Cartesian Coordinate Control for Redundant Modular Robots

*Weihai Chen *I-Ming Chen *Wee Kiat Lim *Guilin Yang

Gintic Institute of Manufacturing Technology
71 Nanyang Drive, Singapore 638075

* School of Mechanical & Production Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

whchen@gintic.gov.sg michen@ntu.edu.sg glyang@gintic.gov.sg

Abstract

This paper is focused on the kinematic control of redundant modular robots for trajectory tracing. Based on the geometric numerical inverse kinematic algorithm developed for modular robots, a new online control method is presented. In this method, the inverse kinematic solution can be optimized through constructing a weighted matrix. Following this approach, some fundamental interpolation algorithms are proposed for Cartesian space (task space) control of redundant modular robots. The effectiveness of the proposed algorithms has been experimentally demonstrated by a 7-DOF serial modular robot that performs pick-and-place task with the avoidance of joint angle limits.

Key words: modular robots, redundant robots, kinematic control

1 Introduction

Because the number of degrees of freedom (DOF) in the joint space is larger than that in the task space, a redundant robot has high dexterity. A robot is redundant also implies that it has the fault-tolerant capability. For those reasons, redundant robots are highly suitable for the unstructured and hazardous environments where the dexterity and reliability become the major concerns [1, 2, 3]. A conventional redundant robot, however, has a fixed configuration that somehow limits its working capability, such as the workspace and the manipulability.

A modular robot consists of a collection of standard modules: actuators, rigid links, and end-effectors. These components can be easily assembled into various robot configurations having different working capabilities [4, 5]. Although one can build a robot with fewer modules and DOFs to perform a simple task [6], it is not always the case. When the task and the working environment are complicated, redundant modular robots should be the optimal choice [7].

Usually, robot redundancy can be employed or resolved either at the joint velocity level or at the joint acceleration level [3]. Such an algorithm, however, is difficult to obtain high control accuracy. Moreover, unlike the conventional robots which have fixed configurations, the modular robots can assume any possible DOFs and configurations. In such case, manual derivation of the closed-form inverse kinematic solution is basically impractical and impossible for modular robots. The generic numerical approach, therefore, has been used to get the inverse kinematic solutions of modular robots [8].

In this paper, we first make some modifications on the iterative inverse kinematic algorithm proposed in [8] to improve the computational efficiency. Then, based on the modified algorithm, a new online redundancy control approach is developed (at joint angle level) through constructing an object-oriented weighted matrix. This new control algorithm, in many ways, is equivalent to the joint velocity control method, but has higher control accuracy. Hence, this approach can also be used as a basis to carry out the kinematic control for fault-tolerant redundant robots. Furthermore, some fundamental interpolation algorithms for straight line, circle, screw and ellipse motions have been developed for Cartesian space control. Finally, a pick-and-place demonstration with the avoidance of joint angle limits for a 7-DOF serial robot is used to verify these algorithms. This 7-DOF robot together with a 6-DOF parallel robot and a slider that consists of some smart modules form an entire modular robot workcell (see Fig. 5).
2 Numerical Inverse Kinematics

This section briefly introduces the numerical inverse kinematic algorithm developed for modular robots [8]. This generic algorithm is suitable for modular robots with various configurations regardless of the number of degrees of freedom and types of joints. Now, consider a serial robot that consists of n+1 links (from 0 to n) connected through n joints (from 1 to n). Let link i-1 and link i be two adjacent links connected by joints i. Link i and joint i are termed as link assembly i. If we denote the body coordinate frame on link assembly i by frame i, then the relative pose of frame i wrt frame i-1, under a joint displacement, q_i, can be described by a 4 x 4 homogeneous matrix, such that

T_{i-1,i}(q_i) = T_{i-1,i}(0)e^{q_i},

where T_{i-1,i}(0) ∈ SE(3) is the initial pose of frame i wrt frame i-1, q_i ∈ R is the joint variable, and g_i ∈ se(3) is the twist of joint axis i expressed in frame i. It has the form of

\[ \begin{bmatrix} 0 & -\omega_i^x & \omega_i^y \\ \omega_i^x & 0 & -\omega_i^z \\ -\omega_i^y & \omega_i^z & 0 \end{bmatrix} \]

Here, \( \omega_i \in \mathfrak{so}(3) \) is a skew-symmetric matrix related to \( \omega_i \in \mathbb{R}^3 \) which is the unit directional vector of the joint axis i expressed in frame i; \( v_i \in \mathbb{R}^3 \) is the position vector of the joint axis i expressed in frame i.

