An Efficient Inverse Kinematic Algorithm for a PUMA560-Structured Robot Manipulator

Regular Paper

Huashan Liu¹*, Wuneng Zhou¹, Xiaobo Lai² and Shiqiang Zhu³

1 College of Information Science and Technology, Donghua University, Shanghai, China
2 College of Information Technology, Zhejiang Chinese Medical University, Hangzhou, China
3 State Key Laboratory of Fluid Power Transmission and Control, Zhejiang University, Hangzhou, China

Received 24 Jan 2013; Accepted 18 Mar 2013

DOI: 10.5772/56403

© 2013 Liu et al.; 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 This paper presents an efficient inverse kinematics (IK) approach which features fast computing performance for a PUMA560-structured robot manipulator. By properties of the orthogonal matrix and block matrix, the complex IK matrix equations are transformed into eight pure algebraic equations that contain the six unknown joint angle variables, which makes the solving compact without computing the inverses of the 4×4 homogeneous transformation matrices. Moreover, the appropriate combination of related equations ensures that the solutions are free of extraneous roots in the solving process, and the wrist singularity problem of the robot is also addressed. Finally, a case study is given to show the effectiveness of the proposed algorithm.

Keywords Inverse Kinematics, PUMA560, Robot Manipulator

1. Introduction

Multi-degree of freedom serial robot manipulators with revolute joints are greatly used currently in various aspects of industrial applications, such as welding, painting, palletizing, transporting, CNC processing and so on. When the geometry structure of the robot manipulator satisfies the Pieper criterion [1], i.e., three consecutive joint axes of the robot are parallel or intersect at a common point, then a certain amount of closed-form solutions can be obtained for the inverse kinematics (IK) by the analytical method [2, 3], e.g., we can find eight closed-form solutions for a PUMA560-structured robot manipulator. The efficiency of the solving process directly affects the results on the motion control, especially in the online control situations.

The inverse transformation method, as a traditional IK algorithm, is widely adopted for its intuition, but it needs to work out the inverse of each 4×4 homogeneous transformation matrix, which results in a complex and time-consuming solving process [4]. More recently, some kinds of special techniques, such as vector-dot-product operations [5], product-of-exponentials (PoE) formulas [6], topological properties [7], double quaternions [8] and Bayesian network [9], are invoked in the IK algorithms to simplify the solving process.
Aiming at making further improvement on the real-time performance of the IK algorithm with closed-form analytical solutions, we propose a novel and efficient approach based on the orthogonality of rotation sub-matrix and multiplication properties of block matrix for a PUMA560-structured robot manipulator QJ-I. The rest of this paper is organized as follows. In Section 2 the kinematics of QJ-I is given. Section 3 presents some useful properties of the orthogonal matrix and block matrix. Section 4 deals with the reconstruction of the IK equations and the solving steps. Lastly, the case study is shown in Section 5 and conclusions are made in Section 6.

2. Kinematic Equations

QJ-I, a PUMA560 type robot, is as shown in Fig.1a, with its D-H link coordinate shown in Fig.1b, and the D-H parameters as well as the joint angle interval in Table 1. The QJ-I’s last three consecutive joint axes intersect at a common point - satisfying the Peiper criterion.

![Figure 1. QJ-I (a) and its D-H link coordinate (b)](image)

The direct kinematics (DK) of QJ-I can be described as

\[
\begin{bmatrix}
  n_x & a_x & p_x \\
  n_y & a_y & p_y \\
  n_z & a_z & p_z \\
\end{bmatrix} = \begin{bmatrix}
  0 & 0 & a \\
  0 & 0 & 0 \\
  0 & 0 & 1 \\
\end{bmatrix} = \begin{bmatrix}
  R & p \\
  0 & 1 \\
\end{bmatrix}
\]

where \( R \) is the 3×3 rotation matrix, including three 3×1 vectors \( n, o \) and \( a \), which respectively denote the normal vector, sliding vector and approach vector; \( p \) is the 3×1 position vector. The 4×4 homogeneous transformation matrix is:

