Scientia Iranica B (2012) 19 (4), 1081–1091

Sharif University of Technology

Scientia Iranica

Transactions B: Mechanical Engineering

www.sciencedirect.com

A new obstacle avoidance method for discretely actuated hyper-redundant manipulators

A. Motahari [a,*], H. Zohoor [b], M. Habibnejad Korayem [c]

[a] *Department of Mechanical and Aerospace Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*

[b] *Center of Excellence in Design, Robotics and Automation, Sharif University of Technology, &, The Academy of Sciences, Tehran, Iran*

[c] *Center of Excellence in Experimental Solid Mechanics and Dynamics, Iran University of Science and Technology, Tehran, Iran*

Received 11 September 2011; revised 25 February 2012; accepted 1 May 2012

KEYWORDS

Binary manipulator;
Discrete actuation;
Hyper-redundant manipulator;
Inverse kinematics;
Obstacle avoidance.

Abstract In this paper, a new method is proposed for solving the obstacle avoidance problem of discretely actuated hyper-redundant manipulators. In each step of the solution, the closest collision to the base is removed and then the configuration of the next part of the manipulator is modified without considering the obstacles. This process is performed repeatedly until no collision is found. The Suthakorn method is applied to solve the inverse kinematics problem. Two new ideas are proposed to reduce the errors of this method: the two-by-two searching method, and iterations. To verify the proposed method, some problems are solved numerically for 2D and 3D manipulators, each in two different obstacle fields, and the results are compared with those obtained by the genetic algorithm method.

1. Introduction

When multiple serial or parallel robots are cascaded serially on a fixed base as modules, the degrees of freedom of the mechanism increase. This causes high dexterity and, as a result, the manipulator can avoid collision with obstacles by snake-like motions. In the present paper, these manipulators are called, "hyper-redundant manipulators". The complicated control of these manipulators motivated some investigators [1–3] to propose the use of discrete actuators instead of traditional continuous ones. Contrary to traditional actuators, which act continuously in an interval, discrete actuators have only a few stable states. Moreover, these actuators usually have a simpler structure in comparison with continuous ones and, as a result, have simpler controls. This enables them to enjoy more accuracy and repeatability. Finally, discrete actuators do not need a feedback in their control [2]. Employment of discrete actuators is not restricted to hyper-redundant manipulators. For some examples, readers are referred to [4–6].

When all actuators in a hyper-redundant manipulator are discrete actors, it is called a "Discretely Actuated Hyper-redundant Manipulator" or, in short, "DAHM". The workspace of this kind of manipulator is like a cloud of discrete points. The number of these points grows exponentially by increasing the number of modules. For example, a Variable Geometry Truss (VGT) module [1] is illustrated in Figure 1(a). This is a planar parallel robot. The two links (AB, CD) have constant lengths. The length of the other three links (AD, AC and BC) can be varied by three binary prismatic actuators. A binary actuator is a kind of discrete actuator with only two stable states. A binary prismatic actuator has only two states: completely contracted (0) and completely extended (1). A, B, C and D are passive revolute joints. Existence of the three binary actuators makes the module have $2^3 = 8$ configurations, which are shown in Figure 1(b).

Figure 2 shows a four-module VGT manipulator, which is in configuration 4836. This manipulator has $8^4 = 4096$ configurations, which are attained by combining the configuration states of its modules. Consequently, the manipulator can reach the same number of points in the working plane.

The aim of this paper is to solve the inverse kinematic problem of DAHMs in the presence of obstacles. This problem is called "obstacle avoidance". Since the number of configurations of a DAHM is limited, the first idea that comes to mind for

* Corresponding author.
E-mail addresses: a.motahari@srbiau.ac.ir (A. Motahari), zohoor@sharif.edu (H. Zohoor), hkorayem@iust.ac.ir (M. Habibnejad Korayem).

Peer review under responsibility of Sharif University of Technology.

Production and hosting by Elsevier

1026-3098 © 2012 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.scient.2012.06.017

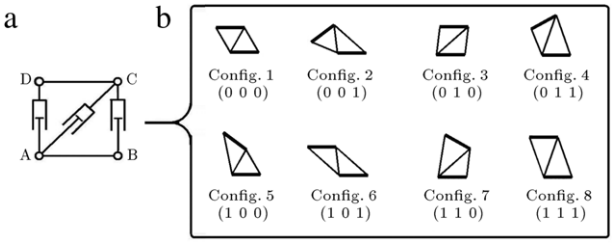

Figure 1: (a) A VGT module; and (b) configurations of a VGT module.

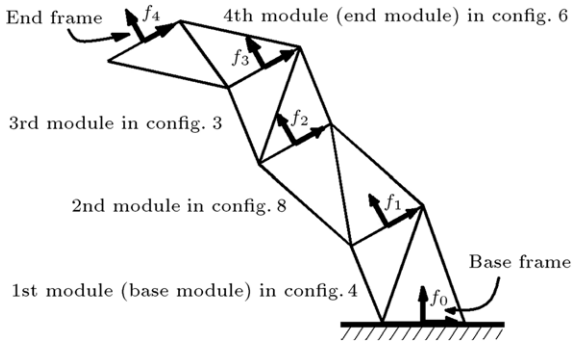

Figure 2: A four-module VGT manipulator in configuration 4836.

solving the problem would be following these steps: checking the collision for all configurations of the manipulator, solving forward kinematics for all collision-free configurations and saving the information offline. Then, the saved information can be used to solve the problems online. This method is not feasible for manipulators that have many modules because of the exponential growth in the number of configurations. The second idea is using methods that are proposed for continuous actuated hyper-redundant manipulators, e.g. the virtual tunnels method proposed by Chirikjian and Burdick [7], or the potential field method used by some investigators [8–10]. When these methods are used for DAHMs, it is necessary to replace the continuous amounts of actuation with the nearest discrete ones. This causes a large error.

Thus, it seems essential to provide an exclusive effective method to solve the obstacle avoidance problem of DAHMs. But, let us first discuss the existent methods of solving the inverse kinematics of DAHMs before reviewing the literature on the obstacle avoidance of DAHMs. Because an inverse kinematic solution is a basic part of solving the obstacle avoidance problem, it is necessary to choose an efficient method to solve the DAHMs inverse kinematic problem. Several methods have been proposed by various researchers to solve this problem. Chirikjian and Ebert-Uphoff [11–13] roughly evaluated workspace density for 2D DAHMs. They used it for solving the DAHMs inverse kinematic problem. In this method, it is necessary to perform an offline evaluation of large amounts of data and storing them. This is especially problematic in 3D cases.

Suthakorn and Chirikjian [14] 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 amounts of offline calculations and stored data were not huge. However, it had the disadvantage of having large errors.

A search for studies on the obstacle avoidance of DAHMs yielded only one result; a study reported by Lanteigne and Jnifene [15]. They solve 2D problems using the workspace density method of the inverse kinematic solution [11–13]. Therefore, their method suffered the same problems as the workspace density method mentioned earlier in this paper.

In this paper, the Suthakorn method [14] is selected to solve the inverse kinematic problem, considering its advantages discussed above. Two heuristic ideas are used to decrease the errors in the Suthakorn method: the two-by-two searching method and iteration.

The proposed method for obstacle avoidance contains four steps: (1) inverse kinematics, (2) finding the first collision, (3) escaping (removing the first collision) and (4) reconfiguration. Steps 2 to 4 are repeated until no collision is found. One essential objective that is considered in the designation of this method is to minimize the amount of collision checking, since collision checking is a time consuming step. Furthermore, the study attempted to design a fast collision checking method which will be explained later. The effectiveness of the proposed method for solving the obstacle avoidance problem of DAHMs is verified by 2D and 3D 20-module manipulators, as case studies, each in two different obstacle fields, and the results are compared with the results of the Genetic Algorithm (GA) method. The numerical results show that the proposed method is considerably faster and more accurate than the GA method. The proposed algorithm can usually solve the problems in time less than a second and the errors are reasonable.

