Parallel Processing Applied to Robot Manipulator Trajectory Planning

Denis Hamilton Nomiyama[1], André Riyuiti Hirakawa[2], Líria Matsumoto Sato[3]
Polytechnic School, University of São Paulo
158 Prof. Luciano Gualberto Ave, lane 3, São Paulo, Brazil
[1] {denis.nomiyama@poli.usp.br}
[2] {andre.hirakawa@poli.usp.br}
[3] {liria.sato@poli.usp.br}

Abstract

The parallel processing has been applied in real time systems which require intensive data processing. An application which requires high performance processing is that related to the control of the trajectory planning of a manipulator.

There are several methods for conducting the trajectory planning. One of them is the Variational method. This method consists of an iteractive process for executing the calculations related to the configuration of joints of a redundant manipulator. As in every iteractive procedure, the Variational method demands high volume of processing, causing a delay that wouldn't allow a real-time system to work, and also its simulations. To solve this problem, parallel processing can be used. This paper presents a parallel processing algorithm to solve the trajectory planning problem, based on redundant manipulator joints. In this case, the Variational method is applied with the use of a parallel architecture.

Keywords **Parallel Processing, Trajectory Planning, Manipulator, Robotics.**

I. INTRODUCTION

The kinematics of manipulators is usually represented by a Jacobian matrix [CRA89], [MAC94], as shown in the equation below (1):

$$\mathbf{x} \quad \mathbf{J} \tag{1}$$

Where $\mathbf{x}$ is an m-dimensional vector specifying the end-effector velocity, is an n-dimensional vector denoting the joint, and $\mathbf{J}$ is the m by n manipulator Jacobian matrix. When $n > m$, the manipulator is denominated redundant, and infinite combinations of joint positions can specify each position of end-effector.

Figure 1 shows a redundant manipulator with three joints that works in a two-dimensional workspace.

It can be seen that there are different configurations to access the same point in a two-dimensional workspace. The configurations illustrated are (P_1, P_2, P_3), (P_1, P_2', P_3') and (P_1, P_2'', P_1'), and there are infinite possible configurations.

The inverse kinematics can be used to make trajectory planning [STA96]. The Jacobian matrix in (1) must be moved to the left hand deriving the equation (2).

$$\mathbf{J}^{1} \quad \mathbf{x} \tag{2}$$

Where J^{-1} is an inverse Jacobian matrix.

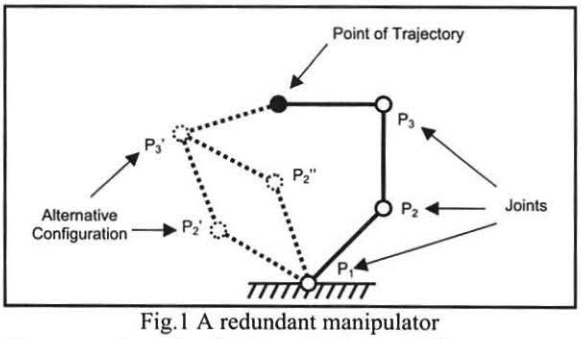

Fig.1 A redundant manipulator

However, in redundant manipulators, J is not a squared matrix, which makes difficult the inverse calculation. Usually, the pseudo-inverse matrix is used applying some criterion. This procedure is complex, and alternative methods can be used [STO87], [FIJ89], [MAC94].

An alternative solution to calculate the inverse kinematics of a redundant manipulator is to use the Variational approach [HIR97]. The Variational approach is an interactive method to determine joint variables, which allows to specify the position of the end-effector in a determined point of the workspace. The main problem of this approach is the high processing needed, which makes its use difficult for real time systems and simulations. To solve this problem, parallel processing can be used [SAT92], [NOM95].

The trajectory planning of industrial applications is generally done in offline calculation requiring an intensive data processing. One calculation of inverse kinematics is used several times because the movements of industrial robot manipulator are repetitive. However, when the trajectory must be changed, a new calculation is necessary. The parallel algorithm proposed can calculate the joint configurations in real time.

This paper presents a parallel algorithm, which uses Variational method in a parallel architecture. The section II presents the specification of the proposed algorithm, describes the Variational approach, and explains the task division between the processes. The section III describes

the structure of parallel architecture used. The section IV describes the system setup and tests done. The results and discussions are in the section V with runtime tables and performance charts. The last section has the conclusion of tests done to verify the performance of the parallel algorithm proposed.

II. Parallel Algorithm Proposed.

The Variational approach has an interactive process to calculate joint configurations.

These joint configurations allow the end-effector of manipulator to go through a determined trajectory.

