3D Hyper-Redundant Robot

Luís Marques[1], João Dinis[1]
A. Paulo Coimbra[1], Manuel M. Crisóstomo[1], João P. Ferreira[1,2]

[1]Institute of Systems and Robotics Electrical and Computer Engineering Department of University of Coimbra
Pinhal de Marrocos, Polo II, 3030 Coimbra, Portugal
Phone: +351 239 79 62 00, fax: +351 239 79 62 47, e-mail: luis.m.f.marques@gmail.com, dinisjmx@gmail.com, acoimbra@deec.uc.pt, mcris@deec.uc.pt

[2]Institute Superior of Engineering of Coimbra
Quinta da Nora, 3030 Coimbra, Portugal
Phone: +351 239 79 02 00, fax: +351 239 79 02 01, email: ferreira@mail.isec.pt

Abstract. There are many approaches to control robot manipulators. Many of them are based on the use of the inverse kinematics and the jacobian matrix. In order to obtain the inverse jacobian, in the cases when this matrix is not square, the pseudo inverse technique can be used. Some researchers suggest empirical approaches based on heuristic methods. In this paper the control of a hyper-redundant robot is analysed and some heuristic methods and the inverse jacobian method are compared. It is shown that the methods have different performances.

Key words

Robotics, hyper-redundant robot, control, pseudo-inverse jacobian, heuristic.

1. Introduction

Hyper-redundant robotic manipulators usually possesses a large number of joints. This characteristic increases the mobility of the manipulator, allowing it to reach positions not possible with normal manipulators, mainly when there are restrictions in the work space or when the manipulator looses one or more degrees of freedom.

Due to the great potential of this type of manipulators, research work has been extensively conducted since the sixties, where Hirose is one of the pioneers. Some of the developed robots are:

- ACM III (Locomotion Device), developed by Hirose [11];
- 30 DOF Planar Hyper-Redundant Manipulator, developed by Chirikjian and Burdick [7];
- Elephant Trunk Manipulator, developed by Hannan and Walker [10];
- The Modular Reconfigurable Robot, developed by Mark Yim [4];
- 3D Snake, developed by Kevin [13] (our robot has a similar structure);
- Serpentine Robot, developed by Howie Choset [9];
- Orochi, NEC, developed by Ikeda and Takanashi [5].

Existing hyper-redundant robots have a variety of forms and configurations. For example the ACM III robot was inspired in the snakes movement and possesses 20 joints servo actuated, with a total length of 2 metres; the Orochi robot uses Hooke joints, developed by Ikeda and Takanashi [5].

The planar hyper-redundant robot, developed by Chirikjian and Burdick, built to demonstrate and validate the hyper-redundant kinematics, have a structure with 30 joints, grouped in 10 modules of 3 joints. Each module has 3 prismatic joints, actuated by DC servo motors and endless screws [7].

There are several control techniques, developed by several authors. Salerno (in 1989) [2] and Chirikjian and Burdick [7] developed control methods based in the "dorsal spine curve". Naccarato and Hughes (1991) [2] compared this method with the traditional inverse kinematics approach.

Salerno (in 1993) [2] and Byers (in 1994) [2] solved the inverse kinematics with the pseudo-inverse Jacobian.

Shinichi proposed a decentralized autonomous algorithm which uses parallel processing. The control algorithm continues executing the positioning with a success higher than 90 %, even when half of the joints failed [3].

Vittor [12] implemented a decentralized controller for a robot with a modular system with 12 joints and 6 links. The simulation results show a smooth trajectory of the end-effector, while avoiding obstacles. The decentralized control method seems to be versatile and robust, using the redundancy and the flexibility of the manipulator.

In order to control a hyper-redundant robot designed at the Institute of Systems and Robotics of the Department

of Electrical and Computer Engineering of the University of Coimbra, in Portugal, an algorithm based in the pseudo-inverse Jacobian and another based in heuristic methods have been implemented. The two algorithms were compared based on execution time, trajectory, and precision.

2. Model of the Robot

The designed robot has 16 joints, assembled in 8 modules of 2, orthogonal to each other (fig. 1). Each joint can rotate between -90 and +90 degrees.

Using the Denavit-Hartenberg model, the **θ** (angle), **d** (eccentricity), **a** (length) and **α** (twist) parameters can be obtained (table 1) [1].