The remainder of this article is organized as follows: Section 2 first presents a comprehensive description of the problem. Then, the step by step solution is explained. After that, the GA method is described. Finally, the error is defined. In Section 3, the solution algorithm is presented in detail. In Section 4, the numerical results are presented, and in Section 5, the paper is concluded.

2. Fundamentals

2.1. Problem statement

The structure and dimensions of the modules, the discrete amounts of actuation, the number of modules, the space occupied by the obstacles and the target frame are known. It is desired to find a manipulator configuration with an end frame as close as possible to the target frame, while collision between the body of the manipulator and the obstacles is not permitted. The distance between the two above-mentioned frames includes both location and orientation differences. This is thoroughly explained in Appendix A.

The modules can be either serial or parallel robots. Also, they can be dissimilar and with an arbitrary number of modules. All actuators are actuated discretely, and the number of stable states of each actuator is arbitrary. The number and form of the obstacles are arbitrary. The problem can be 2D or 3D.

2.2. Solution method

The flowchart of the proposed obstacle avoidance method is shown in Figure 3. In the first step, the inverse kinematic problem is solved for the manipulator without considering the obstacles. In this step, the manipulator configuration is determined in such a way that its end effecter is as close to the target frame as possible. The collision is allowed in this step. Here, the step is called "inverse kinematic".

In the next step, the algorithm checks the existence of a collision between the manipulator body and the obstacles, module by module, starting from the base module. The collision

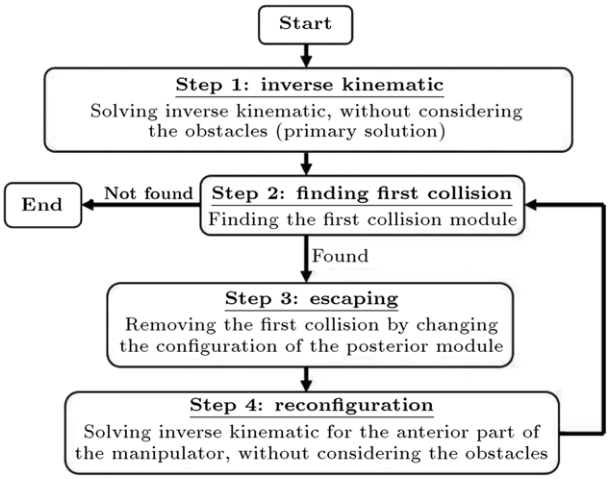

Figure 3: Flowchart of the proposed method of solving the obstacle avoidance problem.

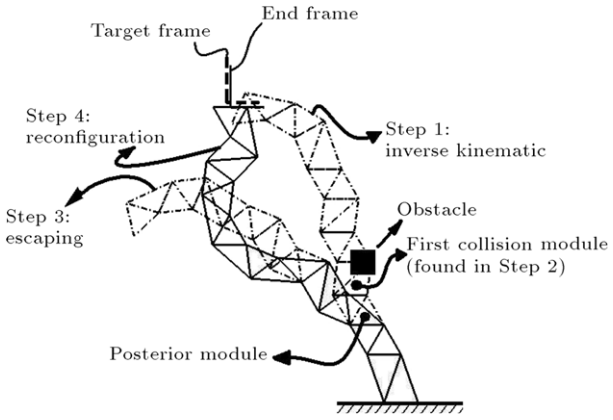

Figure 4: Configurations of a 10-module VGT manipulator in various steps of obstacle avoidance solution by the proposed method.

Table 1: Configurations of the 10-module VGT manipulator in various steps of obstacle avoidance solution in Figure 4.

Steps	Configuration of modules									
	1	2	3	4	5	6	7	8	9	10
1- Inverse kinematic	8	1	1	3	1	3	1	4	1	2
2- Finding first collision										
3- Escaping	8	1	*2*	3	1	3	1	4	1	2
4- Reconfiguration	8	1	2	3	*6*	*5*	*5*	*1*	*3*	*2*

checker continues checking until the first collision is found. The module with the first collision is called the "first collision module". This step of solution is called "finding the first collision". If no collision is found in this step, the manipulator configuration is considered as the answer for the problem, and the solution process is finished, otherwise the solution process is continued to the next step.

In the next step, the first collision should be removed. This can be done by changing the configuration of a module that is located before the first collision module. This module is called "posterior module". At first, the module that is located just before the first collision module is considered a posterior module. But, if the first collision cannot be removed by changing the configuration of this module, the program chooses its previous module as the posterior module. This step is called "escaping". Escaping usually increases the distance between the end frame and the target frame, so it is necessary to improve the manipulator configuration in the next step of the solution.

To avoid duplication, the modification of the configuration is done only on the part of the manipulator that is located after the first collision module. This part of the manipulator is called the "anterior part". This modification is done by solving the inverse kinematic problem of the anterior part. This step is called "reconfiguration". After reconfiguration, it is necessary to find and remove the collisions, so the process of solution is followed by finding the first collision, escaping and reconfiguration again. This loop is repeated until no collision is found. At that point, the configuration of the manipulator is considered as the solution to the problem.

Figure 4 shows the steps of solving an obstacle avoidance problem using the proposed method for a 10-module VGT manipulator. The related configuration changing process in various steps of the solution is presented in Table 1.

Collision checking is a time consuming process. Thus, for decreasing the solution time of the obstacle avoidance problem, it is better to reduce number of times the collision checking process is run. This point is well considered in designation of the proposed method of this article.

In the proposed method, the collision nearest to the base is first removed, because the manipulator is fixed in the base and is free at the other end. On the other hand, as the configurations of the modules located before the first collision module are not changed in the reconfiguration step, after reconfiguration, the collisions, if any, are in the upper modules. Therefore, the iteration of the process will finally lead to no collision.

These steps of the solution are explained, in detail, presently.

2.2.1. Step 1: inverse kinematics

As mentioned earlier, with some modifications, the Suthakorn method [14] is used to solve the inverse kinematic problem in this paper. First, the Suthakorn method is introduced briefly. In the Suthakorn method, at each step of the solution, the configuration of one module is selected among all its configurations. This selection starts from the base module and is continued to the end module. This selection is based on probabilities. It means that the selection criterion is the probability of reaching the target frame. This probability is more when the density of the manipulator workspace in the neighborhood of the target frame is high. In the Suthakorn method it is assumed that the densest position of the workspace is its mean position, and workspace density decreases with the distance of the mean position. Therefore, if the distance between the end frame and the mean frame decreases, then the related workspace density is increased. Evaluating the mean frame of the manipulators can be done easily by the method proposed by Suthakorn and Chirikjian [14]. This method is described briefly in Appendix B.

There are three kinds of module at each step of the solution: Firstly, a module whose appropriate configuration has been selected previously (decided module); secondly, a module whose appropriate configuration is being selected (pending module); and thirdly, a module whose appropriate configuration is unknown (undecided module).

The undecided modules form a sub-workspace, which is a subset of the workspace of the manipulator. This sub-workspace can be translated and rotated by changing the configuration of the pending module. Doing this, the mean frame of the sub-workspace is shifted. A pending module

configuration, which results in the least distance between the sub-workspace mean frame and the target frame, is selected from among all configurations of pending modules as its proper configuration.

At the first step of the inverse kinematic solution by the Suthakorn method, the base module (1st module) is considered a pending module, and the rest of the modules are undecided. At the second step, the 1st module is decided, the 2nd module is pending and the rest of the modules are undecided. These steps are continued until all modules are decided.

