Robotica (2013) volume 31, pp. 1073–1084. © Cambridge University Press 2013
doi:10.1017/S0263574713000337

Discrete kinematic synthesis of discretely actuated hyper-redundant manipulators

Alireza Motahari†*, Hassan Zohoor‡ and M. 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, and The Academy of Sciences, Tehran, Iran*
§*Center of Excellence in Experimental Solid Mechanics and Dynamics, Iran University of Science and Technology, Tehran, Iran*

(Accepted April 10, 2013. First published online: May 14, 2013)

SUMMARY
Discrete kinematic synthesis of discretely actuated hyper-redundant manipulators is a new practical problem in robotics. The problem concerns with determining the type of each manipulator module from among several specific types, so that the manipulator could reach several specified target frames with the lowest error. This paper suggests using a breadth-first search method and a workspace mean frame to solve this problem. To reduce errors, two heuristic ideas are proposed: two-by-two searching method and iteration. The effectiveness of the proposed method is verified through several numerical problems.

KEYWORDS: Breadth-first search; Discrete actuation; Discrete kinematic synthesis; Hyper-redundant manipulator; Workspace mean frame.

1. Introduction

A robot manipulator is said to be redundant in a general sense when it possesses more degrees of freedom than required for the task. The term hyper-redundant manipulator is usually applied to a highly dexterous robotic device composed of serially connected modules. The large number of degrees of freedom allows these manipulators to avoid obstacles and to have high dexterity, concurrently. However, the kinematics and motion control of these manipulators are complex because of their large degree of freedom. Using discrete actuators instead of continuous actuators can alleviate this problem. These actuators have only a few stable states. On the other hand, their actuation is limited to some discrete amounts. For example, a binary prismatic actuator has only two states: completely extended and completely contracted.

The concept of manipulators consisting of discretely actuating joints can be found in early investigations of Pieper's planar serial digital manipulator.[1] Simple joint level control makes discrete actuators generally able to dispense with any feedback sensors. So they are often cheaper and lighter than continuous actuators. A discretely actuated hyper-redundant manipulator (DAHM) is a hyper-redundant manipulator whose all actuators are discrete. High task repeatability and high reliability are the other advantages of a DAHM. When all actuators in a DAHM are binary, the manipulator is called a binary manipulator (BM). The concrete concept of a BM – a DAHM with binary actuators – as a new paradigm in robotics was presented by Chirikjian.[2] In another development, Ebert-Uphoff made a BM with six Stewart–Gough modules.[3] Then, Suthakorn and Chirikjian designed and implemented a new spatial DAHM with three modules.[4] Sujan *et al.* further designed a lightweight BM for space exploration applications.[5]

The problem of kinematic synthesis of DAHMs is discussed in several papers. Chirikjian solved the positional kinematic synthesis problem for a BM in the two-dimensional (2D) case.[6] Here, a BM as base-line design and finite-desired positions of the BM end-effector were given to determine some kinematic parameters (design parameters). Miyahara and Chirikjian solved a similar problem for DAHMs considering both position and orientation, in 2D and 3D cases.[7] Kyatkin and Chirikjian proposed an approximate numerical synthesis method for 2D BMs to achieve a desired workspace density.[8] Kim *et al.* solved the same problem using a different method.[9] Hang *et al.* worked on the sub-workspace design of BMs, which means determining passive and active joints of a specified BM and selecting proper state for passive joints to reach a desired sub-workspace.[10]

The continuous kinematic synthesis problem is dealt with in all of these studies[6–9] but Hang *et al.*[10] Here, design parameters are selected from a continuous interval of numbers. The problem presented in this paper is a new one, dealing with discrete kinematic synthesis and its application will be discussed later in this section. The discrete kinematic synthesis problem of a DAHM can be expressed as follows:

Selecting a proper type for each module of manipulator from among several specific types is desired, so that the manipulator could reach several specified target frames called "targets" with the lowest error. Each module type is a module with a specific structure (mechanism), dimensions, and discrete amounts of actuation. Unlike most synthesis problems where design parameters can be selected from a certain interval, here designing is limited to only a few discrete options. This is why it is called discrete synthesis.

This is a practical problem because most designers are forced to choose from available options in the market. It can

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

also find applications when storing several module types in a warehouse so that, with change in the set of targets, errors in reaching these targets can be reduced by selecting the proper module types and connecting them serially to make a new DAHM.

In order to solve the discrete kinematic synthesis problem for a DAHM, it is necessary to find an effective method for solving the inverse kinematic problem. The reason for this will be discussed in Section 2.2. Several methods have been proposed by various researchers to solve this problem. Chirikjian and Ebert-Uphoff[11,12] estimate workspace density for 2D DAHMs. They used it for solving 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[13] 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. But it had the disadvantage of having large errors.

In this paper, a method is proposed for solving discrete kinematic synthesis of DAHMs using Suthakorn's method.[13] Although this method is fast, its errors are high. Here, error means the average distance of the end frame of manipulator in solution configurations from corresponding target frames. Two ideas are presented to reduce the errors. First, the type of two nonadjacent modules is determined instead of one module in each step of solution, which is called "two-by-two searching method." Second, the process is repeated to improve the results.

The paper is organized as follows. Section 2 describes the problem completely. Then, Suthakorn's method and the two ideas of solving the problem are explained. In Section 3, two algorithms are presented. The first one is based on Suthakorn's method and the second one is based on the two ideas presented in Section 2. Finally, the numerical results for a 2D and a 3D manipulator are presented as case studies.

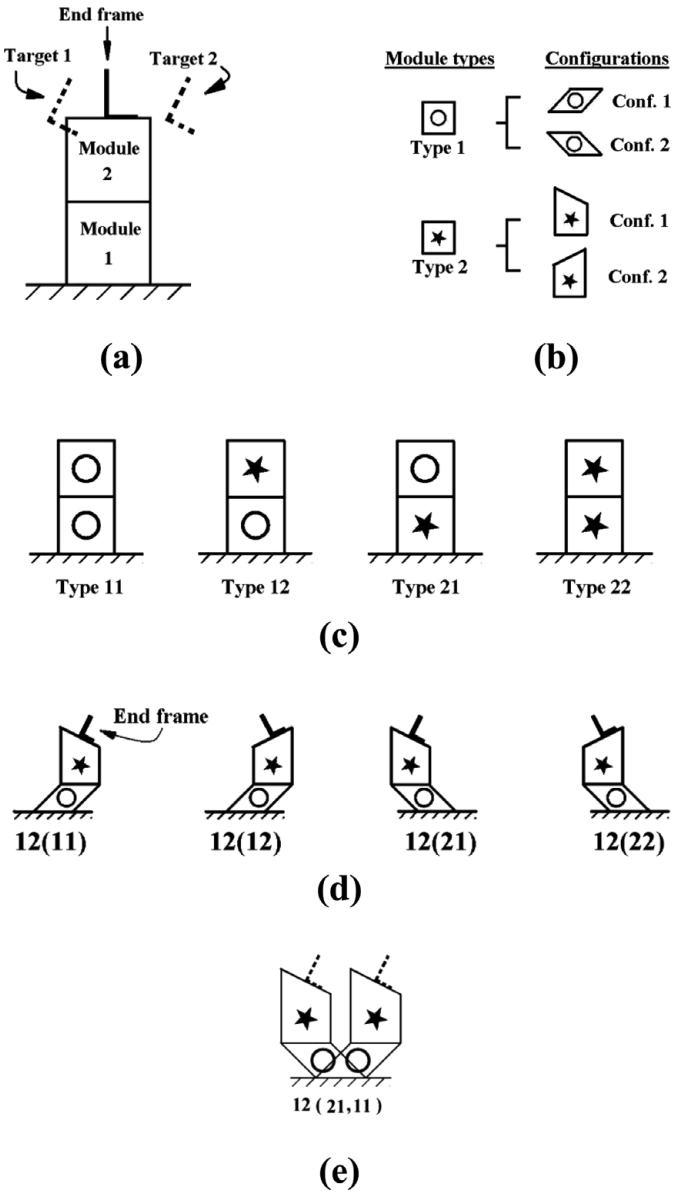

Fig. 1. Schematic illustration of a discrete kinematic synthesis problem for a manipulator.

