AD-A134 852 TUTORIAL WORKSHOP ON ROBOTICS AND ROBOT CONTROL(U) ARMY TANK-AUTOMOTIVE COMMAND WARREN MI 26 OCT 82 1/4

UNCLASSIFIED F/G 14/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

AD-A134852

(2)

TUTORIAL WORKSHOP ON

ROBOTICS AND ROBOT CONTROL

26 OCTOBER 1982

ORGANIZER: A.K. BEJCZY, JPL/CAL TECH

HOSTED BY: OAKLAND UNIVERSITY
ROCHESTER, MICHIGAN

SPONSORED BY:

US ARMY TANK-AUTOMOTIVE COMMAND, WARREN MICHIGAN
US ARMY MATERIEL SYSTEMS ANALYSIS ACTIVITY,
ABERDEEN PROVING GROUNDS, MARYLAND

SESSION I

SUMMARY

research and development

THE ~~OBJECTIVES~~ OF THIS TUTORIAL WORKSHOP ARE: (I) TO PROVIDE A GENERAL INTRODUCTION TO ROBOTICS AND ROBOT CONTROL PROBLEMS, (II) TO OUTLINE THE TECHNICAL CONTENT OF "ADVANCED ROBOT CONTROL," AND (III) TO PROVIDE A FORUM FOR A BRIEF DISCUSSION OF ~~R&D~~ TOPICS IN THE AREA OF APPLIED ROBOT CONTROL.

THE FIRST SESSION DEALS WITH ROBOT MECHANISMS, SENSORS AND CONTROL. KINEMATIC AND DYNAMIC MODELS OF ROBOTS ARE INTRODUCED TOGETHER WITH ILLUSTRATIONS OF EXISTING INDUSTRIAL ROBOTS. BOTH INTERNAL AND EXTERNAL STATE SENSORS ARE DISCUSSED TOGETHER WITH ALTERNATIVE FEEDBACK CONTROL SCHEMES.

THE SECOND SESSION IS DEVOTED TO THE MATHEMATICS AND COMPUTER PROGRAMMING OF ROBOT CONTROL. HOMOGENEOUS TRANSFORMATIONS, KINEMATIC EQUATIONS AND THEIR SOLUTION, MOTION TRAJECTORIES AND DIFFERENTIAL RELATIONSHIPS ARE DISCUSSED AS THEY RELATE TO THE CONTROL OF ROBOTS.

THE THIRD SESSION TREATS TOPICS IN THE AREA OF VISUAL SENSING FOR ROBOT CONTROL. THE MAJOR TOPICS ARE: A) SENSING AND PREPROCESSING, B) SEGMENTATION, C) RECOGNITION AND INTERPRETATION, AND D) REPRESENTATION OF SCENE DATA.

THIS PAGE LEFT BLANK INTENTIONALLY

ROBOT MECHANISMS, SENSORS AND CONTROL

A. K. BEJCZY

OVERVIEW

1. BASIC DEFINITIONS
2. ROBOT MECHANISMS

   $\left.\begin{array}{l}\text{KINEMATIC}\\ \text{DYNAMIC}\end{array}\right\}$ MODELS & REPRESENTATIONS

3. SENSORS

   INTERNAL STATE
   EXTERNAL STATE

4. CONTROL

   OPEN LOOP
   INTERNAL STATE FEEDBACK
   EXTERNAL STATE FEEDBACK (GUIDANCE)
   ADAPTIVE

5. SUMMARY PERSPECTIVES OF CONTROL ISSUES

BASIC DEFINITIONS

1. ROBOTS ARE GENERAL PURPOSE PROGRAMMABLE MACHINES
   a. DEFINED IN TERMS OF MOVEMENTS AND MOTION CAPABILITIES.
   b. OPERATE AUTOMATICALLY

2. TYPES OF ROBOTS
   a. LOCOMOTION ROBOTS
   b. MANIPULATOR ROBOTS

THIS WORKSHOP IS ABOUT MANIPULATOR ROBOTS

HISTORY OF ROBOTS

1. CURIOSITY ABOUT ANTHROPOMORPHIC MACHINES.

2. WORK IN HOSTILE ENVIRONMENT (TELEOPERATION)

   TOXIC OR "HOT" RADIATION

   UNDER SEA

   SPACE, ETC.

3. INDUSTRIAL ROBOTS

   ECONOMY, PRODUCTIVITY, QUALITY

   HUMANIZING FACTORY WORK

4. TODAY: MANY THOUSANDS OF INDUSTRIAL ROBOTS WORLD-WIDE, 30% PREDICTED YEARLY GROWTH RATE, BROAD RANGE OF APPLICATIONS.

SOME DATA ABOUT INDUSTRIAL ROBOTS (DECEMBER, 1980)

COUNTRIES	
JAPAN	10,000
U. S. A.	3,500
W. GERMANY	900
SWEDEN	600
ITALY	500
POLAND	350
NORWAY	200
ENGLAND	200
OTHER	1,600
TOTAL	17,850

APPLICATIONS

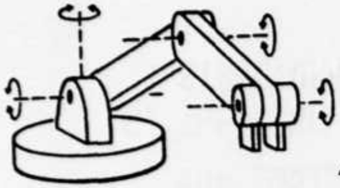
- DIE CASTING
- SPOT WELDING
- ARC WELDING
- PRESS WORK
- SPRAY PAINTING
- PALLETIZING
- HEAT TREATMENT
- METAL PARTS DEBURRING
- PLASTIC MOLDING
- ASSEMBLY
- ETC....

ROBOT MANIPULATOR MECHANISMS

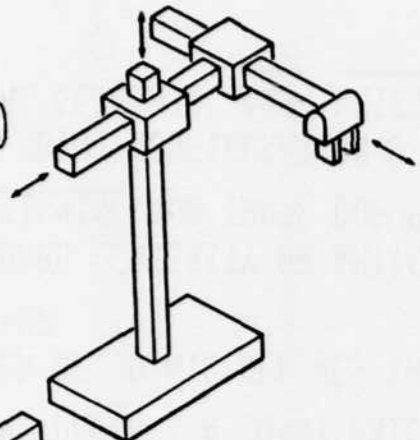
1. ROBOT MANIPULATORS ARE COMPOSED OF LINKS CONNECTED BY JOINTS INTO AN OPEN LOOP KINEMATIC CHAIN.
2. THE TYPICAL JOINTS ARE ROTARY OR LINEAR JOINTS.
3. EACH JOINT DEFINES A JOINT AXIS AROUND OR ALONG WHICH A LINK ROTATES OR SLIDES.
4. THE **NUMBER** OF JOINTS DEFINES THE DEGREE-OF-FREEDOM (DOF) OF A ROBOT MANIPULATOR
5. THE STANDARD DEXTERITY OR ARTICULATION REQUIREMENT IS SIX DOF: THREE DOF FOR POSITIONING AND THREE DOF FOR ORIENTATION.
6. THE LINK-JOINT CONFIGURATION OF A ROBOT MANIPULATOR CONTAINS THREE MAIN STRUCTURAL ELEMENTS: ARM, WRIST AND HAND ("END EFFECTOR").

TYPICAL ARRANGEMENTS OF ROBOT MANIPULATOR JOINTS

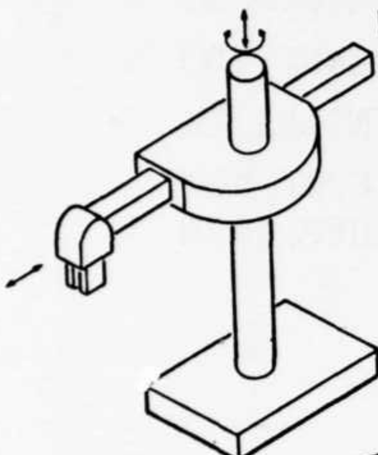
ARM



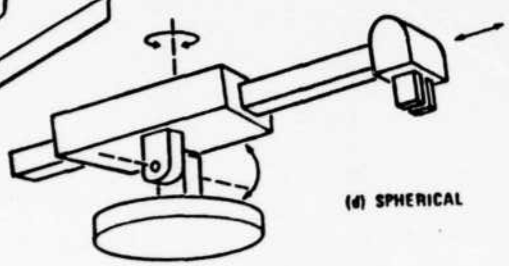
(a) JOINTED ARM



(b) CARTESIAN OR X-Y-Z ARM

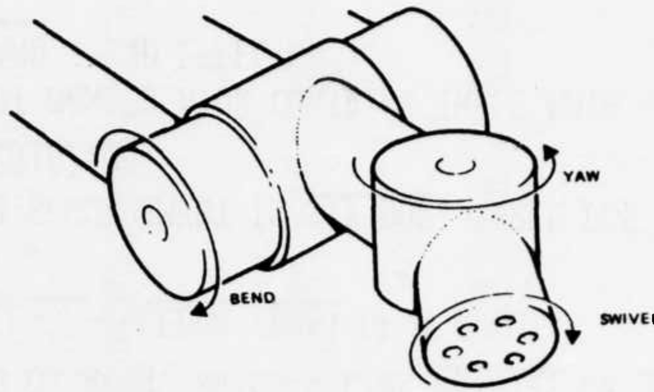


(c) CYLINDRICAL ARM

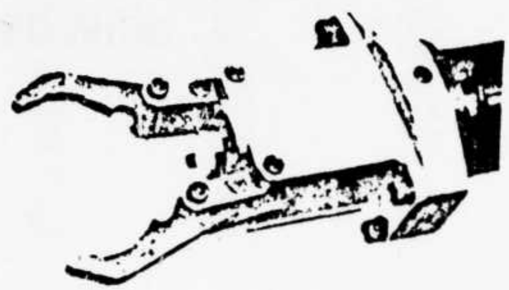


(d) SPHERICAL

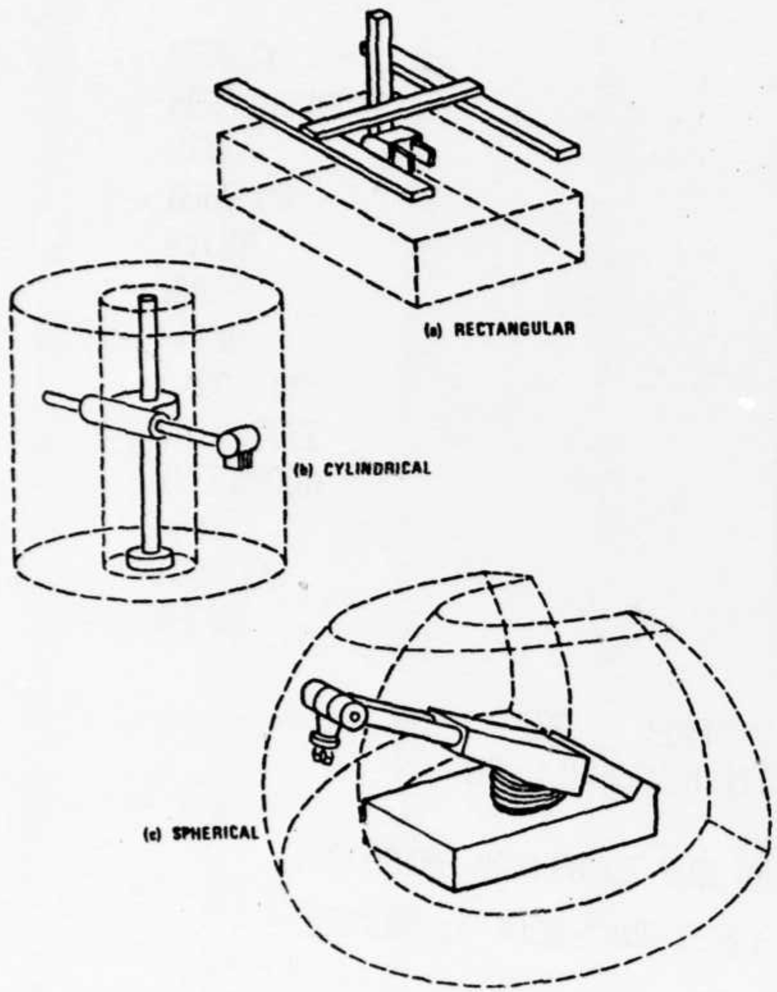
WRIST

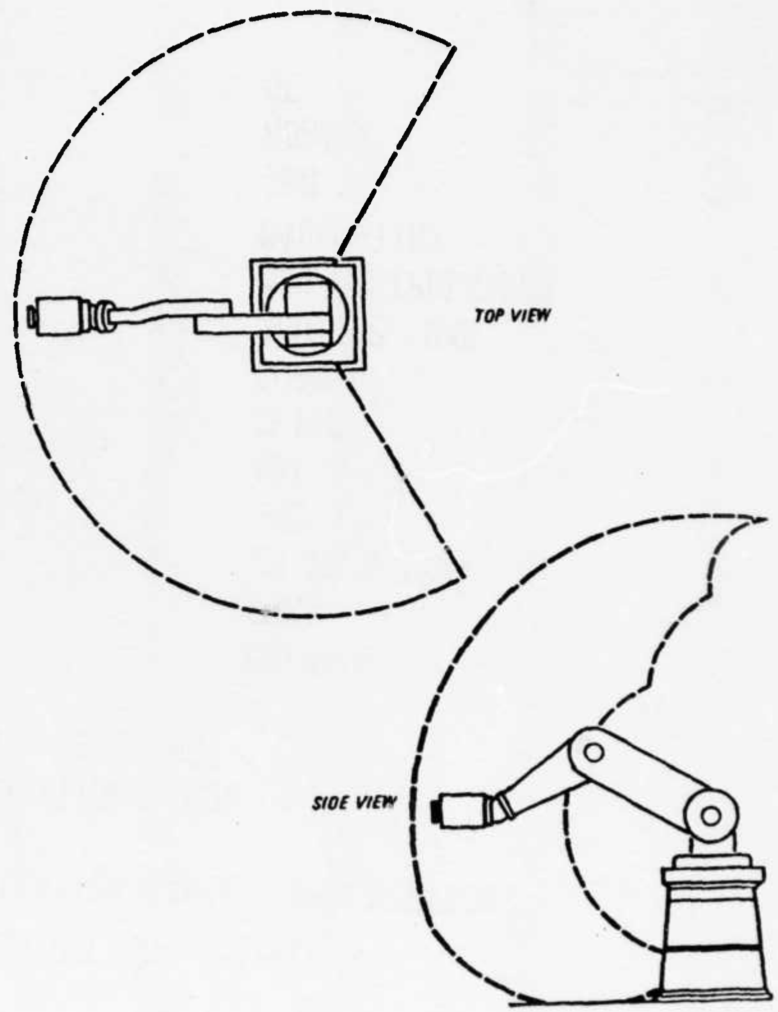


HAND



SOME TYPEICAL WORK VOLUME SHAPES





INDUSTRIAL ROBOT AND OTHER MANIPULATOR EXAMPLES

INCLUDING WORKSPACE AND APPLICATION CONCEPT DESCRIPTIONS

THE VIEWGRAPH ILLUSTRATIONS SHOW
THE FOLLOWING SYSTEMS

ASEA	UNIMATE
TRALLFA	PUMA
PRAB	CM T^3 & T^3R^3
THERMWOOD	GNO
AUTOPLACE	IRI
CYBOTECH	SEIKO
KUKA	SORMEL
MAKER	SHUTTLE RMS
PACER	M/S MANIPULATORS
NORDSON	MICRO-TINY
GCA	IBM
WESTINGHOUSE	HOBART
BENDIX	GE

ROBOT MANIPULATOR KINEMATIC REPRESENTATION AND COORDINATE FRAMES

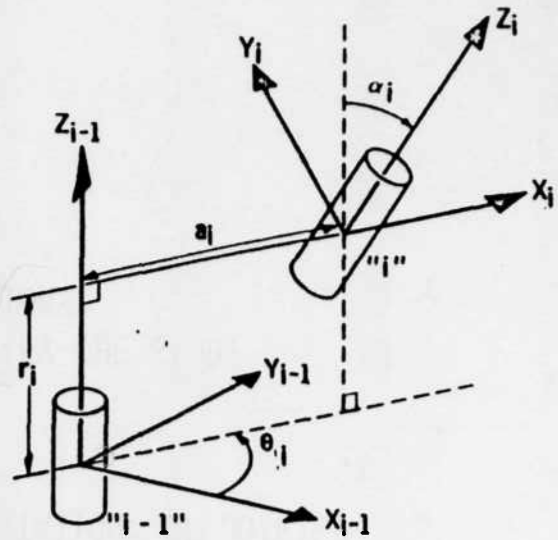
1. JOINT COORDINATES -- WORK SPACE COORDINATES -- HAND/TOOL COORDINATES

2. HOMOGENEOUS COORDINATES: LINEAR TRANSFORMATION - MATRIX OPERATIONS

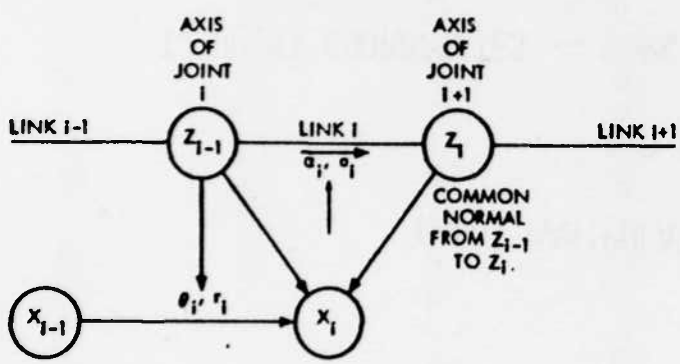
3. HARTENBERG - DENAVIT "FOUR PARAMETER PRESENTATION" OF JOINTED MECHANISMS

NOTE: MATRIX TRANSFORMATIONS GREATLY SIMPLIFY THE STUDY AND COMPUTATION OF COMPLEX ROBOT MOTIONS

COORDINATE FRAMES AND PARAMETERS FOR MANIPULATOR LINK-JOINT PAIRS IN THE HARTENBERG - DENAVIT REPRESENTATION



STANDARD PARAMETERS



a_i - LENGTH OF COMMON NORMAL BETWEEN Z_{i-1} AND Z_i

α_i - ANGLE BETWEEN Z_{i-1} AND Z_i POSITIVE IN THE RIGHT HAND SENSE ABOUT THE COMMON NORMAL X_i

DISPLACEMENT AND ROTATION MEASURES ORTHOGONAL TO THE AXIS OF MOTION (THESE ARE CONSTANTS)

θ_i - JOINT ANGLE, POSITIVE IN THE RIGHT HAND SENSE ABOUT Z_{i-1}

r_i - LENGTH ALONG Z_{i-1} FROM X_{i-1} TO X_i

DISPLACEMENT AND ROTATION MEASURES ALONG THE AXIS OF MOTION (ONE OF THE TWO IS VARIABLE)

THE BASIC 4 BY 4 HOMOGENEOUS JOINT SPACE TRANSFORMATION MATRIX

CONSIDER A SYSTEM OF N MECHANICAL LINKS (NUMBERED 1 THROUGH N), EACH CAPABLE OF LINEAR OR ROTARY MOTION RELATIVE TO THE ADJACENT LINKS. FOR REFERENCE, DEFINE LINK 0 TO BE FIXED TO A BASE (IN A RIGID TABLE, ON A VEHICLE, OR ON THE GROUND, FOR INSTANCE). DEFINE A JOINT TO BE THE INTERSECTION OF TWO ADJACENT LINKS; IN PARTICULAR, JOINT I IS THE INTERSECTION OF LINKS I-1 AND I, I = 1, . . ., N.

THEN (SEE PREVIOUS FIGURE) DEFINE THE FOLLOWING:

$\vec{Z}_I \equiv$ AXIS OF JOINT I+1 (SENSE ARBITRARY). IF THE JOINT IS ROTARY, THEN $\vec{Z}_1$ IS THE AXIS OF ROTATION; IF PRISMATIC (I.E., LINEAR), THEN $\vec{Z}_I$ IS IN THE DIRECTION OF LINEAR MOTION. DEFINING $\vec{Z}_N$ IN AN ARBITRARY MANNER, $\vec{Z}_I$ IS DEFINED FOR I = 0, . . ., N.

$\vec{X}_1 \equiv \vec{Z}_{I-1} \times \vec{Z}_I / \| \vec{Z}_{I-1} \times \vec{Z}_I \|$, THE COMMON NORMAL BETWEEN $\vec{Z}_{I-1}$ AND $\vec{Z}_I$, DIRECTED FROM THE FORMER TO THE LATTER. IF $\vec{Z}_{I-1} \times \vec{Z}_I = 0$, THEN $\vec{X}_I$ IS ARBITRARY SUBJECT ONLY TO $\vec{X}_I \cdot \vec{Z}_I = 0$. DEFINING $\vec{X}_0$ IN AN ARBITRARY MANNER, $\vec{X}_I$ IS DEFINED FOR I = 0, . . ., N.

$\vec{Y}_I \equiv \vec{Z}_I \times \vec{X}_I$, I = 0, . . . , N.

THUS WE HAVE DEFINED AN ORTHONORMAL COORDINATE SYSTEM $(\vec{X}_I, \vec{Y}_I, \vec{Z}_I)$, FIXED IN LINK I, I = 0, . . . N. THE FOLLOWING PARAMETERS DEFINE THE RELATIONSHIP BETWEEN SUCCESSIVE COORDINATE SYSTEMS.

$\theta_i \equiv$ angle from $\vec{x}_{i-1}$ to $\vec{x}_i$, measured positively counterclockwise about $\vec{z}_{i-1}$.

$r_i \equiv$ distance along $\vec{z}_{i-1}$ from $\vec{x}_{i-1}$ to $\vec{x}_i$.

$\alpha_i \equiv$ angle from $\vec{z}_{i-1}$ to $\vec{z}_i$, measured positively counterclockwise about $\vec{x}_i$.

$a_i \equiv$ distance along $\vec{x}_i$ from $\vec{z}_{i-1}$ to $\vec{z}_i$.

A vector $\vec{v}_i$ expressed in system i can be expressed in system i-1 as $\vec{v}_{i-1}$ using the transformation T^i_{i-1}, as follows:

$$\vec{v}_{i-1} = T^i_{i-1}\,\vec{v}_i$$

More specifically, $\vec{v}_i$ is first rotated about $\vec{x}_i$ by $-\alpha_i$ (aligning $\vec{z}_i$ and $\vec{z}_{i-1}$), then translated along $\vec{x}_i$ by a_i (bringing $\vec{z}_i$ and $\vec{z}_{i-1}$ into coincidence), then translated along $\vec{z}_i$ ($\equiv \vec{z}_{i-1}$) by r_i (giving $\vec{x}_i$ and $\vec{x}_{i-1}$ a common origin), and finally rotated about $\vec{z}_i$ by $-\theta_i$ (bringing the two systems into coincidence).

Thus,

$$T^i_{i-1} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CONSEQUENTLY, THE BASIC 4 x 4 HOMOGENEOUS JOINT SPACE TRANSFORMATION MATRIX IS:

$$T_{i-1}^{i} = \begin{bmatrix} c_{\theta_i} & -c_{\alpha_i} s_{\theta_i} & s_{\alpha_i} s_{\theta_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\alpha_i} c_{\theta_i} & -s_{\alpha_i} c_{\theta_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & r_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

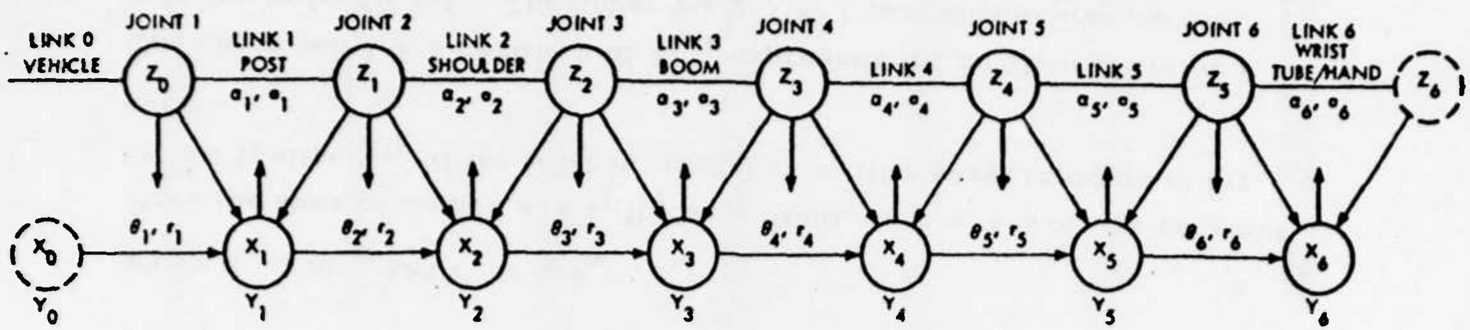
where $s_x \equiv \sin_x$ and $c_x \equiv \cos_x$.

Since the axes of motion are $\vec{z}_i$ (by definition), then a_i and α_i are constant for all i; either θ_i (if the joint is linear) or r_i (if rotary) is constant for each i.

The upper left 3 $\times$ 3 partition of T_{i-1}^{i} expresses the rotation of frame i relative to frame i-1. The upper right 3 $\times$ 1 partition expresses the position of the origin of frame i relative to i-1.

AN EXAMPLE, THE JPL-STANFORD ARM,

USING THE HARTENBERG - DENAVIT REPRESENTATION

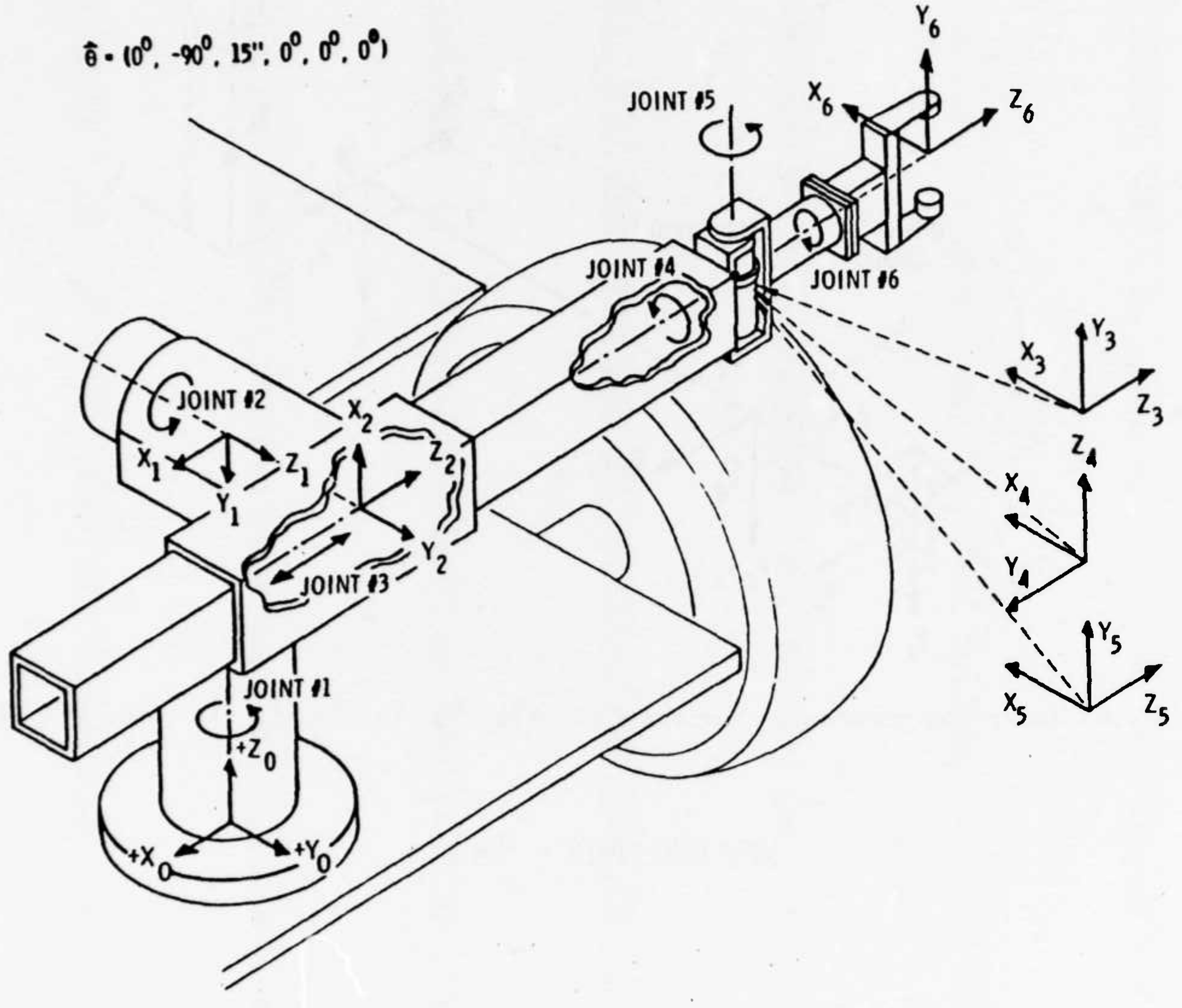


i	α_i (deg)	r_i (in.)	θ_i (deg)	a_i (in.)	Maximum radial dimension about z_i (in.)	Maximum linear extension along $-z_i$ (in.)
1	-90	14	[-175, 175]	0	2.625	0
2	90	6.375	[-175, 175]	0	2.75	12
3	0	[5.5, 44]	-90	0	1.25	$55-r_3$
4	-90	0	[-175, 175]	0	2.375	0
5	90	0	[-110, 110]	0		0
6	0	9.75	[-175, 175]	0		0
fingers	5 in. long, 1 3/16 in. wide, open to 2 1/4 in. each					

Note: The ranges of the six joint variables are indicated in brackets above.

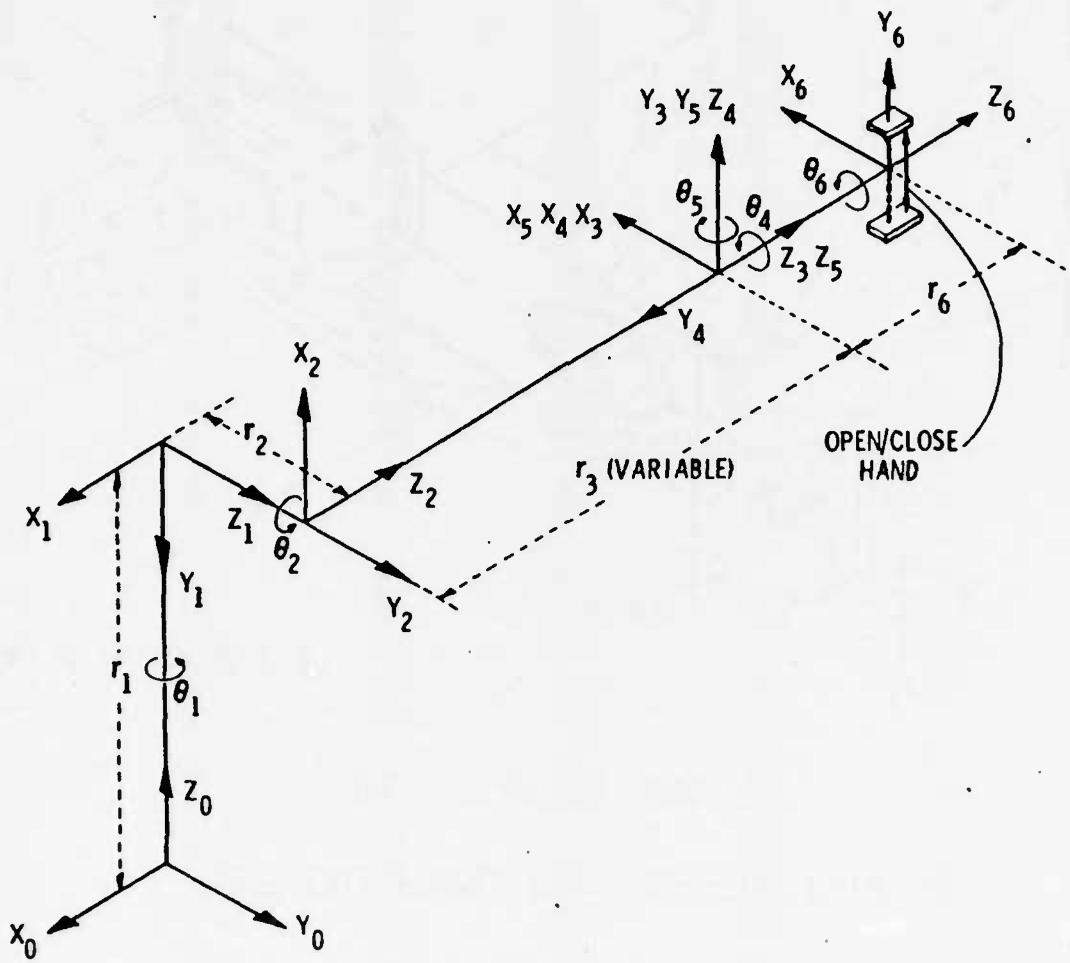
REFERENCE FRAMES FOR LINK-JOINT PAIRS OF

JPL - STANFORD ARM



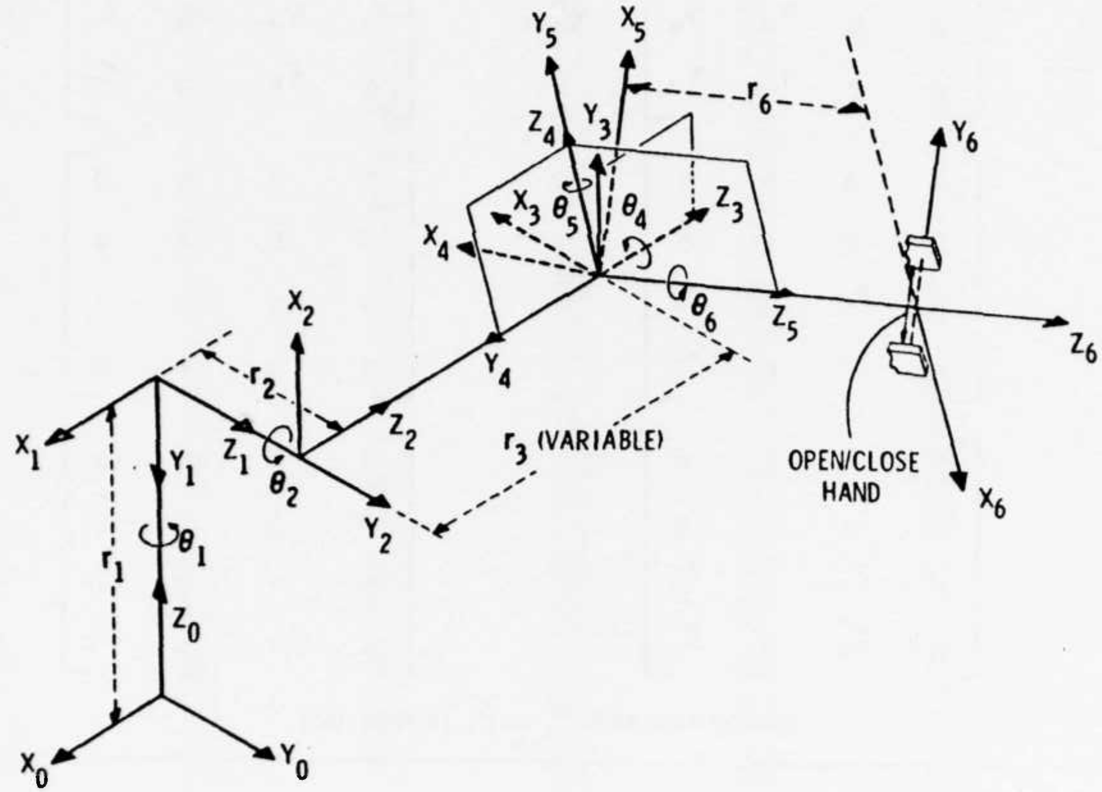
JPL - STANFORD ARM

Joint Variables θ_1, θ_2, r_3, θ_4, θ_5, θ_6, and Constant Orthogonal Displacements r_1, r_2, r_6



JPL - STANFORD ARM

Joint Variables θ_1, θ_2, r_3, θ_4, θ_5, θ_6, and Constant Orthogonal Displacements r_1, r_2, r_6



JPL—STANFORD ARM

Individual Link Transformation

$$T_0^1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & r_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_1^2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & r_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_3^4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

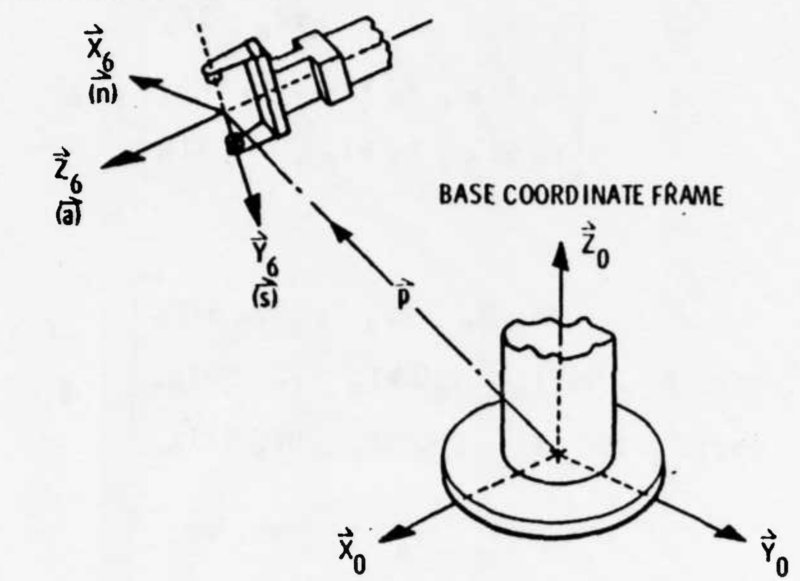
$$T_4^5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_5^6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & r_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The arm transformation T is given by

$$T \equiv T_0^6 = T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6.$$

JPL - STANFORD ARM

HAND POSITION AND ORIENTATION EXPRESSED IN BASE (OR WORLD) COORDINATES



$$\vec{\theta} \equiv (\theta_1, \theta_2, r_3, \theta_4, \theta_5, \theta_6).$$

Arm Transformation

$$T = T_0^6 = \begin{bmatrix} \vec{n} & \vec{s} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\vec{n} = \begin{bmatrix} c_{1256} s_4 + s_1 c_{456} - c_{16} s_{25} + c_{124} s_6 - s_{146} \\ s_{14} c_{256} - c_{1456} - s_{125} c_6 + s_{16} c_{24} + c_1 s_{46} \\ -s_{24} c_{56} - c_{26} s_5 - s_{26} c_4 \end{bmatrix}$$

$$\vec{s} = \begin{bmatrix} -c_{125} s_{46} - s_{16} c_{45} + c_1 s_{256} + c_{1246} - s_{14} c_6 \\ -s_{146} c_{25} + c_{145} s_6 + s_{1256} + s_1 c_{246} + c_{16} s_4 \\ s_{246} c_5 + c_2 s_{56} - s_2 c_{46} \end{bmatrix}$$

$$\vec{a} = \begin{bmatrix} c_{12} s_{45} + s_{15} c_4 + c_{15} s_2 \\ s_{145} c_2 - c_{14} s_5 + s_{12} c_5 \\ c_{25} - s_{245} \end{bmatrix}$$

$$\vec{p} = \begin{bmatrix} r_3 c_1 s_2 - r_2 s_1 \\ r_3 s_1 s_2 + r_2 c_1 \\ r_3 c_2 + r_1 \end{bmatrix} + r_6 \hat{a}$$

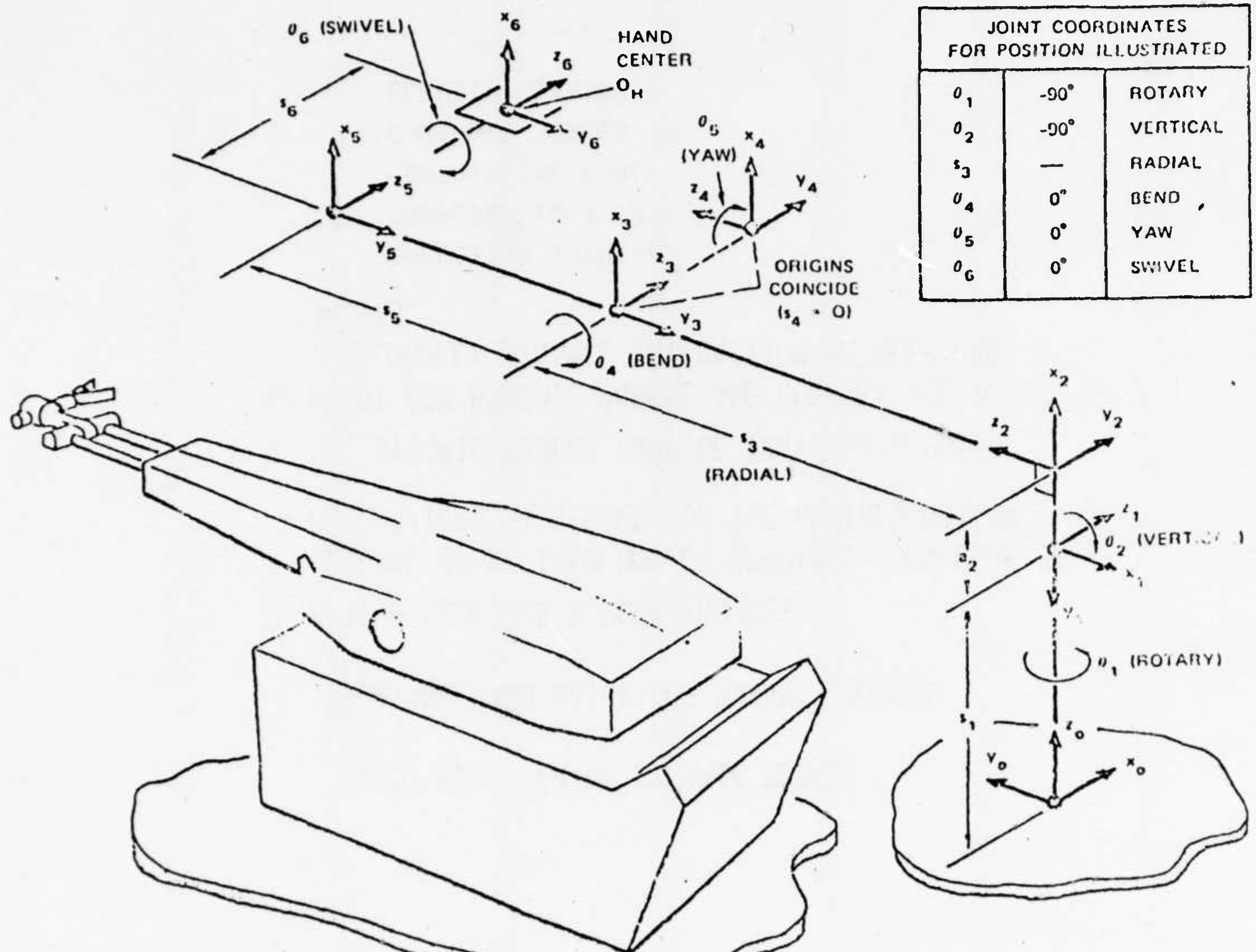
NOTE: $s_i \equiv \sin\theta_i \quad c_i \equiv \cos\theta_i$

$s_{ij} \equiv \sin\theta_i \sin\theta_j$

$c_{ijk} = \cos\theta_i \cos\theta_j \cos\theta_k$, etc.