The numerical results show that the error of this method is high, but the solution time is very short [14]. The distance between the end frame of the manipulator in the answer configuration and the target frame is considered as error. The length of the manipulator is normalized for convenience of analysis. The minimum length of a normalized manipulator is unity.

To reduce error, it is proposed here to consider two modules as a pending pair of modules at each step of the solution, instead of one module. This method is called the "two-by-two searching method". The numerical results show that if the pending pair modules are non-adjacent, the errors reduce effectively. The pending pair modules can be chosen randomly from among the undecided modules at each step of the solution. The iteration of this process can further reduce errors.

An iteration process is done by choosing a pending pair module randomly from among the decided modules and doing the selection process on them. This method is named the "iteration method". In this paper, the iterations are done after running the two-by-two searching algorithm. However, the numerical results show that even if the iterations are implemented on a manipulator with a random configuration, the error will be reduced effectively.

2.2.2. Step 2: finding the first collision

2.2.2.1. Case space, obstacle matrix and manipulator matrix. The method adopted here to define the obstacles has no limit on the form or number of obstacles. At first, a range of space around the manipulator is defined as the "case space". The case space should include the entire region in which the manipulator body can move. In this article, the case space for the 2D manipulators is defined by a square surrounding a circle with the radius equal to the maximum length of the manipulator (L_{max}), and the center located at the origin of the base frame. The case space of a four-module VGT manipulator is shown in Figure 5. The case space of the 3D manipulator is defined by a cube surrounding a sphere, with the same definition for its radius and its center.

The case space should be separated by a Cartesian grid. After that, the obstacle matrix (A_{obs}) can be defined. This matrix specifies which cells of the case space are occupied by obstacles. Each element of this matrix is related to a corresponding cell of the case space. If all or part of a cell is occupied by an obstacle, the corresponding element of the obstacle matrix will be equal to one, otherwise it will be equal to zero.

Another matrix that should be defined to detect collisions is the manipulator matrix. This matrix specifies which cells of the case space are occupied by the body of the manipulator in its configuration. If all or part of a cell is located within the space occupied by the body of the manipulator, then, the corresponding element of the manipulator matrix will be equal to one, otherwise it will be equal to zero. It should be noted that the manipulator matrix should be defined for any configuration separately. So, when there is a collision in a cell, the corresponding element of both the manipulator matrix and the obstacle matrix is equal to one. In other words, there is a collision if at least one corresponding element of the manipulator matrix and the obstacle matrix equals one simultaneously.

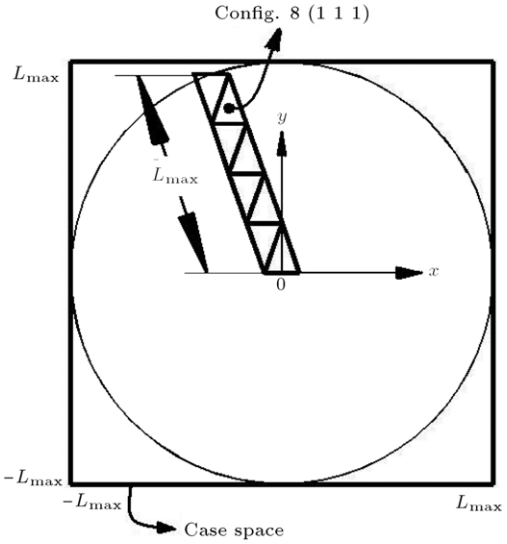

Figure 5: Case space of a four-module VGT manipulator.

The time consuming part of the collision detection process is the evaluation of the space that is occupied by the manipulator body called the "manipulator area". For a faster solution, an approximate method is used for specifying the manipulator area, although it has overestimation.

In the offline part of the algorithm, for each configuration of any one of the dissimilar modules, the location of the center and radius of a circle (sphere for 3D cases) that surrounds the module area is calculated. The conventional location of the module center is defined in the middle of the line that connects the origins of the base frame and the end frame of that module. The radius of the surrounded circle (or sphere) of the module, at an especial configuration, is equal to the maximum of distances between the module center and its corners. This method is designed for modules that are parallel robots. It has more overestimation for serial modules.

After that, in the online part of the algorithm, the location of the center of each module of the manipulator must be calculated in the base frame coordinates (f_0). Then, the area of each manipulator module is specified by a square (a cube in 3D cases) surrounding the corresponding module circle (a sphere in 3D cases). Figure 6(a) illustrates the surrounding circle of a VGT module in configuration 6, which is evaluated offline. Figure 6(b) shows the use of this circle for indicating the end module area, which is in the same configuration.

After that, the elements of the manipulator matrix corresponding to any cell of the case space, all or part of which is occupied by these squares (cubes in 3D cases), are set equal to one. Grey cells in Figure 6(b) are related to the end module area. So, the corresponding elements of the manipulator matrix (A_{man}) are (1, 1), (1, 2), (2, 1) and (2, 2), which are equal to one.

Doing this, the manipulator matrix is completed module by module, starting from the base module and continuing to the end module. Using the surrounded circles (spheres in 3D cases) has the important advantage that the rotation of the module, which is inevitable because of the change in the configuration of the previous modules of the manipulator, has no effect on it.

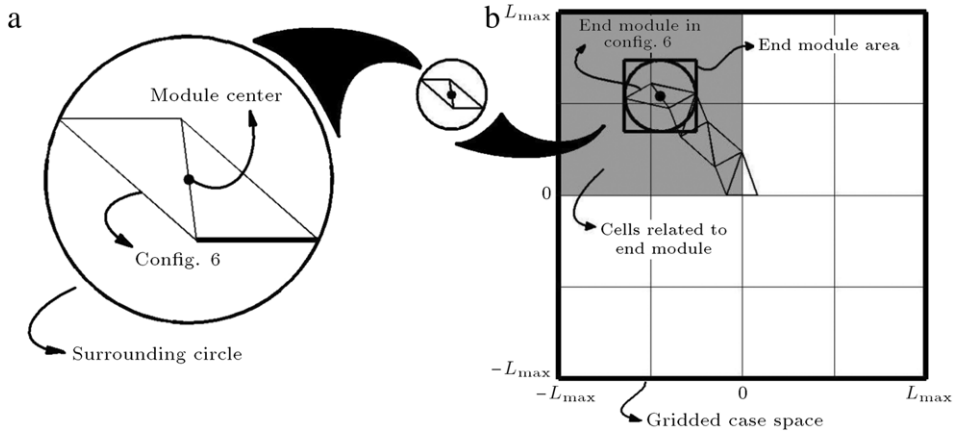

Figure 6: Offline (a) and online (b) parts of finding the end module area of a four-module VGT manipulator in configuration 4836.

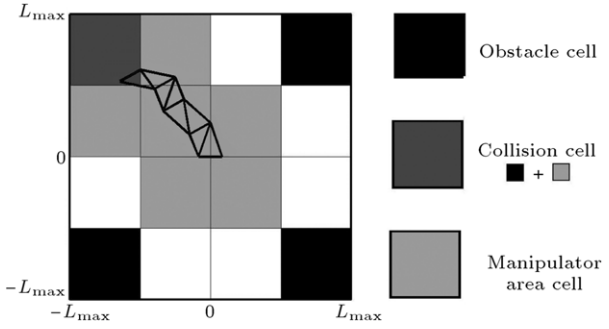

Figure 7: The case space of a four-module VG manipulator with a 4 × 4 grid, and the cells occupied by obstacles and manipulator in configuration 4836.

