

# A Modular Control Scheme for Hyper-redundant Robots

Regular Paper

---

Chang Nho Cho<sup>1</sup>, Hyunchul Jung<sup>1</sup>, Jaebum Son<sup>1</sup> and Kwang Gi Kim<sup>1\*</sup>

<sup>1</sup> National Cancer Center, Goyang, Gyeonggi-do, Republic of Korea

\*Corresponding author(s) E-mail: kimkg@ncc.re.kr

Received 12 February 2014; Accepted 26 March 2015

DOI: 10.5772/60602

© 2015 Author(s). Licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

---

## Abstract

Hyper-redundant robots, robots with many degrees of freedom, are considered to be advantageous in many tasks, such as minimally invasive surgery, surveillance and inspection. However, due to their hyper degrees of freedom, the control of hyper-redundant robots is always challenging. Several fitting algorithms, which iteratively fit a hyper-redundant robot into a continuous curve, have been proposed to control the configuration of hyper-redundant robots. However, these algorithms require heavy computation, preventing them from being used in practice. In this study, we propose a novel modular control scheme for a hyper-redundant robot to reduce the computational load by dividing the robot into smaller modules and fitting each module separately. A Jacobian-based position control algorithm is also used to utilize the redundancy of each module to ensure that the overall configuration of the robot resembles the given desired curve. Simulation results show that the proposed scheme can be used to control hyper-redundant robots effectively.

**Keywords** Fitting Algorithm, Hyper-Redundant Robot, Redundancy Resolution, Weight Least Norm

## 1. Introduction

There has been much study devoted to the hyper-redundant robot, which is a robot with lots of redundant joints. Redundancy improves the dexterity and the robustness of a robot [1] and thus, it is expected that hyper-redundant robots would show better performance than conventional robots, especially in an unstructured environment. For this reason, many hyper-redundant robots have been developed for tasks such as minimally invasive surgery [2-4], surveillance [5] and inspections [6]. However, the practical application of hyper-redundant robots has been limited because the control of a hyper-redundant robot requires all active joints to be controlled in a systematic way to create a well-defined task motion. Often, the pseudo-inverse of Jacobian is used to control a redundant robot, which allows the robot to utilize the redundancy to achieve a secondary goal, such as avoiding an obstacle [7], joint limits [8, 9] and kinematic singularity [10]. However, these Jacobian-based methods are intended for a robot with one or two additional degrees of freedom (DOF) and they cannot be used to control a hyper-redundant robot. Furthermore, these methods cannot effectively control the overall configuration of a hyper-redundant robot, which is often desired.

There has been much research done on the control of hyper-redundant robots. These algorithms focus on expressing the desired posture of the robot as a backbone curve, and controlling the robot to resemble the created curve. A modal-based approach was introduced to control a hyper-redundant robot effectively [11], but that method is limited to a planar robot. A solution based on the shape Jacobian was also proposed to control the shape of a robot [12]. However, the solution is limited to a robot with two DOF revolute joints. A shape estimation method using an extended Kalman filter and an electromagnetic sensor was proposed [13], which is applicable if and only if the robot is controlled by a follow-the-leader algorithm.

On the other hand, fitting algorithms, which iteratively fit the configuration of a hyper-redundant robot into a continuous backbone curve, are also widely studied in order to control hyper-redundant robots. Fitting algorithms are often preferred as they can better reflect the discrete structure of hyper-redundant robots. A fitting algorithm, which approximates the backbone curve by piecewise line segments, was introduced [14], but the algorithm is only applicable to a robot with universal joints. Another algorithm, which can be used on a hyper-redundant robot of any joint configuration, was introduced [15]. A fitting algorithm with reduced computational load was also introduced in [16], which is also limited to robots with universal joints.

However, while these algorithms can be used to control a hyper-redundant robot, the required computational load is still too heavy for practical uses, especially if the robot has many redundant DOF. It was found that to improve the performance of the fitting algorithms, the required computational load must increase exponentially [15]. Another concern with fitting algorithms is that they focus on controlling the configuration of hyper-redundant robots and they cannot be used to place the end-effector of the robots at the desired position. However, since many hyper-redundant robots perform tasks using tools at their end-effectors, its end-effector positioning accuracy must be guaranteed.