2. Fundamentals

2.1. Discrete kinematic synthesis problem of DAHMs

The purpose of design is a DAHM that can reach several specified target frames in space (or plane) with the lowest error. These frames, which are called "target frames" or in short "targets," are the same as precision points. Designing is limited to several specified options in a manner that each module of manipulator can be selected from a list of module types. Each module type on this list is a module with a defined structure, dimensions, and discrete amounts of actuation. There is no limitation in the number of module types and the number of targets. The number of manipulator modules is an input data and all module-type actuators are discrete actuators.

To reach each target, the manipulator is actuated in a configuration. On this basis, each module of the manipulator has a different configuration for each target. Thus, solution algorithm outputs are the proper type for each module of manipulator and its proper configuration for each target. If the targets are obtained from forward kinematic solution of a manipulator with specified module types in some specified configurations, then the existence of an exact solution for the corresponding discrete kinematic synthesis problem will be certain. In this case, the distance between the end frame and the corresponding target frame can be considered as an error. Of course, if there is more than one target, the solution error will be the average of all these distances. The distance between two frames includes both location and orientation differences. The corresponding formulas are presented in Appendix A.

Figure 1 illustrates a discrete kinematic synthesis problem for a two-module manipulator with two targets and two module types. Each module type has two configurations. Figure 1(a) illustrates the manipulator, its end frame and the targets. The target frames are shown by dashed lines. Figure 1(b) illustrates the module types and their configurations. Figure 1(c) shows all possible choices of

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

manipulator type. The *i*th digit of the manipulator-type code defines the type of the *i*th module. Figure 1(d) shows all possible configurations for a manipulator of type 12. The numbers within parenthesis of the presented codes represent the manipulator configuration where *i*th digit defines the configuration of the *i*th module. The numbers outside the parenthesis show the manipulator type, which are the same as the codes in Fig. 1(c). Figure 1(e) illustrates the exact solution of the problem. The code defines the manipulator type and configurations corresponding to each target.

In the following, the discrete kinematic synthesis problem of DAHMs is stated formally.

Minimize:

$$\text{Error}\,(\mathbf{type, conf}) = \frac{1}{N_{tar}} \sum_{tar=1}^{N_{tar}} \text{Distance} \times \left(\text{EndFrame}_{tar},\ \text{TargetFrame}_{tar}\right)$$

Subject to:

$$\mathbf{type} = \left(\text{type}_1,\ \text{type}_2, \ldots, \text{type}_{N_{mod}}\right), \quad \text{where} \quad \text{type}_{mod} \in \left\{\text{CandidateType}_i : i = 1, 2, \ldots, N_{typ}\right\}$$

and

$$\mathbf{conf} = \left(\text{conf}_{mod,tar} : mod = 1, 2, \ldots, N_{mod},\ tar = 1, 2, \ldots, N_{tar}\right), \quad \text{where} \quad \text{conf}_{mod,tar} \in \left\{\text{CandidateConf}_{j,type_{mod}} : j = 1, 2, \ldots, N_{con}\left(\text{type}_{mod}\right)\right\}.$$

where EndFrame_{tar} is the end frame of the manipulator related to *tar*th target, also type_i and $\text{conf}_{i,j}$ are the type of the *i*th module and configuration of *i*th module related to *j*th target, respectively. Furthermore, N_{mod}, N_{tar}, N_{typ}, and N_{con} (k) are the number of modules of the manipulator, number of targets, number of candidate types, number of candidate configurations of $\underline{k}$th candidate type, respectively. $\text{Distance}\,(A, B)$ is the distance between frames A and B. The method for calculating this distance has been given in Appendix A.

2.2. Solution method using workspace mean frame

Our problem is somewhat similar to the inverse kinematic problem of DAHMs. Once the number of targets and the module types equal 1, the problem turns into an inverse kinematic problem. Accordingly, the inverse kinematic solution method can be generalized to a solution method for the synthesis problem. The proposed method in this paper is a generalization of Suthakorn's method[13] for solving inverse kinematic problems. Before describing this method, it is necessary to introduce "workspace density." Due to discrete actuation, the workspace of DAHMs is formed as a cloud of discrete points. The number of workspace points, which are placed into a unit value (or area) around a position, is called workspace density of that position. Therefore, if a position is denser, the probable error of reaching that position is less. Several papers have dealt with calculation of workspace density or workspace generation.[11,14] In Suthakorn's method, it is assumed that the mean position of the workspace is the densest. It is also assumed that the greater the distance from the mean position, the lower the density. A breadth-first search is used in Suthakorn's method. Here, the manipulator configuration is determined module by module, starting from the base module and continuing to the end module. Accordingly, there are three kinds of modules in the manipulator at each step, namely, a module whose appropriate configuration has been selected previously (decided module), a module whose appropriate configuration is being selected (pending module) and a module whose configuration is unknown (undecided module).

If the pending module is considered in a particular configuration, by changing configuration of undecided modules in all possible states, the end frame makes a discrete space that is a subset of the manipulator workspace, called "sub-workspace." The target frame has a density in this sub-workspace. The denser the target frame in a sub-workspace, the less probable the error. Therefore, among all possible configurations for a pending module, a configuration whose sub-workspace is denser in the target frame is selected.

In Suthakorn's method, instead of determining the density, another criterion is used to compare densities. This criterion is the distance between the target frame and mean frame of the sub-workspace. As mentioned earlier, if this distance is less, then the sub-workspace at target frame will be denser. Thus, it is only necessary to calculate the mean frame of the sub-workspace. This can be done easily by a method proposed by Suthakorn and Chirikjian.[13] This method is described in Appendix B.

Similar to inverse kinematic, the solution of discrete synthesis can be done by breadth-first search. At each step of solution, the proper type and configurations (one configuration for each target) are selected for one module (pending module). This process starts from the base module and continues to the end module of the manipulator. Just like inverse kinematic, there are three kinds of modules at each step:

1. Decided module: A module whose appropriate type and configurations have been selected in previous steps.
2. Pending module: A module whose appropriate type and configurations are being selected.
3. Undecided module: A module whose appropriate type and configurations are unknown.