ANOTHER EXAMPLE FOR H-D MATRIX REPRESENTATION

(UNIMATE ROBOT ARM)



JOINT COORDINATES FOR POSITION ILLUSTRATED		
θ_1	-90°	ROTARY
θ_2	-90°	VERTICAL
s_3	—	RADIAL
θ_4	0°	BEND
θ_5	0°	YAW
θ_6	0°	SWIVEL

ROBOT MANIPULATOR DYNAMIC MODELS

IMPORTANCE AND NATURE OF DYNAMIC MODELS

1. MOTION REQUIRES FORCES/TORQUES
2. CONTROL OF DESIRED MOTION REQUIRES KNOWLEDGE OF FORCES/TORQUES NEEDED FOR THE DESIRED MOTION
3. THE DYNAMIC MODELS PROVIDE THAT KNOWLEDGE
4. ROBOT ARM DYNAMIC MODELS ARE COMPLEX DUE TO THE COMPLEX DYNAMIC INTERACTION BETWEEN ARM JOINTS:

    EFFECTIVE INERTIA
    COUPLING INERTIA
    CENTRIPETAL FORCES
    CORIOLIS FORCES
    GRAVITY LOADING

ROBOT MANIPULATOR DYNAMICS AND CONTROL

STATE OF THE ART

1. EXISTING PRACTICE FOR INDUSTRY ROBOTS:
   - RIGID "TEACHING", POINT-TO-POINT PROGRAMMING,
   - POSITIONING SERVOING
2. EXISTING R & D APPROACHES:
   - NUMERICAL TECHNIQUES
   - SYMBOLIC TECHNIQUES
3. NUMERIC TECHNIQUES
   - NUMBER GENERATORS
   - SPEED OF COMPUTATION
4. SYMBOLIC TECHNIQUES
   - STATE EQUATION GENERATORS
   - HIGHER LEVEL AND COMPREHENSIVE INSIGHT INTO THE CONTROL PROBLEM
   - NEED FOR REDUCTIONS AND SIMPLIFICATIONS

ROBOT MANIPULATOR DYNAMIC EQUATIONS USING HOMOGENEOUS COORDINATES AND LAGRANGIAN MECHANICS FOR RIGID BODY

1. $$F_i = \sum_{j=1}^{n} D_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\dot{q}_j\dot{q}_k + D_i$$

where

2. $$D_{ij} = \sum_{p=\max i,j}^{n} \mathrm{Trace}\left(U_{pj} J_p U_{pi}^T\right)$$

3. $$D_{ijk} = \sum_{p=\max i,j,k}^{n} \mathrm{Trace}\left(U_{pjk} J_p U_{pi}^T\right)$$

4. $$D_i = \sum_{p=i}^{n} -m_p \bar{g}\, U_{pi} \bar{r}_p$$

F_i	:	force or torque acting at joint "i"
$q_j, \dot{q}_j, \ddot{q}_j$	:	position, velocity and acceleration of joint "j"
m_p	:	mass of body (link) "p"
$\bar{r}_p$	:	mass center vector of body (link) "p" in the coordinate frame fixed in the same body, given as a 4 x 1 vector
$\bar{g}$	:	acceleration of gravity, given as a 1 x 4 vector
J_p	:	4 x 4 pseudo-inertia matrix for body (link) "p"

$U_{pj} = \dfrac{\partial T_0^p}{\partial q_j}$: 4 x 4 matrix

$U_{pjk} = \dfrac{\partial^2 T_0^p}{\partial q_j \partial q_k}$: 4 x 4 matrix

with $T_0^p = T_0^1 T_1^2 \ldots T_{p-1}^p$, $p \leq n$, concatenation of 4 x 4 joint coordinate transformation matrices.

The quantities defined by Eqs. (2-4) can be called "dynamic projection functions"; they are functions of the joint variables q_j. In particular, D_{ii} represents the effective inertia at joint "i"; D_{ij} represents the coupling inertia between joints "i" and "j"; D_{ijj} represents the centripetal force at joint "i" due to velocity at joint "j"; D_{ijk} represents the Coriolis force at joint "i" due to velocities at joints "j" and "k"; D_i represents the gravity loading at joint "i".

THE 4 x 4 PSEUDO INERTIA MATRIX

$$J_j = m_j \begin{bmatrix} \frac{1}{2}\left(-k_{j11}^2 + k_{j22}^2 + k_{j33}^2\right) & k_{j12}^2 & k_{j13}^2 & \bar{x}_j \\ k_{j12}^2 & \frac{1}{2}\left(k_{j11}^2 - k_{j22}^2 + k_{j33}^2\right) & k_{j23}^2 & \bar{y}_j \\ k_{j13}^2 & k_{j23}^2 & \frac{1}{2}\left(k_{j11}^2 + k_{j22}^2 - k_{j33}^2\right) & \bar{z}_j \\ \bar{x}_j & \bar{y}_j & \bar{z}_j & 1 \end{bmatrix}$$

where

$\bar{x}_j$, $\bar{y}_j$, $\bar{z}_j$ are components of $\bar{r}_j$ mass center vector

k_{jip} = **radius of gyration "ip" (i, p = 1, 2, 3) of body (link) "j" about the origin of the coordinate frame fixed in the same body (link).**

AN EXAMPLE: DYNAMIC EQUATIONS FOR THE JPL - STANFORD ARM

$$
\begin{aligned}
& D_{11}\ddot{\theta}_1 + D_{12}\ddot{\theta}_2 + D_{13}\ddot{r}_3 + D_{14}\ddot{\theta}_4 + D_{15}\ddot{\theta}_5 + D_{16}\ddot{\theta}_6 \\
& + D_{111}\dot{\theta}_1^2 + D_{122}\dot{\theta}_2^2 + D_{133}\dot{r}_3^2 + D_{144}\dot{\theta}_4^2 + D_{155}\dot{\theta}_5^2 + D_{166}\dot{\theta}_6^2 \\
& + D_{112}\dot{\theta}_1\dot{\theta}_2 + D_{113}\dot{\theta}_1\dot{r}_3 + D_{114}\dot{\theta}_1\dot{\theta}_4 + D_{115}\dot{\theta}_1\dot{\theta}_5 + D_{116}\dot{\theta}_1\dot{\theta}_6 \\
& + D_{123}\dot{\theta}_2\dot{r}_3 + D_{124}\dot{\theta}_2\dot{\theta}_4 + D_{125}\dot{\theta}_2\dot{\theta}_5 + D_{126}\dot{\theta}_2\dot{\theta}_6 \\
& + D_{134}\dot{r}_3\dot{\theta}_4 + D_{135}\dot{r}_3\dot{\theta}_5 + D_{136}\dot{r}_3\dot{\theta}_6 \\
& + D_{145}\dot{\theta}_4\dot{\theta}_5 + D_{146}\dot{\theta}_4\dot{\theta}_6 + D_{156}\dot{\theta}_5\dot{\theta}_6 + D_1 = T_1
\end{aligned}
$$

$$
\begin{aligned}
& D_{12}\ddot{\theta}_1 + D_{22}\ddot{\theta}_2 + D_{23}\ddot{r}_3 + D_{24}\ddot{\theta}_4 + D_{25}\ddot{\theta}_5 + D_{26}\ddot{\theta}_6 \\
& + D_{211}\dot{\theta}_1^2 + D_{222}\dot{\theta}_2^2 + D_{233}\dot{r}_3^2 + D_{244}\dot{\theta}_4^2 + D_{255}\dot{\theta}_5^2 + D_{266}\dot{\theta}_6^2 \\
& + D_{212}\dot{\theta}_1\dot{\theta}_2 + D_{213}\dot{\theta}_1\dot{r}_3 + D_{214}\dot{\theta}_1\dot{\theta}_4 + D_{215}\dot{\theta}_1\dot{\theta}_5 + D_{216}\dot{\theta}_1\dot{\theta}_6 \\
& + D_{223}\dot{\theta}_2\dot{r}_3 + D_{224}\dot{\theta}_2\dot{\theta}_4 + D_{225}\dot{\theta}_2\dot{\theta}_5 + D_{226}\dot{\theta}_2\dot{\theta}_6 \\
& + D_{234}\dot{r}_3\dot{\theta}_4 + D_{235}\dot{r}_3\dot{\theta}_5 + D_{236}\dot{r}_3\dot{\theta}_6 \\
& + D_{245}\dot{\theta}_4\dot{\theta}_5 + D_{246}\dot{\theta}_4\dot{\theta}_6 + D_{256}\dot{\theta}_5\dot{\theta}_6 + D_2 = T_2
\end{aligned}
$$

EXAMPLE CONTINUED (1)

$$D_{13}\ddot{\theta}_1 + D_{23}\ddot{\theta}_2 + D_{33}\ddot{r}_3 + D_{34}\ddot{\theta}_4 + D_{35}\ddot{\theta}_5 + D_{36}\ddot{\theta}_6$$
$$+ D_{311}\dot{\theta}_1^2 + D_{322}\dot{\theta}_2^2 + D_{333}\dot{r}_3^2 + D_{344}\dot{\theta}_4^2 + D_{355}\dot{\theta}_5^2 + D_{366}\dot{\theta}_6^2$$
$$+ D_{312}\dot{\theta}_1\dot{\theta}_2 + D_{313}\dot{\theta}_1\dot{r}_3 + D_{314}\dot{\theta}_1\dot{\theta}_4 + D_{315}\dot{\theta}_1\dot{\theta}_5 + D_{316}\dot{\theta}_1\dot{\theta}_6$$
$$+ D_{323}\dot{\theta}_2\dot{r}_3 + D_{324}\dot{\theta}_2\dot{\theta}_4 + D_{325}\dot{\theta}_2\dot{\theta}_5 + D_{326}\dot{\theta}_2\dot{\theta}_6$$
$$+ D_{334}\dot{r}_3\dot{\theta}_4 + D_{335}\dot{r}_3\dot{\theta}_5 + D_{336}\dot{r}_3\dot{\theta}_6$$
$$+ D_{345}\dot{\theta}_4\dot{\theta}_5 + D_{346}\dot{\theta}_4\dot{\theta}_6 + D_{356}\dot{\theta}_5\dot{\theta}_6 + D_3 = F_3$$

$$D_{14}\ddot{\theta}_1 + D_{24}\ddot{\theta}_2 + D_{34}\ddot{r}_3 + D_{44}\ddot{\theta}_4 + D_{45}\ddot{\theta}_5 + D_{46}\ddot{\theta}_6$$
$$+ D_{411}\dot{\theta}_1^2 + D_{422}\dot{\theta}_2^2 + D_{433}\dot{r}_3^2 + D_{444}\dot{\theta}_4^2 + D_{455}\dot{\theta}_5^2 + D_{466}\dot{\theta}_6^2$$
$$+ D_{412}\dot{\theta}_1\dot{\theta}_2 + D_{413}\dot{\theta}_1\dot{r}_3 + D_{414}\dot{\theta}_1\dot{\theta}_4 + D_{415}\dot{\theta}_1\dot{\theta}_5 + D_{416}\dot{\theta}_1\dot{\theta}_6$$
$$+ D_{423}\dot{\theta}_2\dot{r}_3 + D_{424}\dot{\theta}_2\dot{\theta}_4 + D_{425}\dot{\theta}_2\dot{\theta}_5 + D_{426}\dot{\theta}_2\dot{\theta}_6$$
$$+ D_{434}\dot{r}_3\dot{\theta}_4 + D_{435}\dot{r}_3\dot{\theta}_5 + D_{436}\dot{r}_3\dot{\theta}_6$$
$$+ D_{445}\dot{\theta}_4\dot{\theta}_5 + D_{446}\dot{\theta}_4\dot{\theta}_6 + D_{456}\dot{\theta}_5\dot{\theta}_6 + D_4 = T_4$$

EXAMPLE CONTINUED (2)

$$
\begin{aligned}
&D_{15}\ddot{\theta}_1 + D_{25}\ddot{\theta}_2 + D_{35}\ddot{r}_3 + D_{45}\ddot{\theta}_4 + D_{55}\ddot{\theta}_5 + D_{56}\ddot{\theta}_6 \\
&+ D_{511}\dot{\theta}_1^2 + D_{522}\dot{\theta}_2^2 + D_{533}\dot{r}_3^2 + D_{544}\dot{\theta}_4^2 + D_{555}\dot{\theta}_5^2 + D_{566}\dot{\theta}_6^2 \\
&+ D_{512}\dot{\theta}_1\dot{\theta}_2 + D_{513}\dot{\theta}_1\dot{r}_3 + D_{514}\dot{\theta}_1\dot{\theta}_4 + D_{515}\dot{\theta}_1\dot{\theta}_5 + D_{516}\dot{\theta}_1\dot{\theta}_6 \\
&+ D_{523}\dot{\theta}_2\dot{r}_3 + D_{524}\dot{\theta}_2\dot{\theta}_4 + D_{525}\dot{\theta}_2\dot{\theta}_5 + D_{526}\dot{\theta}_2\dot{\theta}_6 \\
&+ D_{534}\dot{r}_3\dot{\theta}_4 + D_{535}\dot{r}_3\dot{\theta}_5 + D_{536}\dot{r}_3\dot{\theta}_6 \\
&+ D_{545}\dot{\theta}_4\dot{\theta}_5 + D_{546}\dot{\theta}_4\dot{\theta}_6 + D_{556}\dot{\theta}_5\dot{\theta}_6 + D_5 = T_5
\end{aligned}
$$

$$
\begin{aligned}
&D_{16}\ddot{\theta}_1 + D_{26}\ddot{\theta}_2 + D_{36}\ddot{\theta}_3 + D_{46}\ddot{\theta}_4 + D_{56}\ddot{\theta}_5 + D_{66}\ddot{\theta}_6 \\
&+ D_{611}\dot{\theta}_1^2 + D_{622}\dot{\theta}_2^2 + D_{633}\dot{r}_3^2 + D_{644}\dot{\theta}_4^2 + D_{655}\dot{\theta}_5^2 + D_{666}\dot{\theta}_6^2 \\
&+ D_{612}\dot{\theta}_1\dot{\theta}_2 + D_{613}\dot{\theta}_1\dot{r}_3 + D_{614}\dot{\theta}_1\dot{\theta}_4 + D_{615}\dot{\theta}_1\dot{\theta}_5 + D_{616}\dot{\theta}_1\dot{\theta}_6 \\
&+ D_{623}\dot{\theta}_2\dot{r}_3 + D_{624}\dot{\theta}_2\dot{\theta}_4 + D_{625}\dot{\theta}_2\dot{\theta}_5 + D_{626}\dot{\theta}_2\dot{\theta}_6 \\
&+ D_{634}\dot{r}_3\dot{\theta}_4 + D_{635}\dot{r}_3\dot{\theta}_5 + D_{636}\dot{r}_3\dot{\theta}_6 \\
&+ D_{645}\dot{\theta}_4\dot{\theta}_5 + D_{646}\dot{\theta}_4\dot{\theta}_6 + D_{656}\dot{\theta}_5\dot{\theta}_6 + D_6 = T_6
\end{aligned}
$$

A "DYNAMIC PROJECTION FUNCTION"
FOR THE JPL - STANFORD ARM

EFFECTIVE INERTIA AT JOINT #1.

$$
\begin{aligned}
D_{11} = {} & m_1 k_{122}^2 \\
& + m_2 \left[k_{211}^2 s^2\theta_2 + k_{233}^2 c^2\theta_2 + r_2(2\bar{y}_2 + r_2) \right] \\
& + m_3 \left[k_{322}^2 s^2\theta_2 + k_{333}^2 c^2\theta_2 + r_3(2\bar{z}_3 + r_3) s^2\theta_2 + r_2^2 \right] \\
& + m_4 \Big\{ \tfrac{1}{2} k_{411}^2 \left[s^2\theta_2 (2s^2\theta_4 - 1) + s^2\theta_4 \right] + \tfrac{1}{2} k_{422}^2 (1 + c^2\theta_2 + s^2\theta_4) \\
& \quad + \tfrac{1}{2} k_{433}^2 \left[s^2\theta_2 (1 - 2s^2\theta_4) - s^2\theta_4 \right] + r_3^2 s^2\theta_2 + r_2^2 - 2\bar{y}_4 r_3 s^2\theta_2 + 2\bar{z}_4 (r_2 s\theta_4 + r_3 s\theta_2 c\theta_2 c\theta_4) \Big\} \\
& + m_5 \Big\{ \tfrac{1}{2} (-k_{511}^2 + k_{522}^2 + k_{533}^2) \left[(s\theta_2 s\theta_5 - c\theta_2 s\theta_4 c\theta_5)^2 + c^2\theta_4 c^2\theta_5 \right] \\
& \quad + \tfrac{1}{2} (k_{511}^2 - k_{522}^2 + k_{533}^2)(s^2\theta_4 + c^2\theta_2 c^2\theta_4) \\
& \quad + \tfrac{1}{2} (k_{511}^2 + k_{522}^2 - k_{533}^2) \left[(s\theta_2 c\theta_5 + c\theta_2 s\theta_4 s\theta_5)^2 + c^2\theta_4 s^2\theta_5 \right] + r_3^2 s^2\theta_2 + r_2^2 \\
& \quad + 2\bar{z}_5 \left[r_3 (s^2\theta_2 c\theta_5 + s\theta_2 s\theta_4 c\theta_4 s\theta_5) - r_2 c\theta_4 s\theta_5 \right] \Big\} \\
& + m_6 \Big\{ \tfrac{1}{2} (-k_{611}^2 + k_{622}^2 + k_{633}^2) \left[(s\theta_2 s\theta_5 c\theta_6 - c\theta_2 s\theta_4 c\theta_5 c\theta_6 - c\theta_2 c\theta_4 s\theta_6)^2 + (c\theta_4 c\theta_5 c\theta_6 - s\theta_4 s\theta_6)^2 \right] \\
& \quad + \tfrac{1}{2} (k_{611}^2 - k_{622}^2 + k_{633}^2) \left[(c\theta_2 s\theta_4 c\theta_5 s\theta_6 - s\theta_2 s\theta_5 s\theta_6 - c\theta_2 c\theta_4 c\theta_6)^2 + (c\theta_4 c\theta_5 s\theta_6 + s\theta_4 c\theta_6)^2 \right] \\
& \quad + \tfrac{1}{2} (k_{611}^2 + k_{622}^2 - k_{633}^2) \left[(c\theta_2 s\theta_4 s\theta_5 + s\theta_2 c\theta_5)^2 + c^2\theta_4 s^2\theta_5 \right] \\
& \quad + \left[r_6 c\theta_2 s\theta_4 s\theta_5 + (r_6 c\theta_5 + r_3) s\theta_2 \right]^2 + (r_6 c\theta_4 s\theta_5 - r_2)^2 \\
& \quad + 2\bar{z}_6 \Big[r_6 (s^2\theta_2 c^2\theta_5 + c^2\theta_4 s^2\theta_5 + c^2\theta_2 s^2\theta_4 s^2\theta_5 + 2s\theta_2 c\theta_2 s\theta_4 s\theta_5 c\theta_5) \\
& \qquad + r_3 (s\theta_2 c\theta_2 s\theta_4 s\theta_5 + s^2\theta_2 c\theta_5) - r_2 c\theta_4 s\theta_5 \Big] \Big\}
\end{aligned}
$$

A "DYNAMIC PROJECTION FUNCTION" FOR THE JPL-STANFORD ARM

EFFECTIVE INERTIA AT JOINT #2

$$
\begin{aligned}
D_{22} = {} & m_2 k_{222}^2 \\
& + m_3 \left[k_{311}^2 + r_3(2\bar{z}_3 + r_3)\right] \\
& + m_4 \left[k_{411}^2 c^2\theta_4 + k_{433}^2 s^2\theta_4 + r_3(r_3 - 2\bar{y}_4)\right] \\
& + m_5 \left[k_{511}^2 c^2\theta_4 c^2\theta_5 + k_{522}^2 s^2\theta_4 + k_{533}^2 c^2\theta_4 s^2\theta_5 + r_3(r_3 + 2\bar{z}_5 c\theta_5)\right] \\
& + m_6 \Big\{ \tfrac{1}{2} k_{611}^2 \big[(s^2\theta_6 - c^2\theta_6)(s^2\theta_4 c^2\theta_5 + s^2\theta_5 - c^2\theta_4) \\
& \qquad + c\theta_5(c\theta_5 - 4s\theta_4 c\theta_4 s\theta_6 c\theta_6) + s^2\theta_4 s^2\theta_5\big] \\
& \quad + \tfrac{1}{2} k_{622}^2 \big[(c^2\theta_6 - s^2\theta_6)(s^2\theta_4 c^2\theta_5 + s^2\theta_5 - c^2\theta_4) \\
& \qquad + c\theta_5(c\theta_5 + 4s\theta_4 c\theta_4 s\theta_6 c\theta_6) + s^2\theta_4 s^2\theta_5\big] \\
& \quad + k_{633}^2 c^2\theta_4 s^2\theta_5 + \left[r_3(2r_6 c\theta_5 + r_3) + r_6^2(s^2\theta_4 s^2\theta_5 + c^2\theta_5)\right] \\
& \quad + 2\bar{z}_6 \left[(r_3 + r_6 c\theta_5)c\theta_5 + r_6 s^2\theta_4 s^2\theta_5\right] \Big\}
\end{aligned}
$$

AN EXAMPLE OF SIMPLIFIED EFFECTIVE INERTIA FUNCTIONS FOR THE JPL-STANFORD ARM

$$\boxed{D_{11} = b_1 + \left[b_2 + (b_3 + \bar{b}_5 c\theta_5)r_3 + \bar{b}_4 r_3^2\right] s^2\theta_2}$$

$$\boxed{D_{22} = b_{13} + (b_3 + \bar{b}_5 c\theta_5)r_3 + \bar{b}_4 r_3^2}$$

where b_1, b_2, b_3, $\bar{b}_4$, $\bar{b}_5$, and b_{13} are constants. The bar over $\bar{b}_4$ and $\bar{b}_5$ signifies that these two constants depend on the load held in the mechanical hand.

The accuracy of these terms is between 92-98%.

SIMPLIFIED EFFECTIVE INERTIA FUNCTIONS

FOR THE REMAINING JOINTS OF THE JPL-STANFORD ARM

$$\boxed{D_{44} = b_{14} + \bar{b}_{15} s^2 \theta_5}$$

$$\boxed{D_{55} = b_{17} + \bar{b}_{18}}$$

$$\boxed{D_{66} = b_{19} + \bar{b}_{20}}$$

where b_{14}, $\bar{b}_{15}$, b_{17}, $\bar{b}_{18}$, b_{19}, and $\bar{b}_{20}$ are constants, and the bar over the constants signifies that their value depends on the load held in the mechanical hand. The accuracy of the simplified expressions given by these terms is between 95-99%.

EXACT DYNAMIC STATE EQUATIONS FOR

THE FIRST THREE JOINTS OF THE JPL-STANFORD ARM

$$\left[181 + 371\, s^2\theta_2 + 15\, c^2\theta_2 + 0.39\, r_3 (r_3 - 51)\, s^2\theta_2\right]\ddot{\theta}_1$$

$$+ \left[(64 - 2.47 r_3)\, c\theta_2\right]\ddot{\theta}_2 + \left[-2.47\, s\ _2\right]\ddot{r}_3$$

$$+ \left[(2.47 r_3 - 64)\, s\theta_2\right]\dot{\theta}_2^2$$

$$+ \left[(712 + 0.78 r_3^2 - 39.5\, r_3)\, s\theta_2\, c\theta_2\right]\dot{\theta}_1\dot{\theta}_2$$

$$+ \left[(0.78\, r_3 - 19.7)\, s^2\theta_2\right]\dot{\theta}_1\, \dot{r}_3$$

$$+ \left[(-2.47\, c\theta_2\right]\dot{\theta}_2\, \dot{r}_3 = T_1 \quad \text{(oz-in)}$$

$$\left[668 + 0.39\, r_3 (r_3 - 51)\right]\ddot{\theta}_2$$

$$+ \left[(64 - 2.47\, r_3)\, c\theta_2\right]\ddot{\theta}_1$$

$$+ \left[(19.7\, r_3 - 356)\, s\theta_2\, c\theta_2\right]\dot{\theta}_1^2$$

$$+ \left[0.78\, r_3 - 19.7\right]\dot{\theta}_2\, \dot{r}_3$$

$$+ 389\left[10 - 0.39\, r_3\right] s\theta_2 = T_2 \quad \text{(oz-in)}$$

$$0.45\ddot{r}_3$$

$$- \left[2.47 s\,\theta_2\right]\ddot{\theta}_1$$

$$+ \left[(9.8 - 0.39 r_3) s^2\,\theta_2\right]\dot{\theta}_1^2$$

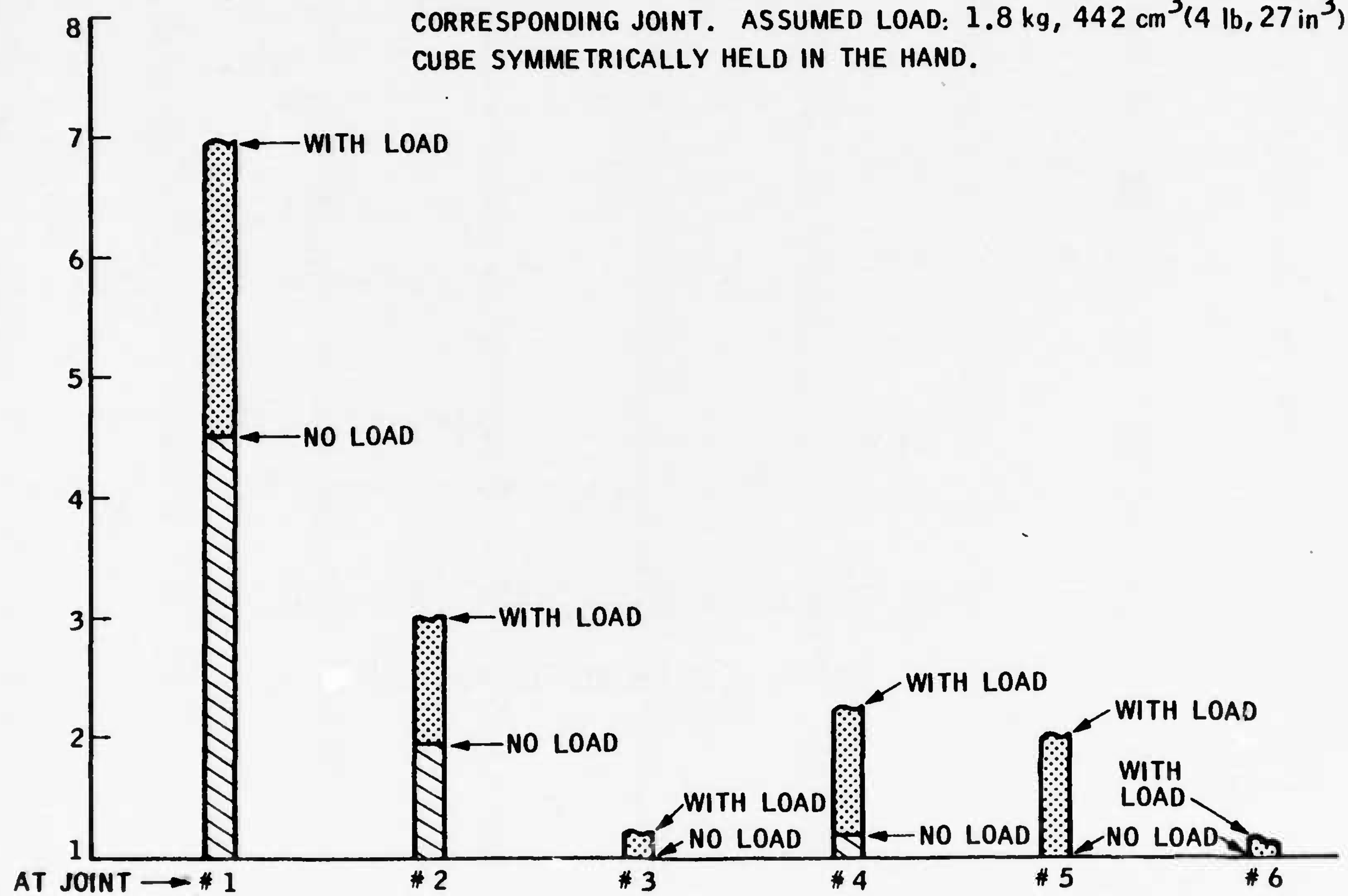
$$+ \left[9.8 - 0.39 r_3\right]\dot{\theta}_2^2$$

$$+ 151 c\,\theta_2 = F_3 \quad \text{(oz)}$$

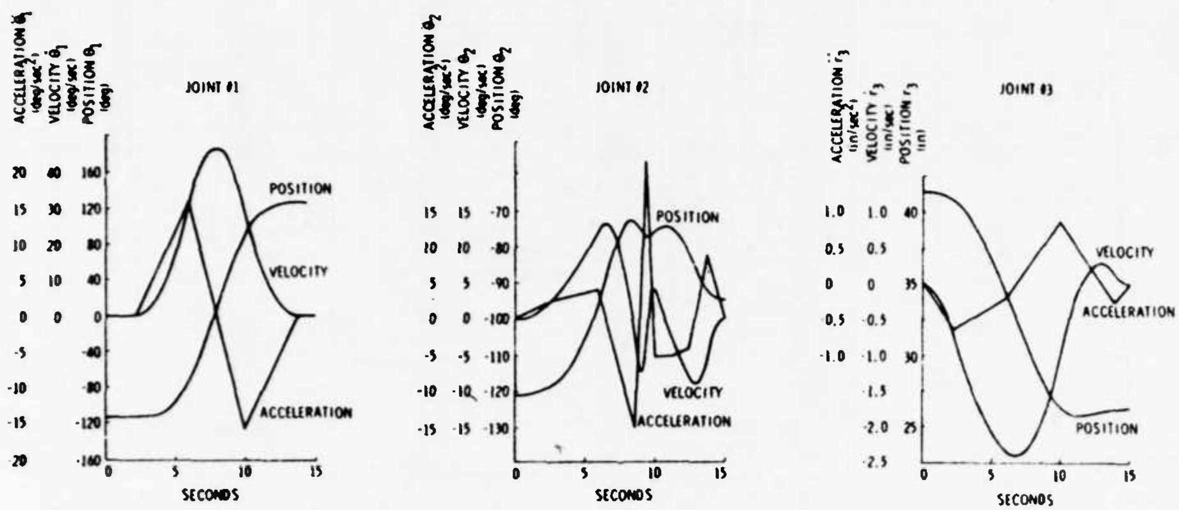
RELATIVE MAXIMUM VARIATIONS IN THE EFFECTIVE INERTIA FUNCTIONS

FOR THE JPL-STANFORD ARM

RELATIVE MAXIMUM VARIATIONS IN TOTAL LINK INERTIAS ARE REFERRED TO THE CORRESPONDING JOINT OUTPUT AND NORMALIZED TO THE MINIMUM TOTAL INERTIA VALUE AT THE CORRESPONDING JOINT. ASSUMED LOAD: 1.8 kg, 442 cm^3 (4 lb, 27 in^3) CUBE SYMMETRICALLY HELD IN THE HAND.

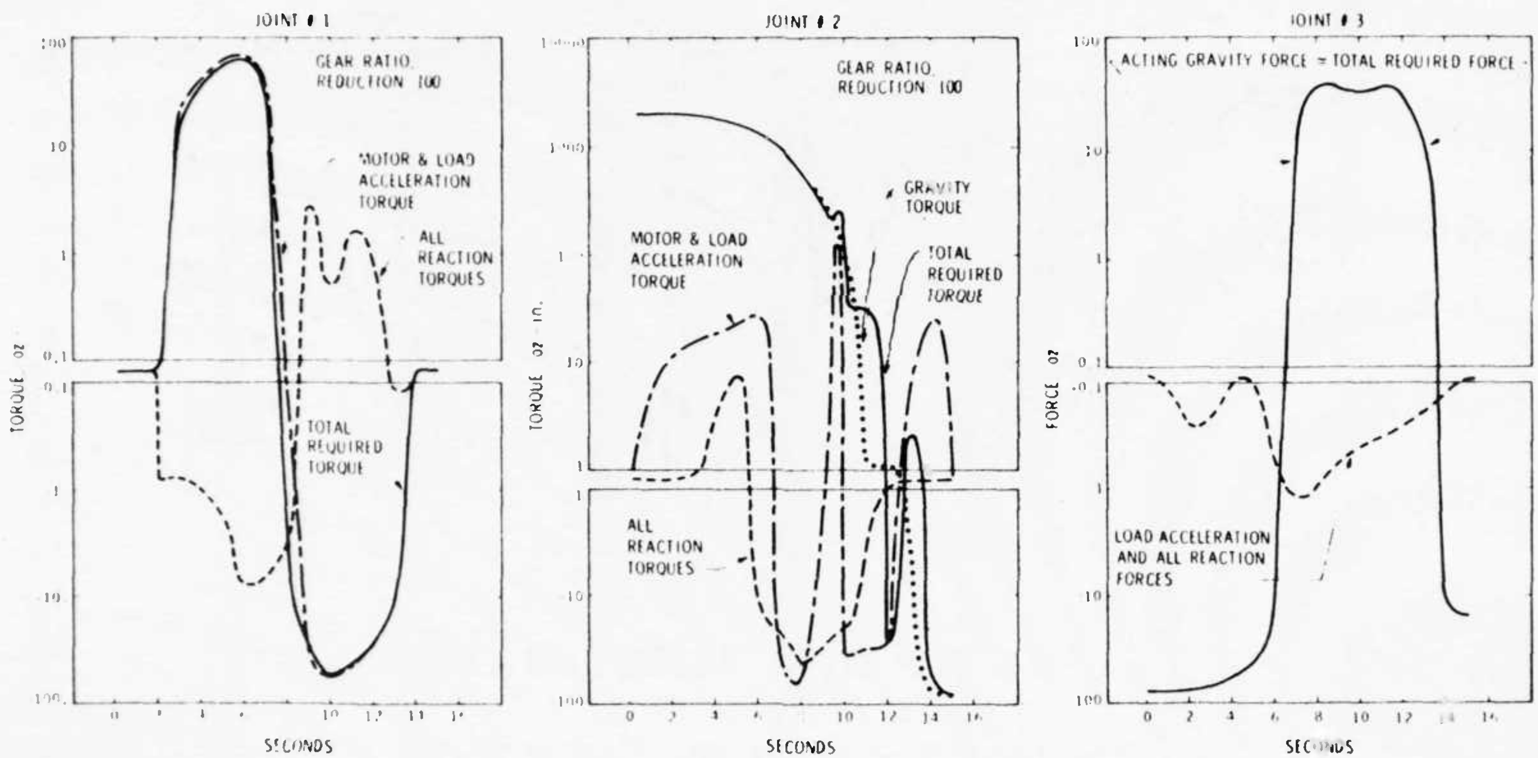


A MOTION PROFILE FOR THE JPL-STANFORD ARM UNDER PATH CONTROL IN THE WORK SPACE



Acceleration, Position, and Velocity Variations

THE DYNAMIC PROFILE OF A PATH-CONTROLLED MOTION OF THE JPL-STANFORD ARM. (THE CORRESPONDING JOINT MOTION VARIABLES ARE SHOWN ON THE PREVIOUS PAGE).



Torque and Force Variations

FURTHER "NATURAL" SIMPLIFICATIONS OF THE DYNAMIC EQUATIONS FOR THE JPL-STANFORD ARM

IT CAN BE SHOWN THAT, FOR THE LAST FOUR JOINTS OF THE JPL-STANFORD ARM THE FOLLOWING "DYNAMIC PROJECTION FUNCTIONS" ARE IDENTICALLY ZERO:

D_{333}, D_{344}, D_{366}, D_{444}, D_{455}, D_{466}, D_{555}, D_{566}, D_{644}, D_{666}, D_{446}, D_{556}, D_{334}, D_{335}, D_{336}, D_{345}, D_{346}, D_{356}, D_{434}, D_{435}, D_{436}, D_{534}, D_{535}, D_{536}, D_{634}, D_{635}, D_{636}, D_{433}, D_{533}, D_{633}, D_{45}, D_{56}, D_{34}, D_{36}, D_6.

References

J.J. Uicker, Jr., Dynamic Force Analysis of Spatial Linkages, ASME Paper No. 66-Mach-1, and published in Trans. ASME 1967.

M.E. Kahn and B. Roth, The Near-Minimum-Time Control of Open-Loop Kinematic Chains, ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 93, No. 3, Sept. 1971, pp. 164-172.

J.M. Hollerbach, A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation, IEEE Trans. on Systems, Man, and Cybernetics SMC-10, 11, Nov. 1980, pp. 730-736.

J.Y.S. Luh, et al., On-Line Computational Scheme for Mechanical Manipulation, Trans. ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 102, No. 2, June 1980, pp. 69-71.

R.P. Paul, Modeling, Trajectory Calculation and Servoing of a Computer Controlled Arm, Stanford Artificial Intelligence Laboratory, Stanford University, AIM 177, 1972.

R.P. Paul, Robot Manipulators: Mathematics, Programming and Control, The MIT Press, Cambridge, MA. and London, England, 1981.

A.K. Bejczy, Robot Arm Dynamics and Control, JPL Technical Memorandum 33-669, Pasadena, CA., February 1974.

A.K. Bejczy, Dynamic Models and Control Equations for Manipulators, JPL Report 715-19, Pasadena, CA., November 1979.

ROBOT MANIPULATOR DYNAMICS SUMMARY

MISSING ELEMENTS

- COMPUTER-BASED INTERACTIVE-AUTOMATIC STATE EQUATION REDUCTION/SIMPLIFICATION CAPABILITY
- SIMPLE AND RELIABLE TECHNIQUES FOR DETERMINING UNKNOWN OR POORLY KNOWN INERTIAL PARAMETERS
- RELATING DYNAMIC MODEL ANALYSIS TO ARM DESIGN IMPROVEMENTS
- MANY MORE IMPLEMENTATION OF FORCE-TORQUE CONTROL
- OPTIMIZATION ANALYSIS AND IMPLEMENTATION

NEW CHALLENGES

- DYNAMIC MODELS AND CONTROL OF FLEXIBLE ARMS
- MODELING AND CONTROLLING DYNAMIC IMPERFECTIONS
- COMPLIANCE CONTROL AND DYNAMICS
- SENSING AND CONTROL OF DYNAMIC BEHAVIOR

ROBOT MANIPULATOR SENSORS FOR CONTROL

1. INTERNAL STATE SENSORS:

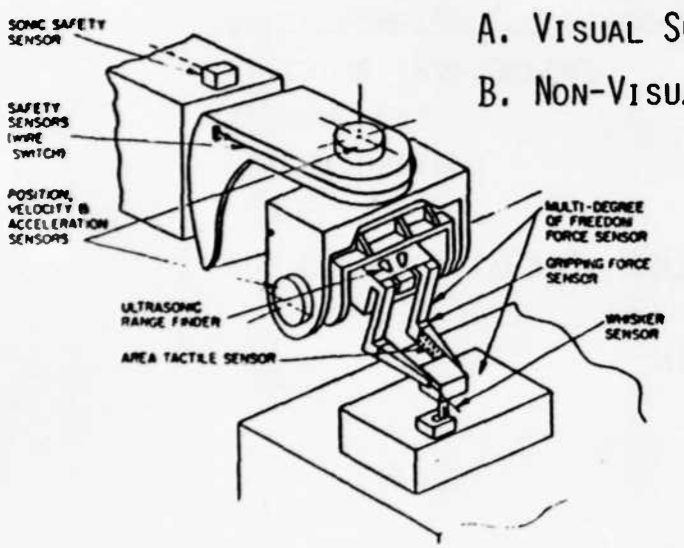
   JOINT POSITION
   VELOCITY
   ACCELERATION OR FORCE/TORQUE

2. EXTERNAL STATE SENSORS:

   ROBOT'S GEOMETRIC AND/OR DYNAMIC RELATION TO THE TASK, OBJECTS AND ENVIRONMENT

   A. VISUAL SENSORS (SEE SESSION 3)

   B. NON-VISUAL SENSORS



INTERNAL STATE SENSORS

1. FOR FEEDBACK (SERVO) CONTROL OF THE ROBOT'S INTERNAL STATE

   POTENTIOMETERS
   TACHOMETERS
   PRESSURE TRANSDUCERS
   ACCEREROMETERS OR FORCE/TORQUE GAGES
   OPTICAL ENCODERS FOR DIRECT DIGITAL FEEDBACK

2. SELECTION TOGETHER WITH DRIVE AND SERVO SYSTEM DESIGN

3. STATE OF THE ART

   A. INDUSTRY
   B. R & D COMMUNITY

NON-VISUAL EXTERNAL STATE SENSORS

It is noted that the information provided by these sensors (proximity range, tactile pressure or slip, force and torque) normally is not contained in visual data at all or can be obtained from visual data sources only indirectly and at a high cost. Hence, these sensors supplement the available visual information for control. Their information content and the related control tasks can be summarized as follows:

Sensors	Information content	Control tasks
Proximity	Short distances in known direction between hand and objects	Avoid near obstacles; adjust positioning and alignment of hand near objects; find objects
Touch/Slip	Distribution and amount of contact area pressure between hand and objects	Contact, push, grasp and hold objects; adjust grip force
Force/Torque	Amount of force and torque exerted by hand on objects along three hand-referenced orthogonal directions	Move objects under external dynamical constraints

5.1 Proximity Ranging

Proximity sensors sense the proximity of an object relative to the sensor. Presently the object can be anywhere from a millimeter to a meter away. Some proximity sensors indicate only the presence of an object within a given sensing region, while others measure the distance between the object and the sensor.

Electro-Optical proximity sensors are the most common. Simple electro-optical switching devices are readily available on the market. The more elaborate electro-optical proximity sensors capable of giving distance information are custom made, and require special optics, electronics and data processing which can be implemented in microprocessors. However they are still simple devices. These sensors require a light source (typically a light emitting diode) and a photodetector. Fiber optic cables can be employed to remove the light source and light detector from the optic head. Custom-made proximity sensors, using fiber optics and microprocessor are described in [5.1-1]. Figure 5-1 shows the scheme of electro-optical proximity range sensing using fiber optics.

For distance measurements, intensity-based electro-optical proximity sensors must be calibrated against the reflectance of the surface since they measure the intensity of reflected light. Autocalibration techniques can also be developed rendering this type of proximity range measurements more adaptable to changing measurement conditions. Proximity sensors can also be based on triangulation utilizing e.g., a linear CCD array and a photo emitter.

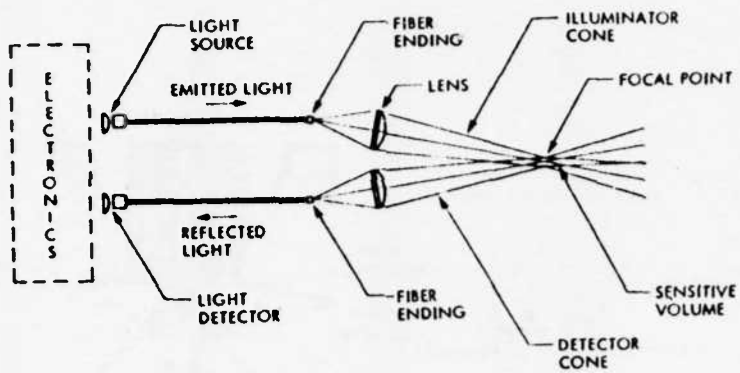


Figure 5-1. Infrared Electro-Optical Proximity Ranging at JPL Using Fiber Optics.

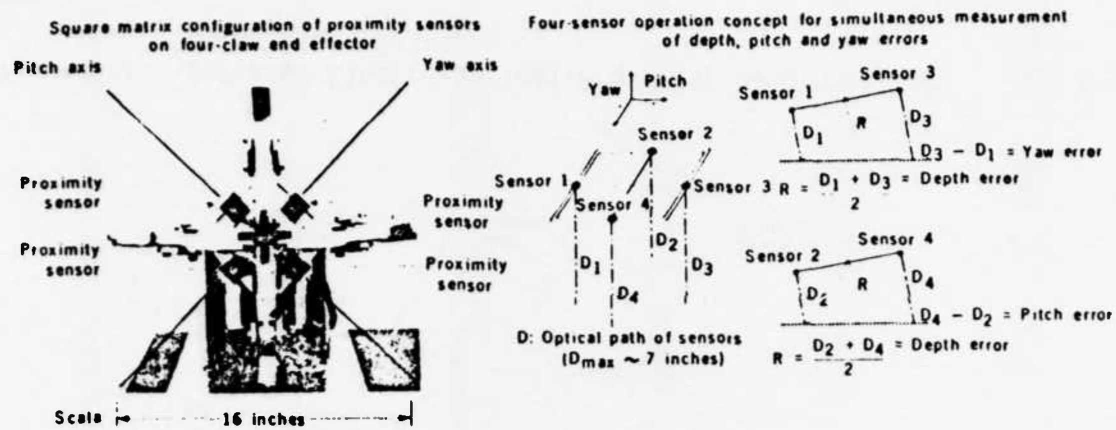


Figure 5-2. Proximity Sensor System for Space Shuttle Remote Manipulator System Control Application. (Left: square matrix configuration of proximity sensors on four-claw end effector. Right: four-sensor operation concept for simultaneous measurement of depth, pitch and yaw errors.)

Application of electro-optical proximity sensing developed at JPL has been successfully demonstrated for manual control of the Space Shuttle Remote Manipulator System using the Shuttle Mock-up Manipulator at the Johnson Space Center in Houston, Texas [5.1-2]. The sensor system measures range, pitch and yaw errors within a range of 15 cm and ± 15 deg, with a resolution of 2 mm and 0.5 deg. Figure 5-2 shows the measurement system.

A number of electromagnetic techniques can be employed for proximity or small displacement measurements. Capacitance, eddy current and Hall-effect devices are useful for proximity sensing whenever electromagnetic principles can be applied to generate some measurable signal. Eddy current devices operate only in very short (1-2 mm) distances. Reference [5.1-3] gives a good state-of-the-art review of electromagnetic proximity measurement techniques and lists some available devices.

Acoustic techniques can also be applied to proximity measurements. An interesting acoustic proximity detector was developed in [5.1-4]. It used an open-ended cylindrical resonator cavity. An acoustic source at the closed end sets up standing waves within the cavity. As the standing wave pattern moves, it changes the sound pressure which is detected by a microphone. Another type of acoustic proximity sensor, based on the sonar principle, is used in Polaroid cameras. The sensor and associated electronics are available for under $200.

