$

where w which is determined by (18) is the longest axis of the model.

All nodes of the point cloud binary trees are enveloped with AABB, and then the AABB tree of the model is obtained.

4.1.3. Intersection Test for AABB Trees. AABB binary trees are constructed for every rigid body of manipulator and environment obstacle, and trees which belong to manipulator or environment join together into an AABB forest. Thus, the scene model M is divided into two subsets: manipulator model R and environmental obstacle E.

$$R \cup E = M, \quad R \cap E = \varnothing. \tag{20}$$

Collision detection between forests can be achieved by traversing the AABB trees which are from the two respective forests. Collision detection between every two AABB trees is achieved by examining nodes from the root node to the leave node between two trees. If there are two nodes that collide each other, further examination of the next layer is needed until leaf nodes still collide. If two leafs collide, a collision between the two objects which are represented by the two trees is considered, and the middle of the leaf nodes is the collision point. If there is no collision among the nodes from a layer and the objects which two trees represent do not collide, there is no need of further testing.

If there is still no intersection when the two AABB trees are traversed to the leaf nodes, then the objects which the two AABB trees represent are unintersected. The collision detection between leaf nodes is the process that compares the distance between the leaf nodes which represent model point with the safety margin β. If the distance is less than β, it is considered as a collision. At this moment, the midpoint of the line which connects the two leaf nodes is the collision point.

4.2. Idea for the Resolution of Task Node Planning. For a manipulator with k degrees of freedom, suppose the number of task nodes is n; then there is n intermediate moments. Each node has 6-dimension pose information. Therefore, the dimension of decision vector is $R = n+n+6n+kn = (8+k)n$. It can be seen that the dimension of the decision vector is related to the number of the task node, that is, the dimension of the decision vector itself is also a variable that needs to be decided. So, it is difficult for PSO, GA, and other mainstream intelligent optimization algorithms to solve the problem.

In this paper, heuristic search algorithm is adopted to avoid the problem of deciding the number of task nodes. The number of task nodes can be determined when the algorithm finishes. Because of the intervention of the heuristic item, the algorithm always conducts the solution towards the optimal resources allocation and therefore promises the minimum cost of the whole task.

A* algorithm is a typical heuristic search algorithm in artificial intelligence; the core of the algorithm is to introduce an evaluation function $f(p)$ when selecting the next point at a current point [26].

$$f(p) = g(p) + h(p), \tag{21}$$

where $g(p)$ is the actual cost from the initial point to p in the state space and $h(p)$ is the estimate cost from p to target point.

4.3. Task Node Planning Using Improved A Algorithm.* This paper proposes a variable topology A* algorithm suitable for space manipulator tasks based on the traditional A* algorithm. The improvement idea is that each searching point can not only arrive from the adjacent parent point, but also arrive directly from the parent point of its parent point. These break the fixed topological relationship in traditional A* algorithm search space. When $g(p)$ is being calculated, the traditional method of "the actual cost of arriving father point" + "the cost of parent point to the current point" is not used. Instead, by using parent point as the guide, the algorithm looks back to the farthest feasible start point. The costs of various ways that can reach the current point are calculated, and the plan with the least cost is selected. The improvement of calculating traditional A* algorithm's $g(p)$:

$$g'(p_c) = g\left(p'_f\right) + C\left(p'_f, p_c\right), \tag{22}$$

where $C(p_1, p_2)$ is the actual cost from point p_1 to point p_2, p_c is current point, and p_f is the father point of the current point. Boundary conditions: $C(p_1, p_1) = 0$.

Define the pseudofather point p'_f: p'_f of p_c as the point between "p_f" and "p'_f of p_f" which makes $g'(p_c)$ minimal. Boundary conditions: the p'_f of the start point p_s is itself.

Then $g'(p_c)$ can be expressed as

$$g'(p_c) = \min\left\{g\left(p_g\right) + C\left(p_g, p_c\right), g\left(p_f\right) + C\left(p_f, p_c\right)\right\}, \tag{23}$$

where p_g is the pseudofather point of the father point of the current point.

Improved algorithm process is as follows.

Step 1. Set the search dimension of the planner as D, search step of position as Δl, and search step of attitude as $\Delta\theta$.

Step 2. Set up two empty lists: OPEN list and CLOSE list.

Step 3. Set the pseudofather point of the start point (pointStart) as itself and add it to the OPEN list, and then calculate $f(p)$.

Step 4. If the CLOSE list contains the target point pointEnd, go to Step 12.

Step 5. If the OPEN list is empty, the target is not reachable and the planning ends.

Step 6. Select the point with smallest $f(p)$ in OPEN list as the current node and move it to CLOSE list.

Step 7. Generate candidate point set P by moving the current point ± 1 step in all dimensions.

Step 8. Eliminate the point which is in CLOSE list or in violation of the constraints from P.

Step 9. Use the typical path planning algorithm to calculate $f(p)$ of all points in P, respectively, and remove the nodes in P which violate constraints in the process of path planning.

