NBSIR 88-3735

COORDINATED JOINT MOTION FOR AN INDUSTRIAL ROBOT

John L. Michaloski

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Manufacturing Engineering
Robotics Systems Division
Gaithersburg, MD 20899

March 1988

Coordinated Joint Motion Control for an Industrial Robot

John Michaloski
National Bureau of Standards
Gaithersburg, MD

ABSTRACT

The Coordinated Joint Level (CJT) of the Real Time Control System (RCS) for the Cincinnati Milicron T3 is used as a part of the Automated Manufacturing Research Facility (AMRF). This paper is divided into four sections covering the main components of the Coordinated Joint Control level including the forward kinematics, the reverse kinematics and smoothing. A brief discussion of the RCS robot representation is presented as well as some problem areas within the coordinated joint level.

1.0 INTRODUCTION

The Cincinnati Milicron "The Total Tool" (herein referred to as the T3) is a 6 degree of freedom robot. At NBS the T3 is used in the AMRF [13] as a transfer device that picks objects from trays on carts and places into machine tools and vice versa after machining is completed. As a transfer device, the T3 goes through a variety of motions encompassing a variety of positions, orientations, velocities, and accelerations, collectively known as the kinematics. Gracefully controlling these kinematic functions is the concern of the coordinated joint level.

At a basic level of control, the T3 can take a commanded position and orientation. However, the T3 provides no finer control of the details of motion, such as velocities and accelerations of the individual joints. Without this capability, the robots apparent motions at times appear to be jerky and uncoordinated. In addition, if the user supplies a change in position and orientation that exceeds the allowable joint limits, the T3 robot completely shuts down. This lack of a more sophisticated degree of control is unacceptable for a robot engaged in a sensory-interactive real-time control.

The coordinated joint level module short-circuits the direct position and orientation communication link between the T3 robot controller and the RCS robot controller. The T3 coordinated joint level provides finer kinematic control of the next robot position and orientation by monitoring the change of robot joint values over time and scaling joint moves to establish a new scaled robot position and orientation.

The coordinated joint module is composed of three important components.

1) The backward solution or inverse kinematics is concerned with finding T3 joint angles that will put the end-effector in the given xyz Cartesian position and orientation.

2) Scaling is performed on the joint angle velocity and acceleration to achieve smoother motion, while at the same time providing finer control over the robot.

3) The forward solution takes as input joint angles (in this case scaled) and with a series of coordinate frame transformation matrices calculates a robot end-effector Cartesian xyz position and orientation.

Figure 1 illustrates the overall function of the coordinated joint module.

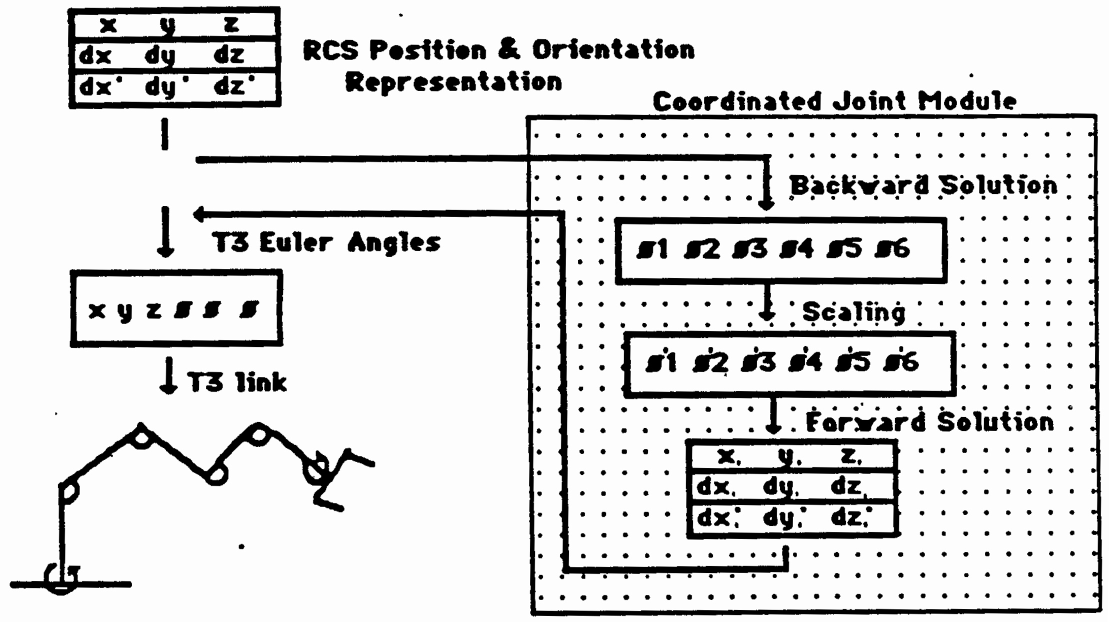


Figure 1. Coordinated Joint Module

Within the scope of this document, many aspects concerning control of the T3 will be discussed. First, an overall perspective of the physical aspects of the T3 robot will be reviewed. Then, the translation from RCS robot end-effector representation into a T3 Euler angles representation will be studied. This preliminary discussion provides a background from which the actual forward and reverse kinematics can be developed. The forward solution or direct kinematics from joint angles into a Cartesian xyz position and orientation is straightforward and will be discussed first because it is the simpler of the two transformations. The backward solution, or inverse kinematics, will be broken down into a discussion of necessary geometry and trigonometry used in developing the wrist pitch point and then each joint angle will be derived. After the derivation is completed, a look at the problem areas within the solution will be explored, including singularities, degenerate cases that give multiple solutions, and any ill-conditioning within the floating point arithmetic.

2.0 ROBOT REPRESENTATION

A robot can be defined as a manipulator consisting of inter-connected links with motors at the joints. Further, the motors can provide a range of joint positions so that the robot is capable of program guidance through a volume of motions. The T3 robot is a hydraulically powered six degree of freedom robot. The six degrees of freedom includes a base swivel, a shoulder rotate, an elbow rotate, a pitch rotate, a yaw rotate and a roll. Each joint is connected by a rigid body link. In this paper, the six links are referred to by a two letter convention that is a combination of the first letter the start of the link and the first letter of the end of the link. This gives the following link notation.

wb	:	world to base
bs	:	base to shoulder
se	:	shoulder to elbow
py	:	pitch to yaw
yr	:	yaw to roll
rt	:	roll to tool
tf	:	tool to finger

The wrist point of the T3 robot is identified as the point at the end of the yaw to roll link. Figure 2 illustrates these linkages.

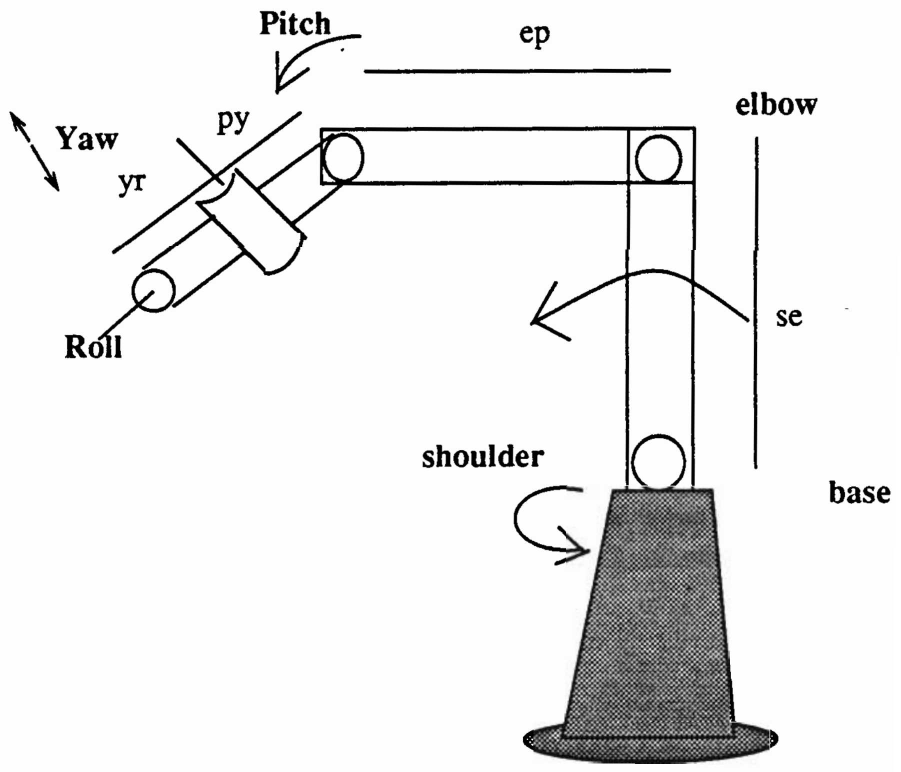


Figure 2. T3 Linakge Representation

RCS supplies an xyz Cartesian position given in tics and 2 delta xyz directional unit vectors given as a scaled integers as input to the T3 coordinated joint level. (The derivation from RCS to T3 will be discussed later.) The conversion factor from RCS to T3 or world space for the Cartesian xyz position is 100.0 tics per inch. The unit vectors along the x-axis and y-axis are given as integers with a scaled magnitude to normalize. Thus, the following equation converts an integer unit vector component to a floating point value.