Union of workspaces of all possible manipulator types forms a discrete space that we call "generalized workspace." The number of frames included in generalized workspace (N_{GW}) can be calculated by the following equation:

$$N_{GW} = \left(\sum_{i=1}^{N_{typ}} N_{con}(i)\right)^{N_{mod}} \tag{1}$$

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

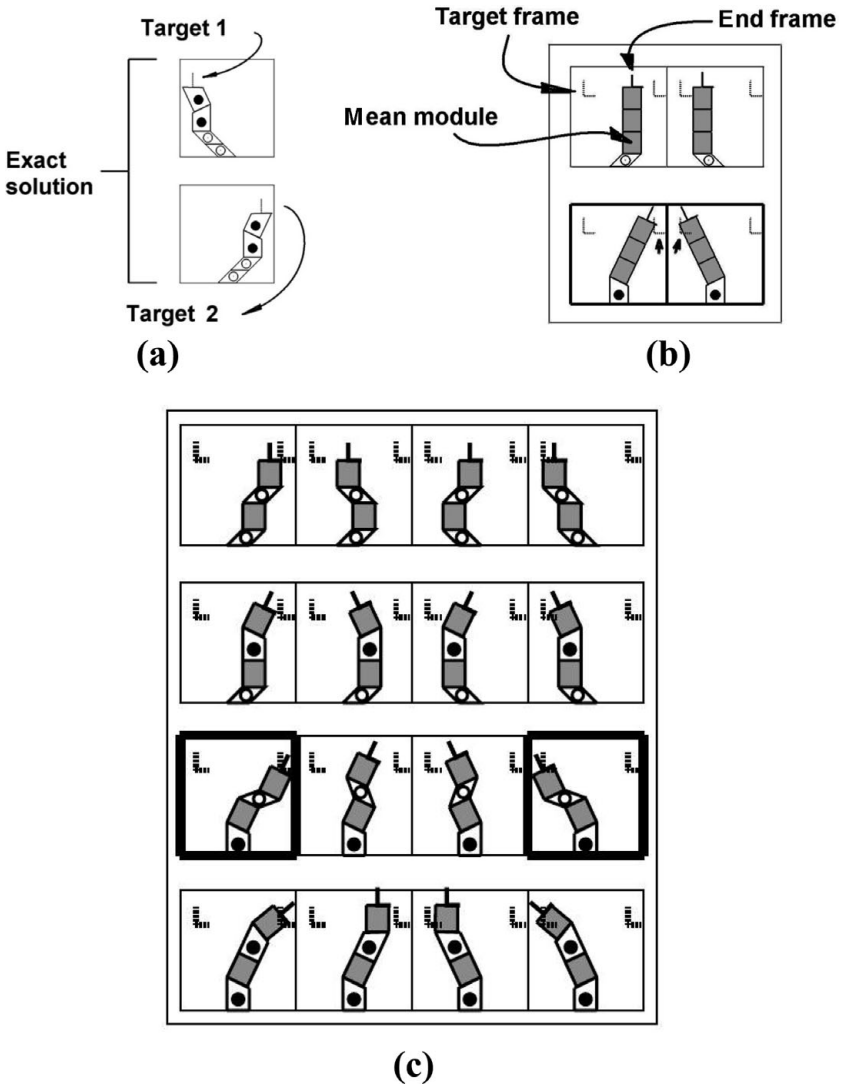

Fig. 2. Schematic illustration of the solution methods of discrete kinematic synthesis for a four-module manipulator: (a) exact solution, (b) first step of the one-by-one method, (c) first step of the two-by-two method, in which pending pair modules are the first and the third modules.

where N_{typ} is the number of module types, $N_{con}(i)$ is the number of possible configurations of module type i, and N_{mod} is the number of modules of the manipulator. The terms in the parenthesis define all possible choices for a module of the manipulator that contains all possible configurations of all module types. If the pending module is considered in a specified configuration of a specified type, by changing the type and configuration of unknown modules in all possible states, the end frame makes a discrete space that is a subset of generalized workspace, called "generalized sub-workspace." If generalized sub-workspace in target position is denser, the probable error of reaching the target will be less.

Like inverse kinematic, we assume that the densest position of a generalized sub-workspace is their mean frame. So, if the distance between the mean frame of generalized sub-workspace and the target frame is less, then the probable error will also be less. Assuming that the pending module type is specified, the best configuration of pending module for each target among all possible configurations is one, which results in less distance between the mean frame of the corresponding generalized sub-workspace and the target frame. Sum of these minimal distances on all targets is called "sum of errors." This value is calculated for all module types, one by one.

The proper type for pending module among all module types is the one whose sum of errors is less. The proper configurations have already been selected. In this manner, the proper type and configurations can be selected for the pending module. By repeating this process module by module, which starts from the base module and continues to the end module, the problem can be solved.

Figure 2 shows a discrete kinematic synthesis problem for a four-module manipulator. As with Fig. 1(b), there are two module types to choose from, each with two different configurations. Modules marked with gray hatches are mean-modules. A mean-module is a virtual module with end frame located in the mean position of generalized workspace. In other words, this frame is the average of all possible end frames for a module.

Figure 2(a) illustrates two configurations of the same manipulator type and their end frames are considered as target frames. In other words, these two configurations are

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

the exact solution of the problem. Figure 2(b) shows the first step of solving the problem using Suthakorn's method (the one-by-one searching method). Therefore, the first module is considered as pending module and the other modules are considered as undecided modules. Undecided modules are replaced with mean module. Each row in Fig. 2(b) belongs to one module type with two different configurations. Target frames are shown with dashed lines. Not only is it necessary to select two configurations each for one target but it is also necessary to select these configurations from one row (i.e., one module type). The rectangles in bold lines show the solution for the first step.

2.3. Two-by-two searching method

Numerical results show large errors in Suthakorn's method of solving the discrete synthesis problem. In order to reduce the errors, the two-by-two searching method is proposed. This method is based on Suthakorn's method, but there are two pending modules at each step of solution. Numerical results show that the errors would be reduced if these two modules are nonadjacent. If the number of modules is odd, the last step of solution is done with one module using Suthakorn's method. The order for choosing modules as pending pair modules is defined in a list that is named "order list." As an example, the order list for a four-module manipulator can be one of the following sets:

$$L_{(1)} = \{(1,2),(3,4)\}, \quad L_{(2)} = \{(1,3),(2,4)\},$$
$$L_{(3)} = \{(1,4),(2,3)\}, \quad L_{(4)} = \{(2,3),(1,4)\},$$
$$L_{(5)} = \{(2,4),(1,3)\}, \quad L_{(6)} = \{(3,4),(1,2)\}.$$

The numbers in the curly brackets define module orders in the manipulator, for example, 1 denotes the base module and 4 denotes the end module. Each parenthesis contains two numbers that define pending pair modules at each step of solution. So, the first parenthesis defines the pending modules of step 1 and so on.

Numerical results show that the selection of order list affects errors. It is better to choose nonadjacent modules as pending pair modules at each step of solution. We cannot obtain a clearer criterion for selecting the order list. We choose pending pair modules randomly from among undecided modules of two halves of the manipulator at each step of solution. The end frame of each undecided module at each step of solution is assumed to be in the mean of its workspace. So for reaching a target, assuming that the type of the pending modules is specified, the proper configuration of pending modules among all possible configuration combinations is the one, which results in a shorter distance between the end frame and the target frame. If this process is done for all targets, a proper configuration is defined for pending modules for each target. Sum of distances between the related end frame and the target frame is called "sum of errors." Each time a type combination is considered for pending modules, it results in a sum of errors. So for all possible-type combinations of pending modules, the procedure above is done and the type, which results in less sum of errors is chosen as proper type for the pending modules. The proper configurations for that type related to each target have already been selected. In this manner, the type and configurations of pending modules are selected. The type and configurations of the whole manipulator will be finally defined by repeating this process for each pair of the order list one by one.

Figure 2(c) shows the first step in the solution of the synthesis problem described in Fig. 2(a) using the two-by-two searching method; the pending modules are the first and third modules of the manipulator. The other modules (second and fourth) are undecided modules whose places are filled with a mean module (gray). Two pending modules, each with two types and each type with two configurations present $(2^2)^2 = 16$ candidates for selection, which are shown in 16 rectangles in Fig. 2(c). Each row in Fig. 2(c) belongs to one-type combination of modules. Therefore, the solution configurations are the two configurations that each of which is close to one of the two targets. In addition, they are in one row in Fig. 2(c), that is, the configurations have the same module type. The solution configurations are marked by bold rectangles.

