# Real-Time Manipulation of a Hybrid Serial-and-Parallel-Driven Redundant Industrial Manipulator

**H. H. Cheng**
Assistant Professor,
Department of Mechanical and Aeronautical Engineering,
University of California,
Davis, CA 95616,
Assoc. Mem. ASME

*The real-time implementation of path planning, trajectory generation, and servo control for manipulation of the prototype UPSarm are presented in this paper. The prototype UPSarm, which is primarily designed for studying the feasibility of loading packages inside a trailer, is a ten degree-of-freedom hybrid serial-and-parallel-driven redundant robot manipulator. The direct, forward, inverse, and indirect kinematic solutions of the UPSarm using three coordinate spaces: actuator space, effective joint space, and world Cartesian coordinate space are derived for real-time path planning, trajectory generation, and control. The manipulation of the UPSarm is based upon a general-purpose path planner and trajectory generator. Provided with appropriate kinematics modules and sufficient computational power, this path planner and trajectory generator can be used for real-time motion control of any degree-of-freedom hybrid serial-and-parallel-driven electromechanical devices. A VMEbus-based distributed computing system has been implemented for real-time motion control of the UPSarm. A PID-based feedforward servo control scheme is used in our servo controller. The motion examples of the UPSarm programmed in our robot language will show the practical manipulation of hybrid serial-and-parallel-driven redundant kinematic chains.*

## 1 Introduction

Most industrial robot manipulators are designed with serial kinematic chains which can provide large workspaces. Much work on kinematics and real-time manipulation of robot manipulators with serial kinematic chains has been reported in the literature (Cheng and Gupta, 1991; 1992; 1993; Featherstone, 1983; Hollerbach and Sahar, 1983; Luh et al., 1980a; 1980b; Pieper, 1969; Tsai and Morgan, 1985; Wang and Chen, 1991; Whitcomb et al., 1991; Whitney, 1969). However, a robot manipulator with a serial kinematic chain generally provides less rigidity and load-carrying capacity in comparing with a robot with closed kinematic chains. The design and analysis of robot manipulators consisting of four-bar-linkage-type closed kinematic chains are also available (Bottema and Roth, 1979; Litvin et al., 1986). The fully parallel manipulators such as the Stewart-platform have been investigated by many researchers (Fichter, 1986; Sugimoto, 19887; Zhang and Song, 1991). Lee and Shah (1987) presented the forward and inverse kinematic solutions for a three-degree-of-freedom fully in-parallel drive mechanism. In general, the workspace volume of the robot arm consisting of only parallel kinematic chains is relatively small. Currently, there has been an increasing interest in the design of hybrid serial-and-parallel-driven robot manipulators which can provide good features of both serial and parallel kinematic chains. Hunt (1983) investigated possible combinations of series and parallel kinematic chains. Waldron et al. (1989) systematically performed kinematic analysis for a six degrees-of-freedom hybrid serial-and-parallel manipulator system with first three joints in serial drive and last three joints in fully parallel configuration. The manipulator they studied in detail is a six-degree-of-freedom hybrid serial-and-parallel-driven robot manipulator, and they indicated that the final design of the system will be a nine degree-of-freedom redundant manipulator.

The prototype UPSarm shown in Fig. 1 is a ten degree-of-

Contributed by the Dynamic Systems and Control Division for publication in the JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL. Manuscript received by the DSCD September 2, 1992; revised manuscript received August 1993. Associate Technical Editor: H. Kazerooni.


Fig. 1 The prototype UPSarm

freedom hybrid serial-and-parallel-driven redundant robot manipulator, it is designed recently for studying the feasibility of loading packages inside a flatbed trailer. In the final design, a self-propelled carriage will support one or two UPSarms inside the trailer to stack packages wall by wall from the front to the rear of the trailer. The UPSarm will operate in a constrained environment. It loads and stacks packages weighing up to 70 lbs and ranging in size from 0.5 in. high to 26 in. wide and 48 in. long. Due to the application-specific nature of this robot manipulator, the design of the UPSarm is unique in its kinematic configuration. The geometrical configuration of the manipulator is shown in Fig. 2. Little literature on the kinematics and real-time motion coordination of the kinematic chains of this kind is available. The closed kinematic chain of the shoulder subsystem provides the UPSarm with sufficient load lifting capabilities. The wrist subsystem of the UPSarm consists of two in-parallel actuators; it is quite different in design from the parallel mechanism, investigated by Hunt (1983), Lee and Shah (1987), and Waldron et al. (1989), which is a fully parallel three-degree-of-freedom mechanism. A numerical method is used to solve the wrist kinematics of the UPSarm. To facilitate the kinematic analysis of hybrid serial-and-parallel-driven redundant robot manipulators, three coordinate spaces: the world Cartesian coordinate space, effective joint space, and actuator space are introduced. The manipulator end-effector can be represented in one of the aforementioned three spaces. The 4×4 homogeneous transformation matrix defines the tool center point

## Nomenclature

$\beta$, $\beta_0$ = offset angles for joint 3 used to calculate $\phi_3$

$\boldsymbol{\phi}$, $\phi_i$; $\dot{\boldsymbol{\phi}}$, $\dot{\phi}_i$; $\ddot{\boldsymbol{\phi}}$, $\ddot{\phi}_i$ = effective joint values, velocities, and accelerations

$\theta_i$ = one of **D-H** parameters

$\tau_f$ = the generalized forces provided by the robot actuators

$a_i$ = one of **D-H** parameters

$(b_x, b_y, b_z)$ = the position vector of the robot base w.r.t. $X_w$ $Y_w$ $Z_w$ $O_w$

$\mathbf{C}$ = a Coulomb friction matrix

$d_i$ = one of **D-H** parameters

$\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}})$ = the Coriolis and centrifugal forces

$l_1$ = the distance between actuator 1 and actuator 2

$\mathbf{M}(\mathbf{q})$ = an $N \times N$ inertia matrix

$\mathbf{P}_{8b,R1}$; $\mathbf{P}_{8b,L1}$ = position vectors for hinge points $R_1$ and $L_1$ in $X_{8b}Y_{8b}Z_{8b}O_{8b}$

$\mathbf{P}_{8b,R6}$; $\mathbf{P}_{8b,L6}$ = position vectors for hinge points $R_6$ and $L_6$ in $X_{8b}Y_{8b}Z_{8b}O_{8b}$

$(P_x, P_y, P_z)$ = temporary vector for inverse kinematics solution

$\mathbf{P}_t^w$ = the position of vector TCP of the end-effector in $X_wY_wZ_wO_w$

$\mathbf{q}$, $q_i$; $\dot{\mathbf{q}}$, $\dot{q}_i$; $\ddot{\mathbf{q}}$, $\ddot{q}_i$ = actuator values, velocities, and accelerations

$(r_x, r_y, r_z)$ = temporary vector for inverse kinematics solution

$\mathbf{R}_t^w$ = the orientation of the end-effector in $X_wY_wZ_wO_w$

$s_i$ = one of **D-H** parameters

$S$ = the sum of the telescopic prismatic joints $\phi_4 + \phi_5 + \phi_6$

$(t_x, t_y, t_z)$ = the position vector of TCP of the end-effector w.r.t. the last link

$\mathbf{T}_t^w$ = the position and orientation of the end-effector in $X_wY_wZ_wO_w$

$\mathbf{T}_i^{i-1}$ = **D-H** matrix transforms vectors in $X_iY_iZ_iO_i$ to $X_{i-1}Y_{i-1}Z_{i-1}O_{i-1}$

$\mathbf{T}_b^w$ = the transformation matrix from $X_bY_bZ_bO_b$ to $X_wY_wZ_wO_w$

$\mathbf{T}_t^{10}$ = the transformation matrix from $X_tY_bZ_tO_t$ to $X_{10}Y_{10}Z_{10}O_{10}$

$T_s$ = the sampling time of the servo control loop

$T_t$ = the trajectory update time

TCP = the Tool Center Point of the end-effector

$\mathbf{V}$ = a viscous friction matrix

$(x_{p0}, y_{p0})$ = the position vector of the pin point $P_0$ on body 3 in $X_3Y_3Z_3O_3$

$(x_p, y_p)$ = the position vector of the pin point $P_0$ on body 3 in $X_{3b}Y_{3b}Z_{3b}O_{3b}$

$(x_f, y_f)$ = the position vector of the pivoting point $F$ in $X_{3b}Y_{3b}Z_{3b}O_{3b}$

$X_{3b}Y_{3b}Z_{3b}O_{3b}$ = the temporary shoulder base coordinate system

$X_{8b}Y_{8b}Z_{8b}O_{8b}$ = the temporary wrist base coordinate system

$X_iY_iZ_iO_i$ = the body coordinate system for the $i$th link

$X_bY_bZ_bO_b$ = the base coordinate system of the robot arm

$X_wY_wZ_wO_w$ = the world coordinate system of the robot workcell

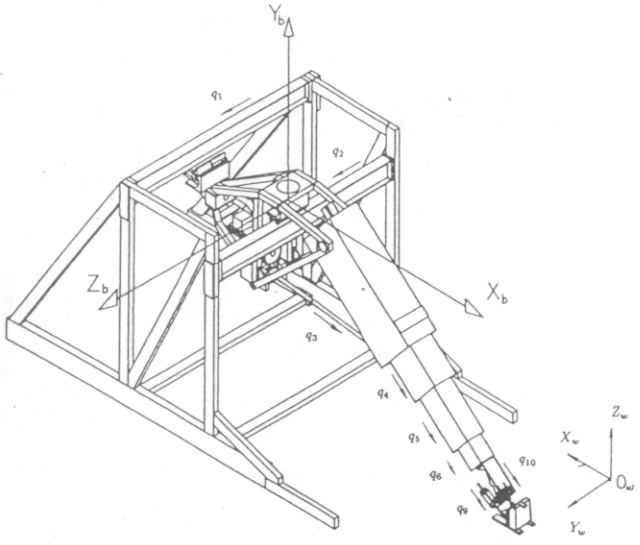


Fig. 2 The schematic configuration of the UPSarm

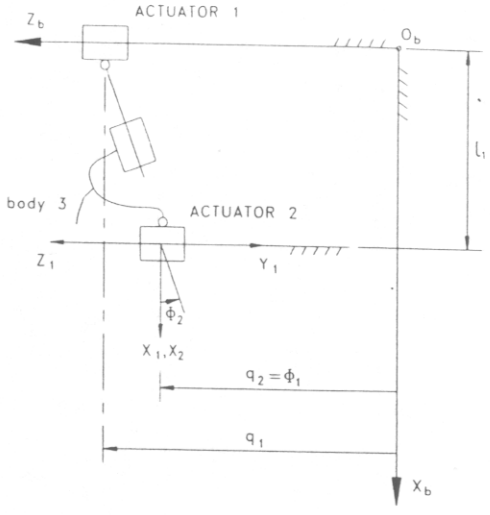


Fig. 3 The schematic diagram for actuators 1 and 2 of the UPSarm

(TCP) and orientation of the end-effector in the world Cartesian coordinate space by the 3×1 translational vector and 3×3 orientational submatrix, respectively. The effective joint space is used to represent the manipulator in an effective serial kinematic chain. An effective joint may not be a physical joint and is a virtual joint such as the second effective joint of the UPSarm. The introduction of the effective joint space is mainly for simplifying the design and analysis of hybrid serial-and-parallel-driven kinematic chains. The actuator space locates the end-effector by specifying actuator angles or translational distances in actuator axes which may be directly connected to motor shaft axes. Unlike an effective joint, the defined actuator joint always physically exists. The actuator value is a linear function of the encoder count of a motor encoder mounted on a motor shaft. Our experience indicates that distinction of effective joint space and actuator space for a redundant manipulator involving both serial and closed kinematic chains is very useful for design and analysis.

