空间机器人坐标逆变换超高速算法

洪炳镕 李立新 姜 辉
(哈尔滨工业大学计算机科学与工程系, 150006)

摘要 本文提出了从机械手末端位置及姿势求出各关节转角的坐标逆变换超高速算法. 这种算法是以 CORDIC 算法[2]为基础的, 首先将坐标逆变换转化成坐标的旋转移动和反正切及其组合, 然后用 CORDIC 算法来快速实现坐标旋转和反正切算法运算. 这种超高速算法的最大特点是把坐标逆变换中复杂而无规则的非标准函数计算变成有规则的计算, 从而容易实现坐标逆变换超高速专用处理器.

关键词: 操作器 CORDIC 算法 坐标逆变换 坐标旋转 反正切

1 前言

从操作器末端位置和姿势求出各关节转角并使其高速化已成为非常重要的问题. 尤其在空间机器人或某些装配机器人动作的实时操作控制中, 为了完成适应于作业环境变换的复杂动作, 这种坐标逆变换必须在很短的时间内完成, 否则, 由于计算时间长, 会给系统带来低效率或误动作.

目前, 处理操作器坐标逆变换有几种方法, 如: 先求出操作器的各关节转角的解析解, 再以它为基础求出各关节转角的数值计算法[1], 这种方法虽然直观, 但是由于解析解中很少有共同的运算式, 并且包含许多特殊的非标准函数计算, 因此处理速度很慢; 第 2 种方法是利用雅可比矩阵的逆矩对其进行反变换; 第 3 种方法是采用并行处理的方法使求解并行化, 用以提高处理速度. 日本的龟谷等人曾利用 7 个 CPU 和 14 个运算处理单元实现了坐标逆变换的并行处理系统[4], 并使坐标逆变换处理时间减少到 3.3ms. 本论文提出了一种通过坐标的旋转移动和反正切及其组合来实现坐标逆变换的方法, 并用 CORDIC 算法来处理这两种计算, 使操作器坐标逆变换的处理时间从毫秒数量级提高到微秒数量级.

2 操作器坐标逆变换

操作器末端位置和姿势用 P 表示, 各关节的转角用 θ 表示, 即

$$P = (P_1, P_2, \cdots, P_m) \tag{1}$$

$$\theta = (\theta_1, \theta_2, \cdots, \theta_n) \tag{2}$$

其中 m 表示作业坐标的维数, n 表示操作器的自由度.

对图 1 给定的例子, 采用齐次变换求得解析解如下[3]:

$$\theta_1 = \arctan(P_y/P_x) \tag{3}$$

$$\theta_3 = \arctan \frac{\sqrt{2(l_1 l_2)^2 - (P_x^2 + P_y^2 + P_z^2 - l_1^2 - l_2^2)^2}}{P_x^2 + P_y^2 + P_z^2 - l_1^2 - l_2^2} \tag{4}$$

$$\theta_2 = \arctan \frac{-P_z}{\sqrt{P_x^2 + P_y^2}} + \arctan \frac{-l_2 s_3}{l_1 + l_2 c_3} \tag{5}$$

收稿时间: 1991—08—08

$$\theta_4 = \arctan\frac{-s_1A_x + c_1A_y}{-s_{23}(c_1A_x + s_1A_y) - c_{23}A_z} \tag{6}$$

$$\theta_5 = \arctan\frac{-c_4(s_{23}(c_1A_x + s_1A_y) + c_{23}A_z) + s_4(-s_1A_x + c_1A_y)}{c_{23}(c_1A_x + s_1A_y) - s_{23}A_z} \tag{7}$$

$$\theta_6 = \arctan\frac{-c_5(c_4(-s_{23}(c_1O_x + s_1O_y) - c_{23}O_z) + s_4(s_{23}(c_1O_x + s_1O_y) + c_{23}O_z) + s_4(-s_1O_x + c_1O_y)) + s_5(c_{23}(c_1O_x + s_1O_y) - s_{23}O_z)}{+c_4(-s_1O_x + c_1O_y)} \tag{8}$$

其中 $s_k = \sin\theta_k (k = 1,2,\cdots,6)$；$c_k = \cos\theta_k (k = 1,2,\cdots,6)$；$s_{23} = \sin(\theta_2 + \theta_3)$；$c_{23} = \cos(\theta_2 + \theta_3)$；$P = (P_x, P_y, P_z)$ 表示手腕的直角坐标；$A = (A_x\ A_y\ A_z)$ 表示手端指向的单位向量；$O = (O_x\ O_y\ O_z)$ 表示垂直于手端面的单位向量.