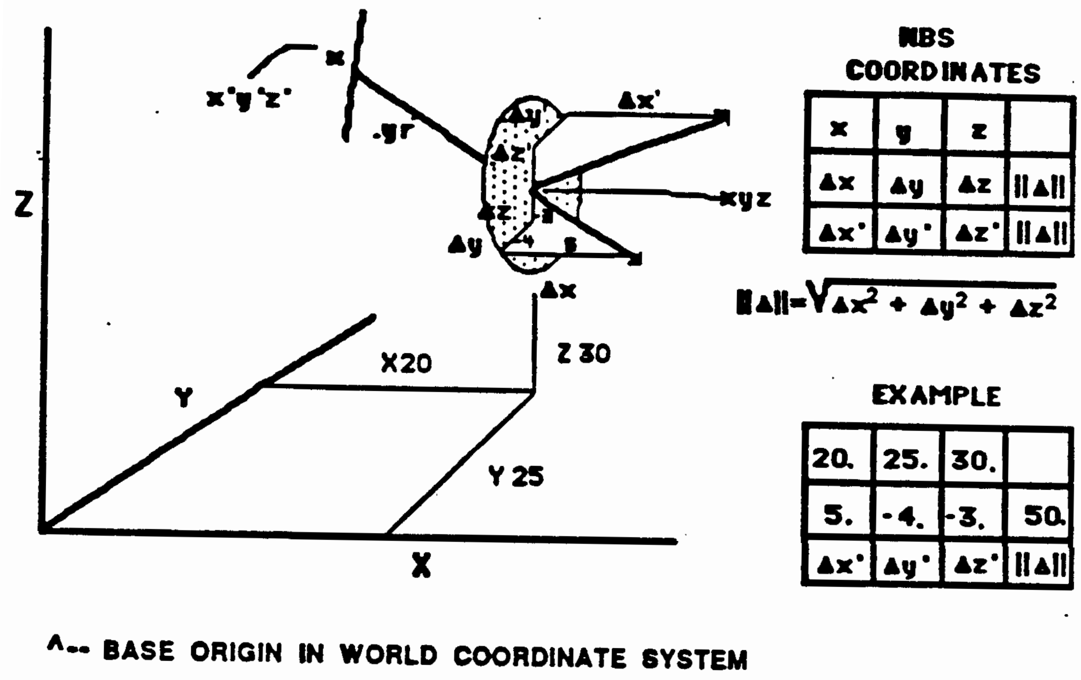
$$\text{magnitude} = \text{SQRT}(x^2 \text{integer} + y^2 \text{integer} + z^2 \text{integer})$$

$$\text{xfl} = \text{xinteger} / \text{magnitude}$$
$$\text{yfl} = \text{yinteger} / \text{magnitude}$$
$$\text{zfl} = \text{zinteger} / \text{magnitude}$$

The T3 also represents angular movement and arm reach in tics. The T3 angular tic identity is 212 tics/radian. The tic to inches identity for the T3 is 102.4 tics/inch. This gives an arm reach of 16,415.74 tics.

2.1 RCS POSE REPRESENTATION

A robot is modeled as a set of interconnected linkages with motors at the joints. The motors at the joints are powered and can take on a range of values so that the spatial links rotate in space are known as revolute joints. Connections between links that are linear, i.e. extend or shrink, are known as prismatic joints. Each joint gives the robot another degree of freedom, i.e. the number of ways a point may independently move through space. The change in the spatial relationships of the connected links over time gives the physical attribute of motion. To command a robot through a task, a control system must generate a series of point and an orientation pairs or *poses* for the robot over time. In the Real Time Control System (RCS) developed at the National Bureau of Standards, this point and orientation pair are represented as an xyz vector in the World Cartesian Space, plus two unit vectors signifying direction. This representation is sufficient to describe all possible robot poses. Other representations which model robot poses include a Cartesian position plus three Euler angles or a quaternian consisting of a axis and an angle of rotation around the angle to determine the orientations. Two unit vectors offer the advantage of a conceptually simple approach to describing typical straight line fine motions along the xy axis that robots commonly require. The illustration below details this pose information. (The illustrations uses numeric components that are normalized by the vectors magnitude.)



NBS COORDINATES

x	y	z	
Δx	Δy	Δz	‖Δ‖
Δx'	Δy'	Δz'	‖Δ‖

$$\|\Delta\| = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$$

EXAMPLE

20.	25.	30.	
5.	-4.	-3.	50.
Δx'	Δy'	Δz'	‖Δ‖

Figure 4. RCS ROBOT POSITION AND ORIENTATION

In defining data in RCS, a pose is the most basic description of an end-effector location and orientation. To simplify notation, the triple vector pose can be mapped into common reference points on the robot end-effector that allows easy description of common robot motions required for end-effector positioning and motion. The position element of the triplet is known as the *wrist point*. By projecting the directional unit vector along the x-axis the end-effector length, the *tool point* can be obtained. By projecting the directional unit vector along the y-axis from the tool point, the *finger point* can be obtained.

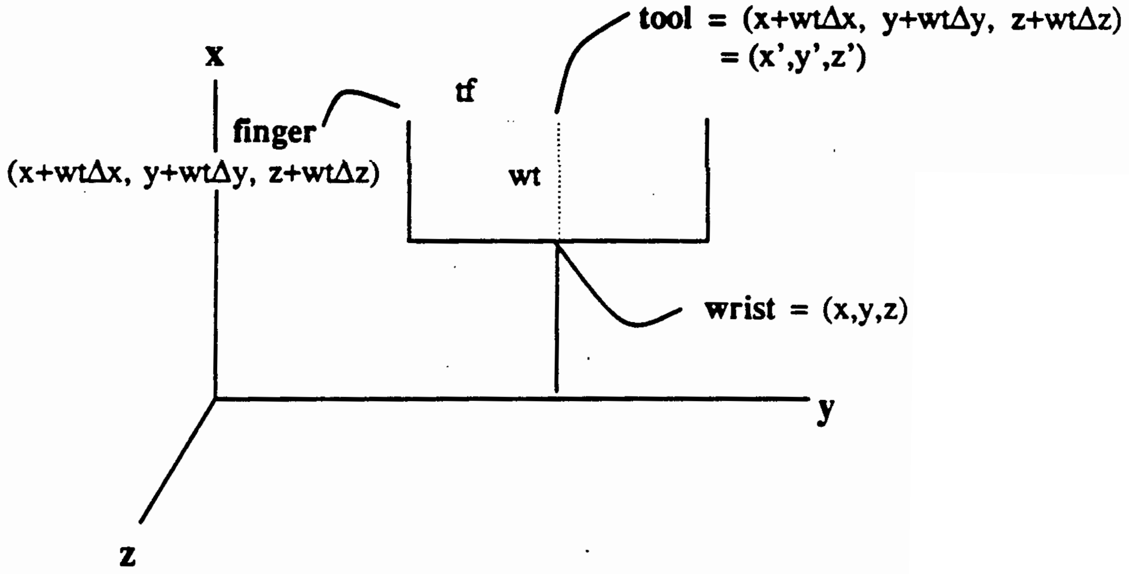


Figure 5. End-effector Naming Conventions

Planning paths of motion the robot will take when moving between Cartesian positions and orientations in RCS is dictated by moving one point on the current end-effector position in a straight line to its corresponding goal point end-effector position. Given the RCS representation of a Cartesian position and two directional unit vectors defining orientation, then a trajectory path between the current position and the goal position is defined as a straight line for one of the vector end-points while the other two points travel an arc path to reach their goal points. For example, suppose the tool point is translated along a vector defined within the work space.

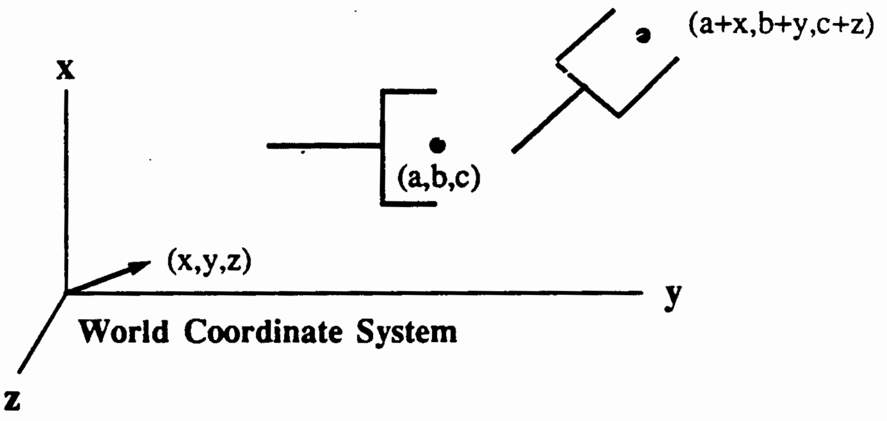


Figure 6. End-effector Motion

2.1 RCS Kinematic Model

To achieve a pose in a robot, the Cartesian representation must be mapped into a spatial relationship among the linkages, i.e. joint angles. The kinematics of a robot are concerned with translating robot joint angles to a Cartesian position and orientation at the end-effector and vice versa. In order to describe the relationship between joint links, coordinate frames are assigned to each link. Coordinate frames are a method of representing rotations and translations of a rigid body (i.e. a robot linkage) about a fixed point.