2.4. Iteration method

Numerical results show that the iteration of two-by-two searching process reduces errors. The iteration process is done after the implementation of the two-by-two searching method. So, there is no undecided module in the iteration process. In each iteration, two modules are selected randomly from among all modules of manipulator, which are considered as pending pair modules. Then, their proper type and configurations are selected again using the two-by-two searching method.

3. Solution Algorithms

Based on what was discussed in Section 2, two individual algorithms are presented to solve discrete synthesis of DAHMs problems: the one-by-one searching algorithm and two-by-two searching with the iteration algorithm (which are called "one-by-one algorithm" and "two-by-two algorithm"). The one-by-one algorithm is based on Suthakorn's method, which was explained in Section 2.2. The term "one-by-one" is referred to one pending module at each step of solution. In the two-by-two algorithm, at first, the two-by-two searching algorithm, which is based on Section 2.3, is run. Afterward, the iteration algorithm, which is based on Section 2.4, is run for prescribed times called "number of iterations."

3.1. One-by-one algorithm

1. Find $g_{typ,con}$ for $typ = 1, 2, \ldots, N_{typ}$ and $con = 1, 2, \ldots, N_{con}(typ)$ where typ defines the module type, N_{typ} is the number of module types, con defines the configuration, and $N_{con}(typ)$ is the number of possible configurations of module-type typ. At this step, the homogeneous transformation matrix (g) is calculated for all possible configurations of all module types.

2. Find g^{mid}, which is named "mean transformation matrix." It is related to the mean frame of one module-generalized workspace. The method of calculating the mean frame is presented in Appendix B. Readers are referred to Suthakorn and Chirikjian[13] for more details.

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

3. Assume $g^{(mod,tar)} = g^{mid}$ for $mod = 1, 2, \ldots, N_{mod}$ and $tar = 1, 2, \ldots, N_{tar}$ where $g^{(mod,tar)}$ is the transformation matrix for *mod*th module of manipulator and for *tar*th target. N_{mod} is the number of manipulator modules, and N_{tar} is the number of targets.

At this step, the transformation matrix of all modules (which are yet undecided) for all targets is assumed equal to mean transformation matrix.

4. Input g^*_{tar} for tar = 1,2,. . .,N_{tar}, where g^*_{tar} is the transformation matrix of *tar*th target. At this step, the target frames are defined as inputs.

5. $mod = 1$ (*mod* defines the pending module).

6. $tar = 1$ (*tar* defines the considered target).

7. $typ = 1$ (*typ* defines the module type).

8. Do the following sub-steps for all possible configurations of pending module, which means all the following options: $con = 1, 2, \ldots, N_{con}(typ)$
where *con* defines a configuration for pending module.

8.1. $g^{(mod,tar)} = g_{typ,con}$.

8.2. Calculate: $G = g^{(1,tar)} \circ g^{(2,tar)} \circ \cdots \circ g^{(N_{mod,tar})}$,
where G is the transformation matrix of manipulator.

8.3. Calculate: $D^{tar}_{typ(con)} = \text{distance}(G, g^*_{tar})$,
where $D^a_{b(c)}$relates to the distance between the end frame of manipulator and ath target frame where the pending module is of bth type and in cth configuration. The distance(A, B) denotes the distance between the frames A and B. The method of calculating the distance between two frames is described in Appendix A.

9. Find the minimum value of $D^{tar}_{typ(con)}$ from among all values calculated in the previous step. This value is indicated by d^{tar}_{typ} and the related configuration is indicated by c^{tar}_{typ}.

10. If $typ < N_{typ}$, then $typ = typ + 1$ and go to step 8.

11. If $tar < N_{tar}$, then $tar = tar + 1$ and go to step 7.

12. Calculate the following quantity for $typ = 1, 2, \ldots, N_{typ}$, $\text{sum} - d_{typ} = \sum_{tar=1}^{N_{tar}} d^{tar}_{typ}$,
where sum – d_{typ} is the sum of errors for *typ*th type as a pending module.

13. Find the minimum value of sum – d_{typ} from among all values calculated at step 12.

The corresponding *typ* is considered as proper type for pending module and is illustrated by $TYP(mod)$. Furthermore, $CON(mod, tar) = c^{tar}_{TYP(mod)}$ for tar = 1,2,. . .,N_{tar},
where *CON(a,b)* illustrates the proper configuration of ath module to reach bth target. So,
$g^{(mod,tar)} = g_{TYP(mod),CON(mod,tar)}$ for $tar = 1, 2, \ldots, N_{tar}$.

14. If $mod < N_{mod}$, then $mod = mod + 1$ and go to step 6.

15. End.

3.2. Two-by-two algorithm

1–4. Follow the same steps in the one-by-one algorithm described above.

5. Make the order list (L) randomly with the following conditions:

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

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

$$\text{Conditions} = \begin{cases} L_{i1} < L_{i2} \\ L_{ij} \in N \\ 1 \le L_{ij} \le N_{mod} \\ L_{ij} \neq L_{rs} if i \neq r or j \neq s \end{cases}.$$

6. $i = 1$, it relates to L_i that defines the ith pending pair modules.

7. tar = 1.

8. *typ-A* = 1, it defines the type of the first pending module.

9. *typ-B* = 1, it defines the type of the second pending module.

10. Do the sub-steps for all possible configuration combinations of pending pair modules, which means all the following options:

$$con - A = 1, 2, \ldots, N_{con}(typ - A)$$
$$con - B = 1, 2, \ldots, N_{con}(typ - B)$$

where *con-A* is related to the configuration of the first pending module and *con-B* defines the configuration of the second pending module.

10.1. $g^{(L_{i1},tar)} = g_{typ-A,con-A}$

10.2. $g^{(L_{i2},tar)} = g_{typ-B,con-B}$

10.3. Calculate:

$$G = g^{(1,tar)} \circ g^{(2,tar)} \circ \cdots \circ g^{(N_{mod,tar})},$$

where G is the transformation matrix of manipulator.

10.4. Calculate:

$$D^{tar}_{typ-A(con-A),typ-B(con-B)} = \text{distance}(G, g^*_{tar}),$$

where $D^a_{b(c),d(e)}$ illustrates the distance between the end frame and ath target frame, where the type and the configuration of the first pending module are b and c, respectively, furthermore, the type and the configuration of the second pending module are d and e, respectively.

11. Find the minimum value of D among all values, which were calculated in the previous step. This value is indicated by $d^{tar}_{typ-A,typ-B}$ and the related two configurations are indicated by $\text{con} - \text{a}^{tar}_{typ-A}$ and $\text{con} - \text{b}^{tar}_{typ-B}$.

12. If *typ-B* < N_{typ}, then *typ-B* = *typ-B* + 1 and go to step 10.

13. If $typ - A < N_{typ}$, then *typ-A* = *typ-A* + 1 and go to step 9.

14. If $tar < N_{tar}$, then $tar = tar + 1$ and go to step 8.

15. Evaluate the following quantity for all combinations of *type-A* = 1, 2, . . . , N_{typ} and *typ-B* = 1, 2, . . . , N_{typ}.

$$\text{sum} - d_{typ-A,typ-B} = \sum_{tar=1}^{N_{tar}} d^{tar}_{typ-A,typ-B},$$

where sum – $d_{a,b}$ is sum of errors related to the pending pair modules of types a and b.

16. Find the minimum value of sum $-d_{typ-A,typ-B}$ among all values, which were calculated in the previous step (step

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

Table I. Time complexity of the proposed algorithms considering various parameters.