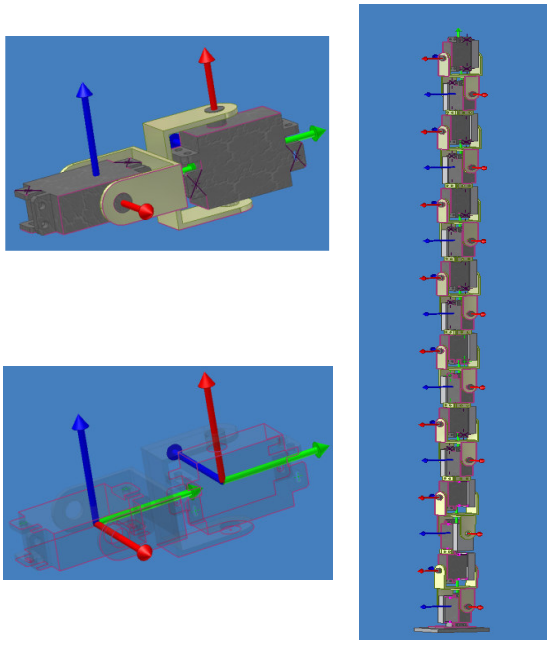

Fig. 1. 3D model of the implemented robot and details of joints and links.

TABLE I - Denavit-Hartenberg Parameters

Transformations	θ_i	α_i	d_i	a_i
1→2	θ_0	-90	0	L_1
2→3	θ_1	90	0	L_2
3→4	θ_2	-90	0	L_3
…	…	…	…	…
15→16	θ_{15}	-90	0	L_{15}

3. Control Algorithms

In this work the inverse kinematics problem is addressed in two different approaches:

- The Pseudo-Inverse Jacobian Method;
- A Heuristic Method.

A. *Pseudo-Inverse Jacobian Method*

This method makes use of the inverse of the jacobian matrix

$$D_q = J^{-1} \times D \tag{1}$$

where D_q represents the joints variation, D the displacement and J^{-1} the inverse jacobian matrix.

The jacobian, witch allows to linearly model the end-effector behavior, is the matrix of the first order partial derivative of the robot variables in joints space, relatively to the cartesian coordinates of the end-effector (px, py, pz).

$$J = \begin{bmatrix} \frac{\partial px}{\partial \theta_1} & \frac{\partial px}{\partial \theta_2} & \cdots & \frac{\partial px}{\partial \theta_n} \\ \frac{\partial py}{\partial \theta_1} & \frac{\partial py}{\partial \theta_2} & \cdots & \frac{\partial py}{\partial \theta_n} \\ \frac{\partial pz}{\partial \theta_1} & \frac{\partial pz}{\partial \theta_2} & \cdots & \frac{\partial pz}{\partial \theta_n} \end{bmatrix} \tag{2}$$

Due to the robot's joints redundancy the jacobian is a non square matrix as shown in (2) and becomes impossible to invert [6].

Due to the difficulties of the inversion of the jacobian, the pseudo-inverse jacobian is used, although it is not so precise as the inverse jacobian would be, if it exists.

Thus from (1) and replacing the inverse jacobian with the pseudo inverse jacobian, we obtain the solution of the inverse kinematics.

$$D_q = J^{\#} \times D \tag{3}$$

Introducing a second term

$$(I - J^{\#} J) \tag{4}$$

a more complete equation is obtained that allows to compensate the error of the pseudo-inverse jacobian and to control the movement accomplished by the robot [14].

$$D_q = J^{\#} \times D + \left(I - J^{\#} J\right) \times H \tag{5}$$

H is a constant column-vector obtained experimentally that controls the influence of the error correction term and the way the robot behaves during the motion. For H=0 the correction member is not used and (5) is the same as (3).

Fig. 2 illustrates the algorithm of the implemented method.

The end-effector displacement, in cartesian coordinates,

is discretized in K intervals and the final position will be reached in K iterations, with an approximately linear path.

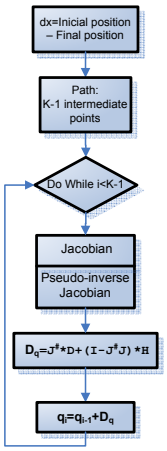

Fig. 2. Flowchart of the implemented method

The pseudo-inverse jacobian is calculated through (6) [15].

$$J^{\#} = J^{T} \times \left(J \times J^{T}\right)^{-1} \qquad (6)$$

The jacobian was obtained using Paul's algorithm instead of the method of partial derivation due to its smaller computation time as shown in table II [8],[15].