In this paper, a practical real-time implementation for manipulation of the hybrid serial-and-parallel-driven redundant UPSarm will be presented. It will show how information in actuator, effective joint, and Cartesian coordinate spaces are integrated in our path planner and trajectory generator for real-time control of the UPSarm. Although the kinematic formulations of the UPSarm are application-specific and manipulator-dependent, the developed path planner and trajectory generator are general-purpose and can be used to control any electromechanical device so long as appropriate kinematics modules are provided. The ideas and principles presented in this paper are also general and should be applicable to other serial-and-parallel-driven electromechanical devices, for which the numerical kinematic solutions may be involved.

The rest of the paper is arranged as follows. Section 2 describes the kinematic representation of the UPSarm in actuator space, effective joint space, and world Cartesian coordinate space. The direct, forward, inverse, and indirect kinematic solutions of the UPSarm based upon these three coordinate spaces are given in Section 3. The algorithms for path planning and trajectory generation of hybrid serial-and-parallel-driven robot manipulators are addressed in Section 4. The simplified dynamic model and servo control algorithm for the UPSarm presented in Section 5 are essential elements of a real-time distributed computing system implemented in Section 6. Practical manipulation examples of the UPSarm programmed in our robot language will be demonstrated in Section 7. Finally, results of the paper will be discussed in Section 8.

## 2 UPSarm in Three Coordinate Spaces

The mechanical configuration of the UPSarm is depicted in Fig. 2. Both shoulder and wrist subsystems of the manipulator are designed with closed kinematic chains primarily due to considerations of rigidity and high payload-carrying capacity of the manipulator. Figure 2 also shows the translational actuator variables of the UPSarm in its actuator space. In this paper, the actuator values and effective joint values of the UPSarm are represented by $\mathbf{q} = [q_1, q_2, \ldots, q_{10}]^T$ and $\phi = [\phi_1, \phi_2, \ldots, \phi_{10}]^T$, respectively. The schematic representation for actuators 1 and 2 is shown in Fig. 3, which is part of the top view of the UPSarm. The direct schematic representation for actuator 3 is shown in Fig. 4, which is part of the side view of the UPSarm. In fact, the mechanical device for actuator 3 is essentially a slider-crank mechanism. The three redundant prismatic joints 4, 5, and 6 provide the UPSarm with long reachability, large workspace, and dextrous maneuverability which is desired since the arm will operate inside a trailer with a constrained working space.

The three degree-of-freedom wrist of the UPSarm involves two in-parallel drive chains. Figure 5 shows the connections between the wrist base and the end-effector. The wrist consists of a platform which is connected to the end-effector, two extensible links for actuators 9 and 10, one ball joint formed by one universal joint and actuator revolute joint 8, and a base connecting the wrist to the positioning structure of the robot. The platform is connected to two extensible links through two universal joints $L_6$ and $R_6$, and to the wrist's base by another universal joint at point $O_{8b}$. The other ends of two extensible links are connected to the wrist's base through two universal joints L1 and R1. By

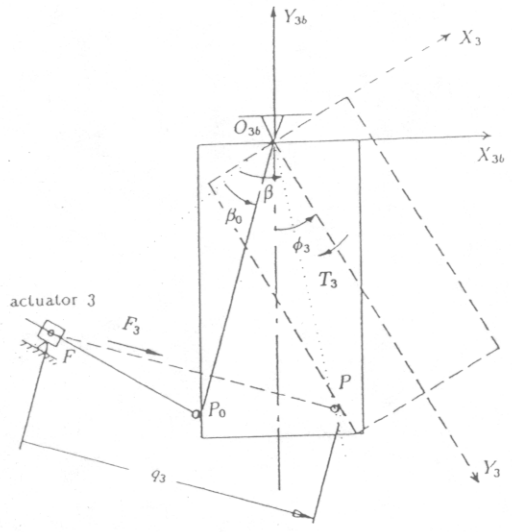


Fig. 4 The schematic diagram for actuator 3 of the UPSarm

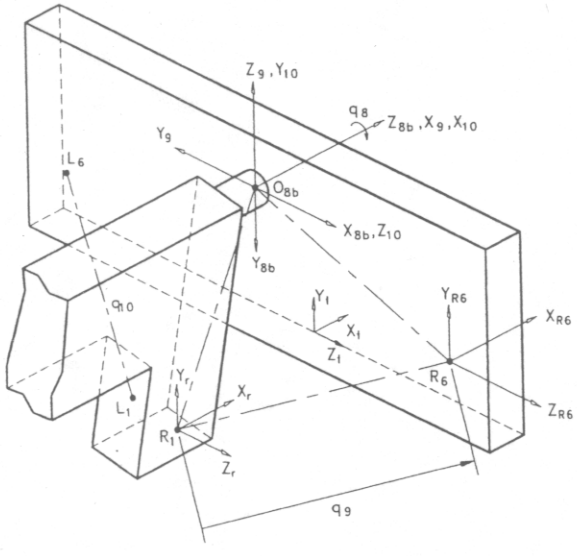


Fig. 5 The schematic diagram of the wrist

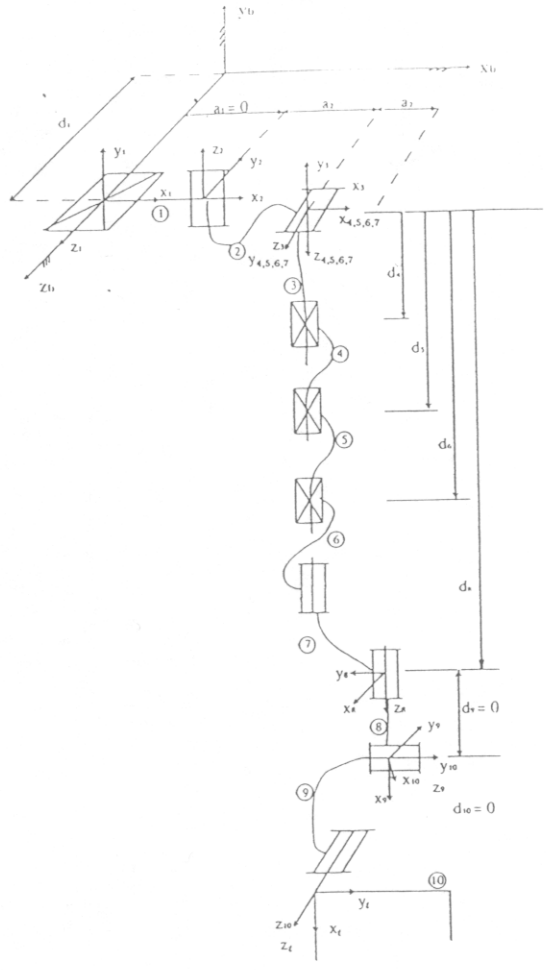


Fig. 6 The UPSarm in effective joint space

varying the link lengths ($q_9$, $q_{10}$) and the roll angle ($q_8$) of the ball joint, the platform can be manipulated with respect to the wrist's base. In order for the wrist to operate as a three degree-of-freedom mechanism, each extensible link was designed with six-degree-of-freedom and the ball joint has three-degree-of-freedom. It can be verified that the wrist structure shown in Fig. 5 has three-degree-of-freedom. The in-parallel actuation has greatly enhanced the rigidity and load-carrying capacity of the designed wrist, but it provides very small joint space for joint 8. The redundant joint 7 of the manipulating is thus introduced which will provide the UPSarm with more dexterity and workspace. Joint 7 is redundant and has essentially the same functionality as joint 8.

As one can see that the UPSarm is a hybrid serial-and-parallel-driven redundant manipulator. To facilitate the kinematic analysis, an effective joint space is introduced. Figure 6 is the equivalent kinematic representation of the UPSarm in a so-called effective joint space. in this effective joint space, the UPSarm is modeled as a ten-degree-of-freedom serial kinematic chain. Therefore, many algorithms developed for manipulation of redundant serial robot manipulators can be used. The wrist of the UPSarm shown in Fig. 6 is modeled as three effective joints with three joint axes intersecting at a common point $O_{8b}$ in Fig. 5. When the manipulator is represented in this effective joint space, the D-H notation (Denavit-Hartenberg, 1955) can be used to parameterize the geometrical information of the UPSarm. According to the D-H notation definition, the coordinate systems used for motion coordination of the UPSarm are shown in Fig. 6, where $X_bY_bZ_bO_b$ is the base coordinate system of the arm. The body coordinate system $X_iY_iZ_iO_i$ is attached to body $i$ which is the physical link between axes $i$ and $i + 1$. The D-H parameters of the UPSarm shown in Fig. 6 are listed in Table 1 where $\alpha_i$ is the angle between $Z_i$ and $Z_{i+1}$ measured along axis $X_i$, $a_i$ is the shortest distance between $Z_i$ and $Z_{i+1}$ measured along $X_i$, $d_i$ is the shortest distance between $X_{i-1}$ and $X_i$ along $Z_i$, and $\theta_i$ is the angle between $X_{i-1}$ and $X_i$ measured along $Z_i$. The joints 1, 4, 5, and 6 of the UPSarm are prismatic joints. Therefore, the D-H parameters of angles $\theta_1$, $\theta_4$, $\theta_5$, and $\theta_6$ are constants with zero values. For prismatic joints 1, 4, 5, and 6, the translational distances $s_1$, $s_4$, $s_5$, and $s_6$ are effective joint variables. For all other effective joints of the UPSarm, the distances $d_i's$ are fixed constant values and angles $\theta_i's$ are effective joint variables. The transformation matrix which

**Table 1 Denavit-Hartenberg parameters of the UPSarm in effective joint space**

| $i$ | $\alpha_{i-1}$ | $a_{i-1}$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0 | 0 | $\phi_1$ | 0 |
| 2 | −90° | 0 | 0 | $\phi_2$ |
| 3 | 90° | $a_2$ | 0 | $\phi_3$ |
| 4 | 90° | 0 | $\phi_4$ | 0 |
| 5 | 0 | 0 | $\phi_5$ | 0 |
| 6 | 0 | 0 | $\phi_6$ | 0 |
| 7 | 0 | 0 | 0 | $\phi_7$ |
| 8 | 0 | $a_7$ | $d_8$ | $\phi_8$ |
| 9 | 90° | 0 | 0 | $\phi_9$ |
| 10 | 90° | 0 | 0 | $\phi_{10}$ |

transforms the position specified in the body coordinate system $X_iY_iZ_iO_i$ to the body coordinate system$X_{i-1}$ $Y_{i-1}Z_{i-1}O_{i-1}$ can be formulated by (Craig, 1989)

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where $c\theta_i$ and $c\alpha_{i-1}$ represent $\cos\theta_i$ and $\cos\alpha_{i-1}$, and $s\theta_i$ and $s\alpha_{i-1}$ stand for $\sin\theta_i$ and $\sin\alpha_{i-1}$, respectively. According to the above definition, body 0 is the base of the robot manipulator.