5.2 Force-Torque Sensing

Force-torque sensors measure forces and torques that arise at the end effector during assembly or other dynamically constrained operations. These sensors can measure forces and torques applied about a point away from the sensor. Therefore, these sensors can be mounted at some appropriate place at the wrist. Force-torque sensors utilize force-summing elements that convert applied force (or pressure) into a physical displacement. The force-summing elements are linked to electrical transduction elements. The most common force summing/transduction element combinations are: strain gages, capacitance, piezoelectric, potentiometer, differential transformer and variable reluctance devices.

Four basic strain gage types are available: thin film (vacuum or sputter deposited), unbonded wire, bonded metal foil, and semiconductor (bonded bar or diffused). Probably, the thin-film type strain gages represent the most advanced state-of-the-art.

Thin film strain gage transducers use resistors deposited onto heat-treated steel substrates. The unbonded wire strain gages typically use very fine platinum tungsten strain wire as the sensing element. The bonded metal foil strain gages use a thin copper alloy foil as the sensing element. The bonded bar semiconductor strain gage is made of silicon elements bonded to a cantilever beam or diaphragm. The diffused-type semiconductor strain gages are made using photolithographic and diffusion techniques similar to those applied in manufacturing integrated circuits.

The capacitance pressure transducers use a moving diaphragm positioned between two fixed plates. Motion of the diaphragm as a result of applied pressure or force will cause a change of capacitance in two circuits which in turn is used to vary the frequency of oscillators or to null a capacitance bridge.

In piezoelectric transducers, a diaphragm is coupled to a selected axis surface of a crystal to induce strain. The most common crystals used for this purpose are quartz, barium titanate, tourmaline and Rochelle salt.

In potentiometric pressure or force transducers, a wiper can travel across a multiturn wire coil or deposited resistor in response to a very active force-actuated bellow or bourdon tube. The basic unit of a differential transformer transducer has only one moving part: a permeable core which changes position, as a result of applied pressure, relative to two identical transformer secondary windings. The changing position of the core will change the voltage induced into each of the secondaries which are connected in series opposition, and a differential voltage will appear across the secondary windings.

In the variable reluctance pressure or force transducers, a pressure or force-driven diaphragm changes the inductance of a pair of coils excited by a carrier frequency. The diaphragm can be either a displacement or a rotating type.

The characteristics and performance of various pressure transducers are summarized in two tables (Tables 5-2 and 5-3).

Table 5-2. Summary of Pressure/Force Transducer Characteristics [5.2-1]

Sensor	Excitation	Output Level	Accuracy	Pressure Range	Frequency Response	Temperature Range and Effects	Shock and Vibration Sensitivity	Stability*	Life or Calibration Shift with Use	Price Range ($)
Capacitance	ac-dc Special	Hi level and frequency bridge	0.05%	0.01 to 200psi	0 Hz to >100 Hz	0-1500°F	Poor to Good	0.05%/yr	>10⁷ cycles with <.05% cal. shift	1500-4500
Differential Transformer	ac Special	Hi level (5 volts) with phase de-mode/bridge	0.5%	30 to 10k psi	>100 Hz	0-165°F	Poor	0.25/yr	>10⁶ cycles life	50-500
Force Balance	ac Line Power	Hi level (>10 V) with servo	0.05%	1 psi to 5k psi	0 Hz to <5 Hz	40-165°F 0.01%/°F	Poor	0.05%/mo	>10⁷ cycles with <0.5% cal. shift	2000-4000
Piezoelectric	dc Amp and self-generating ac	Medium level with amplifier	1%	0.1 psi to 10k psi	1 Hz to >100 kHz	-450 to +400 0.01%/°F	Excellent	1%/yr	Unmeasurable use effects	300-500
Potentiometer	ac dc Regulated	Hi level	1%	5 psi to 10k psi	0 Hz to 50 Hz	−65 to +300 non-linear 0.01%/°F	Poor	0.5%/yr	<10⁶ cycles life	200-500
Strain Gage — Unbonded	ac dc Regulated 10 V ac-dc	Low level 4mV/V	0.25%	0.5 psi to 10k psi	0 Hz to >5 kHz	0.01%/°F −320° to +600°F (0.005%/°F over LCR**)	Good	0.5%/yr	<0.5% cal. shift after 10⁶ cycles	350-700
— Bonded Foil	10 V ac dc	Low level 3 mV/V	0.25%	5 psi to 10k psi	0 Hz to >5 kHz	−65°F to +250°F (0.005%/°F over LCR**)	Very good	0.5%/yr	>10⁶ cycles	150-600
Thin Film	10 V ac dc	3 mV/V	0.25%	1½ psi to 10k psi	0 Hz to >5 kHz	−320° to +600°F (0.005%/°F over LCR**)	Very good	0.25/yr	>10⁶ cycles with <0.25% cal. shift	450-700
Diffused Semiconductor	10 V 28 Vdc	Medium level 3 mV/V to 20 mV/V	0.25%	1½ psi to 5k psi	0 Hz to >5 kHz	−65°F to +250°F (0.005%/°F over LCR**)	Very good	0.25/yr	<0.25% calibration shift after 10⁶ cycles	80-250
Bonded Bar Semiconductor	10 V ac dc	Medium level 3 mV/V to 20 mV/V	0.25%	5 psi to 10k psi	0 Hz to 5 kHz	−65°F to +250°F (0.01%/°F over LCR**)	Very good	0.5%/yr	<0.5% calibration shift after 10⁶ cycles	35-240
Variable Reluctance	ac Special	40 mV/V	0.5%	1" H_2O to 10k psi	0 Hz to 1 kHz	−320° to +600°F (0.02%/°F over LCR**)	Very good	0.5%/yr	>10⁶ cycles life	250-600
Vibrating Wire and Tube	ac Special	Hi level and frequency	0.02%	1 psi to 100 psi	0 Hz to 100 Hz	−65°F to +200°F requires temperature control	Poor	0.01%/yr	>10⁶ cycles life	1500-6000

* Stability and calibration shift should be considered together.

** Limited compensated range.

Table 5-3. Comparative Performance Ranking of Pressure/Force Transducers

Small Size

1. Piezoelectric
2. Diffused Semiconductor Strain Gage
3. Unbonded Strain Gage
4. Bonded Bar Semiconductor Strain Gage
5. Thin Film Strain Gage
6. Variable Reluctance
7. Bonded Foil Strain Gage
8. Potentiometer
9. Capacitance
10. Differential Transformer
11. Vibrating Wire or Tube
12. Force Balance

Long-Term Calibration Stability

1. Vibrating Wire or Tube
2. Capacitance
3. Thin Film Strain Gage
4. Diffused Semiconductor Strain Gage
5. Bonded Bar Semiconductor Strain Gage
6. Bonded Foil Strain Gage
7. Unbonded Strain Gage
8. Variable Reluctance
9. Differential Transformer
10. Piezoelectric
11. Potentiometer
12. Force Balance

Output Level

1. Force Balance
2. Potentiometer
3. Capacitance
4. Vibrating Wire or Tube
5. Differential Transformer
6. Variable Reluctance
7. Piezoelectric
8. Diffused Semiconductor Strain Gage
9. Bonded Bar Semiconductor Strain Gage
10. Unbonded Strain Gage
11. Thin Film Strain Gage
12. Bonded Foil Strain Gage

Frequency Response

1. Piezoelectric
2. Unbonded Strain Gage
3. Diffused Semiconductor Strain Gage
4. Thin Film Strain Gage
5. Bonded Bar Semiconductor Strain Gage
6. Bonded Foil Strain Gage
7. Variable Reluctance
8. Capacitance
9. Vibrating Wire or Tube
10. Differential Transformer
11. Potentiometer
12. Force Balance

Small Static Error

1. Vibrating Wire or Tube
2. Force Balance
3. Capacitance
4. Thin Film Strain Gage
5. Diffused Semiconductor Strain Gage
6. Unbonded Strain Gage
7. Bonded Bar Semiconductor Strain Gage
8. Bonded Foil Strain Gage
9. Variable Reluctance
10. Differential Transformer
11. Piezoelectric
12. Potentiometer

Low Cost in Small Quantity

1. Bonded Foil Strain Gage
2. Unbonded Strain Gage
3. Potentiometer
4. Differential Transformer
5. Variable Reluctance
6. Thin Film Strain Gage
7. Bonded Bar Semiconductor Strain Gage
8. Piezoelectric
9. Diffused Semiconductor Strain Gage
10. Capacitance
11. Force Balance
12. Vibrating Wire/Tube

High Temperature

1.	Unbonded Strain Gage	600°F
2.	Variable Reluctance	600°F
3.	*Thin Film Strain Gage*	600°F
4.	Piezoelectric	400°F
5.	Potentiometer	300°F
6.	Bonded Foil Strain Gage	250°F
7.	Diffused Semiconductor Strain Gage	250°F
8.	Bonded Bar Semiconductor Strain Gage	250°F
9.	Vibrating Wire or Tube	200°F
10.	Force Balance	165°F
11.	Differential Transformer	165°F
12.	Capacitance	150°F

Operation under Acceleration and Vibration

1. Piezoelectric
2. Bonded Foil Strain Gage
3. Diffused Semiconductor Strain Gage
4. Thin Film Strain Gage
5. Unbonded Strain Gage
6. Bonded Bar Semiconductor Strain Gage
7. Vibrating Wire or Tube
8. Potentiometer
9. Variable Reluctance
10. Capacitance
11. Differential Transformer
12. Force Balance

Low Cost in Large Quantity

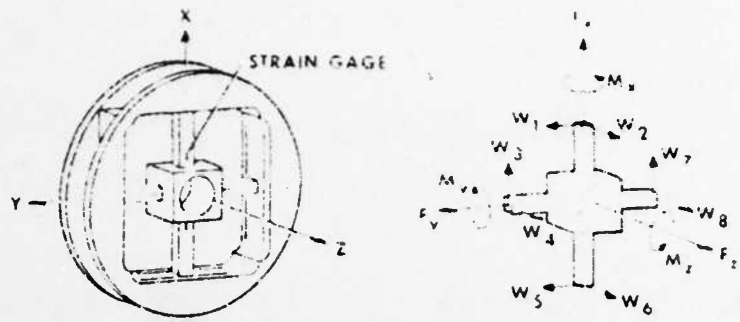
1. Diffused Semiconductor Strain Gage
2. Bonded Foil Strain Gage
3. Unbonded Strain Gage
4. Potentiometer
5. Differential Transformer
6. Thin Film Strain Gage
7. Variable Reluctance
8. Piezoelectric
9. Bonded Bar Semiconductor Strain Gage
10. Capacitance
11. Force Balance
12. Vibrating Wire or Tube

Strain gages (in particular, semiconductor strain gages) are the preferred transducers for six-dimentional force-torque sensors integrated with robot arms at the base of the mechanical band. Figures 5-3, 5-4 and 5-5 schematically illustrate the use of strain gages for six-dimensional force-torque balance sensors.

The sensor mechanism shown in Figure 5-3 was designed by V. Scheinman, now at Stanford University. A modified version of this sensor is presently under development at JPL for testing and evaluation on the Space Shuttle mock-up manipulator at Johnson Space Center (JSC), in Houston, Texas. The sensing range will be up to 1000N (about 200 lb).

The sensor shown in Figure 5-4 was developed at the C. S. Draper Lab. An updated version of this sensor is commercially available [5.2-4].

The sensor mechanism shown in Figure 5-5 was developed at SRI International for the NASA/AMES arm. The sensor is now at JPL and will be rebuilt.



TRANSFORMATION MATRIX
UNDER IDEAL CONDITIONS

FORCES AND TORQUES REFERENCED TO X-Y-Z SENSOR COORDINATES

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & k_{13} & 0 & 0 & 0 & k_{17} & 0 \\ k_{21} & 0 & 0 & 0 & k_{25} & 0 & 0 & 0 \\ 0 & k_{32} & 0 & k_{34} & 0 & k_{36} & 0 & k_{38} \\ 0 & 0 & 0 & k_{44} & 0 & 0 & 0 & k_{48} \\ 0 & k_{52} & 0 & 0 & 0 & k_{56} & 0 & 0 \\ k_{61} & 0 & k_{63} & 0 & k_{65} & 0 & k_{67} & 0 \end{bmatrix} \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \\ W_5 \\ W_6 \\ W_7 \\ W_8 \end{bmatrix}$$

FORCES SENSED AT SPOKE ELEMENTS

Figure 5-3. Use of Strain Gages on Cross-Beam Force Summing Elements for Six-Dimentional Force-Torque Sensing [5.1-1].

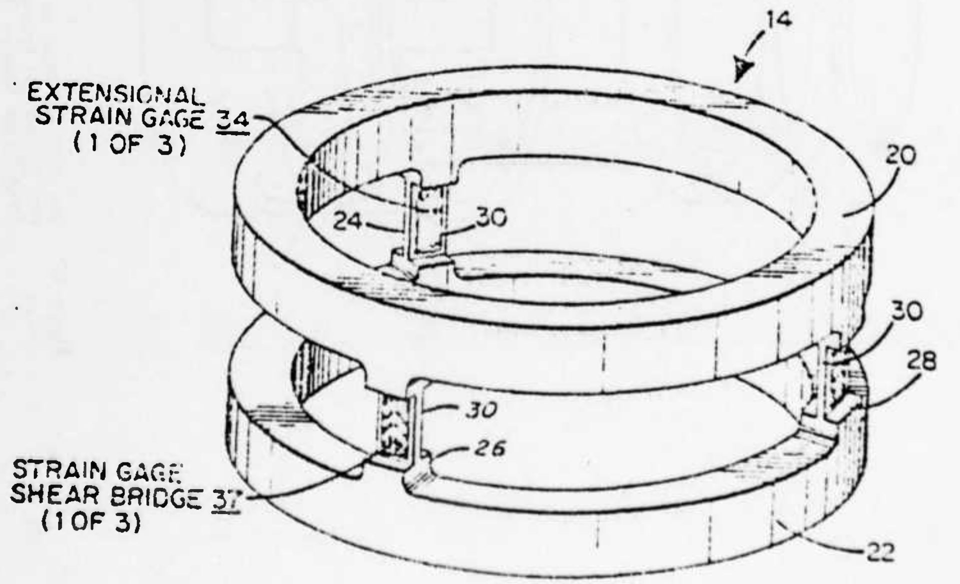


Figure 5-4. Use of Strain Gages on Three Equally Spaced Legs Supported by Cylindrical Disks or Rings for Six-Dimensional Force-Torque Sensing. (From [5.2-2] and U.S. Patent No. 4,094,192.)

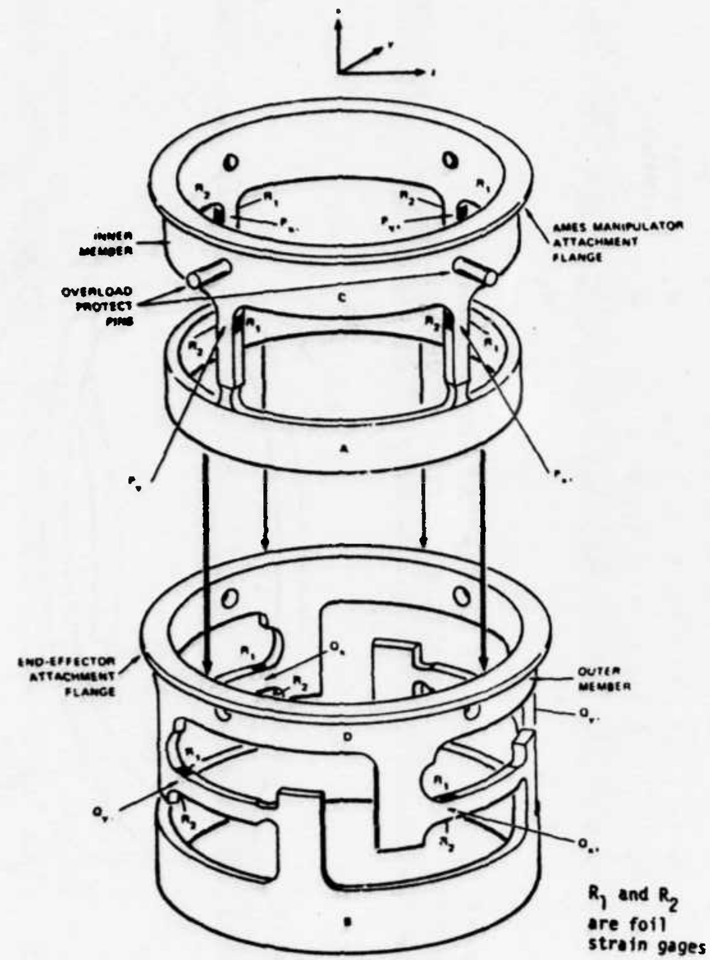


Figure 5-5. Use of Strain Gages on Flexing Beams in a Two-Part Cylindrical Arrangement for Six-Dimensional Force-Torque Sensing [5.2-3].

5.3 Tactile Sensing

Tactile sensors sense mechanical contact of the end effector or some other part of the manipulator with an object. Tactile sensors can be single point sensors, multiple point (array) sensors, simple binary (yes-no)sensors, or proportional sensors.

Tactile sensors usually use microswitches, strain gages, or pressure-conductive elastometers as transducers. Other applicable techniques are based on electromagnetic, capacitive, piezoelectric and piezoresistive transducer elements.

The simplest, least expensive and probably the most commonly used form of tactile sensing is by microswitches. The other most common tactile sensing technique employs pressure-conductive materials. Figure 5-6 illustrates the use of a pressure-conductive material for multi-point (array) tactile sensing. The choice of pressure-conductive materials for tactile sensing is quite limited. Reference [5.3-1] provides useful data on pressure-conductive elastomers.

New entries into the field of tactile sensing include piezo-resistive transducers using bonded doped-silicon bars and silicon-doped resistive elements, formed by IC techniques. Another new sensing concept at JPL uses infrared light and fiber optics [5.3-2] as illustrated in Fig. 5-7.

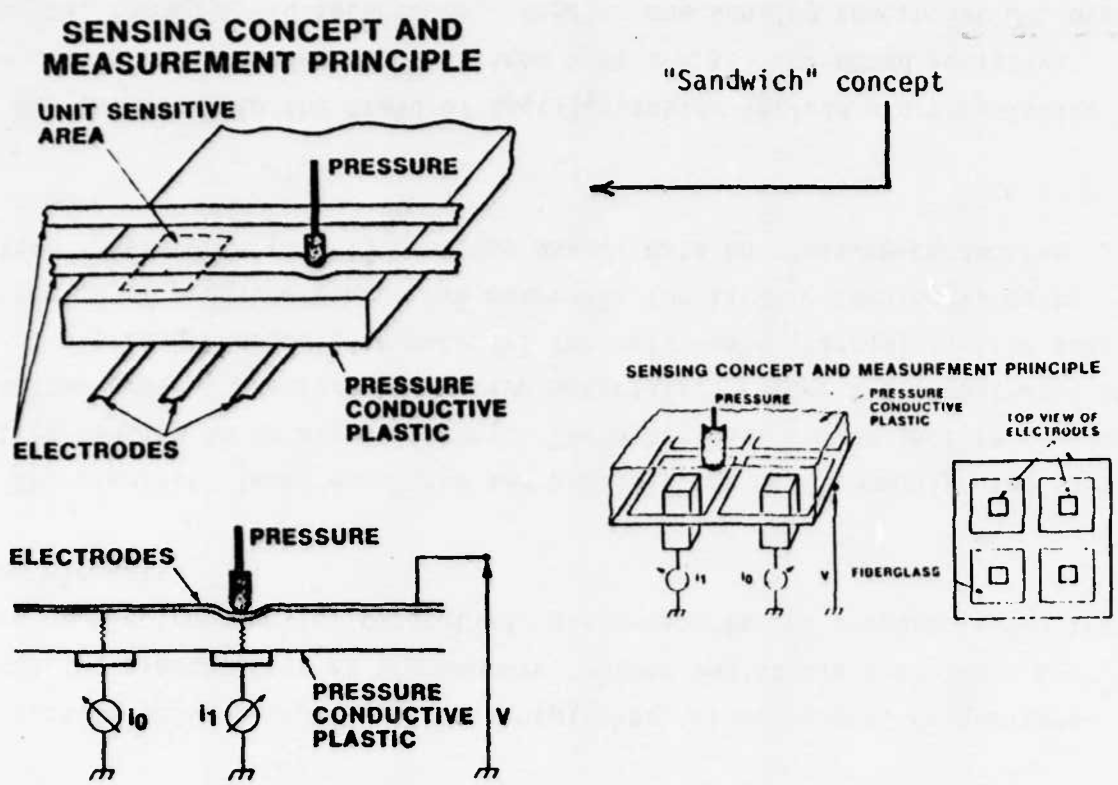


Figure 5-6. Use of pressure-conductive Elastomer for Touch Sensing at JPL (From. Ref. 5.1-1).

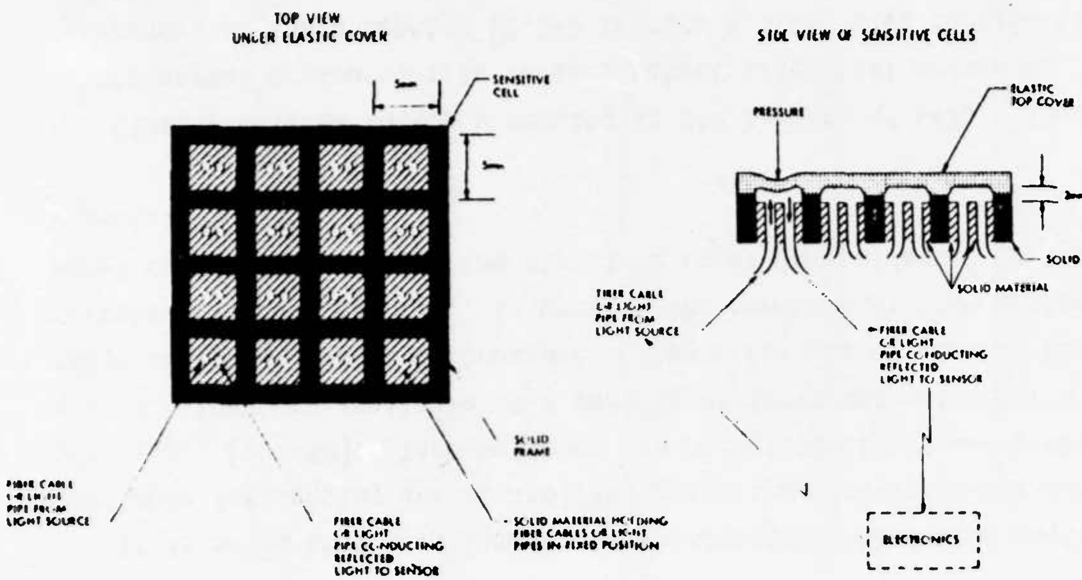


Figure 5-7. Use of Infrared Light and Fiber Optics for Touch Sensing at JPL. (From Ref. 5.3-2).

References [5.3-3 through 5.3-46] list useful literature in the field of touch sensing. Touch sensing technology - in a broad sense - has been reviewed recently in [5.3-47]

A major conclusion regarding tactile sensing is that this field is mostly dominated by R&D efforts. Touch sensors are not quite mature yet for realistic applications, but they are on the verge of becoming applicable within a few years.

It is noted that very simple and inexpensive ribbon tape switches are available for sensing and controlling contact by "press-at-any-point" See, e.g., [5.3-48]. Tape switches can be moistureproof and properly sealed. They are available in a variety of forms and rated for a large range of contact force or pressure (from a few ozs to several lbs). Tape switches can be used, e.g., to protect the manipulator from accidental heavy collisions when the arm structure comes in accidental contact with objects.

Closely related to touch sensing is the sensing of slip. Touch sensors detect normal pressure, slip sensors detect tangential pressure. The general references on touch sensing [5.3-3 through 5.3-46] also contain literature on slip sensing.

Figure 5-8 shows the concept of an omnidirectional slip sensor developed at JPL. This bench-model sensor detects slip from any direction, identifies the slip direction within 20 deg annular segment, and measures the rate of slip. Further information on this sensor can be found in [5.1-1].

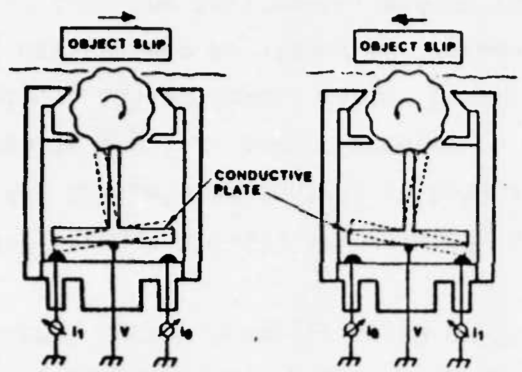


Figure 5-8. Omnidirectional Slip Sensing [5.1-1].

5.4 Application Techniques

From a man-machine interface design point-of-view, there are two markedly different techniques for applying external (proximity, force-torque and tactile) sensors for manipulator control: direct and indirect techniques. In a "direct" technique, the controlsystem acts directly - that is, automatically - on the sensory information. Once the automatic control action has been initiated, the operator only monitors ("supervises") the performance and intervenes when necessary. In an "indirect" technique, the sensory information is first conveyed to the operator through appropriate displays, and the operator commands the control system manually based on the sensory information.

References [5.4-1 through 5.4-63] are related to the application of sensor information for manipulator control in both automatic and manual control modes. Included in the list are references related to special purpose compliance devices, called Remote Center Compliance (RCC) devices. In some sense, the RCC devices can be viewed as "force-torque sensors and actuators" integrated into one instrument, aiding the performance of narrow-tolerance fitting tasks. A "compliant center" is defined as the point about which rotation will occur when a torque is applied and only translation results when a force is applied. Projecting this point to the leading edge of a shaft to be inserted into a hole is the basis for the RCC performing its function [5.4-62]. Figures 5-9 and 5-10 illustrate two somewhat different compliance devices available commercially.

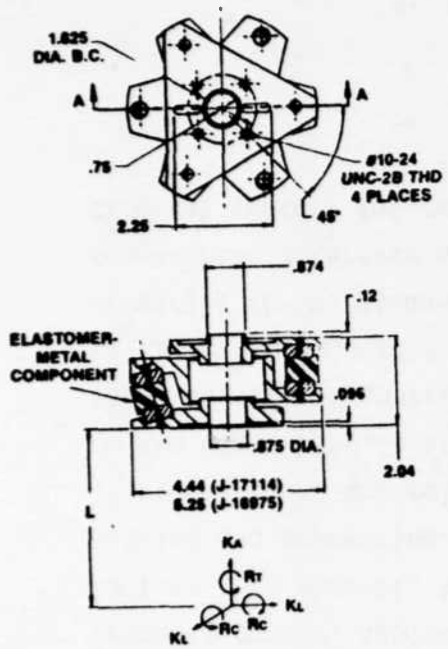


Figure 5-9. Remote Center Compliance Device [5.4-64, Lord Corporation.]

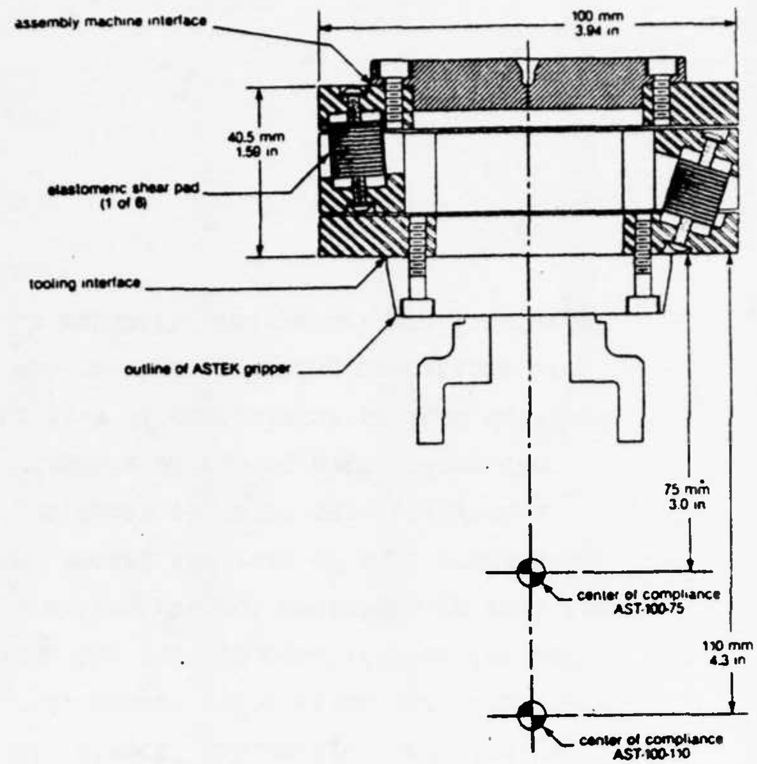


Figure 5-10. Remote Center Compliance Device [5.4-65, Astek Engineering, Inc.]

The advantages and disadvantages of "direct" (automatic) and "indirect" (manual) control applications of external sensor information vary from task to task. In general, the human operator has limited capabilities for perceiving and processing sensor data and making control decisions in real time. The limitations are related to both the amount and rate of data processing. On the other hand, a trained and skilled human operator can represent a large variety of control "algorithms" capable of coping with unforeseen tasks or unpredictable events. In the case of manipulator systems designed primarily for maintenance, repair or non-routine servicing operations, the man-machine interface should allow both automatic and manual application of external sensory information for control.

REFERENCES ON NON-VISUAL EXTERNAL STATE SENSORS AND THEIR APPLICATION TO ROBOT CONTROL

5.1-1. Bejczy, A. K., "Smart Sensors for Smart Hands," Vol. 67 of Progress in Astronautics and Aeronautics, AIAA Publ., New York, New York, 1979, pp. 275-304.

5.1-2. Bejczy, A. K., Brown, J. W., Lewis, J. L., "Evaluation of Smart Sensor Displays for Precision Control of Space Shuttle Remote Manipulator," Proceedings of the 16th Annual Conference on Manual Control, MIT, Cambridge, MA. May 5-7, 1980.

5.1-3. McDermott, J., "Sensors and Transducers," A Special Report in EDN, March 20, 1980, pp. 123-142.

5.1-4. U.K. Patent Application 803/75, "Non Contacting Ultrasonic Distance Measurement," NRDC Inventions, London, England, National Research and Development Corporation, July 1976.

5.2-1. Williams, A. W., "Pressure Transducers; Selection is Never Simple," Electronic Products Magazine, September 1978, pp. 79-89.

5.2-2. Nevins, J. L., et al., "Exploratory Research in Industrial Modular Assembly," Technical Report R-1218, The Charles Stark Draper Laboratory, Inc., August 1978. Also U.S. Patent No. 4,094,192 for "Method and Apparatus for Six Degree of Freedom Force Sensing," June 13, 1978.

5.2-3. Hill, J. and Sword, A., "Study to Design and Develop Remote Manipulation Systems," NASA Contract NAS2-8652, Stanford Research Institute Quarterly Report, Menlo Park, California, June 1, 1975.

5.2-4. Astek Sales Brochure, Astek Engineering, Inc., Watertown, MA.

5.3-1. Snyder, W. E., and J. St. Clair, "Conductive Elastomers as Sensor For Industrial Parts Handling Equipment," IEEE Trans, on Instrum. and Measure, Vol. IM-27, No. 1, March 1978, 94-99.

5.3-2. Bejczy, A. K., "Application of Fiber Optics to Robotics," IFOC Magazine, November 1980.

5.3-3. Aida, S., Cordella, L., and Ivacevic, N., "Visual-Tactile Symbiotic System for Stereometric Pattern Recognition," Second International Joint Conference on Artificial Intelligence, Conf. Two, Imperial College, London 365-375, September 1971.

5.3-4. Bejczy, A. K., "Effect of Hand-Based Sensors on Manipulator Control Performance," Mechanism and Machine Theory, Vol. 12, 1977, 547-567, Pergamon Press.

5.3-5. Bejczy, A. K., "Manipulator Control Automation Using Smart Sensors," Electro/79 Convention, New York, New York, April 24-26, 1979. 16 pp.

5.3-6. Bejczy, A. K.,"Sensors, Controls and Man-Machine Interface for Advanced Teleoperation," Science, 20 June 1980, Volume 208, pp. 1327-1335.

5.3-7. Binford, T. O., "Sensor Systems for Manipulation," Remotely Manned Systems, California Institute of Technology, 1973, 283-291.

5.3-8. Briot, M., "The Utilization of an 'Artificial Skin' Sensor for the Identification of Solid Objects," Ninth International Symposium On Industrial Robots, Washington, D.C., March 13-15, 1979, 529-547.

5.3-9. Collins, C. C., and Madey, J. M. J., "Tactile Sensor Replacement," Proceedings San Diego Biomed. Symposium., 13, 1974.

5.3-10. Dixon, J. K., et al., "Research on Tactile Sensors for an Intelligent Naval Robot," Ninth International Symposium on Industrial Robots, Washington, D. C., March 13 - 15, 1979, 507-517.

5.3-11. Fletcher, J. C., et al., "Tactile Sensing Means for Prosthetic Limbs," U.S. Patent 3,751,733, August 14, 1973.

5.3-12. Folchi, F. A., et al., "Tactile Sensors Operated by a Warping Plate and Buckling Beams," IBM Technical Disclosure Bulletin, Vol. 18. July, 1975, 575-577.

5.3-13. Fraioli, A. V., "Electro-Mechanical Hand Having Tactile Sensing Means," U.S. Patent, 3,509,583 May 5, 1970.

5.3-14. Gilliland, J. R., "Electrically Conductive Resinous Compositions," U.S. Patent 3,412,043 November 19, 1968.

5.3-15. Goto, T., Takeyasu, K., Inoyama, T., and Shimomura, R., "Compact Packaging By Robot With Tactile Sensors," Proceedings of the Second International Sumposium on Industrial Robots, Chicago, Illinois, May 1977.

5.3-16. Gurfinkel, V. S. et al., "Tactile Sensitizing of Manipulators," Engineering Cybernetics, 12 (6), 1974, 47-56.

5.3-17 Hill, J. W., and Sword, A. J., "Manipulation Based on Sensor-Directed Control: an Integrated End Effector and Touch Sensing System," Presented at Seventeenth Annual Human Factors Society Convention, Washington, D. C., October 16-18, 1973.

5.3-18. Hill, J. W., and Sword, A. J., "Touch Sensors and Control," Proc. of the First National Conference on Remotely Manned Systems; California Institute of Technology, 1973, 351-368.

5.3-19. Hirose, S. and Umetani, Y., "The Development of Soft Gripper for the Versatile Robot Hand," Mechanism and Machine Theory, Vol. 13, 1978. 351-359, Pergamon Press.

5.3-20. Ivancevic, N. S., "Stereometric Pattern Recognition by Artificial Touch," Pattern Recognition, Vol. 6, 1974, 77-83, Pergamon Press.

5.3-21. Johnson, J. B., and Huffman, T. R., "A Tactile Prosthetic Device for the Denervated Hand," Eleventh Annual Rocky Mountain Bioengineering Symposium and International ISA Biomedical Science Instrum. Symp., April 11, 1974, 9-10.

5.3-22. Kato, I., et al., "Artificial Softness Sensing - an Automatic Apparatus for Measuring Viscoelasticity," Mechanism and Machine Theory, Vol. 12, 1977, 11-26, Pergamon Press.

5.3-23. Kinoshita, G., Aida, S., and Mori, M., "Pattern Recognition by an Artificial Tactile Sense," Second International Joint Conference on Artificial Intelligence, Conf. 2, Imperial College, London, 376-384, Sept. 1971.

5.3-24. Kinoshita, G. I., "Classification of Grasped Object's Shape by an Artificial Hand with Multi-Element Tactile Sensors," Information-Control Problems in Manufacturing Technology, 1977, 111-118, Pergamon Press.

5.3-25. Kinoshita, G. I., et al., "A Pattern Classification by Dynamic Tactile Sense Information Processing," Pattern Recognition, Vol. 7, 1975, 243-251, Pergamon Press.

5.3-26. Kwee, H. H., "The Spartacus Telethesis: Soft Touch," Conf. on Tele-Manipulators for the Physically Handicapped, Rocquencourt, France, September 4-6, 1978, 305-308.

5.3-27. Legasse, J., et al., *Tactile Pick-Up*, U.S. Patent 4,014,217, March 29, 1977.

5.3-28. Larcombe, M. H. E., "Tactile Perception for Robot Devices," First Conference on Industrial Robot Technology, Univ. of Nottingham, U.K.

5.3-29. Larcombe, M. H. E., "Tactile Sensing Using Digital Logic," Proc. Shop Floor Automation Conf., Paper No. 9, Birniekill Inst., National Engineering Lab., East Kilbride, December 11-13, 1973, pp. 7.

5.3-30. Larcombe, M. H. E., "Tactile Sensors, Sonar Sensors and Parallax Sensors for Robot Applications," Third Conference on Indus-rial Robot Tech. and Sixth Int. Symp. on Industrial Robots, Univ. of Nottingham, U.K. March 24-26, 1976, C3-25-32, 25-32.

5.3-31. Masuda, R., et al., "Slip Sensor of Industrial Robot and Its Application," Electrical Eng. in Japan, Vol. 96, No. 5, 1976, 129-136.

5.3-32. Mitchell, R. J., Pressure Responsive Resistive Material, U.S. Patent 3,806,471 April 1974.

5.3-33. Nevins, J. L., "Sensors for Industrial Automation," Yearbook in Science and Technology, 1975, McGraw-Hill.

5.3-34. Okada, T., and Tsuchiya, S., "Object Recognition by Grasping," Pattern Recognition, Vol. 9. 1977, 111-119, Pergamon Press.

5.3-35. Page, C. J., et al., "New Techniques for Tactile Imaging," The Radio and Electronic Engineer, Vol. 46. No. 11, Nov. 1976, 519-526.

5.3-36. Parks, J. R., "Sensor Devices for Industrial Manipulators and Tools," First CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators, Udine, Italy, September 1973.

5.3-37. Pugh, A., et al., "Novel Technique for Tactile Sensing in a Three-Dimensional Environment," The Industrial Robot, March, 1977, 18-26.

5.3-38. Rosen, C. A., and Nitzan, D., "Use of Sensors in Programmable Automation," IEEE Computer Society Magazine, December 1977, 12-23.

5.3-39. Samaun, S., et al., "An IC Piezoresistive Pressure Sensor for Biomedical Instrumentation," IEEE Int. Solid-State Circuits Conf., U. of Pa. 1971, 104-105.

5.3-40. Samaun, K. D. W., and Angell, J. B., "An IC Piezoresistive Pressure Sensor for Biomedical Instrumentation," IEEE Trans. on Biomedical Eng., Vol. BME-20, No. 2, March 1973, 101-109.

5.3-41. Sessler, G. M., et al., "New Touch Actuator Based on the Foil-Electret Principle," IEEE Trans. on Communications, Vol. COM-21, No. 1 January 1973, 61-65.

5.3-42. Shannon, G. F., "Characteristics of a Transducer for Tactile Displays," Biomedical Engineering, June 1974, 247-249.

5.3-43. Sheridan, T. B., et al., "Tactile Sensing for Remote Palpation and Manipulation in Telediagnosis," Twentieth Annual Conf. on Eng. in Med. and Biol., Boston, Ma. November 13-16, 1967, 23.3.

5.3-44. Wang, S. S. M., and Will, P. M., "Sensors for Computer Controlled Mechanical Assembly," The Industrial Robot, March 1978, 9-18.

5.3-45. Stojiljkovic, Z. and Clot, J., "Integrated Behavoir of Artificial Skin," IEEE Trans. on Biomed. Eng., Vol. BME-24, No. 4, July 1977, 396-399.

5.3-46. Ueda, M., et al., "Tactile Sensors for an Industrial Robot to Detect a Slip," Proc. Second Int. Symp. on Ind. Robots, Chicago, Il. May 16-18, 1972, 63-76.

5.3-47. Harmon, L. D., "Touch Sensing Technology: A Review," SME Technical Report MSR-80-03, October 1980.

5.3-48. Sales Brochure, Tapeswitch Corporation of America, Inc., 1979.

5.4-1. Agin, G., "Vision Systems for Inspection and Manipulator Control," Proc. 1977 Jt. Automatic Control Conf., San Francisco, Ca. 22-24 June, 1977 (Piscataway,NJ): IEEE Svc, Ctr. pp. 132-138.

5.4-2. Agin, G., "Real Time Control of a Robot With a Mobile Camera," Tech. Note 179 (Menlo Park, Ca.: SRI Int'l, Artif. Intel. Ctr., February 1979).

5.4-3. Albus, J. S., and Evans, J. M., "Robot Systems," Scientific American 234(2), 77-87, February 1976.

5.4-4. Bejczy, A. K., "Environment-Sensitive Manipulator Control," Proceedings of the IEEE Conference on Decision and Control, 1974 (Also in Thirteenth Symposium on Adaptive Processes, Phoenix, Az. November 1974).

5.4-5. Bejczy, A. K., "Algorithmic Formulation of Control Problems in Manipulation," Proc. Int. Conf. on Cybernetics and Society, IEEE, September 23-25, 1975, 135-142.

5.4-6. Bejczy, A. K., and Tomovic, R., "Pattern Recognition and Control in Manipulation," 1976 IEEE Conference on Decision and Control, Clearwater Bay, Florida, December 1976.

5.4-7. Bejczy, A. K., "Manipulation of Large Objects," Third CISM-IFTOMM Int. Sump. on Theory and Practice of Robots and Manipulators, Udine, Italy, September 12-15, 1978.

5.4-8. Bejczy, A. K., Vuskovic, M. I., "An Interactive Manipulator Control System," Proceedings of the Second International Symposium on Mini- and Microcomputers in Control, Fort Lauderdale, Florida, December 10-11, 1979.

5.4-9. Bejczy, A. K., et al., "Evaluation of Proximity Sensor Aided Grasp Control for Shuttle Arm," Proc. Fifteenth Annual Conf. on Manual Control, Wright State Univ., Dayton, Ohio, March 20-22, 1979, pp. 26.

5.4-10. Bejczy, A. K., and Zawacki, R. L., "Computer-Aided Manipulator Control," Proc. of the First Int. Symp. on Mini- and Microcomputers in Control, San Diego, Ca. January 8-9, 1979.

5.4-11. Catros, J. Y., et al., "Automatic Grasping Using Infrared Sensors," Eighth Int. Symp. on Industrial Robots, Stuttgart, W. Germany, May 30-June 1, 1978, 132-142.

5.4-12. Craig, J. J., and Raiber, M. H., "A Systematic Method of Hybrid Position/Force Control of Manipulators," Proceedings of Compsac 79, Chicago, Ill., November 6-8, 1979.

5.4-13. Craig, J. J., et al., "Hand-Eye Control in Robotic Assembly Task," Autofact West, Anaheim, Ca. November 17-20, 1980.

5.4-14. Drake, S., "Using Compliance in Lieu of Sensory Feedback for Automatic Assembly," Report T-657 (Cambridge, Ma. C. S. Draper Labs, Inc., September 1977.

5.4-15. Freedy, A., Hull, F., Weltman, G., and Lyman, J., "The Application of Sensory Information and Multifunction Learning to Autonomous Manipulator Control," Proceedings of the First CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators, Udine, Italy, 193-202, September, 1973.

5.4-16. Gleason, G., and Agin, G., "A Modular Vision System for Sensor-Controlled Manipulation and Inspection" Proc. Ninth Int'l. Symp. on Indust. Robots, Washington, D.C., 13-15 March 1979 (Dearborn, Mi: Soc. Manuf. Eng's Mkt. Svc. Dept.), pp. 57-70, and Tech Note 178 (Menlo Park, Ca.: SRI Int'l, Artif. Intel. Ctr., February 1979).

5.4-17. Goto, T., et al., "Compact Packaging by Robot with Tactile Sensors," Proc. Second Int. Symp. on Ind. Robots, Chicago, Il. May 16-18, 1977.

5.4-18. Grossman, D. D., and Blasgen, M. W., "Orienting Mechanical Parts by Computer-Controlled Manipulator," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-5, No. 5, 561-564, September 1975.

5.4-19. Gurfinkel',V., Schneider, A., Gurfinkel', E., Kanaev, E., and Fomin, S., "Some Aspects of Sensory Instrumentation for Robots and Manipulators," Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, 771-774, September 1975.

5.4-20. Hill, J. W., and Sword, A. J., "Touch Sensors and Control," Proceedings of the First National Conference on Remotely Manned Systems, Pasadena, Ca. 351-368, September 1972.

5.4-21. Hill, J. W., "Study to Design and Develop Remote Manipulator Systems," Stanford Research Institute, Menlo Park, Ca. February 1975 - July 1976.

5.4-22. Hill, J. W., "Force-Controlled Assembler," Presented at Robots II Conf. Detroit, Mi., November 1-3, 1977, pp. 10.

5.4-23. Inoue, H., "Force Feedback in Precise Assembly Tasks," Memo No. 308, Artificial Intelligence Laboratory, Mass. Inst. Tech., August 1974, pp. 31.

5.4-24. Johnston, A. R., "Proximity Sensor Technology for Manipulator End Effectors," Mechanism and Machine Theory, Vol. 12, 1977, 95-109, Pergamon Press.

5.4-25. Kinoshita, G., Aida, S., and Mori, M., "Pattern Classification of the Grasped Object by the Artificial Hand," Third International Joint Conference on Artificial Intelligence, Stanford Univ., Calif. 665-670, August, 1973.