TABLE II - Processing time for the jacobian computation

Method	Time (s)
Partial Derivatives	0.141
Paul's Algorithm	0.015

The Pseudo-inverse Jacobian Method was tested using four different H vectors:

1. Null: H=0;
2. Symmetric: H = 0.00168 [1/6 1/4 1/4 1/2 1/2 1 1 1 1 1 1 1/2 1/2 1/4 1/4 1/6];
3. Constant: H = 0.00168 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
4. Decreasing H = 0.00168 [2.5 2.5 2 2 2 1.5 1.5 1.5 1 1 1 1/2 1/2 1/4 1/4 1/6].

The algorithms were implemented and tested using Matlab.

B. Results

Simulations were carried out using (-0.042, 0, 0.942) as the end-effector home position and (0.2, 0.2, 0.2) as its target, fulfilling a 0.8062 m displacement.

As expected, the processing time is not influenced by H and it is proportional to K.

Fig. 3 shows the variation of the end-effector final position error with K using the four vectors H.

The error curves resemble a negative exponential with the exception of the curve for H constant.

It was verified that the curves for H null and H symmetric match perfectly and are the ones that present the lowest error.

The H constant curve doesn't have the same behavior of the remaining ones presenting larger errors.

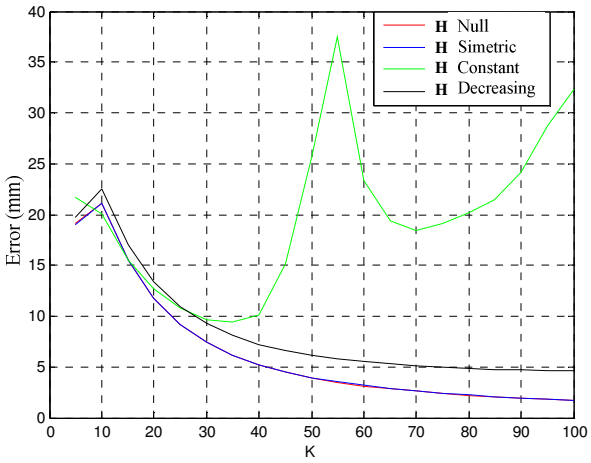

Fig. 3. Final end-effector position error vs K, for different H

Independently of the displacement and of H, the path is approximately linear (fig. 4). Exceptions occur close to a singularity, as it is the situation of the limit of the work area, due to the difficulty in obtaining the inverse pseudo jacobian. In this situation, the path is irregular in the beginning but as soon as a point out of the singularity is reached, it becomes a straight line.

The manipulator's configuration during the paths varies with H. For H null the manipulator begins to bend close to the base taking an S shape. For H symmetric (fig. 5), the manipulator bends in the middle, due to the symmetric configuration of the vector H.

For H constant (fig. 6) and for H decreasing the robot's tendency is to roll up on itself in a helical configuration.

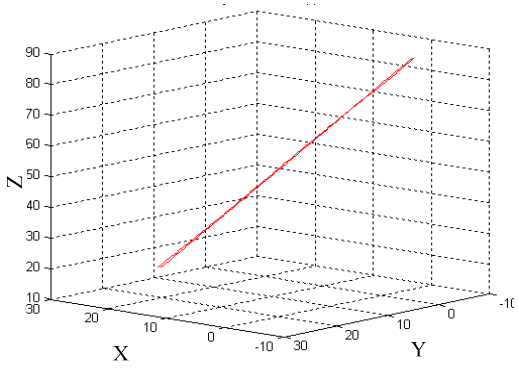

Fig. 4. Overlap of the paths for H constant with K=150 and for H decreasing with K=55.

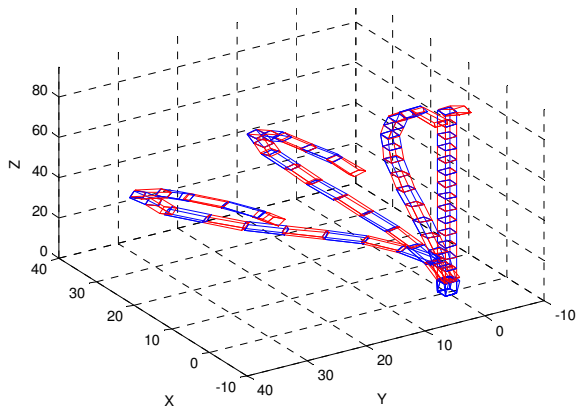

Fig. 5. Robot's simulation with H symmetric and K=50.