The equation (3) describes the formulation of Variational method to solve the trajectory-planning problem.

$$E\{\ (t)\} \quad \min \int_{t_0}^{t_f} F\ t,\ ,\ ,\ dt \tag{3}$$

Where the function $F\ t,\ ,\ ,$ represents the optimization criterion, which depends on time (t) and joint variables $,\ ,\ .$ The integrand of equation (3) is called performance index, which can be defined in two functions.

$$E \quad \min \int_{t_0}^{t_f} f(\ ,t) \quad (t)\ g\ ,\ ,\ ,t\ dt \tag{4}$$

Where $f(\ ,t)$ is the main performance index, $g\ ,\ ,\ ,t$ is the sub-performance index, and (t) is a weight parameter.

The main performance index $f(\ ,t)$ is a positional error function of end-effector between the reference and the actual position in the workspace coordinate. Its definition is given by equation (5)

$$f(\ ,t) \quad x_{ref}(t) \quad x(\ t)\ \ x_{ref}(t) \quad x(\ t) \tag{5}$$

Where $\mathbf{x}_{ref}(t)$ is the reference coordinates of trajectory, and $\mathbf{x}(\ ,t)$ is the actual position in the workspace.

The most used sub-performance indexes are position, velocity, acceleration and potential energy. However, in a simplified formulation, the sub-performance indexes can be omitted. Equations (6) and (7) show the calculus of the variation in the method.

$$\frac{E}{\ } \int_{t_0}^{t_f} f(\ ,t)\ dt \tag{6}$$

$E/\ $ is a vector with the differential of error function. The equation (7) has a numerical solution of (4) using (6) and the steepest gradient method.

$$_{i} \quad _{i\ 1} \quad K \frac{E}{\ } \quad \text{For each iteration i.} \tag{7}$$

Where $_{i}$ and $_{i\ 1}$ are vectors which have the i and i-1 configurations, K is a diagonal matrix which is chosen to allow the convergence of interactive process.

The K value affects directly the result of Variational process. For a fast calculation with fewer iterations, the K value must be high. However, with very high values, the result of equation (7) may diverge.

The figure (2) illustrates a trajectory, reference points and approximation points.

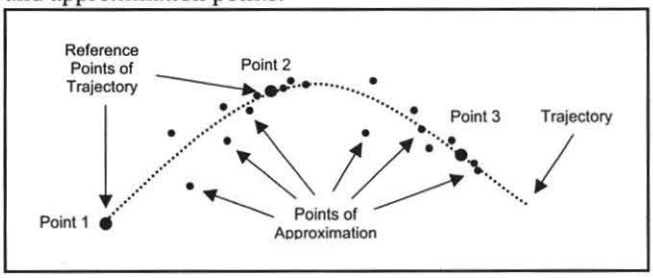

Fig.2 Points used in trajectory planning

The algorithm proposed for the trajectory-planning problem produces a sequence of joint variables, which allows the end-effector to go through a determined trajectory.

The entry data are Cartesian coordinates of reference points. Then, the calculation produces a list of joint variables for all points of trajectory.

The processing is divided between the allocated processes. Each process calculates a subset of reference points.

The points of trajectory are calculated at the same time because the obtainment of points is performed with exclusiveness by single parallel processes. When a process finishes the calculation of a point, a new point is allocated.

For each point, the algorithm performs consecutive approximations while the distance between a reference point and the actual calculated point is higher than the maximum error.

At the beginning of this interactive process, the distance between the reference point and the actual calculated point is the highest. Then, step-by-step, the distance decreases after each iteration.

The figure (3) shows two processes collaborating to do the calculation. The process 1 makes a series of approximations to get the second reference point (point 2), and the process 2 tries to get the third point (point 3).

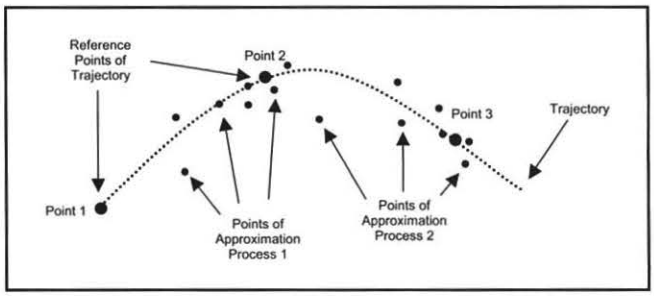

Fig.3 Points of approximation in trajectory planning

The K value in equation (7) must initially be the highest possible and it decreases while the distance reduces. The

motivation for this variation is to obtain a fast approximation.

However, the variation of K factor cannot be determined previously, because it depends on trajectory. The parallel algorithm reduces the K value when the distance between reference point and actual point increase during the approximation processes. This event can happen when the error is higher than the step of approximation. In this case, the point of approximation overtakes the reference point. The K factor decreases by a relevant distance between points. The list (1) shows the parallel algorithm using pseudo-code.

