Chapter 3

Direct geometric model of serial robots

3.1. Introduction

The design and control of a robot requires the computation of some mathematical models such as:

- transformation models between the joint space (in which the configuration of the robot is defined) and the task space (in which the location of the end-effector is specified). These transformation models are very important since robots are controlled in the joint space, whereas tasks are defined in the task space. Two classes of models are considered:
  - direct and inverse geometric models, which give the location of the end-effector as a function of the joint variables of the mechanism and vice versa;
  - direct and inverse kinematic models, which give the velocity of the end-effector as a function of the joint velocities and vice versa;
- dynamic models giving the relations between the input torques or forces of the actuators and the positions, velocities and accelerations of the joints.

The automatic symbolic computation of these models has largely been addressed in the literature [Dillon 73], [Khalil 76], [Zabala 78], [Kreuzer 79], [Aldon 82], [Cesareo 84], [Megahed 84], [Murray 84], [Kircánski 85], [Burdick 86], [Izaguirre 86], [Khalil 89a]. The algorithms presented in this book have been used in the development of the software package SYMORO+ [Khalil 97], which deals with all the above-mentioned models.

The modeling of robots in a systematic and automatic way requires an adequate method for the description of their structure. Several methods and notations have been proposed [Denavit 55], [Sheth 71], [Renaud 75], [Khalil 76], [Borrel 79], [Craig 86a]. The most popular among these is the Denavit-Hartenberg method [Denavit 55]. This method is developed for serial structures and presents ambiguities when applied to robots with closed or tree chains. For this reason, we will use the notation of Khalil and Kleinfinger [Khalil 86a], which gives a unified description for all mechanical articulated systems with a minimum number of parameters.

In this chapter, we will present the geometric description and the direct geometric model of serial robots. Tree and closed loop structures will be covered in Chapter 7.

3.2. Description of the geometry of serial robots

A serial robot is composed of a sequence of n + 1 links and n joints. The links are assumed to be perfectly rigid. The joints are either revolute or prismatic and are assumed to be ideal (no backlash, no elasticity). A complex joint can be represented by an equivalent combination of revolute and prismatic joints with zero-length massless links. The links are numbered such that link 0 constitutes the base of the robot and link n is the terminal link (Figure 3.1). Joint j connects link j to link j – 1 and its variable is denoted q_j. In order to define the relationship between the location of links, we assign a frame R_j attached to each link j, such that:

- the $\mathbf{z}_j$ axis is along the axis of joint j;
- the $\mathbf{x}_j$ axis is aligned with the common normal between $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$. If $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$ are parallel or collinear, the choice of $\mathbf{x}_j$ is not unique. The intersection of $\mathbf{x}_j$ and $\mathbf{z}_j$ defines the origin O_j. In the case of intersecting joint axes, the origin is at the point of intersection of the joint axes;
- the $\mathbf{y}_j$ axis is formed by the right-hand rule to complete the coordinate system $(\mathbf{x}_j, \mathbf{y}_j, \mathbf{z}_j)$.

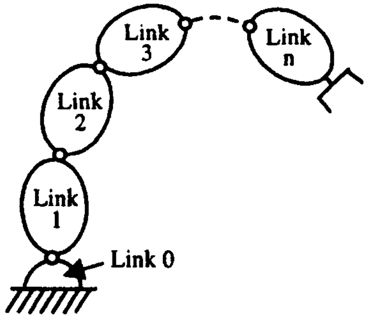

Figure 3.1. *Robot with simple open structure*

The transformation matrix from frame R_{j-1} to frame R_j is expressed as a function of the following four geometric parameters (Figure 3.2):

- α_j: the angle between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ about $\mathbf{x}_{j-1}$;
- d_j: the distance between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ along $\mathbf{x}_{j-1}$;
- θ_j: the angle between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ about $\mathbf{z}_j$;
- r_j: the distance between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ along $\mathbf{z}_j$.

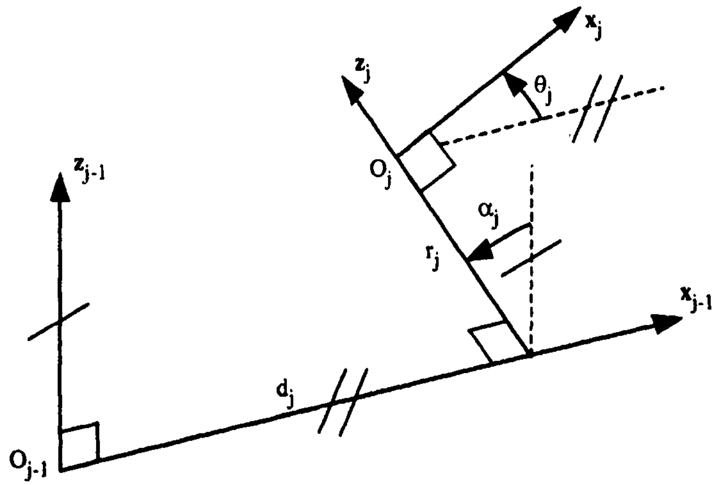

Figure 3.2. *The geometric parameters in the case of a simple open structure*

The variable of joint j, defining the relative orientation or position between links j – 1 and j, is either θ_j or r_j, depending on whether the joint is revolute or prismatic respectively. This is defined by the relation:

$$q_j = \bar{\sigma}_j \theta_j + \sigma_j r_j \qquad [3.1a]$$

with:

- $\sigma_j = 0$ if joint j is revolute;
- $\sigma_j = 1$ if joint j is prismatic;
- $\bar{\sigma}_j = 1 - \sigma_j$.