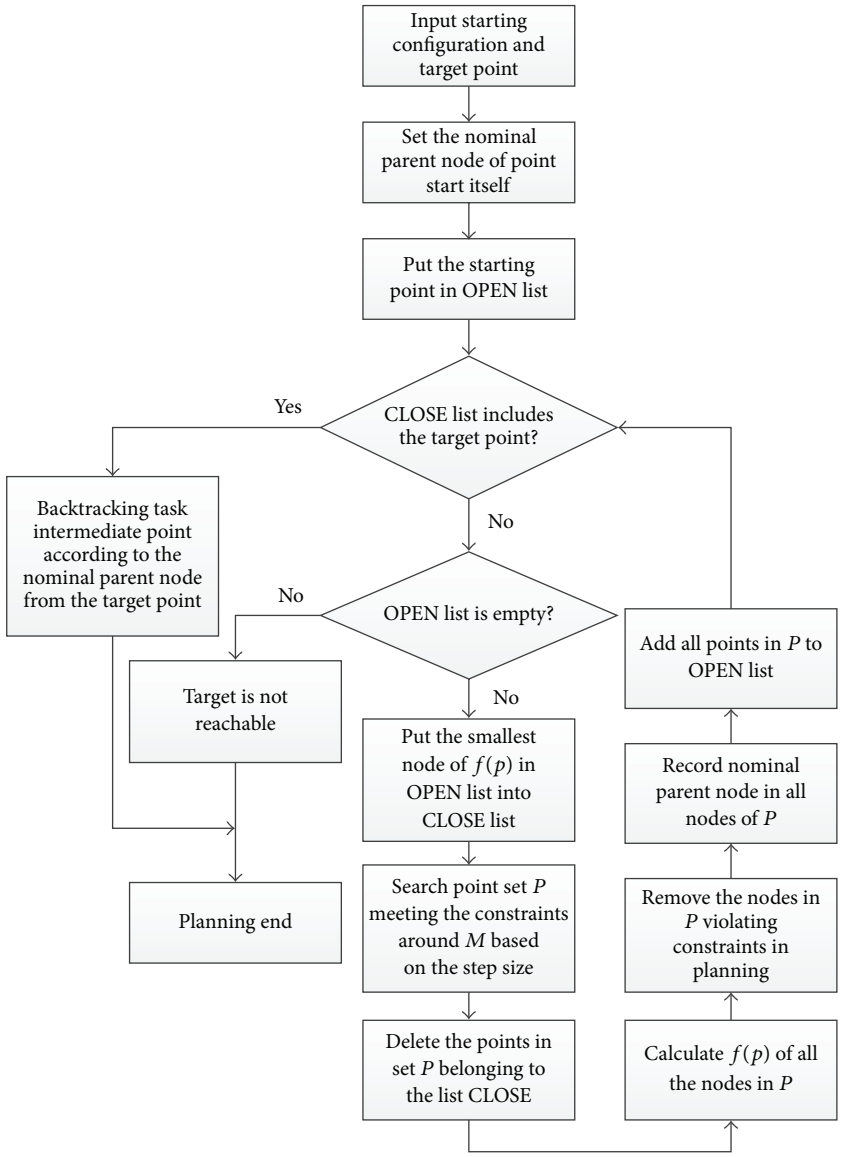


Figure 6: Flow chart of task node planning.

Step 10. Record pseudofather point p'_f of all nodes in P and move all points of P in the OPEN list.

Step 11. Go to Step 4.

Step 12. Go back from the target point (pointEnd) to the start point (pointStart) through pseudofather points to derive the entire path, that is, all the task nodes.

The flowchart of improved algorithm is shown as in Figure 6.

In this paper, joint stroke during the task which is directly related to the life of the transmission system and affects the reliability of the space agency in orbit is used as the optimal resource, using "straight path planning" as the typical path planning algorithm. The cost between two points is $C(p_1, p_2)$:

$$C(p_1, p_2) = \sum_{i=1}^{k} \left| q_{2,i} - q_{1,i} \right|, \tag{24}$$

where k is the degree of freedom of manipulator and the initial and final configuration of the straight path planning are, respectively, $[q_{1,1}, q_{1,2}, \ldots, q_{1,k}]$ and $[q_{2,1}, q_{2,2}, \ldots, q_{2,k}]$.

End velocity, end acceleration, the collision interference of manipulator, and the environment obstacles are chosen as the constraints of the planner, where the velocity and acceleration constraints are guaranteed by the path planning algorithm; obstacle avoidance constraints are calculated by

applying AABB algorithm to each configuration of path planning result.

The proposed task node planning method is mainly used for finding a path that satisfies one or more constraints. The planed path is given in the form of several nodes and needs a trajectory planning method to connect these nodes. For manipulator, trajectory planning is an existing and mature technology, so task node planning can be applied well to the manipulator. In addition, task node planning also can be applied in other resources, if there is trajectory planning method for the application object.