1. Get reference points of trajectory
2. Create processes, and execute in each process:
   2.1. While there is a reference point to be calculated, repeat:
      2.1.1. While the distance between the actual point calculated and the reference point is higher then the maximum error, repeat:
         2.1.1.1. Calculate the partial derivates $\frac{\partial E}{\partial w}$
         2.1.1.2. Calculate the values of angles $K.\frac{\partial E}{\partial w}$
         2.1.1.3. Calculate the values of coordinates x_{actual} and y_{actual} using the vector
         2.1.1.4. What is the actual point, which is the nearest of reference point
         2.1.1.5. Calculate the distance between actual point and reference point
         $E = |x_{atual} - x_{ref}|$
         2.1.1.6. If error increases
            2.1.1.6.1 Decrease K value
      2.1.2. Get next reference point
3. Store the calculated joint variables

List 1 Algorithm parallel for redundant manipulator

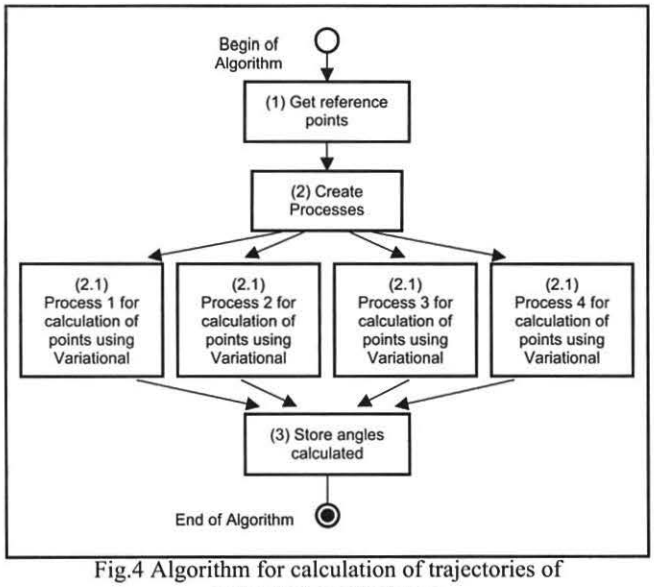

Fig.4 Algorithm for calculation of trajectories of redundant manipulators

Figure (4) and figure (5) describe situations and main decisions that happen during the execution of parallel algorithm described in list (1). To help the comprehension, the items 2.1.1 and 2.1.2 are encapsulated in the item 2.1 and illustrated in figure (5).

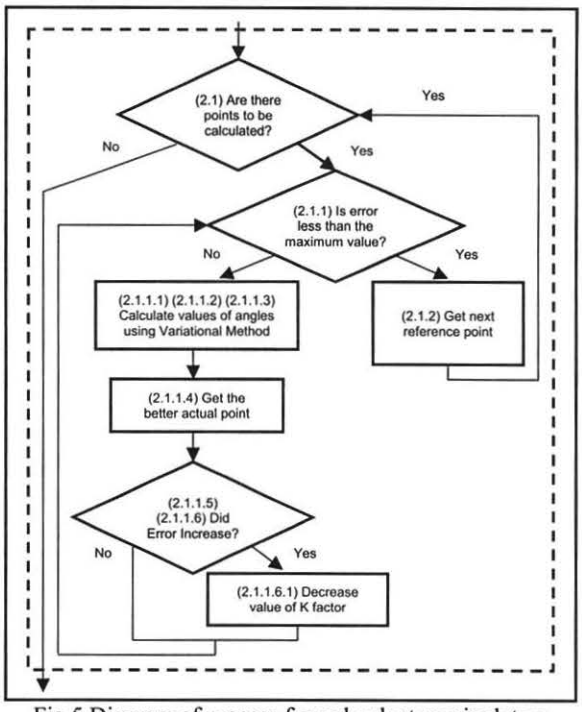

Fig.5 Diagram of process for redundant manipulators

III. Description of Parallel Architecture Used

The model of parallel architecture is a MIMD multiprocessor with four microprocessors connected in the same motherboard and using the same data and instruction bus.

Thus, all processors share the same RAM memory and other installed resources like hard disk, video monitor, keyboard, mouse and network interface.

The diagram in figure (6) describes this configuration.