By analogy, we define the parameter $\bar{q}_j$ by:

$$\bar{q}_j = \sigma_j \theta_j + \bar{\sigma}_j r_j \qquad [3.1b]$$

The transformation matrix defining frame R_j relative to frame R_{j-1} is given as (Figure 3.2):

$$
{}^{j-1}\mathbf{T}_j = \mathbf{Rot}(\mathbf{x}, \alpha_j)\, \mathbf{Trans}(\mathbf{x}, d_j)\, \mathbf{Rot}(\mathbf{z}, \theta_j)\, \mathbf{Trans}(\mathbf{z}, r_j)
$$

$$
= \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [3.2]
$$

We note that the (3x3) rotation matrix ${}^{j-1}\mathbf{A}_j$ can be obtained as:

$$
{}^{j-1}\mathbf{A}_j = \mathbf{rot}(\mathbf{x}, \alpha_j)\, \mathbf{rot}(\mathbf{z}, \theta_j) \qquad [3.3]
$$

The transformation matrix defining frame R_{j-1} relative to frame R_j is given as:

$$
{}^{j}\mathbf{T}_{j-1} = \mathbf{Trans}(\mathbf{z}, -r_j)\, \mathbf{Rot}(\mathbf{z}, -\theta_j)\, \mathbf{Trans}(\mathbf{x}, -d_j)\, \mathbf{Rot}(\mathbf{x}, -\alpha_j)
$$

$$
= \begin{bmatrix} & -d_j C\theta_j \\ {}^{j-1}\mathbf{A}_j^T & d_j S\theta_j \\ & -r_j \\ 0\ 0\ 0 & 1 \end{bmatrix} \qquad [3.4]
$$

NOTES.–

- the frame R_0 is chosen to be aligned with frame R_1 when $q_1 = 0$. This means that $\mathbf{z}_0$ is aligned with $\mathbf{z}_1$, whereas the origin O_0 is coincident with the origin O_1 if joint 1 is revolute, and $\mathbf{x}_0$ is parallel to $\mathbf{x}_1$ if joint 1 is prismatic. This choice makes $\alpha_1 = 0$, $d_1 = 0$ and $\bar{q}_1 = 0$;
- in a similar way, the choice of the $\mathbf{x}_n$ axis to be aligned with $\mathbf{x}_{n-1}$ when $q_n = 0$ makes $\bar{q}_n = 0$;
- if joint j is prismatic, the $\mathbf{z}_j$ axis must be taken to be parallel to the joint axis but can have any position in space. So, we place it in such a way that $d_j = 0$ or $d_{j+1} = 0$;
- if $\mathbf{z}_j$ is parallel to $\mathbf{z}_{j+1}$, we place $\mathbf{x}_j$ in such a way that $r_j = 0$ or $r_{j+1} = 0$;
- assuming that each joint is driven by an independent actuator, the vector of joint variables $\mathbf{q}$ can be obtained from the vector of encoder readings $\mathbf{q}_c$ using the relation:

$$\mathbf{q} = \mathbf{K}\,\mathbf{q}_c + \mathbf{q}_0$$

where **K** is an (n×n) constant matrix and $\mathbf{q}_0$ is an offset vector representing the robot configuration when $\mathbf{q}_c = \mathbf{0}$;

- if a chain contains two or more consecutive parallel joints, the transformation matrices between them can be reduced to one equivalent transformation matrix using the sum of the joint variables. For example, if $\alpha_{j+1} = 0$, i.e. if $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$ are parallel, the transformation ${}^{j-1}\mathbf{T}_{j+1}$ is written as:

$$\begin{aligned}{}^{j-1}\mathbf{T}_{j+1} = {}^{j-1}\mathbf{T}_j\,{}^{j}\mathbf{T}_{j+1} &= \mathbf{Rot}(\mathbf{x}, \alpha_j)\,\mathbf{Trans}(\mathbf{x}, d_j)\,\mathbf{Rot}(\mathbf{z}, \theta_j)\,\mathbf{Trans}(\mathbf{z}, r_j) \\ &\quad \mathbf{Trans}(\mathbf{x}, d_{j+1})\,\mathbf{Rot}(\mathbf{z}, \theta_{j+1})\,\mathbf{Trans}(\mathbf{z}, r_{j+1})\end{aligned} \qquad [3.5]$$

$$= \begin{bmatrix} C(\theta_j+\theta_{j+1}) & -S(\theta_j+\theta_{j+1}) & 0 & d_j+d_{j+1}C\theta_j \\ C\alpha_j S(\theta_j+\theta_{j+1}) & C\alpha_j C(\theta_j+\theta_{j+1}) & -S\alpha_j & d_{j+1}C\alpha_j S\theta_j-(r_j+r_{j+1})S\alpha_j \\ S\alpha_j S(\theta_j+\theta_{j+1}) & S\alpha_j C(\theta_j+\theta_{j+1}) & C\alpha_j & d_{j+1}S\alpha_j S\theta_j+(r_j+r_{j+1})C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the inverse transformation has the expression:

$${}^{j+1}\mathbf{T}_{j-1} = \begin{bmatrix} & {}^{j-1}\mathbf{A}_{j+1}^{T} & & \begin{matrix} -d_jC(\theta_j+\theta_{j+1})-d_{j+1}C\theta_{j+1} \\ d_jS(\theta_j+\theta_{j+1})+d_{j+1}S\theta_{j+1} \\ -(r_j+r_{j+1}) \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [3.6]$$

The above expressions contain terms in $(\theta_j + \theta_{j+1})$ and $(r_j + r_{j+1})$. This result can be generalized for the case of multiple consecutive parallel axes [Kleinfinger 86a].