4.4. Planning for Dual-Arms Robot. For single arm system, the algorithm can decompose the path that the manipulator cannot complete directly under multiple constraints into several simple paths satisfying the constraint. In multiarm systems, one arm's movement relative to the other can be regarded as obstacles constraints. However, this obstacle constraint which is different from general object in the environment is of time-varying dynamic properties, for multiple manipulators always work at the same time. Thus, it is difficult to avoid interference between multiarms by directly using the single arm planning algorithm to each single arm, respectively, in a multiarm system. This paper introduces the concept of virtual obstacles and proposes a planning strategy to solve the conflict between multiarms. A dual-arms task planning algorithm based on this strategy can make manipulator complete the work independently in the same space and at the same time and does not interfere with each other.

For a multiarm system consisting of m manipulators, the scene model M is decomposed into $m + 1$ subset: $E, M^{(1)}, M^{(2)}, \ldots, M^{(m)}, M^{(i)}$ $(i = 1, \ldots, m)$ is m manipulators, using this model sets to reform m virtual scenes $\{E, M^{(i)}\}$ $(i = 1, \ldots, m)$. It means that one arm of the multiarm system is in the scene and other arms are ignored. Do single arm task planning, respectively, for the m virtual scene $\{E, M^{(i)}\}$ $(i = 1, \ldots, m)$, and this process for the multiarm system is defined as MADP (Multi-Arm Distributed Planning). After that, m mission planning results K_i $(i = 1, \ldots, m)$ can be gotten. m virtual scene is planning independently, so each of the manipulators in the planning process does not take into account the impact of other manipulators' movement. Therefore, these manipulators are very likely to interfere with each other when the planning results are overlapped in a common environment.

The direct cause of the interference of multiarm system in common working space is that the collided parts of two manipulators should not appear in the same space at the same time. One way to solve this contradiction is to introduce time and space constraints at all collision points in the scene. So, the manipulators that will have a collision could avoid a particular location at a certain time. The simplest approach to add time and space constraint is to add a small virtual cube obstacle whose length is a configurable fixed value γ at each collision point. Set $G = \{G_1, G_2, \ldots, G_\omega\}$, where ω is the number of collision points between manipulators after the fusion of m virtual scene planning and G_i $(i = 1, \ldots, \omega)$ is the virtual obstacle model added to the collision point. Environment model E can be updated by using the following formula:

$$E^{(i+1)} = E^{(i)} \cup G^{(i+1)}, \tag{25}$$

where $E^{(0)} = E$ and $G^{(i+1)}$ is the new virtual obstacle set of the virtual scene $\{E^{(i)}, M^{(j)}\}$ $(j = 1, \ldots, m)$ after a MADP. Virtual obstacle elements of the set G are not real objects in the scene; they are only used to avoid the collision between multiarm system in some particular moment. So, G_i $(i = 1, \ldots, \omega)$ does not always need to appear during the task and just needs to be added to the scene at the time before and after the moment of collision $[t_i - \Delta\delta, t_i + \Delta\delta]$, where t_i is the moment of collision between manipulators and $\Delta\delta$ is a configurable avoiding time for safety. So, the environment model with time-varying characteristic is as follows:

$$\begin{aligned} E^{(i+1)}(t) = {} & E^{(i)}(t) \\ & \cup \left\{G_1^{(i+1)}(t), G_2^{(i+1)}(t), \ldots, G_\omega^{(i+1)}(t)\right\}, \end{aligned} \tag{26}$$

where expression of $G_i(t)$ is

$$G_i(t) = \begin{cases} G_i, & t \in [t_i - \Delta\delta, t_i + \Delta\delta], \\ \varnothing, & t \notin [t_i - \Delta\delta, t_i + \Delta\delta]. \end{cases} \tag{27}$$

Dual-arm task planning strategy is as follows: using (26) iterative algorithm to update environment model E until virtual obstacle set $G = \varnothing$; the scheme after MADP planning is feasible dual-arm system task planning results.

Table 1: Manipulator D-H parameter table.

Numerical order	Joint angle $\theta/^\circ$	Torsion angle $\alpha/^\circ$	The length of the connecting rod a/m	Connecting rod bias d/m
1	90	0	0	0.38
2	−90	−90	0	0.11
3	0	90	0	0.24
4	0	0	0.13	0
5	−90	0	0.13	0
6	0	−90	0	0.30
7	0	90	0	0
8	0	−90	0	0.09

5. Experiment and Simulation

An 8-DOF fixed base manipulator is used as the test subject to verify the task plan method in this paper. Figure 7 shows the coordinate systems definition of manipulator, and Table 1 shows the D-H parameter of this manipulator.