Without these circles, it is necessary to calculate the module area for any rotation angle, offline. This is not feasible for 3D cases, because there are three rotation angles in 3D space. However, there are some overestimations in the proposed method of defining the manipulator area.

A four-module VGT manipulator in a configuration of 4836, whose case space is a 4 × 4 grid, is shown as an example in Figure 7. The obstacle cells and the manipulator cells are shown in black and grey, respectively. As can be seen in Figure 7, a collision is occurred in cell (1,1). The corresponding manipulator matrix (A_{man}) and obstacle matrix (A_{obs}) are:

$$A_{man} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad A_{obs} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

2.2.2.2. The first collision module. In the step of "finding the first collision", as mentioned earlier, the 1st module colliding with the obstacle, which is named "the first collision module", should be determined. For this purpose, the collision conditions mentioned earlier are checked, module by module separately starting from the base module and continuing until the first collision is detected. The corresponding module is specified as the first collision module.

2.2.3. Step 3: escaping

Escaping means removing the first collision, i.e., escaping is changing the configuration of the manipulator in such a way that the first collision module does not collide with the obstacles. After the first collision module is detected, it is time to determine the posterior module. Initially, the module that is located just before the first collision module is considered the posterior module. But, if it does not work, one module behind is selected as the posterior module.

All configurations of the posterior module are tested. The configuration selection for the posterior module is based on two criteria: first, the first collision module does not collide with obstacles; second, the manipulator end frame is closer to the target frame. The first criterion is more important than the second. To consider both criteria at the same time, an error function is defined:

$$E = D + W.C, \tag{1}$$

where E is the amount of error function, preferably a small quantity. D is the distance between the end frame and the target frame; it is related to the second criteria. C equals one if the first collision modules collide with an obstacle, otherwise it is equal to zero. W is a weighting factor, which is defined based on the importance of the two criteria in comparison to each other.

If changing the configuration of the posterior module cannot remove the first collision, one module behind it is selected as the new posterior module, and the process is repeated. It is possible that the algorithm is forced to take a turn behind the posterior module several times.

2.2.4. Step 4: reconfiguration

In the step of inverse kinematics, the distance between the end frame and the target frame is minimized. After that, in the escaping step, the configuration of one module of the manipulator (posterior module) is changed, so the end frame goes away from the target frame (as can be seen in Figure 4). The end frame can be moved toward the target frame by running the inverse kinematic again. But it is useless if the configuration of the modules that are located before the first collision module is changed, because it can cause a collision in that part of the manipulator, which is a regression in the solving process. Therefore, the inverse kinematic solution is applied only on the anterior part of the manipulator, which contains modules located after the first collision module. Doing this, the end effector goes closer to the target frame. Moreover, the first collision, if any, is transferred to the higher modules.

The numerical results show that it is better to use a two-by-two searching method in the step of the inverse kinematic, and the iteration method in the step of reconfiguration. In the step of finding the first collision, escaping and reconfiguration form a loop (as shown in Figure 3). This loop is repeated until no collision is found.

2.3. Genetic algorithm method

To verify the proposed method of solving the obstacle avoidance problem for DAHMs, the numerical results of the proposed method are compared with the Genetic Algorithm (GA) method. The fitness function is defined as follows:

$$F = D + W \cdot C^*, \tag{2}$$

where F is the cost; D is the distance between the end frame and the target frame; W is the weighting factor; and C^* is the number of cells of the case space in which the collision occurred. Here, not only the first collision, but all collisions should be detected and taken into account.

2.4. Error definition

There are two criteria to examine the accuracy of the obstacle avoidance solution method: firstly, non-existence of collisions; secondly, closeness of end frame and target frame. If the first criteria are not satisfied, or, in other words, if there is a collision (or collisions) between the manipulator body and obstacles, then the answer is unacceptable, so the error is not calculated for it. But, if the answer is collision-free, then the error is defined equal to the distance between the end frame and the target frame. Evaluating the distance between the two frames is described in Appendix A.

For analysis convenience, the manipulator lengths should be normalized. This is done by dividing all lengths by the total length of the manipulator in the configuration where all actuators are actuated minimally (the minimum length of the manipulator).

The target frames for numerical examples of this paper are real targets. This means that each target frame is chosen equal to the end frame of the manipulator in a collision-free configuration. Thus, all problems are guaranteed to have an exact solution. These collision-free configurations can be detected by two methods. In the first method, the configurations are chosen randomly. They are checked by a collision checker algorithm, and any that are collision-free are saved. The end-frames of the saved configurations can be used as the target frame. In the second method, a random problem is solved by GA method. If the answer is collision-free, its end frame can be taken as a real target frame. The first method is not feasible for obstacle-laden workspaces.

3. Solution algorithm

The solution algorithm is divided into two parts: offline and online. In the offline algorithm, the inverse kinematics of each module, in all its configurations, and the space occupied by each module are evaluated and saved, considering its structure and dimensions. After that, in the online algorithm, at first, the problem definition is completed by inputting the target frame and obstacle field. Then, the problem is solved. The solution of the problem (the online algorithm output) is a configuration for the manipulator which should be collision-free and, as far as possible, close to the target frame. In the proposed algorithms, it is assumed that the modules of the manipulator are dissimilar. Moreover, some of the terms which are placed in parenthesis are only applied to 3D cases.

3.1. Offline algorithm

(1) Evaluating and saving g^i_j for $i = 1, 2, \ldots, N_{\text{mod}}$ and $j = 1, 2, \ldots, N^i_{con}$, where g^A_B is a homogeneous transformation matrix that defines the position of the end frame of the Ath module (f_A) in its base frame coordinates (f_{A-1}) when the module is in configuration B. For similar modules, these matrixes are calculated once. N_{mod} is the number of modules. N^i_{con} is the number of configurations of the ith module of the manipulator. These matrixes are formed by solving the forward kinematics.

(2) Evaluating g^i_{mean} for $i = 1, 2, \ldots, N_{\text{mod}}$, where g^i_{mean} is the transformation matrix related to the mean frame of the workspace of the ith module (mean of one module workspace). Evaluation of the mean matrix is done using the Suthakorn method [14]. This method is explained briefly in Appendix B.

(3) Considering all modules in their mean states as follows:

$$g^{(i)} = g^i_{mean} \quad \text{for } i = 1, 2, \ldots, N_{\text{mod}}.$$

(4) Evaluating $C^i_j$and R^i_j for $i = 1, 2, \ldots, N_{\text{mod}}$ and $j = 1, 2, \ldots, N^i_{con}$, where C^A_B is a vector of the coordinates of the Ath module center in its configuration B related to its base frame (f_{A-1}). R^A_B is the amount of radius of the Ath module area circle in its configuration, B. C^i_j can be evaluated from the following equation:

$$C^i_j = g^i_j \circ [0,\ 0\,(,\ 0\,),\ 0.5]^T,$$

$$\begin{cases} C^i_j(3,1) = 1 & \text{for 2D cases} \\ C^i_j(4,1) = 1 & \text{for 3D cases.} \end{cases}$$

If $Corner^i_{j,k}$ for $k = 1, 2, \ldots, N^i_{cor}$ is the coordinate vector of the kth corner of the ith module in configuration j, with respect to its base frame coordinates (f_{i-1}), and N^i_{cor} is the number of corners of the ith module, then, R^i_j can be evaluated by the following equation:

$$R^i_j = \max_k(\|Corner^i_{j,k} - C^i_j\|_2)$$

where $\|\cdot\|_2$ is the vector norm.

3.2. Online algorithm

A- *Step* 1. inverse kinematic (two-by-two searching method).

A.1- Inputting the target frame in the form of the transformation matrix (g_{tar}).