The robot manipulator may also be represented by a transformation matrix $\mathbf{T}_w^t$ which defines the TCP and orientation of the end-effector of the robot arm in the world Cartesian coordinate system. In the following section, formulations for transformation of the kinematic information presented in actuator space, effective joint space, and world Cartesian coordinate space will be derived for the UPSarm.

## 3 Manipulator Kinematics

**3.1 Direct Kinematics.** The robot direct kinematics is a process which calculates the effective joint values $\phi$ when the actuator values $\mathbf{q}$ of the robot arm are given. For actuators 4, 5, 6, 7, and 8 of the UPSarm, the effective joint values equal their corresponding actuator values. Hence,

$$\phi_i = q_i \tag{2}$$

for $i$ = 4, 5, 6, 7, and 8. The direct kinematic solutions for joints 1, 2, and 3 of the shoulder subsystem of the UPSarm are straightforward. From Fig. 3, one can easily obtain the following relation for joints 1 and 2.

$$\phi_1 = q_2 \tag{3}$$

$$\phi_2 = \tan^{-1}\left(\frac{q_1 - q_2}{l_1}\right) \tag{4}$$

where $\phi_1$ and $\phi_2$ are joint values in effective joint space, and $q_1$ and $q_2$ are actuator values in actuator space for actuators 1 and 2, respectively, and $l_1$ is the distance between these two actuator axes shown in Fig. 3.

The mechanism used for joint 3 is basically a planar slider-crank mechanism as is shown in Fig. 4. The temporary base coordinate system $X_{3b}Y_{3b}Z_{3b}O_{3b}$ defined in Fig. 4 will be coincident with the body coordinate system $X_3Y_3Z_3O_3$ shown in Fig. 6 when $\phi_3 = 0$. Points $O_3$ and $F$ in Fig. 4 are the fixed points on body 3. The position vector $(x_{p0}, y_{p0})^T$ defines the pin point $P_0$ in the body coordinate system $X_3Y_3Z_3O_3$ and the position vector $\mathbf{F} = (x_f, y_f)^T$ defines the pivoting point F in the temporary base coordinate system $X_{3b}Y_{3b}Z_{3b}O_{3b}$. According to Fig. 4, if the actuator value $q_3$ is known, the effective joint value $\phi_3$ can be obtained by

$$\phi_3 = \beta - \beta_0 \tag{5}$$

where $\beta = \angle FO_3P$ and $\beta_0 = \angle FO_3P_0$. Using the cosine law, the angle $\beta$ can be calculated by

$$\beta = \cos^{-1}\left(\frac{x_f^2 + y_f^2 + x_{p0}^2 + y_{p0}^2 - q_3^2}{2\sqrt{x_f^2 + y_f^2}\sqrt{x_{p0}^2 + y_{p0}^2}}\right) \tag{6}$$

and the angle $\beta_0$ is a constant value which is calculated by

$$\beta_0 = \tan^{-1}\left(\frac{|y_f|}{|x_f|}\right) - \tan^{-1}\left(\frac{|y_{p0}|}{|x_{p0}|}\right) \tag{7}$$

The wrist subsystem of the UPSarm consists of actuators 8, 9, and 10. The wrist direct kinematics of the UPSarm is a process which will find effective joint values $\phi_8$, $\phi_9$, and $\phi_{10}$ when the actuator values $q_8$, $q_9$, and $q_{10}$ are known. The following derivation will get the wrist direct kinematic solution of the UPSarm. In Fig. 5, the temporary wrist base coordinate system $X_{8b}Y_{8b}Z_{8b}O_{8b}$ is fixed on body 7 shown in Fig. 6. The wrist base coordinate system $X_{8b}Y_{8b}Z_{8b}O_{8b}$ will be coincident with the body coordinate system $X_8Y_8Z_8O_8$ when $\phi_8 = 90°$. The hinge points $R_1$ and $L_1$ on body 7 can be defined in the wrist base coordinate system by

$$\mathbf{P}_{8b,R1} = (X_{8b,R1}, Y_{8b,R1}, Z_{8b,R1})^T \tag{8}$$

and

$$\mathbf{P}_{8b,L1} = (X_{8b,L1}, Y_{8b,L1}, Z_{8b,L1})^T \tag{9}$$

respectively. Both $\mathbf{P}_{8b,R1}$ and $\mathbf{P}_{8b,L1}$ are constant position vectors. The hinge points $R_6$ and $L_6$ on body 10 can be expressed in the body coordinate system $X_{10}Y_{10}Z_{10}O_{10}$ by

$$\mathbf{P}_{10,R6} = (X_{10,R6}, Y_{10,R6}, Z_{10,R6})^T \tag{10}$$

and

$$\mathbf{P}_{10,L6} = (X_{10,L6}, Y_{10,L6}, Z_{10,L6})^T \tag{11}$$

respectively. These two position vectors are also constant vectors. Points $R_6$ and $L_6$ defined by the position vectors $\mathbf{P}_{10,R6}$ and $\mathbf{P}_{10,L6}$ in the body coordinate system $X_{10}Y_{10}Y_{10}O_{10}$ can be expressed in the wrist base coordinate system $X_{8b}Y_{8b}$-$Z_{8b}O_{8b}$ by

$$\begin{aligned} \mathbf{P}_{8b,R6} &= \mathbf{T}_8^{8b}\mathbf{T}_9^8\mathbf{T}_{10}^9\mathbf{P}_{10,R6} \\ &= (X_{8b,R6}, Y_{8b,R6}, Z_{8b,R6})^T \end{aligned} \tag{12}$$

and

$$\begin{aligned} \mathbf{P}_{8b,L6} &= \mathbf{T}_8^{8b}\mathbf{T}_9^8\mathbf{T}_{10}^9\mathbf{P}_{10,L6} \\ &= (X_{8b,L6}, Y_{8b,L6}, Z_{8b,L6})^T \end{aligned} \tag{13}$$

respectively, where the transformation matrix $\mathbf{T}_8^{8b}$ is defined by

$$\mathbf{T}_8^{8b} = \begin{bmatrix} c\phi_8' & -s\phi_8' & 0 & 0 \\ s\phi_8' & c\phi_8' & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s\phi_8 & c\phi_8 & 0 & 0 \\ -c\phi_8 & s\phi_8 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

with $\phi_8 = \phi'_8 + 90°$ because the wrist base coordinate system $X_{8b}Y_{8b}Z_{8b}O_{8b}$ is coincident with the body coordinate system $X_8Y_8Z_8O_8$ at $\phi_8 = 90°$.

It has been shown in our extensive simulation that two-in-parallel drive chains of the wrist can be modeled by two extendible links shown in Fig. 5. The translational and orientational deviation due to this simplification are much smaller than the error tolerance of our system. By this simplification, the actuator values for $q_9$ and $q_{10}$ are equal to the length of two extendible links shown in Figure 5, which can be formulated as

$$q_9 = |\mathbf{P}_{8b,R1} - \mathbf{P}_{8b,R6}| = \sqrt{(X_{8b,R1} - X_{8b,R6})^2 + (Y_{8b,R1} - Y_{8b,R6})^2 + (Z_{8b,R1} - Z_{8b,R6})^2} \tag{15}$$

$$q_{10} = |\mathbf{P}_{8b,L1} - \mathbf{P}_{8b,L6}| = \sqrt{(X_{8b,L1} - X_{8b,L6})^2 + (Y_{8b,L1} - Y_{8b,L6})^2 + (Z_{8b,L1} - Z_{8b,L6})^2} \tag{16}$$

where $(X_{10,R6}, Y_{10,R6}, Z_{10,R6})$ and $(X_{8b,L6}, Y_{8b,L6}, Z_{8b,L6})$ are obtained from Eqs. (12) and (13), respectively. When the actuator values $q_8 = \phi_8$, $q_9$, and $q_{10}$ are known, we can express joint variables $\phi_9$ and $\phi_{10}$ as functions of actuator variables $q_8$, $q_9$, and $q_{10}$ through Eqs. (15) and (16) as follows

$$A_1 s\phi_9 + A_2 c\phi_9 + A_3 s\phi_{10} + A_4 c\phi_{10} + A_5 c\phi_9 c\phi_{10} + A_6 c\phi_9 s\phi_{10} + A_7 s\phi_9 c\phi_{10} + A_7 s\phi_9 s\phi_{10} = A_9 \tag{17}$$

$$B_1 s\phi_9 + B_2 c\phi_9 + B_3 s\phi_{10} + B_4 c\phi_{10} + B_5 c\phi_9 c\phi_{10} + B_6 c\phi_9 s\phi_{10} + B_7 s\phi_9 c\phi_{10} + B_7 s\phi_9 s\phi_{10} = B_9 \tag{18}$$

where coefficients $A'_i s$ and $B'_i s$ are functions of the actuator values $q_8$, $q_9$, and $q_{10}$ which are known in the wrist direct kinematics. Let

$$x = \tan\left(\frac{\phi_9}{2}\right), \; y = \tan\left(\frac{\phi_{10}}{2}\right) \tag{19}$$

Equations (17) and (18) become the following polynomial equations

$$a_1x^2y^2 + a_2x^2y + a_3xy^2 + a_4x^2 + a_5y^2 + a_6xy + a_7x + a_8y + a_9 = 0 \tag{20}$$

$$b_1x^2y^2 + b_2x^2y + b_3xy^2 + b_4x^2 + b_5y^2 + b_6xy + b_7x + b_8y + b_9 = 0 \tag{21}$$

where $a'_i s$ and $b'_i s$ are functions of coefficients $A'_i s$ and $B'_i s$ in Eqs. (17) and (18).

