Robotica (2017) volume 35, pp. 101–118. © Cambridge University Press 2015
doi:10.1017/S0263574714002963

A new motion planning method for discretely actuated hyper-redundant manipulators

Alireza Motahari†*, Hassan Zohoor‡ and Moharam Habibnejad Korayem§

† *Department of Mechanical and Aerospace Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*
‡*Center of Excellence in Design, Robotics and Automation, Sharif University of Technology, &, The Academy of Sciences, Tehran, Iran*
§*Robotics Research Laboratory, Center of Excellence in Experimental Solid Mechanics and Dynamics, School of Mechanical Engineering, Iran University of Science and Technology, Tehran, Iran*

(Accepted December 29, 2014. First published online: February 27, 2015)

SUMMARY
A hyper-redundant manipulator is made by mounting the serial and/or parallel mechanisms on top of each other as modules. In discrete actuation, the actuation amounts are a limited number of certain values. It is not feasible to solve the kinematic analysis problems of discretely actuated hyper-redundant manipulators (DAHMs) by using the common methods, which are used for continuous actuated manipulators. In this paper, a new method is proposed to solve the trajectory tracking problem in a static prescribed obstacle field. To date, this problem has not been considered in the literature. The *removing first collision* (RFC) method, which is originally proposed for solving the inverse kinematic problems in the obstacle fields was modified and used to solve the motion planning problem. For verification, the numerical results of the proposed method were compared with the results of the *genetic algorithm* (GA) method. Furthermore, a novel DAHM designed and implemented by the authors is introduced.

KEYWORDS: Binary manipulator; Discrete actuation; Hyper-redundant manipulator; Motion planning; Obstacle avoidance.

1. Introduction

A hyper-redundant manipulator may be built by stacking serial or parallel robots as modules. High degrees of freedom of such manipulators enable them to avoid obstacles, and to have high dexterity; however, high degrees of freedom result in motion and control complexity. To solve the problem, some researchers have recommended the use of discrete actuators instead of conventional continuous ones, because controlling discrete actuators is easier.[1,2] Discrete actuators have only a few stationary states. For instance, a binary-prismatic actuator may be actuated, either in a completely retracted or in an expanded state. Discretely actuated manipulators in comparison with continuously actuated manipulators are cheaper and also have high repeatability and accuracy.

In spite of such advantages, discrete manipulators have not been taken much into consideration in industries and also in research studies. Given the current status, more research needs to be carried out in kinematics, dynamics, and controlling of these manipulators before they could find their way into industries. In fact, several experimental samples of such manipulators have been built so far. Ebert-Uphoff[3] made a binary manipulator with five Stewart-Gough modules. Suthakorn and Chirikjian[4] designed and implemented a new spatial DAHM with three modules. Sujan *et al*. designed a light weight binary manipulator for space exploration applications.[2] Employing discrete actuators is not restricted to hyper-redundant manipulators. For examples, one could refer to refs. [5–7].

* Corresponding author. E-mail: a.motahari@srbiau.ac.ir

A novel spatial manipulator with three modules is designed and implemented in this paper. Each module in this manipulator involves a 3-RPS parallel mechanism. Thus, each module has three degrees of freedom and the manipulator possesses a total of nine degrees of freedom. Details of the manipulator are presented in Section 4.1.

The present research deals with motion planning of DAHMs. To solve the motion planning problem, first of all it is necessary to decide on an effective method to solve the inverse kinematic problem. In discrete manipulators, each actuator has only a few stationary states; therefore, the manipulator will have a limited number of configurations and its workspace will be similar to a cloud of discrete points. In solving the inverse kinematic problems, the first idea coming to the mind is to calculate the end-frame position by solving the forward kinematic problem for all manipulator configurations and store it. Then, the configuration whose end-frame is closest to the target-frame should be chosen from among all configurations. However, this method takes a lot of calculation time and memory for manipulators with a large number of modules, considering the fact that the number of configurations increases exponentially with an increase in the number of manipulator modules. For example, solving an inverse kinematic problem for a seven-module variable geometry truss (VGT) manipulator using MATLAB software with an Intel 1.66 GHz processor (a common PC) takes more than one hour. The VGT manipulator is a discrete manipulator whose modules have eight configurations each. The manipulator will be introduced in the following section.

The second approach for solving inverse kinematics is using the methods originally proposed for continuous manipulators. However, such methods are not useful for DAHMs, because at the last step of solution, the results (actuations' amounts) should be substituted with the nearest discrete amounts. This causes large errors, especially when the substitution is made for the actuators of the modules, which are closer to the base.[8] Therefore, the researchers decided to search for better approaches. Ebert-Uphoff and Chirikjian[9,10] proposed a method in order to estimate workspace density and used it to solve the inverse kinematic problem. In this method, it is necessary to perform an offline evaluation of and storing large amounts of data which causes problems, especially in 3D cases. Suthakorn and Chirikjian[4] proposed an effective method for evaluating the mean frame of the DAHMs workspace and used it to solve the inverse kinematic problem. Their method was fast and the volume of offline calculations and stored data were not huge. However, it had the disadvantage of having large errors. Motahari *et al*.[11] proposed an effective method for solving inverse kinematic problem of DAHMs. Their method was based on the one proposed by Suthakorn. They modified Suthakorn's method by incorporation of two novelties: two-by-two searching (instead of one-by-one searching) and iteration of the searching process. The method is fairly effective in terms of calculation time, error, and memory in use. Furthermore, two other problems, namely, discrete synthesis and obstacle avoidance may be effectively solved by using this method.[12,13] Thus, the two-by-two searching method[11] was chosen from among all other proposed methods in order to solve the inverse kinematic problem of DAHMs.

The motion planning problem may not be solved without dealing with obstacle avoidance problem. Therefore, it is necessary to choose an effective method to solve the obstacle avoidance problem. A search for studies on obstacle avoidance of DAHMs yielded only one result—a study reported by Lanteigne and Jnifene.[14] They solved 2D problems by using the workspace density method of inverse kinematic solution.[9,10] Therefore, their method suffered the same problems of workspace density as that of the method mentioned earlier in this paper. Motahari *et al*.[13] proposed a new method, called "RFC" in order to solve the obstacle avoidance problem. The speed and precision of this method is fairly acceptable. Furthermore, this method might be applied to solve 3D problems. Thus, the RFC method[13] was selected to solve the obstacle avoidance problem.

There is a sizable body of research on the motion planning problem of continuously actuated hyper-redundant manipulators. The proposed methods may be divided into two major branches: the potential field methods[15,16] and the road map methods.[17,18] However, as mentioned earlier about the continuous methods of solving the inverse kinematic problem, the results of such methods are not necessarily a discrete quantity, and they should be substituted by the nearest discrete quantities. This in turn causes results in errors.

While, there are a few studies in which the problem of motion planning of DAHMs has been addressed, none of them have considered the obstacles. Chirikjian[8] proposed the back-bone curve method (a continuous method) in order to solve the inverse kinematic problem. In this method, the macroscopic view of the manipulator is introduced by a curve called "back-bone curve" to which the manipulator should be attached. Then, the trajectory is defined by some consecutive target-frames.

Each target creates a new inverse kinematic problem, which is solved using the back-bone curve method. Chirikjian minimized the shape changing of the back-bone curve when the curve moved from one frame to the next one. This caused a smooth motion for the manipulator when it followed the trajectory. This method resulted in large errors, which are due to the errors of the continuous method used in solving the inverse kinematic. Kim *et al.*[19] solved the inverse kinematic problem of binary manipulators using a continuous variable optimization method. By using a special weight function, they approximate the results to discrete quantities. They used the method for solving the locus tracing problem. They tried to decrease the manipulator shape changing by changing the upper modules (the modules which are closer to the end). This was accomplished by introducing fewer weights for upper modules. The results of this method are favorable, but they are much dependent on the weight functions. This is why the weight functions should be defined with care.

In this paper, an algorithm is proposed to solve the trajectory tracking problem in an obstacle field. The algorithm utilizes both the two-by-two searching, and the RFC methods[13] for solving inverse kinematics and obstacle avoidance problems, respectively. Furthermore, a collision detection method and a method for decreasing the manipulator movement during the trajectory tracking are introduced. A 2D and 3D case study are introduced and some numerical problems are solved using the proposed method. The results are compared with those of the GA method for validation. Furthermore, a new prototype designed and implemented by the authors is introduced. This prototype is a spatial binary manipulator consisting of three 3-RPS modules. Finally, the motion and the trajectory tracing of this manipulator have been examined by carrying out the relevant experiments.