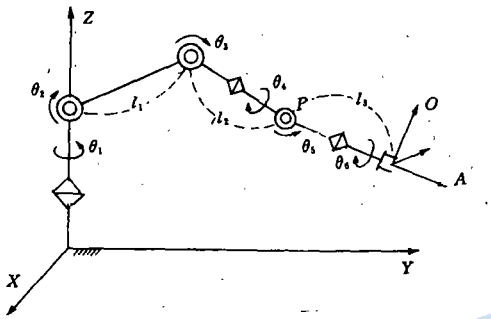


图 1　6 自由度操作器的例子(RPPRPR 式)

从上面的式子可以发现，在坐标逆变换中包含着各种三角函数和反三角函数等非标准函数，而且在各关节角的计算中很少包含共同的计算式，因此提高计算速度很困难.

3　逆变换的高速化

为了使变换高速化，必须在复杂的逆变换式(式(3)～(8))中找出共同的计算式. 本文着眼于将坐标逆变换通过坐标的旋转移动和反正切及其组合的方法来实现，首先要将给定的解析解分别分解为旋转移动的形式，其分解方法如下.

(Ⅰ)将逆变换式分为分子和分母两部分，并把它们各自作为一个独立的关系式. 设 $i=n$，n 表示自由度，i 表示重复次数，G 是 θ 的函数.

(Ⅱ)试看每个关系式是否变成

$$\left.\begin{aligned}\cos\theta_i G_1(\theta) - \sin\theta_i G_2(\theta)\\ \sin\theta_i G_1(\theta) + \cos\theta_i G_2(\theta)\end{aligned}\right\} \tag{9}$$

若能变成式(9)的形式，则可将该关系式表示成坐标 $[G_1(\theta), G_2(\theta)]$ 对 θ_i 的旋转移动，即：

$$\begin{bmatrix}\cos\theta_i G_1(\theta) - \sin\theta_i G_2(\theta)\\ \sin\theta_i G_1(\theta) + \cos\theta_i G_2(\theta)\end{bmatrix} = \begin{bmatrix}\cos\theta_i & -\sin\theta_i\\ \sin\theta_i & \cos\theta_i\end{bmatrix}\begin{bmatrix}G_1(\theta)\\ G_2(\theta)\end{bmatrix} \tag{10}$$

再将 $G_1(\theta)$、$G_2(\theta)$ 各自定义为新的关系式进入步骤(Ⅲ). 若不能变成式(9)的形式，则不改动原关系式进入步骤(Ⅲ).

(Ⅲ)$i \leftarrow i-1$，若 $i \neq 0$ 则进入步骤(Ⅰ)，若 $i=0$，则结束基于旋转移动的分解.

据此算法，对图 1 所示 6 自由度多关节型操作器的第 4 关节角 θ_4(式(6))具体分解如下：

$$\theta_4 = \arctan\frac{-s_1A_x + c_1A_y}{-s_{23}(c_1A_x + s_1A_y) - c_{23}A_z} \tag{11}$$

首先对分母. $i=2,3$，即对 $(\theta_2+\theta_3)$ 可表示为：

$$-[s_{23}G_1 + c_{23}G_2]; \qquad G_1 = A_xc_1 + A_ys_1; \qquad G_2 = A_z \tag{12}$$

因此可表示成旋转移动形式

$$\begin{bmatrix}c_{23}G_1 - s_{23}G_2\\ s_{23}G_1 + c_{23}G_2\end{bmatrix} = \begin{bmatrix}c_{23} & -s_{23}\\ s_{23} & c_{23}\end{bmatrix}\begin{bmatrix}G_1\\ G_2\end{bmatrix} \tag{13}$$

然后对新的关系式 G_1 再进行分解，置 $i=1$，由于 G_1 可表示为：

$$G_1 = s_1G_3 + c_1G_4; \qquad G_3 = A_y; \qquad G_4 = A_x \tag{14}$$

因而 G_1 的旋转移动形式为：

$$\begin{bmatrix} c_1G_3 - s_1G_{14} \\ s_1G_3 + c_1G_4 \end{bmatrix} = \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix} \quad \begin{bmatrix} G_3 \\ G_4 \end{bmatrix} \tag{15}$$

再做 $i \leftarrow i-1$ 得 $i=0$，故分母分解结束. 对分子以同样的方法进行分解，但这个分子的分解在式(15)中已经求出故可采用此式.