5.4-26. Koskinen, K., and Niemi, A., "Object Recognition and Handling in an Industrial Robot System," Proc. Eighth Int'l. Symp. on Indust. Robots Vol. II, Stuttgart, 30 May - 1 June, 1978 (Bedford, England: Int'l. Fluidics Services, Ltd.) pp. 744-755.

5.4-27. Lewis, R. A., "Adaptive Control of a Robotic Manipulator," Proc. 1977 IEEE Conf. on Decision and Control, New Orleans, La. December 7-9, 1977, 743-748.

5.4-28. McGhie, D., and Hill, J., "Vision-Controlled Subassembly Station," Paper MS78-685 (Dearborn, Mi: Soc. Manuf. Eng's, Mkt. Svc. Dept., Nov. 1978), and Proc. Robots III Conf., Chicago, Il. 7-9 November 1978.

5.4-29. Nevins, J. L. and Whitney, D. E., "Information and Control Issues of Adaptable, Programmable Assembly Systems for Manufacturing and Teleoperator Applications," Mechanism and Machine Theory, Vol. 12, 1977, 27-43, Pergamon Press.

5.4-30. Nevins, J. L., and Whitney, D. E. "Assembly Research and Manipulation," Proc. 1977 IEEE Conf. on Decision and Control, New Orleans, La. December 7-9, 1977, 735-742.

5.4-31. Nevins, J. L., and Whitney, D. E., "Adaptable-Programmable Assembly System: An Information and Control Problem," Proceedings of the 5th International Symposium on Industrial Robots, Chicago, Il. September, 1975.

5.4-32. Nevins, J. et al., "Exploratory Research in Industrial Modular Assembly," Internal Report R-1111, The Charles Stark Draper Lab., Inc., Cambridge, Ma. September 1, 1976 to August 31, 1977, pp. 167.

5.4-33. Nevins, J. L., et al., "Exploratory Research in Industrial Modular Assembly," Internal Report R-1218, "The Charles Stark Draper Lab., Inc., Cambridge, Ma. September 1, 1977 to August 30, 1978, pp. 136.

5.4-34. Nitzan, D., "Machine Intelligence Research Applied to Industrial Automation," Sixth NSF Grantees Conf. in Production Research and Technology, Purdue Univ., West Lafayette, IN. September 27-29, 1978, pp. 7.

5.4-35. Novatchenco, S. I., Pavlov, V. A., Teleshev, N.S., and Yurevich, E. J., "Principles and Experience of Application of Sensitized Robots Supervisory Control," Second International CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators, Warsaw, Poland, 407-416, September 1976.

5.4-36. Okada, T., "Object-Handling System for Manual Industry," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-9, No. 2, February 1979, 79-89.

5.4-37. Paine, G. and Bejczy, A. K., "Extended Event-Driven Displays for Manipulator Control," Proceedings of the 15th Annual Conference on Manual Control, Dayton, Oh. March 1979.

5.4-38. Rebman, J., "Compliance: The Forgiving Factor," Robotics Today (Fall 1979), pp. 29-34.

5.4-39. Renaud, M., Briot, M., and Stojiljkovic, Z., "An Approach to Spatial Pattern Recognition of Solid Objects," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-8, No. 9, 690-693, September 1978.

5.4-40. Rosen, C., et al., "Exploratory Research in Advanced Automation," Second Report, Stanford Research Institute, Menlo Park, Ca. October 1973 to June 30, 1974.

5.4-41. Sakai, I., Kumazawa, T., Ueda, M., and Shingu, H., "Approach and Plan: Most Suitable Control of Grasping in Industrial Robot," Proceedings of the 5th International Sumposium on Industrial Robots, Chicago, Il. 525-532, September 1975.

5.4-42. Shannon, G. F., "The Case for Sensory Feedback on Artificial Limbs," Elec. Eng. Trans., 1975, 36-38.

5.4-43. Sheridan, T. B., and Ferrell, W. R., "Human Control of Remote Computer-Manipulators," Proceedings of the International Joint Conference on Artificial Intelligence, Washington, D. C. 1969.

5.4-44. Shimano, B., and Roth, B., "On Force Sensing Information and Its Use in Controlling Manipulators," Report, Mechanical Engineering Department, Stanford University, May 1973.

5.4-45. Shimano, B. E., "The Kinematic Design and Force Control of Computer-Controlled Manipulators," Ph.D. Thesis, Stanford U. Mech. Eng. Dept. Memo AIM-313 and Report STAN-CS-78-660, 1978.

5.4-46. Simunovic, S., "Force Information in Assembly Processes," Proceedings of the Fifth International Symposium on Industrial Robots, Chicago, Il. 415-432, September 1975.

5.4-47. Stojiljkovic, Z., and Saletic, D., "Learning to Recognize Patterns by the Belgrade Hand Prosthesis," Proceedings of the Fifth International Symposium on Industrial Robots, Chicago, Il. 407-414, September 1975.

5.4-48. Stojilkovic, Z., and Clot, J., "Integrated Behavior of Artificial Skin," IEEE Transactions on Biomedical Engineering, Vol. BME-24, No. 4, 396-399, July 1977.

5.4-49. Sword, A. J., "Study to Design and Develop Remote Manipulator Systems," Stanford Research Institute, Contract NAS2-8652, Menlo Park, Ca. 1975.

5.4-50. Sword, A. J., and Park, W. T., "Adaptive Control Study for Industrial Robots," Stanford Research Institute, Contract DAAA15-74-C-0095, Ca. 1975.

5.4-51. Sword, A. J., and Park, W. T., "Location and Acquisition of Objects in Unpredictable Locations," Mechanism and Machine Theory, Vol. 12, No. 1, Oxford, England, Pergamon Press, 123-132, 1977.

5.4-52. Takeyasu, K., Goto, T., and Inoyama, T., "Precision Insertion Control Robot and Its Application," ASME Paper 76-DET-50.

5.4-53. Tesar, D., "Conclusions for the NSF Robotics Workshop," Proc. NSF Workshop in the Impact on the Academic Community of Required Research Activity for Generalized Robotic Manipulators, U. of Florida, February 8-10, 1978, 8-12.

5.4-54. Tomovic, R. and Stojiljkovic, Z., "Multifunctional Terminal Device with Adaptive Grasping Force," Automatica, Vol. 11, 1975, 567-570, Pergamon Press.

5.4-55. Udupa, S. M., "Collision Detection and Avoidance in Computer Controlled Manipulators," Fifth International Joint Conference on Artificial Intelligence, Vol. 2, M.I.T., Cambridge, Ma. 737-748, August 1977.

5.4-56. Ueda, M. and Shimizu, T., "Sensors and Systems Necessary for Industrial Robots in the Near Future," Fourth Int. Symp. on Industrial Robots, Tokyo, November 19-21, 1974, 79-88.

5.4-57. Ward, M. R., et al., "Consight: A Practical Vision-Based Robot Guidance System," Nineth Int. Symp. on Industrial Robots, Washington, D.C., March 13-15, 1979, 195-211.

5.4-58. Whitney, D. E., "Force Feedback Control of Manipulator Fine Motions," Proceedings of the Joint Automatic Control Conference, Purdue Univ. West Lafayette, Id., 687-693, July 1976.

5.4-59. Whitney, D. E., Watson, R. C., Drake, S. H., and Simunovic, S. N., "Robot and Manipulator Control by Exteroceptive Sensors," Proceedings of the Joint Automatic Control Conference, San Francisco, Ca. 155-163, June 1977.

5.4-60. Whitney, D. E., "Force Feedback Control of Manipulator Fine Motions," Proc. 1976 Joint Automatic Control Conf., July 27-30, 1976, 687-693.

5.4-61. Whitney, D. E., et al., "What Is The Remote Center Compliance (RCC) and What Can It Do?" Internal Report P-728, The Charles Stark Draper Lab., Inc., Cambridge, Ma. November 1978.

5.4-62. Whitney, D. E., et al., "Robot and Manipulator Control by Exteroceptive Sensors," Proc. 1977 Joint Automatic Control Conf., 1977, 155-163.

5.4-63. Will, P. M. and Grossman, D. D., "An Experimental System for Computer Controlled Mechanical Assembly," Proc. IEEE Intercon. Session 13, New York, 1975, pp. 18.

5.4-64. Sales Brochure, Remote Center Compliance, Lord Kinematics, Lord Corporation, Erie, Pa. 1980.

5.4-65. Sales Brochure, Accommadator Model ASP-100, Astek Engineering, Inc. Watertown, Ma. 1980.

ROBOT MANIPULATOR CONTROL

1. INDUSTRY ROBOT CONTROL STATE-OF-THE-ART:

   - LIMITED SEQUENCE ROBOTS
   - PLAYBACK ROBOTS (TEACH-RECORDING)
     - END POINT CONTROL
     - CONTINUOUS CONTROL
   - EMERGING COMPUTER CONTROLS

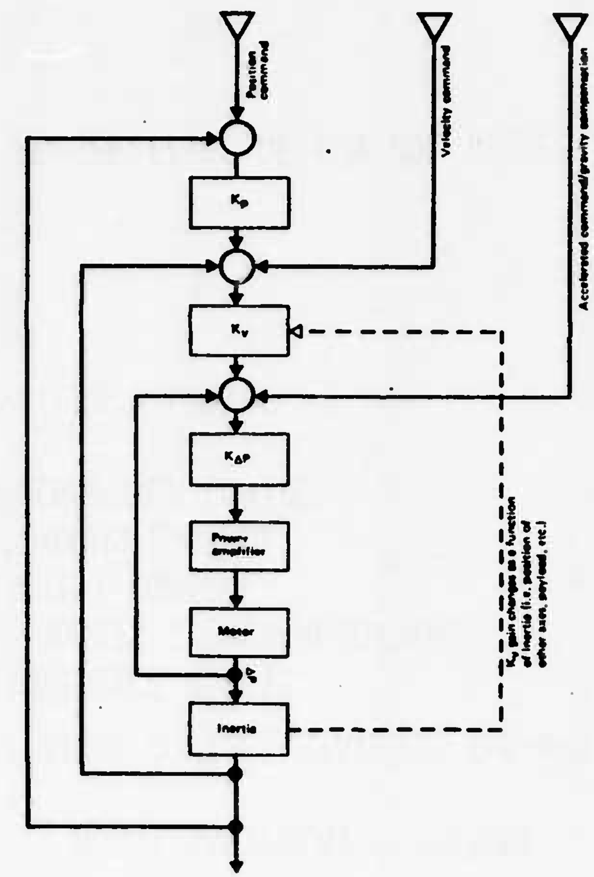
2. DRIVE SYSTEMS & SERVOS

   - PNEUMATIC
   - HYDRAULIC
   - ELECTRIC

3. SUMMARY PERSPECTIVES OF CONTROL ISSUES

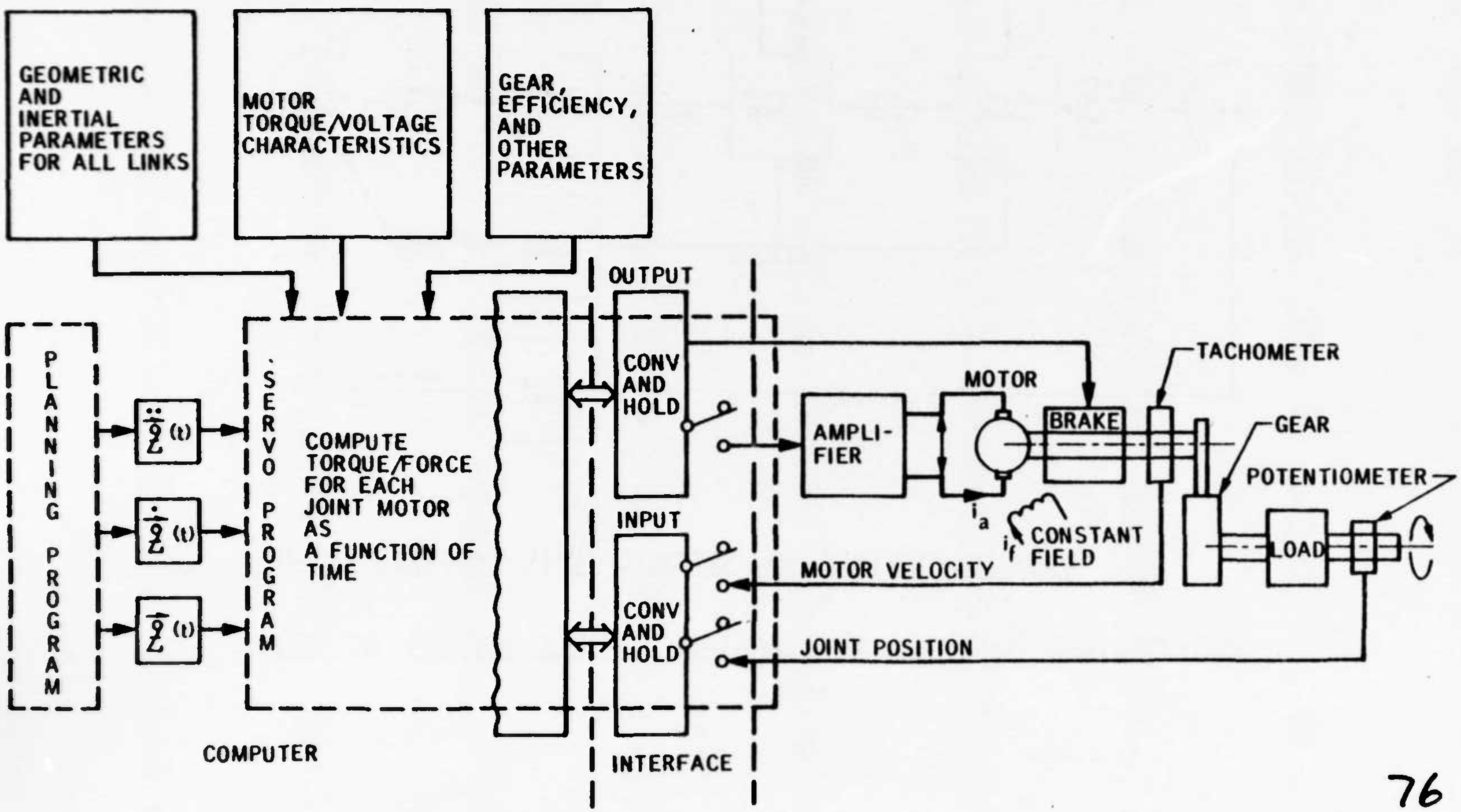
ROBOT MANIPULATOR CONTROL

SINGLE-AXIS SERVO SYSTEM MAIN ELEMENTS



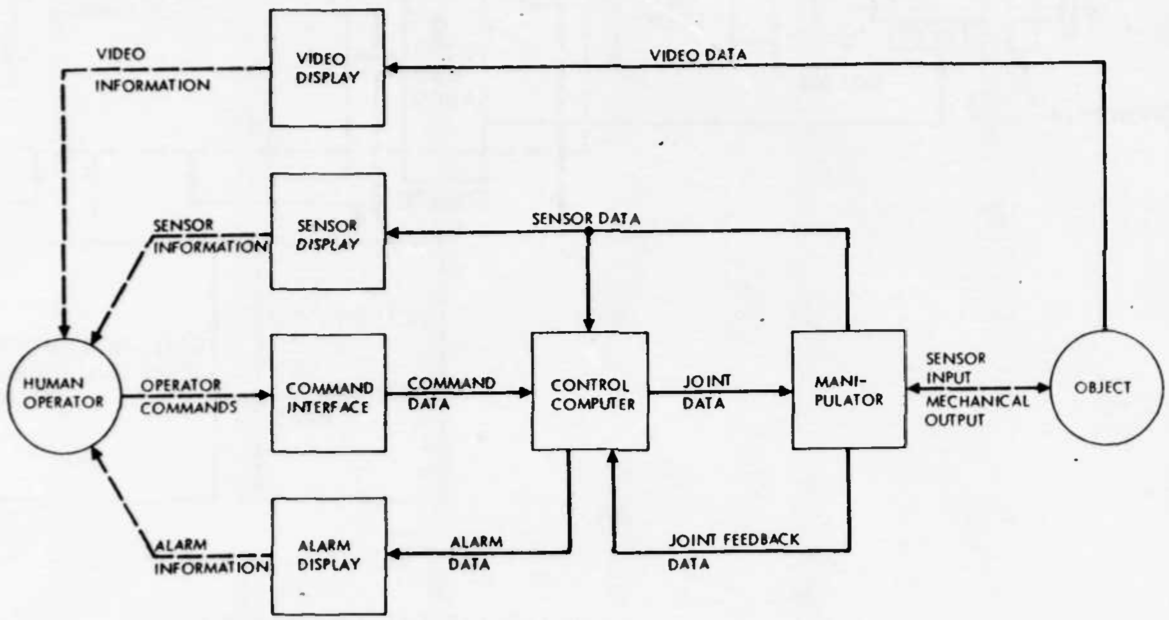
GENERAL ELEMENTS OF A COMPUTER CONTROL SYSTEM

FOR ROBOT MANIPULATORS



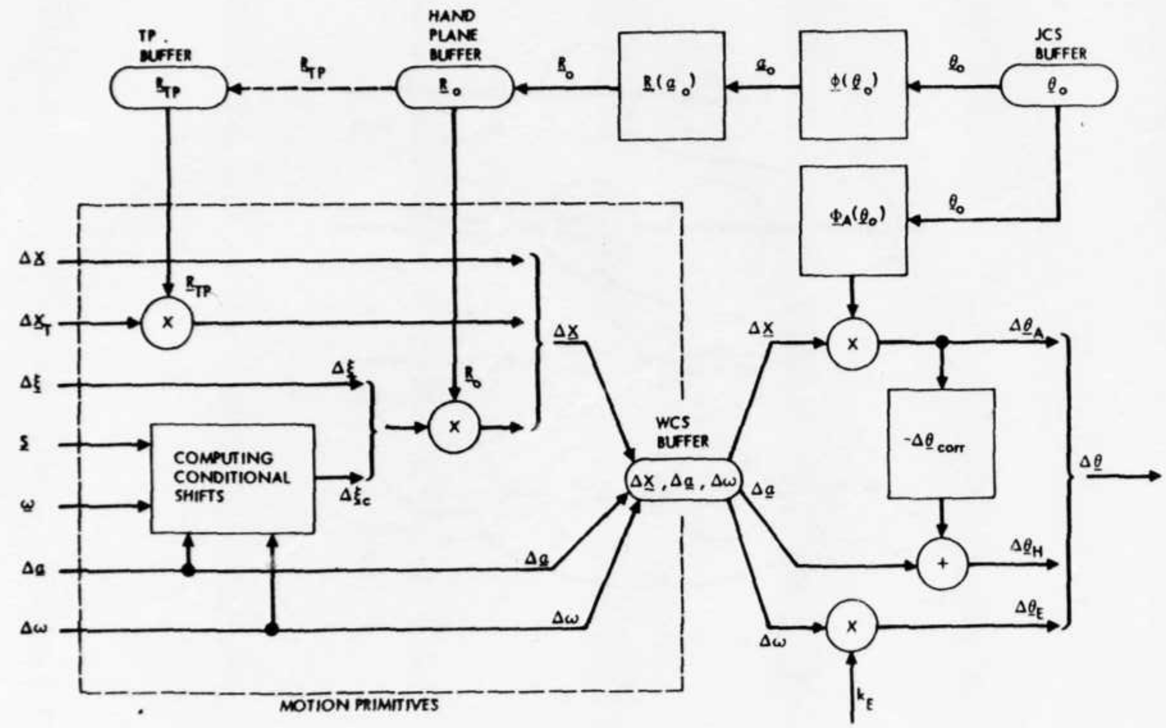
GENERAL CONTROL SYSTEM CONFIGURATION FOR ROBOT MANIPULATORS

USING EXTERNAL STATE SENSORS FOR CONTROL/GUIDANCE



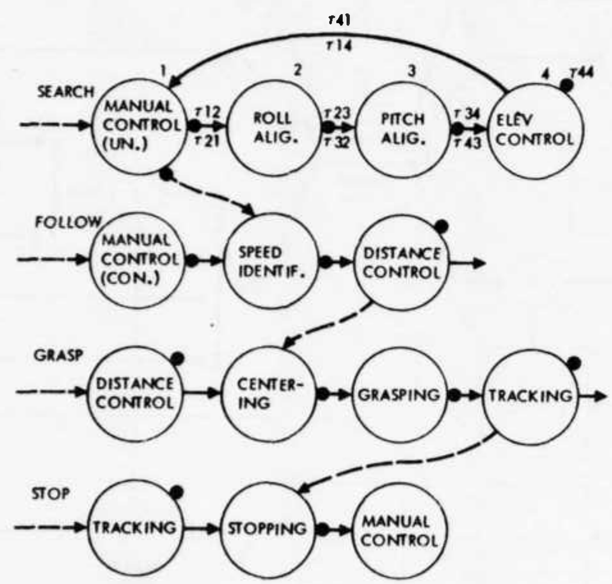
COORDINATE TRANSFORMATIONS IN AN INTERACTIVE SENSOR-REFERENCED ROBOT

MANIPULATOR CONTROL SYSTEM

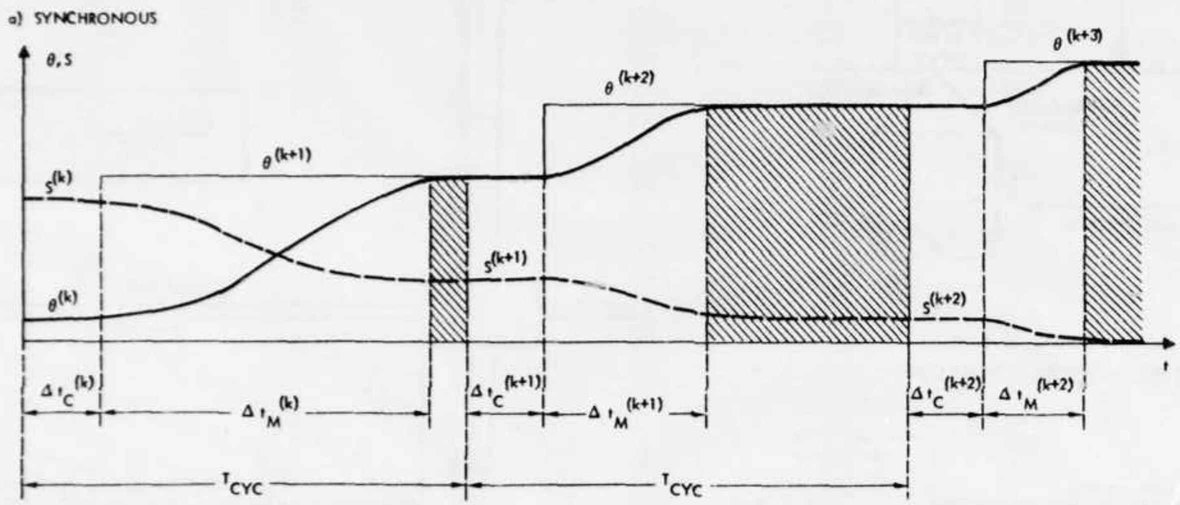


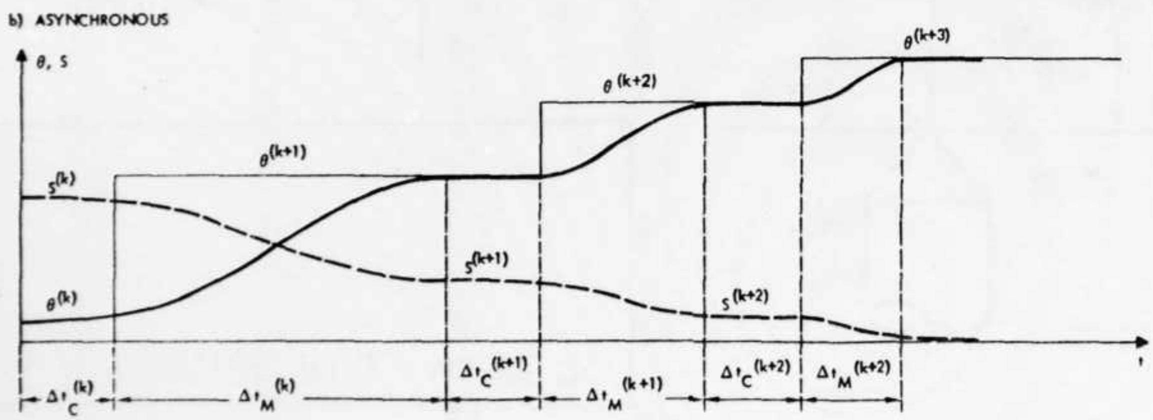
ACTION PRECEDENCE GRAPH LOGIC ORGANIZATION OF AN INTERACTIVE

SENSOR-REFERENCED ROBOT MANIPULATOR CONTROL SYSTEM



APPROACHES TO COMPUTER CONTROLLER SCHEDULING

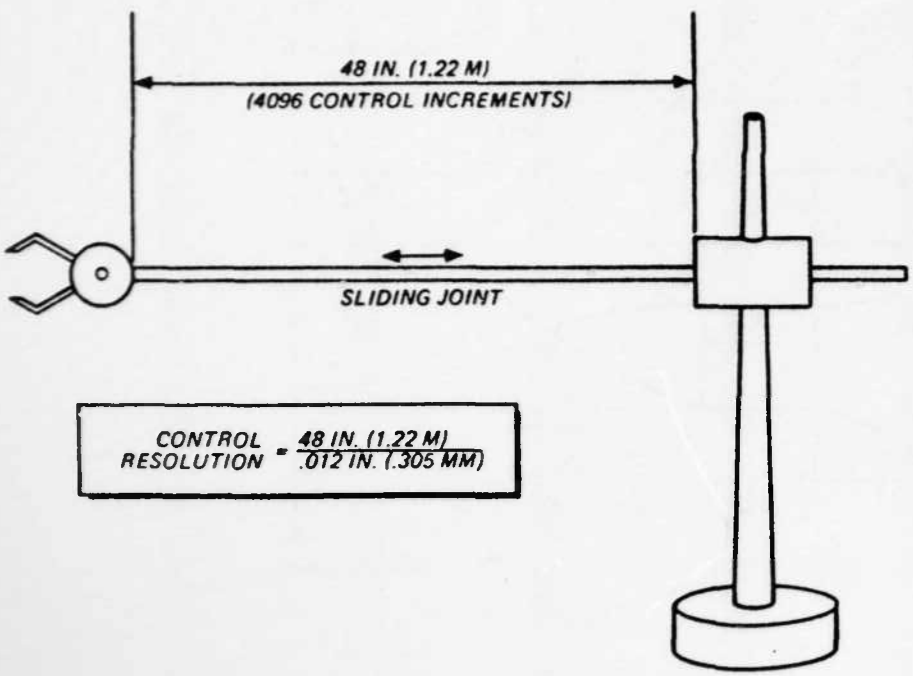




CONTROL - ACCURACY - RESOLUTION

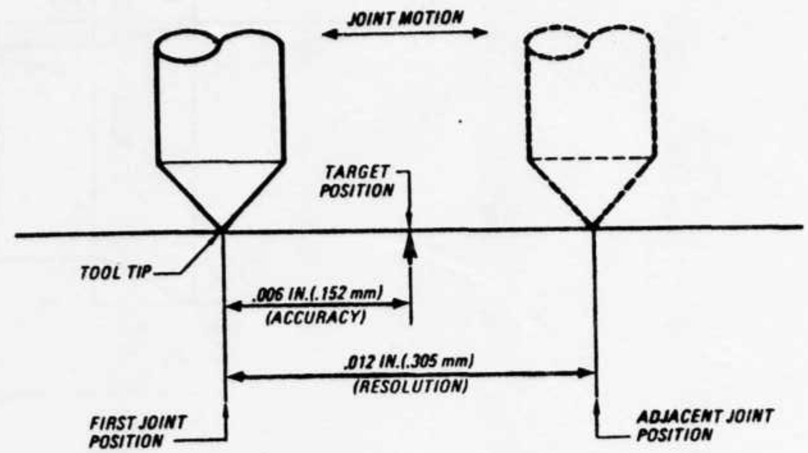
(USEFUL REMARKS)

ICAM ROBOTICS APPLICATIONS GUIDE, VOLUME 2,
AFWAL-TR-80-4042
APRIL, 1980
AFWAL/MLTC
WPAFB, OHIO 45433

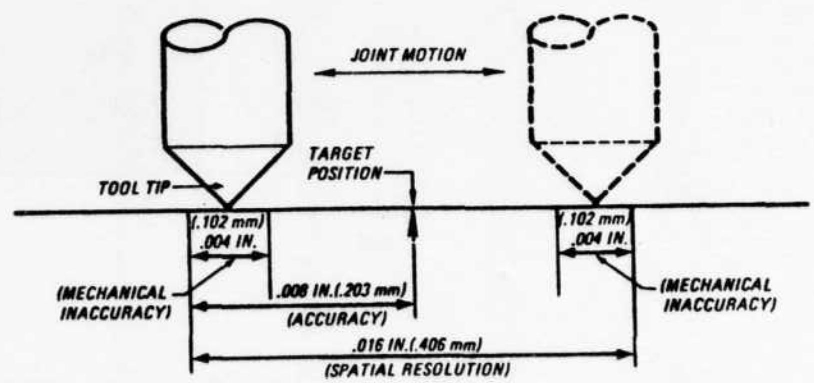


$$\text{CONTROL RESOLUTION} = \frac{48\text{ IN. } (1.22\text{ M})}{.012\text{ IN. } (.305\text{ MM})}$$

CONTROL INFLUENCE ON RESOLUTION



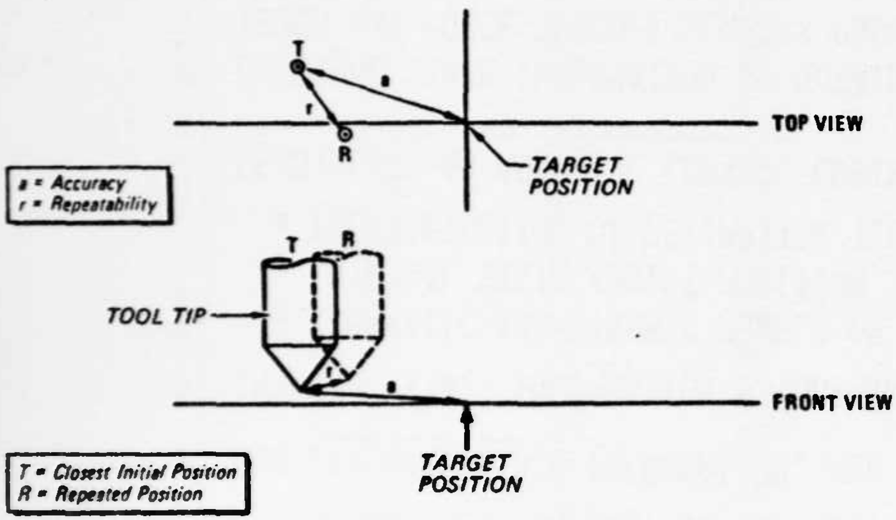
ACCURACY/RESOLUTION RELATIONSHIP



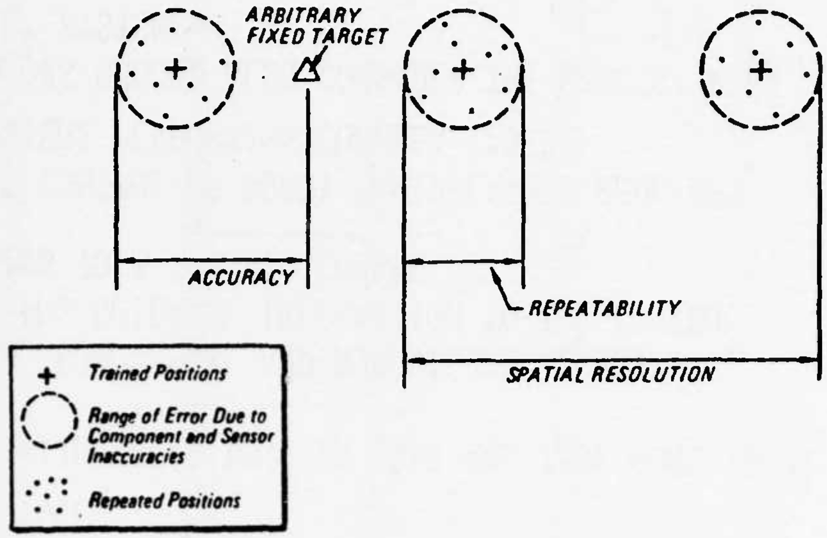
ACCURACY/SPATIAL RESOLUTION RELATIONSHIP

CONTROL - ACCURACY - REPEATABILITY
(USEFUL REMARKS)

ICAM ROBOTICS APPLICATIONS GUIDE, VOLUME 2,
AFWAL-TR-80-4042
APRIL, 1980
AFWAL/MLTC
WPAFB, OHIO 45433



ACCURACY – REPEATABILITY RELATIONSHIP (EXAGGERATED) SHOWING REPEATABILITY BETTER THAN ACCURACY



A TWO-DIMENSIONAL DEPICTION OF TOOL TIP POSITIONS OF ADJACENT INCREMENTS, TRAINED & REPEATED (EXAGGERATED)

- Spatial resolution describes the smallest increment of motion at the tool tip that the robot can control.
- Accuracy relates the robot's spatial-resolution-defined positional ability (including mechanical inaccuracies) to an arbitrary fixed-target position.
- Repeatability describes the positional error of the tool tip when it is automatically returned to a position previously taught.

SUMMARY

1. "ADVANCED FEEDBACK CONTROL" OF ROBOT MANIPULATORS UTILIZES BOTH <u>INTERNAL</u> AND <u>EXTERNAL</u> <u>STATE</u> INFORMATION, AND LEADS TO THE IMPLEMENTATION OF DECISION NETS.

2. INTERNAL STATE INFORMATION IS OBTAINED FROM

   - KINEMATIC AND DYNAMIC MODELS OF MANIPULATORS
   - INTERNAL STATE (JOINT POSITION, VELOCITY, ACCELERATION/FORCE) SENSORS
   - IDENTIFICATION OR ESTIMATION TECHNIQUES,

   LEADING TO AN <u>ADAPTIVE CONTROL</u> COPING WITH VARIATIONS IN LOAD AND TASK CONDITIONS.

3. EXTERNAL STATE INFORMATION IS OBTAINED FROM VISUAL AND NON-VISUAL (PROXIMITY, TOUCH AND FORCE-TORQUE) SENSORS PROVIDING GUIDANCE INFORMATION TO THE CONTROL SYSTEM RELATIVE TO A GIVEN TASK, LEADING TO A <u>"SMART" CONTROL</u>.

4. IMPLEMENTATION OF ADAPTIVE AND "SMART" CONTROL OF ROBOT MANIPULATORS REQUIRES COMPUTERS SINCE ROBOTS ARE COMPLEX SYSTEMS PERFORMING COMPLEX TASKS.

5. ADVANCED AUTOMATION OF ROBOT MANIPULATORS SHOULD ALSO CONSIDER THE ROBOTS' WORK ORGANIZATION WITHIN VARIOUS "WORK CELL" SYSTEMS.

APPENDIX

AN INTERACTIVE MANIPULATOR CONTROL SYSTEM

AN INTERACTIVE MANIPULATOR CONTROL SYSTEM

A. K. Bejczy
Member of Technical Staff
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

M. Vušković
Senior Research Associate
Institute for Technoeconomic Systems
Department of Industrial & Systems Engineering
University of Southern California
Los Angeles, California 90007

PREPRINT

Proceedings of the
Second International Symposium on
Mini & Microcomputers in Control

Fort Lauderdale, Florida
December 10-11, 1979

APPENDIX

THIS PAGE LEFT BLANK INTENTIONALLY

AN INTERACTIVE MANIPULATOR CONTROL SYSTEM

A. K. Bejczy
Member of Technical Staff
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

M. I. Vušković
Senior Research Associate
Institute for Technoeconomic Systems
Department of Industrial and Systems Engineering
University of Southern California
Los Angeles, California 90007

ABSTRACT

A structured approach to the design and implementation of interactive sensor referenced manipulator control systems is proposed. The approach separates the algorithms that perform the manipulator task controls from the local manipulator servo control. According to this, manipulator control is being considered as the problem of execution of an action network. Any manipulator task is treated as an arrangement of interconnected actions grouped into three categories: primitive, composite and complex actions. The software implementation of all three action categories is discussed. The concept of cooperative processes and monitors has been adopted as a powerful method of software organization. The control system is designed and demonstrated for a class of tasks that include searching, following, grasping, and stopping of slowly moving heavy objects so that operator and computer share the control. The demonstration hardware is the six-degree-of-freedom CURV manipulator integrated with proximity and force-torque sensors at the JPL teleoperator laboratory.

Frequent Abbreviations

MCS: Manipulator Control System

CACS: CURV Arm Control System

EE: End Effector

JCS: Joint Coordinate System

WCS: World Coordinate System

TCS: Tracking (Plane) Coordinate System

HCS: Hand Coordinate System

TP: Target Plane

APG: Action Precedence Graph

1. INTRODUCTION

A pilot control system has been developed for a six-degree-of-freedom manipulator equipped with proximity and force-torque sensors in the JPL teleoperator laboratory. The control system, implemented in a dedicated minicomputer, allows an interactive manual and automatic control of the manipulator. In this paper, "interactive control" signifies a hybrid control capability whereby some motions of the manipulator in the work space are under manual control while the remaining motions are under automatic computer control referenced to proximity and force-torque sensor data. "Motion in work space" signifies the three translational and three rotational motions of the end effector. The operator decides which work space motion is in manual or automatic control. In extreme cases all control can be fully manual or fully automatic. The operator's decisions are communicated to the control computer through appropriate function (logic) switches. As a consequence, the operator has a dual (analog and logic) communication with the interactive control system.

The conceptual and implementational intricacies of such an interactive Manipulator Control System (MCS) can best be appreciated by noting that each motion of the end effector in the work space usually requires the coordinated control of several manipulator joints. In general, therefore, control reference inputs to a manipulator joint will be computed from commands originating simultaneously from manual (analog) sources and from computer algorithms referenced to proximity or force-torque sensor data. The computation of control reference inputs addressed to the individual manipulator joints will depend on the setting of the function (logic) switches.

A conventional system control concept would consider the interactive manipulator control as a task planning and decomposition problem. (See, e.g., Refs. 1 - 6.) That concept usually culminates in the formulation of special computer "languages" and procedural or transformation nets for manipulator control, and defines operator interaction with the control system as an essentially off-line and high level programming action. The task planning/decomposition concept is mostly suited to cases when the operator wants to trade control (Ref. 7) with the computer. In contrary to that, the cases considered in this paper are those when the operator wants to share control (Ref. 7) with the computer. In shared control the operator has an essentially on-line interaction with the control system, and the symbolic elements of the on-line interaction should preferably be reduced to simple and discrete function or logic commands.

A usual approach to the design and implementation of sensor-referenced interactive manipulator control would treat the problem as a monolithic multivariable feedback control system where the dynamic and servo characteristics of the manipulator play an essential role in synthesizing the control algorithms. (See, e.g., Refs. 8 - 12.) Such an approach, however, would overshadow the problem of synthesizing the control for a multitude of manipulator tasks. It is noted that the implementation of manipulator task controls are mostly based on discrete events and logic decisions which cannot easily be considered within the frame of the classical continuous or sampled data feedback control systems. In this paper, therefore, another approach is proposed. First, the problem of manipulator servo control is separated from the task control algorithms. I.e., the manipulator together with its servo controllers are considered as a device external to the control computer which interactively controls the execution of a variety of manipulator tasks. Furthermore, the manipulator control tasks are regarded here as a set of interconnected actions which can be more or less complex and which are executed sequentially or simultaneously.

In order to synthesize the control of interconnected actions, three action categories, primitive, composite and complex actions have been introduced. The primitive actions include elementary motions such as one-step shifts of the mechanical hand in a given coordinate system and single increments of spacing or closing the end effector. Composite actions are composed of several primitive actions which are executed sequentially. Examples are: alignment of the end effector to a given surface, controlling the distance of the end effector from a given surface or object, centering on an object, etc. Complex actions consist of composite actions that are executed sequentially and/or in parallel. Examples are: searching for an object, following a moving object, capturing an object, transfering an object from one place to another, etc. The paper presents a separate control software implementation methodology for each of the three action categories.

Due to the inherent parallelism of manipulator control actions in real-time applications, the MCS software outlined in this paper has been designed as a concurrent programming system. The concept of monitors, introduced by Brinch Hansen (Refs. 13-14) and Hoare (Ref. 15), has been employed as the main structuring tool for real-time control software development.

For demonstration and evaluation, the real-time software has been coded for and applied to the CURV Arm Control System (CACS) in the JPL teleoperator laboratory. The first phase of the CACS demonstration and performance evaluation project is limited to relatively simple tasks: interactive searching, following, grasping, and stopping of slowly moving heavy objects using data from proximity and force-torque sensors integrated with the JPL CURV arm. The initial design of the CACS software based on a structured approach employing the monitor concept in its simplest form is given in Ref. 16.

The CACS software is an exploratory development aimed to identify problem areas and to experiment with alternative techniques and approaches rather than to define a final design solution for interactive and sensor referenced manipulator control.

2. MANIPULATOR CONTROL SYSTEM DESCRIPTION

2.1 General Configuration of the MCS

In general the MCS can be regarded as an interface between the human operator and the manipulated object. This interface supports interactive execution of a given class of tasks related to a given class of objects and their environment. As shown in Fig. 1., the components of the MCS are: manipulator, information display devices, command interface, and control computer.

The manipulator itself is a feedback control system, usually implemented in hardware containing a multilinkage mechanism, end-effector (EE), actuators, local feedback controllers, sensors, sensor support electronics, A/D and D/A converters.

Information displays convert the video, sensory and alarm data into forms convenient for the human operator in order to facilitate the interactive control process. Video information is displayed by appropriate monitors. Information from proximity, force-torque and other sensors can be displayed by computer driven graphic displays, using context-dependent formats suited to a particular class of tasks. Alarm information can be displayed by labeled light mosaic, sound signals, etc.

The command interface is a device that convert operator commands to computer readable form. The commands can be in the form of a command language with a given syntax adjusted to a particular class of tasks. Commands can be issued manually by TTY or orally by a voice command system. The operator commands can also be generated in discrete form by switches and pushbuttons, and in analog form by joysticks or potentiometers.

The control computer is the central component of the MCS that generates set-points for manipulator joints on the basis of operator commands and sensory feedback data processed by the MCS software. The MCS software performs also parameter management.

There are two groups of MCS parameters: the system and the task related parameters. The first group contains all parameters that describe the manipulator hardware and the constants used in the control algorithms. The second group of parameters is related to the manipulator environment and to the particular task to be performed. Both parameters are entered via the computer console; the first group during system installation, the second group just before task execution.

2.2 Specific Configuration

The specific interactive control system has been implemented for the JPL/CURV linkage arm. The control system is labeled CACS (Curv Arm Control System). The arm has seven degrees of freedom: three for the arm, three for wrist and one for EE. The mechanical details of the manipulator are given in Ref. 17. The EE is a parallel jaw device with proximity sensors integrated with it. The hydraulic actuators, servo controllers and sensor supporting electronics are described in Ref. 18. The complete spatial position of the manipulator is defined by the following seven joint variables:

θ_1 - arm azimuth

θ_2 - arm elevation

θ_3 - arm extension

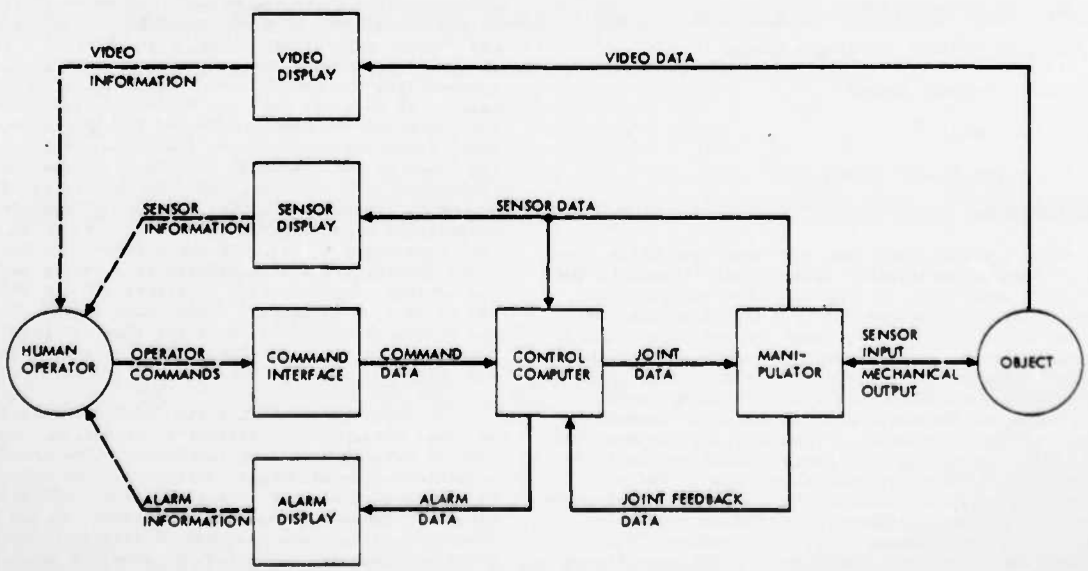


Fig. 1. General Configuration of MCS

θ_4 - wrist yaw angle

θ_5 - wrist pitch angle

θ_6 - wrist roll angle

θ_7 - jaw opening.