• **Example 3.1.** Geometric description of the Stäubli RX-90 robot (Figure 3.3a). The shoulder is of RRR type and the wrist has three revolute joints whose axes intersect at a point (Figure 3.3b). From a methodological point of view, we first place the $\mathbf{z}_j$ axes on the joint axes, then the $\mathbf{x}_j$ axes according to the previously mentioned conventions. Then, we determine the geometric parameters defining each frame R_j with respect to frame R_{j-1}. The link coordinate frames are indicated in Figure 3.3b and the geometric parameters are given in Table 3.1.

Table 3.1. *Geometric parameters of the Stäubli RX-90 robot*

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	θ_1	0
2	0	$\pi/2$	0	θ_2	0
3	0	0	D3	θ_3	0
4	0	$-\pi/2$	0	θ_4	RL4
5	0	$\pi/2$	0	θ_5	0
6	0	$-\pi/2$	0	θ_6	0

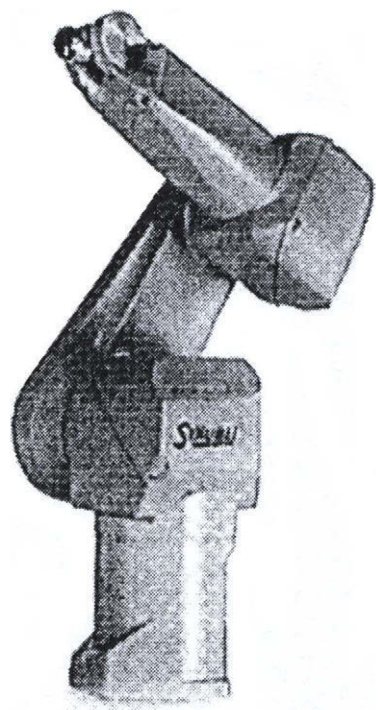

Figure 3.3a. *General view of the Stäubli RX-90 robot (Courtesy of Stäubli company)*

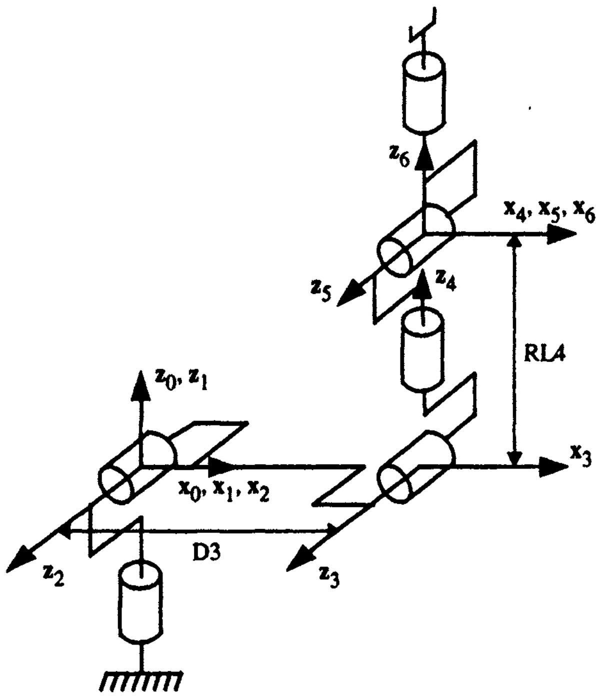

Figure 3.3b. *Link coordinate frames for the Stäubli RX-90 robot*

• **Example 3.2.** Geometric description of a SCARA robot (Figure 3.4). The geometric parameters of a four degree-of-freedom SCARA robot are given in Table 3.2.

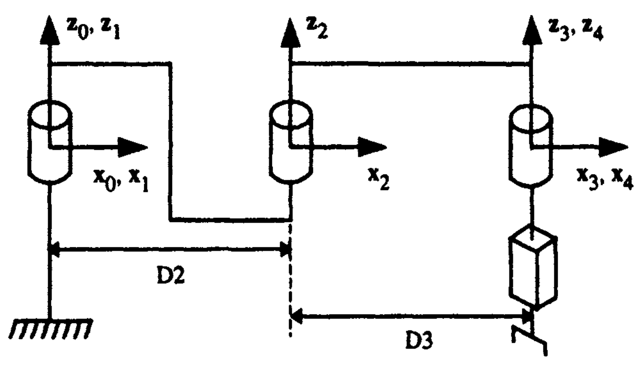

Figure 3.4. *SCARA Robot*

Table 3.2. *Geometric parameters of a SCARA robot*

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	θ_1	0
2	0	0	D2	θ_2	0
3	0	0	D3	θ_3	0
4	1	0	0	0	r_4

3.3. Direct geometric model

The Direct Geometric Model (DGM) is the set of relations that defines the location of the end-effector of the robot as a function of its joint coordinates. For a serial structure, it may be represented by the transformation matrix ${}^0\mathbf{T}_n$ as:

$$ {}^0\mathbf{T}_n = {}^0\mathbf{T}_1(q_1)\ {}^1\mathbf{T}_2(q_2)\ \ldots\ {}^{n-1}\mathbf{T}_n(q_n) \qquad [3.7]$$

This relation can be numerically computed using the general transformation matrix ${}^{j-1}\mathbf{T}_j$ given by equation [3.2], or symbolically derived after substituting the values of the constant geometric parameters in the transformation matrices (Example 3.3). The symbolic method needs less computational operations.