Starting with the base frame, each joint coordinate frame then undergoes a rotation and possible translation to establish the relationship between successive coordinate frames. This series of transformations establishes the position and orientation of the end-effector. By chaining these transformations together, the relationship between the base world coordinate frame and the end-effector coordinate frame is established.

$$ {}^{0}T_{6} = {}^{0}T_{1}\, {}^{1}T_{2}\, {}^{2}T_{3}\, {}^{3}T_{4}\, {}^{4}T_{5}\, {}^{5}T_{6} $$

The NBS RCS representation can be given as such a transformation matrix. To derive the transformation matrix from the RCS representation, the first unit column vector is used as X, the second unit column vector as Y and the cross product of the x and y unit vectors to give the Z direction column vector in the following matrix.

$$ {}^{0}T_6 = | \; X^T \quad Y^T \; Z^T \; | $$

This matrix establishes the orientation of the end-effector. Using a homogeneous matrix representation, the translation amount from the origin can be incorporated into the matrix. The translation amount along each axis, is simply, the xyz point. This gives the following homogeneous matrix equivalent of the RCS representation.

$$ {}^{0}T_6 = \begin{bmatrix} X & Y & Z & x \\ & & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[\begin{array}{ccc|c} X & Y & Z & p \\ \hline 0 & 0 & 0 & 1 \end{array} \right] $$

Once this matrix has been derived, translation to other representations is possible. To equate to other representations the following matrix elements serve as a guide for angle solutions.

$$ \begin{bmatrix} RCS_{1,1} & RCS_{1,2} & RCS_{1,3} \\ RCS_{2,1} & RCS_{2,2} & RCS_{2,3} \\ RCS_{3,1} & RCS_{3,2} & RCS_{3,3} \end{bmatrix} \begin{array}{l} \text{where } X^T = (\, RCS_{1,1} \;\; RCS_{2,1} \;\; RCS_{3,1}) \\ \qquad\quad\; Y^T = (\, RCS_{1,2} \;\; RCS_{2,2} \;\; RCS_{3,2}) \\ \qquad\quad\; Z^T = (\, RCS_{1,3} \;\; RCS_{2,3} \;\; RCS_{3,3}) \end{array} $$

2.2 Translation to Other Pose Representations

Simply specifying a position of the end-effector is not a sufficient description. Many robot poses can achieve the same position. For this reason, an orientation must be specified, and many different orientation descriptions exist. One common robot orientation description uses three Euler angle rotations as a transformation from the base coordinate frame to the orientation coordinate frame. The three Euler angle rotations are a yaw rotate about the z axis by an angle epsilon, then a pitch rotate about the y axis by an angle delta, and then a roll rotate about the x axis by an angle rho. The following diagram illustrated the axis as they are rotated. Each increment in number designates another rotation.

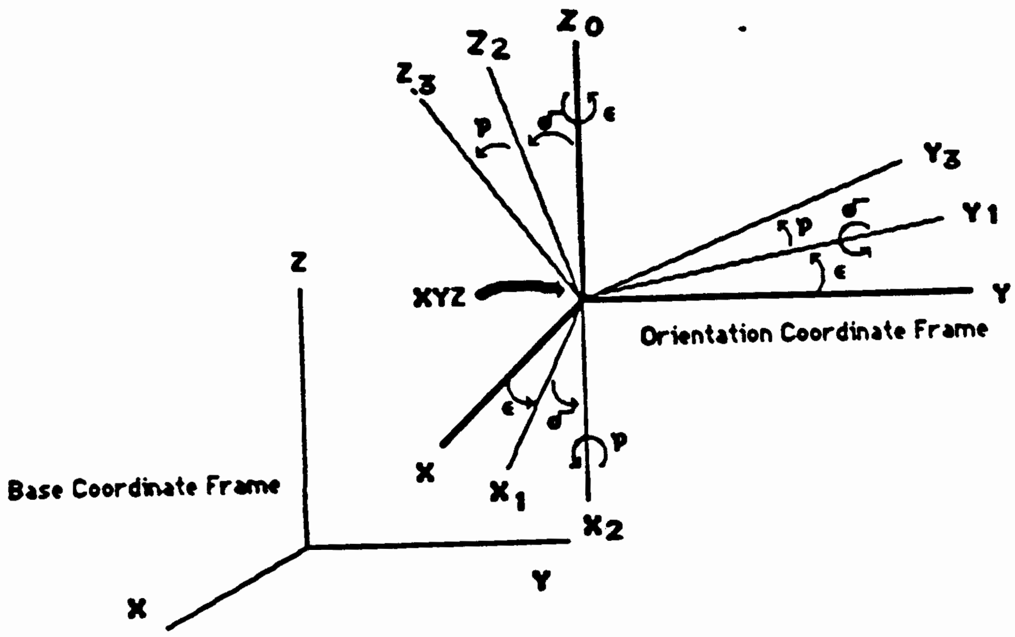


Figure 7. Robot Position and Orientation

Mathematically, these rotations are represented by the following series of matrices.

$^0R_6$ = Rz(ε) Ry(d) Rx(r) where ε=epsilon, δ=delta, ρ=rho.

$$= \begin{bmatrix} c(\varepsilon) & -s(\varepsilon) & 0 \\ s(\varepsilon) & c(\varepsilon) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c(\delta) & 0 & s(\delta) \\ 0 & 1 & 0 \\ -s(\delta) & 0 & c(\delta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\rho) & -s(\rho) \\ 0 & s(\rho) & c(\rho) \end{bmatrix}$$

multiplying these matrices yields,

$$= \begin{bmatrix} c(\varepsilon)c(\delta) & c(\varepsilon)s(\delta)s(\rho) - s(\varepsilon)c(\rho) & c(\varepsilon)s(\delta)c(\rho) + s(\varepsilon)s(\rho) \\ s(\varepsilon)c(\delta) & s(\varepsilon)s(\delta)s(\rho) + c(\varepsilon)c(\rho) & s(\varepsilon)s(\delta)c(\rho) - c(\varepsilon)s(\rho) \\ -s(\delta) & c(\delta)\, s(\rho) & c(\delta)c(\rho) \end{bmatrix}$$

The RCS coordinate frame can be translated into three Euler angles, by equating the matrices derived from the NBS representation to the matrix derived from expanding the 3 Euler rotation matrices. We can now solve for the angles of rotation ε, δ, and ρ using this equality.

$$^0T_6 = {}^0R_6$$

For example, equating the following elements and using algebraic techniques derives the epsilon angle.

$$s(\varepsilon)\, c(\delta) = RCS_{2,1}$$
and
$$c(\varepsilon)\, c(\delta) = RCS_{1,1}$$

Dividing the first equality by the second equality and then taking the arctangent of this value, yields the angle epsilon.

$$\frac{s(\varepsilon)}{c(\varepsilon)} = \frac{RCS_{2,1}}{RCS_{1,1}} \quad \text{so } \varepsilon = ATAN(RCS_{2,1} / RCS_{1,1})$$

The equating of elements method of angle solution has been commonly used to determine the remaining Euler angles. [5]

$$\varepsilon = ATAN(\ RCS_{2,1}\ ,\ RCS_{1,1})$$

$$\delta = ATAN(\ -RCS_{3,1}\ ,\ SQRT(RCS^2_{1,1} + RCS^2_{2,1}\ ))$$

$$\rho = ATAN(\ RCS_{3,2}\ ,\ RCS_{3,3})$$

Thus, we have obtained a translation from RCS that supplies a xyz point equivalent to the center of the wrist plate in the world coordinate or base frame, and three angles of rotation.

These solutions assume that the $\cos(\delta) > 0$, i.e. $-90 < \delta < +90$. For δ at 90 then $\cos(\delta)$ is zero, and a singularity occurs. In this case the $\sin(\rho)$ is one and $\cos(\delta)$ is zero. Only one of the epsilon or delta angles may be computed.

$$= \begin{bmatrix} 0 & c(\varepsilon)s(\delta)s(\rho) - s(\varepsilon)c(\rho) & c(\varepsilon)s(\delta)c(\rho) + s(\varepsilon)s(\rho) \\ 0 & s(\varepsilon)s(\delta)s(\rho) + c(\varepsilon)c(\rho) & s(\varepsilon)s(\delta)c(\rho) - c(\varepsilon)s(\rho) \\ -s(\delta) & 0 & 0 \end{bmatrix}$$

Let $\sigma = -s(\delta) = 1$ giving

$$= \begin{bmatrix} 0 & \sigma\, c(\varepsilon)s(\rho) - s(\varepsilon)c(\rho) & \sigma c(\varepsilon)c(\rho) + s(\varepsilon)s(\rho) \\ 0 & \sigma s(\varepsilon)s(\rho) + c(\varepsilon)c(\rho) & \sigma s(\varepsilon)c(\rho) - c(\varepsilon)s(\rho) \\ \sigma & 0 & 0 \end{bmatrix}$$

Simplifying with cosine laws :

$$= \begin{bmatrix} 0 & \sigma s(\sigma\varepsilon+\rho) & \sigma c(\sigma\varepsilon+\rho) \\ 0 & \sigma c(\sigma\varepsilon+\rho) & \sigma s(\sigma\varepsilon+\rho) \\ \sigma & 0 & 0 \end{bmatrix}$$

From this only one of epsilon and rho may be determined. Let epsilon equal what it was the previous cycle as an approximation, then:

$$\begin{aligned} \varepsilon_{i+1} &= \varepsilon_i \\ \sigma\varepsilon+\rho &= \text{atan2}(-\sigma RCS(2,1), \sigma RCS(2,2)) \\ &= \text{atan2}(\sigma RCS(2,1), \sigma RCS(2,2)) - \sigma\varepsilon \end{aligned}$$

3. FORWARD SOLUTION

Solving for the forward solution of a robot kinematics demonstrates the use of a coordinate frame representation. To solve for the forward solution, the end-effector position and orientation must be derived from the joint angles. This is typically a straightforward solution and can be easily modeled with coordinate reference frames. For example, the Cincinnati Milicron T3 robot is a serial link manipulator with six degrees of freedom. The six degrees of freedom include a base swivel, a shoulder rotate, an elbow rotate, a pitch rotate, a yaw rotate and a roll. Each joint is connected by a rigid body link. In this paper, the six links are referred to by a two letter convention that is a combination of the first letter denoting the start of the link and the first letter of the end of the link. This gives the following notation.

wb	:	world to base
bs	:	base to shoulder
se	:	shoulder to elbow
py	:	pitch to yaw
yr	:	yaw to roll
rt	:	roll to tool
tf	:	tool to finger

The wrist point of the T3 robot is identified as the point at the end of the yaw to roll link.