	N: number of modules	M: number of targets	R: number of module types	S: number of iteration
One-by-one algorithm	$\mathcal{O}(N^2)$	$\mathcal{O}(M)$	$\mathcal{O}(R)$	–
Two-by-two algorithm	$\mathcal{O}(N^2)$	$\mathcal{O}(M)$	$\mathcal{O}(R^2)$	$\mathcal{O}(S)$

15). The corresponding *typ-A* and *typ-B* are considered as the proper types for the first and the second pending module, respectively, and they are illustrated by $TYP(L_{i1})$ and $TYP(L_{i2})$, respectively. Furthermore,
$CON(L_{i1}, tar) = \text{con} - \text{a}^{tar}_{TYP(L_{i1})}$ for $tar = 1, 2, \ldots, N_{tar}$,
$CON(L_{i2}, tar) = \text{con} - \text{b}^{tar}_{TYP(L_{i2})}$ for $tar = 1, 2, \ldots, N_{tar}$,
where $CON(a,b)$ illustrates the proper configuration of ath module to reach bth target. So,
$g^{(L_{i1},tar)} = g_{TYP(L_{i1}),CON(L_{i1},tar)}$ for $tar = 1, 2, \ldots, N_{tar}$,
$g^{(L_{i2},tar)} = g_{TYP(L_{i2}),CON(L_{i2},tar)}$ for $tar = 1, 2, \ldots, N_{tar}$.
17. If $i < N_{mod}/2$, then $i = i + 1$ and go to step 7.
18. The iteration algorithm:
18.1. $itr = 1$, which means first iteration.
18.2. Select two modules (A, B) randomly among all manipulator modules, which are considered as pending modules. Their conditions are as follows: $A, B \in \mathbb{N}1 \leq A, B \leq N_{mod} A < B$
18.3. Do steps 7 to 16 by replacing A and B instead of L_{i1} and L_{i2}, respectively.
18.4. $Error = \frac{1}{N_{tar}} \min(sum\_d)$
185. If $itr < N_{itr}$, then $itr = itr + 1$ and go to sub-step 18.2. N_{itr} is the whole number of iterations.
19. End.

Table I shows the time complexity of the two presented algorithms based on the number of modules, number of targets, number of module types, and the number of iteration. This last item, that is, the number of iteration is only used for the two-by-two algorithm. It should be mentioned that in all cases time complexity is the same for 2D and 3D states. As can be seen in the table, time complexities are polynomial and not exponential. Therefore, the problem is not NP-hard.

With some modifications in the presented algorithm, it is possible to solve the following new problems:

- The *inverse kinematic problem* can be solved using Suthakorn's method; to do this, in the one-by-one algorithm (Section 3.1), we should just consider $N_{typ} = 1$, $N_{tar} = 1$.
- The *inverse kinematic problem* can be solved using the two-by-two method; to do this, in the two-by-two algorithm (Section 3.2), N_{typ}, N_{tar} values are considered as 1.
- *Problem P*2 can be defined with inclusion of $Error \leq \varepsilon$ as a constraint in the discretely synthesis problem (problem P1), where ε is a prescribed small tolerance. This problem can be solved using the two-by-two algorithm; it suffices to replace sub-step 18.5 with the following command:

"If $Error > \varepsilon$, then $itr = itr + 1$ and go to sub-step 18.2. N_{itr} is the whole number of iterations."

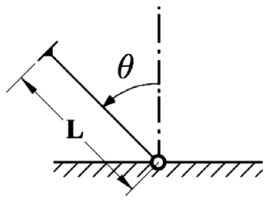

Fig. 3. An R-link module.

Problem P3 is the general version of P2 that is defined considering weight factor for each module type, that is, cost. To include weights in the two-by-two algorithm, the following command is added to step 4:

"Input W_{typ} for $typ = 1, 2, \ldots, N_{typ}$, where W_i is the weight of ith module type."
The following command should also replace step 15:

$$\text{"sum} - d_{typ-A,typ-B} = \sum_{tar=1}^{N_{tar}} \left(W_{typ-A} + W_{typ-B}\right) d^{tar}_{typ-A,typ-B}."$$

Problem P4 is defined considering the number of manipulator modules as a designed parameter. For solving this problem, it is only needed to run the presented algorithms for various numbers of modules. The final answer among all answers is the one whose sum of errors is the least. Choosing proper actuator from a few types is another case that can be solved using each of these two algorithms. In this case, the module types can be defined by changing the type of its actuators.

4. Numerical Results

In this section, discrete synthesis problems of a 2D and a 3D DAHM are solved numerically. Two module types for the 2D case and two module types for the 3D case are introduced in the following subsection. Method of evaluating errors is described in the next subsection. After that, numerical results for these two cases are presented. All calculations are done by MATLAB software with an Intel 1.66 GHz processor.

4.1. Introducing module types

4.1.1. R-link: A module type for the 2D case. An R-link module is shown in Fig. 3. It contains a link with constant length of L and a revolute joint in the beginning of the link, which joints the module to the previous module of manipulator. This joint can be discretely actuated only in four angles of θ (Fig. 3). So, this module type has four configurations. The value of L is $1/N_{mod}$ and the discrete values of θ are $\{-\pi/9, -\pi/18, \pi/18, \pi/9\}$.

4.1.2. VGT: A module type for the 2D case. A VGT module is shown in Fig. 4. It contains three binary prismatic actuators in *AD*, *AC*, and *BC* links. The two other links (*AB, CD*) have constant lengths. A, B, C, and D are passive revolute joints. This module type has $2^3 = 8$ configurations. The length of constant links is $1/N_{mod}$. Discrete actuation amounts of binary actuators (which are the length of links *AD, AC*, and

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

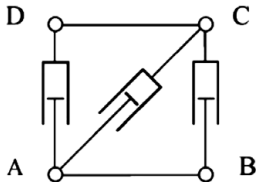

Fig. 4. A VGT module.

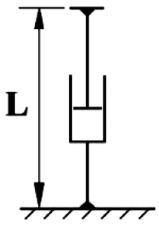

Fig. 5. A P-link module.

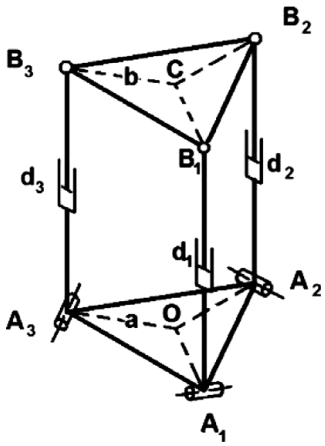

Fig. 6. A 3-RPS module.

BC) are $\{1/N_{mod}, 1.5/N_{mod}\}$. Readers are referred to Kim *et al.* paper[15] for forward kinematics of this module type.

4.1.3. P-link: A module type for the 3D case. A p-link module is shown in Fig. 5. It contains a link with a discrete prismatic actuator that has four stable states. So, this module type has four configurations. The link can be connected vertically to the previous and next module. The discrete amounts of actuation, which are the length of the link (L in Fig. 5), are $\{0.75/\mathrm{N}_{mod}, 1/N_{mod}, 1.25/N_{mod}, 1.5/N_{mod}\}$.

4.1.4. 3-RPS: A module type for the 3D case. A 3-RPS module is shown in Fig. 6. It is a parallel robot with three binary prismatic actuators in its three legs A_1B_1, A_2B_2, and A_3B_3. So, it has $2^3 = 8$ configurations. The base plate ($A_1A_2A_3$) and the moving plate ($B_1B_2B_3$) are the same equilateral triangles. A_1, A_2, and A_3 are passive revolute joints. B_1, B_2, and B_3 are passive spherical joints. The rotation axes of the A_1, A_2, and A_3 joints are parallel to their opposite sides in triangle $A_1A_2A_3$.