The direct geometric model of a robot may also be represented by the relation:

$$\mathbf{X} = \mathbf{f}(\mathbf{q}) \qquad [3.8]$$

where $\mathbf{q}$ is the vector of joint variables such that:

$$\mathbf{q} = [q_1 \quad q_2 \cdots q_n]^T \qquad [3.9]$$

The position and orientation of the terminal link are defined as:

$$\mathbf{X} = [x_1 \quad x_2 \ldots x_m]^T \qquad [3.10]$$

There are several possibilities of defining the vector $\mathbf{X}$ as we will see in § 3.6. For example, with the elements of the matrix ${}^0\mathbf{T}_n$:

$$\mathbf{X} = [P_x \quad P_y \quad P_z \quad s_x \quad s_y \quad s_z \quad n_x \quad n_y \quad n_z \quad a_x \quad a_y \quad a_z]^T \qquad [3.11]$$

Taking into account that $\mathbf{s} = \mathbf{n} \times \mathbf{a}$, we can also take:

$$X = [P_x \quad P_y \quad P_z \quad n_x \quad n_y \quad n_z \quad a_x \quad a_y \quad a_z]^T \qquad [3.12]$$

• **Example 3.3.** Symbolic direct geometric model of the Stäubli RX-90 robot (Figure 3.3). From Table 3.1 and using equation [3.2], we write the elementary transformation matrices ${}^{j-1}\mathbf{T}_j$ as:

$$^0\mathbf{T}_1 = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; ^1\mathbf{T}_2 = \begin{bmatrix} C2 & -S2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; ^2\mathbf{T}_3 = \begin{bmatrix} C3 & -S3 & 0 & D3 \\ S3 & C3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since the joint axes 2 and 3 are parallel, we can write the transformation matrix ${}^1\mathbf{T}_3$ using equation [3.5] as:

$$^1\mathbf{T}_3 = \begin{bmatrix} C23 & -S23 & 0 & C2D3 \\ 0 & 0 & -1 & 0 \\ S23 & C23 & 0 & S2D3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with $C23 = \cos(\theta_2 + \theta_3)$ and $S23 = \sin(\theta_2 + \theta_3)$.

$$^3\mathbf{T}_4 = \begin{bmatrix} C4 & -S4 & 0 & 0 \\ 0 & 0 & 1 & RL4 \\ -S4 & -C4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; ^4\mathbf{T}_5 = \begin{bmatrix} C5 & -S5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; ^5\mathbf{T}_6 = \begin{bmatrix} C6 & -S6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S6 & -C6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In order to compute ${}^0\mathbf{T}_6$, it is better to multiply the matrices ${}^{j-1}\mathbf{T}_j$ starting from the last transformation matrix and working back to the base, mainly for two reasons:

- the intermediate matrices ${}^j\mathbf{T}_6$, denoted as $\mathbf{U}_j$, will be used to obtain the inverse geometric model (Chapter 4);
- this reduces the number of operations (additions and multiplications) of the model.

We thus compute successively $\mathbf{U}_j$ for j = 5, ..., 0:

$$\mathbf{U}_5 = {}^5\mathbf{T}_6$$

$$U_4 = {}^4T_6 = {}^4T_5\, U_5 = \begin{bmatrix} C5C6 & -C5S6 & -S5 & 0 \\ S6 & C6 & 0 & 0 \\ S5C6 & -S5S6 & C5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U_3 = {}^3T_6 = {}^3T_4\, U_4 = \begin{bmatrix} C4C5C6-S4S6 & -C4C5S6-S4C6 & -C4S5 & 0 \\ S5C6 & -S5S6 & C5 & RL4 \\ -S4C5C6-C4S6 & S4C5S6-C4C6 & S4S5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U_2 = {}^2T_6 = {}^2T_3\, U_3$$

The **s**, **n**, **a**, **P** vectors of $\mathbf{U_2}$ are:

$s_x = C3(C4C5C6 - S4S6) - S3S5C6$
$s_y = S3(C4C5C6 - S4S6) + C3S5C6$
$s_z = - S4C5C6 - C4S6$
$n_x = - C3(C4C5S6 + S4C6) + S3S5S6$
$n_y = - S3(C4C5S6 + S4C6) - C3S5S6$
$n_z = S4C5S6 - C4C6$
$a_x = - C3C4S5 - S3C5$
$a_y = - S3C4S5 + C3C5$
$a_z = S4S5$
$P_x = - S3RL4 + D3$
$P_y = C3RL4$
$P_z = 0$

$$U_1 = {}^1T_6 = {}^1T_2\, U_2 = {}^1T_3\, U_3$$

The corresponding **s**, **n**, **a**, **P** vectors are:

$s_x = C23(C4C5C6 - S4S6) - S23S5C6$
$s_y = S4C5C6 + C4S6$
$s_z = S23(C4C5C6 - S4S6) + C23S5C6$
$n_x = - C23(C4C5S6 + S4C6) + S23S5S6$
$n_y = - S4C5S6 + C4C6$
$n_z = - S23(C4C5S6 + S4C6) - C23S5S6$
$a_x = - C23C4S5 - S23C5$
$a_y = - S4S5$
$a_z = - S23C4S5 + C23C5$
$P_x = - S23\, RL4 + C2D3$

$P_y = 0$
$P_z = C23\ RL4 + S2D3$

Finally:

$$U_0 = {}^0T_6 = {}^0T_1\ U_1$$

The corresponding **s**, **n**, **a**, **P** vectors are:

$s_x = C1(C23(C4C5C6 - S4S6) - S23S5C6) - S1(S4C5C6 + C4S6)$
$s_y = S1(C23(C4C5C6 - S4S6) - S23S5C6) + C1(S4C5C6 + C4S6)$
$s_z = S23(C4C5C6 - S4S6) + C23S5C6$
$n_x = C1(- C23\ (C4C5S6 + S4C6) + S23S5S6) + S1(S4C5S6 - C4C6)$
$n_y = S1(- C23\ (C4C5S6 + S4C6) + S23S5S6) - C1(S4C5S6 - C4C6)$
$n_z = - S23(C4C5S6 + S4C6) - C23S5S6$
$a_x = - C1(C23C4S5 + S23C5) + S1S4S5$
$a_y = - S1(C23C4S5 + S23C5) - C1S4S5$
$a_z = - S23C4S5 + C23C5$
$P_x = - C1(S23\ RL4 - C2D3)$
$P_y = - S1(S23\ RL4 - C2D3)$
$P_z = C23\ RL4 + S2D3$