The manipulator is also equipped with force-torque sensors. (See Ref. 17.) Other types of sensors have also been developed, for example touch and slippage sensors (see Ref. 19), but they are not used in the first phase of the CACS project.

Proximity sensors, four of them, are integrated with the two jaws of the EE, one pair pointing in forward direction, the other pair pointing perpendicular to the first one, downward the EE. The sensing range of the proximity sensors is about three inches. The force-torque sensor is mounted to the base of the EE. The sensor measures the three perpendicular forces and the three perpendicular torques referenced to the base of the EE. The dynamic range of the sensor is 0.5 to 300 N.

The information display system contains TV monitors and a microprocessor-driven color graphic display. (Ref. 20.) The alarm display is temporarily implemented on the control computer console.

The analog commands are generated by two 3-dimensional joysticks, and by a potentiometer. The discrete operator commands are generated by a set of switches, all incorporated into a multipurpose device called Universal Control Panel.

The control computer is an Interdata M70 (or 8/16E) minicomputer (Ref. 18).

3. MANIPULATOR ACTIONS

Accomplishment of a given manipulation task can be considered as the execution of a set of actions invoked explicitly by operator commands, or implicitly by control algorithms that are implemented in the MCS software. According to their complexity, the actions can be separated into three groups: complex, composite and primitive actions. These groups will now be discussed separately.

3.1 Complex actions

Complex actions will be defined as elements of a manipulator task that are sufficiently general and have unique physical and logical characteristics. In addition, it will be assumed that these actions can only be initiated by the operator. Examples of complex actions are:

- searching for object
- following the moving object
- grasping and stopping the object
- transferring the object
- mapping the object.

Searching for an object means to bring the EE to the vicinity of the object measured by proximity sensors. The control of this action can be performed by the operator manually, or it can be performed automatically using a search algorithm the parameters of which are specified by the operator. Also, the search can be unconstrained in space, or constrained to a fixed surface.

Following a moving object means to move the EE so that a constant distance is being kept between the EE and the object. This action has also manual and automatic, unconstrained and constrained options. In addition, the object following action can be referenced to the front or to the lower proximity sensors.

Grasping and stopping an object must be performed so that once the object is grasped the motion of the EE is referenced to the force-torque sensors assuring a gentle, "soft" stop.

Transferring an object means to change the location and orientation of the object in the work space. This action can be a part of an assembly/disassembly task.

The goal of object mapping is to create a topographic profile of a partially unknown object or of a terrain in the work space. Mapping is based on proximity sensor measurements.

Using the complex actions listed above a large number of manipulator tasks can be accomplished, including simple assembly and disassembly operations, that are assisted by the operator in an interactive manner. The list could also be extended by adding programmed actions to the control "menu." These actions would be performed by executing fixed stored programs loaded from the MCS library. Programmed fixed actions are not considered in this paper.

3.2 Composite actions

In order to perform a complex action the control system will enforce the execution of a set of appropriate composite actions that are less complex. Execution of composite actions can be in series, in parallel, or combined, depending on the specific complex action commanded by the operator. Examples of composite actions are:

- manual control
- roll alignment
- pitch alignment
- yaw alignment
- elevation control
- distance control
- speed identification
- centering
- grasping
- tracking
- stopping.

Using manual control the operator controls the overall motion of the manipulator, affecting directly or indirectly all joint coordinates via analog commands. The manual commands are resolved into appropriate manipulator joint commands through the MCS software. Manual control can be unconstrained or constrained in space. The coordinate system corresponding to analog commands can be chosen by the operator. Different coordinate systems that can be used will be discussed in the next section. Manual control can have two modes: position and rate control mode. In position mode the manipulator control system executes motion increments defined by analog commands in the sense of changing positions of the EE. In rate mode, the manipulator control system will interpret the analog commands as reference inputs for the directional speed of the manipulator EE in a given coordinate system.

Alignment operations are referred to the equalization of the lengths of proximity sensor beams relative to the alignment surface. The alignment surface can be an object or a tracking surface. In roll alignment, the lower proximity sensors must indicate equal lengths. In pitch alignment the lower proximity sensors must have equal length in several successive longitudinal positions of the EE. In yaw alignment, the front proximity sensor beams must indicate equal lengths relative to the alignment surface. The condition for starting an alignment action is to bring the EE in proximity of the alignment surface. Alignment operations can be performed relative to static or dynamic alignment surfaces. A static surface has a fixed position and shape in the work space. A dynamic surface is slowly moving or changing its profile in the work space. In all alignment operations it is assumed that the alignment surface is sufficiently smooth so that it can be approximated by a tangent plane in the vicinity of the footprints of the proximity sensor beams on the surface.

Elevation and distance control are also referenced to proximity sensors. In elevation control the beam lengths of the lower proximity sensors must be maintained constant and equal to a given value. In distance control the beam lengths of the front proximity sensors are maintained constant and equal to a given value. Elevation and distance control can also be referenced to static or dynamic surfaces.

In speed identification the speed of the EE must be equalized to the speed of the moving object or moving tracking surface. Speed identification must be accomplished before starting an alignment, elevation or distance control operation.

Centering means to adjust the opening and the lateral position of the EE relative to a given object before grasp. Centering can be performed on both static and dynamic objects. In the case of dynamic objects, centering also contains a dynamic alignment operation of the EE relative to the object. It means that the EE must get in the tail position to the object so that the motion of the EE is parallel to the longitudinal axis of the EE. This is the best grasping position.

Grasping an object consists of a short programmed sequence of EE operations, with the final result of having and contacting the object within the claws of the EE. Once this occurred, the motion reference is changed from proximity to force-torque sensors.

Tracking an object means a joint motion of the EE and the grasped object so that the manipulator automatically complies with the forces of the object exerted on the EE. This action is force-torque sensor referenced in the sense that the reaction forces exerted by the object on the EE must be maintained at a specified minimum level.

Stopping an object means to stop the motion of the manipulator gradually, keeping the reaction forces exerted by the object on the EE and measured by the force-torque sensor below some value specified by the operator.

The given list of composite actions is not complete. It can be extended by other actions, for example, edge tracking, scanning, etc.

3.3 Primitive actions

Completion of a complex action is accomplished by the sequential execution of a series of motion increments of the manipulator. Execution of a motion increment will be called a primitive action of the manipulator. In this work two groups of primitive actions are considered:

- unconditional increments (shifts)
- conditional increments (shifts)

Both groups have two options: position and rate control mode. Primitive actions can be referred to four different coordinate systems: joint coordinate system (JCS), world coordinate system (WCS), tracking plane coordinate system (TCS) and hand coordinate system (HCS). The four reference systems are shown in Fig. 2.

The point in JCS is defined by the vector $\underline{\theta} = (\theta_1, \theta_2, \ldots, \theta_N)$, where θ_i are manipulator joint variables already described in subsection 2.2. It is noted that the θ_i positions of the manipulator are measured by potentiometers at each joint.

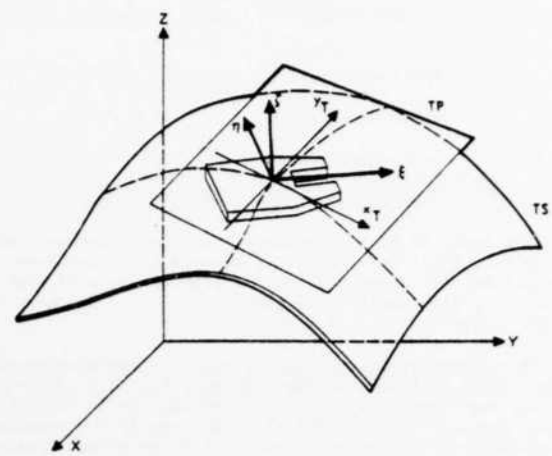


Fig. 2. Representation of Coordinate Systems WCS, TCS, and HCS

WCS is a Cartesian coordinate system fixed to the manipulator environment, i.e. to the world in which the manipulator is acting. In this work the origin of WCS is fixed to the base of the manipulator. The point in WCS is a vector $\underline{q}_W = (\underline{x}, \underline{\alpha}, \omega)$, where $\underline{x} = (x, y, z)$ is position of the wrist of the manipulator, $\underline{\alpha} = (\alpha, \beta, \gamma)$ is the angular position of the hand and ω is the opening of the EE.

TCS represents a two dimensional space defined by the fixed plane TP. The orientation of this plane in world space is defined by the vector $\underline{\alpha} = \underline{\alpha}_{TP}$. Because only motion increments are considered here, the origin of that plane can be ignored. When the EE lies in a plane parallel to the tangent plane TP of the tracking surface TS, i.e. if its roll and pitch axes are aligned with the TP, then the position of the EE is defined by $\underline{q}_T = (\underline{x}_T, \alpha_T, \omega_T)$, where $\underline{x}_T = (x_T, y_T)$ is a point in the TP.

HCS is also a Cartesian coordinate system defined by three orthogonal axes fixed to the EE. The ξ-axis is the longitudinal axis of the EE (it is the roll axis of the EE) and defines the "forward-backward" direction; the η-axis is the lateral axis of the EE and defines the "left-right" direction; and the ζ-axis is the orthogonal complement of the ξ and η axes and defines the "up-down" direction. A point in the HCS is defined by a vector $\underline{q}_H = (\underline{\xi}, \underline{\alpha}_H, \omega_H)$, where $\underline{\xi} = (\xi, \eta, \zeta)$ and $\underline{\alpha}_H = (\alpha_H, \beta_H, \gamma_H)$. The HCS plays an important role in almost all composite actions referenced to proximity sensor measurements.

The unconditional shifts can be defined as increments in the four coordinate systems described above:

$$\begin{aligned} \Delta\underline{q}_J &= \Delta\underline{\theta} && \text{in JCS} \\ \Delta\underline{q}_W &= (\Delta\underline{x}, \Delta\underline{\alpha}, \Delta\omega) && \text{in WCS} \\ \Delta\underline{q}_T &= (\Delta\underline{x}_T, \Delta\alpha, \Delta\omega) && \text{in TCS} \\ \Delta\underline{q}_H &= (\Delta\underline{\xi}, \Delta\underline{\alpha}, \Delta\omega) && \text{in HCS} \end{aligned} \tag{1}$$

where all components of the vectors $\underline{q}_j$ ($j = W,T,H$) can be changed independently. As indicated in expressions (1), the increments of hand angles and the EE opening are invariant in all Cartesian coordinate systems.

Conditional shifts imply conditions that introduce some interdependence between the components of $\Delta\underline{q}_j$. Here we only consider conditional shifts in the HCS, with conditions related to the proximity sensor beams. Four examples are shown in Fig. 3. First a conditional roll increment where the reflection point of the lower-left

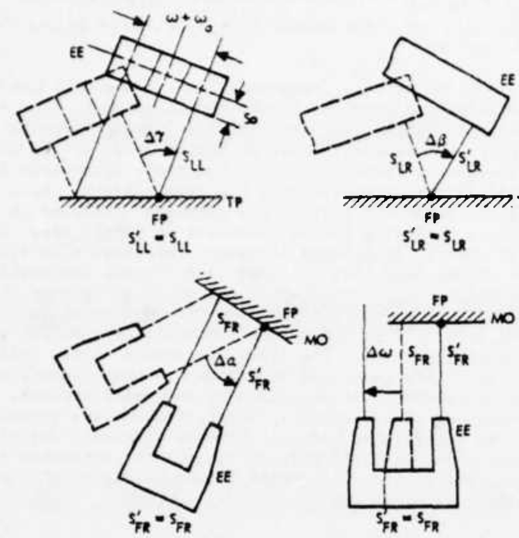


Fig. 3. Conditional Shifts

proximity sensor is kept fixed. As shown, the EE changes its roll angle $\Delta\gamma$ without moving the reflection point of the lower left sensor, and keeping the length of that sensor beam unchanged. Similarly, the conditional pitch and yaw increments are performed with fixed reflection points of the lower-right and front-right proximity sensor beams, respectively. The last example shows a conditional EE opening where the position of the right finger of the EE is kept fixed.

Position and rate control modes are options that can be selected by the operator or by the control algorithms. Position mode corresponds to the regular execution of an increment $\Delta \underline{q}_j$. In the rate mode the given increment $\Delta \underline{q}_j$ will be executed in a repetitive manner in order to generate the speed of the EE that is proportional to the increment $\underline{v} = \Delta \underline{q}/\Delta t$, where Δt is the time interval between two consecutive increments.

Primitive actions are invoked automatically by the control algorithms which implement the composite actions.

Implementation of primitive, composite and complex actions are discussed in the subsequent three sections.

4. IMPLEMENTATION OF PRIMITIVE ACTIONS

4.1 Coordinate transformations

The execution of a primitive action means to move the manipulator from its current position $\underline{\theta}_o$ in JCS to a new position

$$\underline{\theta} = \underline{\theta}_o + \Delta\underline{\theta} \quad , \tag{2}$$

that corresponds to a desired increment $\Delta \underline{q}_j$ defined in a selected coordinate system j. That is, for a given $\Delta \underline{q}_j$ and $\underline{\theta}_o$ the following coordinate transformation must be performed:

$$\Delta\underline{\theta} = \underline{\Phi}_j\,(\Delta \underline{q}_j,\ \underline{\theta}_o) \qquad j = W,T,H. \tag{3}$$

For the WCS one has to consider the equations of manipulator geometry in the general form:

$$\underline{q}_W = \underline{F}(\underline{\theta}), \tag{4}$$

where $\underline{F}$ is a nonlinear analytic vector function that maps a point in JCS onto a point in WCS. If the manipulator is nonredundant, this mapping is unique with a unique inversion $\underline{F}^{-1}\,(\underline{q}_W)$. Combining (2), (3) and (4) we have:

$$\underline{\Phi}_W\,(\Delta \underline{q}_W,\ \underline{\theta}_o) = \underline{F}^{-1}\,(\Delta \underline{q}_W + \underline{F}\,(\underline{\theta}_o)) - \underline{\theta}_o \quad . \tag{5}$$

It should be noted that the calculation of $\Delta\underline{\theta}$ requires the solution of two sets of manipulator equations, $\underline{F}$ and its inverse $\underline{F}^{-1}$. This is so because only the positions in the JCS are measurable.

In the case of TCS, the JCS/WCS mappings are formally as before

$$\underline{\Phi}_T\,(\Delta \underline{q}_T,\ \underline{\theta}_o) = \underline{\Phi}_W\,(\Delta \underline{q}'_W,\ \underline{\theta}_o) \quad , \tag{6}$$

where $\Delta \underline{q}'_W = (\Delta \underline{x}',\ \Delta\alpha,\ 0,\ 0,\ \Delta\omega)$ is an intermediate increment defined by a simple coordinate transformation

$$\Delta \underline{x}' = \underline{R}(\underline{\alpha}_{TP}) \begin{bmatrix} \Delta x_T \\ \cdots \\ 0 \end{bmatrix} \quad , \tag{7}$$

with $\underline{R}$ being a 3×3 rotation matrix that corresponds to the angular position of the tracking plane TP in the WCS.

Similarly, for increments HCS we can write:

$$\underline{\Phi}_H(\Delta \underline{q}_H,\ \underline{\theta}_o) = \underline{\Phi}_W(\Delta \underline{q}''_W,\ \underline{\theta}_o) \quad , \tag{8}$$

where $\Delta \underline{q}''_W = (\Delta \underline{x}'',\ \Delta \underline{o},\ \Delta\omega)$ and

$$\Delta \underline{x}'' = \underline{R}(\underline{\alpha}_o)\,\Delta\underline{\xi} \quad . \tag{9}$$

Vector $\underline{\alpha}_o$ in (9) corresponds to the temporary orientation of the hand in the WCS. As seen, the transformation from the HCS to the WCS is based on a temporary hand orientation which is measured indirectly via the JCS. Therefore, the current value of $\underline{o}_o$ must be calculated

$$\underline{\alpha}_o = \underline{\phi}(\underline{\theta}_o) \quad , \tag{10}$$

where $\underline{\phi}$ represents a subset of the manipulator geometry equations (4).

Conditional shifts are more complex. To give an idea of this, consider only the first and last examples shown in Fig. 3. It can be shown that for a given $\Delta\gamma$, two additional shifts must be executed in the HCS in order to keep the reflection point of the lower-left proximity sensor fixed:

$$\begin{aligned} \Delta\eta &= \frac{\omega + \omega_o}{2}\,(1 - \cos\Delta\gamma) + (S + S_o)\,\sin\Delta\gamma \\ \Delta\zeta &= \frac{\omega + \omega_o}{2}\,\sin\Delta\gamma - (S + S_o)\,(1 - \cos\Delta\gamma) \end{aligned} \tag{11}$$

where $S = S_{LL}$, ω is the EE opening, S_o and ω_o are fixed parameters of the manipulator. Similar equations hold for conditional pitch and jaw increments. The conditional EE operation is much simpler and require only lateral corrections:

$$\Delta\eta = \begin{cases} \dfrac{\Delta\omega}{2} & \text{if right finger is fixed} \\[2ex] -\dfrac{\Delta\omega}{2} & \text{if left finger is fixed.} \end{cases} \tag{12}$$

4.2 Linearization

As seen, all coordinate transformations are based on the function $\underline{\Phi}_W$. This can create a heavy computational burden. Therefore, the linearization of $\underline{\Phi}_W$ is introduced.

It is easy to show that for sufficiently small $\Delta \underline{q}_W$, equations (3) and (5) can be replaced by:

$$\Delta\underline{\theta} \cong \underline{\Phi}(\underline{\theta}_o)\,\Delta\underline{q}_W \quad , \qquad (13)$$

Where $\underline{\Phi}(\underline{\theta}_o)$ is the inverse Jacobian of $\underline{F}(\underline{\theta}_o)$:

$$\underline{\Phi}(\underline{\theta}_o) = \left[\frac{\partial \underline{F}}{\partial \underline{\theta}}\right]^{-1}_{\underline{\theta}_o} \quad . \qquad (14)$$

Linearization simplifies the computations considerably, but introduces cummulative errors. Fortunately, as seen later, the primitive actions are imbedded into iterative feedback loops so that cummulative error cannot occur.

4.3 Decoupling the joint coordinates

The complexity of the mapping (4) depends on the geometric construction of the manipulator. If the manipulator geometry provides only a loose kinematic coupling between arm, wrist and EE coordinates, the joint vector $\underline{\theta}$ can be considered as a composition of three vectors $\underline{\theta}_A$, $\underline{\theta}_H$ and $\underline{\theta}_E$. The three vectors represent the arm, the wrist and the EE, respectively, and their mutual dependence is reduced to a simple addition of angles. Due to the double parallelogram nature of the JPL CURV arm, the oriention dependence between $\underline{\theta}_A$ and $\underline{\theta}_H$ only requires a simple correcrion of θ_4 for every change in θ_1. As a consequence, the 7×7 matrix $\underline{\Phi}(\underline{\theta}_o)$ can be reduced to two 3×3 matrices $\underline{\Phi}_A(\underline{\theta}_o)$, and $\underline{\Phi}_H(\underline{\theta}_o)$, and the equation (13) becomes:

$$\begin{aligned} \Delta\underline{\theta}_A &= \underline{\Phi}_A(\underline{\theta}_o)\Delta\underline{x} \quad , \\ \Delta\underline{\theta}_H &= \underline{\Phi}_H(\underline{\theta}_o)\Delta\underline{\alpha} - \Delta\underline{\theta}_{corr} \quad , \qquad (15) \\ \Delta\underline{\theta}_E &= k_E\Delta\omega \quad , \end{aligned}$$

where $\underline{\theta}_A = (\theta_1,\theta_2,\theta_3)$, $\underline{\theta}_H = (\theta_4,\theta_5,\theta_6)$, $\underline{\theta}_E = \theta_7$, $\Delta\underline{\theta}_{corr} = (-\Delta\theta_1,0,0)$, and k_E is a scaling factor that converts EE opening into a corresponding joint angle. The matrix $\underline{\Phi}_A(\underline{\theta}_o)$ for the JPL CURV arm has a relatively simple form (Ref. 16). The matrix $\underline{\Phi}_H(\underline{\theta}_o)$ is a unit matrix for the JPL CURV arm.

The coordinate transformations in the CACS can be summarized as shown by a data flow diagram in Fig. 4. This diagram contains four kinds of graphic symbols: arrows representing vector or scalar data; square boxes representing computation of a function; arithmetic symbols and oval boxes representing the data buffers. Data buffers play an important role in the approach described in this paper. They will not be discussed in detail here, and will only be considered as memory locations allocated to the shared data that are used in different computations. The part of Fig. 4 fenced by dotted lines represents the coordinate transformations from TCS and HCS to the WCS. Therefore, WCS is the common coordinate system for all primitive actions. The input increments on the left side of Fig. 4 are generated by various control algorithms which implement the execution of composite actions. The results of the coordinate transformations are accumulated in the world-space buffer. This is a transit buffer in the system, with the purpose to collect increments from different control algorithms before the final WCS to JCS mapping is performed.

4.4 Superposition of the control modes

As discussed previously, some composed actions are related to moving objects or moving tracking surfaces. This means that positional shifts, generated by control algorithms, are relative to the moving object or tracking surface. As an example, Fig. 5 illustrates the case where the EE is following an object. The speed of object $\underline{v}_o$ and the speed of the EE $\underline{v}_E$ are equal, but the position of the EE relative to the object must be corrected by a lateral shift $-\Delta n$. This shift must be added to the corresponding rate increment $\underline{v}_E\Delta t$.

In this work such additions are handled within the frame of the WCS. Therefore, two parallel versions of world-space buffers are introduced, the position and the rate world-space buffer. As shown in Fig. 6, the position buffer collects all positional shifts $\Delta\underline{q}_i$ generated by the

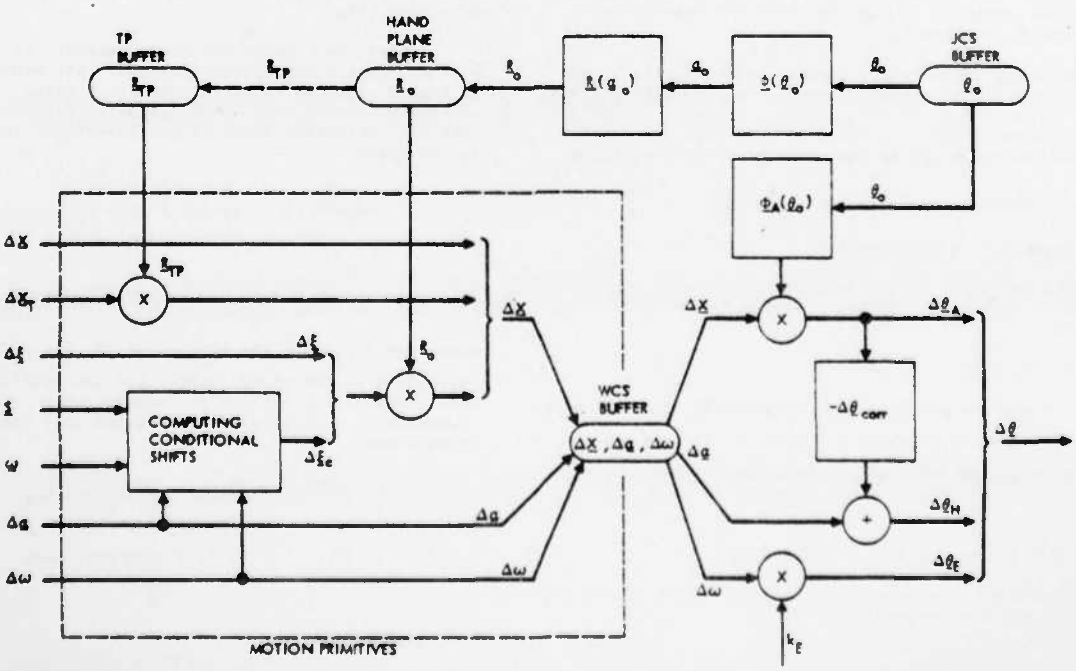


Fig. 4. Coordinate Transformations in CACS

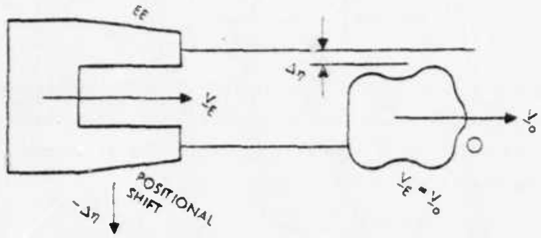


Fig. 5. Example of Combined Motion

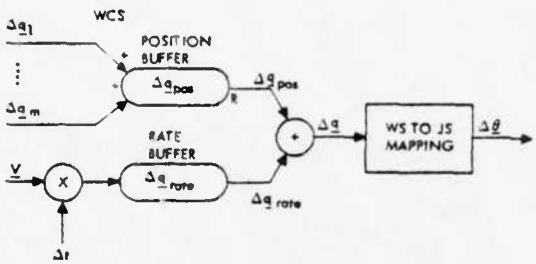


Fig. 6. Superposition of Control Modes

control algorithms. Whenever an increment is sent to the buffer it is added to the actual content of the buffer. This kind of buffer access is marked by an addition symbol on the diagram. In the time of WCS to JCS mapping, the accumulated increment $\Delta\underline{q}_{pos}$, that represents the position contribution to the total increment $\Delta\underline{q}$, must be transfered to the mapping routines. The transfer procedure must reset the buffer after receiving its content. Therefore, the buffer will be empty by the time when the next increment is executed, provided that it was not filled again by the control algorithms. Buffer reseting is marked by "R" on the diagram. The rate buffer containes the rate increment $\Delta\underline{q}_{rate}$ that corresponds to spad $\underline{v}$, and there is no addition property involved in the corresponding accessing procedures. Also, the procedure that reads the buffer does not reset the buffer. Consequently the rate contribution $\Delta\underline{q}_{rate}$ exists in every iteration cycle of the motion increment execution until its content is nullified by the control algorithms.

4.5 Motion primitives

The coordinate tranaformations contained in the dotted area of the Fig. 4. are implemented in software as a set of subroutines that will be called "motion primitives.' The list of motion primitives employed in this work is given in Table 1.

Motion primitives are subroutines called in the programs that implement the execution of composite actions. All motion primitives have only input parameters. These parameters define motion increments in a given coordinate system, the control mode and the additional condition data in the case of conditional shifts. The result of the coordinate transformations is placed into the world-space buffer, position or rate version, depending on the control mode identifier.

5. IMPLEMENTATION OF COMPOSITE ACTIONS

As discussed before, execution of a composite action consists of s sequential execution of primitive actions. Therefore, the problem of implementing composite actions is the problem of generating a sequence of motion increments.

TABLE 1. MOTION PRIMITIVES USED IN CACS PROJECT

Procedure	Description
SHIFT($\Delta\underline{x}$,CM)	Move EE translatory for $\Delta\underline{x}$ in WCS. (Hand angles unchanged.)
YAW($\Delta\alpha$,CM)	Rotate hand for angle $\Delta\alpha$
PITCH($\Delta\beta$,CM)	Rotate hand for angle $\Delta\beta$
ROLL($\Delta\gamma$,CM)	Rotate hand for angle $\Delta\gamma$
MOVEK($\Delta\xi$,CM)	Move EE translatory for $\Delta\xi$ in HCS (forward - backward)
MOVEE($\Delta\eta$,CM)	Move EE translatory for $\Delta\eta$ in HCS (left - right)
MOVEZ($\Delta\zeta$,CM)	Move EE translatory for $\Delta\zeta$ in HCS upward - downward
SLIDE($\Delta\underline{x}_T$,CM)	Move EE translatory for $\Delta\underline{x}_T$ in TCS
CONTR($\Delta\omega$,CM)	Contract the EE for $\Delta\omega$
EXPND($\Delta\omega$,CM)	Expand the EE for $\Delta\omega$
YAWC(FP,S,ω,$\Delta\alpha$,CM)	Rotate hand conditionally for angle $\Delta\alpha$
PITCHC(FP,S,ω,$\Delta\beta$,CM)	Rotate hand conditionally for angle $\Delta\beta$
ROLLC(FP,S,ω,$\Delta\gamma$,CM)	Rotate hand conditionally for angle $\Delta\gamma$
CONTRC(FP,$\Delta\omega$,CM)	Contract the EE conditionally for $\Delta\omega$
EXPNDC(FP,$\Delta\omega$,CM)	Expand the EE conditionally for $\Delta\omega$

Notices: CM is control mode: CM=0 for position, CM=1 for rate. FP is sensor identifier: FP=0 for left, FP=1 for right proximity sensor.

5.1 The controller and its state equation

Generation of a sequence of motion increments $\Delta\underline{q}^{(k)}$ (the coordinate system will be ignored here) can be described by the following iterative equation:

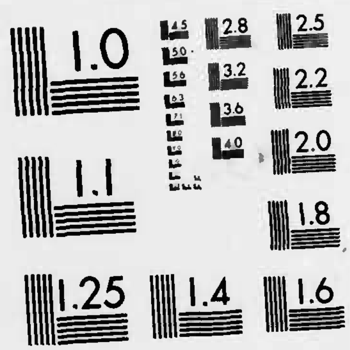
$$\Delta\underline{q}^{(k)} = \underline{f}(\underline{S}^{(k)}, \underline{C}^{(k)}, \underline{\theta}^{(k)}), \quad k=0,1,2,... \qquad (16)$$

where k denotes the iteration cycle, while $\underline{S}^{(k)}$ and $\underline{C}^{(k)}$ are vectors of sensor data and human operator commands taken in the k-th iteration cycle. The value $\underline{S}^{(k)}$ has been measured when the manipulator was in the position $\underline{\theta}^{(k)}$. The vector function $\underline{f}$ represents the control algorithms which generally involve both numeric calculations and logic decisions. These algorithms will be called here "controllers". A controller is allocated to each composite action. (These controllers should not be misinterpreted as local feedback controllers in the manipulator servo hardware. They have nothing in common.)

Because the iterative equation (16) involves only one iteration cycle, $\underline{f}$ is a memoryless controller. Unfortunately, the majority of composite actions are not so simple, and cannot be implemented by memoryless controllers. The control algorithms may require the data from more, say the last M iteration cycles. Thus, the list of arguments of $\underline{f}$ should be extended by $\underline{S}^{(k-j)}$, $\underline{C}^{(k-j)}$, $\underline{\theta}^{(k-j)}$, j = 1,2,..., M-1. A more elegant way to deal with this case is to introduce state variables. State variables provide the

AD-A134 852 TUTORIAL WORKSHOP ON ROBOTICS AND ROBOT CONTROL(U) ARMY TANK-AUTOMOTIVE COMMAND WARREN MI 26 OCT 82 2/4

UNCLASSIFIED F/G 14/2 NL



benefit of the recurrence idea, and considerably simplify the formulation of the control algorithms and the corresponding memory operations.

The state variables are suitably chosen data that are calculated additionally in each iteration cycle and memorized for use in the next iteration cycle. The state variables can be grouped into a state vector. The dimension of the state vector corresponds to the number of iterations contained in the iterative equation. Since the control algorithms contain many logic variables and logic decisions, the state variables for the controller can also be boolean variables. Therefore, two state vectors will be introduced here: a numeric state vector, $\underline{r} = (r_1, r_2, \ldots, r_R)$ whose components r_i are numeric variables, and a logic state vector $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_L)$ whose components ℓ_i are boolean variables. The controller with memory can now be described by the following recurrence relation:

$$
\begin{aligned}
\Delta\underline{q}^{(k)} &= \underline{f}(\underline{r}^{(k)}, \underline{\ell}^{(k)}, \underline{s}^{(k)}, \underline{c}^{(k)}, \underline{\theta}^{(k)}), \\
\underline{r}^{(k+1)} &= \underline{g}(\underline{r}^{(k)}, \underline{\ell}^{(k)}, \underline{s}^{(k)}, \underline{c}^{(k)}, \underline{\theta}^{(k)}), \quad k=1,2,\ldots \\
\underline{\ell}^{(k+1)} &= \underline{h}(\underline{r}^{(k)}, \underline{\ell}^{(k)}, \underline{s}^{(k)}, \underline{c}^{(k)}, \underline{\theta}^{(k)}), \\
\underline{r}^{(0)} &= \underline{r}_o, \\
\underline{\ell}^{(0)} &= \underline{\ell}_o,
\end{aligned}
\tag{17}
$$

where $\underline{r}_o$ and $\underline{\ell}_o$ are initial values of state vectors $\underline{r}$ and $\underline{\ell}$, and $\underline{g}$ and $\underline{h}$ are numeric and boolean vector functions, respectively.

The iterative process (17) is illustrated in Fig. 7. As indicated in this figure a special buffer is allocated to the controller for memorizing the numeric and logic state vectors. The coordinate transformations have already been described in the previous section. The broken arrow at the manipulator input in Fig. 7 indicates the delay between two succesive values of manipulator inputs $\underline{\theta}^{(k)}$ and $\underline{\theta}^{(k+1)}$. This delay will be discussed later.

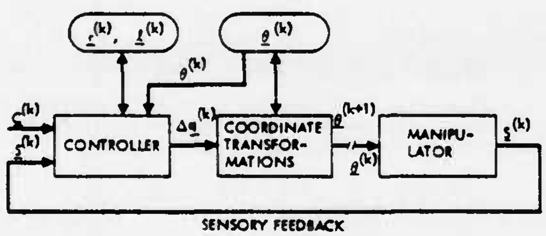


Fig. 7. Iterative Generation of Motion Increments

Two simple examples are considered now: roll and pitch alignment. According to Fig. 3, the roll increment for full roll alignment is given by:

$$
\Delta\gamma^{(k)} = \operatorname{arctg}\left(\frac{S_{LL}^{(k)} - S_{LR}^{(k)}}{\omega^{(k)} + \omega_o}\right) \tag{18}
$$

This equation describes a simple memoryless controller. Pitch alignment can be performed by constant longitudinal test shifts $\Delta\xi_{test}$. This is necessary because the EE has only two lower proximity sensors. (The condition to start pitch alignment is to complete first the roll alignment.)

According to Fig. 3, the pitch increment will be:

$$
\Delta\beta^{(k)} = \operatorname{arctg}\left(\frac{S_L^{(k)} - S_L^{(k-1)}}{\Delta\xi_{test}}\right), \tag{19}
$$

where $S_L^{(k)}$ and $S_L^{(k-1)}$ are lengths of lower proximity sensors (one of them) in two succesive iterations when the test shift $\Delta\xi_{test}$ has been applied. The algorithm that implements the pitch alignment will be:

```
if ℓ = false  then
   begin
      Δξ:=Δξ_test;
      Δβ:=0;
      ℓ:=true;
      r:=S_L
   end
else
   begin
      Δξ:=0;
      Δβ:=(S_L - r)/Δξ_test;
      ℓ:=false
   end
end
```

where the initial conditions are $r_o = 0$, ℓ_o = false.

As seen, the controller uses two state variables, r and ℓ. The first memorizes the preceding value of the proximity sensor beam and the second is a boolean variable that controls alternation of test and correction increments. A more complex example is the centering procedure that requires the use of five logic state variables.

In ideal cases, roll and pitch alignments can be performed in one and two iterations, respectively. However, due to noise in the sensor signals, errors in the manipulator servo and linearization of the manipulator geometry equations, it can be expected that succesful alignment cannot be achieved in the first trial. Therefore, the alignment process must be repeated several times until some termination criteria are satisfied. For practical reasons, therefore, the corrections $\Delta\gamma$ and $\Delta\beta$ should be limited. If the limiting value is sufficiently small, then the function arctg can be linearized. Also, some averaging techniques can be applied to the sensor data. If the general first order filter is used, the state equation of the controller for roll alignment will become:

$$
\begin{aligned}
y^{(k)} &= r^{(k)} + c\Delta S_L^{(k)}, \\
r^{(k+1)} &= ar^{(k)} + b\Delta S_L^{(k)}, \\
\Delta\gamma^{(k)} &= \begin{cases} \Delta\gamma_{max} & \text{if} \quad y^{(k)} > \Delta\gamma_{max} \\ y^{(k)} & \text{if} \quad |y^{(k)}| \le \Delta\gamma_{max} \\ -\Delta\gamma_{max} & \text{if} \quad y^{(k)} < -\Delta\gamma_{max} \end{cases}
\end{aligned}
\tag{20}
$$

where $S_L^{(k)} = S_{LL}^{(k)} - S_{LR}^{(k)}$, $\Delta\gamma_{max}$ is the maximum value of roll increment defined as a MCS parameter, and a, b and c are filtering constants. It is noted that these parameters do not depend on the mechanical characteristics of the manipulator.

5.2 Virtual Manipulator

An importent feature of the iterative process described by equation (17) is that the functions $\underline{f}$, $\underline{g}$ and $\underline{h}$ do not depend on the dynamic charecteristics of the manipulator. The manipulator is here considered as a device externel to the control computer having only two obligetions: to execute the increment $\Delta\underline{\theta}^{(k)}$ within an arbitrary but finite time intervel $\Delta t_M^{(k)}$, and to notify the control computer when the execution is completed. The length of the time interval $\Delta t_M^{(k)}$ depends on the current dynamic performance of the manipulator, on the current position $\underline{\theta}^{(k)}$ of the manipulator, on the length and orientation of the increment $\Delta\underline{\theta}^{(k)}$, and on the load currently being handled by the EE. Therefore, the manipulator execution intervals are very compliceted functiona of many end partly unknown parameters exposed to unexpected chenges. Even for the same $\Delta\underline{\theta}^{(k)}$ the execution time $\Delta t_M^{(k)}$ can be different. However, this is irrelevant to the control algorithms, i.e. to the controller that executes the composite action. The controller is synchronized so that after generating the increment $\Delta\underline{\theta}^{(k)}$ it waits until the manipulator reaches the new position $\underline{\theta}^{(k+1)}$. Then, the controller reada new values of sensor deta $\underline{S}^{(k+1)}$ in order to generate a new increment $\Delta\underline{\theta}^{(k+1)}$. The menipuletor is capable of executing any $\Delta\underline{\theta}^{(k)}$ due to its own local feedback controllers ettached to all manipuletor joints. Because of this autonomous cepability, the manipulator will be called a "virtual manipuletor." The degree of virtualization can be lower or higher, depending on the sophistication and computar support of the local feedback control system. For example, the manipulator can be supported by dediceted microprocessors to improve the performance of the locel feedback control. Also, more sophisticeted algorithms cen be used for the locel feedbeck control.

The concept of a virtual manipuletor divorces the problem of the design and implementetion of interactive manipulator task control elgorithms from the problem of manipuletor local feedback controllers.

The virtuel manipulator will now be defined more precisely. First, the position $\underline{\theta}_o$ of the manipulator is said to be stable if the following inequalities hold:

$$|\theta_{oi} - \theta_{Fi}(t)| \le \varepsilon_{Ji}\ , \quad t > t_o,\ i = 1,2,\ldots,N, \qquad (21)$$

where $\theta_{Fi}(t)$ are ectuel values of manipuletor joints, t_o is the time when observation starts, and ε_{Ji} are tolerances given by the MCS designer. The tolerences must be in accordence with the resolution of joints and sensors, and with the requirements of the perticular manipulator tesk. The virtual manipuletor executes the increment $\Delta\underline{\theta}$ in JCS if it moves from the stable position $\underline{\theta}_o$ to the stable position $\underline{\theta} = \underline{\theta}_o + \Delta\underline{\theta}$ (see Fig. 8).

Now, the following assumption must be satisfied concerning the virtual manipuletor: For given $\underline{\varepsilon}_J$, $\underline{\varepsilon}_J = (\varepsilon_{J1},\ldots,\varepsilon_{JN})$, end for any $\underline{\theta}_o$ and $\underline{\theta}$ belonging to the domain of all edmissible positions of the menipulator, the time Δt_M of execution of the motion increment $\Delta\underline{\theta} = \underline{\theta} - \underline{\theta}_o$ is finite end less than a prescribed velua. The control algorithms and the system synchronization is based on this assumption.

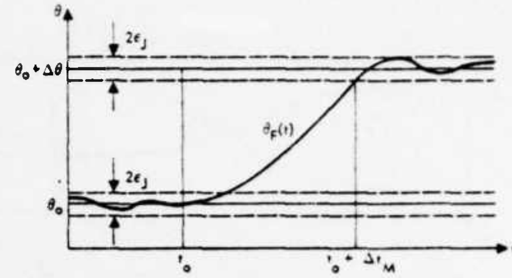


Fig. 8. Execution of Motion Increment

5.3 Controller scheduling

The iterative process shown in Fig. 7 must be organized so that the executions of the controller and the virtual manipulator are performed in the proper sequence. I.e., the controller must wait until the manipulator achieves stable position and sends sensor data. Also, the virtual manipulator must wait for the controller until it generates a new motion increment. This organization will be called scheduling of the controller. There are two basic approaches to controller scheduling: the synchronous and asynchronous approech. Both are illustrated in Fig. 9.

Synchronous scheduling is based on cyclic interrupts generated by the clock. These interrupts initiate reading of the sensor data and execution of the controller. Immedietely after the controller is executed, the new set-point ia sent to the manipulator. When the manipulator executes the set-point, the controller is waiting for a new interrupt signal. It is obvious that for this type of scheduling the execution times of the controller and the manipulator must satisfy the following inequality:

$$\Delta t_C^{(k)} + \Delta t_M^{(k)} \le T_{cyc} \qquad (22)$$

where T_{cyc} is a fixed period of the clock. This period must be chosen to satisfy the worst case, i.e., the maximal possible values of Δt_C and Δt_M. As a consequence, there will be time intervals when both controller and manipulator are idle. These intervels are indicated by the shadowed area in Fig. 9. This results in wasted time and slower execution of the composite action.

With esynchronous scheduling the interrupts are principally generated externally by the manipulator so thet idle intervala do not exist. This scheme requires greater complexity in the scheduling mechanism which increases computer overhead, i.e., the execution time of the controller. Nevertheless, faster execution of composite actions can be achieved by using asynchronous scheduling.

It is noted that the scheduling problem is much more complex than the brief remarks discussed above. To this point the execution of only one controller was considered. As seen leter, some complex ections require simulteneous execution of more than one composite action. Therefore, controller scheduling will involve parallel execution of more than one controller.

6. IMPLEMENTATION OF COMPLEX ACTIONS

Execution of complex actions is enforced by explicit commands issued by the humen operator. In ordec to study the implementetion of complex actions for a given class of menipulator tesks, first the corresponding command structure is considered.

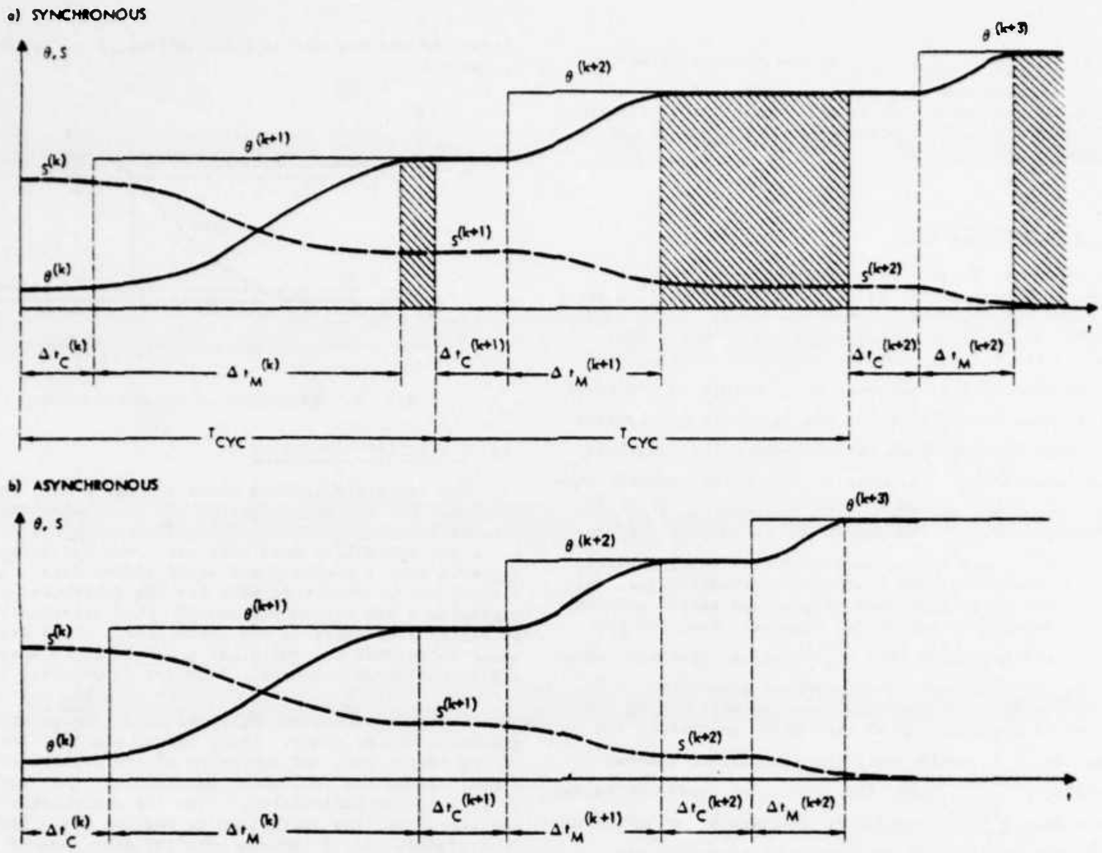


Fig. 9. Two Basic Approaches to Controller Scheduling

6.1 Command structure

Due to the limited repertoir of manipulator tasks anticipated for the initial phase of the CACS project, the corresponding command language is quite simple. It can be defined by the following expressions using the Backus-Naur notation:

⟨command⟩ ::= ⟨command code⟩ ⟨command options⟩
⟨parameter assignments⟩

⟨command code⟩ ::= SEARCH | FOLLOW | GRASP | STOP
| HELP | HOLD | ABORT

⟨command options⟩ ::= ⟨empty⟩ {, ⟨option code⟩}

⟨option code⟩ ::= MANUAL | RATE_MODE | DYNAMIC_OBJECT
| CONSTRAINED_MOTION (23)
| HAND_COORDINATE_SYSTEM

⟨parameter assignments⟩ ::= ⟨empty⟩
{, ⟨parameter assignment⟩}

⟨parameter assignment⟩ ::= ⟨command parameter id⟩
= ⟨parameter value⟩

⟨command parameter id⟩ ::= P1 | P2 | P3 | P4 | P5 | P6 | P7 .