通过对上述已分解的旋转移动进行组合，对 θ_4 的分子分母可按如下过程计算.

$$\begin{bmatrix} A_y \\ A_x \end{bmatrix} \xrightarrow{\text{旋转 } \theta_1} \begin{bmatrix} -s_1A_x + c_1A_y \\ c_1A_x + s_1A_y \end{bmatrix} \rightarrow \text{分子}$$

$$\begin{bmatrix} c_1A_x + s_1A_y \\ A_z \end{bmatrix} \xrightarrow{\text{旋转 } \theta_2 + \theta_3} \begin{bmatrix} c_{23}(c_1A_x + s_1A_y) - s_{23}A_z \\ s_{23}(c_1A_x + s_1A_y) + c_{23}A_z \end{bmatrix} \rightarrow \text{分母}$$

因此，式(11)中的 16 次运算转化成 2 次旋转移动和 1 次加法及 1 次的反正切运算. 同样，对 $\theta_2 \sim \theta_6$ 按上述步骤进行分解，通过旋转移动的组合大大减少运算次数. 这样，将逆变换的解析变换成旋转移动和反正切的组合，可以使式(3)～(8)中的 83 次计算减少到 25 次.

4 CORDIC 算法

本文为使旋转移动和反正切运算高速实现，采用了坐标旋转数值计算方法(即 CORDIC 算法). CORDIC 算法是将任意坐标的旋转移动通过离散地反复旋转直至收敛而得到，操作只有加法、减法和移位，因此很容易实现，从而提高了计算速度，简化了程序设计.

通常，任意的二维直角坐标(x_i, y_i)对角度 θ_j 的旋转移动由

$$x_{i+1} = x_i\cos\theta_j - y_i\sin\theta_j \tag{16}$$

$$y_{i+1} = x_i\sin\theta_j + y_i\cos\theta_j \tag{17}$$

式给出. 其中 i 表示离散重复次数. 图 2 坐标旋转对此式两边除以 $\cos\theta_j$(假定 $\cos\theta_j \neq 0$)得到：

$$x_{i+1}/\cos\theta_j = x_i - y_i\tan\theta_j \tag{18}$$

$$y_{i+1}/\cos\theta_j = y_i + x_i\tan\theta_j \tag{19}$$

在这里适当地选择 θ_j 使下式成立：

$$\tan\theta_j = 2^{-j} \tag{20}$$

则式(18)、(19)可写成：

$$x_{i+1}/\cos\theta_j = x_i - y_i \cdot 2^{-j} \tag{21}$$

$$y_{i+1}/\cos\theta_j = y_i + x_i \cdot 2^{-j} \tag{22}$$

这样，对 θ_j 的离散旋转移动是通过 2 的幂次计算来实现的，而这种计算可以用移位和加法运算及比例修正($1/\cos\theta_j$ 倍)来完成. 亦即，将从(x_i, y_i)到(x_{i+1}, y_{i+1})的旋转可定义为对角度 θ_j 的旋转移动和 $1/\cos\theta_j$ 的比例计算，这种旋转移动只用移位和加减运算就能实现. 对这种运算，如果考虑旋转方向，得到如下基本运算公式：

$$x_{i+1} = x_i - \delta_i y_i 2^{-j} \tag{23}$$

$$y_{i+1} = y_i + \delta_i x_i 2^{-j} \tag{24}$$

$$z_{i+1} = z_i - \delta_i \theta_j \tag{25}$$

在这里，δ_i 表示旋转方向，$\delta_i=+1$ 表示正方向(逆时针)旋转. $\delta_i=-1$ 表示负方向(顺时针)旋转. z_i 表示旋转角的累计值. θ_j 是根据式(20)得到的离散值.

$$j=i-1 \qquad (i\geqslant 1) \tag{26}$$

对于图 1 所给出的操作器的例子，逆变换转换成旋转移动的组合后，需要处理的基本运算有坐标旋转、反正切、乘法运算(比例修正用)和双曲函数运算(θ_3 用)，这些运算都可以用 CORDIC 算法快速实现，具体方法如下.