The relationship between successive links can be modeled via coordinate frame transformations, including rotations and translations. Thus, the end-effector position and orientation relationship to the base reference frame zero can be derived by a series of coordinate frame transformations, and represented by the following equation.

$${}^{0}T_{6} = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3}\,{}^{3}T_{4}\,{}^{4}T_{5}\,{}^{5}T_{6}$$

The Denavit-Hartenberg [5] convention of assigning coordinate frames has been commonly adopted in robotics. [11] Following this convention, all coordinate frames are aligned so that rotation is always done about the z axis. This requires at most 2 rotations and 2 translations. A more expedient method used here is to use one rotation followed by at most two translations in order to bring two frames into coincidence. [9] Then, each transformation T from link$_i$ to link $_{i+1}$ consists of a rotation R around the axis of revolution and a displacement D (or translation) distance down the link.

$${}^{i}T_{i+1} = {}^{i}R_{i+1}\ {}^{i}D_{i+1}$$

The following diagram illustrates the difference of style between robots that require either one or two displacements to bring two frames into coincidence. Robot configurations that have all the joints in coincidence and lie along one axis require only one displacement per coordinate frame alignment, for example, the T3. Other robot configurations require another displacement at the joints for motor housing when the joints are in coincidence and require two displacements for these joints, for example, the Puma 560.

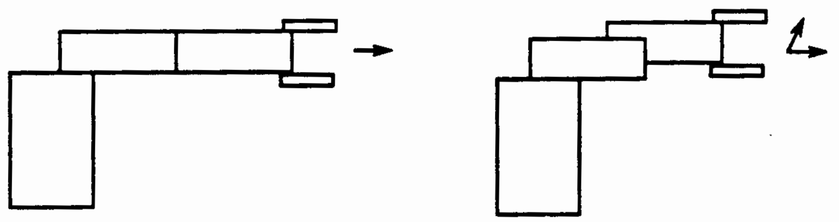


Figure 8. Translation Configurations

Assuming the T3 configuration, the following table outlines the rotations and lengths of displacement for each linkage for each of the T3 coordinate reference frames:

Frame Number	Displacement	X-axis	Y-axis	Z-axis
0	.wb or 0			
1	0			${}^{0}T_{1}$
2	.se		${}^{1}T_{2}$	
3	.ep		${}^{2}T_{3}$	
4	.py		${}^{3}T_{4}$	
5	.yr			${}^{4}T_{5}$
6	.rt	${}^{5}T_{6}$		
7	.tf	only if finger point is needed.		

In these series of transformations, the world coordinate system can be considered to have the World Coordinate space origin at the base of the robot in which a displacement up to the base joint is unnecessary. The choice depends on how the world coordinate frame is to be interpreted. In this paper, the world coordinate system is chosen at the shoulder giving one less translation.

Given these series of transformations ${}^{0}T_{6}$, a point in the end-effector coordinate frame, (i.e. the yaw to roll displacement along the x axis), can be transformed into the base coordinate reference frame.

$$[x \ \ y \ \ z]^{T} = {}^{0}T_{6}[rr \ 0 \ 0\,]^{T}$$

The forward solution is concerned with deriving the end-effector position and orientation in the base coordinate frame. It is composed of transformations that include displacement lengths between T3 joints which are constant, and the angles between linkages which supply the joint rotations. These series of rotations around the x,y, and z axes, and the displacement along the x,y, and z axes, can be conveniently represented by homogeneous rotation matrices, where dx is displacement along the x axis, dy is displacement along the y axis, and dz is displacement along the z axis.

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & c\theta & -s\theta & dy \\ 0 & s\theta & c\theta & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & -sq & dx \\ 0 & 1 & 0 & dy \\ s\theta & 0 & c\theta & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & -s\theta & 0 & dx \\ s\theta & c\theta & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Expanding the ${}^{0}T_{6}$ series of rotations and displacements using the homogeneous matrix notation give the following equation.

$${}^{0}T_{6} = \begin{bmatrix} c1 & -s1 & 0 & 0 \\ s1 & c1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c2 & 0 & s2 & 0 \\ 0 & 1 & 0 & 0 \\ -s2 & 0 & c2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c3 & 0 & s3 & se \\ 0 & 1 & 0 & 0 \\ -s3 & 0 & c3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c4 & 0 & s4 & ep \\ 0 & 1 & 0 & 0 \\ -s4 & 0 & c4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c5 & -s5 & 0 & py \\ s5 & c5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & yr \\ c6 & s6 & 0 & 0 \\ 0 & -s6 & c6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \quad {}^{0}T_{1} \qquad {}^{1}T_{2} \qquad {}^{2}T_{3} \qquad {}^{3}T_{4} \qquad {}^{4}T_{5} \qquad {}^{5}T_{6}$$

By multiplying these matrices from right to left (post-multiplied) we can quite simply obtain the ${}^{0}T_{6}$ transform matrix. However, if the intermediate Cartesian joint positions are required for graphics, or some other debugging purpose, the matrices must be multiplied from left to right (pre-multiplied) to obtain the intermediate Cartesian joint positions. This method involves saving the total effect of the rotations , and using a new origin for translation as the translation moves down the links of the manipulator.

With the base joint Cartesian position at the origin (0,0,0), there is no translational amount and the first link, the elbow joint Cartesian position, can be computed as the transformation of the shoulder to elbow joint displacement, *se*, into the base coordinate reference frame. This transformation consists of the combination of the base and shoulder rotations.

$$\begin{bmatrix} \text{x-elbow} \\ \text{y-elbow} \\ \text{z-elbow} \\ 1 \end{bmatrix} = \begin{bmatrix} c1 & -s1 & 0 & 0 \\ s1 & c1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c2 & 0 & s2 & 0 \\ 0 & 1 & 0 & 0 \\ -s2 & 0 & c2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} se \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} c1 & -s1 & 0 & 0 \\ s1 & c1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} se\,c2 \\ 0 \\ -se\,s2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} se\,c1\,c2 \\ se\,s1\,c2 \\ -se\,s2 \\ 1 \end{bmatrix}$$

To obtain the pitch Cartesian point, the origin of our reference frame is no longer (0,0,0), but the result of the transformation from the shoulder to the elbow point, (x-elbow, y-elbow, z-elbow). Then, the elbow to pitch displacement, *ep*, along the x-axis will be transformed back to the base coordinate frame that includes three joint rotations with a translation to the origin (x-elbow, y-elbow, z-elbow) of a new coordinate space.

$$\begin{bmatrix} \text{x-pitch} \\ \text{y-pitch} \\ \text{z-pitch} \\ 1 \end{bmatrix} = \begin{bmatrix} c1 & -s1 & 0 & 0 \\ s1 & c1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c2 & 0 & s2 & 0 \\ 0 & 1 & 0 & 0 \\ -s2 & 0 & c2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & se \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c3 & 0 & s3 & se \\ 0 & 1 & 0 & 0 \\ -s3 & 0 & c3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ep \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} r1 & r2 & r3 & se\,c1\,c2 \\ r4 & r5 & r6 & se\,s1\,c2 \\ r7 & r8 & r9 & -se\,s2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ep \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} r1 & r2 & r3 & \text{x-elbow} \\ r4 & r5 & r6 & \text{y-elbow} \\ r7 & r8 & r9 & \text{z-elbow} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ep \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

For subsequent joint Cartesian positions, this sequence is repeated of calculating a new origin or translational amounts based on the previous Cartesian joint position and the additional rotation added to the rotation part of the homogeneous matrix.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} = \left[\begin{array}{c|c} {}^{0}R_i \cdot {}^{i}R_{i+1} & \begin{matrix} x_i \\ y_i \\ z_i \end{matrix} \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right] \begin{bmatrix} \text{x-axis}_{\text{displacement}} \\ \text{y-axis}_{\text{displacement}} \\ \text{z-axis}_{\text{displacement}} \end{bmatrix}$$

4.0 BACKWARD SOLUTION

The backward solution or inverse kinematics is concerned with deriving robot joint angles to attain an xyz Cartesian position and orientation for the end-effector. Numerous closed mathematical solutions exist for robots with a spherical wrist, that is, a robot with the first three joints giving the wrist position, and the final three joints giving the orientation of the end-effector.

Several problems occur when the robot does not have a spherical wrist. The Cincinnati Milicron T3 is such a robot. The kinematics of this robot requires working backward from the tool plate to the yaw point to derive the wrist point. With the wrist or pitch point, the closed mathematical solutions for the first three joints similar to other robots is obtainable. In addition, with the wrist point, the final three joint angles can be calculated. A series of geometric projections and scalings are one alternative in obtaining the wrist point and hence a backward solution.