In this study, in order to reduce the computational burden of fitting algorithms, we propose a novel control scheme to distribute the computational load among multiple controllers. The scheme divides the robot into modules and applies a fitting algorithm and a Jacobian-based position control algorithm. The fitting algorithm ensures that the configuration of the robot resembles the desired curve while the position control algorithm utilizes the redundancy of each module to enable parallel computation. The advantages of the proposed scheme are as follows. First, the scheme is computationally efficient, as each controller only has to fit the assigned module. Second, the scheme is not restricted to a robot with a certain joint configuration. Lastly, by using the position control algorithm, the end-effector accuracy can be guaranteed. A similar concept for a hyper-redundant robot was proposed in [17], but the study aimed to

remove the redundancy of each module. In contrast, the proposed scheme focuses on utilizing the redundancy to enable parallel computation.

## 2. Modular Control Scheme

### 2.1 Controller Configuration

To enable modular control, a given desired backbone curve must be divided into smaller segments for each module. Two types of controllers are involved in this scheme: the main controller and the module controllers. The main controller is in charge of dividing the curve and assigning it to each module. Then, each module is fitted to the assigned segment by its module controller. Following the fitting, a Jacobian-based position control algorithm is used to adjust each module so that the resulting configuration of the robot resembles the given backbone curve. The overall flow of the proposed scheme is illustrated in Figure 1 and a detailed explanation of each step will be given in the following sections.

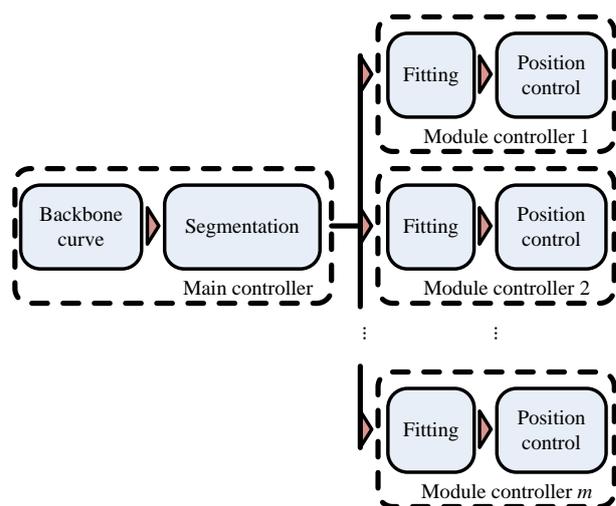


Figure 1. Proposed control scheme and role of each controller

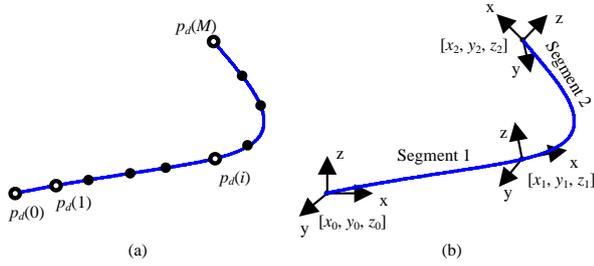
### 2.2 Segmentation of Backbone curve

We first assume that a desired backbone curve is given as a point set  $p_d(\cdot)$ . Many studies have been devoted to how to create a backbone curve for a hyper-redundant robot, such as [18-20], and we will not discuss them here as they are beyond the scope of this study.

Assuming the robot consists of  $m$  modules,  $m$  segments are required. Thus, the main controller divides the given point set  $p_d(\cdot)$  into  $m$  segments. Note that each module is connected in a series and thus, for parallel computation, each module must be able to completely cover the assigned segment. This cannot be done if the given segment is longer than the length of the module. Therefore, the following condition is used to divide the given backbone curve:

$$l_m - \sum \|p_d(i+1) - p_d(i)\| \leq \varepsilon \quad i = 1 \dots p \quad (1)$$

where  $l_m$  is the length of the module,  $p$  is the total number of points in the given backbone curve and  $\varepsilon$  is the pre-defined threshold. This ensures that the total length of the segment is shorter than the length of a module. A special case for the segmentation would be the case when the length of the given backbone curve is shorter than the length of the hyper-redundant robot. In such a case, only parts of the robot will be controlled to resemble the backbone curve while other parts remain uncontrolled.