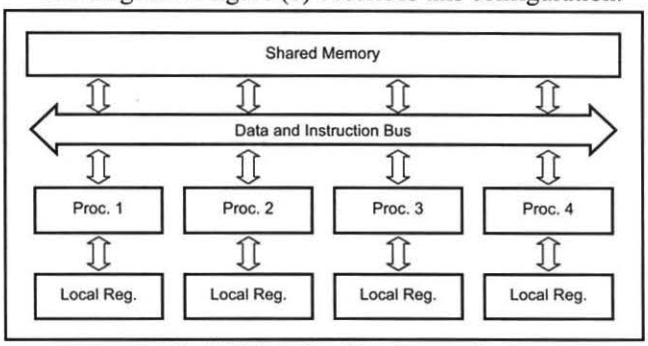

Fig.6 Model of architecture used

This type of architecture can be used for general purpose. Thus, the parallel architecture proposed applies for the majority of applications.

IV. System Setup

A. Multiprocessor Feature

The computer used is a multiprocessor with four Intel Pentium Pro microprocessors running at 200 MHz, one megabyte of cache memory in each microprocessor and 256 megabytes of shared RAM.

The operating system used was Linux version 2.0.30. The algorithm proposed was implemented using the Cpar programming language [SAT92].

B. Model of Manipulator

The model of redundant manipulator used has three joints with angular movement, and it has a two-dimensional workspace, as shown in figure (7).

The length of each segment (d) is 500mm.

The joints have 360° revolution. Thus, the end-effector can access points contained in a semicircle with radius 1500mm and center in the origin of Cartesian system.

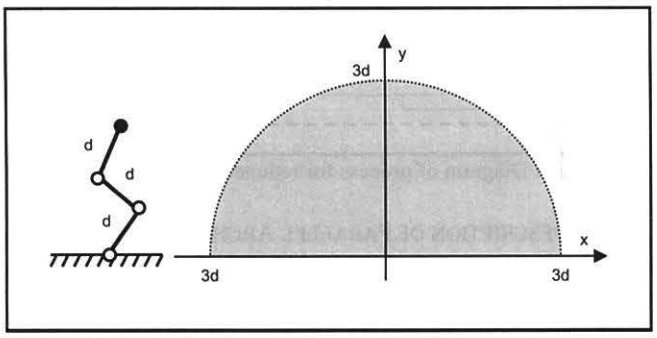

Fig.7 Model and workspace of redundant manipulator

The joints configuration of rotational manipulator is specified using angles, as illustrated in figure (8).

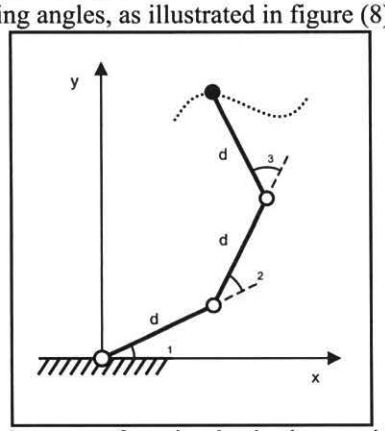

Fig.8 Movement of rotational redundant manipulator

The direct kinematics can be defined using the Denavit and Hartenberg matrix that uses homogenous coordinates [ASA86], [FU87], [STO87]. With this matrix, the position of end-effector (x, y) can be calculated using joint variables ($_1$, $_2$ and $_3$). Equation (8).

$$\begin{bmatrix} x \\ y \\ z \\ \vdots \end{bmatrix} = \mathbf{H}_n \begin{bmatrix} x_n \\ y_n \\ z_n \\ \vdots \end{bmatrix}, \quad \begin{bmatrix} x_n \\ y_n \\ z_n \\ \vdots \end{bmatrix} = \mathbf{H}_{i\ 1,i} \begin{bmatrix} x_n \\ y_n \\ z_n \\ \vdots \end{bmatrix} \tag{8}$$

Where: $\mathbf{H}_{i\ 1,i} = \begin{bmatrix} \cos & \sin & 0 & d_i \cos \\ \sin & \cos & 0 & d_i \sin \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

n is the number of joints

For n=3: $\mathbf{H}_3 \quad \mathbf{H}_{0,1}\ \mathbf{H}_{1,2}\ \mathbf{H}_{2,3}$ (9)

$$\mathbf{H}_3 = \begin{bmatrix} \cos & \sin & 0 & d_1 \cos & d_2 \cos & d_3 \cos \\ \sin & \cos & 0 & d_1 \sin & d_2 \sin & d_3 \sin \\ 0 & 0 & 1 & & 0 & \\ 0 & 0 & 0 & & 1 & \end{bmatrix} \tag{10}$$

Where $_{12}$

The end-effector coordinate of manipulator matches with the origin of last coordinate system, then:

$$\begin{bmatrix} x_n \\ y_n \\ z_n \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{11}$$

Thus, the direct kinematics of manipulator can be defined as:

$$\begin{bmatrix} x \\ y \\ z \\ \vdots \end{bmatrix} = \begin{bmatrix} \cos & \sin & 0 & d_1 \cos & d_2 \cos & d_3 \cos \\ \sin & \cos & 0 & d_1 \sin & d_2 \sin & d_3 \sin \\ 0 & 0 & 1 & & 0 & \\ 0 & 0 & 0 & & 1 & \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{12}$$

$$x \quad d_1 \cos \quad d_2 \cos \quad d_3 \cos \tag{13}$$

$$y \quad d_1 \sin \quad d_2 \sin \quad d_3 \sin \tag{14}$$

The equations (13) and (14) are used to define $E\{\ (t)\}$ in equation (7).