The twist \( \dot{\omega}_i \) can be expressed as a 6 by 1 vector \( s_i \) through a mapping \( \dot{s}_i = (\omega_i, v_i) \in \mathbb{R}^6 \), where \( \dot{s}_i \) is the twist coordinate of joint axis i. Therefore, based on the Product-of-Exponentials formula, the forward kinematics of serial open chain robot can be expressed as [9]:

T_n(q_1, q_2, \ldots, q_n) = T_{n-1,n}(q_n) \cdots T_2(q_2) T_1(q_1),

Differentiate Eq.(4) with respect to \( q_i \),

\[ \frac{dT_n}{dq_i} = \sum_{k=i}^{n} \frac{\partial T_k}{\partial q_k} \frac{dT_k}{dq_i} = \sum_{k=i}^{n} T_{k-1,i} \frac{dT_k}{dq_i} \frac{dT_k}{dq_i}. \]

Left multiplying both sides of Eq.(5) by \( T_{i-1,i}^{-1} \), we have

\[ T_{i-1,i}^{-1} \frac{dT_n}{dq_i} = \sum_{k=i}^{n} T_{k-1,i}^{-1} \frac{dT_k}{dq_i} \frac{dT_k}{dq_i}. \]

Based on the matrix logarithm defined on SE(3) [9, 10],

\[ \log(T_{i-1,i}^{-1} T_n) = \sum_{k=i}^{n} T_{k-1,i}^{-1} \frac{dT_k}{dq_i} \frac{dT_k}{dq_i}. \]

Using the adjoint presentation technique [10, 11], we have

\[ \log(T_{i-1,i}^{-1} T_n) = \sum_{k=i}^{n} Ad_{T_{i-1,i}^{-1}} s_i dq_i, \]

where, \( \log(T_{i-1,i}^{-1} T_n) = \sum_{k=i}^{n} Ad_{T_{i-1,i}^{-1}} s_i dq_i \) represents the 6 by 1 vector mapped from \( \log(T_{i-1,i}^{-1} T_n) = \sum_{k=i}^{n} \log(T_{i-1,i}^{-1} T_n) \). Owing to \( T_{i-1,i}^{-1} T_n = T_{i-1,i}^{-1} T_{i-1,i} T_n \), Eq.(8) can be rewritten as:

\[ \log(T_{i-1,i}^{-1} T_n) = Ad_{T_{i-1,i}^{-1}} \sum_{k=i}^{n} Ad_{T_{i-1,i}^{-1}} s_i dq_i. \]

Based on Eq.(9), the Newton-Raphson method can be employed to derive the inverse solution iteratively [9]. At each iteration, however, we need to compute the inverse of \( T_{i-1,i} \), i.e. \( T_{i-1,i}^{-1} \), which is not computational effective. Alternatively, Eq.(9) can also be written as:

\[ Ad_{T_{i-1,i}^{-1}} \log(T_{i-1,i}^{-1} T_n) = \log(T_{i-1,i}^{-1} T_n) \]

According to the definition of matrix logarithm and taking the first order approximation [12], we have

\[ \log(T_{i-1,i}^{-1} T_n) = \log(T_{i-1,i}^{-1} T_n) \]

Eq.(12) shows that \( \log(T_{i-1,i}^{-1} T_n) \) and \( \log(T_{i-1,i}^{-1} T_n) \) are commutative. Since both \( T_{i-1,i}^{-1} T_n \) and \( T_{i-1,i}^{-1} T_n \) are in the neighborhood of the identity matrix, we can get [12]:

\[ \log(T_{i-1,i}^{-1} T_n) + \log(T_{i-1,i}^{-1} T_n) = \log((T_{i-1,i}^{-1} T_n)(T_{i-1,i}^{-1} T_n)) = \log(I_{6×6}) = 0. \]

Hence,

\[ \log(T_{i-1,i}^{-1} T_n) = \log(T_{i-1,i}^{-1} T_n) \]