A manipulator simulation system with 3D simulation unit which is based on OpenGL is built under IDE of VS2010. The

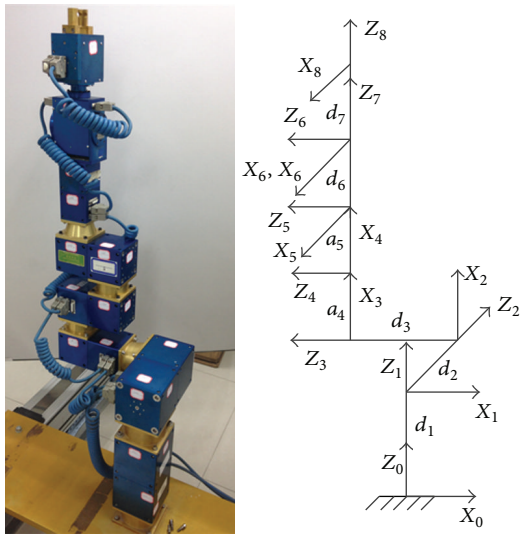


Figure 7: Coordinate systems definition of manipulator.

whole task cycle of manipulator can be simulated visually by this simulation system. The hardware platform for simulation is CPU Intel i3 3.3 GHz, 4 GB RAM. Manipulator control cycle is 50 ms.

There are a manipulator M, an obstacle O, and an object U to be operated in the simulation environment. The initial conditions are as follows: operation interface of U is $A = [-0.54\,\text{m}, 0.45\,\text{m}, 0.18\,\text{m}, 131.78^\circ, -79.64^\circ, -177.62^\circ]^{\text{T}}$; initial configuration of M is $[16^\circ, 18.1^\circ, 67.6^\circ, 56.2^\circ, 21.9^\circ, -29.5^\circ, -41.4^\circ, 0^\circ]^{\text{T}}$. Planning domain of planner is configured as the three primitive task operators described in this paper and a compound tasks "transfer":

$$(\text{trans}(u, v), (n_1 : \text{do}[\text{capture}(u)], n_2 : \text{do}[\text{move}(v)], n_3 : \text{do}[\text{release}(u)], (n_1, n_2, n_3))). \quad (28)$$

The manipulator M only has the ability to plan linear trajectory in Cartesian space and can only move in a straight line. The task goal is to transfer U to the position $C = [-0.54\,\text{m}, 0.55\,\text{m}, 0.18\,\text{m}, 131.78^\circ, -79.64^\circ, -177.62^\circ]^{\text{T}}$.

5.1. Example of Task Profile Analysis for Object Transfer. The goal task network is described as (achieve(trans(U, C))). The end executor's coordinate can be calculated as $B = [-0.48\,\text{m}, -0.51\,\text{m}, 0.18\,\text{m}, -142.09^\circ, -79.64^\circ, -177.62^\circ]^{\text{T}}$ based on the initial configuration and the robot forward kinematics. So, the initial status to the environment is $\begin{bmatrix} B & A \\ 0 & 0 \end{bmatrix}$.

(1) The goal network contains only one nonprimitive task. Thus, search the corresponding Me; in the planning domain, trans(U, C) is replaced by (28).

(2) By retraversing the whole task network, n_1 is confirmed as a primitive task. Then, the current state is compared with the prerequisite of capture(u) by using the state criterion: $\begin{bmatrix} B & A \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} A & A \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} B-A & 0 \\ 0 & 0 \end{bmatrix}$. And, mask of the wrong state is $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$.

(3) By search planning domain, move(v) operator can eliminate this difference of state. So a new task network (n_4 : do[move(A)]) is added in front of n_1. Task network constraints become (n_4, n_1, n_2, n_3).

(4) On retraversing task network again, initial state $\begin{bmatrix} B & A \\ 0 & 0 \end{bmatrix}$ transfers to $\begin{bmatrix} A & A \\ 0 & 0 \end{bmatrix}$ through n_4 operator, then transfers to $\begin{bmatrix} A & A \\ 1 & 1 \end{bmatrix}$ through n_1 operator, then transfers to $\begin{bmatrix} C & C \\ 1 & 1 \end{bmatrix}$ through n_2 operator, and then transfers to $\begin{bmatrix} C & C \\ 0 & 0 \end{bmatrix}$ through n_3 operator at last.