**Figure 2.** Segmentation of the backbone curve: (a) given backbone curve, and (b) segmentation and attached frames

To enable parallel computation, the start and end position and orientation of each segment must be specified. This is due to the fact that all the modules are fitted at the same time, and the end position and orientation of module  $j$  serves as the starting position and orientation of module  $j+1$ . Therefore, each segment must include six constraints: three in position and three in orientation. This can be done by attaching a frame coordinate at each end of the segment, as shown in Figure 2. Each segment and its constraints must also be expressed in the base coordinate of the module to allow independent computation. Thus, once segment  $j$  is found using Eq. (1), we must represent it with respect to frame  $j-1$ , as it is currently expressed in frame 0. It can be done by multiplying the computed segment by  $R_0^{j-1}$ , where  $R$  is a rotation matrix. Then, we compute the normalized tangent vector at the end of the segment. The controller will create frame  $j$  at the end of the segment  $j$ , with the  $x$ -axis of the created frame aligned with the computed tangent vector. This alignment is employed to have the approach direction of the module pointing at the tangent of the curve. To obtain such a frame, we rotate the base frame of the segment, frame  $j-1$ , so that its  $x$ -axis is parallel to the tangent vector. The rotational relationship between frame  $j$  and  $j-1$  defines  $R_{j-1}^j$ . The position constraint would be the end position of the segment and the desired orientation is expressed in  $R_{j-1}^j$ . Once the constraints are obtained, the algorithm proceeds to the next segment. Note that  $R_0^0$  would be a 3-by-3 identity matrix.

Once the segmentation is completed, the main controller will send the rotated segment and the six constraints to each

module controller, and this completes the task of the main controller. The module controller will actually control each module so that the robot can follow the desired curve.

### 2.3 Fitting Algorithm

The data about the segment and its six constraints are provided to each module controller, so that the module controller can perform actual fitting. It was found that fitting multiple joints at the same time yields a better result, but at the cost of a much higher computational load. This coincides with the result presented in [15]. Once the number of joints to be fitted simultaneously is set, we perform bracketing, which coarsely finds the set of backbone curve points to be used to optimize the set of joints. This is done so that the controller does not have to perform the optimization over the entire segment. Bracketing is similar to the algorithm for dividing the curve into segments:

$$\sum \|p_d(i+1) - p_d(i)\| \geq l_k, \quad i = 1 \dots p_s \quad (2)$$

where  $l_k$  is the length of the part of the module to be fitted at the same time and  $p_s$  is the number of points in a segment. Then, the following optimization rule is used:

$$\arg \min \left( \sum_{c=0}^k (X_c(q) - P_d(i))^2 \right), q \in (q_{min}, q_{max}) \text{ and } i \in (p_i, p_e) \quad (3)$$

where  $X$  is the distal-end position of the module,  $q_{min}$  and  $q_{max}$  are the minimum and maximum joint angles, respectively, and  $p_i$  and  $p_e$  are the starting and end points of the bracketed segment. Note that the sum of the squared distance is used as the objective function. This optimization minimizes the distance between the given backbone curve and the distal end of each link of the module, so that the module resembles the given segment. In addition, as can be seen from Eq. (3), the algorithm is computationally demanding, and the computational load increases as the number of joints to be fitted increases.

It is important to note that using the fitting algorithm described above, the final position and orientation of the end-effector of each module cannot be controlled. Thus, to satisfy the imposed six constraints, an additional position control algorithm is required.

### 2.4 Position Control Algorithm

After running the fitting algorithm, we obtain a set of joint angles that allows the configuration of the module to best resemble the shape of the segment. However, the position and orientation constraints are not yet satisfied. To deal with this, we added position control to the control scheme. A Jacobian-based position control is used, as a Jacobian can be easily found in a systematic way, whereas an inverse kinematics solution is configuration-dependent and often

hard to obtain. In addition, by using a Jacobian, we can easily utilize redundancy, which is crucial in the proposed control scheme. Redundancy provides an infinite number of inverse kinematic solutions. Thus, among the infinite solutions that satisfy the six constraints, we search for the one that is the closest to the set of joint angles given by the fitting algorithm. In other words, we use the result from the fitting algorithm as the starting point and adjust it to meet the six constraints while minimizing the change at each joint, so that the overall posture of the robot can be preserved. This implies that each module must consist of more than six joints, and that each module is treated like a redundant robot.