The rest of this article is organized as follows: In Section 2, DAHMs are first introduced; then a comprehensive description of the problem is presented. In Section 3, after describing the proposed method of solving the motion planning problem, the two-by-two searching method of solving inverse kinematics is briefly explained. This is followed by specifying the proposed method of detecting the collisions. In Section 4, first the case studies are introduced and then, the error is defined clearly. After that, the GA method is briefly explained and numerical results are presented and analyzed. In Section 5, first the prototype is introduced and finally the experimental results are presented and analyzed.

2. Fundamentals

2.1. Introducing DAHMs

The manipulators considered in this article as hyper-redundant manipulators are produced by cascading serial and/or parallel mechanisms on a fixed base. The mechanisms may be the same or different. Each of the stages of the manipulators is called a "module". Each module of a DAHM contains several actuators, which are actuated discretely. The discrete actuators have only a few stable states. For example, a binary-prismatic actuator has only two states: completely retracted (0) and completely extended (1).

Figure 1(a) shows a VGT module.[9] Each VGT module is a planar close loop mechanism containing three binary-prismatic actuators (AC, AD, and BC links in Fig. 1(a)), and two links (AB and CD), which have constant lengths. The links are passively joined together in A, B, C, and D. Each VGT module may be actuated in $2^3 = 8$ different configurations by changing the state of its three binary actuators (Fig. 1(b)). The configuration of a manipulator is created by its modules; thus, the number of configurations of a DAHM may be calculated by multiplying the number of configurations of its modules. For instance, a four-module VGT manipulator has $8^4 = 4096$ configurations. Figure 2 shows a 4-module VGT manipulator in one of its configurations.

It is notable that although the number of configurations of a DAHM is limited, the configuration changing is a continuous process. For example, a VGT module is impossible to be held steady in a configuration different from that shown in Fig. 1(b). If a VGT module, which is initially in configuration 1, is ordered to move to configuration 2, the link BC (Fig. 1(a)) should be extended continuously from the completely retracted state to the completely extended state. However, the expansion process is not controllable. In other words, it is impossible to stop an actuated binary cylinder in a position other than the two ends of its stroke. Thus, the controlling of a DAHM

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

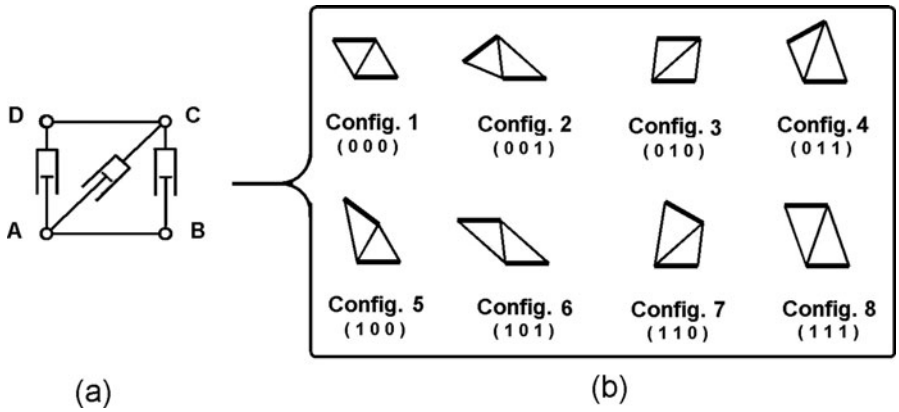

Fig. 1. (a) A VGT module; (b) Configurations of a VGT module.

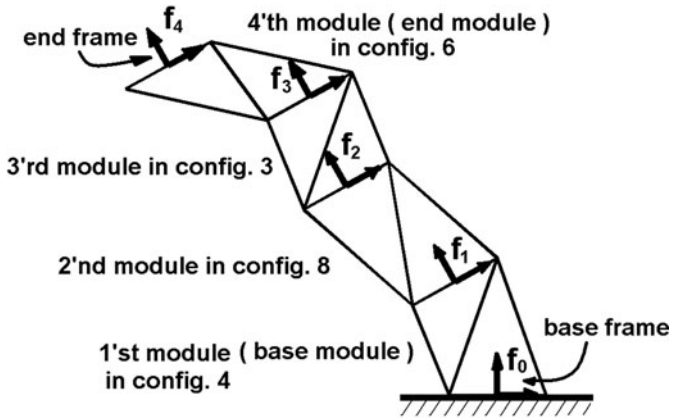

Fig. 2. A four-module VGT manipulator in configuration 4836.

is provided by commanding new configurations, one after the other. Before commanding a new configuration, it is necessary to wait until the manipulator stops.

2.2. Problem definition

In this paper, motion planning is a trajectory tracking in an obstacle field and the trajectory is a prescribed path to be followed by the end-frame. It may contain both position and orientation or just position. The obstacle field is static, and should be clearly defined as one of the problem inputs. The solution of the motion planning problem of a DAHM is a set of configurations, which should be ordered to the manipulator one after another for the purpose of trajectory tracking with the lowest deviation from the path, and without any collision between the manipulator body and the obstacles. It is emphasized that the problem does not include trajectory planning, and the trajectory itself is one of the problem inputs.

3. Solution Method

3.1. Motion planning method

To solve the motion planning problem defined in Section 2.2, first of all, instead of the desired trajectory, some frames should be defined in a way that tracking the frames by the manipulator may produce the desired trajectory. The frames are called “precision frames”. The problem is divided into a number of sub-problems. In each sub-problem, the main objective is to reach the next precision frame. To have a more smooth motion in trajectory tracking, it is suggested that, initially the end-modules and

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

then, if necessary, the first modules start to move. In doing so, each sub-problem is solved through a number of steps. In first step, only two manipulator's end-modules are allowed to change. Later on, in each step, the number of changeable modules will be increased. The amount of the increment is called "ramp". Thus, the number of changeable modules in the ith step will be $2 + (i - 1) \times$ ramp. This continues until the distance between the manipulator and the precision frame reaches an acceptable value. This value is predefined and is called "allowable error". Then the next precision frame should be considered, and the next sub-problem should be solved. It is notable that the number of changeable modules in the first step of solving each sub-problem will be 2. Therefore, each step, in itself is a motion planning problem, which is called "sub-sub-problem". A sub-sub-problem may be defined as follow:

The initial configuration of the manipulator is known as a result of the previous step. By changing the configuration of the changeable modules a new configuration is obtained which gets the end of the manipulator close to the precision frame of the sub-problem, provided that the motion from the initial configuration to the new one does not cause any collision with the obstacles.

The initial configuration in the first sub-sub-problem of the first sub-problem (i.e., the configuration of the manipulator in the beginning of trajectory tracking) should be determined as the input of the problem. In addition, it may be determined by solving the obstacle avoidance problem using the RFC method.[13] In order to solve each sub-sub-problem, a four-step algorithm is proposed, the last three steps to be repeated until the result is obtained. These four steps are: (1) inverse kinematics, (2) finding the first collision, (3) removing the first collision, and (4) reconfiguration. These steps are defined as follows: in the first step, the inverse kinematic problem is solved using the two-by-two searching method so that the precision frame is reached without taking into account the obstacles. The two-by-two searching method is defined in Section 3.2. In the first step, a new configuration is obtained, but it is not clear whether there is collision while moving from the initial configuration to the new one or not. Thus, in the second step modules are examined one by one, starting from the base module, until the first module colliding with obstacles is found. This module is called "first-collision module". The method used to detect the collisions is described in Section 3.3.

In the next step (Step 3), the first collision should be removed by modifying the new configuration. The configuration modification is a searching process from among all the configurations of the module which is located just behind the first-collision module (posterior module). If the searching has no result, i.e., no configuration removing the first collision is found, then the searching process should be continued from among the configurations of the module which is located behind the previous posterior module.