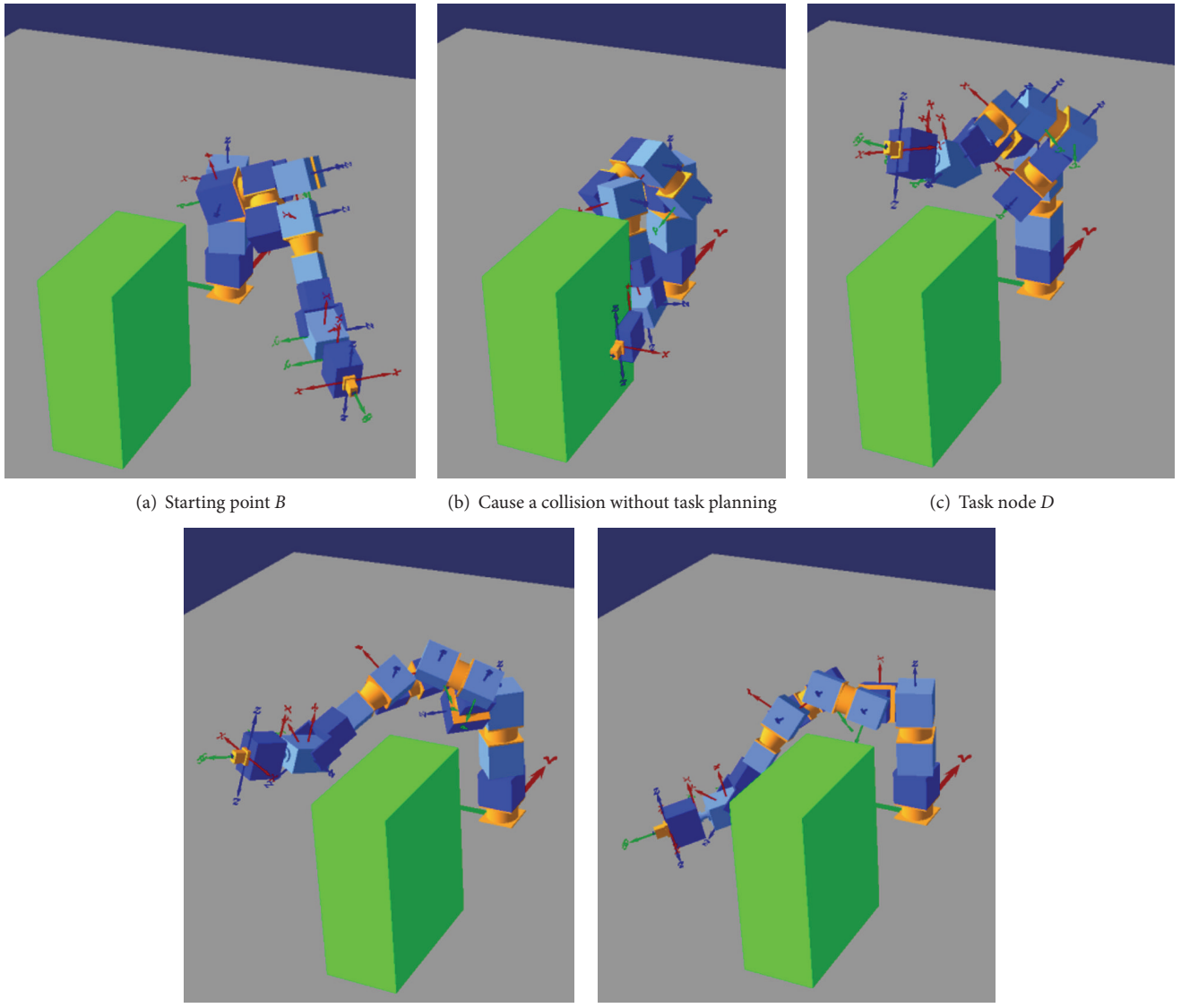
At this moment, the task network has no conflict any more. The primitive task sequence which is given by planner is move(A) → capture(U) → move(C) → release(U). Objects U are successfully transferred to C.

5.2. Simulation of Task Node Planning for Moving Task. The primitive task sequence which is given by task profile analysis is checked by typical path planning method. In this check, move(A) cannot be executed directly, because of the environmental obstacle. The initial configuration of the primitive task is $[16^\circ, 18.1^\circ, 67.6^\circ, 56.2^\circ, 21.9^\circ, -29.5^\circ, -41.4^\circ, 0^\circ]^{\text{T}}$; the target point $A = [-0.54\,\text{m}, 0.45\,\text{m}, 0.18\,\text{m}, 131.78^\circ, -79.64^\circ, -177.62^\circ]^{\text{T}}$. Set the planner parameters as follows: the end velocity constraints are <0.03 m/s, the end acceleration constraints are $<0.03\,\text{m/s}^2$, configuration constraints are to avoid collision from manipulator itself, and environmental constraint is to avoid obstacles O; planner search dimension is three and search step is 0.05 m. Use the improved A* algorithm to plan the task node; the movement of manipulator is shown in Figure 8.

It can be seen in the figure that direct path planning between B and A will cause manipulator's collisions as shown in Figure 8(b). With the introduction of task planning, the task nodes $D = [-0.68\,\text{m}, -0.01\,\text{m}, 0.73\,\text{m}, 16.04^\circ, -55^\circ, -31.51^\circ]^{\text{T}}$ and $E = [-0.68\,\text{m}, 0.34\,\text{m}, 0.48\,\text{m}, -42.4^\circ, -71.62, -17.19^\circ]^{\text{T}}$ are added into the movement of B to A. These make the manipulator M reach A from B passing D and E and bypass obstacles at the same time. All manipulator joints travel with minimum stroke under this condition.

On the basis of single arm task node planning simulation, a dual-arms task node planning simulation is carried out. A new manipulator which is completely symmetrical with the manipulator used before is added to the environment. Then, a dual-arms system is constructed which is shown in Figure 9.