A pseudo-inverse Jacobian has been widely used to control redundant robots. In order to utilize the redundancy to achieve a secondary goal, the gradient projection method and weighted least-norm have also usually been adopted. In this study, we used weighted least-norm, as it does not require gain tuning which has to be found by trial and error, and it minimizes the self-motion [9]. Assuming a hyper-redundant robot of  $n$ -DOF, which consists of modules of  $n_m$ -DOF, the weighted least-norm solution can be represented as:

$$\dot{q} = W^{-1}J^T(JW^{-1}J^T)^{-1}\dot{X}_d \quad (4)$$

where  $\dot{q}$  is the joint velocity,  $\dot{X}_d$  is the desired workspace velocity,  $W$  is the weighting matrix and  $J$  is the module Jacobian. This relationship can also be expressed as:

$$\Delta q = W^{-1}J^T(JW^{-1}J^T)^{-1}\Delta X \quad (5)$$

where  $\Delta X$  would be the error between the current position of the distal end of the module and the constraints in position and orientation. As stated in previous sections, each module controller is given the six constraints. The position part of  $\Delta X$  would simply be the position error between the distal end of the module and the end position of the segment. As for the orientation, unit quaternion is used in this study to avoid any representation singularities [21]. This allows hyper-redundant robots to move towards any arbitrary direction. Rotation matrix  $R_{i-1}^i$  will be the desired rotation matrix, and one can compute  $\Delta X$  accordingly.

The weighting matrix  $W$  can be expressed as:

$$W = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_n \end{bmatrix} \quad (6)$$

where  $w_i$ , the  $i$ -th element of the diagonal matrix  $W$ , is defined as

$$w_i = \begin{cases} 1 + \left| \frac{\partial H(q)}{\partial q_i} \right| & \text{if } \Delta \left| \frac{\partial H(q)}{\partial q_i} \right| \geq 0 \\ 1 & \text{if } \Delta \left| \frac{\partial H(q)}{\partial q_i} \right| < 0 \end{cases} \quad (7)$$

where  $H(q)$  is the desired performance criterion. The user can set up appropriate  $\partial H(q)/\partial q_i$  to achieve the desired secondary goal and it can be seen from Eq. (7) that if  $\partial H(q)/\partial q_i$  decreases,  $w_i$  will become 1 and there will be no null space motion. In this study, the goal is to minimize the deviation of the fitted angles from the fitting algorithm, and thus, the objective function is set to:

$$\frac{\partial H(q)}{\partial q_i} = (q_i - q_{i,fit})^2 \quad (8)$$

where  $q_{i,fit}$  is the result from the fitting algorithm. As can be seen from Eq. (8), the function is equal to zero when the angle is at the fitted position, and it increases rapidly as the joint moves away from the fitted position. Thus, as a joint angle  $q_i$  deviates from the  $q_{i,fit}$ , this function will cause a reduction in the motion. With this redundancy utilization, we can satisfy the constraints in the position and the orientation while keeping the configuration of the module as close as possible to the fitted result.

As we use the fitted solution as the starting point, it is important to obtain a good starting point. As mentioned above, to obtain a good fitting result, one must fit several joints at the same time (increase  $k$  in Eq. (2)). This would increase the computational load. However, with the modular control scheme, the burden of each controller is lessened and thus, the controllers can fit multiple joints at the same time, which would not have been possible using a single controller.

As stated in previous sections, if a hyper-redundant robot is to perform a task it must be able to place its end-effector accurately at the desired location. This can be easily accomplished using the proposed position control method shown in Eq. (5). It would allow the user to control the last module to move the end-effector while minimizing the change in the overall configuration of the robot. Thus, the proposed control scheme not only reduces the computational load, but it also increases the efficiency of the robot.

It should be noted that the proposed algorithm is not limited to a certain joint configuration, and the corresponding Jacobian for any hyper-redundant robot can be easily solved to apply the proposed control scheme. Thus, the proposed algorithm is highly flexible and can be applied to any hyper-redundant robot.

### 3. Simulation

#### 3.1 Robot

A 24-DOF robot with alternating yaw and pitch joints was simulated using MATLAB (MathWorks, Inc., Natick, MA). We chose the configuration with the alternating joints since this configuration is widely used in many hyper-redundant robots, such as [22,23]. To enable the proposed modular control, the robot was divided into three modules of 8-DOF each, and a module is shown in Figure 3. The length of each link is set to 1.5 unit length, and  $\epsilon$  in Eq. (1) is set to 0.05.  $q_{min}$  and  $q_{max}$  are set to -70 and 70 degrees, respectively, and (for optimization) the angle increments by 5 degrees. For a better result, we fitted two joints at the same time.