C. Performance Analysis Metrics and Techniques

There are different techniques and metrics to do performance analysis [KAN89], [JAI91]. Performance analysis for systems with multiprocessors generally uses the speed up and efficiency index [JAJ92], [DEC89]. Their definitions are in equation (15) and (16)

$$SpeedUp \quad \frac{T_s}{T_p} \tag{15}$$

$$Efficiency \quad \frac{T_1}{n\ T_p} \tag{16}$$

Where, T_s is the runtime to execute the best sequential algorithm. T_p is the runtime to execute the parallel algorithm using p processors. n is the quantity of processors.

Speed Up is the ratio of performance between the best sequential algorithm and the parallel algorithm using p processors. *Efficiency* is the ratio of utilization of processors when the parallel algorithm is executed with more than one processor.

V. Experiments

The experiments were done with three different trajectories, which have the following characteristics:

- Square with four segments of same length and orthogonal to x-axe and y-axe. The length of each segment is 250mm and center at coordinate x = 375mm and y = 500mm. Figure (9).
- Lozenge with four segments of same length and transversal to x-axe and y-axe. The length of each segment is 353mm and center at coordinate x = 250mm and y = 500mm. Figure (10).
- Circumference of radius 200mm and center at coordinate x = 300mm and y = 500mm. Figure (11).

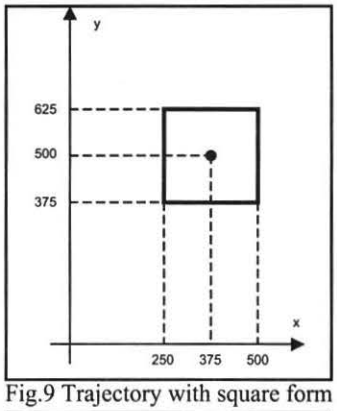

Fig.9 Trajectory with square form

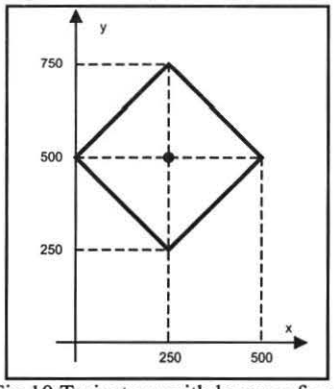

Fig.10 Trajectory with lozenge form

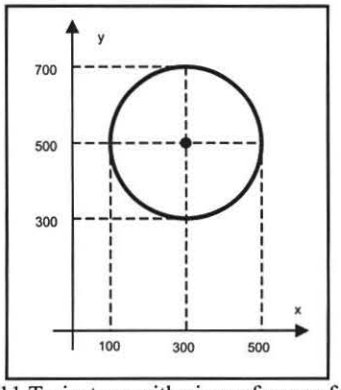

Fig.11 Trajectory with circumference form

The three trajectories have 1000 reference points, which are organized in anticlockwise order, and the initial point is at coordinate (500, 500).

The initial values of joint variables are (90°, -30°, -120°). Figure (12).

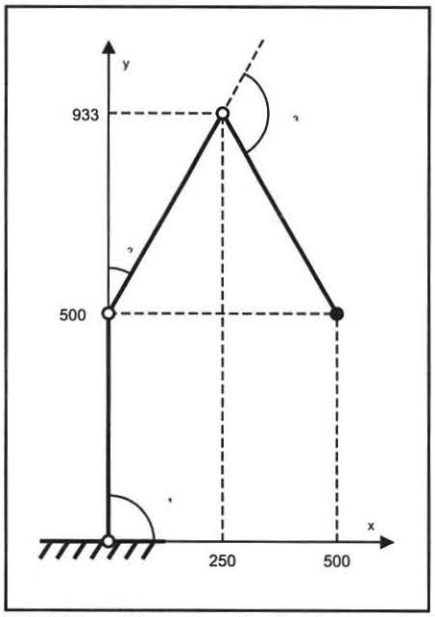

Fig.12 Initial configuration of redundant manipulator

VI. Results and Discussions

As described in section II, the K value decreases when the distance between the reference point and actual point increase. To obtain the best initial K value, the parallel algorithm was executed with different ranges of K values. The initial K value was changed from 10^{-7} to 10^{-1}, and the runtime obtained is described on table (1) and figure (13).

