Encircled Hyper-Redundant Manipulation Using Virtual Constraint, Bezier Curve and Gradient Descent Method

Chatklaw Jareanpon

Abstract—**The hyper-redundant robot can be used in object manipulation task. Due to the complexity of the mechanism, the object manipulation task can be reduced into encircling the robot around an object. The robot is required to be formed into some shapes to cage around an object and then moves an object. This method presented in this paper combine the caging process which is the shape control solution for hyper-redundant arm using the virtual constraint together with the virtual push force and the manipulating process which is the inverse kinematics solution for hyper-redundant arm using the Neural Network and Bezier curve together with the gradient descent method. The algorithm allows the robot to be able to encircle and move the object to the desired position without grasping. A computer simulation of the serial link manipulator has demonstrated the effectiveness of the proposed method.**

Index Terms—**Hyper-redundant robot, neural network, bezier curve, gradient descent method**

I. Introduction

Highly redundant manipulators or *hyper degree of freedom (*HDOF*)* has more degrees of freedom (DOF). A HDOF manipulator can perform many kinds of locomotion like the nature snake or the animal's tentacle to avoid obstacles, follow designated trajectories and manipulation tasks, such as moving in non-convenient environments, and pushing and caging a various sizes and shapes of objects. Due to all-in-one arms, a HDOF manipulator significant enhances the caging method as it, allows caging to perform in a variety of configuration. However, this arm must always maintain a certain shape around an object.

HDOF has been used by several researchers for solving control problems such as kinematic modeling [1], path planning [2], inverse kinematics [3]-[4] locomotive gait design [5], obstacle avoidance [6], and serpentine locomotion control [7] and sidewinding locomotion control [8] problems.

In our work, we study the shape control of a highly kinematic structure, called a HDOF arm manipulator. The HDOF is composed of serial chain links $l_i, i=1,....,N$, connected to other with revolute joints $j_i, i=0,...,N-1$. Each link is a straight rigid part of length L. The link l_1 and link l_N are called the base and the tail, respectively. The angle θ_1 is defined as the angle between link l_1 and x-axis. The set of angle defines the manipulator configuration as shown in Figure 1.

Manuscript received April 19, 2012; revised May 25, 2012

C. Jareanpon is with the Department of Computer Science, Faculty of Informatics, Mahasarakham University, Thailand (e-mail: chatklaw.j@gmail.com).

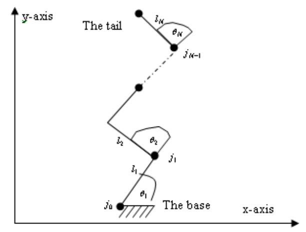


Fig. 1. A hyper degree of freedom (HDOF) structure.

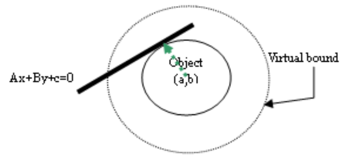


Fig. 2. Virtual push force.

In this paper, the main idea of the motion planning problem is: to find a path from an initial robot configuration to a final configuration, aiming to move an object from its initial position to the desired position. The motion will consist of a sequence of steps. Each step involves moving all joint angles while preserving motion continuity. The angle of a robot body must less than the angle limit to prevent the distortion.

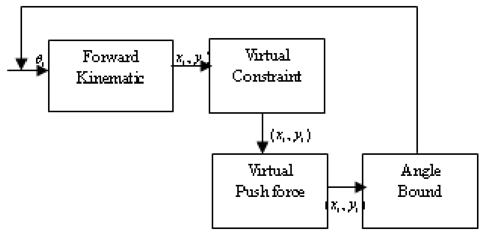


Fig. 3. Caging process diagram.

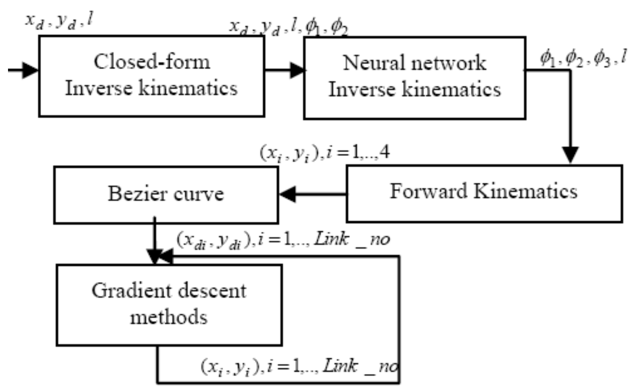

x_d, y_d, l
Closed-form Inverse kinematics
$x_d, y_d, l, \phi_1, \phi_2$
Neural network Inverse kinematics
$\phi_1, \phi_2, \phi_3, l$
$(x_i, y_i), i=1,..,4$
Bezier curve
Forward Kinematics
$(x_{di}, y_{di}), i=1,.., Link\_no$
Gradient descent methods
$(x_i, y_i), i=1,.., Link\_no$

Fig. 4. Manipulating process diagram.