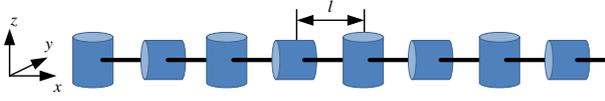


Figure 3. Joint configuration of a module

#### 3.2 Simulation Conditions

Two Bezier curves were used as the desired backbone curve for the simulation. Bezier curves are often used to simulate a backbone curve for a hyper-redundant robot, and they can be described as [14]:

$$p_d(t) = \sum_{r=0}^6 p_{dk} \frac{6!}{r!(6-r)!} t^r (1-t)^{6-r} \quad (9)$$

with seven control points,  $P_{dk}$  and  $M$  is equal to 6.

The simulation was conducted using a single PC, although the proposed scheme involves the use of multiple controllers. This was due to the technical difficulties in performing simulations using multiple PCs. The simulation first plays the role of the main controller to process the given backbone curve. Then, the simulation simulates the module controllers to control each module. This cannot be done in parallel, and the module controllers are simulated one by one. However, throughout the simulation, all the controllers were treated independently. By using this method, it can be shown that if all the outcomes from the module controllers are put together, the robot will resemble the given backbone curve. The reduction of the computational load will be discussed in a later section.

#### 3.3 Result 1: 2-Dimensional Curve

The first curvature is given in the 2-dimensional plane, and the control points for the Bezier curve are given in Eq. (10). The resulting curve is illustrated in Figure 4 (a), and the corresponding robot is shown in Figure 4 (b). Note that the end point of each segment is marked with a large circle,

while yaw and pitch joints of the robot are illustrated by solid squares and hollow circles, respectively.

$$P_{dk}(r) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 8 \\ 6 \\ 0 \end{bmatrix}, \begin{bmatrix} 12 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 12 \\ -6 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ -12 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ -18 \\ 0 \end{bmatrix}, \begin{bmatrix} 12 \\ -24 \\ 0 \end{bmatrix} \quad (10)$$

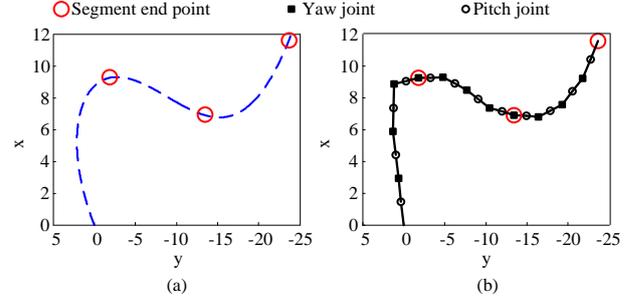


Figure 4. Results of 2-dimensional curve: (a) given desired curve, and (b) resulting robot configuration

#### 3.4 Result 2: 3-Dimensional Curve

The second curvature is given in 3-dimensional space as can be noted from Eq. (11). The results are illustrated in Figure 5.

$$P_{dk}(r) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 12 \\ -6 \\ 0 \end{bmatrix}, \begin{bmatrix} 16 \\ -12 \\ 0 \end{bmatrix}, \begin{bmatrix} 8 \\ -18 \\ 2 \end{bmatrix}, \begin{bmatrix} 8 \\ -24 \\ 2 \end{bmatrix}, \begin{bmatrix} 12 \\ -18 \\ 4 \end{bmatrix}, \begin{bmatrix} 22 \\ -12 \\ 6 \end{bmatrix} \quad (11)$$