The distance between center and corners of these two triangles are equal to $1/N_{mod}$. The discrete amounts of actuation, which are the length of the legs, are $\{1/N_{mod}, 1.5/N_{mod}\}$. Readers are referred to refs. [15, 16] for the forward kinematic solving of this module type.

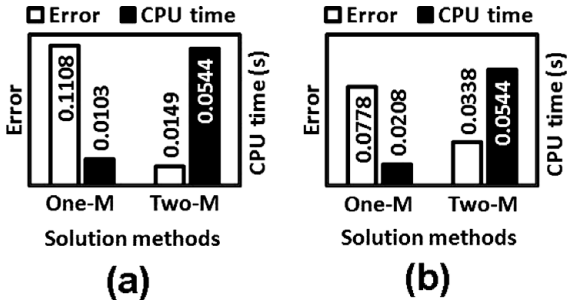

Fig. 7. (a) Error and CPU time of the inverse kinematic solution of the 2D case study (a manipulator that contains 20 VGT modules) using two methods: the one-by-one method (One-M) and the two-by-two method (Two-M), (b) same results for the 3D case study (a manipulator that contains 20 modules of type 3-RPS).

4.2. Defining errors

As mentioned before, the target frames are calculated by solving forward kinematic of a specified manipulator type in some specified configurations. Thus, the problem has an exact solution and the average of the distances between targets and corresponding end frames is considered as error.

Furthermore, for analysis convenience, it is necessary to use dimensionless quantities. Therefore, all lengths are divided by the minimum length of the manipulator whose value is considered equal to the number of modules (N_{mod}) in all cases.

4.3. Numerical results

As mentioned earlier, two case studies are considered to examine the performance of the two proposed algorithms. The first case study is a 2D manipulator with two module types: R-link and VGT. The second case study is a 3D manipulator with two module types: P-link and 3-RPS. The results are described in the following.

Figure 7 makes a comparison between the performance of the one-by-one algorithm (Suthakorn's method) and the two-by-two algorithm in solving the inverse kinematic problem. For the 2D case study, a 20-module manipulator whose all modules are of VGT type was considered. Similarly, for the 3D case study, a 20-module manipulator whose all modules are of 3-RPS type was considered. The target frames are obtained by solving forward kinematic for some random configurations of the manipulator. It guaranties the existence of an exact solution for all sampled problems. Every value presented in this figure is an average of 100 values over 100 random sample problems. A set of real targets are used commonly for both algorithms. The number of iterations in the two-by-two algorithm is 10. According to the presented results in this figure, although the solution time of Suthakorn's method is less than the other method; its errors are high.

Figures 8 and 9 are presented to compare the performance of the one-by-one algorithm and the two-by-two algorithm in solving the discrete synthesis problem (P1) in the 2D and 3D case studies, respectively. Error and CPU time for 4-, 10-, 20-, and 40-module manipulators are presented. There are five random targets in all cases. The number of iterations in the two-by-two algorithm is 50, for both figures. Every

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

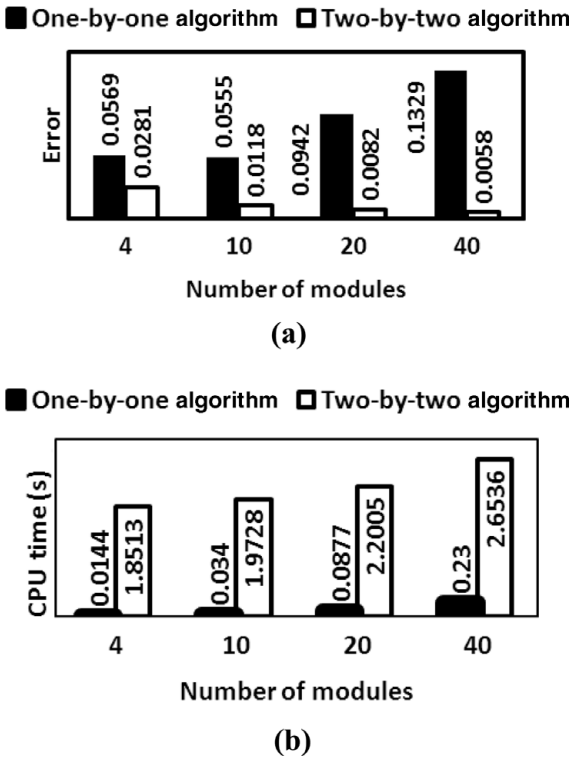

Fig. 8. Comparing (a) error and (b) CPU time of the two presented algorithms results in the 2D case study.

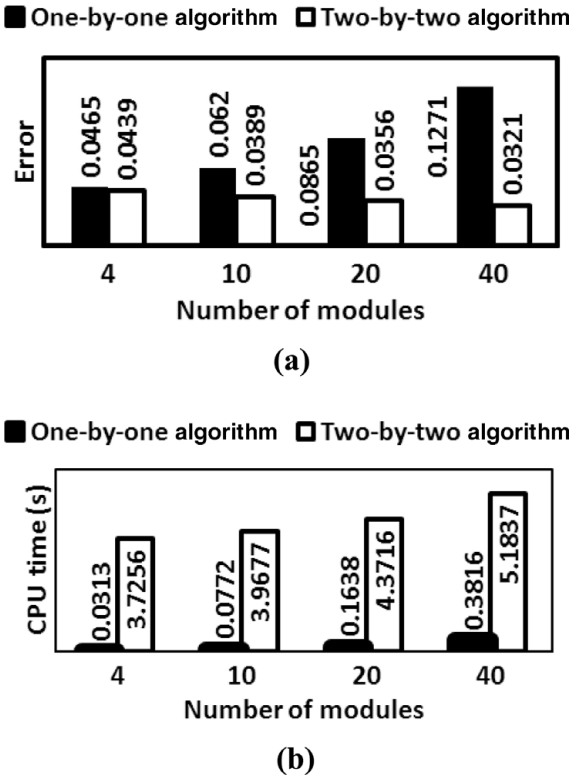

Fig. 9. Comparing (a) error and (b) CPU time of the two presented algorithms results in the 3D case study.

value in these two figures is an average of 100 values over 100 random samples.

As it can be seen, the errors of the two-by-two algorithm are definitely less than the one-by-one algorithm, especially for manipulators with more modules. CPU time of the two-by-two algorithm is longer than the one-by-one algorithm; however, it is not more than a few seconds. Errors in the two-by-two algorithm decrease when the number of modules is increased, but they increase in the one-by-one algorithm. According to the above-mentioned results, the two-by-two algorithm is more appropriate than the one-by-one algorithm.

Figures 10 and 11 illustrate the effect of iterations on error and CPU time of the two-by-two algorithm in the 2D and 3D case studies, respectively. The number of modules is 20 and the number of targets is 5, for both figures. Every value in these two figures is an average of 20 values over 20 random samples. CPU time grows almost linearly with the number of iterations as shown in Figs. 10 and 11. Errors decrease when the number of iterations is increased, but the rate of descending decreases, so the error curve becomes almost horizontal after some iteration. It means that some errors usually remain present; no matter the how many times the iteration algorithm is repeated.

Figures 12 and 13 illustrate the effect of the number of targets on error and CPU time of the two-by-two algorithm in the 2D and 3D case studies, respectively. The number of modules of manipulator is 20 and the number of iterations is 50, for both figures. Every value in these two figures is an average of 20 values over 20 random samples. As it can be seen, both error and CPU time increase by increasing the number of targets. The growth of CPU time is approximately linear.