\[
\begin{bmatrix}
  c_{\theta_1} c_{\theta_2} c_{\theta_3} c_{\theta_4} c_{\theta_5} c_{\theta_6} \\
  s_{\theta_1} c_{\theta_2} c_{\theta_3} c_{\theta_4} c_{\theta_5} c_{\theta_6} \\
  s_{\theta_1} s_{\theta_2} c_{\theta_3} c_{\theta_4} c_{\theta_5} c_{\theta_6} \\
  s_{\theta_1} s_{\theta_2} s_{\theta_3} c_{\theta_4} c_{\theta_5} c_{\theta_6} \\
\end{bmatrix} = \begin{bmatrix}
  R & p \\
  0 & 1 \\
\end{bmatrix}
\]

Corresponding to (1), the IK of QJ-I can be written as

\[
\theta = \begin{bmatrix}
  \theta_1 \\
  \theta_2 \\
  \theta_3 \\
  \theta_4 \\
  \theta_5 \\
  \theta_6 \\
\end{bmatrix} = f_{\text{IK}}^{-1}(R,p) = f_{\text{IK}}^{-1}(R_{IK},p_{IK})
\]

3. Some Properties of the Orthogonal Matrix and Block Matrix

i. If \( A \) is an orthogonal matrix, then \( A^{-1} \) is also an orthogonal matrix, and \( A^{-1} = A^T \).

ii. If matrix \( C = \begin{bmatrix}
  A & B \\
  0 & 1 \\
\end{bmatrix} \), then its inverse can be obtained

\[
\begin{bmatrix}
  A^{-1} & -A^{-1}B \\
  0 & 1 \\
\end{bmatrix}
\]

where \( C \) is a 4×4 matrix, \( A \) is a 3×3 matrix, \( B \) is a 3×1 vector.

Furthermore, we have

\[
C_1 \times C_2 = \begin{bmatrix}
  A_1 & B_1 \\
  0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
  A_2 & B_2 \\
  0 & 1 \\
\end{bmatrix} = \begin{bmatrix}
  A_1A_2 & A_1B_2 + B_1 \\
  0 & 1 \\
\end{bmatrix}
\]

By property (i), we find that \( R, R^{-1}, R^T, R^T \) and \( R^{-1} \) in the kinematic equations above are all orthogonal matrices, and \( R^{-1} = R^T \), \( R^{-1} = R^T \). By property (ii), the

<table>
<thead>
<tr>
<th>Joint No.</th>
<th>( d/\text{mm} )</th>
<th>( a/\text{mm} )</th>
<th>( \alpha/\text{°} )</th>
<th>( \theta_i )</th>
<th>Angle interval/°</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>250</td>
<td>150</td>
<td>-90</td>
<td>( \theta_1 )</td>
<td>(-90,270)</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>550</td>
<td>0</td>
<td>( \theta_2 )</td>
<td>(-270,90)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>160</td>
<td>-90</td>
<td>( \theta_3 )</td>
<td>(-270,90)</td>
</tr>
<tr>
<td>4</td>
<td>594</td>
<td>0</td>
<td>90</td>
<td>( \theta_4 )</td>
<td>(-180,180)</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>( \theta_5 )</td>
<td>(-180,180)</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \theta_6 )</td>
<td>(-270,90)</td>
</tr>
</tbody>
</table>

Table 1. D-H parameters of QJ-I
transformation matrix \( i^{-1}T \) can be partitioned into four units (rotation matrix \( R_c \), position vector \( p_c \), 0 and 1). Therefore, the complex operations of computing the inverse of each 4×4 transformation matrix are avoided.

4. Solving the IK Equations

4.1 Reconstruction of the Equations

In this part, by using the properties given above, we reconstruct the matrix equations described as (1) into relatively ordinary trigonometric function equations, thereby simplifying solving the process so as to enhance the real-time performance of the algorithm.