TABLE I
Result of K variation

K Value	Runtime (s)	Iteractions
0.0000001	39.591645	11312173
0.0000010	5.607276	1556972
0.0000100	1.770620	475771
0.0001000	1.781677	473525
0.0010000	2.679360	721290
0.0100000	3.197007	888495
0.1000000	3.794507	1058256

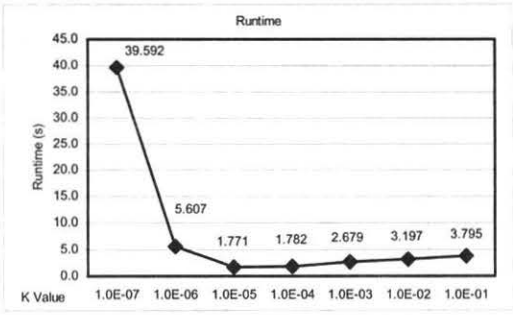

Fig.13 Time for K variation

The table (1) illustrates that the best initial K value is 10^{-5}.

To obtain the best decrement factor, the parallel algorithm was executed varying the factor from 2.0 to 5.0 in each step. The runtime obtained is described on table (2) and figure (14).

TABLE II
RESULT OF VARIATION OF DECREMENT FACTOR OF K

Factor Value	Runtime	Iterations
2.5	1.949138	515262
3.0	1.735694	463516
3.5	1.692396	456315
4.0	1.770886	485912
4.5	1.825743	494129
5.0	1.881447	513256
5.5	2.140153	564328

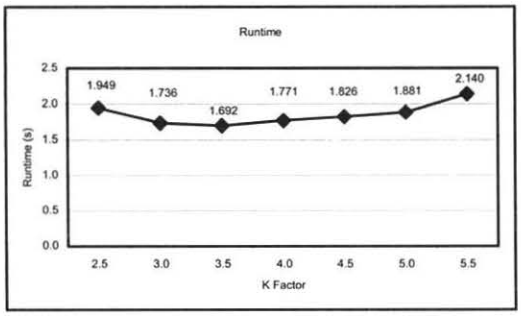

Fig.14 Time for variation of decrement factor of K

The table (2) illustrates that the best value of the decrement factor is 3.5. Thus, the K value has to decrease 3.5 times, when the distance between the reference point and actual point increases during the interactive process.

Using the obtained range of K values and decrement factor of K, the parallel algorithm was executed using the three trajectories described on section IV.

The joint configurations calculated using the parallel algorithm are illustrated in figures (15), (16) and (17).

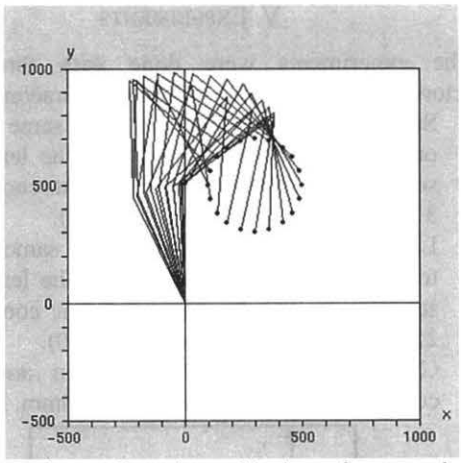

Fig.15 Joint configurations with circumference trajectory

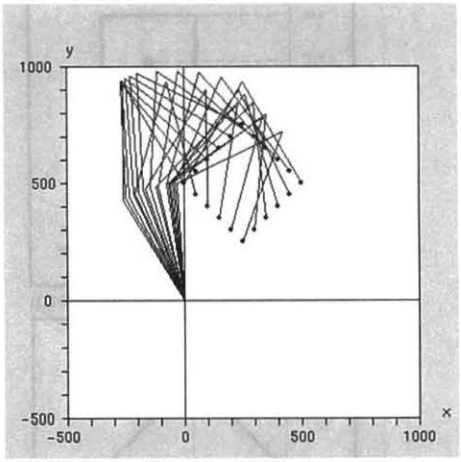

Fig.16 Joint configurations with lozenge trajectory

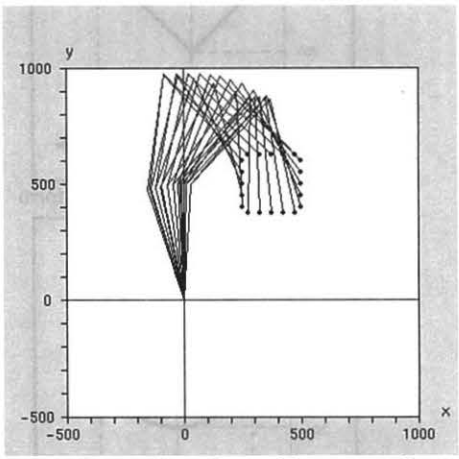

Fig.17 Joint configurations with square trajectory

The runtime is shown in table (3).