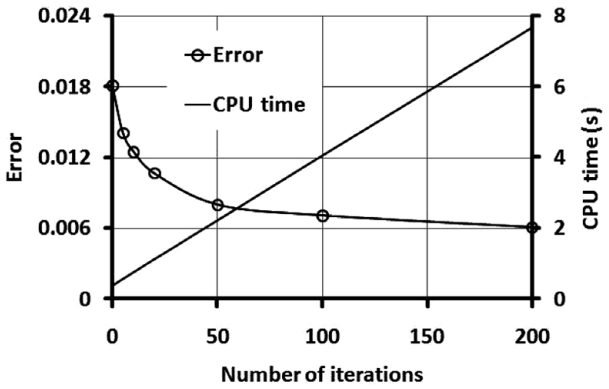

Fig. 10. The effect of iterations on error and CPU time of the two-by-two algorithm in the 2D case study.

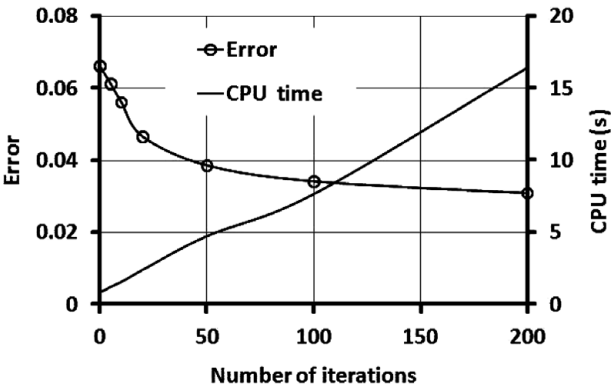

Fig. 11. The effect of iterations on error and CPU time of the two-by-two algorithm in the 3D case study.

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

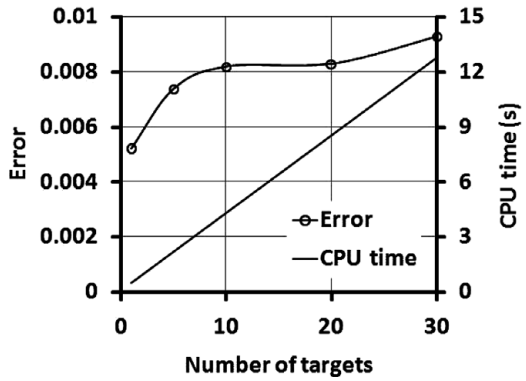

Fig. 12. The effect of the number of targets on error and CPU time of the two-by-two algorithm in the 2D case study.

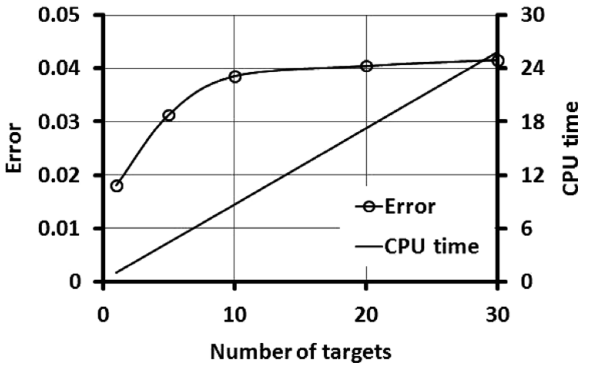

Fig. 13. The effect of the number of targets on error and CPU time of the two-by-two algorithm in the 3D case study.

Figure 14 illustrates the effect of allowable error (ε) on CPU time of the two-by-two algorithm (problem P2). Figures 14(a) and 14(b) for the 2D and 3D case studies are presented, respectively. In both cases, the number of manipulator modules is 20, and the number of targets is 5. Each value in the figure is an average of 20 values over 20 random samples. In both 2D and 3D cases, the graph gradually becomes horizontal by a reduction of ε. Because, after a number of iterations, the manipulator configuration remains stationary.

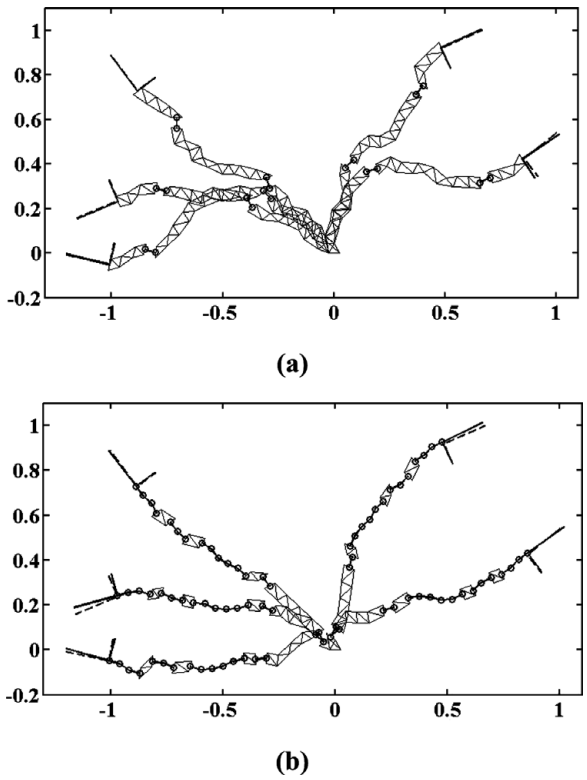

Fig. 15. Solution of a random synthesis problem that has five targets for a 2D 20-module manipulator using the two-by-two algorithm: (a) without considering weights and (b) considering weights.

Figures 15 and 16 show the solution of a random synthesis problem using the two-by-two algorithm in the 2D and 3D case studies, respectively. Figures 15(a) and 16(a) correlated with problem P1 (i.e., weights are not considered). Figures 15(b) and 16(b) are related to problem P3 (i.e., weights are considered). The weights of the modules VGT, R-link, 3-RPS, and P-link are 1, 0.6, 1, and 0.8, respectively. The increase in weight factors brings about an increase in errors. Hence, the algorithm tends to select those module types with less weight. Accordingly, the number of R-link modules in Fig. 15(b) is larger, compared to Fig 15(a). Similarly, the number of P-link modules in Fig. 16(b) is larger, compared to Fig. 16(a). The number of iterations is 50 in both 2D and

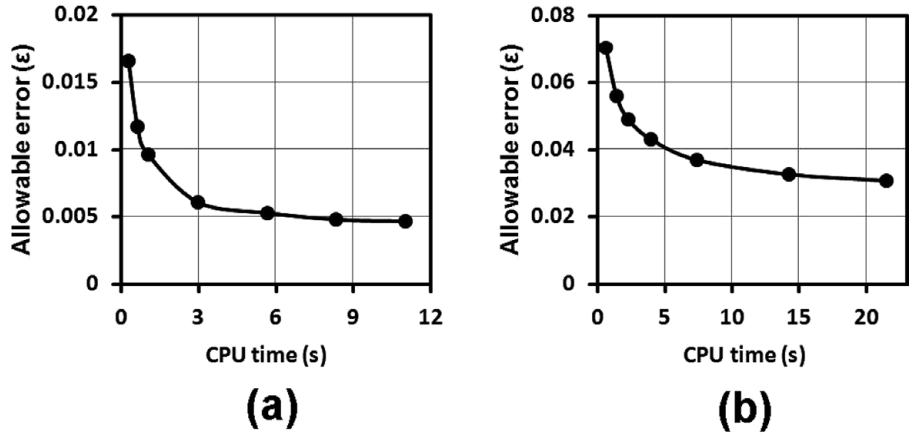

Fig. 14. The effect of allowable error on CPU time of the two-by-two algorithm (problem P2) for: (a) the 2D case study and (b) the 3D case study.