As the new configuration is modified by changing the posterior module's configuration in Step 3, the end of the manipulator goes far from the precision frame. Therefore, in the next step (Step 4) the manipulator configuration is modified to get closer to the precision frames by changing the configuration of the "anterior modules". Anterior modules are the manipulator modules located in front of the first-collision module. If the lower module's configuration is changed, they may collide with obstacles. This may prevent the solution process from leading to any collision-free answer. To modify the configuration, the inverse kinematic problem should be solved without considering obstacles using two-by-two searching method.

By taking the above mentioned steps, the first collision would be removed and the manipulator approaches the precision frame. However, there will be probably a collision in the anterior modules; therefore, the collision detection process should be performed again, i.e. the second step should be repeated. If there is not any collision, the modified configuration is considered as the result of the sub-sub-problem; otherwise, this configuration still needs to be modified; therefore, Steps 3 and 4 should be repeated. The Steps 2, 3, and 4 should be repeated until a collision-free configuration is found. Figure 3 illustrates a flowchart, which describes the proposed method of solving the sub-sub-problems. For greater clarity, four steps of solving the sub-sub-problems are shown in Fig. 4 for a sample problem of a 10-module VGT manipulator.

It should be noted that, in the above mentioned steps, only configuration changing in the changeable modules is allowed. A sub-sub-problem might have no solution, i.e., no collision-free configuration is found, which may get the manipulator closer to the precision frame. In this case, the next sub-sub-problem should be defined only by increasing the number of changeable modules. The algorithm of the presented method for solving the motion planning problem is given in appendix.

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

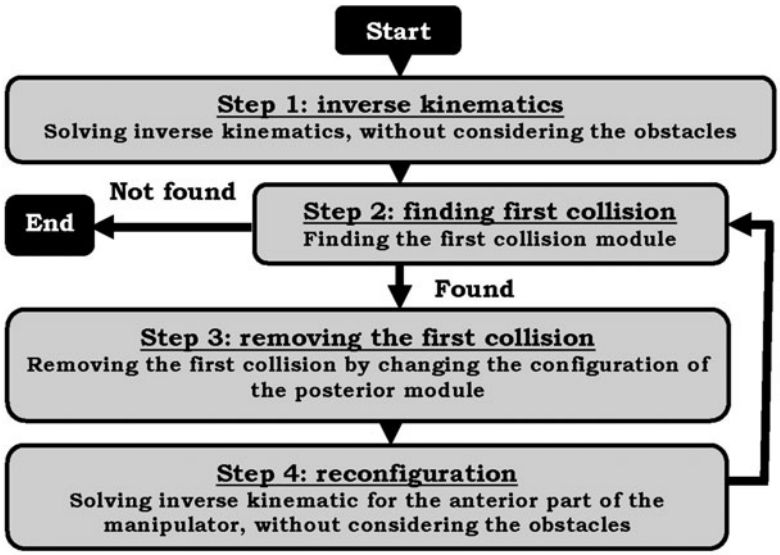

Fig. 3. Flowchart of the proposed method of solving the sub-sub-problems.

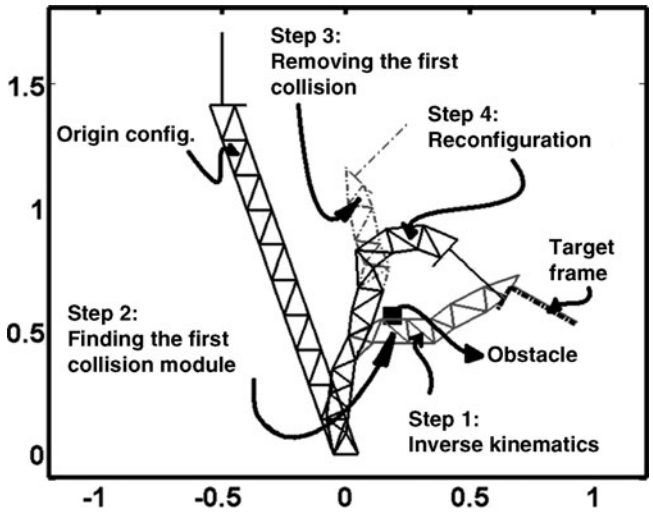

Fig. 4. Configurations of a 10-module VGT manipulator in various steps of solving a sample sub-sub-problem.

3.2. Two-by-two searching method: a method of solving inverse kinematics of DAHMs

As was argued in Section 1, the inverse kinematic problem of DAHMs may not be solved using traditional methods. The method explained in this sub-section is a searching method, which does not necessarily find the exact solution, but only gives a solution with low error. The search is carried out through the configurations, and the error is the distance between end-frame and target-frame, which includes both position and orientation difference. The method of calculating the distance between the two frames is available in ref. [13].

In the two-by-two searching method, first a random configuration is considered as the answer. This answer usually causes a large error. After that, the answer is improved by iterating a searching process. In each iteration process, two modules of the manipulator are randomly selected as "pending modules pair". It is better for the two modules to be non-adjacent. This results in decreasing the solution error. All combinations of configurations of the pending module pair are tested so that the one which may result in the smallest error is identified. The configuration will replace the previous answer. The number of configurations, which should be tested at this step, equals to the multiplication of number of the configurations of the two pending modules. For example, for a VGT manipulator, it is $8 \times 8 = 64$.

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

The number of iterations should be determined in advance. In this paper, it equals twice the number of modules of the manipulator. The error decreases with the increase in the number of iterations, but the rate of the decreasing descends. The readers are suggested to refer to ref. [11] for more information about the two-by-two searching method.

3.3. Collision detection method

It is necessary to find out if a configuration changing is collision-free or not. The configuration change is allowable if it is collision-free. To detect the collision in a configuration changing, it is necessary to specify the area or space which is swept by the manipulator body during the configuration changing, i.e., the swept area. If there is an intersection between swept area and obstacle field, then a collision exists and the configuration changing is not permissible.

To specify the swept area and the obstacle field, a limited area (or space) around the manipulator is first considered as a "case space". The case space should contain the whole area (or space) that may be swept by the manipulator body. For simplicity, the case space of a planar manipulator is defined as a square and it is defined as a cube for a spatial manipulator. This square (cube) should surround a circle (sphere) whose radius equals the maximum length of the manipulator. The center of the circle (sphere) is located in the origin of manipulator base frame.

After defining the square (cube) as the case space, it should be separated by a Cartesian grid. The "obstacle matrix" and the "swept area matrix" may be created by using this grid. The two matrices are both 2D and 3D arrays for planar and spatial manipulators, respectively. Each element of the two matrices corresponds to a cell of the gridded case space. Thus, the obstacle matrix specifies which cells of the gridded case space are occupied by obstacles. Similarly, the swept area matrix specifies which cells of the gridded case space are swept by the manipulator body in a specified configuration changing. If the whole or a part of a cell is occupied by an obstacle, then the corresponding element of the obstacle matrix will be equal to one; otherwise, it will be equal to zero. Similarly, if the whole or a part of a cell is swept by the manipulator body, the corresponding element of the swept area matrix will be equal to one; otherwise, it will be equal to zero. Therefore, if at least one correspondent pair of elements in the two matrices equals to one, there will be a collision.

Now, the question is that how the swept area may be calculated? For simplicity and speed in solving the problem, an approximation method is proposed. Before explaining the method, it is necessary to describe the method of calculating the area (space) which is occupied by a module whose configuration is specified (Question 1). Then, it should be clarified how the swept area of a manipulator module in a specified manipulator configuration changing may be calculated (Question 2). And finally, the swept area of the whole manipulator can be easily computed by calculating the union of all manipulator modules' swept areas.

Answer to Question 1: First, the location of the "module center" should be calculated with respect to module base frame. Thus, location is not dependent on the manipulator configuration, but is dependent on the module configuration. Then, the "circum radius" of the module should be calculated. Circum radius is the radius of a circle (sphere) —"circum circle" for 2D cases and "circum sphere" for 3D cases—which surrounds the entire module body and its center is located at the module center. The circum radius of a manipulator module is not dependent on the manipulator configuration. Figure 5(a) illustrates the surrounding circle of a VGT module in configuration 6. Then, location of the module center should be calculated with respect to the manipulator base frame and the area (volume) occupied by the "circumscribed square (cube)" should also be calculated. The circumscribed square (cube) is a square (cube) which surrounds the circum circle (sphere).Then, cells of the gridded case space, which are completely or partially occupied by the circumscribed square (cube), should be specified. This process is described in more details in Section 2.2.2.1 of ref. [13]. Figure 5(b) illustrates the use of the surrounding circle for indicating the area occupied by the end-module. Grey cells in Fig. 5(b) are related to the end-module's occupied area.