1)旋转移动：假设直角坐标系上某一点(x,y)要旋转某一角度 θ，则令 $x_1=x, y_1=y, z_1=-\theta(-90°<\theta<90°)$，应用式(23)～(25)，使点$(x_1,y_1)$离散地进行旋转，由于考虑了旋转方向(即 δ_i 的选取)，随着 i 的增大，z_i 也逐步趋向于 0. 这样，通过离散地旋转就完成了某一 θ 角的旋转，如图 3 所示.

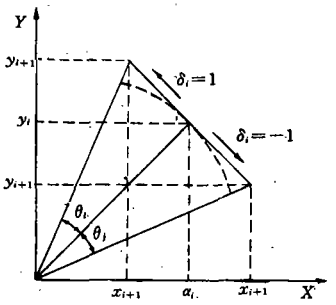


图 2　坐标旋转

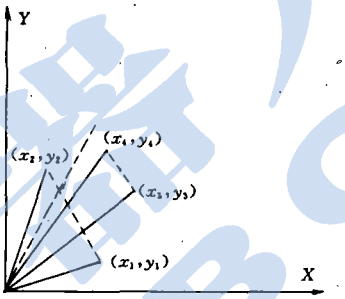


图 3　CORDIC 旋转

因而，点(x_1,y_1)通过从 $i=1$ 到 n 的离散旋转移动得到下式：

$$x_n=K(x_1\cos z_1-y_1\sin z_1) \tag{27}$$

$$y_n=K(x_1\sin z_1+y_1\cos z_1) \tag{28}$$

$$K=\prod_j(1/\cos\theta_j) \tag{29}$$

n 表示基本运算式(23)～(25)的重复次数，将其加大则可提高计算精度，K 是由 n 决定的常数. 据此过程容易得到应用 CORDIC 算法处理旋转移动的流程图. 为使旋转范围增大到 180°～－180°，开始处应增加一个 90°的旋转变换，对 $\theta_0=90°$的旋转，根据式(16)，(17)有

$$x_1=-\delta_0 y_0 \tag{30}$$

$$y_1=\delta_0 x_0 \tag{31}$$

2)反正切运算：对于反正切运算，可在基本式(23)～(25)中令 $z_0=0$，并通过变量 y 来控制 δ_i 的取值，即 $y_i\geqslant 0$ 则 $\delta_i=-1$；$y_i<0$ 则 $\delta_i=1$. 当 n 的取值使 $y_n\to 0$ 时，则得到的 z_n 值为 $\arctan(y_0/x_0)$，x_n 值为 $K\sqrt{x_0^2+y_0^2}$. 根据这个思想，反正切的处理流程很容易得到.

3)乘法运算：假设$|z|<1$，用 z 作为控制变量来控制 δ_i 的取值，并做如下的基本式：

$$x_{i+1}=x_i \tag{32}$$

$$y_{i+1}=y_i-\delta_i x_i 2^{-j} \tag{33}$$

$$z_{i+1}=z_i+\delta_i 2^{-j} \tag{34}$$

控制使 $z_i\to 0$，得 $y_n=x_0z_0+y_0$，从而完成了 x_0 和 z_0 的相乘.($|z|\geqslant 1$ 时，做比例修正即可).