A.2- Making the order list (L). It can be done by setting the natural numbers 1 to N_{mod} in a list in a random order and sorting each pair of numbers (L_i for $i = 1, 2, \ldots, N_{\text{mod}}/2$) from small to big:

$$L = \{L_1, L_2, \ldots, L_{N_{\text{mod}}/2}\}$$

$$L_i = (L_{i1}, L_{i2})$$

$$\textit{Conditions} \begin{cases} L_{ij} \in \mathbb{N} \\ 1 = L_{ij} = N_{\text{mod}} \\ L_{i1} = L_{i2}. \end{cases}$$

If the number of modules (N_{mod}) is an odd number, one of the numbers can be repeated twice in the list. Pending pair modules at each step of the inverse kinematic solution are determined from the order list. For example, L_{11}th and L_{12}th modules are pending pair modules at the first step of the solution.

A.3- $i = 1$; i is the order of the pending pair modules.

A.4- Choosing the proper configuration for pending pair modules, which are the L_{i1}th and the L_{i2}th modules of the manipulator. This is done by following the sub-steps.

A.4.1- Evaluating $G^i_{j,k}$ for $j = 1, 2, \ldots, N^{L_{i1}}_{con}$ and $k = 1, 2, \ldots, N^{L_{i2}}_{con}$ as follows:

$$g^{(L_{i1})} = g^{L_{i1}}_j$$
$$g^{(L_{i2})} = g^{L_{i2}}_k$$
$$G^i_{j,k} = g^{(1)} \circ g^{(2)} \circ \cdots \circ g^{(N_{\mathrm{mod}})}$$

where $g^{(A)}$ is the transformation matrix of the Ath module and $G^A_{B,C}$ is the transformation matrix of the manipulator in which the first pending module (L_{A1}) is in configuration B and the second pending module (L_{A2}) is in configuration C.

A.4.2- Evaluating $D\ (G^i_{j,k},\ g_{tar})$ for $j = 1, 2, \ldots, N^{L_{i1}}_{con}$ and $k = 1, 2, \ldots, N^{L_{i2}}_{con}$ where $D\ (A,\ B)$ is the distance between the two frames related to A and B.

A.4.3- Finding the minimum value of D. If the values of j and k corresponding to the minimum value of D are represented by J and K, respectively, then:

$$g^{(L_{i1})} = g^{L_{i1}}_J \quad \text{and} \quad Config\ (L_{i1}) = J$$
$$g^{(L_{i2})} = g^{L_{i2}}_K \quad \text{and} \quad Config\ (L_{i2}) = K.$$

At this stage, J and K are chosen as proper configurations for the pending pair modules, L_{i1} and L_{i2}, respectively. Henceforth, these two modules are defined modules. *Config* is a vector of size $N_{\mathrm{mod}} \times 1$, which indicates the chosen configuration of each manipulator module.

A.5- if $i < N_{\mathrm{mod}}/2$, then $i = i + 1$ and go to A.4.

B- *Step* 2. finding the first collision.

B.1- $COL = 0,\ i = 0$.

B.2- if $COL = 0$ and $i \neq N_{\mathrm{mod}}$, then do the following sub-steps:

B.2.1- $i = i + 1$.

B.2.2- Evaluating Cen_i from the following equation:

$$Cen_i = g^{(1)} \circ g^{(2)} \circ \cdots \circ g^{(i-1)} \circ C^i_{Config\ (i)}$$

where Cen_A is the coordinate vector of the center point of the Ath module, with respect to the base frame of the manipulator (f_0).

B.2.3- Evaluating the vectors $X^i_{\min}$ and X^i_{max} as follows:

$$X^i_{\min} = Cen_i - R^i_{Config\ (i)} \cdot V$$
$$X^i_{\max} = Cen_i + R^i_{Config\ (i)} \cdot V$$

where $X^i_{\min}$ and X^i_{max} are two vectors which represent the ranges of the ith module area. They can be illustrated in the following form:

$$X^i_{\min} = [x^i_{\min},\ y^i_{\min},\ (z^i_{\min}),\ 1]^T$$
$$X^i_{max} = [x^i_{max},\ y^i_{max},\ (z^i_{max}),\ 1]^T$$

where x, y (and z) are the Cartesian coordinates, with respect to the base frame of the manipulator. Moreover, V is:

$$V = [1,\ 1\ (,\ 1\ ),\ 0]^T.$$

B.2.4- Evaluate the ranges of rows and columns (and pages) of the manipulator matrix corresponding to the ith module area. This can be done as follows:

$$N^i_{\alpha,\beta} = ceil(\alpha^i_\beta/\Delta\alpha) + N_{dis,\alpha}$$

where α can be replaced by x or y (or z) and β can be replaced by *min* or *max*. For example, $N^1_{x,\min}$ is the first column number of the manipulator matrix that is related to the first module area. $ceil(A)$ is an operator that rounds A toward infinity. Δx and Δy (and Δz) illustrate the cell. $N_{dis,\alpha}$ indicates half the number of divisions of the case space along the α axis. This can be evaluated as follows:

$$N_{dis,\alpha} = \frac{L_{\max}}{\Delta\alpha}, \quad \alpha \equiv x \text{ or } y \text{ (or } z)$$

where $L_{\max}$ is the maximum length of the manipulator.

B.2.5- Input obstacle matrix (A_{obs}). Its size should be $2N_{dis,y} \times 2N_{dis,x} \times (2N_{dis,z})$.

B.2.6- for $row = N^i_{y,\min},\ N^i_{y,\min} + 1, \ldots, N^i_{y,\max}$ and $column = N^i_{x,\min},\ N^i_{x,\min} + 1, \ldots, N^i_{x,\max}$ (and $e = N^i_{z,\min},\ N^i_{z,\min} + 1, \ldots, N^i_{z,\max}$), if $A_{obs}(row,\ column, page) = 1$, then $COL = 1$, otherwise $COL = 0$.

B.2.7- Go to B.2.

B-3- if $\neq 0$, then $m_{col} = i$, Otherwise, the solution for the problem is the vector *Config*, which indicates the chosen configuration of each manipulator module, and breaks the algorithm (end). m_{col} indicates the first collision module.

C- *Step* 3. escaping.

C.1- $m_{pos} = m_{col} - 1$, where m_{pos} indicates the posterior module.

C.2- The steps B.2.2 to B.2.6 are implemented for the m_{col}th module (this means that i should be replaced with m_{col} in those steps) in all configurations of the m_{pos}th module ($g^{(m_{pos})} = g^{m_{pos}}_j,\ j = 1, 2, \ldots, N^{m_{pos}}_{con}$). At step B.2.6, *COL* is replaced with COL^*_j. If the first collision module (the m_{col}th module) does not collide with any obstacles when the posterior module (the m_{pos}th module) is in configuration j, then, the COL^*_j value is zero; but, if there is a collision, the COL^*_j value is one.

C.3- Evaluating $D(G^{m_{pos}}_j, g_{tar})$ for $j = 1, 2, \ldots, N^{m_{pos}}_{con}$, where $D(A, B)$ is the distance between the two frames related to transformation matrixes A and B. Also, $G^{m_{pos}}_j$ is the transformation matrix of the manipulator, when the posterior module is in configuration j. This matrix can be evaluated as follows:

$$g^{(m_{pos})} = g^{m_{pos}}_j$$
$$G^{m_{pos}}_j = g^{(1)} \circ g^{(2)} \circ \cdots \circ g^{(N_{\mathrm{mod}})}.$$

C.4- Evaluating E_j for $j = 1, 2, \ldots, N^{m_{pos}}_{con}$ as follows:

$$E_j = D(G^{m_{pos}}_j, g_{tar}) + W \cdot COL^*_j$$

where E_j is a quantity similar to the error, which is preferred to be less. Also, j indicates the configuration of the m_{pos}th module. W is a weighting factor. The value of $W = 0.5$ is used in this paper.