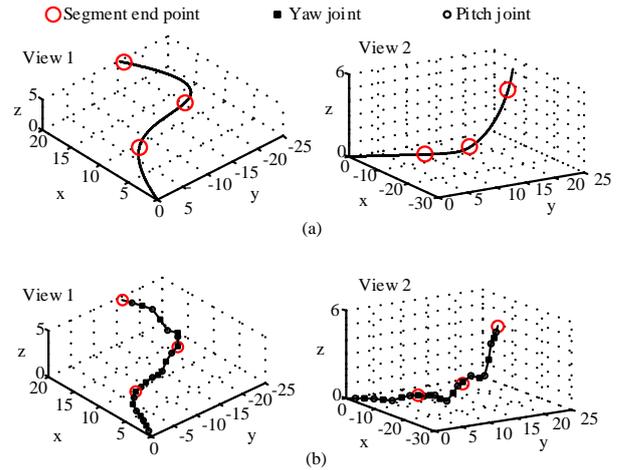


Figure 5. Results of 3-dimensional curve: (a) given desired curve, and (b) resulting robot configuration

The given desired curve is shown in Figure 4 (a) and 5 (a), and the resulting robot configurations are shown in Figure 4 (b) and 5 (b). As can be seen from the results, the control scheme was able to pick up most of the geometric features of the given curve. It should be noted that the result is limited by the dexterity of the robot. Although the config-

uration used in this study is widely used for hyper-redundant robots, it does not offer high dexterity compared to other configurations, such as robots with universal joints. Thus, although the resulting configuration resembles the given curve, it may show a “zigzag” pattern due to the joint configuration of the robot. This pattern was also reported in other studies, such as [14]. Reducing the link length,  $l$ , or increasing the dexterity of the robot would further improve the outcome of the control scheme.

The proposed control scheme treats each module as an independent redundant robot and, thus, many issues in controlling robots can be resolved using the solutions proposed for redundant robots. For example, joint limits and kinematic singularities often limit the operation of a robot. However, using the proposed control scheme, the operator may impose additional Jacobian-based redundancy resolutions on the robot to effectively avoid joint limits and singularities at the cost of a heavier computational load, as the use of multiple redundancy resolutions may require an increase in the number of redundant joints.

### 3.5 Analysis of Computational Load Reduction

The fitting algorithm can be regarded as the iterative computation of the end position of a link. Given the DH notation of a module, the end position of link  $i$  can be easily found using the forward kinematics computation as shown in Eq. (12):

$$A_i^0 = A_{i-1}^0 A_i^{i-1} \quad (12)$$

where  $A_i^{i-1}$  is a 4-by-4 transformation matrix between frame  $i$  and  $i-1$ . Note that this multiplication requires 64 operations. To fit a single joint, this multiplication must be done for all possible joint angles. By denoting  $n_q$  as the number of possible joint angles between  $q_{min}$  and  $q_{max}$ , the total number of operations required to fit a single joint is  $64n_q$ . By denoting  $k$  as the number of joints to be fitted at the same time and generalizing the result given above, it can be concluded that the required number of operation to fit a module is:

$$\frac{c(64n_q)^k}{k} \quad (13)$$

Normally,  $c$  is the total number of DOF of a hyper-redundant robot. However, using the proposed modular control scheme,  $c$  can be reduced to the  $1/m$  of the value without the modular scheme, thus reducing the computational load to  $1/m$ . The reduction may seem less apparent in a hyper-redundant robot with a small number of DOF. However, the proposed scheme would be particularly helpful for a hyper-redundant robot with a large number of DOF, for which a single controller cannot handle the fitting algo-

rithm in real-time. Note that this analysis focused on the computation of the end position of links as it requires the heaviest computation. Other necessary computations, such as the Jacobian, can be easily computed without much computational burden.

## 4. Conclusion

In this study, we proposed a novel modular control scheme for hyper-redundant robots. Through the use of modularization, redundancy resolution and a fitting algorithm, the control of a hyper-redundant robot in three-dimensional space was achieved. The performance of the scheme is verified through simulations. From this study, the following conclusions were drawn:

1. The proposed scheme reduces the computational load of hyper-redundant robots by enabling parallel computation. The robot is divided into modules, and each module can be fitted separately.
2. The proposed scheme is applicable to any hyper-redundant robot. The algorithm is not restricted to a joint configuration and is applicable for the curve given in both two- and three-dimensional spaces.
3. A weight least-norm solution that preserves the configuration of the robot is proposed. This can be used to enable parallel computation. Furthermore, it can be used to control the end-effector position of a hyper-redundant robot to perform various tasks.

## 5. Acknowledgements

The authors disclosed receipt of the following financial support for the research, authorship and/or publication of this article: This study was supported by grants from the National Research Foundation of Korea (NRF-2014024875) and the National Cancer Center of Korea (NCC-1210170).