Substituting Eq.(14) and (10) into Eq.(9), we have

\[ D_T = Ad_{T_{i-1,i}^{-1}} \]

where \( D_T = \frac{\partial}{\partial q_i} \frac{dT_n}{dq_i} \) is the manipulator Jacobian matrix of the robot, \( dq = [dq_1, dq_2, \ldots, dq_n] \in \mathbb{R}^{6×1} \) represents joint angle increment. \( dq \) can be solved by

\[ dq = J^+ D_T, \]

where \( J^+ \) is Moore-Penrose pseudoinverse of \( J \). In this algorithm, \( T_n \) is the target end-effector pose. It will remain unchanged during iterations. Therefore, \( (T_n)_{-1} \) will only be needed to calculate one time for a given end-effector pose \( T_n \).
3 Optimization in Redundancy Resolution

To make use of the redundant actuation, let:

\[ J_M = JW^{-1} \quad dq_w = W^{-1}dq, \]  

where \( W \in R^{n \times n} \) is a symmetric positive-definite matrix, and is referred to as a weighted matrix. Usually it is a diagonal matrix of the form:

\[ W = \text{diag}[w_1, w_2, \ldots, w_n]. \]  

Its \( i^{th} \) entry is

\[ w_i = 1 + H(q)_i, \quad i = 1, 2, \ldots, n, \]  

where \( H(q) \) is termed as the kinematic performance criterion. Since Eq.(15) can be rewritten as:

\[ Dr = J_M dq_w, \]  

the least-norm solution for the weighted joint angle increment \( dq_w \) is of the form

\[ dq_w = J_M^+ Dr, \]  

where,

\[ J_M^+ = (J_M^TM)^{-1}J_M^T. \]  

On the other hand, as \( dq = W^{-\frac{1}{2}}dq_w \), the weighted least-norm solution for joint angle increment \( dq \) is

\[ dq = J_w^+ Dr, \]  

where the weighted pseudoinverse \( J_w^+ \) can be given by

\[ J_w^+ = W^{-\frac{1}{2}}J^{-1}W^{-\frac{1}{2}}J^T(W^{-\frac{1}{2}}J^TW^{-\frac{1}{2}}J)^{-1}. \]  

Without loss of generality, kinematic inverse solution can be described as:

\[ dq = J_w^+ Dr + K(I - J^+J)\delta q, \]  

where, \( K \) is the coefficient of the homogeneous solution, it can be found online according to joint velocity constraints [13]; \( \delta q \) is the time interval between two interpolations.

4 Interpolation Algorithm

Based on the optimized inverse kinematic solution, some interpolation algorithms are proposed for robot trajectory control in this section. Among them, the most fundamental interpolation algorithms are for straight-line motion and circular motion.

4.1 Straight Line Interpolation

As shown in Fig. 1, the interpolation algorithm can be divided into the following parts.

Determine the velocity profile

\[ \dot{q} = \frac{\text{vel}}{\text{acc}}, \]  

where, \( \text{vel} \) and \( \text{acc} \) are the pre-determined end-effector velocity and acceleration respectively.

2) Find the length of straight-line

If the starting point and the terminating point are denoted by \( P_0 \) and \( P_F \) respectively, the distance between the two points is:

\[ \text{dist} = \sqrt{\sum_{i=1}^{3} (P_{fi} - P_{oi})^2}. \]  

3) Modify the velocity profile

If the straight-line length is not long enough

\[ \begin{align*}
\dot{t} &= \sqrt{\text{dist} / \text{acc}} \\
\dot{q} &= \frac{\text{acc}^2}{2} \times \frac{\text{dist}}{2}.
\end{align*} \]
If the straight-line length is long enough
\[ t_s = \frac{\text{dist} - \text{acc} \cdot t^2}{\text{vel}} \]
The final velocity profile is shown in Fig. 2.