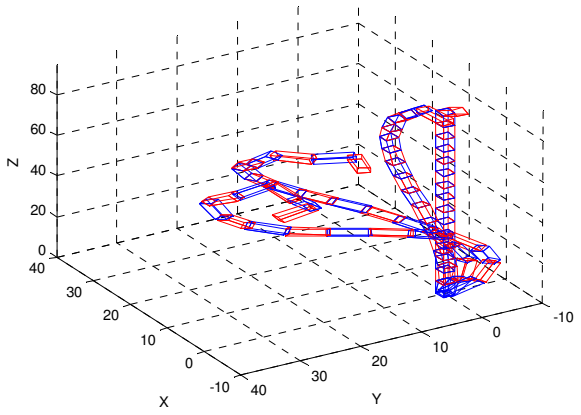

Fig. 6. Robot's simulation with H constant and K=50.

The H vector influences the way how the angles of the joints evolve. For H symmetric (fig. 7) some joints have a different variation from the other joints along the path unlike what happens with H constant (fig. 8) in which all the joints change more or less uniformly since the beginning of the path.

For H constant (fig. 8) some joints reach the physical limit (+90º); other, are changed in a redundant way, being the final position similar to the initial one. The reaching of the physical limit by some joints is one of the possible explanations for the curve with vector H constant, in fig. 3, do not present a negative exponential behavior.

For H constant and H decreasing the decrease of K reduces the variations of the angles of some of the joints and the manipulator reaches the goal with similar variations in the angles for all the joints.

For H null and H symmetric the decrease of K doesn't influence the variation of the joint angles.

H vector influences the movement type and the robot configuration. K influences the execution time and the final end-effector position error.

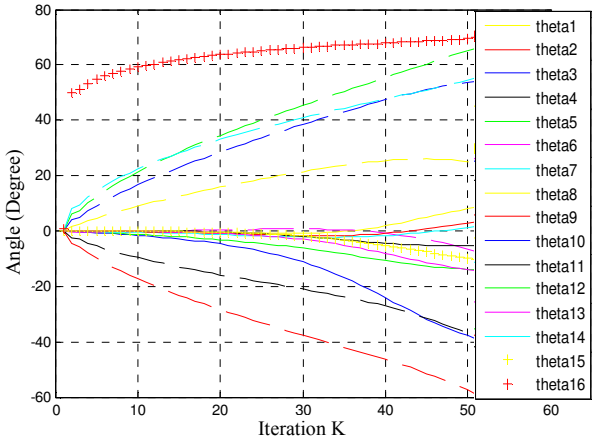

Fig. 7. Variation of the joint angles along the path for H symmetric and for K=50.

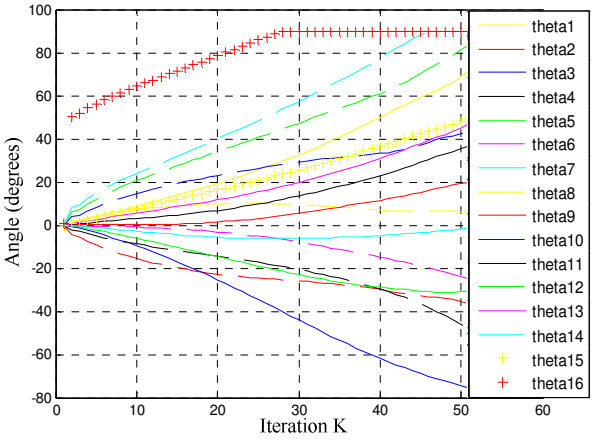

Fig. 8. Variation of the joint angles along the path for H constant and for K=50.

C. Heuristic Control Method

To solve the problem of the hyper-redundant robot's inverse kinematics a heuristic iterative method has been developed.

In this method, a small variation is applied to a joint in every iteration, until the position of the end-effector is the intended. In each iteration the direct kinematics is applied to determine the end-effector new position.

Which joint to move and how much in each iteration lead to different options of this algorithm. In this paper two of these variants are presented: a version that changes the joint that has larger influence in the approach of the end-effector to the goal (method 1), and another version that always starts moving first the joint closer to the end-effector (method 2).