The planning parameters in previous section are reused. In addition, side length of virtual obstacles is $\gamma = 0.1\,\text{m}$ and avoiding time is $\Delta\delta = 10\,\text{s}$. A new task is set: initial configurations of manipulator 1 are $[-48^\circ, -25.7^\circ, 126.7^\circ, -10.5^\circ, -25.7^\circ, 0^\circ, -6.7^\circ, 0^\circ]^{\text{T}}$, initial configurations of manipulator 2 are $[18.1^\circ, -33.3^\circ, -56.2^\circ, -48.6^\circ, -27.6^\circ, 0^\circ, -58.1^\circ, 0^\circ]^{\text{T}}$, manipulator 1 is required to move to point $F = [-0.348\,\text{m}, 0.43\,\text{m}, 0.654\,\text{m}, 99.12^\circ, -77.35^\circ, -180^\circ]^{\text{T}}$, and manipulator 2 is required to move to point $G = [-0.463, 0.43\,\text{m}, 0.733\,\text{m}, 36.67^\circ, 25.78^\circ, -142.66^\circ]^{\text{T}}$ at the same time. When simple linear trajectory path planning is used for arms, the task is broken by a collision between these two arms, as shown



(a) Starting point B (b) Cause a collision without task planning (c) Task node D

(d) Task node E (e) Target point A

Figure 8: Simulation movement of task node planning.

in Figure 10(a). Then, multiarm task planning algorithm is used to plan the task again, and the planner of manipulator adds a task node for manipulators 1 and 2, respectively: $H = [-0.516\,\text{m}, 0.342\,\text{m}, 0.159\,\text{m}, 48.59^\circ, -73.28^\circ, -117^\circ]^{\text{T}}$ and $I = [-0.645\,\text{m}, 0.355\,\text{m}, 0.502\,\text{m}, 25.32^\circ, 15.18^\circ, -122.61^\circ]^{\text{T}}$. At this time, the two manipulators can complete their tasks at the same time satisfying all constraints, as shown in Figure 10.

It can be proved through simulation that the proposed algorithm can well decompose the object moving task and give the feasible task nodes for manipulator. And, the whole process is smooth and continuous.

6. Conclusion

In this paper, a double-layer framework of task plan for complex working environment is proposed considering the characteristic of space manipulator and its various kinds of on-orbit servicing missions. First, state matrix, state mask, and their relative operation are proposed based on the characteristic of space manipulator. Meanwhile, three typical primitive tasks and their operators of manipulator are defined in order to realize the HTN planner for space manipulator. Then, complex tasks of manipulator can be analyzed and resolved by using HTN automatically. Second, in order to ensure the feasibility of the task, collision avoidance which is a necessary constraint for all tasks is given special attention. A collision modeling method based on point cloud model is proposed in this paper and is applied to the AABB algorithm for collision detection. The improved collision detection method could get higher precision with little cost of calculating time, thus satisfying the requirement of space application. Finally, a task node planning method is designed by improving the A* algorithm. Based on this, manipulator

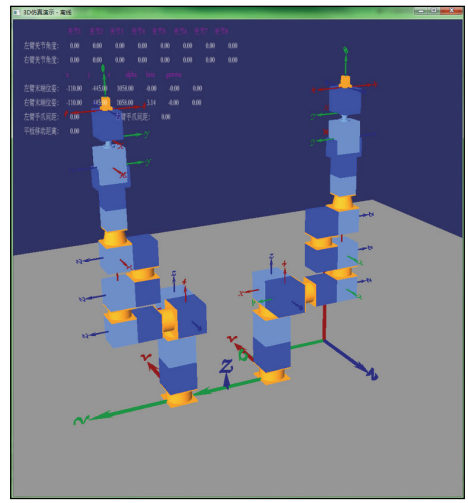
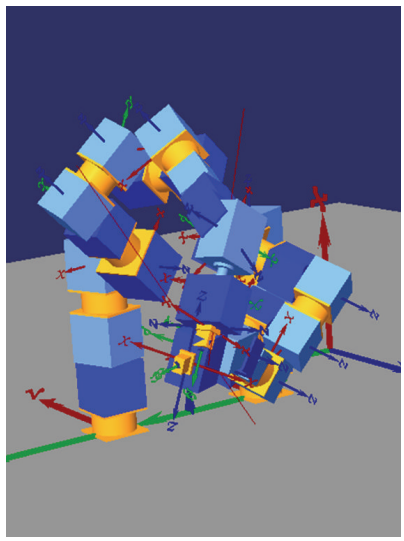
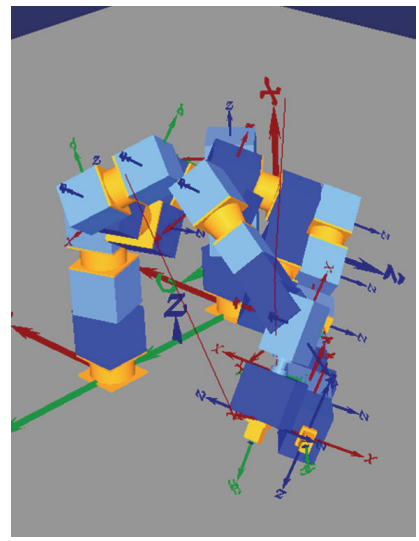