The Newton-Raphson method has been used to solve two simultaneous nonlinear Eqs. (20) and (21). For the Newton-Raphson method, good initial values are critical for convergence and computational efficiency of this numerical iterative algorithm. In our control software implementation, an efficient memory-saving three dimensional look-up table for initial values of the algorithm is introduced. For a given set of actuator values $q_8$, $q_9$, and $q_{10}$, the initial values for $\phi_9$ and $\phi_{10}$ will be obtained from the look-up table to numerically solve two simultaneous nonlinear Eqs. (20) and (21) when the relation between effective joint values and actuator values are completely unknown such as at the system startup or at the boundaries of a motion segment during path planning of the robot motion. During the trajectory generation phase for a motion segment, the joint values of the previous set point can be used as initial guesses for $\phi_9$ and $\phi_{10}$. It has been shown in simulation that under maximum actuator velocity, acceleration, and deceleration constraints of the UPSarm, the numerical solutions using the joint values $\phi_9$ and $\phi_{10}$ of the previous set point as initial guesses for the wrist direct kinematic solution are more efficient and have the same accuracy as those obtained by using the look-up table. Therefore, whenever possible, the table look-up for the wrist direct kinematic solution will be avoided during the trajectory generation. As a matter of fact, the direct kinematic solution is normally not needed at the trajectory generation phase for the real-time manipulation of the UPSarm no matter whether the target position of a motion segment is specified in actuator space, effective joint space, or Catesian space. However, it may be required for data logging and performance analysis. The direct kinematics is also needed when the feedback by the vision sensor is integrated into the system.

**3.2 Forward Kinematics.** The forward kinematics of the robot manipulator is a process which calculates the TCP location and orientation of the end-effector with given effective joint values $\phi$. The forward kinematic solution for the UPSarm is straightforward. When the effective joint values $\phi = [\phi_1, \phi_2, \ldots \phi_{10}]^T$ are given, the forward kinematic solution of the UPSarm can be simply expressed as follows.

$$\mathbf{T}_t^w = \begin{bmatrix} \mathbf{R}_t^w & \mathbf{P}_t^w \\ \mathbf{O} & 1 \end{bmatrix} = \mathbf{T}_b^w \mathbf{T}_{10}^b \mathbf{T}_t^{10} = \mathbf{T}_b^w \mathbf{T}_1^b \mathbf{T}_2^1 \cdots \mathbf{T}_{10}^9 \mathbf{T}_t^{10} \tag{22}$$

where the rotation matrix $\mathbf{R}_t^w$ gives the orientation of the end-effector and vector $\mathbf{P}_t^w$ locates the TCP of the end-effector represented in the world coordinate system $X_wY_w$-$Z_wO_w$. The coordinate transformation matrix $\mathbf{T}_{i+1}^i$ is defined in Eq. (1). The transformation matrix $\mathbf{T}_b^w$ transforms the base coordinate system to the world coordinate system and transformation matrix $\mathbf{T}_t^{10}$ transforms the coordinates expressed in the tool coordinate system $X_tY_tZ_tO_t$ to the coordinates expressed in the body coordinate system $X_{10}Y_{10}Z_{10}O_{10}$. According to Figs. 2 and 6, these two transformation matrices can be expressed as

$$\mathbf{T}_b^w = \begin{bmatrix} -1 & 0 & 0 & b_x \\ 0 & 0 & 1 & b_y \\ 0 & 1 & 0 & b_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

and

$$\mathbf{T}_t^{10} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

where $b_x$, $b_y$, $t_x$, $t_y$, and $t_z$ are constant values, and $b_z$ is a variable which will be updated during operation of the UPSarm.

**3.3 Inverse Kinematics.** The inverse kinematics refers to the process which computes effective joint values $\phi$ when

the TCP position vector and orientation of the end-effector in the world coordinate system are given. The pose of the end-effector is usually specified by the transformation matrix $\mathbf{T}_t^w$. For a redundant robot manipulator, multiple effective joint values from the inverse kinematic solution exist for a given TCP location and orientation of the end-effector. Much research has been done in resolving this redundancy resolution by using various optimization schemes (Whitney, 1969; Kazerounian and Nedungadi, 1987) or for collision avoidance (Oh et al., 1984). The redundancy of the UPSarm is resolved by our application-specific rules. In order to solve the inverse kinematics problem, we can rearrange Eq. (22) as

$$(\mathbf{T}_b^w)^{-1}(\mathbf{T}_t^w)(\mathbf{T}_t^{10})^{-1} = \mathbf{T}_{10}^b \tag{25}$$

or

$$\begin{pmatrix} \mathbf{R} & \mathbf{P} \\ 0 & 1 \end{pmatrix} = \mathbf{T}_7^b \begin{pmatrix} \mathbf{R}_{10}^7 & \mathbf{P}_{10}^7 \\ 0 & 1 \end{pmatrix} \tag{26}$$

where transformation matrices $\mathbf{T}_b^w$, $\mathbf{T}_t^w$, and $\mathbf{T}_t^{10}$ are known. If we define $[P_x, P_y, P_z]^T = \mathbf{P}$ and $[r_x, r_y, r_z] = \mathbf{P}_{10}^7 = [a_7, 0, d_8]^T$, then the translational part of Eq. (26) can be described by equation

$$(\mathbf{T}_3^b)^{-1} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix} = \mathbf{T}_7^3 \begin{pmatrix} r_x \\ r_y \\ r_z \\ 1 \end{pmatrix} \tag{27}$$

Using Eqs. (1), (23), and (24), and D-H parameters defined in Table 1, the above equation can be expanded as

$$P_x c\phi_2 c\phi_3 + P_y s\phi_3 - P_z s\phi_2 c\phi_3 + \phi_1 s\phi_2 c\phi_3 = r_x c\phi_7 - r_y s\phi_7 \tag{28}$$

$$-P_x c\phi_2 s\phi_3 + P_y c\phi_3 + P_z s\phi_2 s\phi_3 + a_2 s\phi_3 - \phi_1 s\phi_2 s\phi_3 = -r_z - S \tag{29}$$

$$P_x s\phi_2 + P_z c\phi_2 - \phi_1 c\phi_2 = r_x s\phi_7 + r_y c\phi_7 \tag{30}$$

where $S = \phi_4 + \phi_5 + \phi_6$ is the sum of the translational distances of prismatic joints 4, 5, and 6. The redundancy of the UPSarm at the path planning phase is resolved as follows. As is mentioned before that joint 7 is introduced mainly for effectively enlarging the joint space of joint 8, in our specific application, $\phi_7$ either equals 0 or $\pi$ corresponding to the front-loading mode or top-loading mode of the UPSarm, respectively. If a package is loaded to the stacking position in horizontal direction during the final motion it is called front-loading. If a package is loaded to the stacking position downward during the final motion, it is called top-loading. Once the package loading mode is chosen by our stacking algorithm, the joint value $\phi_7$ will be assigned zero for front-loading or $\pi$ for top-loading. The small orientational deviation is adjusted by controlling joint 8. Notice that the value of $\sin(\phi_7)$ is zero if $\phi_7$ is 0 or $\pi$, and $r_y$ is also zero. In our application, the redundant joint 2 is introduced essentially to increase the joint limits for joint 1. If the stacking position is reachable without moving joint 2, joint 2 will be locked; otherwise, joint 2 will move in order to reach the stacking position. This algorithm can be expressed as follows.

$$\phi_1 = \begin{cases} P_z, & \text{if } \phi_{1,\min} \le P_z \le \phi_{1,\max} \\ \phi_{1,\min}, & \text{if } P_z < \phi_{1,\min} \\ \phi_{1,\max}, & \text{if } P_z > \phi_{1,\max} \end{cases} \tag{31}$$

where $\phi_{1,\min}$ and $\phi_{1,\max}$ are the minimum and maximum joint limits for effective joint 1, respectively. The effective joint value $\phi_2$ is

$$\phi_2 = \begin{cases} 0, & \text{if } \phi_{1,\min} \le P_z \le \phi_{1,\max} \\ \tan^{-1}\left(\dfrac{P_z - \phi_1}{-P_x}\right), & \text{otherwise} \end{cases} \tag{32}$$

The inverse kinematic solutions for $\phi_1$ in Eq. (31) and $\phi_2$ Eq. (32) are used only at the path planning phase. To ensure the motion continuity, the following inverse kinematic solutions for $\phi_1$ and $\phi_2$ are used by the trajectory generator

$$\phi_1 = \phi_{1,\text{begin}} + \frac{\phi_{1,\text{end}} - \phi_{1,\text{begin}}}{P_{z,\text{end}} - P_{z,\text{begin}}}(P_z - P_{z,\text{begin}}) \tag{33}$$

where $\phi_{1,\text{begin}}$ and $P_{z,\text{begin}}$ are the values at the beginning of a motion segment; and $\phi_{1,\text{end}}$ and $P_{z,\text{end}}$ are the values at the end of the motion segment. These values are obtained at the path planning phase according to the target position of each motion segment specified by the robot program. Note that the end position of the previous segment will become the beginning position of the next motion segment. The details of path planning and trajectory generation will be discussed in the next section. The effective joint value $\phi_2$ is derived from Eq. (30) as follows

$$\phi_2 = \begin{cases} 0, & \text{if joint 2 is moved} \\ \tan^{-1}\left(\dfrac{P_z - \phi_1}{-P_x}\right) - \tan^{-1}\left(\dfrac{r_x s\phi_7 + r_y c\phi_7}{-\sqrt{P_x^2 + (P_z - \phi_1)^2 - (r_x s\phi_7 + r_y c\phi_7)^2}}\right), & \text{otherwise} \end{cases} \tag{34}$$

The effective joint value $\phi_3$ can be obtained from Eq. (28)

$$\phi_3 = \tan^{-1}\left(\frac{k}{-P_y}\right) - \tan^{-1}\left(\frac{r_x c\phi_7 - r_y s\phi_7}{\sqrt{P_y^2 + k^2 - (r_x c\phi_7 - r_y s\phi_7)^2}}\right) \tag{35}$$

where the variable $k$ is defined as

$$k = P_x c\phi_2 - P_z s\phi_2 - a_2 + \phi_1 s\phi_2$$

The effective total translational motion $S$ for joints 4, 5, and 6 can be calculated from Eq. (29)

$$S = P_x c\phi_2 s\phi_3 - P_y c\phi_3 - P_z s\phi_2 s\phi_3 - a_2 s\phi_3 + \phi_1 s\phi_2 s\phi_3 - r_z \tag{36}$$

One of methods that we have used to resolve the prismatic redundancy for joints 4, 5, and 6 of the UPSarm is to set the displacements of these joints by the formula

$$\phi_i = \frac{S * \phi_{i,\max}}{\phi_{4,\max} + \phi_{5,\max} + \phi_{6,\max}} \tag{37}$$

where $i = 4$, 5, and 6, $\phi_{i,\max}$ is the maximum joint limit for prismatic joint $i$, and $S$ is obtained by Eq. (36). The minimum joint limits for joints 4, 5, and 6 are zeroes for the UPSarm.