## 6. References

- [1] Siciliano B, Khatib O. Handbook of Robotics. Berlin: Springer; 2008. 246 p.
- [2] Degani A, Choset H, Wolf A, Zenati MA. Highly articulated robotic probe for minimally invasive surgery. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA); 15-19 May 2006; Orlando, Florida; 2006. p. 4167-72
- [3] Slatkin AB, Burdick J, Grundfest W. The development of a robotic endoscope. In: Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robotics and Systems (IROS). 5-9 August 1995; Pittsburgh, PA; 1995. p. 162-71
- [4] Huang X, Abdalbari A, Ren J, 3D surface reconstruction of stereo endoscopic images for minimally invasive surgery. Biomedical Engineering Letters. 2013;3:149-157.

- [5] Low KH. *Industrial Robotics: Programming, Simulation and Applications*, Mammendorf: Pro Literatur Verlag; 2006. 633-662 p.
- [6] Wakimoto S, Nakajima J, Takata M, Kanda T, Suzumori K. A micro snake-like robot for small pipe inspection. In: *Proceedings of the International Symposium on Micromechatronics and Human Science*. 19-22 October 2003; Nagoya, Japan; 2003. p. 303-308
- [7] Maciejewski AA, Klein CA. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *International Journal of Robotics Research*. 1985;4:109-17.
- [8] Liegeois A. Automatic supervisory control of the configuration and behaviour of multibody mechanisms. *IEEE Transactions on Systems, Man and Cybernetics*. 1977;7:868-71.
- [9] Chan TF, Dubey RV. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*. 1995;11:286-92.
- [10] Buss SR Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least square methods. *IEEE Journal of Robotics and Automation*. 2004;17:1-19.
- [11] Chirikjian GS, Burdick JW. Modal approach to hyper-redundant manipulator kinematics. *IEEE Transactions on robotics and automation*. 1994;10:343-54.
- [12] Mochiyama H, Kobayashi H. The shape Jacobian of a manipulator with hyper degree of freedom. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 10-15 May 1999; Detroit, MI; 1999. p. 2837-42.
- [13] Tully S, Kantor G, Zenati MA, Choset H. Shape estimation for image guided surgery with a highly articulated snake robot. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 25-30 September 2011; San Francisco, CA; 2011. p. 1353-8.
- [14] Andersson SB. Discrete approximations to continuous curves. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 15-19 May 2006; Orlando, FL; 2006. p. 2546-51.
- [15] Hatton RL and Choset H. Generating gaits for snake robots by annealed chain fitting and keyframe wave extraction. *Autonomous Robots*. 2010;28:271-81.
- [16] Fahimi F, Ashrafiuon H, Nataraj C. Inverse kinematic solution for universal-jointed hyper-redundant robots. *IEEE Transactions of Robotics and Automation*. 2002;18:103-7.
- [17] Chirikjian GS, Burdick JW. Parallel formulation of the inverse kinematics of modular hyper-redundant manipulators. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 9-11 April 1991; Sacramento, CA; 1991. p. 708-13.
- [18] Lipkin K, Brown I, Peck A, Choset H, Rembisz J, Gianfortoni P, Naaktgeboren A. Differentiable and piecewise differentiable gaits for snake robots. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct 29-Nov 2 2007; San Diego, CA; 2007. p. 1864-9.
- [19] Tesch M, Lipkin K, Brown I, Hatton R, Peck A, Rembisz J, Choset H. Parameterized and scripted gaits for modular snake robots. *Advanced Robotics*. 2009;23:1131-58.
- [20] Chen IM, Yeo SH, Gao Y. Locomotive gait generation for inchworm-like robots using finite state approach. *Robotica*. 2001;19:535-42.
- [21] Sciavicco L, Siciliano B. *Modelling and control of robot manipulators*. Berlin: Springer;2000. 35-37 p.
- [22] Wright C, Buchan A, Brown B, Geist J, Schwerin M, Rollinson D, Tesch M, Choset H. Design and architecture of the unified modular snake robot. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 14-18 May 2012; Saint Paul, MN; 2012. p. 4347-54.
- [23] Hirose S, Mori M. Biologically inspired snake-like robots. In: *Proceedings of the IEEE International Conference on Robotics and Biomimetics*. 22-26 August 2004; Shenyang; 2004. p. 1-7.