First, a set of positions to describe the robot are necessary.

xyz - is the position of the tool plate and is given.

x'y'z' - is the position of the yaw position and can be derived from the initial x axis directional unit vector with the following calculation.

$$(x'y'z') = (x-.yr\Delta x/\|\Delta\|,\ y-.yr\Delta y/\|\Delta\|,\ z-.yr\Delta z/\|\Delta\|)$$

where : .yr is the length of the yaw offset
$\|\Delta\|$ is the magnitude of the direction vector

x"y"z" - is the position of the wrist plate and is obtained geometrically via a series of projections and scalings.

$\underline{xyz}$ - (xyz bar or underscore) is the projection of the tool plate point onto the x'y'z'-x''y''z'' vector. It is the intermediate point used in calculating x''y''z''.

Given x'y'z', we can determine the joint one angle θ1 via this equation :

$$\theta 1 = \mathrm{ARCTAN}(y'/x')$$

Then the goal of the transformation is to have the line xyz-x'y'z' projected onto the line x'y'z'-x"y"z". The transformation from x'y'z' to x"y"z" problem is broken down into two parts. First the projection <u>xyz</u> in the xy plane alone is derived.

To determine the point <u>xyz</u>, the projection onto the x'y'z'-x"y"z" vector, we need to calculate, the distance along the x and y axis from x-x' and y-y'.

$\Delta x' = x - x'$	
$\Delta y' = y - y'$	
$\beta = \mathrm{ARCTAN}(\Delta y' / \Delta x')$	
$fl = \mathrm{SQRT}(\Delta x'^2 + \Delta y'^2)$	the xy-x'y' length
$nd = fl * (\cos(\beta - \alpha))$	the projection length

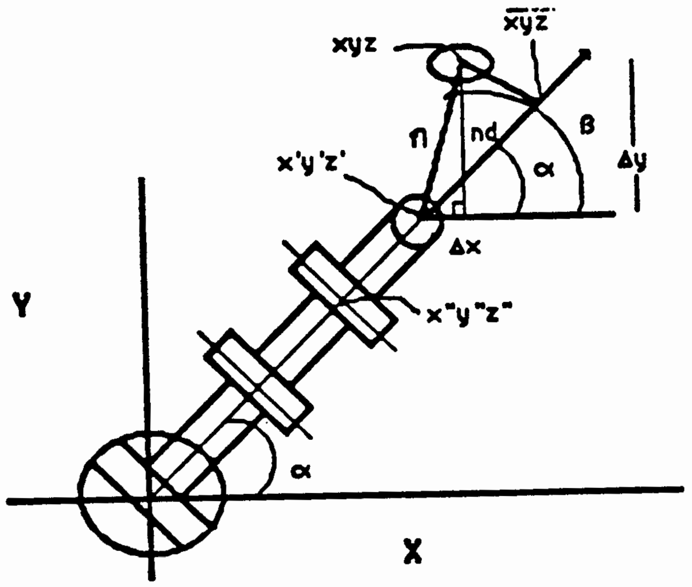


Figure 4.1 Projection Length Calculation- Top View

Then the projection point xyz can be calculated.

$$\underline{x} = x' + (nd * x') / \mathrm{SQRT}(x'^2 + y'^2)$$
$$\underline{y} = y' + (nd * x') / \mathrm{SQRT}(x'^2 + y'^2)$$
$$\underline{z} = z$$

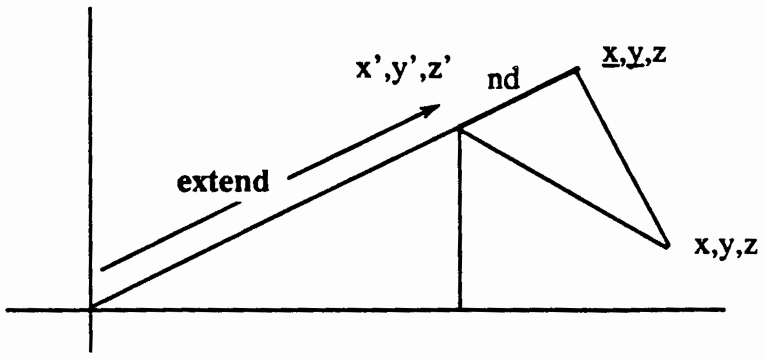


Figure 4.2 Projection Point Scaling

$$\Delta x' \rightarrow \underline{x} = (nd * x') / SQRT(x'^2 + y'^2))$$
$$\Delta y' \rightarrow \underline{y} = (nd * y') / SQRT(x'^2 + y'^2))$$

$$\underline{x} = x' + \Delta x'$$
$$\underline{y} = y' + \Delta y'$$

To calculate the wrist point x"y"z", we use the previous calculations of the projection of the xyz-x'y'z' vector onto the x'y'z'-x"y"z" vector. With this information, scale back from <u>xyz</u>-x'y'z' the yaw length (.yr) distance to determine the wrist point.

$$x'' = x' - .py\ (x'-\underline{x}) / SQRT((x'-\underline{x})^2 + (y'-\underline{y})2 + (z'-\underline{z})^2)$$

$$y'' = y' - .py\ (y'-\underline{y}) / SQRT((x'-\underline{x})^2 + (y'-\underline{y})^2 + (z'-\underline{z})^2)$$

$$z'' = z' - .py\ (z'-\underline{z}) / SQRT((x'-\underline{x})^2 + (y'-\underline{y})^2 + (z'-\underline{z})^2)$$

With the wrist point, joint angles two, three, four, and five can be calculated geometrically. The kinematic model is diagramed on the next page with the origin (0,0,0) at the shoulder. A two part procedure is used to determine joints θ2 and θ3.

First, the pivot amount in the z plane is calculated.

$$\beta = ARCTAN(z''/ SQRT(x''^2 + y''^2))$$

Second, the shoulder-elbow-pitch triangle is established to determine the xy angle transformation. Because the shoulder to elbow, se and the elbow to pitch, py links are equal, this triangle is isosceles so that a perpendicular bisecting line segment can be used to partition the triangle into two back-to-back right triangles. With these back-to-back right angle triangles, some simple trigonometry can be used to determine the angles of the larger triangle.

The arctangent of the height divided by the base will yield the angle alpha. The base of the right triangle is one-half the norm or distance from (0,0,0) to the wrist pitch point xyz.

$$b = 1/2 \text{ SQRT}(x''^2 + y''^2 + z''^2)$$

Since the base of the triangle has been calculated and the hypotenuse of the triangle is the length of the shoulder to the elbow is given, the height of the triangles can be determined using the Pythagorean Theorem :

$$h = \text{SQRT}(.se^2 - b^2)$$

Now the angle alpha can be determined using the arctangent of the height of the right triangle divided by the base.

$$\alpha = \text{ARCTAN} (h/b)$$

Using the fact that the sum of angles in a triangle equals 180?, the apex phi of the larger triangle can be determined.

$$180° = \alpha + \alpha + \phi$$

Giving $\phi = 180° - 2 (\alpha)$. Joint angles two and three can be determined. Joint 2 is the sum of the pivot up z plus the angle alpha. Joint three is the adjacent colinear angle to phi, and is simply 180° minus ϕ. The following figure summarizes the kinematic model.

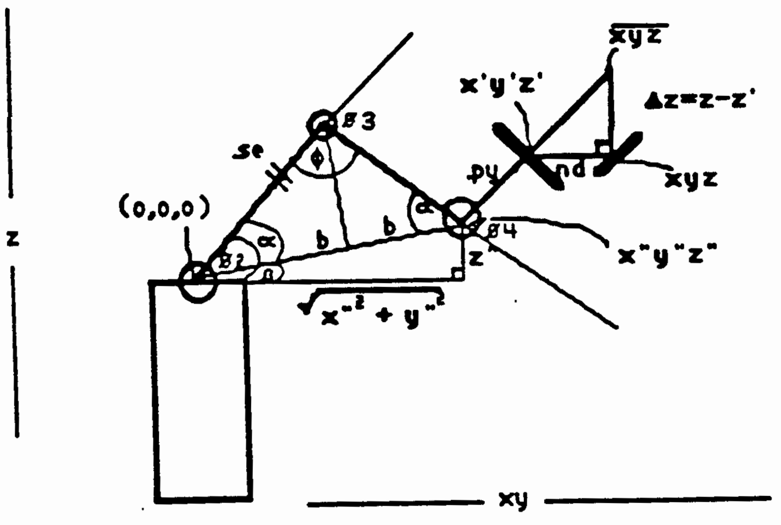


Figure 4.3 Kinematic Model

Intermediate joint angle calculations summary :

$$\beta = \text{ARCTAN}(z''/\ \text{SQRT}(x''^2 + y''^2))$$
$$b = 1/2\ \text{SQRT}(x''^2 + y''^2 + z''^2)$$
$$h = \text{SQRT}(.se^2 - b^2)$$
$$\alpha = \text{ARCTAN}(h/b)$$
$$\phi = 180 - 2 \qquad \text{(because of isosoles triangle)}$$

Once these intermediate values have been determined, actual joint angles can be calculated.

$$\theta 2 = \alpha + \beta$$

$$\theta 3 = 180 - \phi$$

Joint angle θ4 can be determined using the intermediate calculations derived from computing the x''y''z'' wrist point.