II. Components of the Algorithm

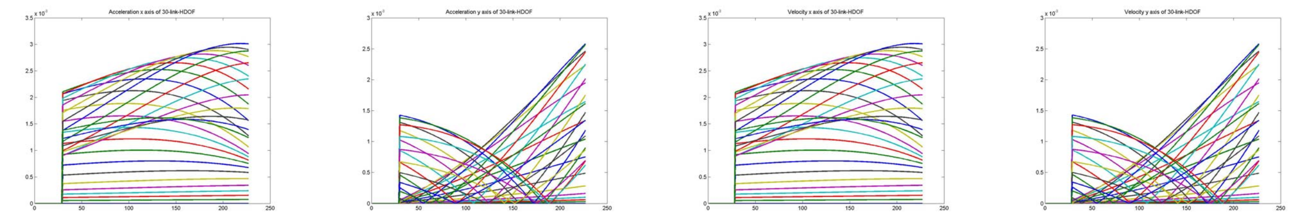
The manipulation process is divided into 2 parts: caging process that encircles around object and manipulating process that moves object from an initial position to the destination.

A. Caging Process

Before caging process, the first problem is how to control the robot to surround an object. We imitate the characteristic of human hugs. The definitions of human hugs are

"to put your arms around something" and "to fit tightly around something"

Based on this definition, we create the algorithm which consists of Virtual constraint to move the robot gait from an initial gait to a surrounding gait. In each step, the algorithm checks the collision-free with Virtual push force and avoids the distorted shape with an angle bound.



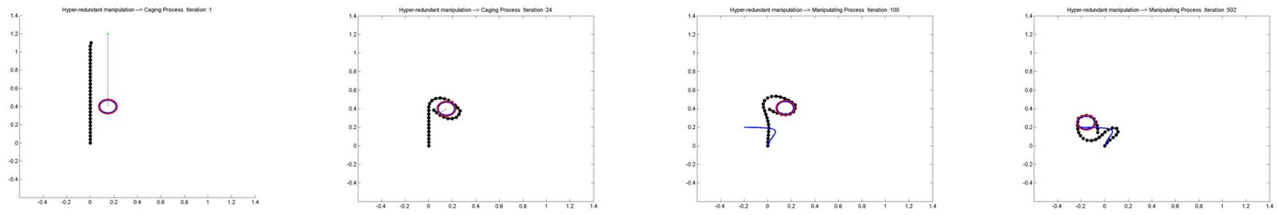
Velocity x axis of 30-link-HDOF

Velocity y axis of 30-link-HDOF

Acceleration x axis of 30-link-HDOF

Acceleration y axis of 30-link-HDOF

Fig. 5. Velocity and Acceleration of 30-link-HDOF from Manipulating process incase (-0.2,0.2) desired position.



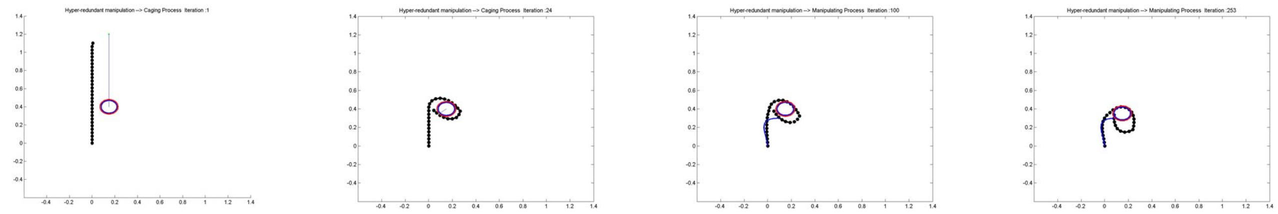
Iteration 1

Iteration 24

Iteration 100

Iteration 501

Fig. 6. Simulation of this algorithm in case (-0.2,0.2) desired position.