As seen, there are seven basic commands, four of which have already been discussed previously. Command HELP enables the operator to interfere with controllers in order to support manually the automatic operations if it is needed. HOLD command immediately freezes the motion of the manipulator in emergency cases. After abortion of this command the manipulator will continue the action from the point where it has been interrupted. ABORT command is used in conjunction with other command codes. Any command preceded by the parole ABORT will be withdrawn and disregarded until it is issued again. Command options are also discussed before. If an option is not quoted, the corresponding default will be assumed. Defaults are: automatic operation, position control mode, static object, unconstrained motion, and world coordinate system.

The implementation of this command language was motivated by the following requirements:

- The commands must be convenient for the human operator, i.e., the operator should be able to issue them easy and fast. This is very important in the case of an interactive MCS.
- The commands must have parallel capability, i.e., certain commands will be issued simultaneously.
- The response of the command interface must be fast.

The motives quoted above have led to the development of a Universal Control Panel which is an electronic device that incorporates discrete and continuous command signals in one device. The command codes and option codes are implemented by a set of "on-off" switches. A switch has been sllocated to each commsnd code. The control parameter sssignments are implemented by two 3-dimensionsl joysticks and s potentiometer. These analog devices can simultaneously generste seven analog signsls.

The human operator command $\underline{C}$ which sppears in equations (16) snd (17) can now be expressad by three vectors: $\underline{C} = (\underline{c}, \underline{o}, \underline{p})$, where $\underline{c} = (c_1, c_2, \ldots, c_7)$ and $\underline{o} = (o_1, o_2, \ldots, o_5)$ are boolean vactors representing command and option codas, respectively, and $\underline{p} = (p_1, p_2, \ldots, p_7)$ is a vactor of rasl variables assigned to the parameters. The ordering of thase vectors is definad by the expressions (23). For example, bit c_3 of $\underline{c}$ corresponds to command "GRASP", bit o_1 of $\underline{o}$ corresponds to the option code "MANUAL", and the component p_5 of $\underline{p}$ contains the numeric value assigned to paramater P5, which is relsted to the output of the pitch channel of the right joystick. Abortion of a commsnd is done by setting the corrasponding switch in "off" position.

6.2 Event variables

To execute a complex action some conditions must be satisfied. First, the appropriata command must ba issuad. Second, the manipulator must be in s proper stata ralstiva to the objact so that the MCS can successfully proceed with the commanded sction. For example, to stsrt the FOLLOW operetion, the EE must be in tha proximity of tha object; or to start the GRASP oparation tha MCS must complete speed identification of tha object and centering of the EE. Therefore, the control algorithms must in every moment of the tsak execution ba awera of the current stete of tha manipulator relative to tha object and its anvironment. For this purpose special logic variebles, called evant variablee, have been introduced eleewhere (Refs. 20-21). For the CACS projact, the event variables are organized in an evant status scheme shown partially in Teble 2.

The event variablas can be reprasantad in tha vactor form $\underline{e} = (e_1, e_2, \ldots, e_m)$. The components of this vactor ara boolaan veriables thet indicate occuranca of tha correeponding event (when trua), or ebaance of the evant (when felsa).

Tha avent statua vector $\underline{e}$, togethar with the command code vector $\underline{c}$ and the option coda vector $\underline{o}$, provida sufficient information for meking tha dacision whan to start or to tarminata a complex or s composita sction. Also, tha axacution of a composite action will changa the svent status vactor $\underline{a}$. Thus, tha vectors $\underline{c}$, $\underline{o}$, and $\underline{s}$ play an assantial rola in tha coordinstion of both complex actions and ths antire manipulator control task.

6.3 Action precedenca graph

Exacution of s complax action is datarmined by precedence rulee that dafina tha ordsr of execution of tha corrasponding composita sctions. Tha rules slso spacify the conditions that must be satisfiad to etart or to tarminate tha execution of the actions. Thesa rulss can ba expressed grephicslly by diagrams that will ba callad Action Pracedanca Graphs (APG).

An APG example is given in Fig. 10. Tha figura shows four composita actions i, j, k, and ℓ, rapraeentad by circlas. Tha arrows batwaan circlas indicata the transitions from one action to anothsr, and $\tau_{ij}, \tau_{ji}, \ldots, \tau_{\ell j}$ era vsriablas corraaponding to the transition conditions. Theee variablas ara boolaan functions of command code, option code, and avant status vactors:

$$\tau_{mn} = \tau_{mn}(\underline{c}, \underline{o}, \underline{e}) \;, \quad m,n = i,j,k,\ell \;. \qquad (24)$$

TABLE 2. CACS EVENT STATUS SCHEME

Event Variable	Description
e_1/e_2	Front-left/right proximity sensor indicates proximity distsnce
e_3/a_4	Front-left/right proximity sensor indicatas following distance
e_5/e_6	Front-left/right proximity sensor indicates collision distance
e_7/e_8	Lower-left/right proximity sensor indicates proximity distance
e_9/e_{10}	Lower-left/right proximity sensor indicates tracking distance
e_{11}/e_{12}	Lower-laft/right proximity sensor indicates collision distance
e_{13}	EE is roll aligned to the tracking plane
e_{14}	EE.ie pitch alignad to the tracking plane
e_{15}	EE is ysw aligned to the object
e_{16}	Spaad of tha object is identifiad
e_{17}	EE is centerad to tha object
e_{18}	Object is graaped
e_{19}	Object is stoppad
.	.
.	.
.	.

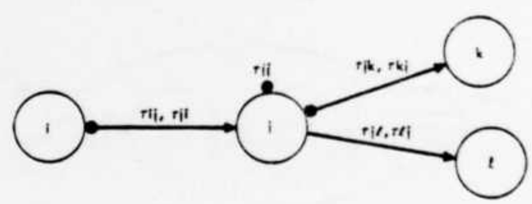


Fig. 10. Exempla of An Actions Precedence Graph

The ordar of indices of the condition variables indicates the diraction of transition when the condition becomee trua.

Two kinds of transitions sre shown in Fig. 10: unblocking and blocking trsnsitions. Unblocking transition occurs whan en action, say j, raalizes the truth of the condition $\tau_{j\ell}$, invokes the execution of the succeeding sction ℓ, and continues its own execution. After that the two actions are in axecution simultaneously. Blocking transition occurs whan an action, say j again, realizee the truth of condition τ_{jk}, invokes axecution of the succaeding action k, and immediately termineces its own execution. This typa of transition is indicetad by a point at the baginning of tha arrow. An action can elso be termineted by itself, without transition to another action. This ie indicated by en arrowlass point on the circle and merked by tha condition variable τ_{jj}. As shown in Fig. 10, tha transitions csn be in a raverse order. The condition variablas for rsvarssd trensitions ere τ_{ji}, τ_{kj} and $\tau_{\ell j}$ in Fig. 10. A raversa transition occurs whan an action, say j, currantly being in axecution, loses the condition τ_{ji} for its active steta and must be terminated, whila its predecessor i must be activated again. In symmetric cases we have $\tau_{ji} = \overline{\tau}_{ij}$, but it is not so in the general case.

The construction of an APG, including the boolean expressions (24), must be done with special care to prevent deadlocks or instability, i.e., to prevent endless reversed transitions between two actions.

The APGs for four complex actione of the CACS are shown in Fig. 11. As an example, the logic expressions for the transition conditions of the first complex action are given below:

$$
\begin{aligned}
\tau_{12} &= c_1 \wedge o_1 \wedge o_4 \wedge (e_7 \vee e_8) \\
\tau_{21} &= \overline{\tau}_{12} \\
\tau_{23} &= \tau_{12} \wedge e_{13} \\
\tau_{32} &= \overline{\tau}_{23} \\
\tau_{34} &= \tau_{23} \wedge e_{14} \qquad (25) \\
\tau_{43} &= \overline{\tau}_{34} \\
\tau_{41} &= \tau_{34} \wedge (e_9 \wedge e_{10}) \\
\tau_{14} &= \tau_{34} \wedge (\overline{e_9 \wedge e_{10}}) \\
\tau_{44} &= \overline{o}_4 \vee e_{19} \; .
\end{aligned}
$$

where e_1, through e_{19} are event variables defined in Table 2, c_1 corresponds to command code for searching, o_1 and o_4 indicate manual constrained options.

Dotted arrows in Fig. 11 indicate ths transitions between complex actions for a particular control task: manual searching, following, grasping, and stopping an object that is slowly moving on a fixed plane. The dottsd arrows connect an APG for the entire control task.

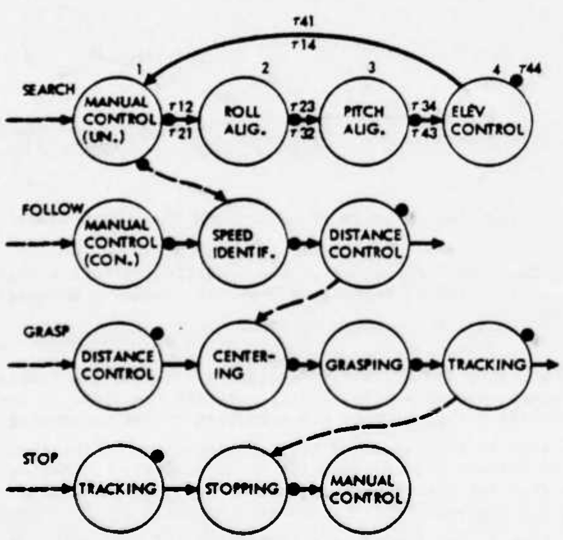


Fig. 11. Action Precedence Graph of the CACS

6.4 Cooperating procssses

The composite actions shown in Figs. 10 and 11 can be implemented in software as sequential programs to be executed on sequential processors. The following questions arise: how to implement an APG which requires parallel execution of several composite actions; how to coordinate action transitions; and how to exchange data among the actions, i.e., among the corresponding sequential programs. The answer to this is the concept of cooperative processes largely employed in the field of operating systems and real-time programming. (See Refs. 13-15, 23-26.) The process will be considered here as an activity of execution of a sequential program on a sequential processor. In general, the process has two states: an active state when it is in execution by the processor, and a blocked state when it is not in execution. Furthermore, the process has a form of an endless cyclic program, where the cycling can be terminated if the process explicitly blocks itself, or if it is blocked indirectly by synchronization mechanisms. The synchronization mechanisms are part of a special set of procedures that support the parallel execution of processes in one or more processors. More about this in the next section.

Each composite action has an associated process. The processes contain essentially two parts: a part where they compute and check the transition conditions and perform the corresponding transition operations, and another part where they execute the sequential program that implements the controller of the composite action.

For a transition operation two primitives are used: AWAKE(i) and BLOCK. The first primitive ie a procedure that activates process "i" quoted in the argument list, and the second procedure puts the calling process into the blocked state. This procedure also initializes the caller controller, i.e., it assigns the initial values to the state vectors $\underline{r} = \underline{r}_o$ and $\underline{l} = \underline{l}_o$, so that the controller will be ready to start again if its process is activated again. To give an idea how this works, consider the APG in Fig. 10. This example is sufficiently general to cover all complex actions of the CACS. The actions i, j, k and ℓ have to be considered now as processes. The process "j" will have the following form:

```
"Process j"
begin
   repeat
      compute transition conditions;
      if τji = true then begin AWAKE(i); BLOCK end
      if τjℓ = true then AWAKE(ℓ);
      if τjk = true then begin AWAKE(k); BLOCK end
      if τjj = true then BLOCK;
      execute controller (one iteration);
   forever
end
```

Processes i, k and ℓ have a similar form.

To this point only processes that perform compoeite actions are considered. But, for full implementation of an APG, it is necessary to communicate with the external world, i.e., with the command interface and the manipulator. These communications can be performed by additional processes that support various I/O functions. These proceeses are executed independently and simultaneously with the action proceeses.

7. SOFTWARE ORGANIZATION

Several software components of the MCS have been discussed in the previous sections: motion primitives (subsection 4.5), controllers (subsection 5.1), two synchrnnization primitives, AWAKE and BLOCK (subsection 6.3), and two kinds of processes, controller processes and I/O processes (subsection 6.4). In this section a new system component will be discussed that plays an important role in the implementation of cooperative processes. The new component is called monitor.

The concept of monitors was introduced by Brinch Hansen (Refs. 13-14) and Hoare (Ref. 15) as a systematic and powerful approach to the design and implementation of real time systems. This concept was later extended to high level languages for concurrent programming (Ref. 25-26). Here the monitors will only be discussed very briefly. More about monitors and their application in the MCS will be given in a companion paper (Ref. 27).

Buffers that are used to save data for later use by the same part of the program, or for simultaneous use by different parts of a program have been discussed in previous sections. Examples are rotation matrices $\underline{R}_o$ and $\underline{R}_{TP}$, WCS and JCS buffers (Fig. 4), and controller buffers for handling state variables $\underline{r}$ and $\underline{\ell}$ (Fig. 7.). There are also other examples, as event variables $\underline{e}$, command data $\underline{C}$, and sensor data $\underline{S}$.

All these data are shared among different processes which can run simultaneously. In such situation, the mutual exclusion of access to shared data must be ensured. In other words, it must be guaranteed that the same shared data buffer can only be accessed by one process at a time. If two processes attempt to access the same buffer at the same time, one must wait until the other completes its operation on the buffer data. It is also important to control the access rights to the shared data, i.e., it must be explicitly declared who has the right to access the shared data and what it can do with these data. The mutual exclusion and the access rights control of shared data buffers is provided by monitors.

In general, a monitor is an abstract software object that contains the following entities: data structure, procedures, initial operations, and access right definition. The data structure contains the shared data as well as some additional administrative data necessary for the functioning of the monitor. Monitor procedures explicitly define all operations which the processes can perform on the shared data. These procedures are called directly from the processes, or indirectly from other system components, e.g., from controllers and motion primitives. The initial operations define all operations that must be performed during the time of creation of the monitor. For example, assignments of initial values $\underline{r}_o$ and $\underline{\ell}_o$ to the controller state variables $\underline{r}$ and $\underline{\ell}$ are such initial operations. Finally, access rights define all connections of the monitor to the rest of the system. This is achieved by giving an explicit list of all processes or other monitors which can access the monitor.

Monitors are used to protect the shared data and to schedule the other resources of the system, as for example some programs, I/O devices, and even a processor or processors. Monitors can also be used to schedule the processes that are in producer-consumer relations. Example for this are sensor data buffers which are accessed by at least two processes: by an input process which provides the sensor data, and by a controller process which uses these data to generate motion increments. In this example, the input process is a producer, while the controller process is a consumer. Synchronization of these processes can be done if the monitor data structure is extended by two single element queues, one for the producer and one for the consumer. The scheduling will work as follows. If the consumer tries to take the data from the empty buffer when the producer has not yet provided the data, it will be blocked and placed into the consumer queue. Also, if the producer tries to put new data into the full buffer when the consumer has not yet taken the previous data from it, it will be blocked and placed into the producer queue. If the consumer has taken the data from the buffer leaving the monitor, the monitor support procedures will automatically check the producer queue, and if it is not empty, the producer process waiting in this queue will be activated. Also, when the producer puts the new data into the buffer, the consumer queue will be checked, and the possibly waiting consumer process will be activated. This method of scheduling can be used for asynchronous scheduling of the controller discussed in subsection 5.3.

A monitor is allocated to all shared data in the MCS. There is no other way to access shared data except by monitor procedures. Some monitors are equipped by the consumer-producer facility, depending on the role of the monitor in the system. The monitors are defined by the application programmer who has written the MCS software. Functioning of the monitors, i.e., scheduling of simultaneous monitor calls, management of producer and consumer queues, corresponding preemption and resumption of the processes, must also be provided by software. However, this part of the software is external to the monitors and it is called "kernel." The kernel is a collection of procedures which support all monitors declared by the application programmer. They implement also the other synchronization mechanisms (for example AWAKE and BLOCK are kernel procedures), and perform scheduling of all other resources, including processor or processors in case of a multiprocessor environment. Therefore, the kernel deals with I/O drivers, interrupt handlers, and other communication and synchronization facilities implemented in the hardware. On the other hand, the kernel is completely free of the application software, and it can be written by a system programmer who is familiar with a particular machine, and may not be familiar with the specifics of an application software, i.e. with the MCS. So is with the application programmer, who does not have to care about details of hardware and communication and synchronization software facilities. The kernel hides these details, extending the particular machine to a virtual machine with given characteristics, pertinent to the MCS requirements.

The whole MCS software can be presented as a three level structure shown in Fig. 12. The first (highest) level is the manipulator task dependent software. It contains controller and I/O processes. This includes processing of event variables and transition conditions, and preparation of arguments for controllers. The second level of the MCS software contains the general system components like motion primitives, controllers and monitors, which can be used for implementation of any complex action or manipulator task. This level supports the first level. The third (lowest) level is the kernel. The kernel is the

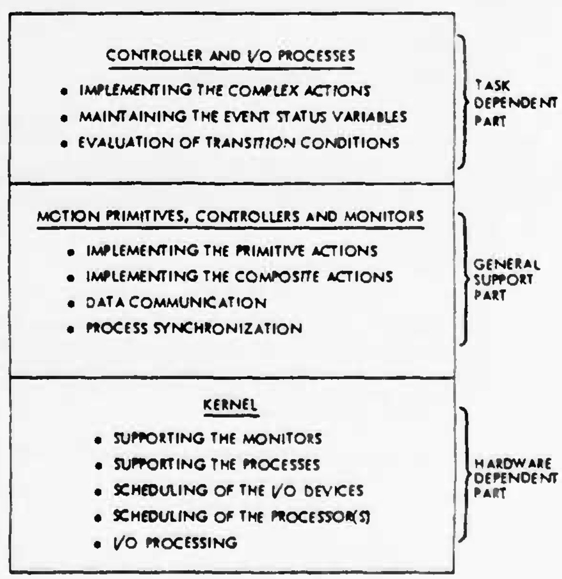


Fig. 12. Structure of the MCS Software

hardware dependent part of the MCS software, and supports the first two levels. The complexity of the kernel depends on the synchronization method used for controller synchonization, on the scheduling strategies used for dispatching the processor, and on a number of processors and their interconnection technique.

In general, using this approach, the changes in hardware will not affect the first two software levels. For example, if the system was running on a given computer and later the original computer configuration is extended to two or more processors, only the kernel must be rewritten. The other parts of the MCS software will remain unchanged.

8. CONCLUSION

The structured approach to the design and implementation of interactive sensor referenced MCS outlined in this paper is related to both control algorithms and their software implementation. The control algorithms are based on the separation of the algorithms that perform the manipulator tasks from the local servo control problem. According to this approach, the interactive sensor referenced manipulator control is being considered as the problem of execution of an action network, and any manipulator task is treated as an arrangement of interconnected actions.

The design approach of this paper has the following advantages: (a) It clearly brings out the problem of logic decision nets as the dominating elements in this type of control. (b) It modularizes software development. (c) It provides a transparent tool for both programmers and users. (d) It may provide design guidelines for special purpose computing machines to improve performance in this type of control. (e) The results can be generalized to similar control problems.

Future work will address the following problems: (a) Extension of the existing manipulator task repertoir and the corresponding command structure. (b) Integration of the existing command interface with a voice command system developed previously. (c) Further investigation of the action precedence graphs together with their application to more complex manipulator tasks and analysis of their structural properties including the inherent deadlock problem. (d) Development of new system components like motion primitives, controllers and monitors.

The results will be evaluated experimentally. The project is still in progress, and systematic experiments are planned to evaluate, improve, and justify the design approach.

ACKNOWLEDGMENT

Part of the research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under NASA Contract NAS7-100. Discussions with Dr. E. Heer on different aspects of this work are appreciated. The programming system support of Mr. R. L. Zawacki in this work is gratefully acknowledged.

REFERENCES

1. D.J. Barber, "MANTRAN: A Symbolic Language for Supervisory Control of an Intelligent Remote Manipulator," Ph.D. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Mass., June 1967.

2. P.H. Hardin, "AND TREE Computer Data Structures for Supervisory Control of Manipulation," Ph.D. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Mass., October 1970.

3. J.W. Hill and A.J. Sword, "Algorithmic Remote Manipulation (ARM) Language" in "Studies to Design and Develop Improved Remote Manipulator Systems," NASA CR-2238, April 1973.

4. E. Shaket and A. Freedy, "A Model and Language Design for Man-Machine Communication in Computer Aided Manipulation," Proceedings of the 1977 IEEE Conference on Decision and Control, New Orleans, Louisiana, December 1977.

5. K. Takase, R.P. Paul, E.J. Berg, "A Structured Approach to Robot Programming and Teaching," IEEE COMPSAC '79, Chicago, Illinois, November 1979.

6. B. Shimano, "User's Guide to VAL, A Robot Programming and Control System," Unimation Inc., Shelter Rock Lane, Danbury, Connecticut 06810, February 1979.

7. T.B. Sheridan and W.L. Verplank, "Human and Computer Control of Undersea Teleoperators," Technical Report, ONR Contract N00014-77-C-0256, Man-Machine Systems Laboratory, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, July 15, 1978.

8. M. Vukobratović, "New Control Concept for Anthropomorphic Manipulators," Mechanism and Machine Theory, Pergamon Press, 1977, Vol. 12, pp. 515-530.

9. D.E. Whitney, "Force Feedback Control of Manipulator Fine Motions," Journal of Dynamic Systems, Measurement and Control, June 1977, pp. 91-97.

10. G.N. Saridis and H.E. Stephanou, "A Hierarchical Approach to the Control of a Prosthetic Arm," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-7, No. 6, June 1977, pp. 407-420.

11. J.Y.S. Luh and M.W. Walker, "Controller for a Mechanical Manipulator," Proceedings of the First International Symposium on Mini- and Microcomputers in Control, San Diego, California, January 8-9, 1979, Acta Press, pp. 123-128.

12. R.C. Paul, "Compliance and Control," Proceedings of the 1976 Joint Automatic Control Conference, Purdue University, West Lafayette, Indiana, July 27-30, 1976, pp.

13. P. Brinch Hansen, "Structured Multiprogramming," Comm. ACM, Vol. 15, No. 7, pp. 574-578, July 1972.

14. P. Brinch Hansen, "Operating System Principles," Prentice-Hall, Inc., Englewood Cliffs, NJ, July 1973.

15. C.A.R., Hoare, "Monitors: an Operating System Structuring Concept," Comm. ACM, Vol. 17, No. 10, pp. 549-557, Oct. 1974.

16. M.I. Vušković, "Experimental Modeling and Evaluation of Sensor-Aided Manipulator Control," Technical Report ITE-10-79, Institute for Technoeconomic Systems, University of Southern California, Los Angeles, California, June 1979.

17. A.K. Bejczy, "Issues in Advanced Automation for Manipulator Control," Proceedings of the 1976 Joint Automatic Control Conference, Purdue University, W. Lafayette, Indiana, July 27-30, 1976.

18. A.K. Bejczy and R.L. Zawacki, "Computer Aided Manipulator Control," Proceedings of the First International Conference on Mini- and Microcomputers in Control, San Diego, California, January 1979.

19. A.K. Bejczy, "Smart Sensors for Smart Hands," Vol. 67 of Progresses in Astronautics and Aeronautics, Publ. AIAA, 1979.

20. G. Paine, "Microprocessors for Real Time Displays and Control of Space Teleoperators," Proceedings of the First International Conference on Mini- and Microcomputers in Control, San Diego, California, January 1979.

21. A.K. Bejczy and G. Paine, "Event-Driven Displays for Manipulator Control," Proceedings of the 14th Annual NASA-University Conference on Manual Control, University of Southern California, Los Angeles, California, April 25-27, 1978.

22. G. Paine and A.K. Bejczy, "Extended Event-Driven Displays for Manipulator Control," Proceedings of the 15th Annual Conference on Manual Control, Dayton, Ohio, March 20-22, 1979.

23. E.W. Dijkstra, "Cooperating Sequential Processes," In Programming Languages, P. Genuys (ed.), Academic Press, New York, NY, 1968.

24. N. Wirth, "On Multiprogramming, Machine Coding and Computer Organization," Comm. ACM, Vol. 12, No. 9, pp. 489-498, Sept. 1969.

25. P. Brinch Hansen, "The Programming Language Concurrent Pascal," IEEE Transactions, Vol. SE-1, No. 2, pp. 199-207, June 1975.

26. P. Brinch Hansen, "The Architecture of Concurrent Programs," Prentice-Hall, Inc., Englewood Cliffs, NJ, 1977.

27. M. Vušković and R.L. Zawacki, "Structured Approach to the Design and Implementation of Computer Control of Manipulators," to be presented at the 11th International Symposium on Mini- and Microcomputers, Pacific Grove, California, January 30 - February 1, 1980.

THIS PAGE LEFT BLANK INTENTIONALLY

SESSION II

NOTES ON COMPUTER-CONTROLLED ROBOTS

J. Y. S. Luh

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907, U.S.A.

I. INTRODUCTION

In the past decade, the problem of productivity attracted an international attention. One of the possible solutions of increasing productivity is to modernize the manufacturing facilities by means of automation. Among others, the programmable automation is most suitable to handle the low volume batch production of discrete parts. The industrial robots which are defined as computer controlled mechanical manipulators used in industrial applications, fall into this category. Typically they perform the tasks of arc welding, paint spraying, foundry operation, etc. One may assign a robot to perform a variety of job assignments simply by changing the appropriate computer program and thereby increase the capability and flexibility of the industrial robots.

Typically an industrial robot has six joints, giving six degrees of freedom, with a gripper which is referred to as a hand or an end effector. Figure 1 shows the commercially available Cincinnati Milacron Model T3, and Unimation PUMA 600. Figure 2 shows a Stanford manipulator [1] which is also an industrial robot by definition existing among research institutes in the United States. Each joint of these robots is driven hydraulically, pneumatically or electrically with a feedback control loop. As an example, a block diagram for a joint control of the Stanford manipulator at the Purdue

University, which has a permanent magnet motor drive [2], is shown in Figure 3.

There are two groups of input signals to the closed-loop control system. The first group consists of the desired compliant torque T_d, the anticipated gravity torque T_a, and the desired angular displacement θ_d. The values of torques T_d and T_a may be computed using the methods described in references [3] by Paul, [4] by Bejczy, or [5] by Luh, Walker and Paul, while θ_d can be obtained from the scheme in [6] by Luh and Lin. These signals actuate the system so that enough voltage is applied to the motor to produce sufficient torque to handle the load. In the existing control system, these three input quantities are computed on a PDP 11/45 and loaded into LSI-11 microprocessor which is interfaced to the manipulator. The second group represents the reaction from the physical burden to the robot, i.e., load T_L, gravitational torque T_g, and frictional torque f_m of the motor-tachometer assembly. The load T_L is transferred through the output shaft and geared down to the motor shaft. The harmonic drive has a gear ratio of $n = 1/100$. Besides the tachometer feedback, the armature controlled DC motor also has a back emf with a gain of K_b volts/rad. contributing to the velocity feedback. In addition, an optical encoder is connected to the shaft of the motor to provide a positional feedback. In Figure 3, K_T and K_V are gains in the forward path; and L and R are the inductance and resistance of the motor-armature winding, respectively. A backlash nonlinearity is added immediately after the gear reduction since, at the output side of the harmonic drive, there is an unavoidable backlash. Thus, an industrial robot is a positioning device in that each of its joints has a positional control system.

II. CARTESIAN AND JOINT COORDINATES

In reality a robot task is naturally specified in terms of its hand in Cartesian coordinates. It consists of position, described by a position vector $\underline{p}(t)$, and orientation, described by a unit approach vector $\underline{a}(t)$ and a unit sliding vector $\underline{s}(t)$, as indicated in Figure 4(a). All these vectors are defined with reference to the base coordinates. For convenience, a unit normal vector is defined as $\underline{n}(t) = \underline{s}(t) \times \underline{a}(t)$ where $\times$ denotes the "cross product." The orientation may also be defined in terms of Euler angles with respect to the base coordinates. Initially at $t = t_0$, let $\underline{n}(t_0)$, $\underline{s}(t_0)$ and $\underline{a}(t_0)$ align with $\underline{x}_0$, $\underline{y}_0$ and $\underline{z}_0$, respectively, as shown in Figure 4(b). Any orientation $\left[\underline{n}(t)\ \ \underline{s}(t)\ \ \underline{a}(t)\right]$ may be obtained by a rotation of γ radians about $\underline{z}_0$ so that $\underline{s}(t_0)$ aligns with $\underline{s}(t_1)$; then a rotation of β radians about $\underline{s}(t_1)$ so that $\underline{a}(t_0) = \underline{a}(t_1)$ aligns with $\underline{a}(t)$, and finally a rotation of α radians about $\underline{a}(t)$ to obtain the required $\underline{n}(t)$ and $\underline{s}(t)$. This is equivalent to rotate the $\left[\underline{n}(t_0)\ \ \underline{s}(t_0)\ \ \underline{a}(t_0)\right]$ coordinate, which aligns with $\left[\underline{x}_0\ \ \underline{y}_0\ \ \underline{z}_0\right]$ originally, α radians about $\underline{z}_0$, then β radians about $\underline{y}_0$, and finally γ radians about $\underline{z}_0$ again. These three consecutive rotations may be represented by rotational operators

$$\underline{\mathcal{R}} = \underline{R}(\underline{z}_0,\gamma)\underline{R}(\underline{y}_0,\beta)\underline{R}(\underline{z}_0,\alpha)$$

$$= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Since $[\underline{n}(t_0)\ \underline{s}(t_0)\ \underline{a}(t_0)]$ aligns with $[\underline{x}_0\ \underline{y}_0\ \underline{z}_0]$ originally, then

$$\begin{cases} \underline{n}(t) = \underline{\mathcal{R}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ \underline{s}(t) = \underline{\mathcal{R}} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \\ \underline{a}(t) = \underline{\mathcal{R}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{cases} \tag{2}$$

Now the origin of $[\underline{n}(t_0)\ \underline{s}(t_0)\ \underline{a}(t_0)]$ is translated in the based coordinates $(\underline{x}_0, \underline{y}_0, \underline{z}_0)$ with an amount and in the direction of $\underline{p}(t)$. Thus a 3 by 4 matrix, which describes the orientation and position of the hand at time t, may be written as

$$[\underline{n}(t)\ \underline{s}(t)\ \underline{a}(t)\ \underline{p}(t)] = \underline{\mathcal{R}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + [\underline{0}\ \underline{0}\ \underline{0}\ \underline{p}] \tag{3}$$

To simplify the representation of the hand and the mathematical transformation of its orientation and position, the above operations described by (3) may be written as

$$\underline{H}(t) = \begin{bmatrix} \underline{n}(t) & \underline{s}(t) & \underline{a}(t) & \underline{p}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} & \underline{\mathcal{R}} & & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

which represents the state of hand at time t in Cartesian coordinates. The 4 by 4 matrix on the right side of the first equation of (4) is referred to as the coordinate frame of the hand at time t with reference to the base coordinates. The first 4 by 4 matrix in the second equation of (4) is the homogeneous transformation in the base coordinates which includes both rotation and translation [7]. Since, in this case, $[\underline{n}(t_0)\ \underline{s}(t_0)\ \underline{a}(t_0)]$ aligns with $[\underline{x}_0\ \underline{y}_0\ \underline{z}_0]$ originally, the two 4 by 4 matrices mentioned above are identical. More discussions on the coordinate frame and the homogeneous transformation are given in Appendix A.

From (2) it is seen that the state of hand at time t in Cartesian coordinates with reference to the base coordinates may also be represented by a 6-dimensional vector $[\underline{p}(t)'\ \ \underline{\theta}(t)']$ where $[\underline{\theta}(t)]' = [\alpha\ \ \beta\ \ \gamma]$ and $(\ )' =$ transpose of ( ). The hand, however, is driven by the actuators at the joints. Intuitively, if all the joint displacements are given, the position and orientation of the hand are determined. Let n be the number of joints. For i = 1, 2, ..., n, let q_i be the displacement of the i-th joint with respect to its own reference point. Then, for any given robot with known geometrical dimensions, there is a relation

$$\left[\underline{p}(t)' \ \underline{\theta}(t)'\right] = \underline{f}\left[q_1, q_2, \cdots, q_n\right] \tag{5}$$

where $\underline{f}(\cdot)$ is a 6 by 1 vector valued function. This relation is known, but almost always nonlinear which complicates the problem [8]. Since in reality, one specifies $\left[\underline{p}(t)' \ \underline{\theta}(t)'\right]$ in Cartesian coordinates and desires to determine the corresponding $\left[q_1, \cdots, q_n\right]$ so that one may command the joint actuators to comply with the specification in Cartesian coordinates. The solution requires the inverse vector *function* $\underline{f}^{-1}(\cdot)$ *of* n-*dimension*. This solution, if it can be found, may not be unique. for the commercially available robots in operation, n is usually either 5 or 6. The geometrical configuration of these robots with proper definitions and ranges of q_i enables one to obtain a unique solution of equation (5) [8]. The kinematic equations and their solutions are discussed further in Appendix B.

Knowing the transformation of position in Cartesian and joint coordinates, it is ready to examine the following simple task. As shown in Figure 5, the robot is required to go to the bolt magazine to fetch a bolt, and place it on top of the bolt hole on the beam. Intuitively one expects that the robot hand travels along the path of straight-line segments, as indicated in Figure 5, so that it will reach the bolt first and then arrive at the beam. The question is how would one control the joint actuators to accomplish the goal? Before one arrives with an answer, one may examine the following two possible specifications:

(a) Is the work space free from obstacles?

(b) Must the hand follow the specified path?

The answers to each of these two questions could be either yes or no. They combine to form four different classes of problems as follows:

		Is the work space free from obstacles? Yes	No
Must the hand follow the specified path?	No	Class 1. Positional Control Problem	Class 4. Positional Control Plus On-line Collision Avoidance Travelling
	Yes	Class 2. Path Tracking Problem	Class 3. Off-line Collision-free Path Planning Plus On-line Path Tracking

These problems will be analyzed briefly in the following sections.

III. POSITIONAL CONTROL

If there is no path constraints and if the work space is free, then the controllers only have to make sure that the hand passes through all the specified corner points of the path. It involves the coordinates transformation, which computes the corresponding joint coordinates $\left[q_1, \cdots, q_n\right]$ of the specified corner points in Cartesian coordinates by means of $\underline{f}^{-1}(\cdot)$, and then positionally control the robot in joint coordinates from point to point. A linear positional servo for each joint, such as the one shown in Figure 3, will serve the purpose. With this elementary control scheme, the hand stops at every corner point (when the position error vanishes) before moves towards the next corner point which, from the engineering point of view, is not efficient. Since each joint moves with its own velocity with no coordination of motions of other joints, the traveled path by the hand in Cartesian coordinates does not, in general, consists of straight line segments between corners.

The input to the control system is the desired Cartesian corner points of the path, which may be numerically fed into the system, or furnished through so-called "teaching by doing," i.e., corner points are recorded while the hand of the robot is led through these points manually by an operator.

Appendix C presents a detailed discussion on conventional controllers design for industrial robots.

IV. PATH TRACKING

If the robot is required to travel along a prescribed path, the controller must keep up with path tracking. With positional controllers the path tracking can be accomplished by dividing the Cartesian path into a number of segments. Each end point of the segments is transformed into joint coordinates, and then the positional control is applied from point to point in joint coordinates. A number of facts related to this approach should be mentioned. By transforming all the end points of segments of Cartesian path, one essentially constructs n corresponding paths in joint coordinates, one for each of the n joints. If these segments, $\left[d\underline{p}(t)' \quad d\underline{\theta}(t)'\right]$, are very short, the increments of joint displacement, dq_i, between adjacent points are very small so that $\sin dq_i \simeq dq_i$ and $\cos dq_i \simeq 1$. Thus the transformation $\underline{f}(\cdot)$ defined by (5) becomes a differential transformation which is usually linear. This transformation is the Jacobian matrix of the displacement, which contains trigonometric functions of the joint displacement with respect to the joint coordinates before the differential increment takes place [9]. Analytically, the solution dq_i in terms of $d\underline{p}$ and $d\underline{\theta}$ can be obtained simply by inverting the Jacobian matrix. Although it is sometimes possible, it is usually difficult since the Jacobian is quite complicated. Numerical solution is also possible but usually requires long computing time. Moreover, the Jacobian matrix becomes singular when the robot reaches a degenerate position at which the solution dq_i is not unique (i.e., more than one value of dq_i yields a same $d\underline{p}$ and $d\underline{\theta}$.) An alternative method proven to work successfully is to differentiate the solution of equation (5) directly [9]. This is possible since for a given robot with fixed dimensions, the transformation $\underline{f}^{-1}$ is known. Using this approach, one must set dq_i to zero if it is physically impossible due to constraints, or if it is undetermined. It usually results in a simpler expression [9]. A detailed discussion on differential or Jacobian transformation is included in Appendix D.

The desired Cartesian corner points of the path may be fed into the control system either numerically, or through the teaching by doing procedure. The intermediate points of the small segments of the path are usually specified, or interpolated by the computer-controller, or sampled and recorded through teaching by doing.

By controlling the robot from point to point using the positional controller, it has a natural tendency to stop at each point. This undesirable phenomenon may be eliminated by specifying a nonzero velocity at each Cartesian point, which can be transformed into the corresponding nonzero joint velocities by an inverse Jacobian matrix of the velocity. The joint velocities are often referred to as the "resolved rate" [10]. For implementation a velocity control loop must be added to each controller. Alternatively, one may take advantage of the associated digital computer to compute the required joint torques which would yield appropriate velocities at fixed accelerations. The computation requires the knowledge of the dynamical equation of the robot. Since the robot is a nonlinear mechanism with couplings among its joints, the computation is time consuming.

A possible way to avoid the storage of numerous pre-computed points of joints paths or the computation of these points on-line is to determine the functional representation of n joint paths. However, the transformation f in (5) is pointwise. Since no transformation of a function is known, the curve fitting procedure may be adopted as follows [3,11]. The corner points of the Cartesian path are first transformed into joint coordinates. A cubic function is assigned to each segment between every two adjacent joint coordinate points. The adjacent cubic functions are then splined together with the continuity conditions of desired velocity and, if required, desired acceleration. Curve fitting is done by using a large number of points on the path, after they are transformed into joint coordinates, and applying the least square error fit to obtain the coefficients of the spline function. Thus one needs to store only the coefficients. The points on the joint paths can be obtained by a simple computation when needed.

An alternative method to the above mentioned differential straight line motion is the on-line evaluation and decision making scheme developed by Paul [12]. the scheme is divided into three processes: (a) first the current position and velocity of the hand in Cartesian coordinates are evaluated for the path accuracy to decide whether a change in velocity is needed; (b) points on the path of hand between which the change of velocity takes place, are generated and transformed into the corresponding points in joint coordinates; and (c) finally a quadratic interpolation between these points in joint coordinates is determined for the robot to follow. These processes are repeated frequently at a sampling rate not less than the structural resonant frequency of the robot. Taylor [13] employs the quaternion representation [14] for the coordinate frame of the hand instead of the homogeneous transformation representation, so that the rotational operations require less computations. The translational operations however, do not yield any advantage. In his approach, enough intermediate points in the joint coordinates are added so that the path is determined by linear interpolation which is restricted by a prescribed, maximum allowable path deviation. Thus the computation is simpler than that for the case of quadratic interpolation.

V. COLLISION-FREE PATH PLANNING

If the work space is not free from obstacles, then it is best to determine a collision-free Cartesian path and then impose the path tracking requirement. Conventionally the path is created by the operator and transferred to the computer storage for the robot by recording a sequence of path points while the operator leads the robot hand through the path. When the work space near the goal point is crowded with obstacles, the teaching by doing method by operator becomes insufficient, especially if minimum execution time or distance is imposed.

The subject of collision-free path planning is relatively new. Within the past four or five years, only a handful people are actively working on this topic [15-20]. Essentially the robot is treated as a point by suitably expanding obstacles to compensate for robot body dimensions. A 2-dimensional example of robot-environment of Lozano-Perez is shown in Figure 6 as an illustration. A triangular cross-section of robot is required to move around a stationary, rectangular obstacle without any rotation. When the triangle touches the rectangle and slides around it, the lower left corner of the triangle yields a hexagon around the rectangle. Thus, if the robot is treated as a point located at its lower left corner, the rectangular obstacle must be expanded to a hexagon to compensate the robot body. The concept can be extended to three-dimensional case with moving obstacles. One must be cautious during the final approach to a goal so that the expansion does not engulf the goal point. Using a set of heuristically derived rules, it is possible to develop an algorithm which determines a feasible and best (in the sense of either minimum execution time or minimum distance traversed) path [20]. Intuitively the algorithm should be implemented with an interactive computer graphics display of a robot in various environments which allows simulation of the robotic assembly line. However, one must be careful about the hidden lines of the 2-dimensional projection since they essentially indicate whether there is a collision.

VI. COLLISION AVOIDANCE TRAVELLING

The problem discussed in Section V involves the conditions that the work space is not free from obstacles and that the collision-free paths must be followed. If the second condition is not imposed, then the robot is free to move anywhere in the space which runs a risk of collision. Thus, when the positional control is implemented, an on-line collision avoidance capability is needed to ensure the safety of the operation. Very little research results are known in this area. A common practice is to stop the motion whenever an obstacle in the path is detected.

VII. COMMAND LANGUAGE AND TASK DESCRIPTION

To delegate the task of controlling a robot to a computer, it is necessary to establish communications between the computer and the operator through a command language. Practically for each different product of robot, there is a different command language. Each was developed for its own goal tasks, and unfortunately they are not unified.

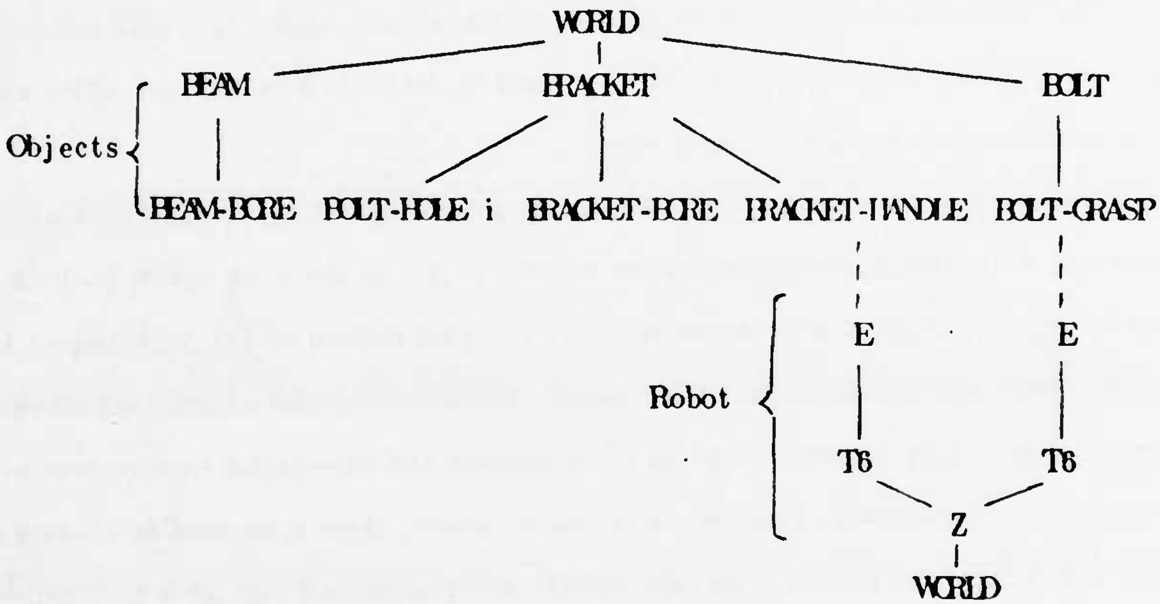
The first of such language may be traced back to LISP [21]. At the Stanford University, Paul [22] originally developed WAVE programming language which is capable of performing assembly by accepting force and visual sensing inputs and computing joint torques using Lagrangian formulation of dynamic equation. It is later elevated to high level language AL by Finkel [23] in which AFFIX statements are used for expressions. The drawback is the involvement of vast amount of computations. To simplify the implementation, POINTY [24] was developed but unfortunately it is still complicated. Based on WAVE, Shimano [25] developed the robot control language VAL for Unimation Inc. which simplifies the task by interpreting the program on line by line basis. Independently at IBM Research Center, Lieberman and Wesley developed AUTOPASS [26] as a model-based program automation tool while Summers, Taylor and Meyer created AML [27] which is a semantically interactive language. There are many other programming languages in use developed in U.S.A. and abroad. Each of them is a complete language system which includes scanners, parsers and symbol table. In Italy Gini and Gini [28] worked on a goal orientated language while Bernorio [29] et al. developed a quasi-natural language. A language for part assembly, RAPT [30] was introduced by Popplestone et al. of University of Edinburgh, Scotland. Falek and Parent [31] in France, and Blume and Dillmann [32] in Federal Republic of Germany also developed robotic languages independently. At Purdue University, Paul [33] takes a different approach by embedding the motion primitives and data structure into a high level computer language PASCAL thereby eliminating the development of his own parser, etc. He combines the joint coordinates control with Cartesian coordinates task description to include compliance for part assembly using the joint torque sensing. The resulting language is called PAL [34] (Purdue Arm Language) which is in the process of further development, and which uses the homogeneous transformation to describe the variations of hand position and orientation of the robot. In the following the homogeneous transformation approach of PAL is used to illustrate how the task of the robot is described and its motion is commanded.