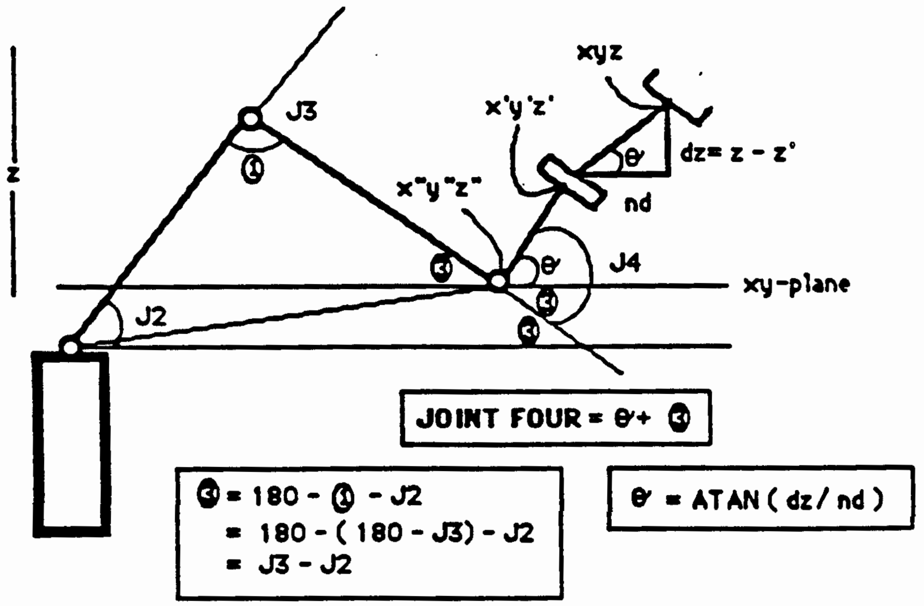


Figure 4.4 Joint Four Calculation

Thus, joint four is computed as follows :

$$\theta 4 = \theta 3 - \theta 2 + \text{ATAN}(dz/nd)$$

Joint angle θ5 can be determined using the intermediate calculations derived from computing the x''y''z'' wrist point.

$$xxx = SQRT((\underline{x}-x')^2 + (\underline{y}-y')^2 + (\underline{z}-z')^2)$$

$$yyy = SQRT(.yr^2 - xxx2)$$

$$\theta 5 = ARCTAN(yyy/xx)$$

Unfortunately, the action of squaring then taking the square root removes the availability of a sign from either xxx or yyy to determine the sign of the angle of rotation. For this reason, the z component of the cross product between the x directional unit vector and the normalized x'y'z'-x''y''z'' vector will provide the sign of the angle. This is because the cross product produces a positive z othogonal vector within a right handed coordinate system whenever the angle between the two crossed vectors is positive.

$$(sx, sy, sz) = (\Delta x_x, \Delta y_x, \Delta z_x) \quad X \quad (x'y'z'-x''y''z'')$$

$$\theta 5 = SIGN(sz) * \theta 5$$

4.1 Joint 6 : Roll Angle

The straightforward mathematical solution in solving for joint six, is to use the ${}^0T_5$ transformation matrix with the wrist point and then equate to ${}^0T_5$ and finger point derived from the y-axis directional unit vector. Using equating elements, a solution can be derived.

$${}^0T_5 [x y z] = {}^5T_6 [x_f y_f y_f]$$

However, this solution is a time-consuming task requiring the ${}^0T_5$ calculation. With so much information already available and the time-constraint imposed by real-time control, the more involved, but computationally more efficient, geometric approach is used.

The geometrical strategy in solving for joint six works back down the arm joint angles, (yaw, pitch, ...) until a non-zero joint angle is found. The first non-zero joint angle will supply a vector that is non-colinear with the directional unit vector along the x axis. The cross product from these two vectors will produce a vector that is normalized to the directional unit vector along the y axis. Once this relationship between these vector is established, then the same bisected triangle method used as an intermediate step in deriving joint angles two and three will yield the roll joint angle.

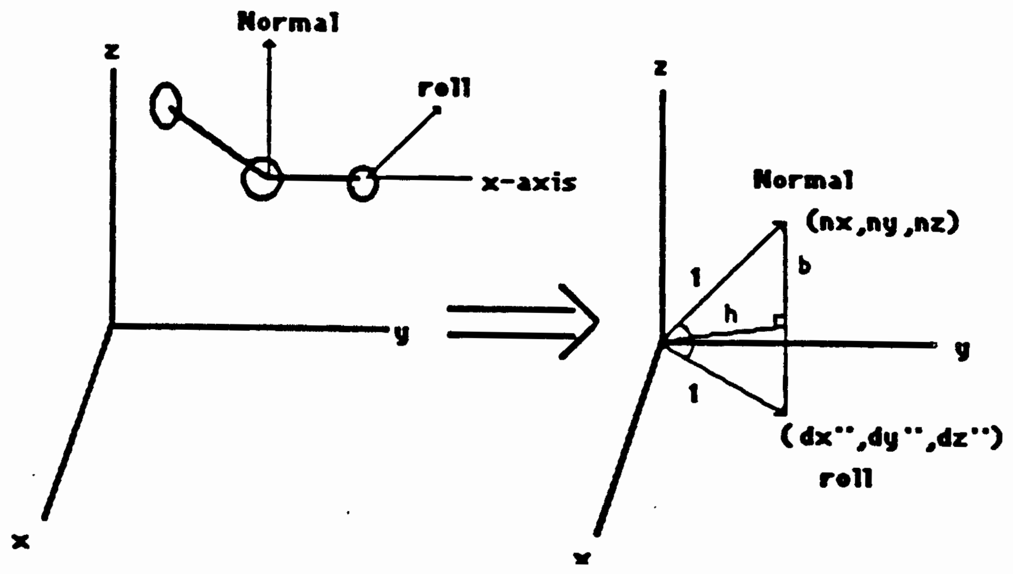


Figure 4.5 Geometric Strategy for Joint 6

If the yaw angle is non-zero, then this involves taking the same cross product used in determining the sign of the yaw joint angle. This cross product will produce a normalized vector along the z axis. It is important the difference vector be normalized and that the sign of the yaw is used in deciding the order of crossing the two vectors.

```
if yaw positive
  then (dx, dy, dz) = $(\Delta x_x, \Delta y_x, \Delta z_x)$  X  $\|(x''y''z''-x'y'z')\|$
  else (dx, dy, dz) = $(\Delta x_x, \Delta y_x, \Delta z_x)$  X  $\|(x'y'z'-x''y''z'')\|$
endif
```

If the yaw angle is zero, we need to compute the shoulder Cartesian point, (dx,dy,dz) using the forward solution techniques discussed earlier. If the pitch angle is non-zero, the difference vector from the shoulder to the elbow point x''y''z'' is used as the vector to cross with the x-axis directional unit vector.

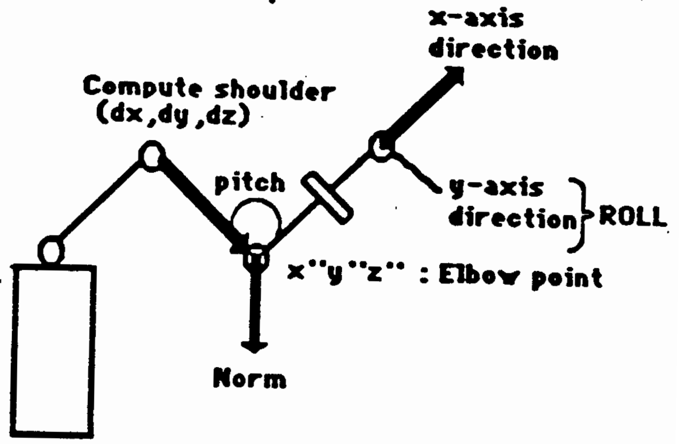


Figure 4.6 Pitch as Normalizing Vector

If the pitch angle is zero, the physical limitations of the T3 are exploited. Because of the physical limitations of the elbow joint, the elbow angle will never be zero. So, the shoulder point (dx,dy,dz) provides a non-linear vector to cross with the x-axis directional unit vector.

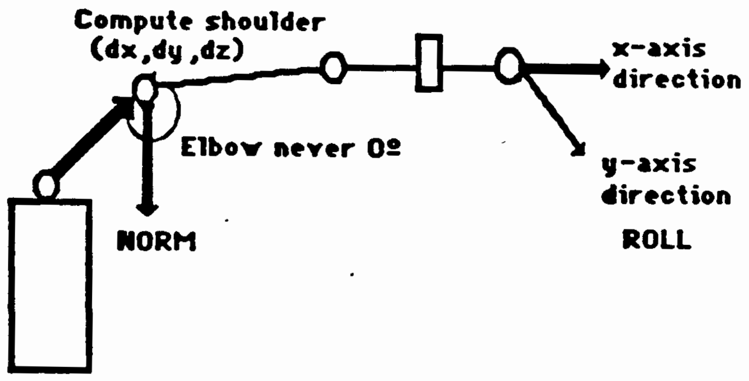


Figure 4.7 Shoulder as Normalizing Vector

This gives the following algorithm for a zero yaw.

```
"compute shoulder point (dx,dy,dz)"
if pitch non-zero
  then (dx,dy,dz) = (dx,dy,dz) - ( $\Delta x$, $\Delta y$, $\Delta z$)
  else (dx,dy,dz)
endif
```

Once a non-colinear vector has been crossed with the x-axis directional unit vector to give the normalized vector along the z-axis, the computed roll angle can be calculated. Because the roll angle will be determined used the bisecting triangle method, the squaring and square root sequence will cause all sides of the triangle to be positive. The leads to the subsequent loss of the roll angle sign.