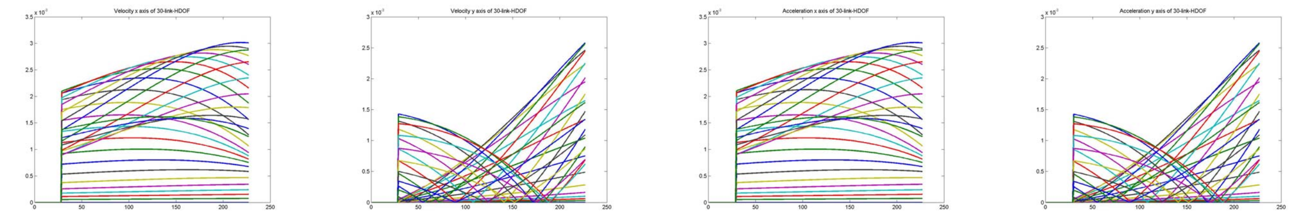
Iteration 1

Iteration 24

Iteration 100

Iteration 253

Fig. 7. Simulation of this algorithm in case (0.1,0.3) desired position



Velocity x axis of 30-link-HDOF

Velocity y axis of 30-link-HDOF

Acceleration x axis of 30-link-HDOF

Acceleration y axis of 30-link-HDOF

Fig. 8. Velocity and Acceleration of 30-link-HDOF from Manipulating process incase (0.1,0.3) desired position.

B. Virtual Constraint with Control Points

Virtual constraint is the concept proposed in [9],[10] and it is a method that guarantees collision-free push force and no-distortion of joint angle generates the control function to satisfy all constraint functions added to serve some specific purposes such as stabilizing the system or maintaining the distance from an obstacle. The Virtual constraint is used in this application in order to create a virtual force that follows the control point so that the HDOF robot arm can take a rough shape that encircles a specified object. The virtual force is calculated in proportion to the distance between the end-effector position and the control point that slowly moves clockwise or counterclockwise around the object. The Virtual constraint with control points has 6 steps:

1) Update the control point position according to the control function.

2) Calculate the error vector between the control point position $(x,y)_c$ and the end-effector position

$$(x,y)_{end-effector}$$

$$e = (x,y)_c - (x,y)_{end-effector} \tag{1}$$

3) Update joint angles according to (2) and (3).

$$\theta_n^{t+1} = f((x,y)_n, k \times e)) \qquad ,k = 0.45 \tag{2}$$

$$\theta_i^{t+1} = \theta_{i+1}^t \qquad ,i = n-1...1 \tag{3}$$

4) Move the end-effector $(x,y)_{end-effector}$ to its new position $p_n(\Theta)^{t+1}$.

5) Update the position of all the joints and links.

6) Repeat step 1) – 5).

C. Collision-free with Virtual push force

After using the Virtual constraint with control points, we consider the collision-free problem for all robot links. The concept of collision-free is based on the Geometry concepts. From Figure 2, when the link attacks to the object, the virtual force pushes the link by perpendicular force (dot line).

The Virtual push force can be explained in 3 steps:

1) Calculate the distance between the center of the object and the link position.

$$D = \frac{|Aa * Bb * c|}{\sqrt{a^2 + b^2}} \tag{4}$$

2) If distance D is less than the radius of the object plus the virtual bound (from Figure 3, the Virtual bound is represented by a doted circle) then calculate the push force.

$$dis\_a = \sqrt{(y_i - y_{i-1})^2} \;; \tag{5}$$

$$(6)$$

$$dis\_b = \sqrt{(x_i - x_{i-1})^2} \;; \tag{7}$$

$$dis\_c = \sqrt{(dis\_a)^2 - (dis\_b)^2} \;;$$

$$h = (r - D) + k \;; \tag{8}$$

$$\theta = \tan\left(\frac{dis\_a}{dis\_b}\right) + \tan\left(\frac{h}{dis\_c}\right). \tag{9}$$

where r = redial of object ; k = constraint force to push the object, and x_i, y_i = x and y position at $link_i$.

3) Update all the joint angles according to (9)

D. Angle bound

The virtual constraint pulls the HDOF robot to surround the object; however, the virtual constraint may cause the distorted angle. In addition, all angles should be of the same angle (i.e. forming a circle). Another is the circle which usually consists of equal angle. The virtual constraint pulls the HDOF robot, causing the configuration angle to change and this configuration angle is bounded by "Angle bound". If the angle 1 equals to angle bound, the algorithm will not update this angle. The algorithm repeats for the rest of the angles.