The wrist effective joint values $\phi_8$, $\phi_9$, and $\phi_{10}$ can be obtained by using the following orientational equation derived from Eq. (22)

$$\mathbf{R}_{10}^{7} = (\mathbf{R}_7^w)^{-1} (\mathbf{R}_t^w) (\mathbf{R}_t^{10})^{-1} \tag{38}$$

Since orientational matrices $\mathbf{R}_7^w$, $\mathbf{R}_t^w$, and $\mathbf{R}_t^{10}$ are known, Eq. (38) can be expanded as follows

$$\begin{bmatrix} c_8c_9c_{10} + s_8s_{10} & -c_8c_9c_{10} + s_8c_{10} & c_8c_9 \\ s_8c_9c_{10} - c_8s_{10} & -s_8c_9s_{10} - c_8c_{10} & s_8s_9 \\ s_9c_{10} & -s_9s_{10} & -c_9 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{11} & r_{12} & r_{13} \end{bmatrix} \tag{39}$$

where $s_i$ and $c_i$ represent sin $\phi_i$ and cos $\phi_i$, respectively; and $r_{ij}$ will be calculated by the expression on the right-hand side of Eq. (38). From Eq. (39), the wrist effective joint values can be obtained by the following formulas

$$\phi_8 = 2 \tan^{-1}\left(\frac{r_{23}}{r_{13} + \sqrt{r_{31}^2 + r_{32}^2}}\right) \tag{40}$$

$$\phi_9 = 2 \tan^{-1}\left(\frac{\sqrt{r_{31}^2 + r_{32}^2}}{1 - r_{33}}\right) \tag{41}$$

$$\phi_{10} = 2 \tan^{-1}\left(\frac{-r_{32}}{r_{31} + \sqrt{r_{31}^2 + r_{32}^2}}\right) \tag{42}$$

**3.4 Indirect Kinematics.** The robot indirect kinematics is a process which calculates actuator values $\mathbf{q}$ with given effective joint values $\phi$ of the robot arm. As in direct kinematics, the effective joint values for joints 4, 5, 6, 7, and 8 of the UPSarm are the same as their corresponding actuator values. According to Fig. 3, when the effective joint values $\phi_1$ and $\phi_2$ are known, the actuator values $q_1$ and $q_2$ of the UPSarm can be easily obtained by the following formulas

$$q_1 = l_1 \tan \phi_2 + q_2 \tag{43}$$

$$q_2 = \phi_1 \tag{44}$$

When the effective joint value $\phi_3$ is known, the actuator value $q_3$ can be calculated by

$$q_3 = |\mathbf{P} - \mathbf{F}| = \sqrt{(x_f - x_p)^2 + (y_f - y_p)^2} \tag{45}$$

where the constant position vector $\mathbf{F} = (x_f, y_f)^T$ is the coordinates of the pivoting point F in the temporary base coordinate system $X_{3b}Y_{3b}Z_{3b}O_{3b}$. The position vector $\mathbf{P} = (x_p, y_p)^T$ for point $P$ in the temporary base coordinate system $X_{3b}Y_{3b}Z_{3b}$-$O_{3b}$ can be calculated by

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} c\phi_3 & -s\phi_3 \\ s\phi_3 & c\phi_3 \end{pmatrix} \begin{pmatrix} x_{p0} \\ y_{p0} \end{pmatrix} \tag{46}$$

With given effective joint values $\phi_8$, $\phi_9$, and $\phi_{10}$, the actuator values $q_9$ and $q_{10}$ can be computed by Eqs. (15) and (16) which is referred to as the wrist indirect kinematics of the UPSarm.

## 4 Path Planning and Trajectory Generation

The direct, forward, inverse, and indirect kinematic solutions derived in the previous section will be used for manipulation of the UPSarm. These formulations are essential for path planning and trajectory generation of a hybrid serial- and parallel redundant robot manipulator. In the context of this paper, the path planning refers to the process which calculates the required parameters, such as duration times of acceleration, slew motion, and deceleration, etc. of motion segments based on certain physical constraints. The trajectory generation is a process which computes the intermediate positions of the robot arm in a motion segment based upon the information obtained by the path planner. The physical constraints of the robot manipulator can be specified in many different ways. For practical application and implementation simplicity, we assume that each actuator has limited maximum velocity, acceleration, and deceleration, which are directly related to the available motor torque. These motion constraints are specified in actuator space. Path planning will mainly calculate the necessary parameters based upon the information in actuator space. Mechanical joint limits of each actuator are also physical constraints of the manipulator, which have to be checked in path planning and trajectory generation phases of the robot manipulation. In a less rigorous terminology, the motion of a robot manipulator may be classified as point-to-point and continuous-path motions, which are implemented in our path planner and trajectory generator for practical manipulation of the UPSarm. In this section, the path planning and trajectory generation for these motions will be discussed.

**4.1 Point-to-Point Motion.** Suppose that A is the current position of the robot arm, B is the desired target position, and the objective is to move the robot arm from A to B, the point-to-point motion will generate a coordinated motion in which all actuator motions begin as the robot arm leaves from position A and end at the same time as the robot arm arrives at B. For a hybrid serial-and-parallel-driven manipulator, the position of the robot arm can be defined by using one of three specifications: actuator values, effective joint values, and a homogeneous transformation matrix which defines the TCP and orientation of the end-effector. As mentioned before, the process of path planning mainly uses information in actuator space to calculate the necessary parameters for the trajectory generator. The actuator information at boundaries of a motion segment are needed by the path planner. Therefore, if the target position of the robot arm is specified using effective joint values, the indirect kinematic solution presented in the previous section will be used to get the actuator values for the computation of the required parameters. For UPSarm, there is one-to-one correspondence between an actuator value and an effective joint value within the robot arm working joint limits. In general, multiple solutions may exist for transformation between actuator values and effective joint values, which will add additional complexity to a system. If the target position of the

robot arm is specified using a transformation matrix of the end-effector, the inverse and indirect kinematic solutions should be used to obtain the actuator values at the target position of a segment. This target position, which will become the initial position of the next segment, will be stored in the program so that the information of the initial position for the current planning segment is always available for the path planner. If the robot arm is redundant or has multiple configurations for a given pose of the end-effector, the final configuration of the robot arm at the target position may also need to be specified. If a motion involves only a single actuator, the actuator is first accelerated, then slewed at a constant velocity, and finally decelerated to its target actuator position. Two acceleration motion profiles have been implemented in our path planer and trajectory generator. One is of square wave acceleration and the other is of sinusoidal acceleration. Both square wave and sinusoidal motion profiles can also be used for deceleration. The sinusoidal motion profile will provide smoother acceleration, but the maximum acceleration will be reduced by a factor of $2/\pi$ in comparing with the square wave acceleration under the same constraints of the maximum velocity, acceleration, and deceleration of the actuator. If the point-to-point coordinated motion involves multiple actuators, the calculated maximum duration times of acceleration, slew motions, and decelerations for each individual actuator will be selected as the planned times for acceleration, slew motion, and deceleration of the planned motion segment, respectively. This is a very conservative approach, but it ensures that all constraints of the actuators will not be violated during motion of the robot arm.

**4.2 Continuous-Path Motion.** There are two kinds of continuous-path motions: path tracking and blending point-to-point motion segments. The TCP of the end-effector of the robot arm will follow a well-defined trajectory in continuous-path tracking, which has many practical applications. For example, in our specific applications, the TCP of the UPSarm will follow a straight-line path during the final stage of stacking packages. Blending motion segments tends to round the corners between segments.

Three path geometries: straight-line, circular curve, and spline curve, have been implemented in our path planner and trajectory generator for path tracking. In all these three prescribed continuous-path motions, the TCP and orientation change of the end-effector of the robot arm can be smoothly accelerated in either sinusoidal or square wave motion profile, moved at constant velocity, and then decelerated also in either sinusoidal or square wave motion profile. While the TCP of the end-effector of the robot arm is moved along the prescribed path, the orientation of the end-effector can be smoothly changed from its beginning position to the final position, which is referred to as orientation interpolation. The orientation of the end-effector is interpolated by essentially using a technique similar to the so-called two-rotation method, which involves two rotations in order to orient the end-effector from its initial orientation to the target orientation (Paul, 1979). The first rotation is about the common normal vector which normals to both approach vectors of the end-effector at the initial and final positions of the motion segment. The approach vector is defined as a unit vector along the $z_t$ axis of the tool coordinate system at the end-effector shown in Fig. 6. The direction along $z_t$ is usually the one along which the end-effector of the robot arm approaches workpieces. The second rotation is about the approach vector itself from the initial position to the final position. These two rotation interpolations are carried out at each trajectory generation cycle simultaneously while the TCP of the end-effector is moving along the prescribed path. Some other rotation interpolation schemes, such as rotation about a single vector along which the robot arm will move the end-effector from its initial position to the final position, are available. For more information, one can refer to (Paul, 1979; Taylor, 1979; Brady, 1983).

In planning continuous-path for the TCP of the end-effector, the position vector of the TCP and orientation of the end-effector in the world coordinate system will be needed. As we discussed before, the position of the robot arm can be defined in one of three coordinate spaces. If the position is defined using a transformation matrix, the TCP and orientation of the end-effector is readily available for path planning. If the position is specified using effective joint values or actuator values, the forward and/or direct kinematic solutions presented in the previous section will be needed. However, if the robot arm has multiple configurations for a given position of the end-effector, then the desired configuration for the target position should be specified. If the target position is specified using effective joint values, the forward kinematic solution will be sufficient to get the information for planning the prescribed path. If the position is specified using actuator values, there is no confusion of the robot configurations for the UPSarm. But the direct and forward kinematic solutions have to be used to find the TCP and orientation of the end-effector. Since no prior knowledge of the effective joint values are known for the target position, a look-up table has to be used to get initial guesses of the effective joint angles $\phi_9$ and $\phi_{10}$ for the numerical wrist direct kinematic solution discussed in the previous section.