Answer to Question 2: For an exact solution, the manipulator configuration changing should be applied in several steps. To do so, the actuation changing extent of manipulator actuators should be divided into several parts and then applied to actuators step by step. Accordingly, the speed of actuation changing affects the swept area (space) of the modules. Furthermore, it affects the path of manipulator end-frame. In this paper, it is assumed that all actuators move with the same constant speed from one state to another, which is called "constant speed assumption". This assumption is discussed and applied in Section 6 of ref. [8].

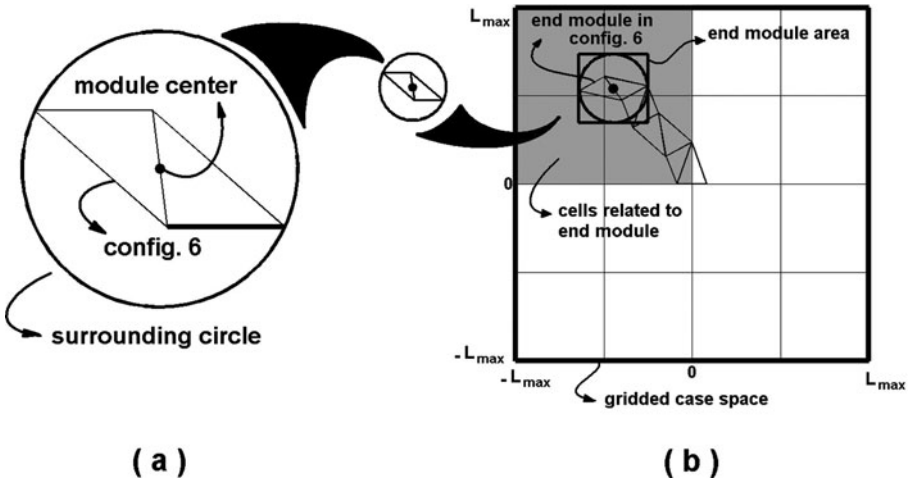

Fig. 5. (a) module center and surrounding circle of a VGT module in configuration 6; (b) finding the end-module area of a four-module VGT manipulator in configuration 4836 using the surrounding circle.

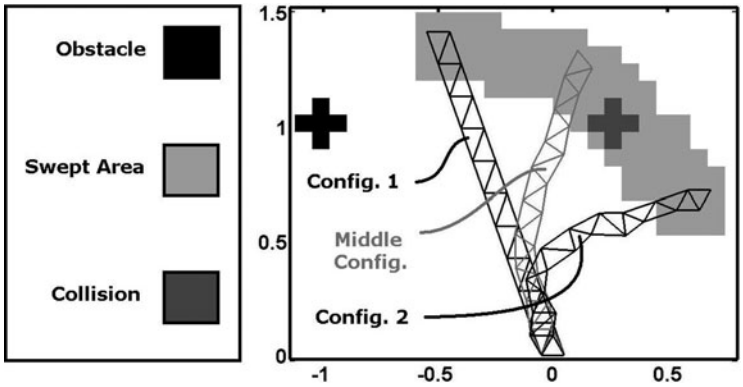

Fig. 6. The swept area of the end-module of a 10-module VGT manipulator in a specified configuration changing (from config. 1 to config. 2). The configuration changing is applied in two steps: from config. 1 to the middle config. and from the middle config. to config. 2.

Accordingly, the union of the areas (spaces) occupied by the module in all steps will be equal to the area (space) swept by the module. To reduce solution time, we can reduce the number of the steps and assume that the module center moves along a straight line, whose two ends are located in the module center at the beginning and end positions of the step. In fact, this assumption is an approximation where as the number of the steps increases, it becomes less approximate. In this paper, configuration changing is carried out in two steps. In order to calculate the area (space) swept by the module in one step, the square (cube) is moved along the line and the area (space) swept by it is calculated. Figure 6 illustrates the areas swept by the end-module of a 10-module VGT manipulator in a specified configuration changing. There are two steps in Fig. 6. Considering the second question, the process continues as follows: The gridded case space's cells that have been occupied completely or partly by the module swept area (space) in a certain configuration changing should be determined to help complete the swept matrix.

4. Numerical Results

A 2D and 3D manipulator are considered as case studies. The case studies are described in the following section. Then, the error is defined in details. For validating the proposed method, the results will be compared with those of the GA method. Therefore, this section proceeds with briefly explaining the GA method. Then, the numerical results will be presented and discussed.

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

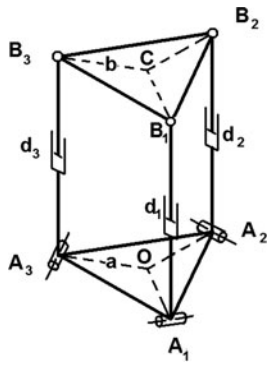

Fig. 7. A 3-RPS module.

4.1. Introducing the case studies

4.1.1. 2D case study. A 20-module VGT manipulator is considered as a 2D case study. The VGT modules—as described in Section 2.1—are shown in Fig. 1. All actuators of the modules (AC, AD, and BC in Fig. 1(a)) are binary-prismatic actuators. The minimum and maximum lengths of the actuators are 1/20 and 1.5/20, respectively. The length of constant links (AB and CD in Fig. 1(a)) is 1/20. Accordingly, the minimum and maximum lengths of the manipulator are $L_{\min} = 1$ and $L_{\max} = 1.5$, respectively. Readers are suggested to refer to Kim *et al.*[19] for further kinematics of this module.

The case space for this manipulator is a square whose side lengths are $2L_{\max} = 3$. The center of this square is located in the manipulator base. The number of division along both axes is $4 \times 20 = 80$. The obstacle field is formed by some cross-shaped obstacles which are regularly distributed in the case space. Each cross occupies five cells of gridded case space. The distance between the central cells of the crosses is 17 cells. Finally, some of the numerical results are related to the obstacle-free space.

4.1.2. 3D case study. A special manipulator with 20 similar 3-RPS modules is considered as 3D case study. A 3-RPS module as shown in Fig. 7 is a special parallel robot with three binary-prismatic actuators in its three legs, i.e., A_1B_1, A_2B_2, and A_3B_3. Base plate ($A_1A_2A_3$) and moving plate ($B_1B_2B_3$) are congruent equilateral triangles. A_1, A_2, and A_3 are passive revolute joints with rotation axes parallel to their opposite sides in triangle $A_1A_2A_3$. B_1, B_2, and B_3 are passive spherical joints. The distance between center and corners of the two triangles is $a = b = 1/20$. The discrete quantities of actuation, which are the length of the legs are $\{1/20$ and $1.5/20\}$. So the minimum and maximum lengths of the manipulator are $L_{\min} = 1$ and $L_{\max} = 1.5$, respectively. Readers are suggested to refer to Kim *et al.*[19] for further kinematics of this module.

The case space of this manipulator is a cube of edge length $2L_{\max} = 3$. The cube center is located in the origin of manipulator base frame. The number of divisions in each of the three directions is $4 \times 20 = 80$. The obstacle field is formed by creating the obstacles of the 2D case study in xz-plane and extending the obstacles in direction of y in the entire cubic case space.

4.2. Error definition

Two types of errors are presented in numerical results, namely, error I and error II. Error I only considers the distance between manipulator end-frame and precision frames. Deviation of the followed path from the desired trajectory is not considered in error I. In each sub-problem, the distance between final configuration and precision frame is calculated. Finally, error I will be equal to the average of all these distances. Two criteria are considered in error II. The first criterion is the same as the one considered in error I. The second criterion is the deviation from the trajectory. This is calculated for each sub-problem and then the average of obtained values is calculated. To calculate the deviation in each sub-problem, a number of frames on corresponding path and between precision frames should be considered. These frames are called "trajectory frames". Afterwards, a number of "traced frames" should be calculated. The traced frame is a frame which defines the position of the manipulator end-frame during trajectory tracking. For calculating some traced frames, it is sufficient to change the actuation amounts during several stages. Finally, to calculate the deviation in each sub-problem,

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

the average of shortest distances between trajectory frames and traced frames should be calculated. Error II is an average of amounts of error I and the average deviation.