3.4. Optimization of the computation of the direct geometric model

The control of a robot manipulator requires fast computation of its different models. An efficient method to reduce the computation time is to generate a symbolic customized model for each specific robot. To obtain this model, we expand the matrix multiplications to transform them into scalar equations. Each element of a matrix containing at least one mathematical operation is replaced by an intermediate variable. This variable is written in the output file that contains the customized model. The elements that do not contain any operation are kept without modification. We propagate the matrix obtained in the subsequent equations. Consequently, customizing eliminates multiplications by one and zero, and additions with zero. Moreover, if the robot has two or more successive revolute joints with parallel axes, it is more interesting to replace the corresponding product of matrices by a single matrix, which is calculated using equation [3.5]. We can also compute ${}^0\mathbf{s}_n$ using the vector product (${}^0\mathbf{n}_n \times {}^0\mathbf{a}_n$). In this case, the multiplication of the transformation matrices from the end-effector to the base saves the computation of the vectors ${}^j\mathbf{s}_n$ of the intermediate matrices ${}^j\mathbf{T}_n$, ($j = n, \ldots, 1$).

• **Example 3.4.** Direct geometric model of the Stäubli RX-90 robot using the customized symbolic method.

a) computation of all the elements (s, n, a, P)

We denote Tijrs as the element (r, s) of the matrix ${}^{i}T_{j}$. As in Example 3.3, the product of the matrices is carried out starting from the last transformation matrix. We obtain the following intermediate variables for the matrix ${}^{4}T_{6}$:

T4611 = C5 C6
T4612 = – C5 S6
T4631 = S5 C6
T4632 = – S5 S6

Proceeding in the same way, the other intermediate variables are written as:

T3611 = C4 T4611 – S4 S6
T3612 = C4 T4612 – S4 C6
T3613 = – C4 S5
T3631 = – S4 T4611 – C4 S6
T3632 = – S4 T4612 – C4 C6
T3633 = S4 S5
T1314 = D3 C2
T1334 = D3 S2
T1611 = C23 T3611 – S23 T4631
T1612 = C23 T3612 – S23 T4632
T1613 = C23 T3613 – S23 C5
T1614 = – S23 RL4 + T1314
T1631 = S23 T3611 + C23 T4631
T1632 = S23 T3612 + C23 T4632
T1633 = S23 T3613 + C23 C5
T1634 = C23 RL4 + T1334
T0611 = C1 T1611 + S1 T3631
T0612 = C1 T1612 + S1 T3632
T0613 = C1 T1613 + S1 T3633
T0614 = C1 T1614
T0621 = S1 T1611 – C1 T3631
T0622 = S1 T1612 – C1 T3632
T0623 = S1 T1613 – C1 T3633
T0624 = S1 T1614
T0631 = T1631
T0632 = T1632
T0633 = T1633
T0634 = T1634

Total number of operations: 44 multiplications and 18 additions *b) computing only the columns (**n**, **a**, **P**)*

T4612 = – C5 S6
T4632 = – S5 S6
T3612 = C4 T4612 – S4 C6
T3613 = – C4 S5
T3632 = – S4 T4612 – C4 C6
T3633 = S4 S5
T1314 = D3 C2
T1334 = D3 S2
T1612 = C23 T3612 – S23 T4632
T1613 = C23 T3613 – S23 C5
T1614 = – S23 RL4 + T1314
T1632 = S23 T3612 + C23 T4632
T1633 = S23 T3613 + C23 C5
T1634 = C23 RL4 + T1334
T0612 = C1 T1612 + S1 T3632
T0613 = C1 T1613 + S1 T3633
T0614 = C1 T1614
T0622 = S1 T1612 – C1 T3632
T0623 = S1 T1613 – C1 T3633
T0624 = S1 T1614
T0632 = T1632
T0633 = T1633
T0634 = T1634

Total number of operations: 30 multiplications and 12 additions

These equations constitute a complete direct geometric model. However, the computation of ${}^0\mathbf{s}_6$ requires six multiplications and three additions corresponding to the vector product (${}^0\mathbf{n}_6 \times {}^0\mathbf{a}_6$).

3.5. Transformation matrix of the end-effector in the world frame

The robot is a component among others in a robotic workcell. It is generally associated with fastening devices, sensors..., and eventually with other robots. Consequently, we have to define a reference world frame R_f, which may be different than the base reference frame R_0 of the robot (Figure 3.5). The transformation matrix defining R_0 with reference to R_f will be denoted as $\mathbf{Z} = {}^f\mathbf{T}_0$.

Moreover, very often, a robot is not intended to perform a single operation at the workcell: it has interchangeable different tools. In order to facilitate the programming of the task, it is more practical to define one or more functional frames, called *tool frames* for each tool. We denote $\mathbf{E} = {}^n\mathbf{T}_E$ as the transformation matrix defining the tool frame with respect to the terminal link frame.