4)双曲线运算：选 y 做控制量来控制 δ_i 的取值，并做如下基本式：

$$x_{i+1}=x_i+\delta_i y_i 2^{-j} \tag{35}$$

$$y_{i+1} = y_i + \delta_i x_i 2^{-j} \tag{36}$$

$$z_{i+1} = z_i + \delta_i r_j \tag{37}$$

其中，令 $\tan hr_j = 2^{-j}$，或(35)～(37)中满足 $r_j(i) \leqslant \sum_{k=i-1}^{h-1} r_j(k)$. 控制 y，使 $y_n \to 0$，则可得到：

$$x_n = K' \sqrt{x_o^2 - y_0^2} \tag{38}$$

$$z_n = -\tan h^{-1}(y_0/x_0) + z_0 \tag{39}$$

$$K' = \prod_j (1/\cos hr_j) \tag{40}$$

由此可得到以上 4 种运算的 CORDIC 算法基本模型(图 4).

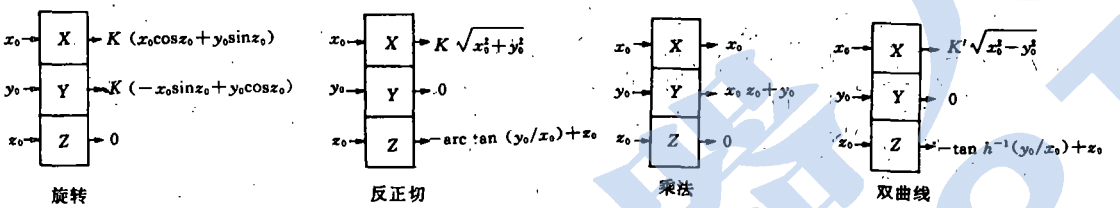


图 4 CORDIC 基本模型

5 坐标逆变换超高速处理流程

本论文的基本思想是将坐标逆变换解析解转变为坐标旋转和反正切等基本运算，并用 CORDIC 算法来处理这些基本运算. 对图 1 所给出的操作器的例子，其逆变换处理流程如图 5 所示，其解析式见公式(3)－(8). 图中所使用的基本符号定义如下：

$M(0)=P_x$ $M(1)=P_y$ $M(2)=P_z$ $M(3)=-P_z$

$M(4)=A_x$ $M(5)=A_y$ $M(6)=A_z$ $M(7)=O_x$

$M(8)=O_y$ $M(9)=O_z$ $M(10)=-\theta_1$ $M(11)=-\theta_9$

$M(12)=-\theta_{23}$ $M(13)=-\theta_4$ $M(14)=-\theta_5$ $M(15)=-\theta_6$

$M(16)=1/K$ $M(17)=-1/K$ $M(18)=K'$ $M(19)=l_2$

$M(20)=K \cdot l_1$ $M(21)=-l_1^2-l_2^2$ $M(22)=2l_1l_2$ $M(23)=K\sqrt{P_x^2+P_y^2}$

$M(24)=P_x^2+P_y^2+P_z^2-l_1^2-l_2^2$ $M(25)=K\sqrt{(2l_1l_2)^2-(P_x^2+P_y^2+P_z^2-l_1^2-l_2^2)^2}$

$M(26)=-\theta_3-\arctan(-s_3l_2/(l_1+c_3l_2))$ $M(27)=K(-s_1A_x+c_1A_y)$

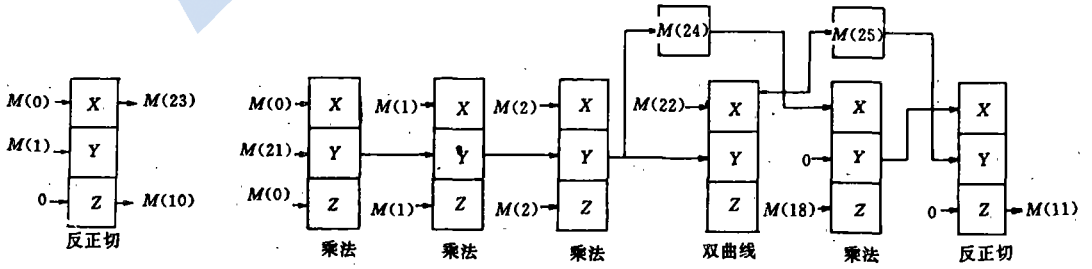
$M(28)=K(c_{23}(c_1A_x+s_1A_y)-s_{23}A_z)$ $M(29)=K(s_{23}(c_1A_x+s_1A_y)+c_{23}A_z)$