4.3. GA method

For validating the proposed methods of solving the motion planning problem, its numerical results are compared with results of the GA method. Similar to the proposed method, motion planning in GA method is divided into several sub-problems based on the precision frames and each sub-problem is divided into some sub-sub-problems using the content of changeable modules. Individuals are the manipulator configurations defined by a bit string containing the states of the actuators. Fitness function is defined for each sub-problem as follows:

$$F = D + W \times C, \tag{1}$$

where F is the cost, D is the distance between end-frame and the relevant precision frame, W is a weighting factor, and C is either zero or one depending on the instance of a collision in configuration changing from the initial configuration to the testing configuration. It equals 1 if there is a collision and 0 when no collision occurs.

4.4. Numerical results

All the calculations related to the presented results were carried out by MATLAB software with an Intel 1.66 GHz processor. To define the desired trajectory and the precision frames for all of the numerical problems, a similar process was performed. First, the initial configuration of the first sub-problem was defined with a code of $88\cdots8$ (a twenty-digit code). This configuration is formed by extending all the actuators completely. This configuration is collision-free for both 2D and 3D case studies. Then, the last precision frame (this frame represents the end of the trajectory) were defined. To do so, at first a random configuration was considered and then checked out to have a special property. If it had the property, then the related end-frame was considered as the last precision frame. Otherwise, it was rejected and another random configuration was checked. The desired property of the selected configuration is that, the configuration changing from the first origin configuration to the tested configuration should be collision-free. This helps to reject the motion planning problems, which do not have any collision-free solutions. The remaining problems are called "real problems".

The desired trajectory is a straight line, which connects the origins of the first precision frame to the last precision frame. Direction of trajectory at each point is defined by a frame whose difference from the first and the last frame of the trajectory is proportional to the distance between the point and two end points of the trajectory. If the number of precision frames is larger than 2, other precision frames are obtained by dividing the distance between the first and the last frame into equal distances. The required formulas for dividing a distance between two frames into equal distances are available in ref. [13].

Results of the proposed method and GA method in solving the motion planning problem are presented in Table I. These are presented for both 2D and 3D case studies, each one in both a cross-shaped obstacle field and an obstacle-free case space. The quantities presented in Table I are average results of 100 random real problems. The problems are common to both methods. For both methods, the number of precision frames is two, the ramp is two, and the number of iterations is twice larger than the number of changeable modules. In GA method, population size is 10, number of generations is 100, elite count (number of elite children transferred to the next population without any changes) is 2, and finally crossover fraction is 0.8.

As seen in Table I, the error of the proposed method in all cases is much less than the GA method. This difference is more intense in 2D case study. As expected, the error in the obstacle-free case space is less than that in the plus shape obstacle field. Furthermore, the error in the 2D case study is less than that of the 3D case study. Probably, this difference is caused by high ability of the shape changing of the selected mechanism for 2D case study as compared with the 3D one. In all cases, computation time of the proposed method is definitely less than that of the GA method.

The algorithms used in both methods are not adapted for the obstacle-free case study. In the proposed method, calculation time for the obstacle-free case space—especially in the 2D case study—is less than that of the cross-shaped obstacle field. However, in the GA method, the difference is not noticeable. Decreasing the computation in case spaces with fewer obstacles is one of the advantages

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

Table I. Results of the motion planning using the proposed method and GA method for the 2D and 3D case studies; each of the 2D and 3D cases are studied in both obstacle field and obstacle-free field.

Dimensions	Case space	Solution method	Error I	CPU-time (sec)
2D	Cross-shaped obstacle field	Proposed	0.00,751	1.016
		GA	0.02,449	11.823
	Obstacle-free	Proposed	0.00,417	0.722
		GA	0.01,539	12.019
3D	Cross-shaped obstacle field	Proposed	0.02,101	2.751
		GA	0.04,427	15.289
	Obstacle-free	Proposed	0.01,199	2.406
		GA	0.04,222	15.488

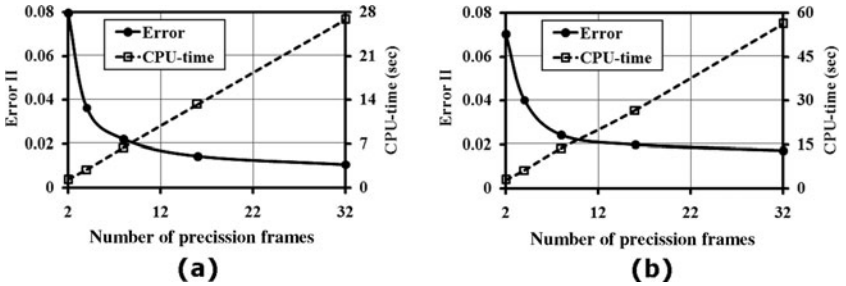

Fig. 8. The effect of the number of precision frames on error and solution time of the proposed method in the 2D (a) and 3D (b) case studies.

of the proposed method. This is because a part of the proposed algorithm runs only if there is a collision; however, in the GA method the solution process is stationary, independent from collisions (the fitness function should be calculated for all individuals).

Figure 8 illustrates the effect of the number of precision frames on error and solution time of the proposed method. Figures 8(a) and (b) are concerned with the 2D and the 3D case studies, respectively. These are related to cross-shaped obstacle field. For both 2D and 3D case studies, 10 random real problems are selected separately. Each problem is solved for various numbers of precision frames and the average results are shown in Fig. 8. The ramp is 2 for all problems. The presented error is error II. According to Fig. 8, for both 2D and 3D case studies the error decreases when number of precision frames is increased; but the rate of this reduction is descending. This shows that deviation from the desired trajectory is reduced by increasing the number of precision frames. Solution time increases linearly when the number of precision frames is increased. Thus, it is not wise to decrease the number of precision frames haphazardly to achieve lower error, considering the growth of the solution time with the number of precision frames. It is notable that in practice, increment of the number of precision frames increases the time that manipulator needs to track the trajectory completely. This is due to the increment in the number of ordering configuration where each configuration change takes a special time.

Figure 9 illustrates the effect of ramp on the error and the solution time of the proposed method. Figures 9(a) and (b) correspond to 2D and 3D case studies, respectively, and both are related to cross-shaped obstacle field. One hundred random real problems are selected separately for both 2D and 3D case studies. Then each problem is solved for various amounts of ramp. Figure 9 shows average of the results. The number of precision frames for all of the cases is two. As seen in Fig. 9, the error in both 2D and 3D case studies is less for cases with fewer ramps. Solution time is higher for both 2D and 3D case studies with fewer ramps.

Figures 10(a) and (b) illustrate the solution of a sample motion planning problem, using the proposed method for 2D and 3D case studies, respectively. The number of precision frames is two and the amount of ramp equals two. Configurations concerned with the first and the last answers are in black while the other configurations are grey. The dashed line is the desired trajectory and the solid curve is the end-frame path, which is drawn based on the constant speed assumption. Errors I and II related to Fig. 10(a) are 0.0032 and 0.0830, respectively. They are 0.0126 and 0.0954 for Fig. 10(b).

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

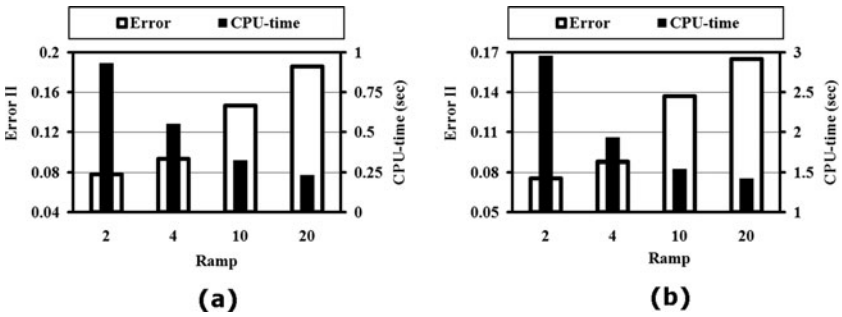

Fig. 9. The effect of the amount of the ramp on error and solution time of the proposed method in the 2D (a) and 3D (b) case studies.