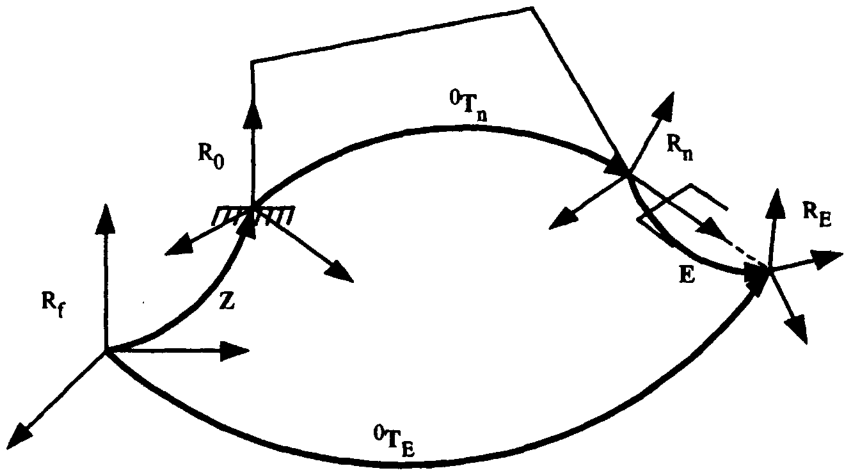

Figure 3.5. *Transformations between the end-effector and the world frame*

Thus, the transformation matrix ^fT_E can be written as:

$$^fT_E = \mathbf{Z}\, {}^0\mathbf{T}_n(\mathbf{q})\, \mathbf{E} \qquad [3.13]$$

In most programming languages, the user can specify **Z** and **E**.

3.6. Specification of the orientation

Previously, we have used the elements of the matrix $^0\mathbf{T}_n$ to represent the position and orientation of the end-effector in frame R_0. This means the use of the Cartesian coordinates to describe the position:

$$^0\mathbf{P}_n = [\ P_x\ \ P_y\ \ P_z\ ]^T \qquad [3.14]$$

and the use of the direction cosine matrix for the orientation:

$$^0\mathbf{A}_n = [\ {}^0\mathbf{s}_n\ \ {}^0\mathbf{n}_n\ \ {}^0\mathbf{a}_n\ ] \qquad [3.15]$$

Practically, all the robot manufacturers make use of the Cartesian coordinates for the position even though the cylindrical or spherical representations could appear to be more judicious for some structures of robots.

Other representations may be used for the orientation, for example: Euler angles for CINCINNATI-T3 robots and PUMA robots, Roll-Pitch-Yaw (RPY) angles for

ACMA robots, Euler parameters for ABB robots. In this section, we will show how to obtain the direction cosines **s**, **n**, **a** from the other representations and vice versa. Note that the orientation requires three independent parameters, thus the representation is redundant when it uses more than that.

3.6.1. *Euler angles*

The orientation of frame R_n expressed in frame R_0 is determined by specifying three angles, ϕ, θ and ψ, corresponding to a sequence of three rotations (Figure 3.6). The plane ($\mathbf{x}_n$, $\mathbf{y}_n$) intersects the plane ($\mathbf{x}_0$, $\mathbf{y}_0$) following the straight line ON, which is perpendicular to $\mathbf{z}_0$ and $\mathbf{z}_n$. The positive direction is given by the vector product $\mathbf{a}_0 \times \mathbf{a}_n$. The Euler angles are defined as:

- ϕ: angle between $\mathbf{x}_0$ and ON about $\mathbf{z}_0$, with $0 \le \phi < 2\pi$;
- θ : angle between $\mathbf{z}_0$ and $\mathbf{z}_n$ about ON, with $0 \le \theta \le \pi$;
- ψ: angle between ON and $\mathbf{x}_n$ about $\mathbf{z}_0$, with $0 \le \psi < 2\pi$.

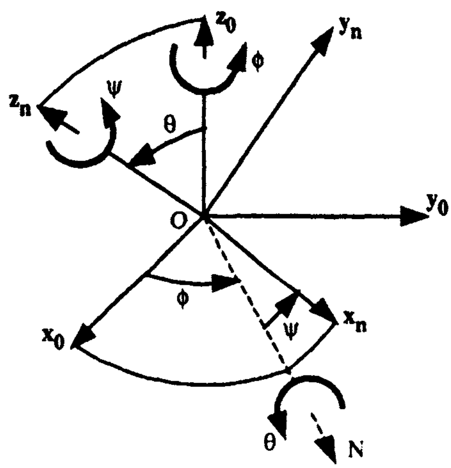

Figure 3.6. *Euler angles (z, x, z representation)*

The orientation matrix is given by:

$$
{}^0\mathbf{A}_n = \mathbf{rot}(\mathbf{z}, \phi)\, \mathbf{rot}(\mathbf{x}, \theta)\, \mathbf{rot}(\mathbf{z}, \psi)
= \begin{bmatrix} C\phi C\psi - S\phi C\theta S\psi & -C\phi S\psi - S\phi C\theta C\psi & S\phi S\theta \\ S\phi C\psi + C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & -C\phi S\theta \\ S\theta S\psi & S\theta C\psi & C\theta \end{bmatrix} \quad [3.16]
$$

Inverse problem: expression of the Euler angles as functions of the direction cosines. Premultiplying equation [3.16] by **rot(z, –ϕ)**, we obtain [Paul 81]:

$$\mathbf{rot}(\mathbf{z}, -\phi)\ {}^0\mathbf{A}_n = \mathbf{rot}(\mathbf{x}, \theta)\ \mathbf{rot}(\mathbf{z}, \psi) \qquad [3.17]$$

Using relations [3.15] and [3.17] yields:

$$\begin{bmatrix} C\phi s_x+S\phi s_y & C\phi n_x+S\phi n_y & C\phi a_x+S\phi a_y \\ -S\phi s_x+C\phi s_y & -S\phi n_x+C\phi n_y & -S\phi a_x+C\phi a_y \\ s_z & n_z & a_z \end{bmatrix} = \begin{bmatrix} C\psi & -S\psi & 0 \\ C\theta S\psi & C\theta C\psi & -S\theta \\ S\theta S\psi & S\theta C\psi & C\theta \end{bmatrix} \qquad [3.18]$$

Equating the (1, 3) elements of both sides, we obtain:

$$C\phi\ a_x + S\phi\ a_y = 0$$

which gives:

$$\begin{cases} \phi = \operatorname{atan2}(-a_x, a_y) \\ \phi' = \operatorname{atan2}(a_x, -a_y) = \phi + \pi \end{cases} \qquad [3.19]$$

NOTE.– atan2 is a mathematical function (Matlab, Fortran, ...), which provides the arc tangent function from its two arguments. This function has the following characteristics:

- examining the sign of both a_x and a_y allows us to uniquely determine the angle ϕ such that $-\pi \le \phi < \pi$;
- the accuracy of this function is uniform over its full range of definition;
- when $a_x = 0$, $a_y = 0$, $a_z = \pm 1$ the angle ϕ is undefined (singularity).