C.5- Find the minimum value among all E_j that have been evaluated in the previous step (C.4). If J is the value of j corresponding to the minimum value of E_j, then J is chosen as the proper configuration for the posterior module; $Config\ (m_{pos}) = J$.

D- *Step* 4. Reconfiguration.

D.1- Inputting the number of iterations (N_{itr}).

D.2- $itr = 1$, which indicates the order of iteration.

D.3- Selecting two modules (m'_1 and m'_2) randomly as the pending pair modules among all modules of the anterior part of the manipulator;

$$m'_1, m'_2 \in \mathbb{N}$$
$$m_{col} + 1 \leq m'_1, \qquad m'_2 \leq N_{\mathrm{mod}}$$
$$m'_1 < m'_2.$$

D.4- Running step A.4, where L_{i1} and L_{i2} are replaced with m'_1 and m'_2, respectively.

D.5- If $itr < N_{itr}$, then $itr = itr + 1$ and return to D.3.

D.6- $m'_{col} = m_{col}$.

D.7- Replace step C.1 with the following statement:
"If $m_{col} = m'_{col}$, then $m_{pos} = m_{pos} - 1$; else $m_{pos} = m_{col} - 1$".

D.8- Return to step B.

4. Numerical results

A 2D manipulator and a 3D manipulator, each in two different obstacle fields are considered as case studies, which are described completely in the following sections. Afterwards, the numerical results for these case studies are presented and analyzed. All calculations are done by MATLAB software with an Intel 1.66 GHz processor.

4.1. Introducing the case studies

4.1.1. 2D case study

- A 20-module manipulator, with similar VGT modules, is considered as the 2D case study. This manipulator is called the "VGT manipulator". A VGT module introduced earlier in this paper is illustrated in Figure 1(a). Lengths of constant links (AB and CD in Figure 1(a)) are 1/20. As mentioned earlier, all actuators of this module are binary. The discrete amounts of actuation (which are the length of the links, AD, AC and BC) are $\{1/20, 1.5/20\}$. Thus, the minimum and maximum lengths of the manipulator are $L_{\min} = 1$ and $L_{\max} = 1.5$, respectively. Readers are referred to Kim et al. [16] for the forward kinematics of this module.

The case space of this manipulator is a square, whose side lengths are $2L_{\max} = 3$. The number of divisions along both axes is $4N_{\mathrm{mod}} = 80$. Therefore, the dimensions of the cells would be $\Delta x = \Delta y = 3/80$. For a 2D case, two different obstacle fields are considered:

(1) A plus shape obstacle field (P-field) is formed by some plus shape obstacles that are regularly distributed in the case space. Each plus shape obstacle occupies five adjacent cells. The distance between the central cells of the pluses is 10 cells, in both vertical and horizontal directions. This obstacle field is presented as an easy to pass field.

(2) A square shape obstacle field (S-field) is like a square shaped fence, which surrounds the manipulator base (central point of the square). The only way for the manipulator to cross this fence out is through one of the four openings in the corners of the square. The distance between the fence walls and the central point (manipulator base) is 12 cells. The thickness of the fence walls is three cells. Each of the openings is created by removing the obstacle cells that are located inside a square with a side length of five cells placed in one of the corners of the square fence. This obstacle field is presented as a difficult to pass field.

4.1.2. 3D case study

A 20-module manipulator, with similar 3-RPS modules, is considered as the 3D case study. This manipulator is called, in short, the "3-RPS manipulator". A 3-RPS module, as shown in Figure 8, is a spatial parallel robot with three binary prismatic actuators in its three legs, A_1B_1, A_2B_2 and A_3B_3. The base plate ($A_1A_2A_3$) and the moving plate ($B_1B_2B_3$) are congruent equilateral triangles. A_1, A_2 and A_3 are passive revolute joints

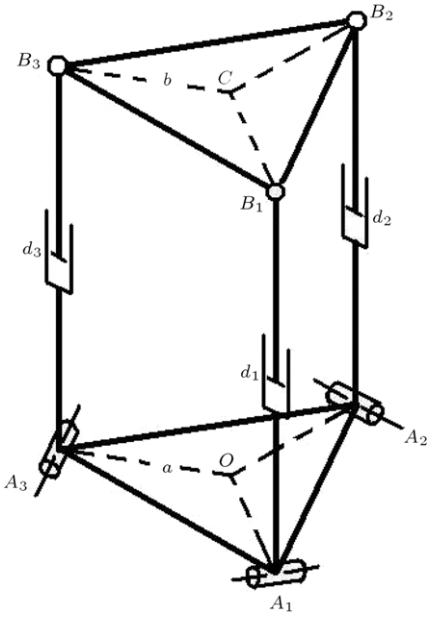

Figure 8: A 3-RPS module.

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 distances between the center and corners of the two triangles are $a = b = 1/20$. The discrete amounts of actuation, which are the length of the legs, are $\{1/20,\ 1.5/20\}$. So, the minimum and maximum lengths of the manipulator are $L_{\min} = 1$ and $L_{\max} = 1.5$, respectively. Readers are referred to Kim et al. [16] for the forward kinematic of this module.

The case space of this manipulator is a cube of edge length $2L_{\max} = 3$. Its central point is located in the origin of the manipulator base frame. This frame is fixed in the central point of the 1st module base plate. Its x axis passes through A_1 and its z axis is perpendicular to the base plate of the 1st module. The number of divisions in each of the three directions is $4N_{\mathrm{mod}} = 80$. Thus, the size of the manipulator matrix and the obstacle matrix is $80 \times 80 \times 80$. Therefore, the edge lengths of the cubic cells would be $\Delta x = \Delta y = \Delta z = 3/80$.

For the 3D case, two different obstacle fields are considered: (1) Plus shape obstacle field (P-field), and (2) Square shape obstacle field (S-field). These two are formed by creating the corresponding 2D obstacle fields (2D P-field and S-field) in the yz-plane of the base frame coordinates of the manipulator, and by extending the obstacles in the direction of x in the whole of the cubic case space.

The P-field and S-field are presented as easy and difficult to pass obstacle fields in the 3D case, respectively.

4.2. Numerical results

Figure 9 compares the results of the two proposed methods of inverse kinematics (two-by-two searching method and iteration method) with the Suthakorn [14] and GA methods, for 2D (Figure 9(a)) and 3D (Figure 9(b)) case studies. The target frames are obtained by solving the forward kinematic for some random configurations of the manipulator. It guarantees the existence of an exact solution for all sampled problems. Every value presented in Figure 9(a) and (b) is an average of 100 values over 100 random sample problems. A set of real targets are used commonly for all four methods. The number of iterations in the

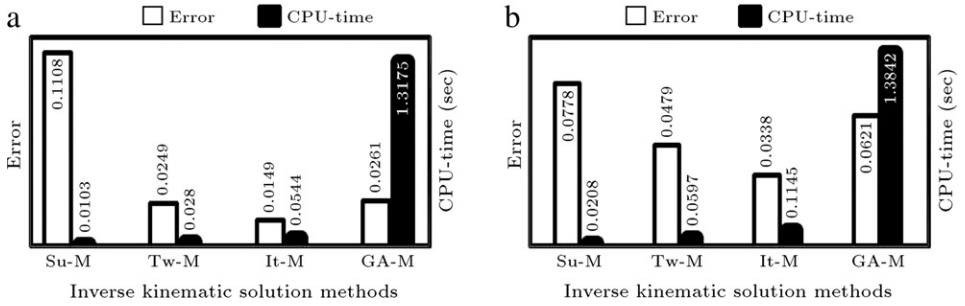

Figure 9: (a) Error and CPU time of inverse kinematic solution of the 2D case study using four methods: Suthakorn method (Su-M), two-by-two searching method (Tw-M), iteration method (It-M) and GA method (GA-M). (b) Same results for 3D case study.