TABLE III
RUNTIME OF SERIAL AND PARALLEL ALGORITHMS

Trajectory		Serial	1	2	3	4
Square	Runtime	0.410148	0.422566	0.262434	0.196172	0.166699
	Speed Up		0.970613	1.562862	2.090757	2.460411
	Efficiency			0.805090	0.718020	0.633726
Lozenge	Runtime	0.399995	0.410373	0.278132	0.209027	0.181598
	Speed Up		0.974711	1.438148	1.913604	2.202640
	Efficiency			0.737731	0.654418	0.564947
Circumference	Runtime	0.401886	0.412573	0.263856	0.202245	0.176175
	Speed Up		0.974097	1.523126	1.987125	2.281175
	Efficiency			0.781815	0.679989	0.585459

The values of table (3) are illustrated in figures (18) to (26).

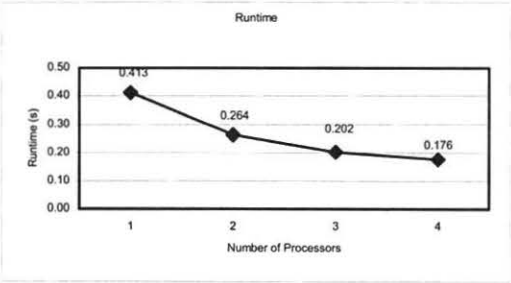

Fig.18 Runtime using circumference trajectory

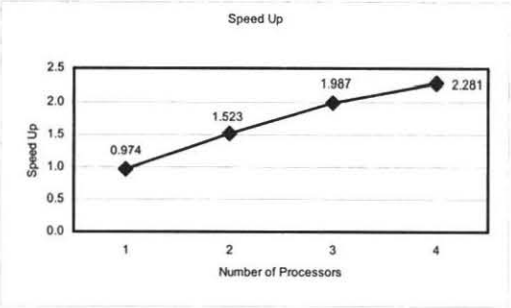

Fig.19 Speed up using circumference trajectory

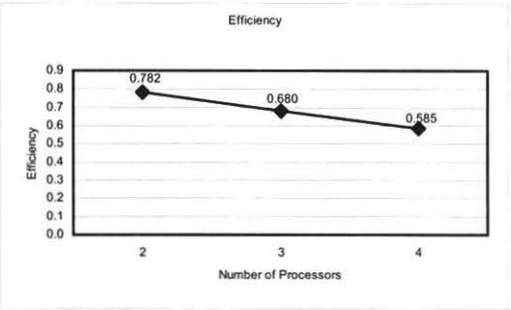

Fig.20 Efficiency using circumference trajectory

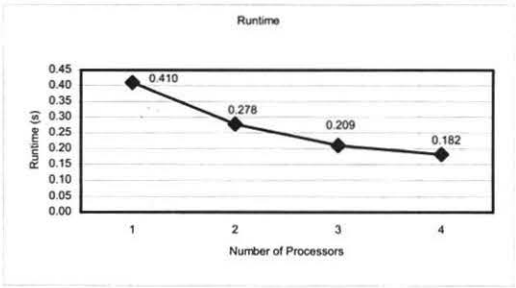

Fig.21 Runtime using lozenge trajectory

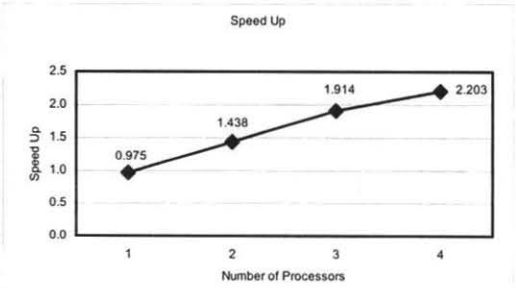

Fig.22 Speed up using lozenge trajectory

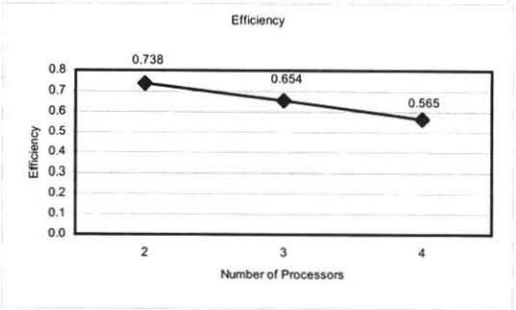

Fig.23 Efficiency using lozenge trajectory

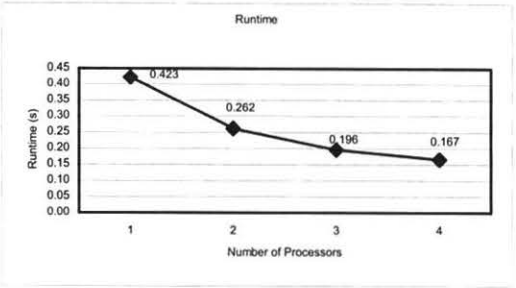

Fig.24 Runtime using square trajectory

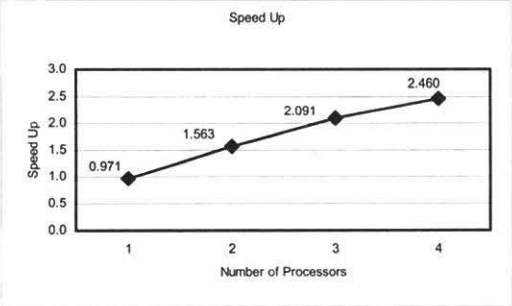

Fig.25 Speed up using square trajectory

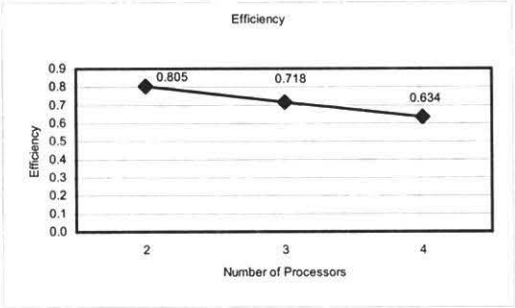

Fig.26 Efficiency using square trajectory

VII. Conclusion

In this study, a new parallel algorithm for trajectory planning of redundant manipulators was proposed to reduce the runtime. The algorithm is based on Variational approach without the use of an inverse Jacobian matrix.

The algorithm has high interaction between the allocated processes, thus the efficiency decreases when the number of processors is increased.

The experiments on a parallel architecture confirm the expected performance.

The maximum time of response of manipulators must be between 10ms and 100ms, which corresponds the elapsed time in the robot work cycle. The runtime obtained in the used robot manipulator is between 0,17s and 0,18s for a trajectory with 1000 points. Thus, the time of each point is between 1,7ms and 1,8ms.