Using the (2, 3) and (3, 3) elements of equation [3.18], we obtain:

$$\theta = \operatorname{atan2}(S\phi\ a_x - C\phi\ a_y, a_z) \qquad [3.20]$$

We proceed in a similar way to calculate ψ using the (1, 1) and (1, 2) elements:

$$\psi = \operatorname{atan2}(-C\phi\ n_x - S\phi\ n_y, C\phi\ s_x + S\phi\ s_y) \qquad [3.21]$$

When a_x and a_y are zero, the axes $\mathbf{z}_n$ and $\mathbf{z}_0$ are aligned, thus θ is zero or π. This situation corresponds to the singular case: the rotations ϕ and ψ are about the same axis and we can only determine their sum or difference. For example, when $a_z = 1$, we obtain:

$$^0\mathbf{A}_n = \mathbf{rot}(\mathbf{z}, \psi + \phi)$$

and from this, we deduce:

$$\psi + \phi = \operatorname{atan2}(-n_x, n_y) \qquad [3.22]$$

NOTE.– The Euler angles adopted here correspond to a (**z**, **x**, **z**) representation where the first rotation is about $\mathbf{z}_0$, followed by a rotation about the new **x** axis, followed by a last rotation about the new **z** axis. Some authors prefer the (**z**, **y**, **z**) representation [Paul 81]. A specific but interesting case can be encountered in the PUMA robot controller [Lee 83], [Dombre 88a] where an initial shift is introduced so that the orientation matrix is written as:

$$^0\mathbf{A}_n = \mathbf{rot}(\mathbf{z}, \phi)\, \mathbf{rot}(\mathbf{x}, \theta + \frac{\pi}{2})\, \mathbf{rot}(\mathbf{z}, \psi - \frac{\pi}{2}) \qquad [3.23]$$

3.6.2. *Roll-Pitch-Yaw angles*

Following the convention shown in Figure 3.7, the angles ϕ , θ and ψ indicate roll, pitch and yaw respectively. If we suppose that the direction of motion (by analogy to the direction along which a ship is sailing) is along the **z** axis, the orientation matrix can be written as:

$$^0\mathbf{A}_n = \mathbf{rot}(\mathbf{z}, \phi)\, \mathbf{rot}(\mathbf{y}, \theta)\, \mathbf{rot}(\mathbf{x}, \psi)$$

$$= \begin{bmatrix} C\phi C\theta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi \\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix} \qquad [3.24]$$

Inverse problem: expression of the Roll-Pitch-Yaw angles as functions of the direction cosines. We use the same method discussed in the previous section. Premultiplying equation [3.24] by $\mathbf{rot}(\mathbf{z}, -\phi)$, we obtain:

$$\mathbf{rot}(\mathbf{z}, -\phi)\, ^0\mathbf{A}_n = \mathbf{rot}(\mathbf{y}, \theta)\, \mathbf{rot}(\mathbf{x}, \psi) \qquad [3.25]$$

which results in:

$$\begin{bmatrix} C\phi s_x+S\phi s_y & C\phi n_x+S\phi n_y & C\phi a_x+S\phi a_y \\ -S\phi s_x+C\phi s_y & -S\phi n_x+C\phi n_y & -S\phi a_x+C\phi a_y \\ s_z & n_z & a_z \end{bmatrix} = \begin{bmatrix} C\theta & S\theta S\psi & S\theta C\psi \\ 0 & C\psi & -S\psi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix} \quad [3.26]$$

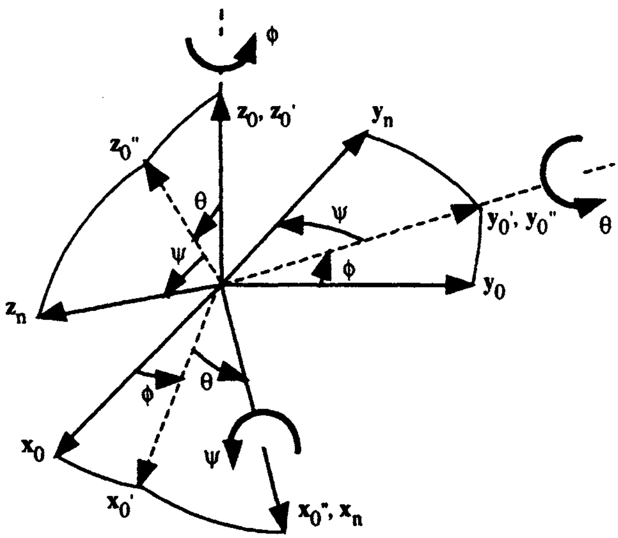

Figure 3.7. *Roll-Pitch-Yaw angles*

From the (2, 1) elements of equation [3.26], we obtain:

$$-S\phi\, s_x + C\phi\, s_y = 0$$

thus:

$$\begin{cases} \phi = \mathrm{atan2}(s_y, s_x) \\ \phi' = \mathrm{atan2}(-s_y, -s_x) = \phi + \pi \end{cases} \quad [3.27]$$

There is a singularity if s_x and s_y are zero ($\theta = \pm\frac{\pi}{2}$).