Table 2: The results of the obstacle avoidance solution using the proposed method of this paper and GA method for the 2D and 3D case studies, each in two different obstacle fields.

Case study	Obstacle field	Solution method	Offline CPU-time (s)	Online CPU-time (s)	Total CPU-time (s)	Error	Unacceptable answers
2D	*P*-field	Proposed	0.0034	0.0950	0.0984	0.0537	0
		GA	0.0036	2.5970	2.6006	0.1217	0
	S-field	Proposed	0.0069	0.2039	0.2108	0.0632	0
		GA	0.0077	2.6802	2.6879	0.2464	0
3D	*P*-field	Proposed	0.1717	0.2109	0.3827	0.0603	0
		GA	0.1706	7.3702	7.5408	0.3725	1
	S-field	Proposed	0.4375	0.8287	1.2662	0.1321	0
		GA	0.4214	7.4037	7.8252	0.6579	3

iteration method is 10. In the GA method, the population size is 20; the number of generations is 100; the elite count (number of elite children transferred to the next population without any changes) is 2, and the crossover fraction is 0.8. The fitness value of the GA method is the distance between the end frame and the target frame.

According to the presented results in Figure 9(a) and (b), although the solution time of the Suthakorn method is less than the other methods, its errors are high. The GA method has a long solution time. The two-by-two searching method and the iteration method are more effective than the other two methods.

Table 2 compares the proposed method with the GA method in their efficiency of obstacle avoidance. All sampled problems are solved by two methods: the proposed method of this article and the GA method. Every value in this table is an average of 100 values over 100 random samples. A set of real targets are used commonly for the two methods. The number of iterations in the reconfiguration step of the proposed method is 10.

In the GA method, the population size is 20; the number of generations is 100; the elite count is 2; the crossover fraction is 0.8, and the weighting factor is $W = 0.5$ (the same value is used in the proposed method). As mentioned earlier, unacceptable answers are those that have collisions with obstacles. The number of unacceptable answers is evaluated among the answers of 100 random sample problems.

The results presented in Table 2 show that the proposed method in this paper, according to solution time, errors and the number of unacceptable answers, is significantly more efficient than the GA method. This superiority is observable in both 2D and 3D case studies and in both obstacle fields wherein one of them is an easy to pass sample and another is a difficult to pass sample.

Figure 10 illustrates the effect of the number of iterations (which is used in the reconfiguration step) on errors and the solution time of the proposed method for 2D (Figure 10(a)) and

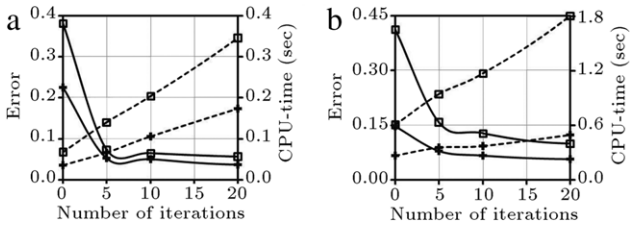

Figure 10: (a) The effect of the number of iterations on error (shown by solid line) and CPU-time (shown by dashed line) of the proposed method in two different obstacle fields: *P*-field (shown by plus sign markers) and *S*-field (shown by square sign markers) for the 2D case study. (b) Same results for 3D case study.

3D (Figure 10(b)) case studies. The results of the two obstacle fields (*P*-field and *S*-field) are illustrated by two curves. Every value in this figure is an average of 100 values over 100 random samples. All used targets are real.

As Figure 10 shows, errors usually reduce when the number of iterations is increased, but the rate of this reduction is descending. So, error curves become almost horizontal after some iteration. On the other hand, the CPU-time ascends almost linearly with the number of iterations. Therefore, excessive increasing of the number of iterations is not reasonable. A comparison of the curves of the two obstacle fields shows that the error and CPU-time of problems with the *S*-field are more than in the other field. It shows that with a more difficult to pass obstacle field, the CPU-time and errors increase.

Figures 11 and 12 show five different configurations of the case study (20-module VGT manipulator in Figure 11, and 20-module 3-RPS manipulator in Figure 12). Each configuration is the answer to a random real problem. The end frames and target frames are shown by solid lines and dashed lines, respectively. Figs. 11(a) and 12(a) are related to the *P*-field; and Figs. 11(b) and 12(b) correspond to *S*-field. Figure 12(a) and (b) show the projection of the 3D figures on the yz-plane.

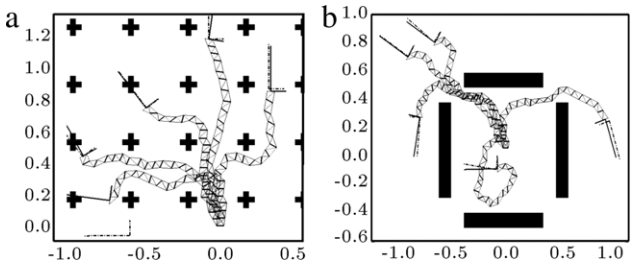

Figure 11: Five different configurations of the 2D case study corresponding to five random real problems; (a) in *P*-field, and (b) in *S*-field.

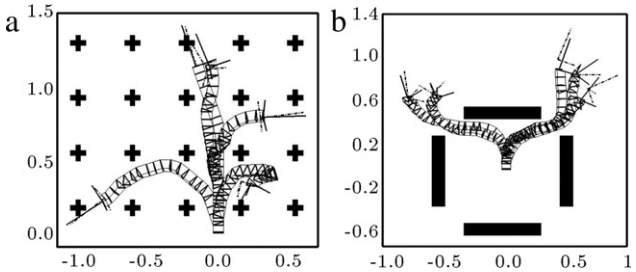

Figure 12: Five different configurations of the 3D case study corresponding to five random real problems; (a) in *P*-field, and (b) in *S*-field.

A projection is carried out to easily observe the lack of collision in the 3D case.

5. Conclusions

In this paper, a new method for solving the obstacle avoidance problem of DAHMs is presented. Two ideas are used to reduce the errors in the Suthakorn method of solving the inverse kinematic problem of DAHMs, which is used to solve the obstacle avoidance problem repeatedly: the two-by-two searching method and the iteration method.

The numerical results show that the error and solution time of the proposed method are definitely less than in the GA method. This indicates the effectiveness of the proposed method. The proposed method is an approximate one. It means that even though there is an exact solution, the method does not necessarily find it. Furthermore, although there is an exact solution, it is possible that the algorithm cannot find a collision-free answer. However, the numerical results showed that the probability of this event is low.

Appendix A

Each frame can be defined by a homogeneous transformation matrix (g), which can be expressed as follows:

$$g = \begin{bmatrix} R & b \\ 0^T & 1 \end{bmatrix} \in SE(N), \tag{A.1}$$

where $R \in SO(N)$ is the rotation matrix and $b \in R^N$ is the position vector. N for 2D cases is 2 and for 3D cases is 3. The distance between the two frames, which are illustrated by g_1 and g_2, can be expressed as follows:

$$D(g_1, g_2) = \sqrt{\|b_1 - b_2\|^2 + L^2 \|\log R_1^T R_2\|^2}, \tag{A.2}$$

where $\|\cdot\|$ is the Euclidean norm. L is a parameter mainly introduced to match units of the squared terms. In this paper, the value of $L = 0.1$ was used. Readers are referred to Park [17] for more information.