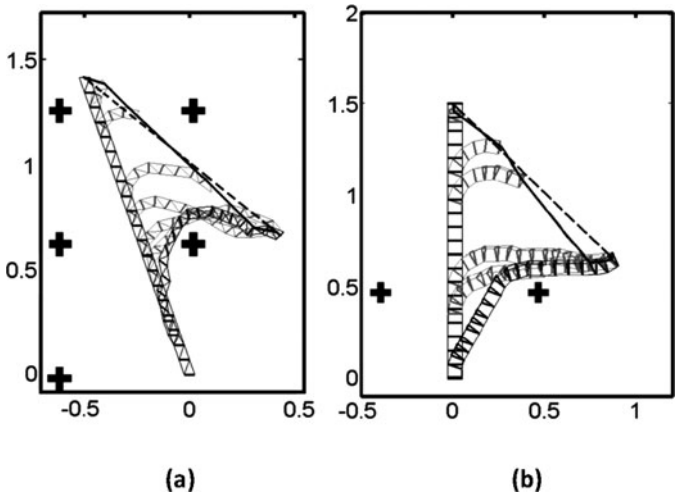

Fig. 10. Solution of a sample motion planning problem using the proposed method for both (a) 2D and (b) 3D case studies; the number of the precision frames is two and so is the amount of ramp; the dashed line is the desired trajectory and the solid curve is the end-frame path, which is drawn based on constant speed assumption.

5. Experimental Result

5.1. Introducing the prototype

A novel spatial binary manipulator is designed and implemented to perceive specialties of the motion and trajectory tracing of DAHMs. Furthermore, we tried to find out the validity of the constant speed assumption by experimental analysis. The manipulator contains three modules stacked on top of each other. Each module is a 3-RPS parallel mechanism, which was described previously in Section 4.1.2. A schematic representation of this mechanism is available in Fig. 7. Upper joint of each leg of the mechanism should be a spherical joint; however, it is possible to replace it by a universal joint without limiting the motion. This is due to the fact that the moving platform may not yaw in this special arrangement of revolute joints' axes. Furthermore, in practice, universal joints available in the market have a wider range of bending angle than those of the spherical joints. Thus, universal joints were used instead of spherical joints in the prototype.

A pneumatic system was used to move the actuators (double acting pneumatic cylinders). Each pneumatic cylinder has two air passages at its top and bottom ends. A one-way flow control valve is mounted on each air passage of the cylinders. The flow control valves are, in fact, adjustable orifices, which are adjusted manually. The air flow rate of each cylinder air passage may be controlled by adjusting these valves independently. Thus, speed of extending and retracting of each cylinder is adjustable through the valves. However, it is notable that the adjusting is performed only once, before ordering the manipulator.

Direction of inlet and outlet air of each cylinder is controlled independently, using a 5/2 solenoid valve for each cylinder. Figure 11 shows the manipulator in a configuration with maximum bend. For

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

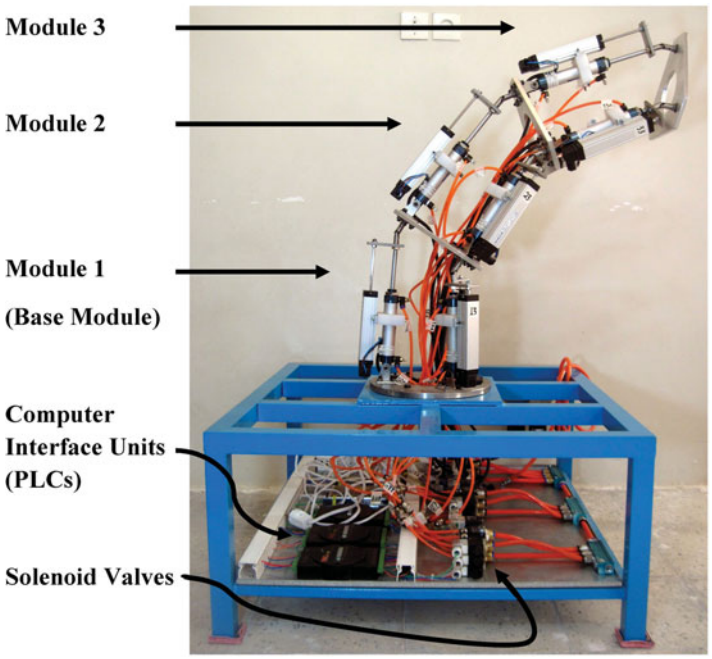

Fig. 11. The prototype in a completely bended configuration (code 333).

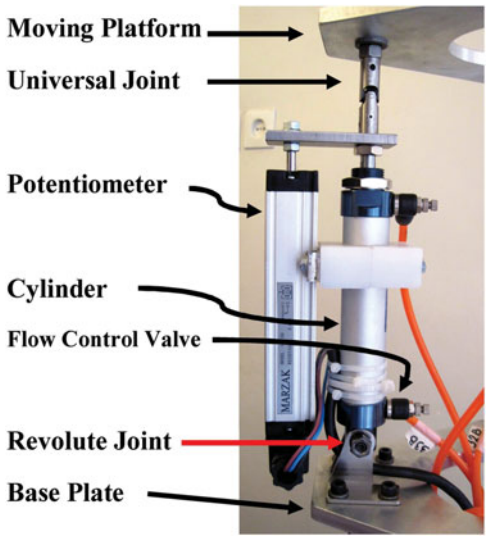

Fig. 12. Details of a leg of a prototype's module.

online measurement of the length of the cylinders, nine piston-type linear potentiometers are used. Each potentiometer is mounted on one cylinder and it is lengthened and shortened by the cylinder so that it can measure the cylinder's length independently (Fig. 12).

The prototype is relatively light. Construction of a similar manipulator with continuous actuation will be definitely more expensive, especially considering the necessity of employing more complicated controlling equipment. Furthermore, controlling the prototype needs no feedback. Therefore, the precision and the repeatability are high. The selected mechanism for the modules makes kinematic analysis easier in comparison with more common mechanisms such as the one proposed by Stewart-Gough. Moving platform of each module may tilt to one side more than 30°; therefore, the manipulator may bend more than 90°. The minimum and maximum lengths of the manipulator are 0.9 and 1.14 m, respectively.