With the yaw non-zero, there are two possible scenarios that can exist to determine the roll sign. The angle between the finger and the normal vector can be positive and out of sync by a difference of 90°. Or the angle between the finger and the normal can be negative and out of sync by 90°.

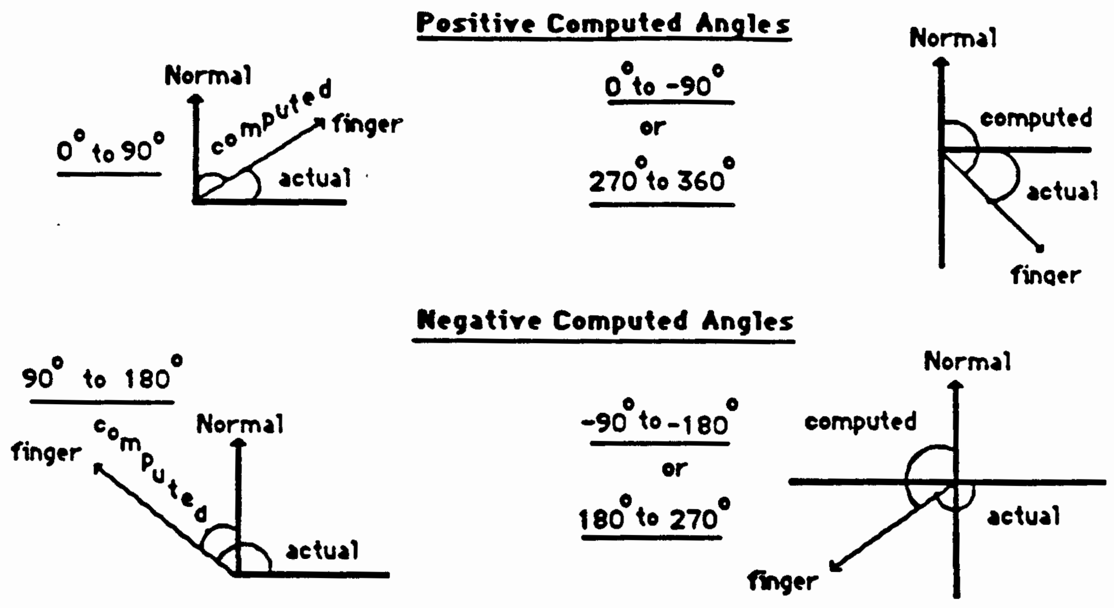


Figure 4.7 Computing Angles

The sign of the angle can be derived using the dot product between the finger vector and the norm vector. In the following algorithms .DOT. will denote a dot product operator between two vectors, producing a scalar result. Ignoring the normalizing factor of the dot production, the cosine of this dot production will yield the positive or negative orientation of the bisecting triangle angle method.

$$\cos(\phi) = (\text{NORM .DOT. FINGER}) / (\|\text{NORM}\| \times \|\text{FINGER}\|)$$
$$\in (\text{NORM .DOT. FINGER})$$

This leads to the following algorithm for the case when the yaw angle is non-zero.

```
if NORM .DOT. FINGER > 0
   then  90° - angle --> angle          "difference from 90°"
   else  90° + angle --> angle   "add 90°"
endif
```

Illustrated by the following diagram.

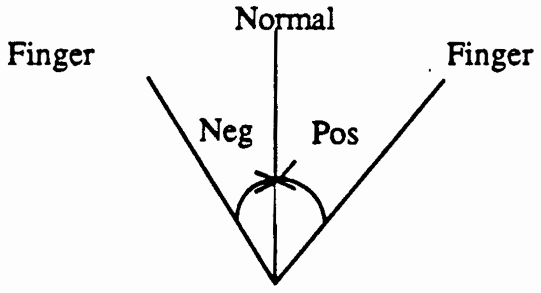


Figure 4.8 Finger Angle Determination

For the cases when the yaw is zero, the normalized vector is not 90< out of sync. The bisecting triangle angle method does not determine the sign of the angle, so that a positive dot product between the normal vector and the finger vector means a negative angle between the finger and the normal vector.

```
if  yaw = 0°
  then if   NORM .DOT. FINGER
          then  roll  =  -roll
  endif endif
```

5.0 COORDINATED JOINT MOTION

The coordinated joint level of processing offers a finer degree of control over the actions of the robot. Maximum joint velocities and accelerations are specified to smooth the motion of the robot. These maximum joint restrictions can be dynamically adjusted to meet performance requirements. For example, within a tight error tolerance near a machine tool, robot motion must be exacting and thus the parameters must be scaled back to limit robot play. In an open area, performance is less critical and thus the parameters can be loosened.

All of this performance is based on the robot motion over time. The delta change in one delta time period T from one configuration to the next provides the velocity for each joint. The change in deltas provides the accelerations for each joint.

$$\text{velocity}_i = (\text{joint}_i - \text{joint}_{i-1})\,\Delta T$$

$$\text{acceleration}_i = (\text{velocity}_i - \text{velocity}_{i-1})\,\Delta T$$

The coordinated joint process inputs a commanded robot xyz position and orientation. The joint angles required to achieve this position and orientation are calculated. Then a workspace envelope test compares the calculated joint positions with each corresponding joint's upper and lower limit. Should the working envelope boundaries be crossed, the current implementation halts processing so that the user can study the trail of events that led to this undesirable robot configuration. The workspace envelope is software determined, and is typically set as the hardware limits of the joints.

After completing the boundary testing, the calculated joint angles are then used to determine joint velocity and acceleration. These values are compared with the performance parameters max-velocity and the max-acceleration for each joint. Based on current set of performance requirements for each joint, scaling may be performed on the joint changes, i.e. velocities, or accelerations if a maximum value has been exceeded. The delta deltas could also be scaled but were not for this implementation. Because the acceleration depends on velocity, acceleration is scaled and then the velocity is scaled before computing the new joint angles. This leads to the following scaling algorithm.

First, scaling is performed on the accelerations of the joints. The amount of scaling on the acceleration is based on the largest percentage any one of the joints exceeds its acceleration maximum. Should none of the joints exceed their acceleration limit, no scaling of the acceleration is required. If an acceleration maximum has been exceeded, the joint maximum acceleration divided by the joint acceleration producing the smallest percentage is used as the scaling amount. All the joints accelerations are scaled back the percentage to stay within the acceleration performance tolerance. With the new set of accelerations, a new joint velocity can be computed for each joint.

With a possible set of new velocities, the amount of scaling on the velocity is based on the smallest percentage less than one produced when any one of the joints exceeds its velocity maximum. Should none of the joints exceed their velocity maximum, no further scaling on the velocity is required. Otherwise, the joint maximum velocity divided by the joint velocity producing the smallest percentage is used as the scaling amount. All the joints velocities are scaled back the percentage to stay within the velocity performance tolerance.

New joint angles are then computed based on the acceleration and velocity scaling. Should no scaling have been performed on either the accelerations or the velocity, the new joint angles are the same as the computed joint angles. Otherwise, the new joint angles are computed as the sum of the current joint angle plus the scaled velocity amount. With these new joint angles, the forward solution is performed to produce a new scaled position and orientation for the robot end-effector.

A problem that can occur within the context of scaling is a change of direction within any joint. Should this occur, the concept of scaling is no longer is applicable. This is due to the fact that it can no longer be easily determined how much scaling is appropriate. Letting the hardware servos scale the motion seems to be the simplest yet most effective action. Given this change of direction condition, the current implementation merely checks joint boundary limits and ignores scaling. Thus, the coordinated joint module returns the same position and orientation for the robot end-effector for a motion that has undergone no scaling.

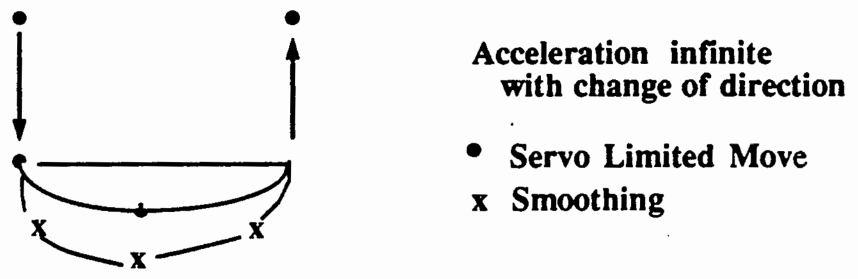


Figure 5.1 Change of Direction Anomaly

Coordinated Joint Algorithm

```
" Test current joint against upper/lower joint limits"
 for i = 1,#joints
    do if jointt { i } > uj-joint-limit  OR
          jointt { i } < lj-joint-limit
```

```
            then " halt move "
        endif
enddo

change-of-direction <- 0
" Compute velocity and acceleration and any change of direction"
for i = 1,#joints
   do velocityt { i } = next-jointt { i } - jointt-1 { i }
        accelerationt { i } = veloctiyt { i } - last-velocityt-1 { i }

        if ( velocityt { i } < 0 AND last-velocityt-1 { i } > 0 ) OR
           ( velocityt { i } > 0 AND last-velocityt-1 { i } < 0 )
            then change-of-direction <- 1
   endif

" Determine if acceleration scaling is necessary"
1.0 -> percent-acc
for i = 1,#joints
   do max-acceleration { i } / acceleration { i } -> scaled-acc
        if scaled-acc < percent-acc
            then scaled-acc - > percent-acc
        endif
    endif

" Determine if further velocity scaling is necessary"
1.0 -> percent-vel
for i = 1,#joints
   do max-velocity { i } / velocity { i } -> scaled-acc
        if scaled-vel < percent-vel
            then scaled-vel - > percent-vel
        endif
enddo

" Compute new joint angles"
if percent-acc + percent-vel < 2.0
 for i = 1,#joints
    do velocityt { i } * percent-vel -> velocityt { i }
         jointt-1 { i } + velocityt { i } + -> next-jointt { i }
enddo
```

6.0 PROBLEM AREAS

So far, problem areas within the T3 kinematics have been overlooked. These problems include degeneracy and singularity of the T3 arm kinematics and numerical errors due to the arithmetic hardware solving the kinematics.