During the trajectory generation phase, the inverse and indirect kinematic solutions will be obtained for each intermediate position when the TCP of the end-effector of the robot arm is traveling along the prescribed path. As discussed in the previous section, formulas of the inverse kinematic solution used in the trajectory generator are different from ones used in the path planner for the redundant UPSarm. The joint redundancy will be resolved at both planning and trajectory generation phases. For a robot manipulator with multiple configurations for a given position of the end-effector, if the robot arm has to change the configuration in order to keep the TCP of the end-effector moving along the prescribed path, the robot singular position will be encountered where the robot will switch from one configuration to another. It is impossible for the robot to change its configuration under this situation. Therefore, during the path planning phase, if the configuration of the robot arm at the initial position is different from the one at the final target position, the path planner will send an error message to warn the user of the potential failure so that an alternative path may be chosen. The wrist with its three joint axes intersecting at a common point is one of the most popular designs because it generally renders a manipulator with closed-form inverse kinematic solutions (Pieper, 1969). However, for a wrist with three joint axes intersecting at a common point, the

singular position of the wrist may be encountered when the second joint of the wrist reaches the value zero. This singular position corresponds to $\phi_9$ being zero for the UPSarm. There are applications in which, except at the initial and target positions of a motion segment, the orientations of the end-effector are not concerned while the TCP of the end-effector is moving along a prescribed path. In this case, the path planning and trajectory generation for the wrist can be performed using the point-to-point motion scheme. Thus, the wrist singularity of the inverse kinematic solution can be avoided. Once the wrist joint solutions for $\phi_8$, $\phi_9$, and $\phi_{10}$ are obtained, the effective joint values $\phi_1, \phi_2, \ldots, \phi_7$ then can be obtained according to the prescribed path for the TCP of the end-effector. This combined Cartesian-joint motion has been implemented for the UPSarm. The combined Cartesian-joint inverse kinematic solution for the UPSarm can be found in (Cheng et al., 1991). It should be pointed out that our path planner and trajectory generator are general-purpose and can be used to control multiple electromechanical devices or robot manipulators. But, our manipulator is application-specific and designed so that it will not operate near singular positions under normal working conditions.

Besides the continuous path tracking, multiple motion segments can also be blended together to form a continuous path. Suppose that $A$ is the initial position of the robot arm, C the final target position, and B the specified intermediate position. The continuous-path obtained by blending two point-to-point motion segments will move the robot arm from A to C via B without stopping at the intermediate position B. This continuous-path may be used to avoid obstacles so that the robot arm will move near position B but not necessary to stop the robot arm at B. This motion feature has many practical applications. As discussed before, each point-to-point path segment is made up of three regions: acceleration, slew motion, and deceleration. The continuous-path transition of two segments is started at the end of slew motion of the first segment and ended at the beginning of the slew motion of the second segment. Based upon the position and velocity of the trajectory at these two transition points, a cubic polynomial can be derived. Using this polynomial as a transition trajectory of two adjacent segments, the positions and velocities of the blended trajectory are continuous. In the motion planning stage, all coefficients of this polynomial are calculated. This polynomial will be used by the trajectory generator to interpolate the trajectory at the transition region for blended continuous-path motion.

## 5 Dynamics and Control

In general, the dynamic model for an N degree-of-freedom robot manipulator with rigid bodies can be formulated as

$$\tau_f = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \mathbf{V}\dot{\mathbf{q}} + \mathbf{C}\ \text{sign}\ (\dot{\mathbf{q}}) \quad (47)$$

where $\tau_f$ is the generalized forces provided by the robot actuators; $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are actuator values, velocities, and accelerations of the robot manipulator, respectively; $\mathbf{M}(\mathbf{q})$ is an $N \times N$ inertia matrix; $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}})$ are the Coriolis and centrifugal terms, $\mathbf{g}(\mathbf{q})$ are the gravity terms, $\mathbf{V}$ is a viscous friction matrix; and $\mathbf{C}$ is a Coulomb friction matrix. For serial robot manipulators with or without redundancy, a general-purpose efficient recursive scheme for the computation of the torques $\tau_f$ can be found in the literature (Luh et al., 1980a; Cheng and Gupta, 1992, 1993). However, for hybrid serial-and-parallel-driven redundant robot manipulators, no such efficient general computational scheme is available. On the other hand, the complexity and limited computational capability of a system may make the computation of the complete dynamic model unfeasible. Therefore, like many other industrial robot manipulators, a significant simplification of the dynamic model of the UPSarm has been made in our control system. For our practical application, only the principal inertia forces of the term $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}$ in Eq. (47) for major joints 1 to 6 are calculated. The Coriolis and centrifugal forces $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}})$ and viscous friction forces $\mathbf{V}\dot{\mathbf{q}}$ are ignored for all joints. The gravity forces $\mathbf{g}(\mathbf{q})$ are considered for major joints 1 to 6 only. The Coulomb friction forces are implemented as $\mathbf{C}\ \text{atan}(\dot{\mathbf{q}})\ \frac{2}{\pi}$ where the diagonal coefficient matrix $\mathbf{C}$ is obtained experimentally. For major joints 1 to 6 except for joint 3, the computation of principal inertia forces and gravity forces are straightforward. For joint 3, only the inertia of the crank of the slider-crank mechanism shown in Fig. 4 is considered. Note that the summation of the weights of bodies 3 to 10 is more than 500 lb. When the moment $T_3$ is known, the force $F_3$ along the actuator motor axis shown in Fig. 4 is derived by the Principle of Virtual Work as follows.

$$F_3 = T_3 \frac{\partial \phi_3}{\partial q_3} \quad (48)$$

where the partial derivative $\partial\phi_3/\partial q_3$ can be obtained from Eqs. (45) and (46).

Many robot control algorithms have been developed in the past two decades (Luh et al., 1980b; Whitcomb et al., 1991). However, in order to achieve good performance by these model-based control algorithms, an accurate dynamic model of the robot manipulator is needed. But, in practice, it is very difficult to get this ideal model, even without considering the compliances, frictions, and flexibilities of the robot arm. In addition, the computation of a complete dynamic model updated at the fast servo rate $T_s$ may not be feasible with the limited computational resource under the constraint of costs. This is also a concern for practical implementation of many adaptive control algorithms (Craig et al., 1986; Slotine and Li, 1987). The previous work which is closely related to our control scheme was presented in (Liegeois et al., 1980). The control algorithm implemented by Liegeois et al. is a PD-control with feedforward compensation torques. Their feedforward torque terms were computed off-line from the nominal trajectory during the programming of the robot. However, this off-line programming scheme can not be applied to our control system since there is no way to plan all possible motion trajectories of the robot arm beforehand in our dynamic operation environment. Therefore, for our practical application, a PID-based nonlinear feedforward control scheme is used for our servo control system. Comparing with a PD-control, the addition of an integral term in the servo control loop will have better performance for path-tracking when the control parameters are appropriately tuned. In our implementation, the feedforward torque is computed on the target board updated at the trajec

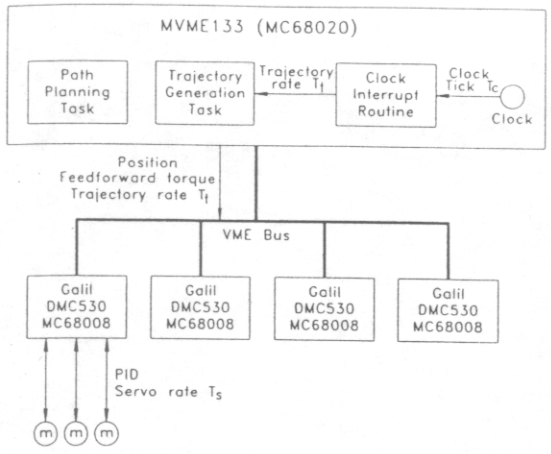


**Fig. 7 Real-time distributed computing system**

tory rate whereas the PID fast loop is computed on the servo control boards at 1000 hz update rate. In our system, no velocity and acceleration sensors, such as tachometers and accelerometers, are installed to capture the velocity and acceleration of a motor. However, since the sampling time is known, the velocity and acceleration of an actuator at each sampling tick can be obtained numerically.

## 6 Real-Time Implementation

The above-presented algorithms for path planning, trajectory generation, and control were first implemented in the C programming language and tested under a Unix operating system. The program has been ported to pSOS$^+$, a real-time operating system environment used for real-time motion coordination of the UPSarm. The computer hardware for real-time motion control is shown in Fig. 7, which is a VMEbus-based real-time distributed computing system. There are five VME boards in an industrial standard VME chassis. The Motorola MVME133 board is the target host running a MC68020 microprocessor with clock speed 16.67 MHz and 1Mb dynamic RAM. This target board is connected to an Ethernet network. The C program for path planning and trajectory generation is cross-compiled on a Sun SPARCstation and then down loaded to the target board via the Ethernet connection. The user interface with the robot arm is accomplished through two RS-232 serial communication ports of the target board. One serial communication port is connected to the SPARCstation for keyboard input of our robot language instructions and the other is connected to a teach pendant. The teach pendant is tailored from an off-the-shelf Termiflex hand-held control/display terminal for our specific application. Four Galil DMC530 VME servo control boards are responsible for servo control and low level communication of the control system. The motion command generated from the target board is sent to these servo boards via the VMEbus. Each servo board controls three brushless DC motors. The PID closed-loop control algorithm executed on a MC68008 microprocessor controls motion of each brushless DC motor. The sampling time interval $T_s$ of the servo control is 1 ms. Each servo board can also provide eight PLC-type (Programmable Logic Control) software controllable I/O ports, which have been used to control the emergency stop, motor brakers, and gripper.

Path planning task and trajectory generation task are two major tasks executed on the target board under real-time operating system pSOS$^+$. The system real-time clock with preset time unit $T_c$ of 1 ms enables the clock interrupt service routine to keep track of the system cycle time. A priority-based preemptive algorithm has been used for real-time task scheduling, which means that at any point in time, the running task is the one with highest priority among all ready-to-run tasks. If a task, which has higher priority than the current running task, is ready to run, it will take over the CPU resource and the current running task will be swapped out. In our implementation, the path planning task is created with lower priority than the trajectory task. The following paragraph describes how a motion path is planned and the trajectory is generated in real time.

The clock interrupt service routine is interrupted by the real-time clock every $T_c$ ms. There is an integer variable in the clock interrupt service routine which is incremented at every clock interruption. This integer cycle count variable will keep track of the trajectory update cycle time. When this number is equal to the trajectory update cycle number $T_t$, which is 32 corresponding to 32 ms cycle time in our current implementation, the clock interrupt service routine will resume the suspended trajectory generation task and reset the cycle count variable to zero. At the beginning of each trajectory update cycle, the trajectory task resumed by the clock interrupt service routine in the target board will send a set of motor encoder counts, which are linear functions of actuator values $\mathbf{q}$, and feedforward torques in DAC counts to servo boards via the VMEbus. After communication with servo boards, the trajectory task will increment the time tick and calculate the new set of actuator values and feedforward torques for the next position of the trajectory. After these activities, the trajectory task will suspend itself so that it will not contend for the CPU time. Once the trajectory task has been suspended, the planning task which has lower priority will become the running task and take over the CPU time of the target board. The path planning task will execute our robot language instructions if the user has typed in any command on the keyboard. For a motion statement, the path planning task will plan the path for the next motion segment based upon the target positions of the current and next segments along with physical constraints of the system. The collision detection may also be performed in this task (Cheng, 1993). In the meantime, the real-time clock keeps interrupting the clock interrupt service routine which is responsible for maintaining the trajectory cycle time. As discussed before, at every $\mathbf{T}_t$ ms, the clock interrupt service routine will reset the cycle count and resume the suspended trajectory task. Since the trajectory task has higher task priority than the path planning task, the real-time operating system will swap out the path planning task when a task with higher priority is ready to run. The trajectory task will start running right after it is resumed by the clock interrupt service routine. The next trajectory update cycle will start. The process described above will repeat every trajectory update cycle. If necessary, the encoder count of an actuator motor may also be obtained from servo boards via the VMEbus by the trajectory task or clock interrupt service routine on the target board for data logging.