The runtime, speed up and efficiency obtained validate using the Variational method to solve the trajectory planning problem of redundant manipulators.

VIII. Future Works

The future works include the use of more processors to evaluate the speed up and efficiency behavior, and new implementations using manipulators with more joints which demand more volume of processing.

References

[HIR 97] HIRAKAWA, A.R. Study on Trajectory Generation for Redundant Manipulators Using Variational Approach. *Dissertation (Doctoral)*. Yokohama National University, Yokohama, 1997.

[SAT 92] SATO, L.M. A System of Programming and Processing to Multiprocessing Systems. *IV SBAC-PAD*, São Paulo, 1992

[STA 96] STADLER, W. Analytical Robotics and Mechatronics, *Mc Graw Hill*, 1995.

[JAJ 92] JÁJÁ, J. An Introduction to Parallel Algorithms. University of Maryland, 1992

[JAI 91] JAIN, R. The Art of Computer Systems Performance Analysis. *Wiley Professional Computing*, 1991.

[NOM 95] NOMIYAMA, D.H.; ZORZO, S.D.; AKAMATU, D.M. Concurrent Computing: Concepts and Programming Languages. Federal University of São Carlos, São Carlos, 1995

[CRA 89] CRAIG, J. Introduction to robotics: mechanics and control, 1989

[ASA 86] ASADA, H. Robot Analysis And Control, 1986

[FU 87] FU, K.S. Robotics: Control, Sensing, Vision, And Intelligence, 1987

[STO 87] STONE, H.W. Kinematic Modeling, Identification, And Control Of Robotic Manipulators, 1987

[FIJ 89] FIJANY, A.; BEJCZY, A.K. A Class Of Parallel Algorithms For Computation Of The Manipulator Inertia Matrix, *IEEE Transactions on Robotics and Automation*, 1989

[MAC 94] MACIEJEWSKI, A.A.; REAGIN, J.M. A Parallel Algorithm And Architecture For The Control Of Kinematically Redundant Manipulators, *IEEE Transactions on Robotics and Automation*, 1994

[KAN 89] KANT, K. Introduction to Computer System Performance Evaluation, *McGraw-Hill*, 1992

[DEC 89] DECEGAMA, A.L. Parallel Processing Architectures and VLSI Hardware, *Prentice-Hall*, 1989