In both of these methods, only after having obtained the final configuration, the joints of the robot are physically rotated between the initial and the final configurations.

D. Heuristic Method 1 - Joint with larger influence first

In this method, in each iteration, virtual configurations are computed for a ±Δθ degrees variation in all the joints. Then, the configuration with the end-effector closer to the goal is chosen, being executed the corresponding joint rotation.

The process is repeated while the end-effector isn't placed at the wanted point (goal) with the intended precision.

The algorithm can be described in the following way:

1. Computes the virtual positions of the end-effector, through the direct kinematics, if a ±Δθ degrees variation is applied to each of the 16 joints;
2. Computes the distance d_i between the goal and each one of the 32 virtual positions of the end-effector, computed in 1;
3. Determines which is the index i and the sense of rotation of the joint that minimizes the distance;
4. Executes in the virtual model the previous rotation and returns to 1 while the end-effector doesn't arrive to the goal.

To reduce the number of iterations the Δθ variation applied to each joint is not constant: Δθ initiates with a high value and is decreased gradually to obtain the wanted positioning precision.

Δθ begins equal to 20 degrees (found to be a good starting value) and when in an iteration the end-effector moves away from the goal, Δθ is diminished using (7).

$$\Delta\theta = \frac{\Delta\theta_{old}}{f_r} \tag{7}$$

f_r is a reduction factor, between 1.1 and 2.

Incorporating to the algorithm the joints restrictions and the considerations previously mentioned we have tested the robot's behavior with f_r =1.2, using the same home position (P_0) and goal position (P_1) as before (fig. 9).

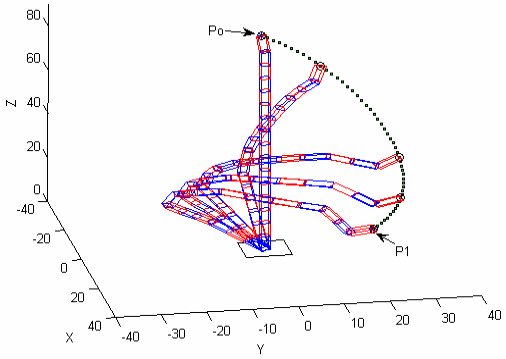

Fig. 9. Simulation and path with the heuristic method 1.

In this example the end-effector final position error was of 1.6×10^{-10} m. It should be noticed that the calculation time was of 1.2 seconds and it is directly related with the specified error.

E. Heuristic Method 2 - Last joints first

In this method, in each iteration, it is first rotated by ±Δθ degrees the joint closer to the end-effector. If none of these two rotations approaches the end-effector to the goal it is rotated the next joint (and so on).

When the rotation of ±Δθ degrees doesn't decrease the distance of the end-effector to the goal, then the value of Δθ is reduced according to (7).

This algorithm is partially described next:

1. Starts with i=16, corresponding to the joint closer to the end-effector and calculates the distance between the goal point and the end-effector virtual position;
2. Computes the virtual position of the end-effector applying direct kinematics for a ±Δθ variation applied to joint i;
3. Calculates the minimum of the two distances between the goal point and the end-effector virtual position;
4. If this distance is greater than the previous one, i is decremented and returns to 2;
5. If the position of the end-effector doesn't match the goal position returns to 1.

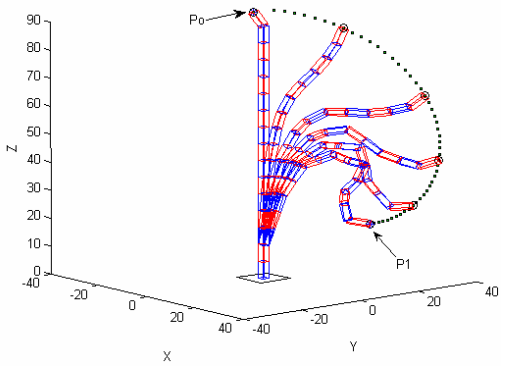

Fig. 10. Simulation and path with the heuristic method 2.

As shown in fig. 10 the robot rotates only the joints necessary to reach the goal, as it was expected.

The main difference between these two algorithms is the criteria to choose the rotating joint in each iteration. The final configurations are different but with the same final end-effector position.

4. Conclusions

There are several control techniques of redundant robots. In this paper it has been presented a traditional approach

consisting in the use of the pseudo inverse jacobian matrix and two new heuristic iterative methods.