$M(30)=K(-s_1O_x+c_1O_y)$ $M(31)=K(c_{23}(c_1O_x+s_1O_y)-s_{23}O_z)$

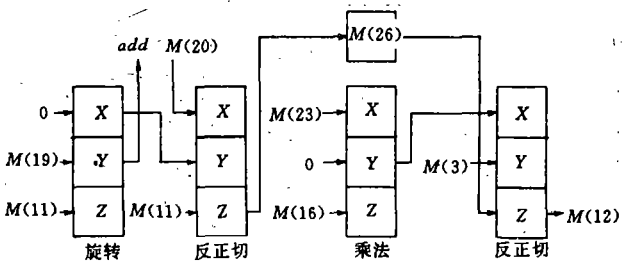
$M(32)=K^2(s_4(s_{23}(c_1O_x+s_1O_y)+c_{23}O_z)+c_4(-s_1O_x+c_1O_y))$

(a) $\theta_1=\arctan(P_y/P_x)$

(b) $\theta_3=\arctan(\sqrt{(2l_1l_2)^2-(P_x^2+P_y^2+P_z^2-l_1^2-l_2^2)^2}/(P_x^2+P_y^2+P_z^2-l_1^2-l_2^2)$

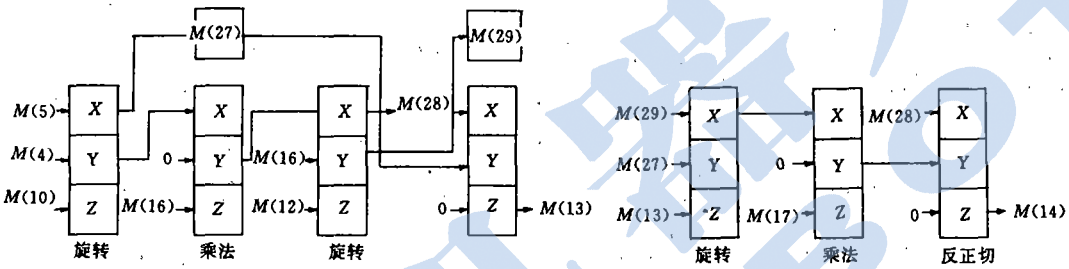


(c) $\theta_2=\arctan(-P_z/\sqrt{P_x^2+P_y^2})+\arctan(-l_2s_3/l_1+l_2c_3)$



(d) $\theta_4=\arctan\dfrac{-s_1O_x+c_1O_y}{-s_{23}(c_1A_x+s_1A_y)-c_{23}A_z}$

(e) $\theta_5=\arctan\dfrac{-c_4(s_{23}(c_1A_x+s_1A_y)+c_{23}A_z)+s_4(-s_1A_x+c_1A_y)}{c_{23}(c_1A_x+s_1A_y)-s_{23}A_z}$



(f) $\theta_6=\arctan\dfrac{-c_5(c_4(-s_{23}(c_1O_x+s_1O_y)-c_{23}O_z)+s_4(s_{23}(c_1O_x+s_1O_y)+c_{23}O_z)+s_4(-s_1O_x+c_1O_y))+s_5(c_{23}(c_1O_x+s_1O_y)-s_{23}O_z)}{+c_4(-s_1O_x+c_1O_y)}$