In mathematics, degeneracy refers to the problem of multiple solutions to a problem. In regard to the kinematics, degeneracy occurs when several robot joint configurations achieve the same end-effector position and orientation. For the T3, the physical limitation of the arm prevents most of the multiple solutions associated with other robots, i.e. shoulder flip, waist flip, etc. The T3 has but two degenerate cases.

The first occurs when the yaw is at +/- 90°. This configuration allows an infinite number of configurations involving the shoulder, elbow and pitch angles to maintain the same point while moving the robot joints. To test for this condition, the xyz position point is crossed with the (Δx; Δy, Δz,) x-axis directional unit vector. If this dot product is zero, then the vectors are orthogonal, and hence a 90° yaw angle. When this condition occurs, the coordinated joint algorithm does not attempt a backward solution, but instead uses the values of the current joints as an approximation to the next joint angles.

The second degeneracy occurs when the shoulder flips back and places the end-effector into the opposite quadrant than would be expected. This configuration has a direct bearing on the waist or base angle. This condition occurs only when the shoulder flip has caused the x'y'z' or pitch point to cross the vertical 0° perpendicular through the base.

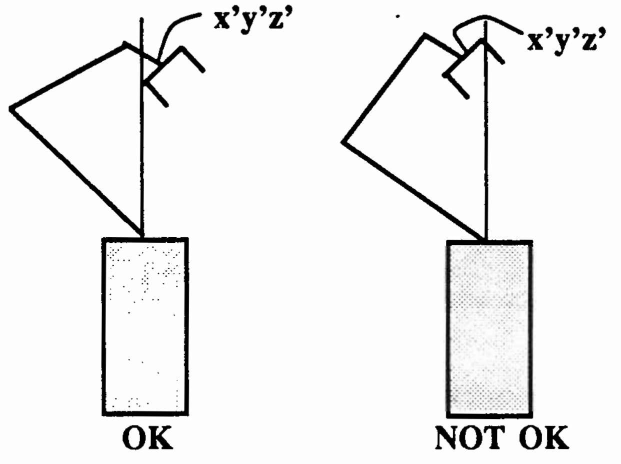


Figure Degenerate End-Effector Position

Normally, such a end-effector configuration would give a base angle derived via the arctangent of the xyz position to x'y'z' pitch position.

$$\theta 1 = \text{ARCTAN} (y'/x')$$

However, the shoulder flip causes the angle to be off by 180<. So far the use of a shoulder flip flag has been used to differentiate among possible configurations.

Singularity refers to the an undefined derivative or determinant of a matrix. With respect to the kinematics, singularity occurs when a large movement of the arm is necessary to change a small position, leading to infinite acceleration. This condition usually occurs when joints are lined up resulting in the loss of degrees of freedom. This allows large swings of motion to via joint angle sign flips. For the T3, the yaw at +/- 90° is not only a degenerate condition, but also a singularity. Thus, if the yaw was 89.99° before attempting a 90° yaw angle, the coordinated joint module will approximate the solution with the previous yaw joint angle, 89.99<.

Numerical computation error in the mathematics refers to errors that occur due to floating point round-off and ill-conditioning of the mathematics. Round-off error refers to errors that occur due to loss of precision due to the limitation of the number of significant digits with floating point hardware. Ill-conditioning refers to slight perturbations in the answers that cause large errors. Ill-conditioning usually occurs at boundary conditions of problems, such as a discontinuous point or when the determinant of a matrix approaches zero with the subsequent loss of a degree of freedom.

For the T3, round-off error combining with ill-conditioning of the problem solution occurs when solving for the roll angle and joints. The use of the Pythagorean Theorem to derive a joint angle using the formula

$$a^2 = \mathrm{SQRT}(\ \mathrm{hypotenuse}^2 - b^2)\ .$$

However, errors at boundary conditions that cause negative square roots, which in the case of the Intel 8087 coprocessor, ceases processing. To alleviate this problem, a zero threshold test was inserted into the algorithm to prevent negative square roots.

```
if $\mathrm{hypotenuse}^2 - b^2 - \mathrm{threshold} < 0$
  then $a=0$
  else $a^2 = \mathrm{SQRT}(\ \mathrm{hypotenuse}^2 - b^2)$
endif
```

7.0 CONCLUSION

The tools to build a coordinated joint motion level within a robot control system have been presented. A general overview of how robot poses are translated into joint angles with the use of coordinate frames is included to provide the necessary background to understand how robot motion is controlled. Within the coordinated joint level, a forward kinematic solution that provides intermediate Cartesian point and a geometric backward solution were described for the Cincinnati Milicron T3 industrial. Given the kinematics a coordinated joint algorithm to scale the velocities and acceleration of joint motion was also described. Finally, problems with the coordinated joint level approach as described in this paper were reviewed.

REFERENCES

[1] A.J. Barbera, M.L. Fitzgerald, J.S. Albus, and L. Haynes, "RCS: The NBS Real-Time Control System," *Proceedings of Robots VIII Conference*, Detroit, June 1984.

[2] A.J. Barbera, M.L. Fitzgerald, and J.S. Albus, "Concepts for a Real-Time Sensory-Interactive Control system Architecture Architecture", *Proceedings of the Fourtheenth Southeastern Symposium on System Theory*, April 1982, pp. 121-126.

[3] M. Brady, J. Hollerbach, W.T. Johnson, T. Lozano-Perez, and M. Mason, Robot

Motion: Planning and Control, MIT Press, Cambridge, Mass., 1982.

[4] J. Craig, Introduction to Robotics: Mechanics and Control, Addison-Wesley, Reading, MA, 1986.

[5] J. Denavit and R.S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms based on Matrices," *ASME J. Applied Mechanics*, June 1955, pp. 215-221.

[6] R. Featherstone, "Positions and Velocity Transformations Between Robot End-effector Coordinate and Joint Angles," *Internations Journal of Robotics Research*, Vol. 2 No.2, Summer 1983, pp. 35-45.

[7] M.L. Fitzgerald and A.J. Barbera, "A Low-Level Control Interface for Robot Manipulators," *NBS-Navy NAV/SIM Workshop on Robots Standards*, June 6-7, 1985.

[8] C.S.G. Lee and M. Ziegler, "A Geometric Approach in Solving the Inverse Kinematics of Puma Robots. *Conference Proceedings - 13th International Symposium on Industrial Robots and Robots 7*, Robotics Industrial of SME, Dearborn, Michigan, 1983.

[9] D. Myers and D. Gordon, "Kinematic Equations for Industrial Manipulators," *Industrial Robot*, September 1982.

[10] R.P. Paul, Robot Manipulators : Mathematics, Programming and Control, MIT Press, Cambridge, Mass., 1981.

[11] R.P. Paul, B. Shimano, and G.E. Mayer, "Kinematic Control Equations for Simple Manipulators," *IEEE Transactions on System, Man, Cybernetics*, Vol. SMC-11, No.6, June 1981, pp. 449-455.

[12] A. Sadre, R. Smith, and W. Cartwright, "Coordinate Transformations for Two Industrial Robots," *IEEE Conference on Automation and Robotics*, 1984. pp. 45-61.

[13] J.A. Simpson, R.J. Hocken, and J.S. Albus, "The Automated Manufacturing Research Facility of the National Bureau of Standards," *Journal of Manufacturing Systems*, Vol. 1(1):17-32, 1982.

-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NBSIR 88-3735	2. Performing Organ. Report No.	3. Publication Date MARCH 1988

4. TITLE AND SUBTITLE

Coordinated Joint Motion Control for an Industrial Robot

5. AUTHOR(S)

John Michaloski

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

National Bureau of Standards
Center for Manufacturing Engineering
Robotics Systems Division
Gaithersburg, MD 20899

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

The tools to build a coordinated joint motion controller for a robot have been presented. A general overview of how robot poses are translated into joint angles with the use of coordinate frames is included to provide the necessary background to understand how a robot achieves motion. Within the coordinated joint level, a forward kinematic solution that provides intermediate Cartesian point and a geometric backward solution were described for the Cincinatti Milicron T3 industrial robot. Given the kinematics a coordinated joint algorithm to scale the velocities and acceleration of joint motion was also described. Finally, problems with the coordinated joint level were reviewed.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

coordinate joint motion, kinematics, forward solution, backward solution, robot poses, coordinate frames

13. AVAILABILITY

[✓] Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
[✓] Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

31

15. Price

$11.95