Considering that, the first three joints of QI-I are intended to determine the three-dimensional coordinates of the end-effector, while with the last three joints to determine the orientation, and to balance the equation on both sides with the same quantity of the unknowns, we break up the 6R-chain of QI-I as described in (1) into two 3R-chain[10], and then obtain

\[
3^T \cdot 4^T = 3^T \cdot 4^{-1} \cdot 3^{-1} \cdot 4^{-1} \cdot 0^T \cdot 2^{-1} \cdot 0^T = 0^T \cdot 2^{-1} \cdot 0^T
\]

By using the properties (i) and (ii) we get

\[
3^T \cdot 4^{-1} = \begin{bmatrix}
R_1^T & -R_1^T R_2 p_3 & 0 & 1 \\
0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

Then the left side of (4) can be written as

\[
3^T \cdot 4^{-1} = \begin{bmatrix}
R_1^T & -R_1^T R_2 p_3 & 0 & 1 \\
0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

In addition, the right side of (4) can be written as

\[
3^T \cdot 4^{-1} \cdot 3^{-1} \cdot 4^{-1} \cdot 0^T = \begin{bmatrix}
R_1^T & -R_1^T R_2 p_3 & 0 & 1 \\
0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

Finally, from (4)-(6), we obtain

\[
R_4 R_5 R_b = R_1^T R_2^T R_3^T R
\]

\[
R_4 R_5 p_b + p_s + p_t = R_1^T R_2^T R_3^T p - R_1^T R_2^T p - R_1^T R_2^T p - R_1^T R_2^T p
\]

By using the symbol processing software Maple©, equations (7) and (8) lead to

\[
s_4 p_z - c_4 p_y = 0
\]

\[
s_{23}(p_z + a_2 s_2 - d_1) - c_{23}(c_1 p_x + s_1 p_y - a_2 c_2 - a_1) = -a_3
\]

\[
s_{23}(c_1 p_x + s_1 p_y - a_2 c_2 - a_1) + c_{23}(p_z + a_2 s_2 - d_1) = -d_4
\]

\[
s_4 s_5 = c_{23} c_4 p_x + c_{23} s_4 p_y - s_{23} d_2
\]

\[
s_4 s_5 = s_4 d_2 - c_4 p_y
\]

\[
s_5 c_6 = -s_{23} c_4 p_x - s_23 s_4 p_y - c_{23} d_2
\]

\[
s_5 c_6 = s_{23} c_4 p_x - s_23 s_4 p_y - c_{23} d_2
\]

\[
s_6 s_8 = -s_{23} c_4 p_x - s_23 s_4 p_y - c_{23} d_2
\]

\[
s_6 s_8 = s_{23} c_4 p_x - s_23 s_4 p_y - c_{23} d_2
\]

where \( s_23 = \sin(\theta_2 + \theta_3) \), \( c_{23} = \cos(\theta_2 + \theta_3) \).

Up to now we have obtained eight pure algebraic trigonometric equations (9)–(16), which is less than the algorithm proposed in [4] (with nine equations). Although the proposed algorithm has the same quantity of equations as the algorithm proposed in [5], all the equations are generated from (1) directly without constructing new equations as \( p p \) and \( p \alpha \) in [5], which may be a little abrupt.

4.2 Solving Steps

Here, the four-quadrant inverse tangent function atan2 will be involved frequently in the specific solving process to obtain the complete solutions as follows.

Step 1: By (9), we get two solutions of \( \theta_1 \).

Step 2: Square both sides of (10) and (11), respectively, then add together. We get

\[
(d_1 - p_x) s_2 - (c_1 p_x + s_1 p_y - a_2) c_2 = \frac{a_4^2 + d_4^2 - a_2^2 - (c_1 p_x + s_1 p_y - a_2)^2 - (d_1 - p_x)^2}{2a_2}
\]

For each \( \theta_1 \) we can get two solutions of \( \theta_2 \), i.e., four solutions of \( \theta_2 \) in total.

Step 3: By (10), one solution of \( \theta_1 \) corresponding to each \( \theta_2 \), i.e., four solutions of \( \theta_1 \) can be obtained.