E. Manipulation Process

The manipulating process moves the object from initial position to given goal position. The inverse kinematics is to solve for the joint displacement when the end-effector position is given. The solution of the inverse kinematics of HDOF is difficult to find. The numerical method is usually employed to solve this problem to minimize the error between the current position and the desired position. However, if only the end-effector position is considered, the robot gait may be distorted and the solution from inverse kinematics may specify a heavy load for some joints. Therefore, the shape control must consider on all joint positions. The manipulating process uses the Neural network for generating the Bezier curve control point, uses the Bezier curve for generating the desired shape and uses the gradient descent method to find the joint angle.

F. The Bezier Curve Control Point Using the Neural network

The inverse kinematics is to solve for the joint displacement when the end-effector position is given. The HDOF inverse kinematics problem have multiple or infinite number of solutions. Moreover, the inverse kinematics equations are usually nonlinear and are difficult to find closed-form solutions. Differential methods are used for HDOF, it has recently been shown that the Neural network can solve the HDOF by optimization [22]. Using the back-propagation Neural network based inverse kinematics solution to control the HDOF manipulator's end-effectors to realize the tasks such as point to point and path following, standard has to calculate all the joint angles, to solve the inverse problem. It is complicated and used time consuming.

From required the control point position of parametric Bezier curve, we solve the inverse kinematics of 4 links planar serial link chain due to solve the three joint angles. By using the closed-form, it remains an unknown joint angle. In this propose, we solve the two joint angles with close-forms and solve the last joint angle with Neural network.

1) Closed-form 4-link inverse kinematics

From forward kinematics using Denavit-Hartenberg, the transformations relating the tail to the base link and a given end-effector orientation is given by (10), (11), respectively.

$${}^3_0T = \begin{pmatrix} \cos(\phi_1+\phi_2+\phi_3) & -\sin(\phi_1+\phi_2+\phi_3) & 0 & L_1\cos(\phi_1) + L_2\cos(\phi_1+\phi_2) \\ \sin(\phi_1+\phi_2+\phi_3) & \cos(\phi_1+\phi_2+\phi_3) & 0 & L_1\sin(\phi_1) + L_2\sin(\phi_1+\phi_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{10}$$

$${}^{tail}_{base}T = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 & x \\ \sin(\phi) & \cos(\phi) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

Equating the two previous expressions results in (12-17)

$$\cos(\phi) = \cos(\phi_1 + \phi_2 + \phi_3) \tag{12}$$

$$\sin(\phi) = \sin(\phi_1 + \phi_2 + \phi_3) \tag{13}$$

$$x = L_1 \cos(\phi_1) + L_2 \cos(\phi_1 + \phi_2) \tag{14}$$

$$y = L_1 \sin(\phi_1) + L_2 \sin(\phi_1 + \phi_2) \tag{15}$$

$$\cos(\phi_1 + \phi_2) = \cos(\phi_1)\cos(\phi_2) - \sin(\phi_1)\sin(\phi_2) \tag{16}$$

$$\sin(\phi_1 + \phi_2) = \cos(\phi_1)\sin(\phi_2) + \sin(\phi_1)\cos(\phi_2) \tag{17}$$

Squaring the expressions and adding them in (18), and Solving the $\cos(\phi_2)$ in (19), (20), (21).

$$x^2 + y^2 = L_1^2 + L_2^2 + 2L_1 L_2 \cos(\phi_2) \tag{18}$$

$$\cos(\phi_2) = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1 L_2} \tag{19}$$

$$\sin(\phi_2) = \pm\sqrt{1 - \cos(\phi_2)^2} \tag{20}$$

$$\phi_2 = \arctan(\sin(\phi_2), \cos(\phi_2)) \tag{21}$$

We can solved the theta(1) by (22-33)

$$x = p_1 \cos(\phi_1) - p_2 \text{ sin} \tag{22}$$

$$y = p_1 \sin(\phi_1) + p_2 \text{ co} \tag{23}$$

where (24)

$$p_1 = L_1 + L_2 \cos(\phi_2),$$

$$\vartheta = \sqrt{p_1^2 + p_2^2} \tag{25}$$

$$\delta = \arctan(p_2, p_1)$$

$$p_1 = \vartheta \cos(\delta) \tag{26}$$

$$p_2 = \vartheta \sin(\delta)$$

(27)