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

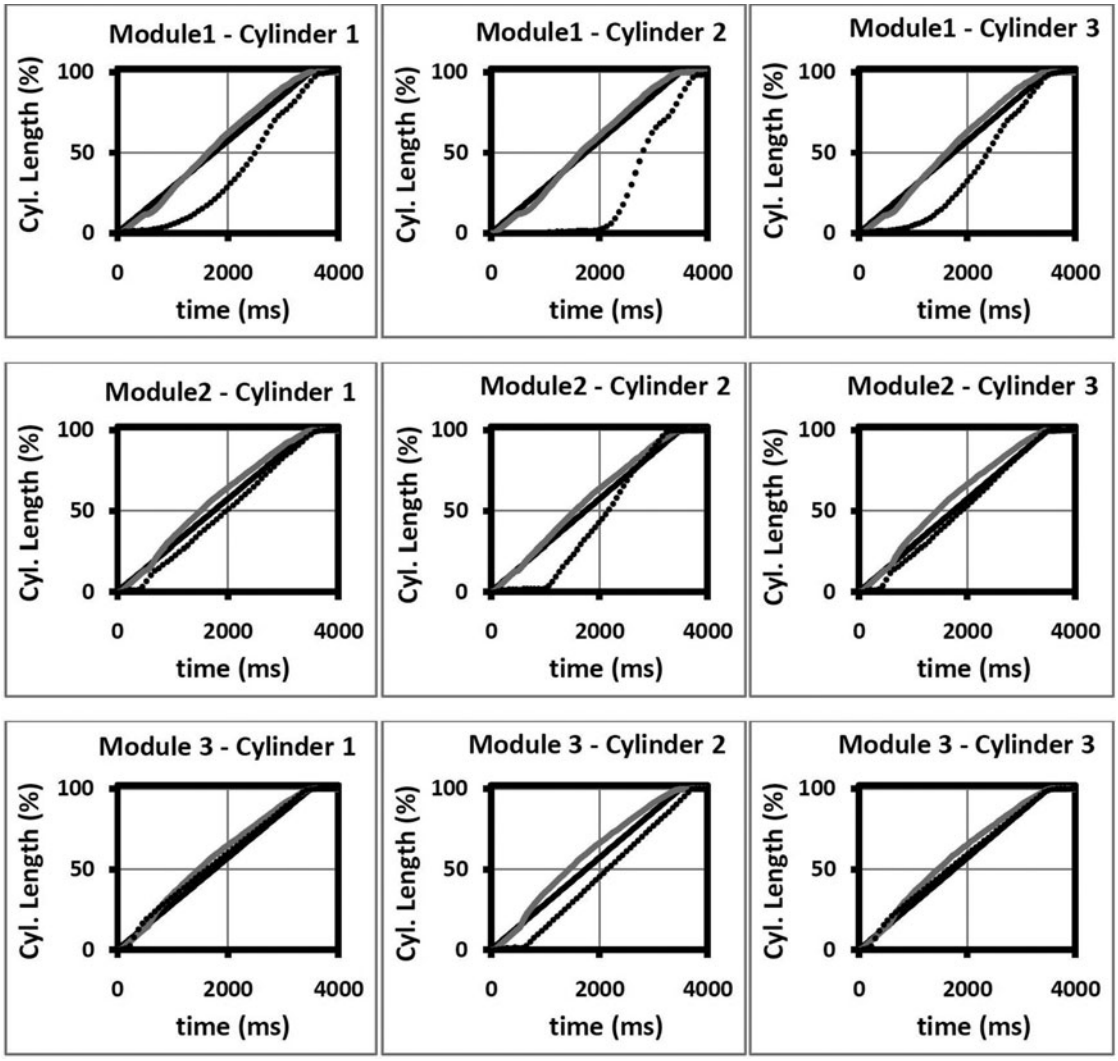

Fig. 13. Changes in the prototype cylinders' length with respect to time: the black solid line is concerned with constant speed assumption, the grey curve is concerned with configuration change from 111 to 888, and the dashed curve is concerned with configuration change from 333 to 666.

5.2. Experimental results

One of the problems with the prototype was that its motions were not smooth. To have smooth motions, the cylinders were required to move with an almost constant speed. Therefore, flow control valves were mounted on two air passages of each cylinder. These valves damp cylinder motions by reducing the dimensions of air passages. Flow control valves were first adjusted while changing the manipulator configuration from state 111 to state 888. In configurations 111 and 888, all of the cylinders are completely retracted and extended, respectively. Figure 13 shows the speed of each of the nine cylinders in the manipulator. The black solid lines in Fig. 13 are related to the constant speed assumption (ideal motion). The grey curves on the other hands, are related to the experimental results for the configuration changing from 111 to 888. The results presented in this figure (grey and dashed curves) are averages of 10 tests of the prototype. Each graph is concerned with a cylinder and the results from output data of the corresponding potentiometer. As shown in Fig. 13, the grey curves are close to the black solid curves. Therefore, constant speed was achieved well in the case of configuration changes from state 111 to state 888 by adjusting flow control valves. However, this necessarily does not mean that the other motions will have constant speed as well. To prove this, a configuration change from state 333 to state 666 was examined without changing the adjustment of flow control valves.

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

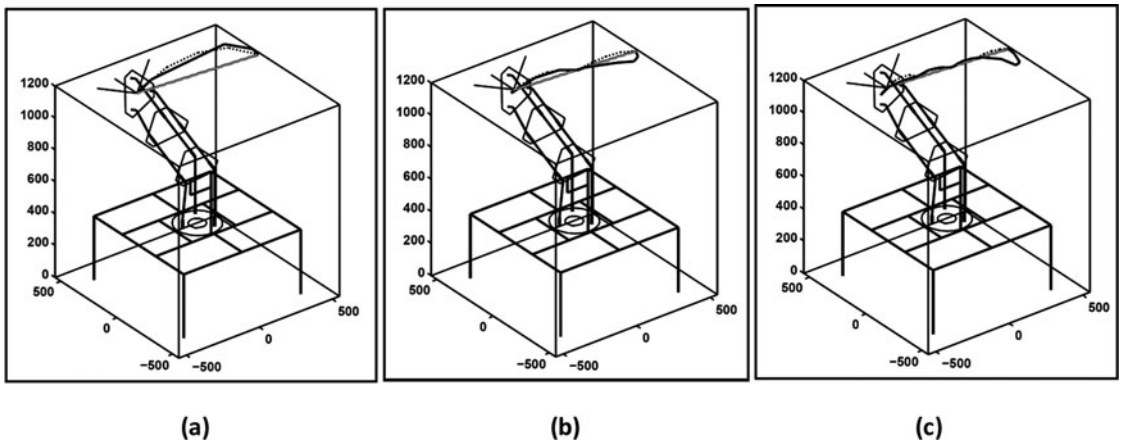

Fig. 14. Simulation of prototype motion planning tests with 2, 3, and 5 precision frames are shown in figures a to c respectively; the grey line in these figures is the desired trajectory, the dashed curve is drawn based on constant speed assumption, and the black solid curve is drawn based on the experimental results.

Configuration 333 is a fully bended manipulator state, where each module's first and third cylinders are completely retracted and each module's second cylinder is completely extended. In configuration 666, all of the actuators have a state that is exactly opposite of the state in configuration 333. The dashed curves in Fig. 13 are concerned with this configuration changing. As it is seen in Fig. 13, the dashed curves have considerable deviation from the corresponding black solid curves (constant speed).

The greatest amount of deviation of the dashed curves from the black solid lines took place in module 1 (the base module). The reason is that weight of two upper modules should be carried by this module. Therefore, the load on this module is higher than other two modules. Furthermore, Fig. 13 shows that dashed curves of cylinder 2 had more deviation from the constant speed compared to the other two cylinders. The reason is that, in configuration changing from 333 to 666, cylinders 1 and 3 have the same motion, but cylinder 2 moves in opposite direction. Thus, cylinders 1 and 3 carry loads together while cylinder 2 should carry the loads on its own. Also, delay of cylinder motion is obvious especially in each module's cylinder 2 (in the dashed curves). The reason is that, after changing the direction of air inlet and outlet by a solenoid valve, the cylinder needs some time until the pressure in the relevant cylinder reaches the level necessary to move the piston. Thus, when the load over a cylinder (cylinder 2 in this case) is higher, the delay will be longer.

In order to have constant speed in cylinders, it is suggested to use more powerful cylinders. This results in the load changes, which take place in various configuration changes to induce much less effects on the cylinder motion speed. Also, these cylinders may be more damped by flow control valves making the speed constant. Furthermore, it is suggested to use hydraulic systems instead of pneumatic ones, because of considerably low compressibility of hydraulic oils in comparison with the air in pneumatic systems. Compressibility of the air results in high deviations from constant speed in pneumatic systems.

To examine the trajectory tracing, an experiment was designed in an obstacle-free field. The trajectory tracing problem was solved using the proposed algorithm; however, the assumption of constant speed in cylinders involves errors. Figures 14 and 15 are presented to examine the errors that are under influence of the number of precision points. The desired trajectory is a line which connects end-frame origin of configuration 383 to the end-frame origin of configuration 686. The motion planning problem is solved for various numbers of precision frames (2, 3, and 5) through the proposed method. In Fig. 14, the desired trajectory is shown in grey. The dashed curves are drawn based on the constant speed assumption, and each of the black solid curves is drawn based on average results of testing the prototype five times.

Figure 15 is presented for numerical comparison of the black solid curves with dashed curves. In this figure, errors I and II are presented for various numbers of precision frames, once based on the constant speed assumption (dashed lines in Fig. 15), and once based on the average of testing the prototype five times (solid lines in Fig. 15). As it is seen in Fig. 15(a), error I increases with a

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

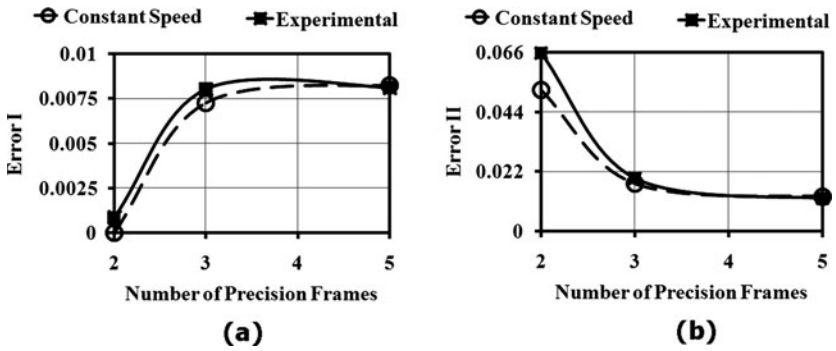

Fig. 15. Error I (a) and Error II (b) related to Fig. 14.

descending rate while the number of the precision frames is increased. As Fig. 15(b) shows, error II decreases with a descending rate while the number of the precision frames is increased. Considering the definition of error II, the final analysis is clearly depicted in Figs. 14(a)–(c).