Since the clock interrupt service routine is invoked every

clock tick of the system, it has been kept at the briefest form possible. At the beginning of every trajectory update cycle, a new set of encoder counts and feedforward torques is sent to servo boards. Therefore, the computation of this information in the trajectory task must be completed within the trajectory update cycle time $T_t$. If for some reason, it could not finish these calculations, a new set of encoder counts will not be available to the servo controller, which would result in a non-smooth motion trajectory and the desired path of the robot arm, say a straight-line, will not be achieved. Therefore, the trajectory task has also been kept as simple as possible in our software design and will guarantee the completion of the computations for the next set of encoder counts and feedforward torques. The computations, which can be carried out at the path planning phase rather than at the trajectory generation phase, will be performed in the path planning task. The path planning task will be swapped out at the beginning of every trajectory update cycle. After the trajectory task is suspended, it will gain the CPU resource and restart the execution at the place where the task was previously stopped. If the planning task has insufficient time to run, it is likely that a continuous-path will be broken up as point-to-point motions. As one can see that the clock interrupt routine, trajectory task, and path planning task are organized in hierarchy. The clock interrupt routine is most time critical and the trajectory task is the second. Although the path planning task has lower priority in comparing with other two, but, it is also time critical. Therefore, in our software design, all computations are attempted to be carried out in a less time critical task so that they will not contend the CPU time with more time critical tasks. Whenever possible, the computations will be performed at the system initialization stage so that the CPU time can be more effectively used for real-time motion coordination.

## 7 Motion Examples

Two robot motion examples in this section will show how information presented in actuator space, effective joint space, and Cartesian coordinate space are integrated in our path planner and trajectory generator for practical manipulation of the hybrid serial-and-parallel-driven UPSarm.

**7.1 Single Joint Motion.** The first motion example moves the robot manipulator from point A to point C via point B with robot arm stopped at point B. The robot motions are specified using our robot language program which is given in Listing 1.

LISTING 1

```
acceleration 0.5
move @3 50.75
speed 0.15
acceleration 0.2
disable sinusoidalaccel
disable sinusoidaldecel
move #3 90
```

The actuator value (inch), velocity (in/s), and acceleration ($in/s^2$) for actuator 3 obtained by running the program in Listing 1 are shown in Fig. 8, where the solid line is the actuator value with respect to trajectory update cycle tick, dotted line for linear velocity, and dashed line for linear acceleration. Each trajectory update tick corresponds to 32 ms in our current implementation. In our system, the default speed, acceleration, and deceleration for the robot motion are twenty percent of the maximum defined ones. The first program statement in Listing 1 changes the default acceleration from twenty percent of the maximum acceleration of the system to fifty percent. For move statement of our robot language, the symbol '@' means that all actuator values of the robot arm will keep their current values, except the actuator indicated by the integer value following the symbol '@'. The last parameter of this move statement specifies the target actuator value. Hence, the second statement of the program will move actuator 3 from its current position, which is the home position after calibration in this example, to the position with the actuator value of 50.75 inches. Since the default motion profile is a sinusoidal motion, the robot will first be accelerated at fifty percent of the maximum acceleration to its twenty percent of the maximum speed in sinusoidal acceleration profile. At time $t_1$, the robot arm then slows at the twenty percent of the maximum speed till time $t_2$. The robot arm will be decelerated at its default deceleration which is twenty percent of the maximum deceleration from time $t_2$ to $t_3$. At time $t_3$, actuator 3 of the robot arm reaches 50.75 ins. The duration times for acceleration, slew motion, and acceleration of this motion segment are 20, 92, and 50 trajectory update ticks, respectively. The robot language statements following the first move statement in Listing 1 change the characteristics of the next motion segment. The speed for the next segment is changed to fifteen percent of the maximum speed by statement `speed 0.15`. The acceleration is changed back to the original default value. The motion profiles for acceleration and deceleration are toggered to the square wave profile by disabling the sinusoidal acceleration and deceleration. The final target position is specified by using an effective joint value indicated by the symbol '#' in the move statement. So last statement in the program will move effective joint 3 of the robot arm from position B to 90 degrees with fifteen percent of the maximum speed, twenty percent of maximum acceleration and deceleration in square wave profiles. The duration times

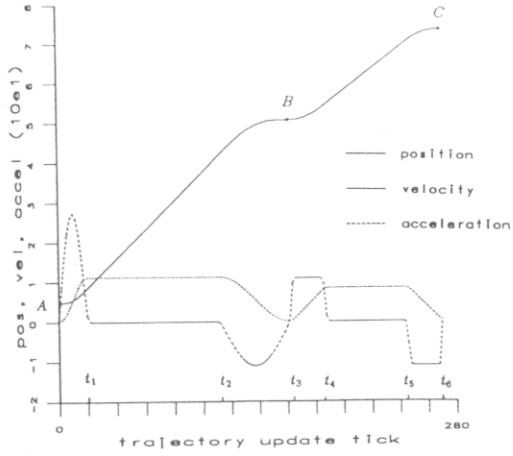


Fig. 8 The motion profile of actuator 3 in point-to-point motions

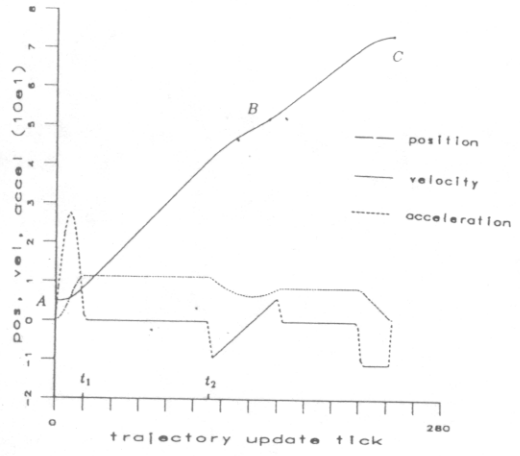


Fig. 9 The motion profile of actuator 3 with blended motion segments

for acceleration, slew motion, and deceleration of this second motion segment are 24, 59, and 24 ticks, respectively. Since the defined maximum acceleration and deceleration for each actuators are the same in our manipulator, when the same percentage of the maximum acceleration and deceleration is used for a motion segment, the acceleration time for the manipulator to reach the defined velocity should equal to the deceleration time for the manipulator to come to stop from slew motion. For a single actuator motion, the target position can be specified by either an actuator value or effective joint value. The value of 90 degrees of effective joint 3 is equivalent to the actuator value of 73.3 inches for the UPSarm. If the last statement in the program Listing 1 was replaced by the program statement `move @ 73.3`, the resulted motion would be the same as one shown in Figure 8.

Adding a statement `enable continuouspath` at the beginning of the robot program in Listing 1, two point-to-point motion segments shown in Fig. 8 will be blended as a continuous-path motion. The continuous-path without an intermediate stop is shown in Fig. 9. The transition trajectory of two motion segments is a cubic polynomial which is derived using actuator values and velocities at two transition positions where the robot arm ends the slew motion of the first segment and starts the slew motion of the second segment, respectively. The boundary conditions of the transition region are the positions and velocities at times $t_2$ and $t_4$ shown in Fig. 8. Using a cubic polynomial as a transition trajectory will result in continuous position and velocity for the blended trajectory. The second derivative of the third order polynomial is a linear function which can not guarantee the continuity of the acceleration of the trajectory as is shown in Fig. 9. If higher accuracy and smoother motion are desired, a fifth order polynomial may provide better performance with continuous acceleration at the boundaries of the transition region of two motion segments. But, more computational power will be demanded for higher order polynomials. This is a trade-off between the accuracy and computational speed. For our practical applications, the continuous-path based upon a cubic polynomial does provide a satisfactory performance. When blending two segments together, the planned deceleration time for the first segment and the acceleration time for the second segment are available. Whichever the larger of these two duration times will be selected as the transition time in our path planner. This is conservative, but the final trajectory will comply the system constraints. In this example, the deceleration time for the first segment is 50 ticks whereas the acceleration time for the second segment is 24 ticks. Hence, the transition time will use the larger value which is 50 ticks. Using this scheme to blend motion segments, the total motion time for blended continuous-path motion in Fig. 9 is reduced from 269 ticks to 245 ticks in comparing with the point-to-point motion shown in Fig. 8.

**7.2 Coordinated Motion.** The second example illustrates the motion along a prescribed trajectory or path tracking. The robot program listed in Listing 2 will move the TCP of the end-effector of the UPSarm along straight-lines.

LISTING 2

```
E = actuator(5.0, 5.0, 64.5, 0.0, 0.0,
0.0, 1.2, 90.0, 9.8, 9.8)
F = joint(0.0, 0.0, 72.3, 9.5, 7.6,
19.1, 0.0, 90.0, 90.0, 95.3)
G = trans(rxyz(-10.0, -10.0, 62.6),
vect(125, -2, 10.0))
move E
moves F
moves G
```

The trajectories of the TCP of the end-effector in the world coordinate system are shown in Fig. 10 where the solid curve is for non-continuous path and the dotted line for continuous-path. The continuous-path is achieved by adding the statement `enable continuouspath` at the top of the robot program in Listing 2. The initial position of the robot arm is at position D. The subsequent positions E, F, and G are defined in the program. The position E is defined by actuator values, F is defined by effective joint values, and G is specified by the transformation matrix of the end-effector. The first parameter of the transformation specification is three orientation angles about the X, Y, and Z axes, respectively; and the second parameter is the distance of the TCP from the origin $O_w$ of the world coordinate system $O_wX_wY_wZ_w$ shown in Fig. 2. The equivalent transformations of the end-effector for positions E and F obtained from direct and forward kinematics are trans(*rxyz*(0,0,77.6), vect (84.2, −14.1, 5)) and trans(*rxyz*(0, 0, 77.6), vect(110, −15.9, 0)), respectively. The intended straight-line motion information is passed to the path planner and trajectory generator by the move statement reserved-word "`moves`." The motion from the initial position D to E is a point-to-point motion. The motion segments EF and FG are two straight-lines as indicated by the move command "`moves`". If the continuous-path is not enabled, the trajectory of the TCP has sharp corners at the intersection of two straight-lines, which is typical for noncontinuous path motions. The continuous-path tends to rounding these corners as is shown in Fig. 10. In this example, the motion is continuously transited from the actuator point-to-point motion of segment DE to the prescribed straight-line segment EF. The continuous-path also blends two straight-line segments EF and FG. The blending trajectories for both TCP and orientation of the end-effector are third order polynomials.