Appendix B

Consider a set of N homogeneous transformation matrixes. It is illustrated by $\{g_i = g(b_i, R_i) : i = 1, 2, \ldots N\}$ where g, R and b are described in Appendix A. The mean of this set, which is illustrated by $g_m = g\ (b_m, R_m)$, can be evaluated as follows:

$$b_m = \frac{1}{N} \sum_{n=1}^{N} b_n, \tag{B.1}$$

$$M = \frac{1}{N} \sum_{n=1}^{N} R_n, \tag{B.2}$$

$$R_m = M(M^T M)^{-1/2}. \tag{B.3}$$

Because M in Eq. (B.2) does not include $SO\ (N)$, R_m is taken to be the closest rotation matrix to M in Eq. (B.3). If $g_m = g\ (b_m, R_m)$ is the homogeneous transformation matrix related to the generalized workspace mean frame of a module, the mean frame of a generalized workspace of P similar modules, which is illustrated by $g_m^* = g\ (b_m^*, R_m^*)$, can be evaluated as follows:

$$b_m^* = \left(I + \sum_{k=1}^{P-1} m^k \right) b_m, \tag{B.4}$$

$$M^* = M^P, \tag{B.5}$$

$$R_m^* = M^* (M^{*^T} M^*)^{-1/2}, \tag{B.6}$$

where I is the unit matrix. For a 2D case, $R_m^* = (R_m)^P$, but it is not true for a 3D case, and Eqs. (B.5) and (B.6) should be used. Readers are referred to [14] for more information.

References

[1] Chirikjian, G.S. "A binary paradigm for robotic manipulators", *IEEE Int. Conf. on Robotics and Automation*, San Diego, pp. 3063–3070 (1994).
[2] Sujan, V.A., Lichter, M.D. and Dubowsky, S. "Lightweight hyper-redundant binary elements for planetary exploration robots", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1273–1278 (2001).
[3] Bayram, A. and Ozgoren, M.K. "The conceptual design of a spatial binary hyper redundant manipulator and its forward kinematics", *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 226, pp. 217–227 (2011).
[4] Proulx, S. and Plante, J.-S. "Design and experimental assessment of an elastically averaged binary manipulator using pneumatic air muscles for magnetic resonance imaging guided prostate interventions", *Journal of Mechanical Design*, 133(11), p. 111011-9 (2011).
[5] Chen, Q., Haddab, Y. and Lutz, P. "Microfabricated bistable module for digital microrobotics", *Journal of Micro-Nano Mechatronics*, 6, pp. 1–12 (2011).
[6] Petit, L., Prelle, C., Dore, E. and Lamarque, F. "Digital electromagnetic actuators array", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Montréal, Canada, pp. 720–725 (July 6–9 2010).
[7] Chirikjian, G.S. and Burdick, J.W. "An obstacle avoidance algorithm for hyper-redundant manipulators", *IEEE International Conference on Robotics and Automation*, 1, pp. 625–631 (1990).
[8] Khatib, O.Y. "Real-time obstacle avoidance for manipulators and mobile robots", *The International Journal of Robotics Research*, 5(1), pp. 90–98 (1986).
[9] Wang, C.-C., Kumar, V. and Chiu, G.-M. "A motion control and obstacle avoidance algorithm for hyper-redundant manipulators", *International Symposium on Underwater Technology*, pp. 466–471 (1998).
[10] Agirrebietia, J., Avile's, R., de Bustos, I.F. and Ajuria, G. "A method for the study of position in highly redundant multibody systems in environments with obstacles", *IEEE Transactions on Robotics and Automation*, 18(2), pp. 257–262 (2002).
[11] Ebert-Uphoff, I. and Chirikjian, G.S. "Efficient workspace generation for binary manipulators with many actuators", *Journal of Robotic Systems*, 12(6), pp. 383–400 (1995).
[12] Chirikjian, G.S. and Ebert-Uphoff, I. "Numerical convolution on the Euclidean group with applications to workspace generation", *IEEE Transactions on Robotics Automation*, 14(1), pp. 123–136 (1998).

[13] Ebert-Uphoff, I. and Chirikjian, G.S. "Inverse kinematics of discretely actuated hyper redundant manipulators using workspace densities", *IEEE International Conference on Robotics and Automation*, pp. 139–145 (1996).
[14] Suthakorn, J. and Chirikjian, G.S. "A new inverse kinematics algorithm for binary manipulators with many actuators", *Advanced Robotics*, 15(2), pp. 225–244 (2001).
[15] Lanteigne, E. and Jnifene, A. "Obstacle avoidance of redundant manipulators using workspace density functions", *Transactions of the Canadian Society for Mechanical Engineering*, 33(4), pp. 597–608 (2009).
[16] Kim, Y.Y., Jang, G.W. and Nam, S.J. "Inverse kinematics of binary manipulators by using the continuous-variable-base optimization method", *IEEE Transactions on Robotics*, 22(1), pp. 33–42 (2006).
[17] Park, F.C. "Distance metrics on the rigid-body motions with applications to mechanism design", *Journal of Mechanical Design*, 117(1), pp. 48–54 (1995).

Alireza Motahari was born in Saveh, Iran, in 1980. He received his B.S. Degree in Mechanical Engineering from Bu-Ali Sina University, Hamedan, Iran, in 2002, and an M.S. Degree from the South Tehran Branch of the Islamic Azad University, Iran, in 2004. He is currently working towards a Ph.D. Degree in Mechanical Engineering at Science and Research Branch of Islamic Azad University, Tehran, Iran. Recently his research has focused on kinematic analysis of discretely actuated hyper-redundant robotic manipulators.

Hassan Zohoor was born in Esfahan, Iran, in 1945. He received his Ph.D. Degree from Purdue University, USA, where he also spent his postdoctoral time. Currently, he is Distinguished Professor of Mechanical Engineering at Sharif University of Technology, in Iran, and also Academician and Secretary of The Academy of Sciences of IR Iran (IAS). He is author or co-author of over 400 scientific papers, two chapters of two books published by UNESCO, three chapters of three other books, and author of four technical pamphlets. Dr. Zohor was the coordinator for compiling one e-book and four CDs for four courses. He was the conductor of a country project in the area of energy. He has conducted more than twenty five research funded projects, and holds one patent approved by the Office of Patent Management, Purdue Research Foundation, USA. He has also supervised over 180 graduate theses.

Dr. Zohor was the founder, president and developer of the principal codes and regulations of Payame Noor University in Iran. He was also Head of the Department of Engineering Sciences, IAS; Deputy for the Infrastructure Affairs, Budget and Planning Organization of Iran; Head of the Institute of Research and Planning in Higher Education, Iran; Academic Vice-Minister at the Ministry of Science and Higher Education, Iran; Acting President of Alzahra University, Iran; and President of Shiraz University, Iran. He has also received several honors and awards.

M. Habibnejad Korayem was born in Tehran, Iran, in 1961. He received B.S. (Hon) and M.S. Degrees in Mechanical Engineering from Amirkabir University of Technology, Iran, in 1985 and 1987, respectively, and his Ph.D. Degree in Mechanical Engineering from the University of Wollongong, Australia, in 1994. He is, currently, Professor of Mechanical Engineering at Iran University of Science and Technology, where he has been involved in teaching and research activities in the area of robotics for the last 17 years. His research interests include: dynamics of elastic mechanical manipulators, trajectory optimization, symbolic modelling, robotic multimedia software, mobile robots, industrial robotics standards, robot vision, soccer robots, and analysis of mechanical manipulators with maximum load carrying capacity. He has published more than 370 papers in international journals and conferences in the field of robotics.