In the same way, from the (1, 1) and (1, 3) elements, then from the (2, 2) and (2, 3) elements, we deduce that:

$$\theta = \mathrm{atan2}(-s_z, C\phi\, s_x + S\phi\, s_y) \quad [3.28]$$

$$\psi = \mathrm{atan2}(S\phi a_x - C\phi\, a_y, -S\phi\, n_x + C\phi\, n_y) \quad [3.29]$$

3.6.3. *Quaternions*

The quaternions are also called *Euler parameters* or *Olinde-Rodrigues parameters*. In this representation, the orientation is expressed by four parameters that describe the orientation by a rotation of an angle θ ($0 \le \theta \le \pi$) about an axis of unit vector **u** (Figure 3.8). We define the quaternions as:

$$\begin{cases} Q_1 = C(\theta/2) \\ Q_2 = u_x\, S(\theta/2) \\ Q_3 = u_y\, S(\theta/2) \\ Q_4 = u_z\, S(\theta/2) \end{cases} \qquad [3.30]$$

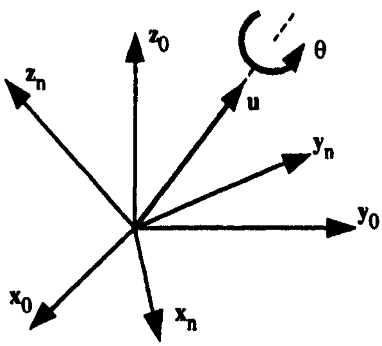

Figure 3.8. *The quaternions*

From these relations, we obtain:

$$Q_1^2 + Q_2^2 + Q_3^2 + Q_4^2 = 1 \qquad [3.31]$$

The orientation matrix ${}^0A_n$ is deduced from equation [2.30], defining **rot(u, θ)**, after rewriting its elements as a function of Q_i. We note that:

$$C\theta = C^2(\theta/2) - S^2(\theta/2) = 2Q_1^2 - 1 \qquad [3.32]$$

and that:

$$\begin{cases} Q_2^2 = u_x^2\, S^2(\theta/2) = \frac{1}{2} u_x^2\, (1 - C\theta) \\ Q_3^2 = \frac{1}{2} u_y^2\, (1 - C\theta) \\ Q_4^2 = \frac{1}{2} u_z^2\, (1 - C\theta) \\ Q_2\, Q_3 = \frac{1}{2} u_x\, u_y\, (1 - C\theta) \\ Q_2\, Q_4 = \frac{1}{2} u_x\, u_z\, (1 - C\theta) \\ Q_3\, Q_4 = \frac{1}{2} u_y u_z\, (1 - C\theta) \\ u_x\, S\theta = 2\, u_x\, S(\theta/2)\, C(\theta/2) = 2\, Q_1\, Q_2 \\ u_y\, S\theta = 2\, Q_1\, Q_3 \\ u_z\, S\theta = 2\, Q_1\, Q_4 \end{cases} \tag{3.33}$$

Thus, the orientation matrix is given as:

$$^0A_n = \begin{bmatrix} 2(Q_1^2+Q_2^2) - 1 & 2(Q_2Q_3-Q_1Q_4) & 2(Q_2Q_4+Q_1Q_3) \\ 2(Q_2Q_3+Q_1Q_4) & 2(Q_1^2+Q_3^2) - 1 & 2(Q_3Q_4-Q_1Q_2) \\ 2(Q_2Q_4-Q_1Q_3) & 2\,(Q_3Q_4+Q_1Q_2) & 2\,(Q_1^2+Q_4^2) - 1 \end{bmatrix} \tag{3.34}$$

For more information on the algebra of quaternions, the reader can refer to [de Casteljau 87].

Inverse problem: expression of the quaternions as functions of the direction cosines. Equating the elements of the diagonals of the right sides of equations [3.34] and [3.15] leads to:

$$Q_1 = \frac{1}{2}\sqrt{s_x + n_y + a_z + 1} \tag{3.35}$$

which is always positive. If we then subtract the (2, 2) and (3, 3) elements from the (1, 1) element, we can write after simplifying:

$$4\, Q_2^2 = s_x - n_y - a_z + 1 \tag{3.36}$$

This expression gives the magnitude of Q_2. For determining the sign, we consider the difference of the (3, 2) and (2, 3) elements, which leads to:

$$4\,Q_1Q_2 = n_z - a_y \qquad [3.37]$$

The parameter Q_1 being always positive, the sign of Q_2 is that of $(n_z - a_y)$, which allows us to write:

$$Q_2 = \frac{1}{2}\,\mathrm{sign}\,(n_z - a_y)\sqrt{s_x - n_y - a_z + 1} \qquad [3.38]$$

Similar reasoning for Q_3 and Q_4 gives:

$$Q_3 = \frac{1}{2}\,\mathrm{sign}\,(a_x - s_z)\sqrt{-s_x + n_y - a_z + 1} \qquad [3.39]$$

$$Q_4 = \frac{1}{2}\,\mathrm{sign}\,(s_y - n_x)\sqrt{-s_x - n_y + a_z + 1} \qquad [3.40]$$

These expressions exhibit no singularity.

3.7. Conclusion

In this chapter, we have shown how to calculate the direct geometric model of a serial robot. This model is unique and is given in the form of explicit equations. The description of the geometry is based on rules that have an intrinsic logic facilitating its application. This method can be generalized to tree and closed loop structures (Chapter 7). It can also be extended to systems with lumped elasticity [Khalil 00a].

We have also presented the methods that are frequently used in robotics to specify the orientation of a body in space. We have shown how to calculate the orientation matrix from these representations and inversely, how to find the parameters of these descriptions from the orientation matrix.

Having calculated the direct geometric model, in the next chapter we study the inverse geometric problem, which consists of computing the joint variables as functions of a given location of the end-effector.