Determine the length of motion steps

Let \(k(k = 0,1,\ldots)\) be the \(k^\text{th}\) interpolation, \(t_s\) be the step factor, and \(t_k = k \cdot t_s\) be the \(k^\text{th}\) interpolation time. The step length for the \(k^\text{th}\) interpolation, \(\Delta S_k\), can be given by:

1) For acceleration > 0
\[ \Delta S_k = \frac{1}{2} \text{acc} \cdot t^2 - \frac{1}{2} \text{acc} \cdot t^2_{k-1}. \]  (30)

2) For acceleration = 0
\[ \Delta S_k = \text{vel} \cdot t_s. \]  (31)

3) For acceleration < 0
\[ \Delta S_k = \frac{1}{2} \text{acc}(2t_s + t - t_{k-1})^2 - \frac{1}{2} \text{acc}(2t_s + t - t_k)^2. \]  (32)

Determine the pose for the next interpolation

1) Position
\[ P_{k+1} = P_k + (P_s - P_o) \frac{\Delta S_{k+1}}{\text{dist}} \quad K = 0,1,2,\ldots. \]  (33)

2) Orientation
\[ S_{k+1} = S_k + \Delta S_{k+1}, \quad (S_s = 0) \]
\[ R_{k+1} = R_e^{\text{beg} R_k S_{k+1} \text{dist}}. \]  (34)

where \(R_e \in R^{3 \times 3}\) and \(R_k \in R^{3 \times 3}\) are the orientation of the starting point and the terminating point respectively; \(S_k\) is the distance from the starting point to the \(k^\text{th}\) interpolation point.

Motion commands

1) Find the numerical inverse kinematic solution to get joint angles for the \((k+1)^\text{th}\) interpolation.
\[ q_{k+1} = q_k + dq_{k+1} \quad K = 0,1,2,\ldots. \]  (35)

2) To smooth joint velocities, the next pose should be sent before the present pose has arrived. Fig. 3 explained this process except for the 1\text{st} interpolation whose signal is to be sent at the starting point.

4.2 Circle Interpolation

The interpolation method is similar to the straight line case, except for the following parts.

Determine the angle of the arc

This part will calculate the angle of the arc from the starting point to the terminating point.

1) Find the vector from the circle center \(P_c\) to the starting point and the terminating point respectively as shown in Fig. 4.
\[ \text{Beg} = P_o - P_c \]
\[ \text{End} = P_t - P_c \]  (36)

2) Find the angle from the vector \(\text{Beg}\) and \(\text{End}\) to X axis respectively as shown in Fig. 4.
\[ \alpha = \tan \frac{\text{Beg}_y}{\text{Beg}_x} \]
\[ \beta = \tan \frac{\text{End}_y}{\text{End}_x} \]  (37)

3) Find the radius of the arc from the starting point to the terminating point.
\[ r = \sqrt{\text{Beg}_x^2 + \text{Beg}_y^2}. \]  (38)

4) Transfer the negative angles to the positive angles.
\[ \tau_a = 2\pi + \alpha \quad (\text{if } \alpha < 0) \]
\[ \tau_a = \alpha \quad (\text{if } \alpha \geq 0) \]
\[ \tau\beta = 2\pi + \beta \quad (\text{if } \beta < 0) \]
\[ \tau\beta = \beta \quad (\text{if } \beta \geq 0) \]  (39)

5) Find the angle of rotation from the vector \(\text{Beg}\) to \(\text{End}\)
If \(\tau\beta > \tau\alpha\), move counter clockwise

3256
If $t f > t a$, move clockwise
\[
\text{ang} = |\beta - ta|;
\]  

(40a)  

If $t f > t a$, move counter clockwise
\[
\text{ang} = |2\pi - (t f - ta)|;
\]  

(40b)  

If $t f < t a$, move clockwise
\[
\text{ang} = |2\pi + (t f - ta)|;
\]  

(40c)  

If $t f < t a$, move clockwise
\[
\text{ang} = |\beta - ta|;
\]  

(40d)  

If $tE = tB$  
\[
\text{ang} = 0.
\]  

(40e)  

Determine the pose for the next interpolation

Let $c_j$ be a direction coefficient whose value depends on the rotating direction; Let $\varphi_k$ be the angle of the arc from the starting point to the $k$th interpolation point. We have

\[
\varphi_{k+1} = \varphi_k + c_j \Delta \varphi_{k+1}; \quad (\varphi_0 = 0; \ K = 0,1,2,\ldots)
\]

\[
c_j = \begin{cases} 
1 & \text{for counter clockwise motion} \\
-1 & \text{for clockwise motion} 
\end{cases}
\]

(41)  

\[
P_{k+1} = P_k + c_j r \cos(\varphi + \varphi_{k+1}) \quad (c_j \leq 1)
\]

\[
P_{k+1} = P_k + c_j r \sin(\varphi + \varphi_{k+1}) \quad (c_j \leq 1)
\]

\[
P_{k+1} = P_k + (P_k - P_o) \Delta S_{k+1}/\text{ang}
\]

\[
R_{k+1} = R_k e^{\cos(\varphi_{k+1})/\text{ang}}
\]

where the definition of $R_k$, $R_o$ and $\Delta S_{k+1}$ are the same as in the straight line case. However, dist should be replaced by ang in Eq.(28) and (29).