$$\frac{x}{\vartheta} = \cos(\delta)\cos(\phi_1) + \text{s}$$

(28)

$$\frac{y}{\vartheta} = \cos(\delta)\cos(\phi_1) + \text{s} \tag{29}$$

or

$$\cos(\delta + \phi_1) = \frac{x}{\vartheta} \tag{30}$$

$$\sin(\delta + \phi_1) = \frac{y}{\vartheta} \tag{31}$$

$$y + \phi_1 = \arctan(y, x) \tag{32}$$

$$\phi_1 = \arctan(y, x) - \text{arct} \tag{33}$$

However, the theta (3) is unknown. We will solve the theta (3) using the neural network.

2) *Neural Network inverse kinematics*

The basic back-propagation algorithm consists of three steps. The input pattern is presented to the input layer of the network. These are propagated through the network until they reach the output units. This forward pass produces the actual or predict output pattern. Because back-propagation is a supervised learning algorithm, the desired outputs are given as part of the training vector. The actual network outputs are subtracted from the desired output and an error is produces. This error is then the basis for the back-propagation step, whereby the error are passed back through the Neural network by computing the contribution of each hidden processing unit and deriving the corresponding adjustment needed to produce the correct output. The connection weights are then adjusted and the Neural network has just "learned" from an experience. In this propose, the Neural network structured is used as shown in Table I.

TABLE I: NEURAL NETWORK STRUCTURE

Input	[x_desired, y_desired, Theta1, Theta2]
Output	Theta3
Network structure	3 Layes [30:10:1]
Training sets	5000 records
Maximum epoch	500
Acceptance error	< 0.000002

TABLE II: SIMULATION RESULT

Desired position	Computational time	Caging Process iteration	Manipulation Process iteration	Total iteration	Manipulated gap
(-0.2,0.2)	409s	24	477	501	0.002
(0.1,0.3)	214s	24	229	253	0.002
Mean	311.5s	24	353	377	0.002

TABLE III: OBJECT CHARACTERISTIC

Object	Single
Central position	(0.15,0.4)
Radius	0.07
Bound	0.01

The desired position using the Bezier curve

In the mathematical field of numerical analysis, a Bezier curve is a parametric curve important in computer graphics and related fields. For HDOF robots, the Bezier is used for generating the desired shape. In this paper, the cubic Bezier is used. The four control points are calculated by the Neural network. The desired shape generated from the curve is equal to the HDOF joint. The Cubic Bezier is generated from (34)

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P \qquad ,t \in [0,1]. \tag{34}$$

where P_0 = the base position; P_1 = the average position from P_0 and P_4;

P_2 = the average position plus bias, and P_4 = the end-effector desired position.

The shape control using Gradient descent method

For HDOF robots, the gradient descent method is used to find the joint angles solution for shape control problem that satisfies the desired robot shape. The cost function for the HDOF robot shape control consists of two parts: the error function of the joint position and the smoothness constraint. The error function of the joint position requires the position of each joint coordinate $p(\Theta)$ to be close to the desired joint coordinate $p_d(\Theta)$ specified by the shape function. The smoothness constraint requires that the joint position relative to both neighbor link i+1 (x_{i+1}, y_{i+1}) and i-1 (x_{i-1}, y_{i-1}) to be rather straight (or smooth) as specified in the second term of (35).

The following cost function is used:

$$J=\sum_{i=1}^{n}((x_i,y_i)-(x_i,y_i)_d)^2+\alpha((x_{i+1},y_{i+1})+(x_{i-1},y_{i-1})-2(x_i,y_i)) \quad (35)$$

where (x,y) is the actual position and $(x,y)_d$ is the desired position of each link of the manipulator. The α is the weight of smoothness constraint [12].

From the Gradient descent method, the joint position can be updated from (36).

$$\Theta(k+1)=\Theta(k)-\eta\frac{\partial J}{\partial\Theta} \quad (36)$$

where Θ denoted the joint angle variables, and η is the step size. $\frac{\partial J}{\partial\Theta}$ can be calculated by:

$$\frac{\partial J}{\partial\Theta}=2[(x_i,y_i)-(x_i,y_i)_d]+\alpha[(x_{i+1},y_{i+1})+(x_{i-1},y_{i-1})-2(x_i,y_i)] +m \quad (37)$$

$$m=\alpha[(x_i,y_i)+(x_{i+2},y_{i+2})-2(x_{i+1},y_{i+1})]+ \alpha[(x_{i-2},y_{i-2})+(x_i,y_i)-2(x_{i-1},y_{i-1})] \quad (38)$$

The algorithm can be summarized as follows:

1) Given the desired position of each joint $(x,y)_d$ from Bezier curve.