Step 4: When we solve \( \theta_1 \) by (12) and (13), we should judge whether the robot manipulator is singular or not, due to that singularities inherently limiting the capability of a manipulator to complete its task [11].

If \( \theta_5 \neq 0^\circ \), then \( \theta_1 = \text{atan2}(s_5 s_\phi, c_5 s_\phi) \) and \( \text{atan2}(-s_5 s_\phi - c_5 s_\phi) \) for each set of \( \theta_1 \), \( \theta_2 \) and \( \theta_3 \);

By (1) and (2), we can obtain

\[
3^T \cdot 4^T = \begin{bmatrix}
R_1^T & -R_1^T R_2 p_3 & 0 & 1 \\
0 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

If \( \theta_5 = 0^\circ \), i.e., QI-I is in wrist singularity, (16) can be written as
Now that the fourth and the sixth joint axes of QJ-I are overlapped, the value of \( \theta_6 \) can be chosen arbitrarily in theory. It is worth noting that from (19) we find that the transformation matrix relates solely to \((\theta_1-\theta_3)\), and on this condition, to keep away from the singularities and fix the pose of QJ-I in the meantime, we only need to rotate both \( \theta_1 \) and \( \theta_3 \) at an equal angle (which is small enough) in the same direction, to change the axis direction of the fifth joint, thereby avoiding the singular point.

**Step 5:** By (12) and (14), we can get one solution of \( \theta_5 \) corresponding to each set of \( \theta_1, \theta_2, \theta_3 \) and \( \theta_6 \).

**Step 6:** By (15) and (16), one solution of \( \theta_6 \) can be obtained corresponding to each set of \( \theta_1, \theta_2, \theta_3 \) and \( \theta_6 \). So far, we have obtained all eight closed-form solutions of \( \theta_1, \theta_2, \theta_3, \theta_4, \theta_5 \) and \( \theta_6 \). It is worth pointing out that benefiting from the proper combinations of the related equations, there generates no extraneous root in each solving step, which always occurs in the inverse transformation method in [4] and the vector-dot-product-based approach in [5]. This advantage excuses the IK solving from both verifying and matching the roots, which further enhances the efficiency of the algorithm.

5. Case Study

In this part, sample calculations are executed, together with simulations and tests to verify the effectiveness of the proposed IK algorithm.

\[
^sT = \begin{bmatrix}
\cos(\theta_4 - \theta_2) & \sin(\theta_4 - \theta_2) & 0 & 0 \\
\sin(\theta_4 - \theta_2) & -\cos(\theta_4 - \theta_2) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The pose matrix of the end-effector relative to the bottom base is given as

\[
^sT = \begin{bmatrix}
-0.0188 & 0.4154 & 0.9095 & 206.7566 \\
0.4810 & 0.8012 & -0.3560 & 55.4003 \\
-0.8765 & 0.4307 & -0.2148 & -418.0041 \\
0 & 0 & 0 & 1.0000
\end{bmatrix}
\]

According to the IK algorithm proposed and D-H parameters, as well as the joint angle interval shown in Table 1, we get eight closed-form solutions as shown in Table 2, and the corresponding orientation simulation in Fig. 2.

To testify to the real-time performance of the proposed IK algorithm, we firstly sampled 681 discrete points from a closed-curve called QS Eagle referred to in [12], and recorded the three-dimensional coordinate of each point, then put them into the pose matrix of the end-effector, after getting these 681 samples of pose matrices. With the proposed method in this paper, (called “Ours”), the inverse transformation method in [4] (called “ITM”) and the vector-dot-product-based approach in [5] (called “VDP”), we solved out the IK solutions of each given pose matrix respectively, in the VC++ compiling environment on a notebook computer platform (Windows XP 32-bit SP3 OS, Intel Core 2 Duo Dual 2.2GHz CPU, 2GB DDR3 800MHz RAM) for 500 times with an accuracy of 0.00000001°. The performance comparisons of the three schemes are shown in Table 3.