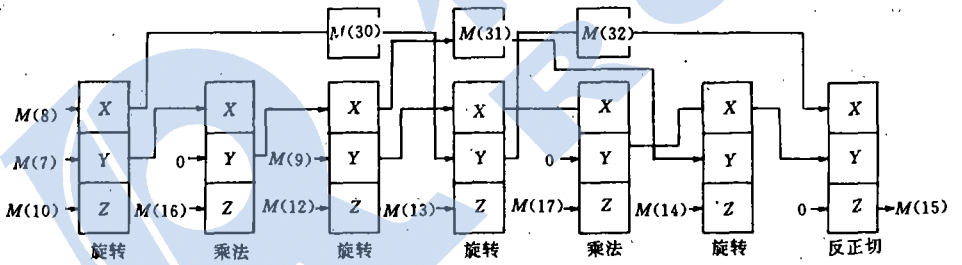


图 5　逆变换处理流程

对 θ_1 角如图 5(a)，可置 $x_0=P_x, y_0=P_y, z_0=0$，经 CORDIC 算法的反正模式计算可得：

$$\theta_1 = \arctan P_y/P_x$$

对 θ_3 角如图 5(b)所示，经过三次乘法运算得到 $M(24)=P_x^2+P_y^2+P_z^2-l_1^2-l_2^2$；再经过一次双曲线运算得到 $M(25)=K'\sqrt{(2l_1l_2)^2-(P_x^2+P_y^2+P_z^2-l_1^2-l_2^2)^2}$，再经过一次乘法运算对 $M(24)$ 乘以比例系数 K'，最后通过反正切运算得到 $M(11)=\dfrac{M(25)}{K'M(24)}=\theta_3$. 对于其它角度，经过图 5 所示的流程处理也可精确地得到.

6　算法评价

如前所述，对图 1 所示的操作器的具体例子，其逆变换解析解(式(3)～(8))的 83 次各种

复杂的三角函数等计算经过转换变成如图 5 所示的 25 次旋转和反正切等基本运算，这些运算可用 CORDIC 算法进行处理. 这样，只需加法和移位操作就能得到 $\theta_1 \sim \theta_6$ 的值，这两种操作用硬件实现是快速和简单的[5,6]. 并且，根据 CORDIC 算法，在硬件处理过程中，x, y, z 三个变量可以以三路全并行的方式进行处理，这就大大提高了处理速度. 如果计算精度为 16 位，则每一个 CORDIC 基本运算式需要 16 次加法和 16 次移位操作. 因而，图 5 所示的处理流程，25 次 CORDIC 基本运算以三路全并行的方式进行处理，共需 400 次加法和 400 次移位操作. 这样，操作器坐标逆变换的处理时间可很容易地达到几十到几百个微秒的数量级.

7 结论

本文讨论了以 CORDIC 算法为基础的操作器坐标逆变换超高速算法，使毫无规则的复杂的逆变换的解析解，可以以三路全并行的方式进行处理，并且这种处理只用加法和移位操作就能实现，从而提高了处理速度，简化了程序设计，实现了运算的规则化. 由于运算只需加法移位操作，因而硬件也是十分简单和易于实现的. 本文论述的方法是以 6 自由度 RPPRPR 型操作器为例进行讨论的，这种方法同样适用于其它类型和多个自由度的操作器.

参 考 文 献

1 Andrew A Goldenberg, Benhabib B, Robert B, Fenton G. A complete generalized solution to the inverse kinematics of robots. IEEE Journal, Robots and Automation, 1985; RA-1(1): 14-20

2 Jack E Volder. The CORDIC trigonometric computing technique. IRE Trans, Electron, Comput, 1959; EC-8: 330-334

3 Richard P Paul, Brice Shimano, Golden E Mayer. Kinematic control equation for simple manipulators. 1981; SMC-11(6)

4 亀谷雅嗣，渡部铁透，河田键一，铁屋克浩. “マルチマイクロプロセフサによるロボット关节角计算の高速化”，日本ロボット学会志，1985; 3 (4): 263-276

5 Lai J C. HSPICE user's manual Honeywell circuit simulation program. Solid State Electronics Division, 1981

6 Kal Hwang. Computer arithmetic. John Wiley & Sons, Inc, 1979

A SUPER-HIGH-SPEED ALGORITHM OF COORDINATE INVERSION FOR SPACE ROBOT

HONG Bingrong LI Lixin JIANG Hui

(Dept of Computer Science, Harbin Polytechnic University, 150006)

Abstract

In this paper, a super-high-speed algorithm of coordinate inversion is outlined, which draws each joint rotative angle from the end-position and posture of manipulator. This algorithm is based on the CORDIC (Coordinate Rotation Digital Computation) algorithm. Its basic idea is to convert the coordinate inversion to the combination of coordinate rotation and arctangent which is implemented with the CORDIC algorithm rapidly. The most obvious characteristics of the algorithm is converting the sophisticated, irregular and non-standard function computation in coordinate inversion to regular computation. In this way a high-speed and special purpose computer based on the algorithm can be realized easily.

Key words: manipulator CORDIC algorithm coordinate inversion coordinate rotation arctangent