In table III it is presented a comparison of the main features of the three implemented methods.

It is concluded that the pseudo-inverse jacobian approach is generally faster but less accurate than the heuristic methods.

The paths described by the end-effector in the heuristic methods tend to be an S curve and in the pseudo-inverse jacobian method are approximately a strait line.

The pseudo-inverse jacobian approach presents limitations in the case of singularities as it is the case of the work area limit. In these situations it is difficult to obtain the pseudo-inverse matrix and the algorithm has an undesired behavior but at the end of some iterations it moves away from the critical point and reaches the goal. The robot's final configuration depends on the configuration of the H vector. The discretization of the end-effector trajectory influences the execution time and the final end-effector position error.

The heuristic methods allow more exact solutions but the generated end-effector path is not a strait line.

The second heuristic method (last joints first) is faster than the fist heuristic method (joint with the larger influence first). These two heuristic methods present similar position errors.

TABLE III - Comparison of the main features of the implemented methods.

	Pseudo-Inverse Jacobian (H=0)		Heuristic Methods	
Displacement =0.80831 m	K=5	K=3000	Method 1	Method 2
Computation time (s)	0.12	47.01	0.74	0.27
End-effector position error (m)	19.01×10^{-3}	46×10^{-6}	2.6×10^{-6}	2.9×10^{-6}
End-effector trajectory	Linear		Curve, variable	
Final configuration	Depends on H		Depends of initial $\Delta\theta$ and of f_r	
Number of iterations	K		Variable	Variable

References

[1] John J. Craig, "Introduction to Robotics Mechanics and Control", Second Edition, 1990, cap.3 and 5.

[2] Robert L. Williams II, James B. Mayhew IV, "Obstacle-Free Control of The Hyper-Redundant Nasa Inspection Manipulator", Fifth National Conference on Applied Mechanisms and Robotics Cincinnati OH, October 12-15, 1997.

[3] Shinichi Kimura, Masato Takahashi, Toshiyuki Okuyama, Shigeru Tsuchiya, and Yoshiaki Suzuki, "A Fault-Tolerant Control Algorithm Having a Decentralized Autonomous Architecture for Space Hyper-Redundant Manipulators", IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, VOL. 28, Nº. 4, July 1998.

[4] Jackrit Suthakorn, "Binary Hyper-Redundant Robotic Manipulator Concept", Department of Mechanical Engineering, The Mahidol University, Salaya, Nakorn Patom 73170 Thailand, 2004.

[5] NEC Corporation, "Orochi 12DOF Snake Like Robot", Press Release, NEC Corp. Melville, NY, Jan, 1996, 6 pp.

[6] Mike Tabaczynski, "Jacobian Solutions to the Inverse Kinematics Problem" Tufts University, Math 128 Fall 2005 Final Project.

[7] Chirikjian, GS, and Burdick, JW, "Kinematically Optimal Hyper-Redundant Manipulator Configurations," IEEE Trans. on Robotics and Automation, 11(6), Dec 1995, pp. 123-136.

[8] Corke, P. O. (2002). Robotic Toolbox. Obtained from Peter O. Corke site: http://www.petercorke.com/Robotics%20Toolbox.html.

[9] http://www.snakerobot.com/.

[10] Hannan, MW, and Walker, ID, "The 'Elephant Trunk' Manipulator, Design and Implementation," Proc. of the 2001IEEE/ASME AIM, Italy, 2001.

[11] Hirose, S., "Biologically Inspired Robots: Snake-like Locomotors and Manipulators", Oxford University Press, 1993.

[12] T.Vittor, R.Willgoss and T.Furukawa, "Modular Decentralized Control of Fruit Picking Redundant Manipulator", Australian Conference on Robotics and Automation, Brisbane, December, 2003.

[13] Kevin J. Dowling, "Limbless Locomotion: Learning to Crawl with a Snake Robot", PhD Thesis, The Robotics Institute, Carnegie Mellon University, December 1997.

[14] M. Meredith & S. Maddock, "Using a Half-Jacobian for Real-Time Inverse Kinematics", International Conference on Computer Games: Artifitial Inteligence, Design and Education, November 2004.

[15] R. J. Schilling, "Fundamentals of Robotics: Analysis and Control", Prentice-Hall Inc., Englewood Cliffs, NJ, USA, 1990.