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

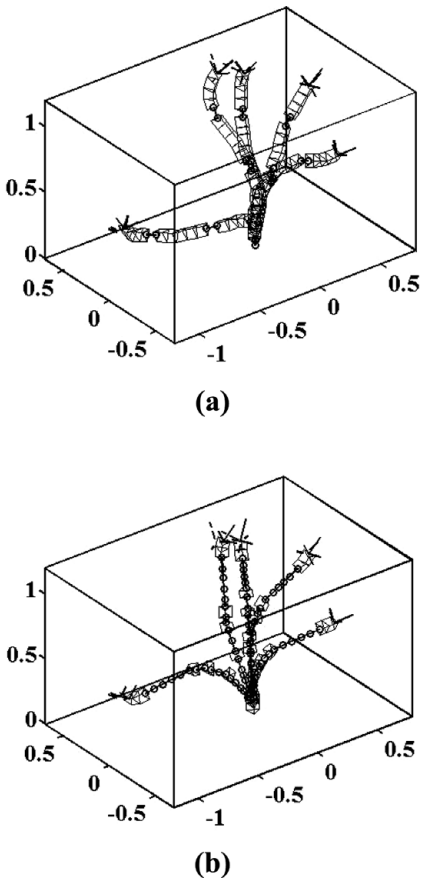

Fig. 16. Solution of a random synthesis problem that has five targets for a 3D 20-module manipulator using the two-by-two algorithm: (a) without considering weights and (b) considering weights.

3D cases. The errors without applying the weights (i.e., the average of distances between end frames and target frames) for Figs. 15(a), 15(b), 16(a), and 16(b) are 0.0057, 0.0092, 0.0500, and 0.0963, respectively. As expected, the average of distances, considering the weights, increases. The CPU time of Figs. 15(a), 15(b), 16(a), and 16(b), respectively, is equal to 2.1406, 2.1406, 4.2656, and 4.8906 s. The CPU time does not have a significant difference in the comparison between weighted and weightless states. The end frames and the target frames in Figs. 15 and 16 are shown by solid lines and dashed lines, respectively.

5. Conclusion

Two algorithms have been proposed to solve the discrete kinematic synthesis problem for DAHMs: the one-by-one algorithm and the two-by-two algorithm. A breadth-first search method, which uses single module searching in each step and the workspace mean frame are used in the one-by-one algorithm. The two-by-two algorithm reduces the errors using two heuristic ideas: two-by-two searching method and iteration. In the two-by-two searching method, a breadth-first search method with two nonadjacent modules in each step is used. Numerical results show that the iteration of this method reduces the errors. The proposed method is an approximation method and the errors will not necessarily go to zero. The authors did not manage to mathematically prove the effectiveness of their two ideas; and this can be considered as an open problem by researchers.

References

1. D. L. Pieper, The Kinematics of Manipulators Under Computer Control *Ph.D. Dissertation* (Stanford, CA: Stanford University, Oct. 1968).
2. G. S. Chirikjian, "A Binary Paradigm for Robotic Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, California (May 8–13, 1994) pp. 3063–3070.
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, "Design and implementation of a new discretely-actuated manipulator," *Exp. Robot. VII, Springer Series: Lecture Notes in Control and Information Sciences* **271**, 151–158 (2001).
5. V. A. Sujan, M. D. Lichter and S. Dubowsky, "Lightweight Hyper-Redundant Binary Elements for Planetary Exploration Robots," *Proceedings of the IEEE/ASME International Conferences Advanced Intelligent Mechatronics*, Como, Italy (2001) pp. 1273–1278.
6. G. S. Chirikjian, "Kinematic synthesis of mechanisms and robotic manipulators with binary actuators," *ASME J. Mech. Des.* **117**, 573–580 (1995).
7. K. Miyahara and G. S. Chirikjian, "General Kinematic Synthesis Method for a Discretely Actuated Robotic Manipulator (D-Arm)," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, (2006) pp. 5889–5894.
8. A. B. Kyatkin and G. S. Chirikjian, "Synthesis of binary manipulators using the Fourier transform on the Euclidean group," *ASME J. Mech. Des.* **121**, 9–14 (1999).
9. P. T. Kim, Y. Liu, Z. Luo and Y. Wang, "Deconvolution on the Euclidean motion group and planar robotic manipulator design," *Robotica* **27**, 861–872 (2009).
10. G.-W. Hang, D. J. Nam and Y. Y. Kim, "Sub-workspace design of binary manipulators using active and passive joints," *J. Mech. Sci. Technol.* **22**, 1707–1715 (2008).
11. I. Ebert-Uphoff and G. S. Chirikjian, "Efficient workspace generation for binary manipulators with many actuators," *J. Robot. Syst.* **12**(6), 383–400 (1995).
12. I. Ebert-Uphoff and G. S. Chirikjian, "Inverse Kinematics of Discretely Actuated Hyper Redundant Manipulators Using Workspace Densities," *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis (1996) pp. 139–145.
13. J. Suthakorn and G. S. Chirikjian, "A new inverse kinematics algorithm for binary manipulators with many actuators," *Adv. Robotics* **15**(2), 225–244 (2001).
14. Y. F. Wang and G. S. Chirikjian, "Workspace generation of hyper-redundant manipulators as a diffusion process on SE(N)," *IEEE Trans. Robot. Autom.* **20**(3), 399–408 (2004).
15. 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).
16. N. Mohan Rao and K. Mallikarjuna Rao, "Dimensional synthesis of a spatial 3-RPS parallel manipulator for a prescribed range of motion of spherical joints," *J. Mech. Mach. Theory* **44**, 477–486 (2009).
17. F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *Trans. ASME* **117,** 48–54 (1995).
18. A. B. Kyatkin and G. S. Chirikjian, *Engineering Applications of Noncommutative Harmonic Analysis*, Chapter 6. (CRC Press, 2000).

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press

Appendix A

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

$$g = \begin{bmatrix} R & b \\ 0^T & 1 \end{bmatrix} \in SE(N), \tag{A1}$$

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 two frames, which are illustrated by g_1 and g_2 can be expressed as follows:

$$D(g_1, g_2) = \sqrt{b_1 - b_2^2 + L^2 log R_1^T R_2^2}, \tag{A2}$$

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 refs. [17, 18] for more information.

Appendix B

Consider a set of N homogeneous transformation matrixes. It is illustrated by $\{g_i = g(b_i, R_i) : i = 1, 2, ...N\}$, 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 calculated as follows:

$$b_m = \frac{1}{N} \sum_{n=1}^{N} b_n, \tag{B1}$$

$$M = \frac{1}{N} \sum_{n=1}^{N} R_n, \tag{B2}$$

$$R_m = M(M^{\mathrm{T}} M)^{-1/2}. \tag{B3}$$

Because M in Eq. (B2) is not included in $SO(N)$, R_m is taken to be the closest rotation matrix to M in Eq. (B3). 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 calculated as follows:

$$b_m^* = \left(I + \sum_{k=1}^{P-1} m^k \right) b_m, \tag{B4}$$

where I is the unit matrix.

$$M^* = M^{\mathrm{P}}, \tag{B5}$$

$$R_m^* = M^* \left(M^{*^{\mathrm{T}}} M^* \right)^{-1/2}. \tag{B6}$$

For the 2D case, $R_m^* = (R_m)^{\mathrm{P}}$, but it is not true for the 3D case and Eqs. (B5) and (B6) should be used. Readers are referred to ref. [13] for more information.

https://doi.org/10.1017/S0263574713000337 Published online by Cambridge University Press