2) Calculate the actual position of joint from the current joint angles and the value of the cost function J by using (35).

3) Calculate the new value of joint angles by using (36), (37) and (38).

4) Go to step 2, and repeat the process until the prescribed value of the cost function J is reached.

III. Simulation and Result

Computer simulation is used to test the performance of the proposed algorithm. The kinematics model of 30 DOF HDOF planar revolute robot arm in free space is built in MATLAB. The object is assumed into circle-shaped, light weight and contact less friction with the robot. The desired shape of the robot arm is designed for a non-prehensile pulling type of manipulation. The α in this experiment is set to 0.0000001. The experiment is tested on two cases that are different desired position. The desired position are (-0.2,0.2) and (0.3,0.1)

From the experiment, the caging process uses the 24 iterations. The virtual constraint is pulling the end-effector of robot to encircle around object while the virtual push force prevents to contact between robot and object. The average iteration of manipulating process is 353 iterations using control point form Neural network, the desired shape from Bezier curve and adjusts shape from Gradient descent method. From Figure 5 and 6, the velocity of x and y axis and the accelerations of x and y axis shows the smooth value and like trends therefore no joints is heavy load than other. The manipulation gap is less than the radius of object.

IV. Conclusion

In this paper, we developed a novel approach to control the shape of a HDOF robot based on concept of the virtual constraint and inverse kinematics problem based on the concept of the Bezier curve together with Gradient descent method.

We have simulated this approach on a 30 degree of freedom planar revolute manipulator. The caging process is able to encircle around object without contact with object while the manipulating process is able to solve the inverse kinematics. The Virtual constraint and Virtual push force is give the fast shape to encircle the object. The Bezier curve and Gradient descant method is give a distributed joint angle. Further study is planned for avoid the obstacle in space.

References

[1] S. Chirikjain and J. Burdisk, "The kinematics of Hyper-Redundant robot locomotion," *IEEE Transactions on robotics and automation.* vol. 11, no. 6, Dec. 1995.

[2] H. Poor, "*An Introduction to Signal Detection and Estimation*," New York: Springer-Verlag, 1985.

[3] H. Chang, "A closed-form solution for inverse-kinematic of robot manipulators with redundant," *IEEE journal of robotics and automation*,vol. ra-3, no. 5, Oct. 1987.

[4] Y. Li and H. Leong, "Kinematics control of redundant manipulators using a CMAC neural network combined with genetic algorithm," *Robotica*, vol. 22, pp. 611-621, 2004.

[5] G. Kulali and et al, "Intelligent gait synthesizer for serpentine robots," *Proc. of the 2002 IEEE international conference on robotics and automation*. pp. 1513-1518, 2002.

[6] S. Ma and M. konno, "An obstacle avoidance scheme for hyper-redundant manipulators – Global motion planning in posture space –," *Proc. of the 1997 IEEE international conference on robotics and automation*. pp. 161-166, 1997.

[7] J. Ostrowski and J. Burdick, "Gait kinematics for a serpentine robot," *Proc. of the 1996 IEEE international conference on robotics and automation*. pp. 1294-1299, 1996.

[8] J. Burdick, J. Radford, and S. Chirikjian, ""Sidewinding" locomotion gait for hyper-redundant robots,", *Proc. of the 1993 IEEE international conference on robotics and automation*., pp. 101-106, 1993

[9] T. Sugar and V. Kumar, "Multiple Cooperating Mobile Manipulators", *Proceeding of the 1999 IEEE International Conference on Robotics and Automation,* pp. 1538-1543, 1999.

[10] T. Maneewarn and P. Detudom, Mechanics of Cooperative Nonprehensile Pulling by Multiple Robots**,** *Proceedings of IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS2005)*, pp. 1319-1324, 2005.

[11] C. Jareanpon, S. Maneewongvatana and T. Maneewarn, Shape Control for Hyper-redundant Robot using Gradient Descent Method with Virtual Constraint. Procedding *the 13th International Conference on Advance*, 2007.