<table>
<thead>
<tr>
<th>No.</th>
<th>( \theta_1/(°) )</th>
<th>( \theta_2/(°) )</th>
<th>( \theta_3/(°) )</th>
<th>( \theta_4/(°) )</th>
<th>( \theta_5/(°) )</th>
<th>( \theta_6/(°) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15.00000931</td>
<td>-215.95388774</td>
<td>-184.84918850</td>
<td>51.85908138</td>
<td>132.56569742</td>
<td>-5.57849644</td>
</tr>
<tr>
<td>3</td>
<td>15.00000931</td>
<td>24.99999937</td>
<td>35.00000104</td>
<td>45.00289124</td>
<td>54.99852255</td>
<td>65.00379351</td>
</tr>
<tr>
<td>4</td>
<td>15.00000931</td>
<td>24.99999937</td>
<td>35.00000104</td>
<td>-134.99710876</td>
<td>-54.99852255</td>
<td>-114.99620494</td>
</tr>
<tr>
<td>5</td>
<td>195.00000931</td>
<td>-188.34210158</td>
<td>-173.60896143</td>
<td>-143.86066631</td>
<td>100.83006201</td>
<td>27.34910700</td>
</tr>
<tr>
<td>6</td>
<td>195.00000931</td>
<td>-188.34210158</td>
<td>-173.60896143</td>
<td>36.13933369</td>
<td>-100.83006201</td>
<td>-152.65018930</td>
</tr>
<tr>
<td>7</td>
<td>195.00000931</td>
<td>65.52127702</td>
<td>23.75977397</td>
<td>-70.51870198</td>
<td>142.09005479</td>
<td>-78.98705841</td>
</tr>
<tr>
<td>8</td>
<td>195.00000931</td>
<td>65.52127702</td>
<td>23.75977397</td>
<td>109.48129802</td>
<td>-142.09005479</td>
<td>-258.98705841</td>
</tr>
</tbody>
</table>

Table 2. IK solutions of QJ-I

<table>
<thead>
<tr>
<th>Main means</th>
<th>Extraneous root</th>
<th>Average time cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>matrix blocking</td>
<td>has not</td>
</tr>
<tr>
<td>ITM</td>
<td>inverse</td>
<td>transformation</td>
</tr>
<tr>
<td>VDP</td>
<td>vector dot</td>
<td>product</td>
</tr>
</tbody>
</table>

As shown in Table 3, by our scheme, it just took an average time of 6.753 \(\mu\)s to figure out all eight solutions for one sample pose matrix, which is 49.5% of the average time cost by ITM and 76.6% of that by VDP, due to no calculating the inverse of the matrix and producing no extraneous root in the IK solving.
6. Conclusions

This paper addresses the problem of an efficient IK algorithm for a PUMA560-structured robot manipulator. First, some properties of the orthogonal matrix and block matrix are applied to help simplify reconstructing the IK matrix equations into trigonometric function equations, without calculating the inverses of the homogeneous transformation matrices. As a second contribution, the proposed IK algorithm is free of producing extraneous roots by appropriately combining the related equations for a certain unknown, which allows the solving to avoid identifying the roots and matching the real root of one joint with those of the other joints. All of these qualify the proposed IK algorithm for high real-time control situations with efficiency computing performance as presented in the case study. Furthermore, although the wrist singularity, as the most common singular type for a PUMA560-structured robot manipulator, is discussed in the IK solving, the other singularities, such as boundary singularity or inner singularity, are not mentioned and remain to be further studied.

7. Acknowledgements

We express our sincere thanks to the editors, referees and all the members of our discussion group for their beneficial comments, as well as special thanks to Zhou Zhou for her work on the image processing. The presented work has been partially supported by the National Natural Science Foundation of China under grant nos. 61203337, 61075060, the Specialized Research Fund for the Doctoral Program of Higher Education under grant no. 20120075120009, the Natural Science Foundation of Shanghai under grant no. 12ZR1440200, the Zhejiang Province Natural Science Foundation of China under grant no. LQ12F01004, the Fundamental Research Funds for the Central Universities under grant no. 12D10408, and the Young Teacher Training Program for the Shanghai Universities under grant no. DHU11035.

8. Reference