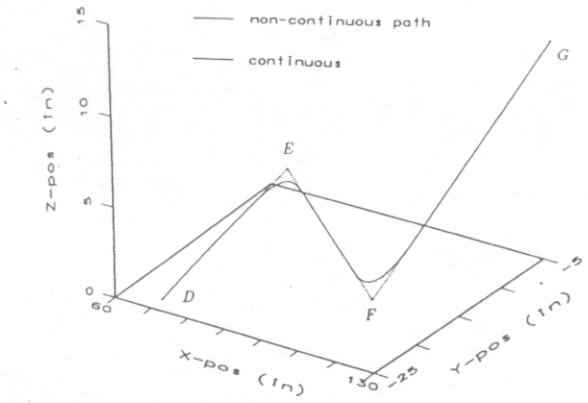


**Fig. 10 The TCP of the end-effector in point-to-point and continuous-path motions**

## 8 Conclusions

Three coordinate spaces: actuator space, effective joint space, and world Cartesian coordinate space, are introduced for coordinated manipulation of hybrid serial-and-parallel-driven redundant industrial manipulators. Based upon these three coordinate spaces, the concepts of direct, forward, inverse, and indirect kinematics are also introduced. A general-purpose path planner and trajectory generator are developed, which can be used for real-time control of $N$ degree-of-freedom electromechanical devices with hybrid serial-and-parallel-driven kinematic chains. This path planner and trajectory generator can provide point-to-point and continuous-path motions with the target information being specified in the above three coordinate spaces. Three Cartesian pathes: straight-line, circular curve, and spline curve, and combined Cartesian-joint motion have been implemented for path tracking. Each motion segment consists of three regions: acceleration, slew motion, and deceleration. The manipulator can be accelerated and decelerated in either square wave or sinusoidal motion profiles. Multiple motion segments can be blended by using cubic polynomials which will provide continuous positions and velocities along the trajectory. Software implementation of this path planner and trajectory generator in the C language are modular. With appropriate kinematics modules, this path planner and trajectory generator can be used for motion coordination of other hybrid serial-and-parallel-driven electromechanical devices.

A practical industrial application of hybrid serial-and-parallel-driven redundant kinematic chains has been demonstrated by a ten degree-of-freedom hybrid serial-and-parallel-driven UPSarm which is mainly designed for studying the feasibility of stacking packages inside a truck. The direct, forward, inverse, and indirect kinematic solutions have been derived for the UPSarm. These kinematic formulations involving iterative numerical solution techniques are used for real-time path planning, trajectory generation, and control of the manipulator. The simplified dynamic model, PID-based feedforward servo control scheme, and implementation of a real-time distributed computing system for manipulation of the UPSarm are presented in this paper. Motion examples programmed in our robot language demonstrate the practical manipulation of hybrid serial-and-parallel-driven kinematic chains.

Although this paper mainly addresses the issues related to the practical implementation of the UPSarm, the presented ideas and principles are general. They can be applied to manipulation of other hybrid serial-and-parallel-driven redundant robot manipulators as well.

## 9 Acknowledgment

This work described in this paper was done when the author was with Research and Development of United Parcel Service, Inc. I would like to thank A. Brandorff, C. C. Cai, R. Lancraft, J. J. Lee, J. Lecko, R. Penkar, S. Smith, N. Tan, and Z. Tumeh for numerous stimulating discussions and their contributions to the development of the UPSarm; and K. Nielsen, L. Newport, and J. Snyder for the support they provided. N. Tan and L. Lancraft read the manuscript and provided valuable comments.

## References

Bottema, O. and Roth, B., 1979, *Theoretical Kinematics*, North-Holland Publishing Comp., Amsterdam.

Brady, M., 1983, Trajectory Planing, *Robot Motion*, Brady et al., eds, MIT Press, Cambridge, pp. 221–243.

Cheng, H. H., 1993, "Real-Time Four-Dimensional Collision Detection for an Industrial Robot Manipulator" *Proc. of the Third National Conference on Applied Mechanisms and Robotics*, Cincinnati, Ohio, Nov. 8–10, Vol. 1, pp. AMR-93-016-1 to pp. AMR-93-016-13.

Cheng, H. H., and Gupta, K. C., 1991, "A Study of Robot Inverse Kinematics Based upon the Solution of Differential Equations," *J. Robotic Systems*, Vol. 8, Apr., pp. 159–175.

Cheng, H. H. and Gupta, K. C., 1993, "Vectorization of Robot Inverse Dynamics on a Pipelined Vector Processor," *IEEE Trans. on Robotics and Automation*, Vol. 9, No. 6, December, pp. 858–863.

Cheng, H. H. and Gupta, K. C., 1992, "An Efficient Manipulator Dynamics Formulation Based upon Newton-Euler Equations and the ZRP Method," *Flexible Mechanisms, Dynamics, and Analysis*, by G. Kinzel, eds., et al., ASME 22nd Biennial Mechanisms Conference, Scottsdale, AZ, Sept. 13–16, DE-Vol. 47, pp. 81–87.

Cheng, H. H., Lee, J. J., and Penkar, R., 1991, "Kinematics of the Prototype UPSarm," *Proc. of Second National Applied Mechanisms and Robotics*, Vol. II, Cincinnati, OH, Nov. 3–6, pp. IXC.2-1 to IXC.2-12.

Craig, J. J., 1989, *Introduction to Robotics, Mechanics and Control*, Addison-Wesley Publishing Company.

Craig, J. J., Hsu, P., and Sastry, S., 1986, "Adaptive Control of Mechanical Manipulators," *Proc. of IEEE Conf. on Robotics and Automation*, San Francisco, Apr., pp. 190–195.

Denavit, J. and Hartenberg, R. S., 1955, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," ASME JOURNAL OF APPLIED MECHANICS, Vol. 22, pp. 215–221.

Featherstone, R., 1983, "Position and Velocity Transformations between Robot End-effector Coordinates and Joint Angles," *Int. J. Robotics Res.*, 2(2), pp. 35–45.

Fichter, E. F. and McDowell, E. D., 1980, "A Novel Design for a Robot Arm," The International Computer Technology Conference, Vol. 1, Aug., pp. 250–256.

Hollerbach, J., and Sahar, G., 1983, "Wrist-Partitioned, Inverse Kinematic Accelerations and Manipulator Dynamics," *Int. J. Robotics Res.*, Vol. 2(4), pp. 61–76.

Hunt, K. H., 1983, "Structural Kinematics of In-Parallel-Actuated Robot Arm," ASME *Journal Mechanisms, Transmissions, and Automation in Design*, Vol. 105: pp. 705–712.

Kazerounian, S. M. K. and Nedungadi, A., 1987, "Redundancy Resolution of Robot Manipulators at the Acceleration Level," *Proc. 7th IFToMM World Congress on TMM*, Vol. 2, Sevilla, Spain, Sept. 17–22, pp. 1027–1211.

Lee, K. M., and Shah, D. K., 1980, "Kinematic Analysis of a Three Degrees of Freedom In-parallel Actuated Manipulator," *IEEE Conference*, pp. 345-350.

Liegeois, A., Fournier, A., and Aldon, M., 1980, "Model Reference Control of High Velocity Industrial Robots," *Proc. Joint Automatic Control Conference*, San Francisco, CA, pp. TP10-D.

Litvin, F. L., Zhang, Y., Castelli, V. P., and Innocenti, C., 1986, "Singularities, Configurations and Displacement Functions for Manipulators," *Int. J. Robotics Res.*, Vol. 5(2), pp. 66–74.

Luh, J. Y. S., Walker, M. W., and Paul, R. P., 1980a, "Resolved Acceleration Control of Mechanical Manipulators," *IEEE Trans. Automatic Control*, Vol. 25(3), pp. 468–474.

Luh, J. Y. S., Walker, M. W., and Paul, R. P., 1980b, "On-Line Computational Scheme for Mechanical Manipulators," ASME JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL, Vol. 102, pp. 69–76.

Oh, S. Y., Orin, D., and Bach, M., 1984, "Inverse Kinematic Solution for Kinematically Redundant Robot Manipulators," *J. Robotic Systems*, Vol. 1(1), pp. 235–249.

Paul, R. P., 1979, "Manipulator Cartesian Path Control," *IEEE Trans. Systems, Man, and Cybernetics*, SMC-9, pp. 702–711.

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

Pieper, D. L., 1969, The Kinematics of Manipulators under Computer Control, PhD Dissertation, Stanford University.

Slotine, J. J., and Li, W., 1987, On the Adaptive Control of Mechanical Manipulators, International J. of Robotics Research, Vol. 6, No. 3, pp. 49–59.

Sugimoto, K., 1987, "Kinematic and Dynamic Analysis of Parallel Manipulators by Means of Motor Algebra," ASME *Journal of Mechanisms, Transmissions, and Automation in Design* Vol. 109, pp. 3–7.

Taylor, R. H., 1979, "Planning and Execution of Straight-Line Manipulator Trajectories," *IBM J. Research and Development*, Vol. 23, pp. 424–436.

Tsai, L. W., and Morgan, A. D., 1985, "Solving the Kinematics of the Most General Six- and Five-degree-freedom Manipulators by Continuation Methods," ASME *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107(2), pp. 189–200.

Waldron, K. J., Raghavan, M., and Roth, B., 1989, "Kinematics of a Hybrid Series-Parallel Manipulation System," ASME JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL Vol. 111, June.

Wang, L.-C. T. and Chen, C. C., 1991, "A Combined Optimization Method for Solving the Inverse Kinematics Problem of the Mechanical Manipulator," *IEEE Trans. on Robotics and Automation*, Vol. 7, No. 4, pp. 489–499.

Whitcomb, L. L., Rizzi, A. A., and Koditschek, D. E., "Comparative Experiments with a New Adaptive Controller for Robot Arms," *Proc. of IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 9–11, Vol. 1, pp. 2–7.

Whitney, D. E., 1969, "Resolved Motion Rate Control of Manipulators and Human Prosthesis," *IEEE Trans. Man-Machine Systems*, MMC(10), June, pp. 47–53.

Zhang, C. and Song, S. M., 1991, "Forward Kinematics of a Class of Parallel (Steward) Platforms with Closed Form Solutions," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, Apr., pp. 2676–2681.