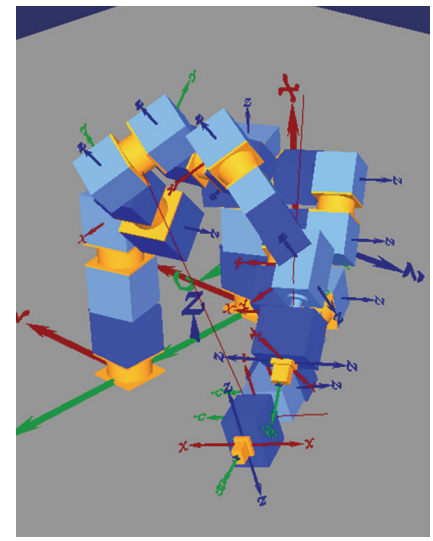
Figure 9: 16-DOF manipulator system.



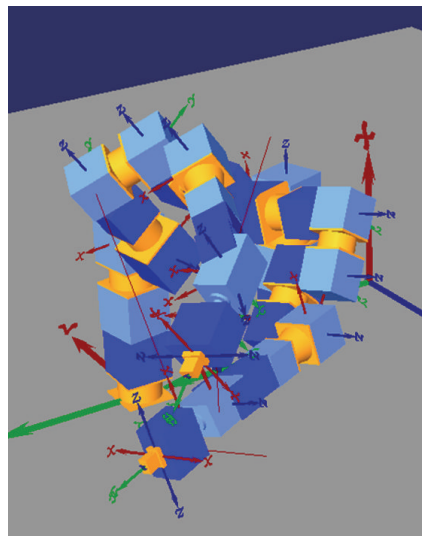
(a) Cause collision without task planning



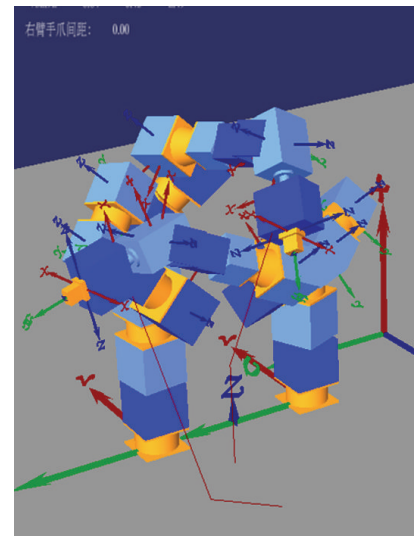
(b) Initial configuration



(c) Task node H



(d) Task node I



(e) Target of task

Figure 10: Simulation of moving task for dual-arm system.

could finish the task considering multiconstraint. In addition, the problem of collision between multiarms is also solved by introducing the concept of virtual obstacles.

A typical task of space manipulator is simulated based on a 3D visual manipulator simulation system. It has been shown that manipulator could finish a complex task successfully under the condition of multiconstrains by using a combination of simple paths. The task planning method has high computation efficiency, so that the computation of a typical task for an 8-DOF manipulator can be finished in 1-2 minutes on a small business computer. Thus, the method would work fine with spaceborne computer for space application.

In addition, HTN is a universal task plan method for lots of fields. However, when it is applied to a field, it may need to have some changes in order to adapt the field's characteristic. For example, when we apply HTN to the space manipulator's task planning, we designed a characterization method for space manipulator's primitive task. Based on this characterization method, various kinds of manipulator tasks can be calculated mathematically, and HTN is used for space manipulator successfully. Thus, HTN can also be used for industrial related fields, but some work has to be done in order to adapt the method to related fields. Task node planning in this paper is a peculiar planning method for manipulator. It is appropriate for manipulator's characteristic of smooth and continuous movement. So, task node planning is not suitable for other industrial fields, unless there is a trajectory planning method to guarantee the smooth and continuous movement.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The authors would like to thank their colleagues from the Robotics Research Group for helpful discussions and comments on this paper. This study was supported by the National Basic Research Program of China [2013CB733000] and the National Natural Science Foundation of China [61403038, 61573058, 61573066].

References

[1] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich, "A review of space robotics technologies for on-orbit servicing," *Progress in Aerospace Sciences*, vol. 68, pp. 1–26, 2014.

[2] C. F. Lillie, "On-orbit assembly and servicing for future space observatories," in *Proceedings of the Space Conference*, American Institute of Aeronautics and Astronautics, San Jose, Calif, USA, September 2006.

[3] M. Chu, Y. Zhang, G. Chen, and H. Sun, "Effects of joint controller on analytical modal analysis of rotational flexible manipulator," *Chinese Journal of Mechanical Engineering*, vol. 28, no. 3, pp. 460–469, 2015.

[4] R. E. Fikes and N. J. Nilsson, "Strips: a new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[5] K. Erol, J. Hendler, and D. S. Nau, "HTN planning: complexity and expressivity," in *Proceedings of the 12th National Conference on Artificial Intelligence. Part 1 (of 2)*, pp. 1123–1128, Seattle, Wash, USA, August 1994.