PAL is developed to suit the position-dependent structure of robots' tasks. By assigning a Cartesian coordinate system to each object in the work space, each object can be represented by a 4 by 4 matrix, similar to $\underline{H}(t)$ defined by (4), with respect to a chosen reference coordinates. Any specific geometry on the object can then be described by a 4 by 4 matrix in terms of its position and orientation with respect to its own objects' coordinates. Thus, when the object moves, there is no need to up-date the 4 by 4 matrix of the geometry on that object. For convenience, all the 4 by 4 matrices are called the Cartesian coordinate frames. If, for some reason, the reference coordinates of a frame need to be changed, i.e. changing its position and/or orientation, it can be done by a simple homogeneous transformation [7], i.e., premultiply the subject frame by a 4 by 4 matrix which describes the old reference coordinates with respect to the new coordinates.

Using an example in Reference [35] as an illustration, refer to Figure 7. The task of the robot hand is to lift the bracket by its handle and set it on top of the beam to align their bores, and then fetch a bolt and place it in the bore of bracket. To set up the task, select a convenient WORLD coordinates as a Cartesian base coordinates. Its frame is simply a 4 by 4 identity matrix. Then define Z with respect to WORLD to represent the robot, define T6 with respect to Z to represent the robot wrist, and define

E with respect to T6 to represent the robot end-effector (or the hand in this example.)

The objects must be specified in the same base coordinates. Thus define BEAM, BOLT and BRACKET with respect to WORLD. The geometries on the objects are defined as follows: BEAM-BORE with respect to BEAM; BRACKET-HANDLE, BRACKET-BORE, and BOLT-HOLE i(i = 1, ···, 4) with respect to BRACKET; and BOLT-GRASP with respect to BOLT. The coordinate structure is as follows:



To make it possible for the robot to perform the task, the following position equations must be satisfied:

$$Z * T6 * E = BRACKET * BRACKET\text{-}HANDLE \tag{6}$$

and $$Z * T6 * E = BOLT * BOLT\text{-}GRASP \tag{7}$$

These two equations allow the end-effector making contact with the BRACKET-HANDLE and with BOLT-GRASP. From (6) and (7) it is seen that BRACKET can be expressed as $Z * T6 * E * (BRACKET\text{-}HANDLE)^{-1}$ and BOLT as $Z * T6 * E * (BOLT\text{-}GRASP)^{-1}$. Thus for BRACKET-BORE contacting BEAM-BORE, one requires

$$Z * T6 * E * (BRACKET\text{-}HANDLE)^{-1} * BRACKET\text{-}BORE = BEAM * BEAM\text{-}BORE \tag{8}$$

Likewise, from (7) and (8), one obtains $BOLT = Z * T6 * E * (BOLT\text{-}GRASP)^{-1}$ and $BRACKET = BEAM * BEAM\text{-}BORE * (BRACKET\text{-}BORE)^{-1}$. The final requirement is to satisfy

$$\begin{aligned} Z * T6 * E * (BOLT\text{-}GRASP)^{-1} = {} & BEAM * BEAM\text{-}BORE \\ & * (BRACKET\text{-}BORE)^{-1} * BOLT\text{-}HOLE\ i \end{aligned} \tag{9}$$

Equations (6) through (9) form the basis of the positional requirement for the task. For the convenience of programming, define two macro symbols

$$ARM ::= NAME * Z * T6 \tag{10}$$

and $$TOL ::= E \tag{11}$$

where

$$NAME = \begin{cases} (BRACKET)^{-1} & \text{for eq. (6)} \\ (BOLT)^{-1} & \text{for eq. (7)} \\ [BEAM * BEAMBORE * (BRACKET\text{-}BORE)^{-1}]^{-1} & \text{for eq. (8)} \\ [BEAM * BEAM\text{-}BORE * (BRACKET\text{-}BORE)^{-1} * BOLT\text{-}HOLE\ i]^{-1} & \text{for eq. (9)} \end{cases}$$

Then the requirement becomes

$$ARM * TOL = \begin{cases} BRACKET\text{-}HANDLE, & \text{for eq. (6)} \\ BOLT\text{-}GRASP, & \text{for eq. (7)} \\ BRACKET\text{-}HANDLE, & \text{for eq. (8)} \\ BOLT\text{-}GRASP, & \text{for eq. (9)} \end{cases}$$

To manipulate the robot, introduce a MOV statement such that MOV<expression> yields a motion of positioning the robot such that

$$\text{ARM} * \text{TOL} = \langle\text{expression}\rangle \tag{12}$$

Thus

$$\text{expression} = \begin{cases} \text{BRACKET-HANDLE, for eq. (6)} \\ \quad \text{==> robot contacts BRACKET-HANDLE} \\ \text{BOLT-GRASP, for eq. (7)} \\ \quad \text{==> robot contacts BOLT-GRASP} \\ \text{BRACKET-HANDLE, for eq. (8)} \\ \quad \text{==> BRACKET-BORE contacts BEAM-BORE} \\ \text{BOLT-GRASP, for eq. (9)} \\ \quad \text{==> BOLT contacts BOLT-HOLE i} \end{cases}$$

For further discussions, readers are referred to Chapters 5 and 10 of Reference [33].

REFERENCES

1. Scheinman, V. D., *Design of a Computer Manipulator,* A.I. Memo 92, Artificial Intelligence Laboratory, Stanford University, June 1969.
2. Paul, R., J. Luh, et al., *Advanced Industrial Robot Control Systems*, Seventh Report, TR-EE 81-8, School of Electrical Engineering, Purdue University, March 1981, p. 23.
3. Paul, R. C., *Modeling, Trajectory Calculation and Servoing of a Computer Controlled Arm,* A. I. Memo 177, Artificial Intelligence Laboratory, Stanford University, September 1972.
4. Bejczy, A. K., *Robot Arm Dynamics and Control,* Technical Memorandum 33-669, Jet Propulsion Laboratory, February 1974.
5. Luh, J. Y. S., M. W. Walker and R. P. C. Paul, "Resolved-Acceleration Control of Mechanical Manipulators," IEEE Transactions on Automatic Control, Vol. 25, No. 3, June 1980, pp. 468-474.
6. Luh, J. Y. S. and C. S. Lin, "Optimum Path Planning for Mechanical Manipulators," Trans. of ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 103, No. 2, June 1981, pp. 142-151.
7. Roberts, L. G., *Homogeneous Matrix Representation and Manipulation of N-dimensional Constructs,* Lincoln Laboratory Document, No. MS 1045, MIT, 1965.
8. Paul, R. P., B. Shimano and G. E. Mayer, "Kinematic Control Equations for Simple Manipulators, "IEEE Transactions on Systems, Man and Cybernetics, Vol. 11, No. 6, June 1981, pp. 449-455.
9. Paul, R.P., B. Shimano and G. E. Mayer, "Differential Kinematic Control Equations for Simple Manipulators," ibid, pp. 456-460.
10. Whitney, D. E., "Resolved Motion Rate Control of Manipulators and Human Prostheses," IEEE Transactions on Man-Machine Systems, Vol. 10, No. 2, June 1969, pp. 47-53.
11. Finkel, R. A., *Constructing and Debugging Manipulator Programs,* A. I. Memo 284, Artificial Intelligence Laboratory, Stanford University, August 1976.
12. Paul, R., "Manipulator Cartesian Path Control," IEEE Transaction on Systems, Man, and Cybernetics, Vol. 9, No. 11, November 1979, pp. 702-711.
13. Taylor, R. H., "Planning and Execution of Straight Line Manipulator Trajectories," IBM Journal of Research and Development, Vol. 23, No. 4, July 1979, pp. 424-436.
14. Yang, A. T. and F. Freudenstein, "Application of Dual-Number Quaternion Algebra to the Analysis of Spatial Mechanisms," Trans. of ASME, Journal of Applied Mechanics, Vol. 86, 1964, pp. 300-308.

15. Pieper, D. L., *The Kinematics of Manipulators Under Computer Control*, A. I. Memo 72, Artificial Intelligence Laboratory, Stanford University, October 1968.
16. Widdoes, C., *A Heuristic Collision Avoider for the Stanford Robot Arm*, C. S. Memo 227, ibid, June 1974.
17. Udupa, S. M. *Collision Detection and Avoidance in Computer Controlled Manipulators*, Ph.D. Thesis, California Inst. of Technology, 1977.
18. Lozano-Perez, T. and Wesley, M. A., "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM*, Vol. 22, #10, Oct. 1979, pp. 560-570.
19. Lozano-Perez, T., "Spacial Reasoning in the Planning of Robot Motions," Proceedings of 1981 Joint Automatic Control Conference, Vol. 1, June 17-19, 1981, Charlottesville, Va., pp. WP-2D.
20. Campbell, C. E. and Luh, J. Y. S., *A Preliminary Study on Path Planning of Collision Avoidance for Mechanical Manipulators*, Technical Report TR-EE 80-48, Purdue University, December, 1980.
21. Weissman, C., *LISP 1.5 Primer*, Dickenson Publishing Company, 1967.
22. Paul, R., "WAVE - A Model Based Language for Manipulator Control," The Industrial Robot, Vol. 4, No. 1, March 1977, pp. 10-17.
23. Finkel, R. et al., *AL: A Programming System for Automation,* A. I. Memo 243, Artificial Intelligence Laboratory, Stanford University, November 1974.
24. Binford, T. O. et al., *Exploratory Study of Computer Integrated Assembly Systems,* A. I. Memo 285 (STAN-CS-76-568, Progress Report 4), ibid, August 1, 1976 through March 31, 1977.
25. Shimano, B. "VAL: A Versatile Robot Programming and Control System," Proc. COMPSAC 79 Third International Computer Software and Applications Conference, Chicago, November 6-8, 1979, pp. 878-883.
26. Lieberman, L. I. and M. A. Wesley, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly," IBM Journal of Research and Development, Vol. 21, No. 4, 177, pp. 321-333.
27. Summers, P., R. Taylor and J. Meyer, "AML: A Programming Language for Automation," Proceedings of 5th International Computer Software and Applications Conference (COMPSAC), November 18-20, 1981, Chicago, pp. 154-155.
28. Gini, G. and M. Gini, "Control of Intelligent Robots and Goal Orientated Languages," The Industrial Robot, Vol. 2, No. 2, June 1975, pp. 67-74.
29. Bernorio, M. et al., "Programming a Robot in Quasi-natural Language," ibid, Vol. 4, No. 3, September 1977, pp. 132-140.
30. Popplestone, R. J., A. P. Ambler and I. Bellos, "RAPT: A Language for Describing Assemblies," ibid, Vol. 5, No. 3, September 1978, pp. 131-137.

31. Falck, D. and M. Parent, "An Evolutive Language for an Intelligent Robot," ibid, Vol. 7, No. 3, September 1980, pp. 168-171.
32. Blume, V. C. and R. Dillmann, "Struktur und Programmierung von Industrieroboten, Part 1," VDI-Z, Vol. 122, No. 5, March 1980, pp. 159-237.
33. Paul, R. P., *Robot Manipulators: Mathematics, Programming and Control,* MIT Press, 1981.
34. Paul, R., J. Luh, et al., *Advanced Industrial Robot Control Systems,* Fourth Report, TR-EE 80-29, School of Electrical Engineering, Purdue University, July 1980.
35. Takase, K., R. Paul, and E. Berg, "A Structured Approach to Robot Programming and Teaching," IEEE Transactions on Systems, Man and Cybernetics, Vol. 11, No. 4, April 1981, pp. 274-289.

APPENDIX A

COORDINATE FRAME AND HOMOGENEOUS TRANSFORMATION

In Section II, equation (4), it is shown that the hand of the robot can be described by a 4 by 4 hand matrix:

$$\underline{H} = \begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\underline{n}$, $\underline{s}$ and $\underline{a}$ are orthonormal unit vectors describing the orientation of the hand, and $\underline{p}$ is the position vector. All the vectors are defined with respect to the base coordinates, hence the hand matrix represents the orientation and position of the hand with reference to the base coordinates. In H, the 3 by 3 orientation matrix $[\underline{n},\underline{s},\underline{a}]$ can be viewed as a coordinate system $[\underline{x}_n,\underline{y}_n,\underline{z}_n]$ with respect to base coordinates for the joint n of the robot having n joints (i.e., the farthermost joint away from the base). This coordinate system is fixed on joint n so that it moves with joint n. Consequently rotating and then translating joint n, and hence the hand, with respect to base coordinates is the same as rotating and then translating $[\underline{x}_n,\underline{y}_n,\underline{z}_n]$ with respect to the base coordinates. Thus the resulting orientation and position of the hand after some rotation and translation can be computed by premultiplying $\underline{H}$ by an appropriate homogeneous transformation corresponding to the specified rotation and translation.

From equation (4), the first 4 by 4 matrix in the second equation is a homogeneous transformation. It has the form of

$$\begin{bmatrix} & \underline{\mathcal{R}} & & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

in which $\underline{\mathcal{R}}$ is a 3 by 3 rotation matrix and $\underline{p}$ is a 3 by 1 translation vector. Because $\underline{\mathcal{R}}$ is a rotation matrix, its three columns are orthonormal vectors with a unity norm.

Thus the homogeneous transformations for pure rotation and pure translation are, respectively,

$$\begin{bmatrix} & \underline{\mathcal{R}} & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \underline{p} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that

$$\begin{bmatrix} 1 & 0 & 0 & \underline{p} \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} & & & 0 \\ & \underline{\mathcal{R}} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \underline{\mathcal{R}} & & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that

$$\begin{bmatrix} & \underline{\mathcal{R}} & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \underline{p} \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \neq \begin{bmatrix} & \underline{\mathcal{R}} & & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Supposing the hand aligns with the base coordinates initially so that

$$\begin{bmatrix} \underline{x}_n^o & \underline{y}_n^o & \underline{z}_n^o & \underline{p}^o \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is the initial coordinate frame of the hand with respect to base coordinates. Now rotate and then translate the hand by $\underline{\mathcal{R}}$ and $\underline{p}$ respectively so that the final coordinate frame of the hand is

$$\begin{bmatrix} \underline{x}_n & \underline{y}_n & \underline{z}_n & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \underline{p} \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & \underline{\mathcal{R}} & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{x}_n^o & \underline{y}_n^o & \underline{z}_n^o & \underline{p}^o \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} & \underline{\mathcal{R}} & & \underline{p} \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus

$$\begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \underline{\mathcal{R}} & & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

which is identical to equation (4).

Let $\underline{T}(\underline{q})$ be a translational operator which is a 4 by 4 matrix. If the hand is further rotated θ_x radians about x_o-axis, then θ_y radians about y_o-axis, then θ_z radians about z_o-axis, and finally it is translated $\underline{q}$ with respect to the base coordinates; the final orientation and position of the hand with reference to the base coordinates is thus given by

$$\begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{final} = \underline{T}(\underline{q}) \begin{bmatrix} \underline{R}(\underline{z}_0,\theta_z) & & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{R}(\underline{y}_0,\theta_y) & & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{R}(\underline{x}_0,\theta_x) & & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & & & \underline{p} \\ & \underline{\mathcal{R}} & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \underline{q} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & \underline{R} & & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The idea of coordinate frame may be applied to an object to describe the change of its orientation and position. To illustrate the point consider two identical wedges which are placed in a scene coordinate system $[\underline{x}_s, \underline{y}_s, \underline{z}_s]$ as shown in Figure 8. It is required to relocate them as shown in Figure 9. First, the two wedges are labelled arbitrary W1 and W2 in both figures. Obviously W1 may reach its required location if it is rotated 90° about $\underline{x}_s$ and then 90° about $\underline{z}_s$. Mathematically, coordinate systems $[\underline{x}_{W1}, \underline{y}_{W1}, \underline{z}_{W1}]$ and $[\underline{x}_{W2}, \underline{y}_{W2}, \underline{z}_{W2}]$ with reference to $[\underline{x}_s, \underline{y}_s, \underline{z}_s]$ are assigned to W1 and W2, respectively, as shown in Figure 10. Then

$$\begin{bmatrix} \underline{x}_{W1} & \underline{y}_{W1} & \underline{z}_{W1} & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \underline{x}_{W2} & \underline{y}_{W2} & \underline{z}_{W2} & 0 \\ & & & 5 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

are the two coordinate frames for W1 and W2, respectively, with respect to $[\underline{x}_s, \underline{y}_s, \underline{z}_s]$.

Thus the final location of W1 may be achieved by the following operations

$$\begin{bmatrix} \underline{x}_{W1} & \underline{y}_{W1} & \underline{z}_{W1} & \underline{p}_{W1} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{final} = \begin{bmatrix} \underline{R}(\underline{z}_s, 90^\circ) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{R}(\underline{x}_s, 90^\circ) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{x}_{W1} & \underline{y}_{W1} & \underline{z}_{W1} & \underline{p}_{W1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since $\sin 90^\circ = 1$ and $\cos 90^\circ = 0$, and in addition, $\underline{p}_{W1} = (0,0,0)$ and $[\underline{x}_{W1}, \underline{y}_{W1}, \underline{z}_{W1}]$ aligns with $[\underline{x}_s, \underline{y}_s, \underline{z}_s]$ initially; then the above expression reduces to

$$\begin{bmatrix} \underline{x}_{W1} & \underline{y}_{W1} & \underline{z}_{W1} & \underline{p}_{W1} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{final} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus

$$\underline{x}_{W1}\Big|_{final} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \underline{y}_{W1}\Big|_{final} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \underline{z}_{W1}\Big|_{final} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \underline{p}_{W1}\Big|_{final} = \underline{0}$$

which implies $\underline{x}_{W1}$ finally aligns with $\underline{y}_s$, $\underline{y}_{W1}$ with $\underline{z}_s$, $\underline{z}_{W1}$ with $\underline{x}_s$, and there is no translation.

Assume that the wedges are rigid bodies. Then the same rotational operators apply to every point on the wedge. As an example, consider the corner point on W1 represented by the vector [-1,4,0] (see Figure 8). To apply the homogeneous transformation, a "1" is added to the vector as the fourth element so that the final location of the corner point may be computed as:

$$\begin{bmatrix} \underline{v} \\ 1 \end{bmatrix}_{\text{final}} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 4 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 4 \\ 1 \end{bmatrix}$$

which implies

$$\underline{v}\Big|_{\text{final}} = \begin{bmatrix} 0 \\ -1 \\ 4 \end{bmatrix}$$

Intuitively, for W2 to reach its final location, it may be rotated -90° about $\underline{x}_s$, then 90° about $\underline{z}_s$, and finally translated 9 units in the positive $\underline{z}_s$ direction. The final location of W2 may thus be computed as

$$\begin{bmatrix} \underline{x}_{W2} & \underline{y}_{W2} & \underline{z}_{W2} & \underline{p}_{W2} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{\text{final}} = \begin{bmatrix} 1\,0\,0 & 0 \\ 0\,1\,0 & 0 \\ 0\,0\,1 & 9 \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} \underline{R}(\underline{z}_s, 90^\circ) & 0 \\ & 0 \\ & 0 \\ 0\;0\;0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \underline{R}(\underline{x}_s, -90^\circ) & 0 \\ & 0 \\ & 0 \\ 0\;0\;0 & 1 \end{bmatrix} \begin{bmatrix} \underline{x}_{W2} & \underline{y}_{W2} & \underline{z}_{W2} & \underline{p}_{W2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

so that

$$\underline{x}_{W2}\Big|_{final} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \underline{y}_{W2}\Big|_{final} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \quad \underline{z}_{W2}\Big|_{final} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \quad \underline{p}_{W2}\Big|_{final} = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}.$$

Initially, $\underline{p}_{W2}$ indicates that the origin of the coordinate frame of W2 is 5 units away from the origin of coordinates $[\underline{x}_s, \underline{y}_s, \underline{z}_s]$ along the positive $\underline{y}_s$ direction. Finally the origin of the coordinate frame is 4 units away from the origin of $[\underline{x}_s, \underline{y}_s, \underline{z}_s]$ along the positive $\underline{z}_s$ direction. The final orientation is that $\underline{x}_{W2}$ aligns with $\underline{y}_s$, $\underline{y}_{W2}$ with negative $\underline{z}_s$, and $\underline{z}_{W2}$ with negative $\underline{x}_s$.

In the above discussion, it is seen that an operation often involves a sequence of rotations. Mathematically this is represented by the product of a corresponding sequence of matrices in a proper order. Since the product of the rotation of matrices is also a rotation matrix, then the sequence of rotations in base coordinates is equivalent to a single rotation of an appropriate angle θ about an appropriate axis of rotation $\underline{r}$ where $\underline{r}$ is a unit vector in the same base coordinates. Now the problem is to determine θ and $\underline{r}$.

To do this, let

$$\underline{A} = \begin{bmatrix} \underline{u} & \underline{v} & \underline{r} & \underline{o} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{where } \underline{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

be a coordinate frame with reference to base coordinates. $\underline{A}$ is obtained by a rotation of the base coordinates such that its z-axis will align with $\underline{r}$, the z-axis of $\underline{A}$. The directions of $\underline{u}$ and $\underline{v}$ are arbitrary provided $\underline{u}$, $\underline{v}$ and $\underline{r}$ are orthonormal such that $\underline{r} = \underline{u} \times \underline{v}$. Thus $\underline{A}^{-1} = \underline{A}'$, i.e. $\underline{A}$ inverse equals the transpose of $\underline{A}$. By definition, then, $\underline{A}$ transforms the z-axis of base coordinates into $\underline{r}$, or

$$\begin{bmatrix} \underline{r} \\ 1 \end{bmatrix} = \underline{A} \begin{bmatrix} \underline{z} \\ 1 \end{bmatrix} = \underline{A} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Conversely, $\underline{A}^{-1}$ transforms $\underline{r}$ into the z-axis of base coordinates, or

$$\begin{bmatrix} \underline{z} \\ 1 \end{bmatrix} = \underline{A}^{-1} \begin{bmatrix} \underline{r} \\ 1 \end{bmatrix} = \underline{A}' \begin{bmatrix} \underline{r} \\ 1 \end{bmatrix}$$

Hence a rotation of an angle θ about $\underline{r}$ in coordinate frame $\underline{A}$ is the same as a process of the following three steps:

(a) transforming the $\underline{A}$ frame back to the base coordinates (this involves an operation of $\underline{A}^{-1}$ or $\underline{A}'$);

(b) rotating about z-axis of base coordinates an angle θ, i.e., $\mathrm{R}(\underline{z},\theta)$; and

(c) transforming the resulting frame $\begin{bmatrix} \underline{R}(\underline{z},\theta) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0\ 0\ 0 & 1 \end{bmatrix} \underline{A}'$ back to the $\underline{A}$ frame (this involves an operation $\underline{A}$).

Mathematically, the equivalence relation can be expressed as

$$\begin{bmatrix} \underline{R}(\underline{r},\theta) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0\ 0\ 0 & 1 \end{bmatrix} = \underline{A} \begin{bmatrix} \underline{R}(\underline{z},\theta) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0\ 0\ 0 & 1 \end{bmatrix} \underline{A}'$$

$$= \begin{bmatrix} u_x & v_x & r_x & 0 \\ u_y & v_y & r_y & 0 \\ u_z & v_z & r_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ r_x & r_y & r_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The right side of the above equation involves multiplications of three 4 by 4 matrices. Through a tedious algebraic manipulation, one obtains, from above, a sum of a symmetrix and a skew symmetrix matrix for $\underline{R}(\underline{r},\theta)$:

$$\underline{R}(\underline{r},\theta) = \begin{bmatrix} r_x^2 & r_y r_x & r_z r_x \\ r_x r_y & r_y^2 & r_z r_y \\ r_x r_z & r_y r_z & r_z^2 \end{bmatrix} (1-\cos\theta) + \begin{bmatrix} \cos\theta & -r_z\sin\theta & r_y\sin\theta \\ r_z\sin\theta & \cos\theta & -r_x\sin\theta \\ -r_y\sin\theta & r_x\sin\theta & \cos\theta \end{bmatrix}$$

Now let $\underline{P} = [\underline{n}\ \underline{s}\ \underline{a}]$ be a product-rotation matrix which is the product of a sequence of rotation matrices in an appropriate order. By setting

$$\underline{P} = \underline{R}(\underline{r},\theta)$$

one specifies θ and $\underline{r}$ as the equivalent angle and axis of rotation, respectively, corresponding to the sequence of rotations mentioned above. To solve for θ, add the diagonal terms of the matrix of each side together to yield

$$n_x + s_y + a_z = (r_x^2 + r_y^2 + r_z^2)(1-\cos\theta) + 3\cos\theta$$

Since $\underline{r}$ is a unit vector, the above reduces to

$$\cos\theta = (n_x + s_y + a_z - 1)/2$$

Although this is a simple formula for computing the value of θ, an excessive error may result if the absolute value of the right side is close to unity. In addition, there is an ambiguity so far as the quadrant is concerned. An alternative choice is to take the differences of off-diagonal terms to yield three equations:

$$n_y - s_x = (r_x r_y - r_y r_x)(1-\cos\theta) + r_z\sin\theta - (-r_z\sin\theta)$$

$$= 2r_z\sin\theta$$

$$n_z - a_x = -2r_y\sin\theta$$

$$s_z - a_y = 2r_x \sin\theta$$

Since $\underline{r}$ is a unit vector, the components of $\underline{r}$ in the above three equations may be eliminated by first squaring each equation individually, and then adding them together. This yields

$$\sqrt{(n_y-s_x)^2+(n_z-a_x)^2+(s_z-a_y)^2}/2 = \sin\theta$$

where the positive sign of the square-root is chosen for $0 \leq \theta \leq 180°$. Likewise, the ambiguity in quadrant and excessive error when the left side having a very small value still exist. A third possibility is to compute the ratio of these two formulas:

$$\theta = \tan^{-1}\left[\sqrt{(n_y-s_x)^2+(n_z-a_x)^2+(s_z-a_y)^2}\right]/(n_x+s_y+a_z-1)$$

The quadrant is thus determined by the sign of the denominator.

Once θ is determined, the axis of rotation $\underline{r}$ may be determined from the three equations already derived by taking the differences of off-diagonal terms. This yields

$$r_x = (s_z-a_y)/(2\sin\theta)$$

$$r_y = (a_x-n_z)/(2\sin\theta)$$

$$r_z = (n_y-s_x)/(2\sin\theta)$$

One must make sure that $r_x^2+r_y^2+r_z^2 = 1$. This condition, however, may be difficult to satisfy when $\sin\theta$ is small. In this case, one may equate the diagonal terms of

$\underline{P} = \underline{R}(\underline{r},\theta)$ to yield

$$\begin{cases} n_x = r_x^2(1-\cos\theta)+\cos\theta \\ s_y = r_y^2(1-\cos\theta)+\cos\theta \\ a_z = r_z^2(1-\cos\theta)+\cos\theta \end{cases}$$

so that

$$\begin{cases} r_x = \sqrt{(n_x-\cos\theta)/(1-\cos\theta)} \\ r_y = \sqrt{(s_y-\cos\theta)/(1-\cos\theta)} \\ r_z = \sqrt{(a_z-\cos\theta)/(1-\cos\theta)} \end{cases}$$

Whenever the hand of the robot and the object have their own coordinate frames, it is a simple matter to describe mathematically the contact between them. Let

$$\underline{H} = \begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \underline{W} = \begin{bmatrix} \underline{x}w2 & \underline{y}w2 & \underline{z}w2 & \underline{p}w2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

be the coordinate frames for, respectively, the hand and the wedge W2 defined previously. Intuitively one would set $\underline{H} = \underline{W}$ to describe their rendezvous. This intuition, however, is false since $\underline{H}$ and $\underline{W}$ are defined with reference to two different coordinate systems. Thus to achieve the goal of describing the rendezvous, one must have the coordinate frames for the hand and the wedge referring to a same coordinate systems. Call this the world coordinates, and let the base coordinate frame $\begin{bmatrix} \underline{x}_o & \underline{y}_o & \underline{z}_o & \underline{p}_o \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and the scene coordinate frame $\begin{bmatrix} \underline{x}_s & \underline{y}_s & \underline{z}_s & \underline{p}_s \\ 0 & 0 & 0 & 1 \end{bmatrix}$ be defined in the world coordinates (see Figure 11).

Then $\begin{bmatrix} \underline{x}_o & \underline{y}_o & \underline{z}_o & \underline{p}_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$ specifies the hand in world coordinates while

$$\begin{bmatrix} \underline{x}_s & \underline{y}_s & \underline{z}_s & \underline{p}_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{x}w2 & \underline{y}w2 & \underline{z}w2 & \underline{p}w2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

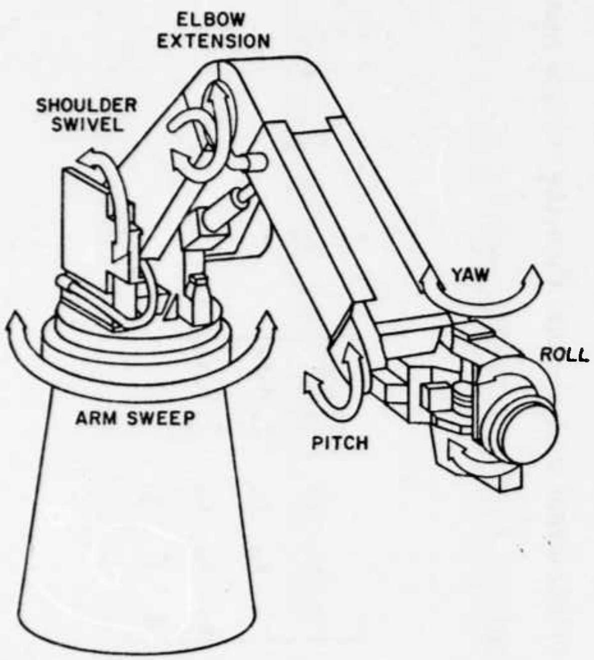
specifies the wedge in the same world coordinates. By equating these two expressions, one specifies the rendezvous. Usually the base coordinate, the scene coordinate and the wedge coordinate frames are either given or measured. Thus one may compute the required coordinate frame or location of the hand for rendezvous as

$$\begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \underline{x}_o & \underline{y}_o & \underline{z}_o & \underline{p}_o \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \underline{x}_s & \underline{y}_s & \underline{z}_s & \underline{p}_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{x}w2 & \underline{y}w2 & \underline{z}w2 & \underline{p}w2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

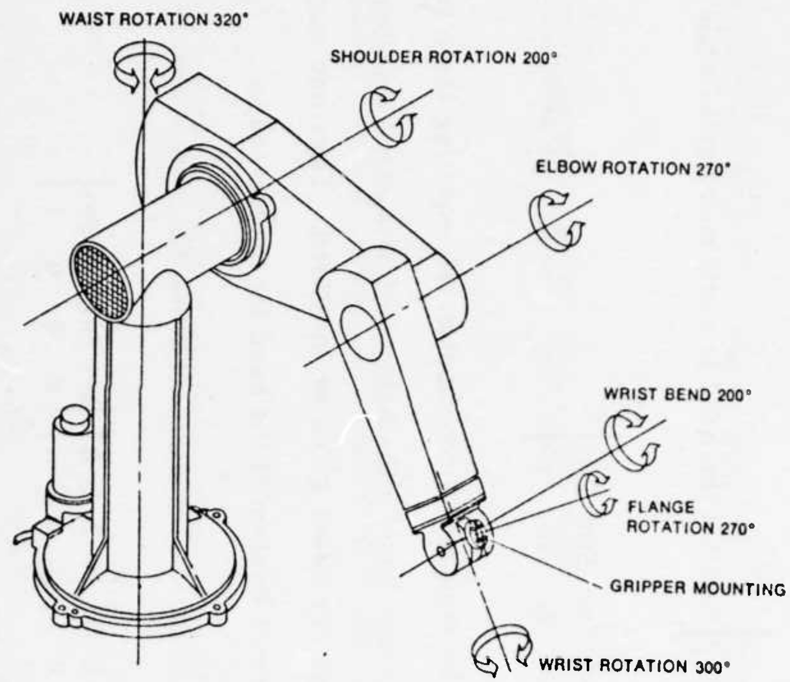
Since $\underline{x}_o$, $\underline{y}_o$ and $\underline{z}_o$ are orthonormal unit vectors, then the inverse matrix in the above equation may be computed as

$$\begin{bmatrix} x_{ox} & y_{ox} & z_{ox} & p_{ox} \\ x_{oy} & y_{oy} & z_{oy} & p_{oy} \\ x_{oz} & y_{oz} & z_{oz} & p_{oz} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} x_{ox} & x_{oy} & x_{oz} & -\sum_i p_{oi} x_{oi} \\ y_{ox} & y_{oy} & y_{oz} & -\sum_i p_{oi} y_{oi} \\ z_{ox} & z_{oy} & z_{oz} & -\sum_i p_{oi} z_{oi} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, it is the controller's task to drive the hand to the location $\begin{bmatrix} \underline{n} & \underline{s} & \underline{a} & \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$ to accomplish the rendezvous.



(a) Cincinnati Milacron T3



(b) Unimation PUMA 600

Figure 1. Examples of Industrial Robots.

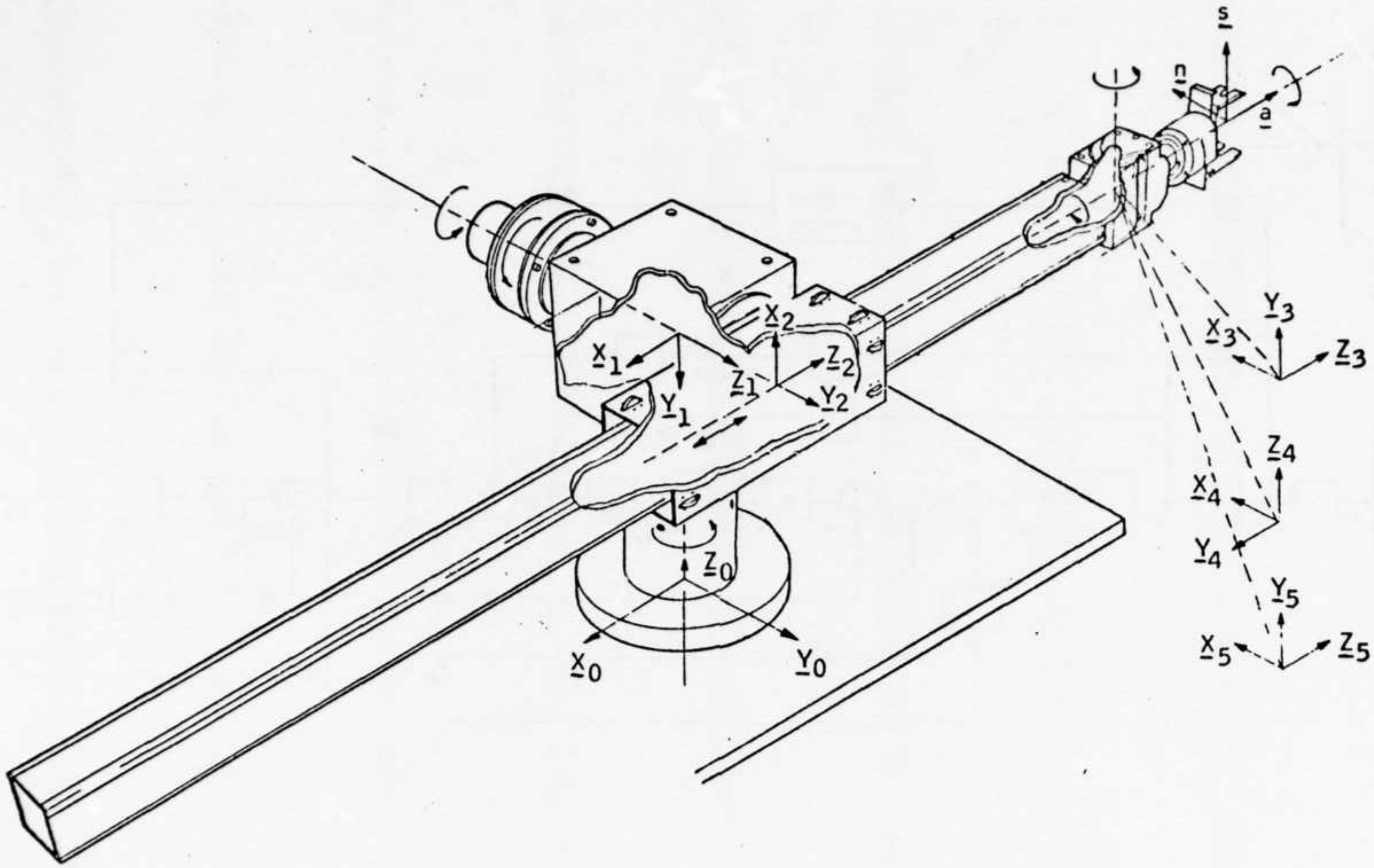


Figure 2. The Stanford Arm

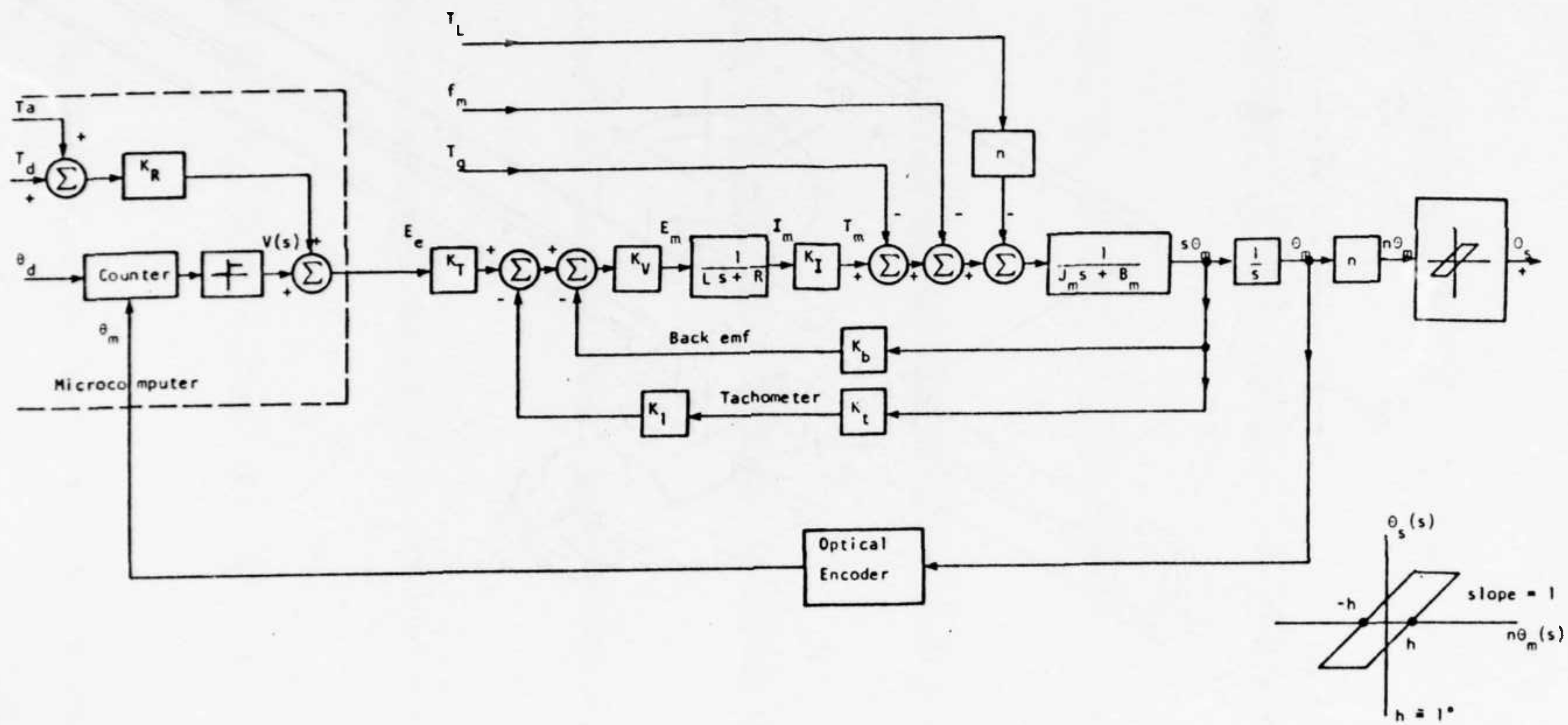


Figure 3. Block Diagram of a Positional Control System

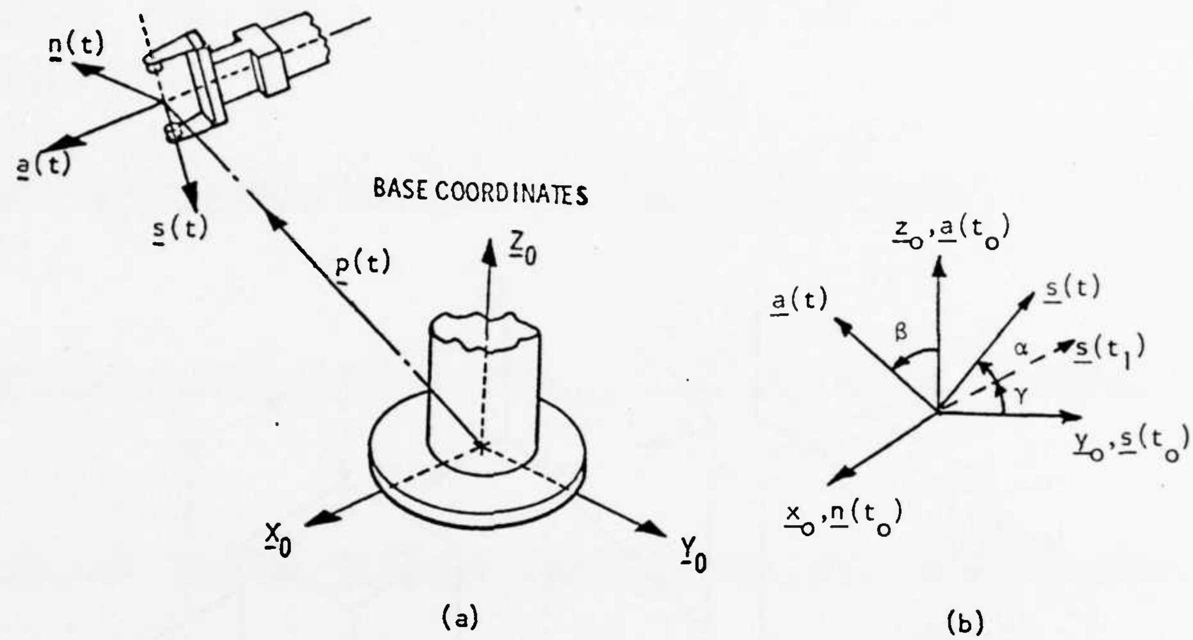


Figure 4. (a) Position and Orientation Vectors of the Hand
(b) Euler Angles of Orientation