6. Conclusions

In this paper, a new method is proposed for solving the motion planning problem of DAHMs. In fact, the proposed method is a development of the RFC method,[13] which was originally presented for solving the obstacle avoidance problem. The two-by-two searching method[11] is used to solve inverse kinematic problem, which is a part of solving the motion planning problem. Also, a method was proposed for detecting the module's collision with obstacles during configuration changing of the manipulator. To verify the performance of the presented methods, numerical results were compared with those of the GA method. The findings suggest that the proposed method is definitely more sophisticated than GA in the scales of solution time and errors. Furthermore, a novel prototype, which is a binary manipulator, is introduced for which the motion and the trajectory tracing were examined. Nearly constant speed in the manipulator cylinders was achieved by the help of flow control valves. To alleviate this problem, stronger cylinders are recommended for more damping possibility. Alternatively, other systems like hydraulic systems are recommended.

References

1. G. S. Chirikjian, "A Binary Paradigm for Robotic Manipulators," *Proceedings of the 1994, IEEE International Conference on Robotics and Automation*, San Diego, (1994) pp. 3063–3070.
2. V. A. Sujan, M. D. Lichter and S. Dubowsky, "Lightweight Hyper-redundant Binary Elements for Planetary Exploration Robots," *Proceedings of the 2001, IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, (2001) pp. 1273–1278.
3. I. Ebert-Uphoff, On the Development of Discretely-Actuated Hybrid-Serial-Parallel Manipulators *Ph.D. Dissertation* (Johns Hopkins University, 1997).
4. J. Suthakorn and G. S. Chirikjian, "A new inverse kinematics algorithm for binary manipulators with many actuators," *Adv. Robot.* **15**(2), 225–244 (2001).
5. S. Proulx and J.-S. Plante, "Design and experimental assessment of an elastically averaged binary manipulator using pneumatic air muscles for magnetic resonance imaging guided prostate interventions," *Trans. ASME, J. Mech. Des.* **133**, 1–9 (2011).
6. Q. Chen, Y. Haddab and P. Lutz, "Microfabricated bistable module for digital microrobotics," *J. Micro-Nano Mech.* **6**, 1–12 (2011).
7. L. Petit, C. Prelle, E. Dore and F. Lamarque, "Digital Electromagnetic Actuators Array," *Proceedings of the 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Montréal, Canada, (Jul. 6–9, 2010).
8. G. S. Chirikjian, "Inverse kinematics of binary manipulators using a continuum model," *J. Intell. Robot. Syst.* **19**, 5–22 (1997).
9. I. Ebert-Uphoff and G. S. Chirikjian, "Efficient workspace generation for binary manipulators with many actuators," *J. Robot. Syst.* **12**(6), 383–400 (1995).
10. G. S. Chirikjian and I. Ebert-Uphoff, "Numerical convolution on the Euclidean group with applications to workspace generation," *IEEE Trans. Robot. Automat.* **14**(1), 123–136 (1998).
11. A. Motahari, H. Zohoor and M. H. Korayem, "A new inverse kinematic algorithm for discretely actuated hyper-redundant manipulators," *Latin Am. Appl. Res. J.* **43**, 161–168 (2013).

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

12. A. Motahari, H. Zohoor and M. H. Korayem, "Discrete kinematic synthesis of discretely actuated hyper-redundant manipulators," *Robotica*, **31**(7), 1073–1084 (2013).
13. A. Motahari, H. Zohoor and M. H. Korayem, "A new obstacle avoidance method for discretely actuated hyper-redundant manipulators," *Sci. Iranica*, **19**(4), 1081–1091 (2012).
14. E. Lanteigne and A. Jnifene, "Obstacle avoidance of redundant manipulators using workspace density functions," *Trans. Can. Soc. Mech. Eng.* **33**(4), 597–608 (2009).
15. O. Y. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.* **5**(1), 90–98 (1986).
16. J. Agirrebietia, R. Avile's, I. F. de Bustos and G. Ajuria, "A method for the study of position in highly redundant multibody systems in environments with obstacles," *IEEE Trans. Robot. Autom.* **18**(2), 257–262 (2002).
17. L. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.* **12**(4), 566–580 (1996).
18. G. Sanchez and J. C. Latombe, "A Single-Query Bi-directional Probabilistic Roadmap Planner with Lazy Collision Checking," *Int. Symposium on Robotics Research (ISRR)*, Lorne, Victoria, Australia, November (2001).
19. Y. Y. Kim, G. W. Jang and S. J. Nam, "Inverse kinematics of binary manipulators by using the continuous-variable-base optimization method," *IEEE Trans. Robot.* **22**(1), 33–42 (2006).

Appendix

The algorithm of the proposed method of solving motion planning problem is presented here in detail:

Input $\left(\left\{P_1, P_2, \cdots, P_{N_{pf}}\right\}, \varepsilon, \quad \text{Ramp}\right)$;

% P_i: ith precision frames; N_{pf}: number of precision frames; ε: allowable error

% $Ramp$: amount of increment in the number of changeable modules at each step of the solution

$SolCfg_1 = RFC\,(P_1)$;

% $SolCfg_i$: ith solution configuration; $RFC\,(P_1)$: the solution of obstacle avoidance problem whose target is P_1 using RFC method

$k = 1$;

% k: the index of solution configurations

$Fori = 2 : N_{pf}$

% i: the index of the sub-problem

$Forj = 2 : Ramp : N_{\text{mod}}$

% j: the number of changeable modules; N_{mod}: the number of modules

$CandidCfg = TbT\,(SolCfg_k, P_i, j)$;

% $CandidCfg$: a potential solution; $TbT\,(SolCfg_k, P_i, j)$: the solution of an inverse kinematic problem whose origin configuration is $SolCfg_k$, target is P_i, number of changeable module is j, and the solution method is two-by-two searching

$COL = 1$;

% If there is a collision,COL equals one, if not it equals zero

$WhileCOL = 1$

$\left[COL, m_f\right] = ColDet\,(SolCfg_k, CandidCfg)$;

% m_f: first-collision module; $ColDet\,(A, B)$: an algorithm which finds the first-collision module during configuration changing from A to B. If there is any collision COL equals one and m_f is the order of the first-collision module, if not, COL equals zero and $m_f = NaN$

$IfCOL = 0$

$k = k + 1$;

$SolCfg_k = CandidCfg$;

$Error = Distance\,(SolCfg_k, P_i)$;

% $Distance\,(A, B)$: the distance between frames A and B

$Break$

$Else$

$m_p = m_f$;

% m_p: posterior module

$While\,(COL \neq 0)\ and\ \left(m_p > 1\right)$

$m_p = m_p - 1$;

$Forr = 1 : N_{cfg}\left(m_p\right)$

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press

```
                    % N_cfg(A): the number of configurations of the Ath module in the
manipulator
                    [TestCfg_r] = [CandidCfg];
                    % TestCfg_r: a manipulator configuration tested for removing the first
collision
                    TestCfg_r(m_p) = r;
                    % TestCfg_r(A): testing configuration of the Ath module
                    col_r = ColDet(SolCfg_k, TestCfg_r);
                    dis_r = Distance(P_i, TestCfg_r);
                    E_r = dis_r + W * col_r;
                    % E_r: error related to TestCfg_r, W: weighting factor
                End
                R = Index(Min{E_r : r = 1, 2, ..., N_cfg(m_p)});
                COL = col_R;
                If COL = 0
                    CandidCfg = TestCfg_R;
                End
            End
            CandidCfg = TbT(CandidCfg, P_i, N_mod − m_f);
        COL = 1;
    End
End
If Error ≤ ε
  Break
End
End
End
```

https://doi.org/10.1017/S0263574714002963 Published online by Cambridge University Press