If $c_i = c_j = 1$, the arc is a part of a circle. If $c_i \neq c_j$, the arc is a part of an ellipse. If there is a change in Z axis, the planar arc will be transformed into a spiral trajectory.

5 Experiment

A modular reconfigurable robot workcell is shown in Fig. 5. This is a joint project between Gintic Institute of Manufacturing Technology and Nanyang Technological University. A 7-DOF serial robot controls the pick-and-place task online. A 6-DOF parallel robot controls a polishing task. A slider conveys the workpieces to be processed between the 7-DOF serial robot and the 6-DOF parallel robot. The supervisory PC coordinates every unit task and controls the demonstration.

The known kinematic parameters for the 7-DOF robot are:

\[
s_1 = [0, 0, -1, 0, 0, 0, 0]^T; \quad s_2 = [0, 0, -1, 0, 0, 0, 0]^T;
\]

\[
s_3 = [0, 0, -1, 0, 0, 0, 0]^T; \quad s_4 = [0, 0, -1, 0, 0, 0, 0]^T;
\]

\[
s_5 = [0, 0, 0, 0, 0, 0, 0]^T; \quad s_6 = [0, 0, 0, 0, 0, 0, 0]^T;
\]

\[
s_7 = [0, 0, 0, 0, 0, 0, 0]^T.
\]

Table 1 shows that the proposed algorithm has higher computation efficiency than the original algorithm.

Table 1 Computation time

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Iterations</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on Eq.(9)</td>
<td>3</td>
<td>10 ms</td>
</tr>
<tr>
<td>Based on Eq.(15)</td>
<td>3</td>
<td>9 ms</td>
</tr>
</tbody>
</table>

The results of optimization:

1) Without optimization: When the end-effector of 7-DOF serial robot moves over the slider, the wrist joint of the robot is $q_3 = -90.72$ deg. This is very close to its hard limit $[-93 93]$ deg (see Table 2 and Fig. 6).
2) With optimization by Eq.(23): $q_5 = -79.42$ deg, the wrist joint leaves its hard limit (see Table 3 and Fig. 7).

Table 2 Joint angles distribution without optimization (deg)

<table>
<thead>
<tr>
<th>$q_0$</th>
<th>$q_1$</th>
<th>$q_2$</th>
<th>$q_3$</th>
<th>$q_4$</th>
<th>$q_5$</th>
<th>$q_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>88.67</td>
<td>17.91</td>
<td>4.65</td>
<td>-54.06</td>
<td>14.27</td>
<td>-90.72</td>
<td>93.87</td>
</tr>
</tbody>
</table>

Table 3 Joint angles distribution with optimization (deg)

<table>
<thead>
<tr>
<th>$q_0$</th>
<th>$q_1$</th>
<th>$q_2$</th>
<th>$q_3$</th>
<th>$q_4$</th>
<th>$q_5$</th>
<th>$q_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>89.92</td>
<td>24.16</td>
<td>3.49</td>
<td>-22.00</td>
<td>53.36</td>
<td>-79.42</td>
<td>93.87</td>
</tr>
</tbody>
</table>

6 Summary

This paper presented a new method for the online control of redundant modular robots. Through constructing an object-oriented weighted matrix, the numerical inverse kinematic solution can be optimized. Since the robot redundancy is resolved at the joint angle level, the control accuracy can be improved significantly. Using the optimal inverse kinematic solution, some fundamental interpolation algorithms for Cartesian space motion controls, e.g., algorithms for straight-line and circular motion controls, are also developed. A pick-and-place experiment with the avoidance of joint angle limits for a 7-DOF serial robot verified these proposed algorithms.

Acknowledgment

This research is supported by Gintic Institute of Manufacturing Technology, Singapore.

References