[6] E. Erol, J. Hendler, and D. S. Nau, "Semantics for hierarchical task-network planning," Tech. Rep. CS TR-3239, UMIACS TR-94-31, ISR-TR-95-9, Institute for Systems Research, University of Maryland, College Park, Md, USA, 1994.

[7] K. Erol, J. Hendler, and D. S. Nau, "UMCP: a sound and complete procedure for hierarchical task-network planning," in *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS '94)*, Chicago, Ill, USA, June 1994.

[8] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 1470–1477, Institute of Electrical and Electronics Engineers, Shanghai, China, May 2011.

[9] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '14)*, Institute of Electrical and Electronics Engineers, Chicago, Ill, United states, September 2014.

[10] J. Barry, L. P. Kaelbling, and T. Lozano-Perez, "A hierarchical approach to manipulation with diverse actions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '13)*, pp. 1799–1806, IEEE, Karlsruhe, Germany, May 2013.

[11] Z. Honghao, J. Dapeng, P. Yongjie et al., "Workflow modeling and management on AUV," *CAAI Transactions on Intelligent System*, no. 5, pp. 433–438, 2013.

[12] D. Zong, C.-P. Li, S.-H. Xia, and Z.-Q. Wang, "Hierarchical task-planning method based on key-state for virtual human," *Journal of System Simulation*, vol. 7, pp. 1535–1542, 2013.

[13] S. Singh, R. Simmons, T. Smith et al., "Recent progress in local and global traversability for planetary rovers," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, pp. 1194–1200, IEEE, San Francisco, Calif, USA, April 2000.

[14] P. K. Kim, H. Park, J.-H. Bae, J.-H. Park, M.-H. Baeg, and J. Park, "Practical control strategy for packing multiple boxes simultaneously with dual-arm robot," in *Proceedings of the 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI '14)*, pp. 567–571, Institute of Electrical and Electronics Engineers, Kuala Lumpur, Malaysia, November 2014.

[15] P. K. Kim, J.-H. Bae, J.-H. Park, M.-H. Baeg, and J. Park, "Control strategy for simultaneously inserting multiple boxes with dual arm manipulator," in *Proceedings of the 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI '13)*, pp. 251–254, Jeju, South Korea, November 2013.

[16] S. Charoenseang, A. Srikaew, D. M. Wilkes, and K. Kawamura, "Integrating visual feedback and force feedback in 3-D collision avoidance for a dual-arm humanoid robot," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3406–3411, IEEE, San Diego, Calif, USA, October 1998.

[17] D.-H. Kim, S.-J. Lim, D.-H. Lee, J. Y. Lee, and C.-S. Han, "A RRT-based motion planning of dual-arm robot for (Dis)assembly tasks," in *Proceedings of the 44th IEEE International Symposium on Robotics (ISR '13)*, pp. 1–6, IEEE, Seoul, South Korea, October 2013.

[18] J.-S. You, D.-H. Kim, S.-J. Lim, S. Kang, J. Y. Lee, and C. Han, "Development of manipulation planning algorithm for a dual-arm robot assembly task," in *Proceedings of the IEEE International Conference on Automation Science and Engineering: Green Automation Toward a Sustainable Society (CASE '12)*, pp. 1061–1066, IEEE Computer Society, Seoul, Republic of Korea, August 2012.

[19] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '09)*, pp. 2464–2470, IEEE Computer Society, St. Louis, Mo, USA, October 2009.

[20] T. Takahama, K. Nagatani, and Y. Tanaka, "Motion planning for dual-arm mobile manipulator—realization of 'tidying a room motion'," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4338–4343, IEEE, New Orleans, La, USA, April 2004.

[21] R. Qi, W. Zhou, and T. Wang, "An obstacle avoidance trajectory planning scheme for space manipulators based on genetic algorithm," *Robot*, vol. 36, no. 3, pp. 263–270, 2014.

[22] J. B. Chen, M. Zhao, and H. Zhang, "On-line real-time obstacle avoidance path planning of space manipulator," *Control Engineering of China*, vol. 14, no. 4, pp. 445–450, 2007.

[23] Y. J. Yoo, K. J. Oh, Y. J. Choi, and S. C. Won, "Obstacle avoidance for redundant manipulator without information of the joint angles," in *Proceedings of the 9th Asian Control Conference (ASCC '13)*, pp. 1–6, IEEE, Istanbul, Turkey, June 2013.

[24] K. Kaltsoukalas, S. Makris, and G. Chryssolouris, "On generating the motion of industrial robot manipulators," *Robotics and Computer-Integrated Manufacturing*, vol. 32, pp. 65–71, 2015.

[25] M. R. Pedersen, L. Nalpantidis, R. S. Andersen et al., "Robot skills for manufacturing: from concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[26] Q. Jia, G. Chen, H. Sun, and S. Zheng, "Path planning for space manipulator to avoid obstacle based on A* algorithm," *Journal of Mechanical Engineering*, vol. 46, no. 13, pp. 109–115, 2010.