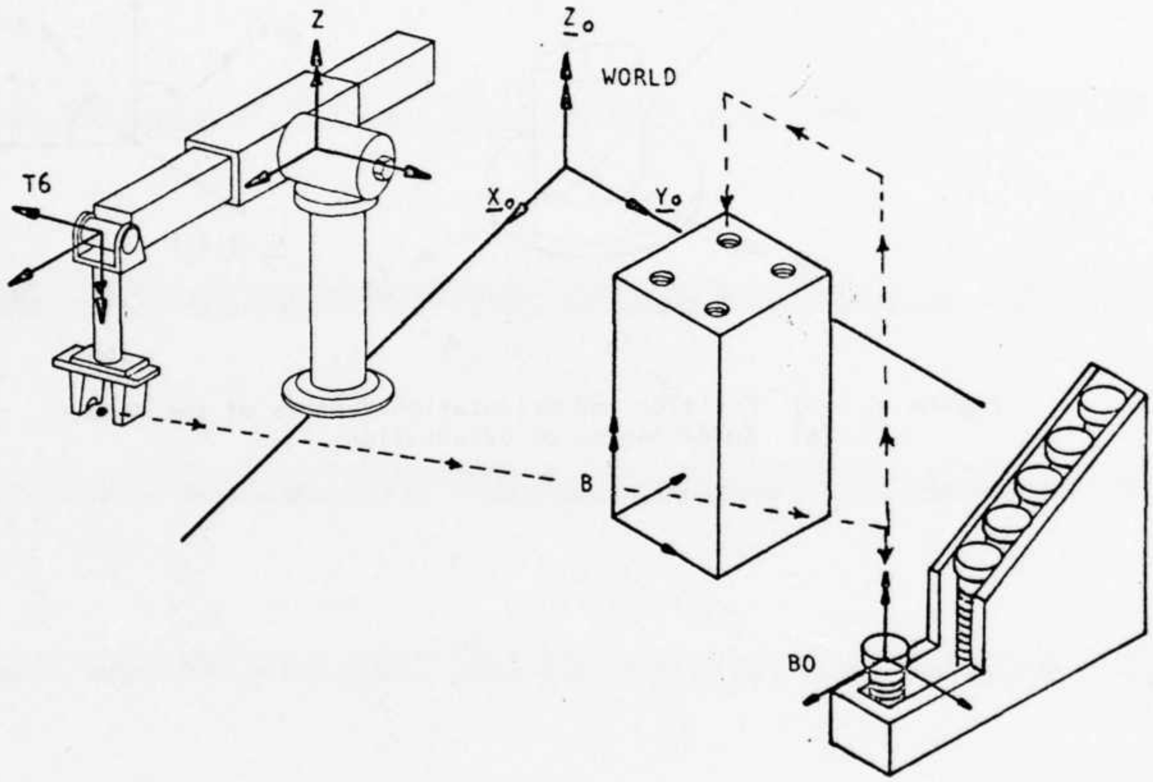


Figure 5. Simple Robot Task for Illustration

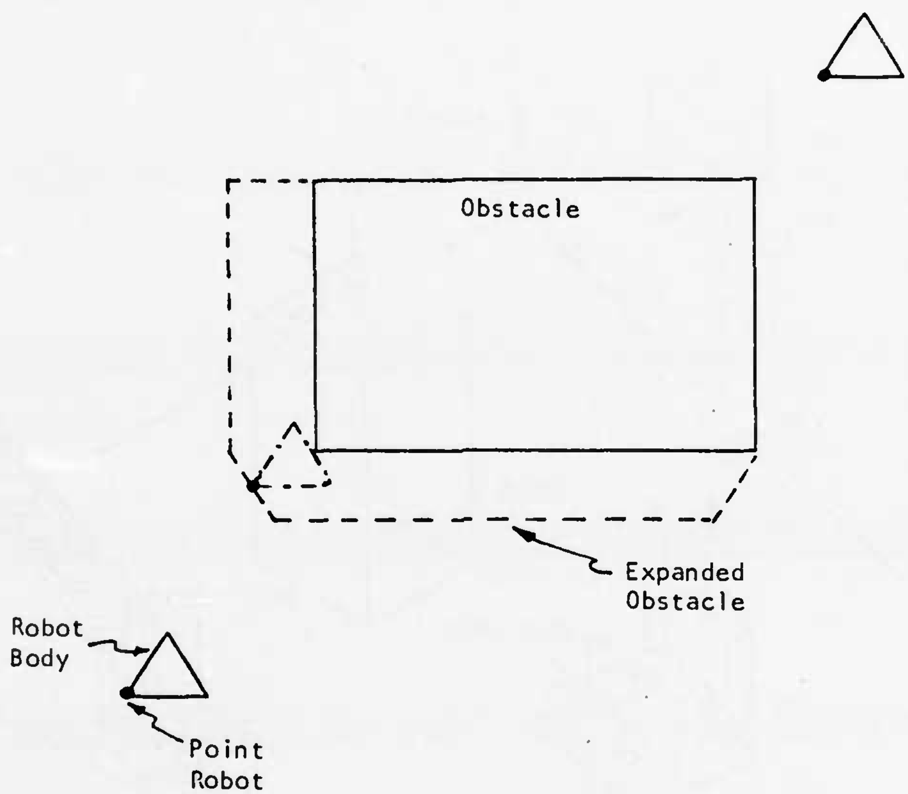


Figure 6. Robot Sliding Around the Obstacle with No Rotation

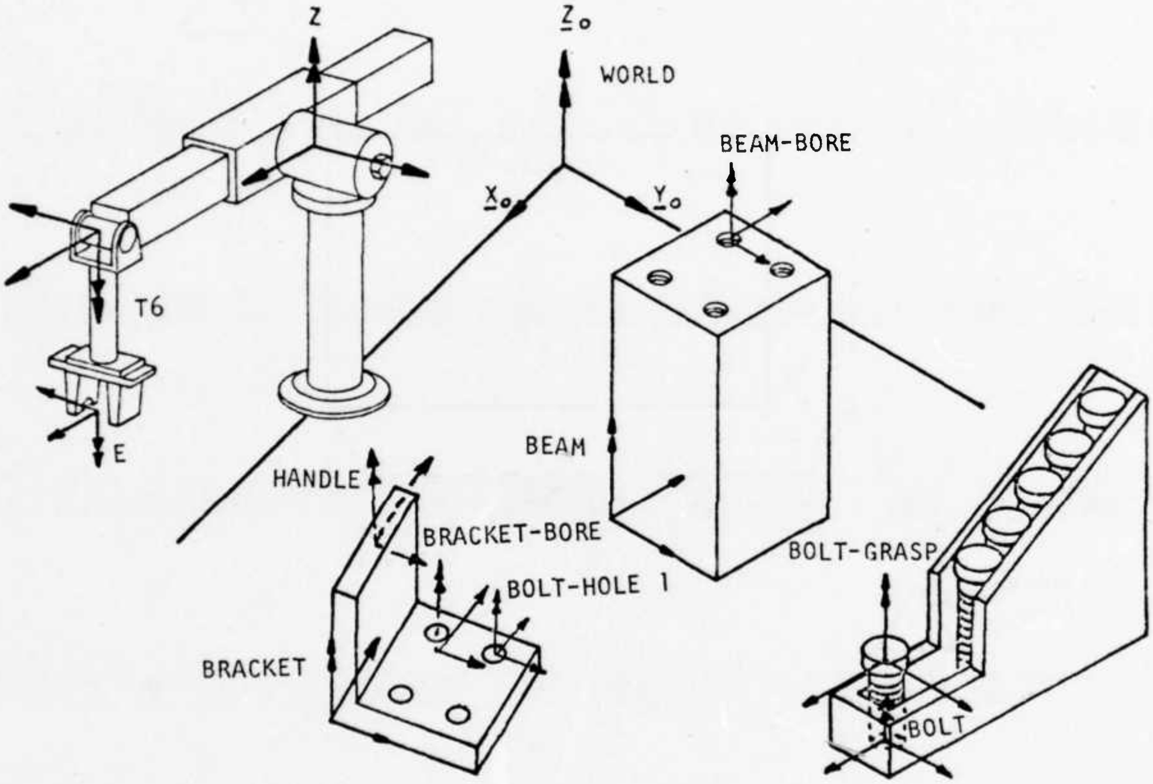


Figure 7. Bolting a Bracket Task

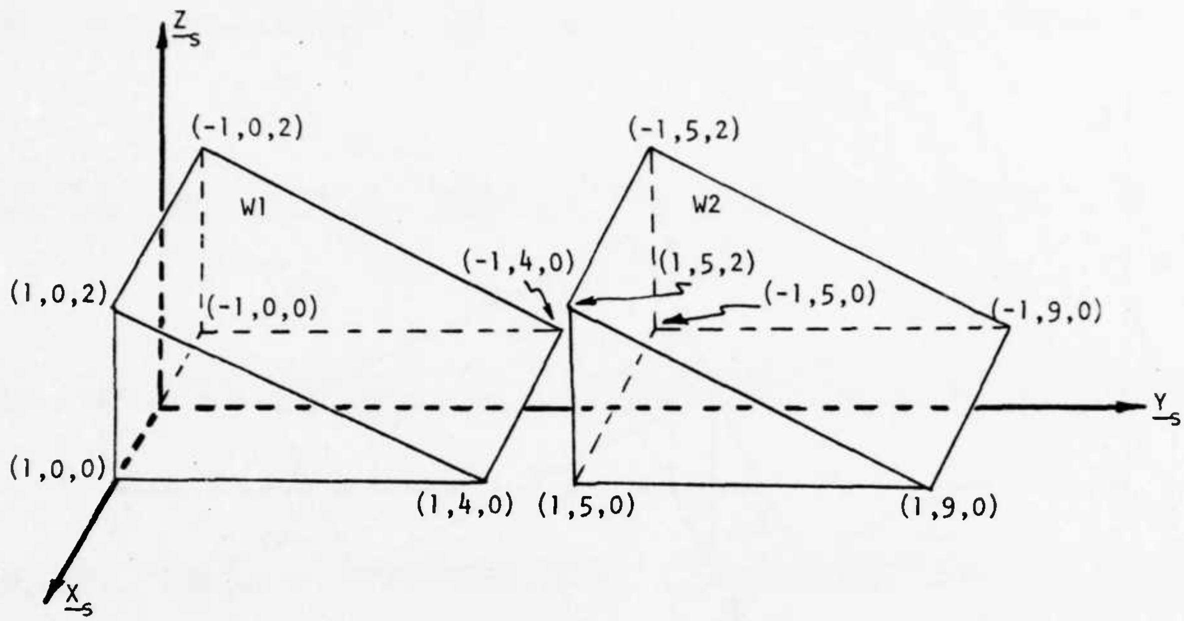


Figure 8. Initial Location of Two Identical Wedges.

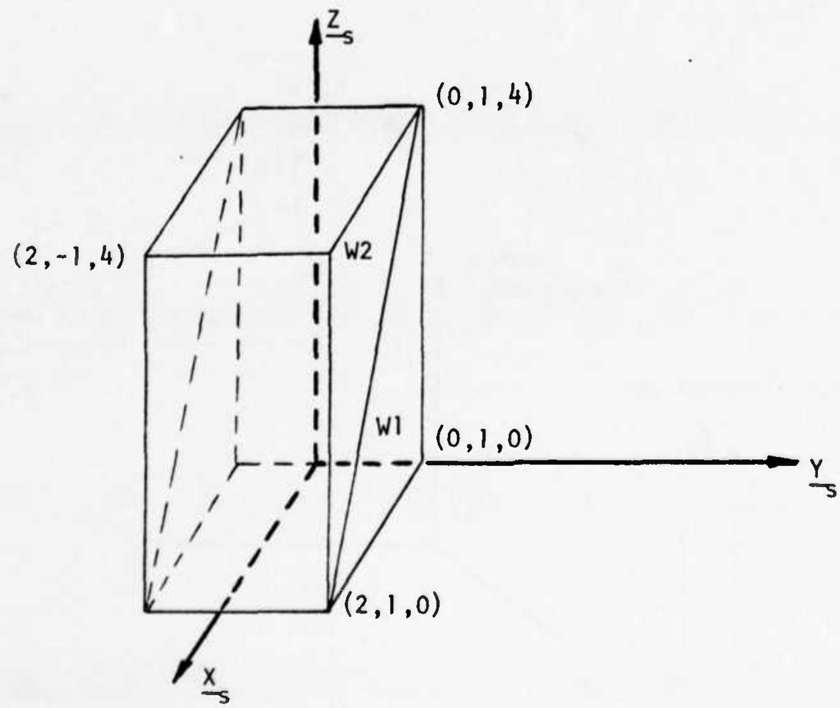


Figure 9. Required Location of Two Identical Wedges.

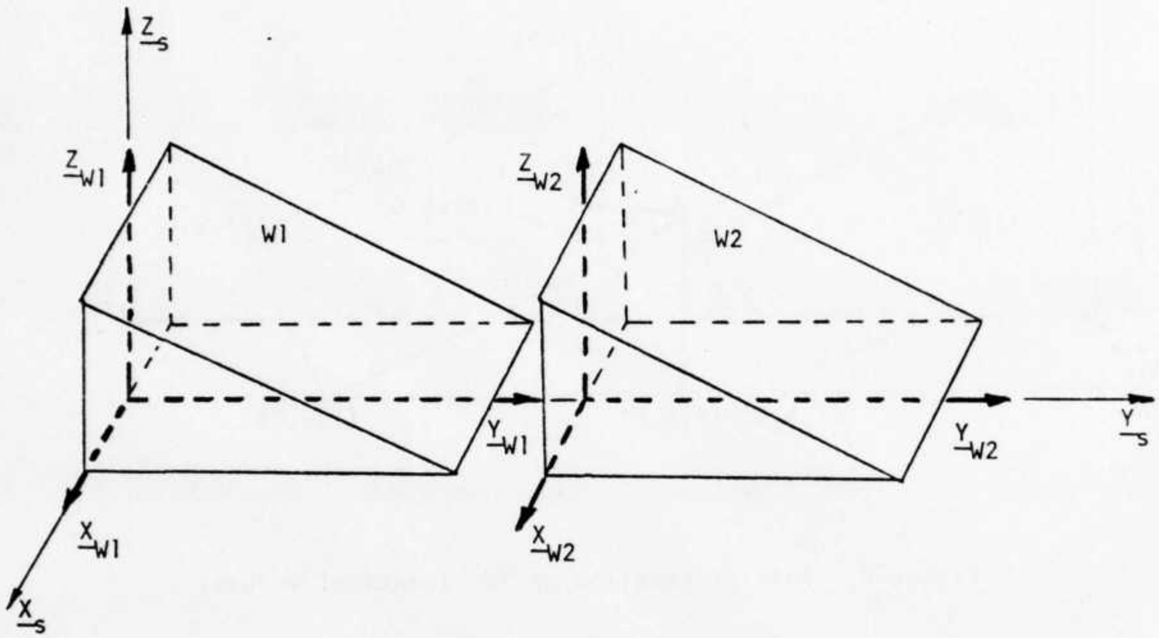


Figure 10. Assigned Coordinates for W1 and W2

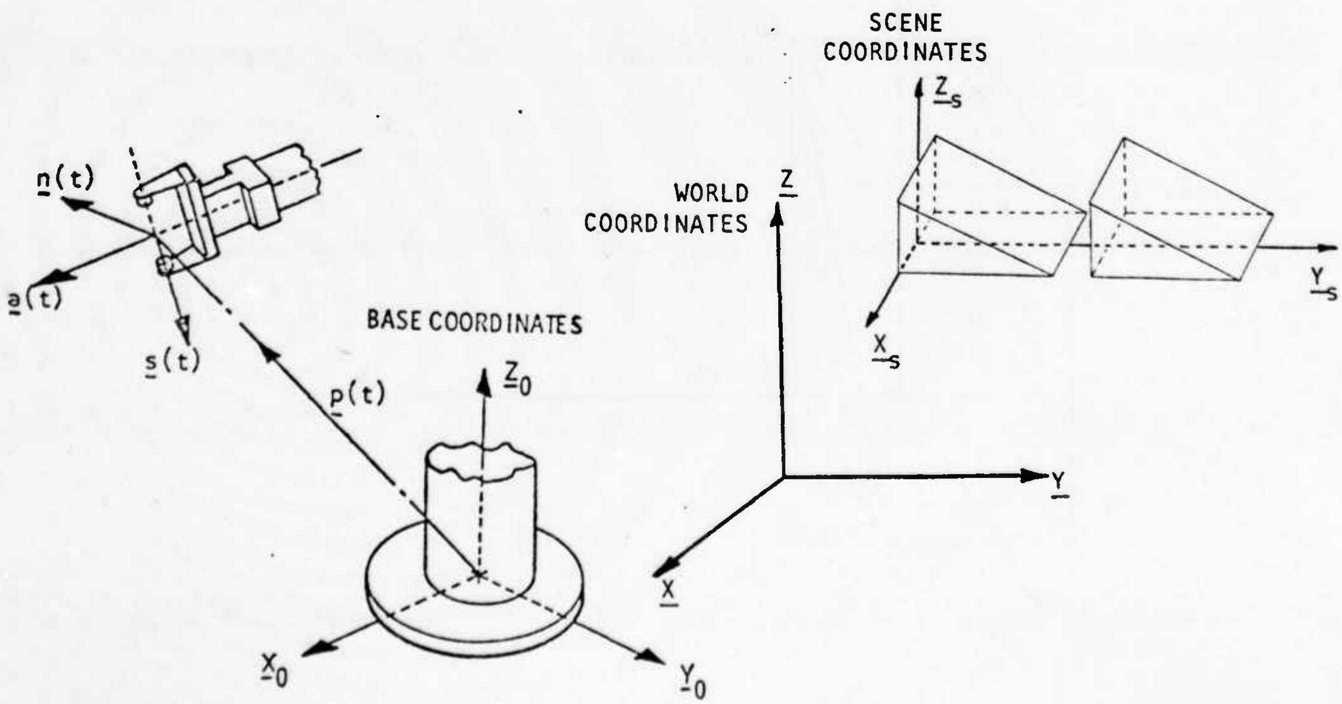


Figure 11. Scene and Base Coordinates with Reference to World Coordinates.

Kinematic Control Equations for Simple Manipulators

RICHARD P. PAUL, SENIOR MEMBER, IEEE, BRUCE SHIMANO, AND GORDON E. MAYER

Abstract—The basis for all advanced manipulator control is a relationship between the Cartesian coordinates of the end-effector and the manipulator joint coordinates. A direct method for assigning link coordinate systems and obtaining the end-effector position in terms of joint coordinates is reviewed. Techniques for obtaining the solution to these equations for kinematically simple manipulators, which includes all commercially available manipulators, are presented.

INTRODUCTION

A serial link manipulator consists of a sequence of mechanical links connected together by actuated joints. Such a structure forms a kinematic chain and may be analyzed by methods developed by Denavit and Hartenberg [10]. The results of this analysis are the matrix equations expressing manipulator end-effector Cartesian position and orientation in terms of the joint coordinates. These equations may be obtained for any manipulator independent of the number of links or degrees of freedom.

In this correspondence we first review the method of obtaining these equations extending the procedure of assigning coordinate frames to include simple manipulators which have many zero length links and intersecting joint axes. While we may obtain these kinematic equations for any manipulator it is their solution which is of interest. Given a desired Cartesian position and orientation of the manipulator's end-effector what are the necessary joint coordinates? While there is only one end-effector position corresponding to a given set of joint coordinates, there are a number of configurations of the manipulator's links all of which place the end-effector in the same position and orientation. Normally only one solution corresponding to a given kinematic configuration is desired (e.g., elbow up or down, etc.), rather than the entire set of solutions. Frequently the solution is to be embedded in a real-time servo loop and only a very minimum number of mathematical operations may be performed.

When the manipulator geometry is simple and well understood a trigonometric solution may often be obtained [1]–[3], [8], [9]. However, six-degree-of-freedom manipulators are sufficiently complex that the direct trigonometric method is too difficult to apply. We present a method of obtaining a solution to the kinematic equations based on the Hartenberg–Denavit matrices from which the solution is obtained explicitly in the case of simple manipulators. The existence of an explicit solution to the kinematic equations for any manipulator is of great importance in evaluating the manipulator's suitability for computer control. Iterative solution techniques can involve an order of magnitude and more computation than an explicit solution. Pieper [5] in his thesis considers a series of simple manipulators for which a closed-form solution is obtainable. It is to these "simple" manipulators that the solution method presented in this correspondence is applicable. We have solved the kinematic equations for all commercially available manipulators and find that the equations can be readily obtained in a matter of hours.

Manuscript received March 15, 1979; revised July 23, 1979; March 30, 1981. This material is based upon research supported by the National Science Foundation under Grants APR77-14533, APR75-13074, and APR74-01390. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

R. P. Paul is with the Advanced Technology Laboratory, GTE Laboratories Inc., 40 Sylvan Road, Waltham, MA 02254, on leave from the Department of Electrical Engineering, Purdue University, West Lafayette, IN 47905.

B. Shimano is with the West Coast Division of Unimation Inc., 5841A Uplander Way, Culver City, CA 90230.

G. E. Mayer is with Wright-Patterson AFB, AFWAL/MLTC, OH 45433.

0018-9472/81/0600-0449$00.75 ©1981 IEEE

Coordinate Frames

A serial link manipulator consists of a sequence of links connected together by actuated joints. For an n-degree-of-freedom manipulator, there will be n links and n joints. The base of the manipulator is link 0 and is not considered one of the six links. Link 1 is connected to the base link by joint 1. There is no joint at the end of the final link. The only significance of links is that they maintain a fixed relationship between the manipulator joints at each end of the link (7). Any link can be characterized by two dimensions: the common normal distance a_n, and α_n the angle between the axes in a plane perpendicular to a_n. It is customary to call a_n "the length" and α_n "the twist" of the link (see Fig. 1). Generally, two links are connected at each joint axis (see Fig. 2). The axis will have two normals connected to it, one for each link. The relative position of two such connected links is given by d_n, the distance between the normals along the joint n axis, and θ_n the angle between the normals measured in a plane normal to the axis. d_n and θ_n are called 'the distance" and "the angle" between the links, respectively.

In order to describe the relationship between links, we will assign coordinate frames to each link. We will first consider revolute joints in which θ_n is the joint variable. The origin of the coordinate frame of link n is set to be at the intersection of the common normal between joints n and $n+1$ and the axis of joint $n+1$. In the case of intersecting joint axes, the origin is at the point of intersection of the joint axes. If the axes are parallel, the origin is chosen to make the joint distance zero for the next link whose coordinate origin is defined. The z axis for link n shall be aligned with the axis of joint $n+1$. The x axis will be aligned with any common normal which exists and is directed along the normal from joint n to joint $n+1$. In the case of intersecting joints, the direction of the x axis is parallel or antiparallel to the vector cross product $z_{n-1} \times z_n$. Notice this condition is also satisfied for the x axis directed along the normal between joints n and $n+1$. For the nth revolute joint when x_{n-1} and x_n are parallel and have the same direction, θ_n is at its zero position.

In the case of a prismatic joint the distance d_n is the joint variable. The direction of the joint axis is the direction in which the joint moves. Although the direction of the axis is defined, unlike a revolute joint, its position in space is not defined (see Fig. 3). In the case of a prismatic joint the length a_n has no meaning and is set to zero. The origin of the coordinate frame for a prismatic joint is coincident with the next defined link origin. The z axis of the prismatic link is aligned with the axis of joint $n+1$. The x_n axis is parallel or antiparallel to the vector cross product of the direction of the prismatic joint and z_n. For a prismatic joint, we will define its zero position, with $d_i = 0$, to be when x_{n-1} and x_n intersect. With the manipulator in its zero position, the positive sense of rotation for revolute joints or displacement for prismatic joints can be decided and the sense of the direction of the z axes determined.

The origin of the base link (zero) will be coincident with the origin of link 1. If it is desired to define a different reference coordinate system then the relationship between the reference and base coordinate systems can be described by a fixed homogeneous transformation [6]. At the end of the manipulator the final displacement d_6 or rotation θ_6 occurs with respect to z_5. The origin of the coordinate system for link 6 is chosen to be coincident with that of the link 5 coordinate system. If a tool or end-effector is used whose origin and axes do not coincide with the coordinate system of link 6, the tool can be related by a fixed homogeneous transformation to link 6.

Having assigned coordinate frames to all links according to the preceding scheme, we can establish the relationship between successive frames $n-1$, n by the following rotations and translations.

Rotate about z_{n-1}, an angle θ_n.
Translate along z_{n-1}, a distance d_n.

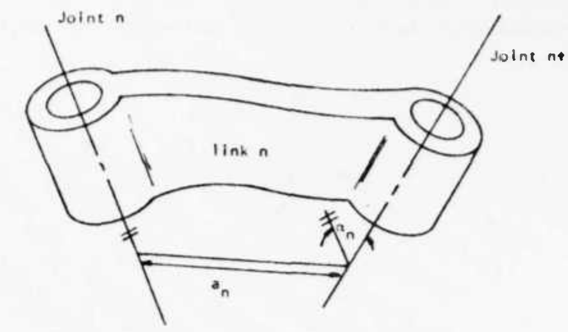


Fig. 1. Length a, and twist α, of a link.

Translate along rotated $x_{n-1} = x_n$, a length a_n.
Rotate about x_n, the twist angle α_n.

This may be expressed as the product of four homogeneous transformations relating the coordinate frame of link n to the coordinate frame of link $n-1$. This relationship is called an A matrix:

$$A_n = \begin{bmatrix} C\theta & -S\theta C\alpha & S\theta S\alpha & aC\theta \\ S\theta & C\theta C\alpha & -C\theta S\alpha & aS\theta \\ 0 & S\alpha & C\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where S and C refer to sine and cosine, respectively. For a prismatic joint the A matrix reduces to

$$A_n = \begin{bmatrix} C\theta & -S\theta C\alpha & S\theta S\alpha & 0 \\ S\theta & C\theta C\alpha & -C\theta S\alpha & 0 \\ 0 & S\alpha & C\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

Once the link coordinate frames have been assigned to the manipulator the various constant link parameters can be tabulated: d, a, and α for a link following a revolute joint and, θ and α for a link following a prismatic joint. Based on these parameters, the constant sine and cosine values of α may be evaluated and the values for the six A_i transformation matrices determined.

Kinematic Equations

Having assigned coordinate frames to a manipulator it is possible to obtain the Cartesian position and orientation of the manipulator end-effector when given the joint coordinates.

The description of the end of the manipulator, link coordinate frame 6, with respect to link coordinate frame $n-1$ is given by U_n where

$$U_n = A_n \cdot A_{n+1} \cdot \cdots \cdot A_6. \tag{3}$$

The end of the manipulator with respect to the base, known as T_6, is given by U_1:

$$T_6 = U_1 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6. \tag{4}$$

If the manipulator is related to a reference coordinate frame by a transformation Z and has a tool attached to its end described by E, we have the description of the end of the tool with respect to the reference coordinate system described by X as follows (4):

$$X = Z \cdot T_6 \cdot E. \tag{5}$$

In Fig. 4 the PUMA arm (Unimate 600 Robot) is shown with coordinate frames assigned to the links. The parameters are shown in Table I.

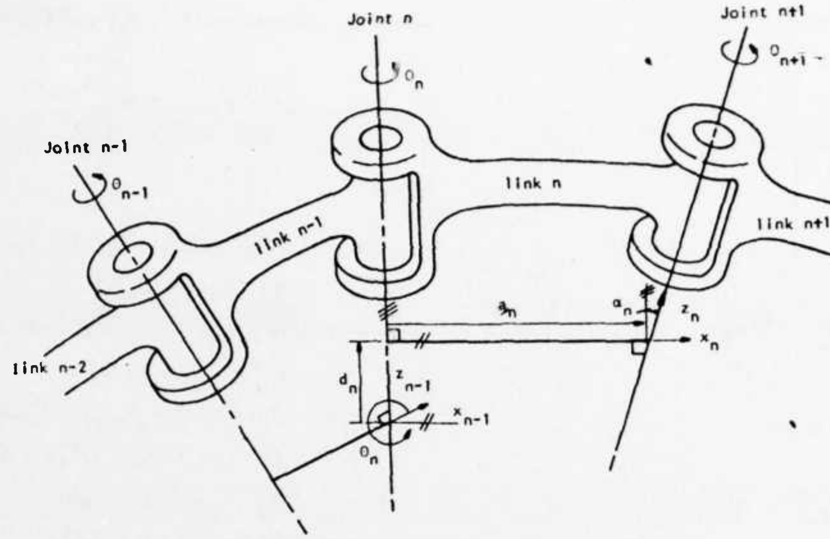


Fig. 2. Link parameters θ, d, a, α.

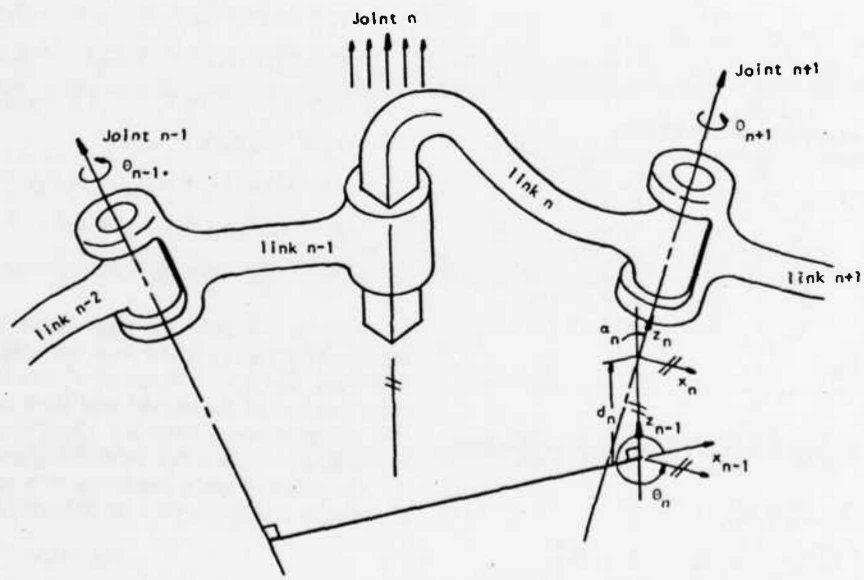


Fig. 3. Link parameters d, α for prismatic joint.

The A matrices for the PUMA arm are as follows:

$$A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$A_3 = \begin{bmatrix} C_3 & 0 & S_3 & a_3C_3 \\ S_3 & 0 & -C_3 & a_3S_3 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$A_4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$A_5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

where S_i refers to $\sin(\theta_i)$ and C_i refers to $\cos(\theta_i)$. The product of

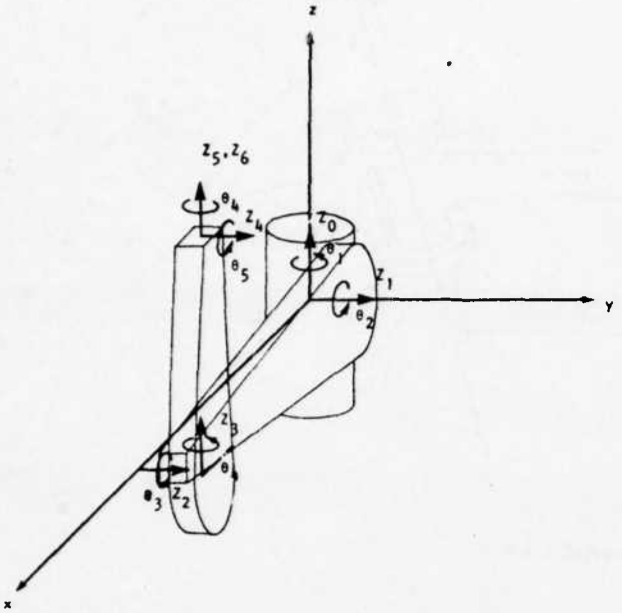


Fig. 4. PUMA manipulator.

TABLE I
LINK PARAMETERS FOR PUMA ARM

Joint	$\alpha°$	$\theta°$	d	a	Range
1	$-90°$	θ_1	0	0	θ_1: +/−160°
2	0	θ_2	0	a_2	θ_2: +45 → −225°
3	90°	θ_3	d_3	a_3	θ_3: 225° → −45°
4	$-90°$	θ_4	d_4	0	θ_4: +/−170°
5	90°	θ_5	0	0	θ_5: +/−135°
6	0	θ_6	0	0	θ_6: +/−170°

$a_2 = 17.000$ $a_3 = 0.75$
$d_3 = 4.937$ $d_4 = 17.000$

the A matrices, starting at link 6 and working back to the base, for the PUMA arm are

$$U_6 = A_6 \tag{12}$$

$$U_5 = A_5 U_6 = \begin{bmatrix} C_5C_6 & -C_5S_6 & S_5 & 0 \\ S_5C_6 & -S_5S_6 & -C_5 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$U_4 = A_4 U_5 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 & 0 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & 0 \\ -S_5C_6 & S_5S_6 & C_5 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

$$U_3 = A_3 U_4 = \begin{bmatrix} C_3(C_4C_5C_6 - S_4S_6) - S_3S_5C_6 & -C_3(C_4C_5S_6 + S_4C_6) + S_3S_5S_6 & C_3C_4S_5 + S_3C_5 & d_4S_3 + a_3C_3 \\ S_3(C_4C_5C_6 - S_4S_6) + C_3S_5C_6 & -S_3(C_4C_5S_6 + S_4C_6) - C_3S_5S_6 & S_3C_4S_5 - C_3C_5 & -d_4C_3 + a_3S_3 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$U_2 = A_2 U_3 =$

$$\begin{bmatrix} C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6 & -C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6 & C_{23}C_4S_5 + S_{23}C_5 & d_4S_{23} + a_3C_{23}a_2C_2 \\ S_{23}(C_4C_5C_6 - S_4S_6) + C_{23}S_5C_6 & -S_{23}(C_4C_5S_6 + S_4C_6) - C_{23}S_5S_6 & S_{23}C_4S_5 - C_{23}C_5 & -d_4C_{23} + a_3S_{23} + a_2S_2 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

where S_{23} refers to $\sin(\theta_2 + \theta_3)$ and C_{23} refers to $\cos(\theta_2 + \theta_3)$.

$$U_1 = A_1 U_5 = T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

where

$$n_x = C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_1[S_4C_5C_6 + C_4S_6] \tag{18}$$

$$n_y = S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + C_1[S_4C_5C_6 + C_4S_6] \tag{19}$$

$$n_z = -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6 \tag{20}$$

$$o_x = C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] - S_1[-S_4C_5S_6 + C_4C_6] \tag{21}$$

$$o_y = S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] + C_1[-S_4C_5S_6 + C_4C_6] \tag{22}$$

$$o_z = S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6 \tag{23}$$

$$a_x = C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5 \tag{24}$$

$$a_y = S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5 \tag{25}$$

$$a_z = -S_{23}C_4S_5 + C_{23}C_5 \tag{26}$$

$$p_x = C_1(d_4S_{23} + a_3C_{23} + a_2C_2) - S_1d_3 \tag{27}$$

$$p_y = S_1(d_4S_{23} + a_3C_{23} + a_2C_2) + C_1d_3 \tag{28}$$

$$p_z = -(-d_4C_{23} + a_3S_{23} + a_2S_2). \tag{29}$$

In order to compute the right hand three columns of T_6, we require 12 transcendental function calls, 34 multiplies, and 16 additions. The first column of T_6 can be obtained as the vector cross product of the second and third columns.

If the joint coordinates are given, the position and orientation of the hand are obtained by evaluating these equations to obtain T_6. The position and orientation of a tool with respect to a base coordinate frame can now be obtained from (5).

SOLUTION

In order to control the manipulator, we are interested in the reverse problem, that is, given X in (5), what are the corresponding joint coordinates?

We may first obtain T_6 from (5) as

$$T_6 = Z^{-1} \cdot X \cdot E^{-1} \tag{30}$$

and then the traditional approach is to solve the matrix equation

$$T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \tag{31}$$

where T_6 is given numeric values. With numeric values assigned to the elements of T_6, the required values of $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$, and θ_6 can be obtained by simultaneously solving (18)–(29). This approach is difficult for the following reasons: the equations are transcendental; we will need both the sine and cosine in order to determine angles uniquely and accurately; the manipulator exhibits more than one solution for a given position; and we have twelve equations in six unknowns.

There are, however, six other matrix equations obtained by successively premultiplying (31) by the A matrix inverses:

$$A_1^{-1} * T_6 = U_2 \tag{32}$$

$$A_2^{-1} * A_1^{-1} * T_6 = U_3 \tag{33}$$

$$A_3^{-1} * A_2^{-1} * A_1^{-1} * T_6 = U_4 \tag{34}$$

$$A_4^{-1} * A_3^{-1} * A_2^{-1} * A_1^{-1} * T_6 = U_5 \tag{35}$$

$$A_5^{-1} * A_4^{-1} * A_3^{-1} * A_2^{-1} * A_1^{-1} * T_6 = U_6. \tag{36}$$

The matrix elements of the left sides of these equations are functions of the elements of T_6 and of the first $n-1$ joint variables. The matrix elements of the right hand sides are either zero, constants, or functions of the nth to 6th joint variables. As matrix equality implies element by element equality we obtain 12 equations from each matrix equation, that is, one equation for each of the components of the four vectors $\boldsymbol{n}$, $\boldsymbol{o}$, $\boldsymbol{a}$ and $\boldsymbol{p}$. Equating elements of these matrix equations frequently results in equations yielding joint variables explicitly. We will illustrate the various forms of these equations by developing the equations for the PUMA arm.

If we premultipy (31) by A_1^{-1} we obtain

$$A_1^{-1} * T_6 = A_2 * A_3 * A_4 * A_5 * A_6 \tag{37}$$

$$A_1^{-1} * T_6 = U_2. \tag{38}$$

The left side of (38) is given by

$$A_1^{-1} * T_6 = \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{39}$$

The inverse of a homogeneous transformation is simple to obtain (see Appendix I) and the product of these two matrices is

$$A_1^{-1} * T_6 = \begin{bmatrix} f_{11}(n) & f_{11}(o) & f_{11}(a) & f_{11}(p) \\ f_{12}(n) & f_{12}(o) & f_{12}(a) & f_{12}(p) \\ f_{13}(n) & f_{13}(o) & f_{13}(a) & f_{13}(p) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

where

$$f_{11} = C_1 x + S_1 y \tag{41}$$

$$f_{12} = -z \tag{42}$$

$$f_{13} = -S_1 x + C_1 y \tag{43}$$

and x, y, and z refer to components of the vectors given as arguments to f_{11}, f_{12}, and f_{13}, for example

$$f_{11}(n) = C_1 n_x + S_1 n_y. \tag{44}$$

The right side of (38) is obtained from (16) and is given by

$$U_2 = \begin{bmatrix} C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6 & -C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6 & C_{23}C_4S_5 + S_{23}C_5 & d_4S_{23} + a_3C_{23} + a_2C_2 \\ S_{23}(C_4C_5C_6 - S_4S_6) + C_{23}S_5C_6 & -S_{23}(C_4C_5S_6 + S_4C_6) - C_{23}S_5S_6 & S_{23}C_4S_5 - C_{23}C_5 & -d_4C_{23} + a_3S_{23} + a_2S_2 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{45}$$

All the elements on the right side of (45) are functions of θ_2, θ_{23}, d_4, θ_4, θ_5, and θ_6 except for element 34. We may equate the 34 elements to obtain

$$f_{13}(p) = d_3 \tag{46}$$

or

$$-S_1 p_x + C_1 p_y = -d_3. \tag{47}$$

In order to solve equations of this form we make the following trigonometric substitutions:

$$p_x = r\cos\phi \tag{48}$$

$$p_y = r\sin\phi \tag{49}$$

where

$$r = +\left(p_x^2 + p_y^2\right)^{1/2} \tag{50}$$

$$\phi = \tan^{-1}\left(\frac{p_y}{p_x}\right). \tag{51}$$

As either the numerator or denominator of (51) can be zero we will use the arctangent function of two arguments to obtain values of ϕ. This arctangent function uses the sign of the numerator and denominator to determine the correct quadrant for the resulting angle and is defined over the range $-\pi \leq \phi < \pi$. Substituting for p_x and p_y in (47) we obtain

$$S\phi C\theta_1 - C\phi S\theta_1 = d_3/r \tag{52}$$

with

$$0 < d_3/r \leq 1.$$

Equation (52) reduces to

$$S(\phi - \theta_1) = d_3/r \tag{53}$$

with

$$0 < \phi - \theta_1 < \pi.$$

We may obtain the cosine as

$$C(\phi - \theta_1) = \pm\sqrt{1 - (d_3/r)^2} \tag{54}$$

where the minus sign corresponds to a left-hand shoulder configuration of the manipulator and the plus sign corresponds to a right-hand shoulder configuration. Finally,

$$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) - \tan^{-1}\frac{d_3}{\pm\sqrt{r^2 - d_3^2}}. \tag{55}$$

Having determined θ_1 the left side of (38) is now defined. Whenever we have the left side of one of (32)–(36) defined, we examine the right side for elements which are a function of individual joint coordinates. In the case of the PUMA arm, as with any arm with two or more joint axes parallel, the T_6 matrix is expressed in terms of sums or differences of the angles relating to the parallel axes. In order to solve the kinematic equations, the sum or difference of the angles must be determined before the angles themselves can be found. In addition the solution for these sums of angles involves the sum of the squares of two equations. Such is the case in order to solve for θ_2 and θ_3. The 14 and 24 elements of (38) are

$$d_4S_{23} + a_3C_{23} + a_2C_2 = C_1p_x + S_1p_y \tag{56}$$

$$-d_4C_{23} + a_3S_{23} + a_2S_2 = -p_z \tag{57}$$

where

$$C_1 p_x + S_1 p_y = f_{11p} \tag{58}$$

$$-p_z = f_{12p}. \tag{59}$$

Squaring, adding, and simplifying:

$$f_{11p}^2 + f_{12p}^2 - d_4^4 - a_3^2 - a_2^2 = 2a_2 d_4 S_3 + 2a_2 a_3 C_3. \tag{60}$$

Since the left side is known and the only variables are S_3 and C_3, this equation is of the form of (47). It can be solved to yield

$$\theta_3 = \arctan\frac{a_3}{-d_4} - \arctan\frac{d}{\pm\sqrt{e-d^2}} \tag{61}$$

where

$$d = f_{11p}^2 + f_{12p}^2 - d_4^2 - a_3^2 - a_2^2 \tag{62}$$

$$e = 4a_2^2 a_3^2 + 4a_2^2 d_4^2 \qquad \text{(constant)}. \tag{63}$$

Evaluating the elements of (33) we obtain

$$\begin{bmatrix} f_{21}(n) & f_{21}(o) & f_{21}(a) & f_{21}(p) - a_2 \\ f_{22}(n) & f_{22}(o) & f_{22}(a) & f_{22}(p) \\ f_{23}(n) & f_{23}(o) & f_{23}(a) & f_{23}(p) \\ 0 & 0 & 0 & 1 \end{bmatrix} = U_3 \tag{64}$$

where

$$f_{21} = C_2(C_1 x + S_1 y) - S_2 z \tag{65}$$

$$f_{22} = -S_2(C_1 x + S_1 y) - C_2 z \tag{66}$$

$$f_{23} = -S_1 x + C_1 y. \tag{67}$$

Since this yields nothing, we evaluate (34) as

$$\begin{bmatrix} f_{31}(n) & f_{31}(o) & f_{31}(a) & f_{31}(p) - a_2 C_3 - a_3 \\ f_{32}(n) & f_{32}(o) & f_{32}(a) & f_{32}(p) + d_3 \\ f_{33}(n) & f_{33}(o) & f_{33}(a) & f_{33}(p) - a_2 S_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = U_4 \tag{68}$$

$$f_{31} = C_{23}(C_1 x + S_1 y) - S_{23} z \tag{69}$$

$$f_{32} = -S_1 x + C_1 y \tag{70}$$

$$f_{33} = S_{23}(C_1 x + S_1 y) + C_{23} z \tag{71}$$

equating the 14 and 34 terms we obtain

$$C_{23} f_{11p} - S_{23} p_z = a_2 C_3 + a_3 \tag{72}$$

$$S_{23} f_{11p} + C_{23} p_z = d_4 + a_2 S_3. \tag{73}$$

Since C_{23} and S_{23} are the only variables, we can solve the above equations simultaneously to yield:

$$S_{23} = \frac{w_2 f_{11p} - w_1 p_2}{f_{11p}^2 + p_z^2} \tag{74}$$

$$C_{23} = \frac{w_1 f_{11p} + w_2 p_z}{f_{11p}^2 + p_z^2} \tag{75}$$

where

$$w_1 = a_2 C_3 + a_3 \tag{76}$$

$$w_2 = d_4 + a_2 S_3 \tag{77}$$

therefore

$$\theta_{23} = \arctan\frac{w_2 f_{11p} - w_1 p_z}{w_1 f_{11p} + w_2 p_z} \tag{78}$$

and

$$\theta_2 = \theta_{23} - \theta_3. \tag{79}$$

With the left side of (68) now defined, we check the right side for functions of single variables. The 13 and 23 elements give us equations for the sine and cosine of θ_4 if $\sin(\theta_5)$ is not zero. When $\sin\theta_5 = 0$, $\theta_5 = 0$ and the manipulator becomes degenerate with both the axes of joint 4 and joint 6 aligned. In this state it is only the sum of θ_4 and θ_5 which is significant. If θ_5 is zero we are free to choose any value for θ_4. The current value is frequently assigned:

$$C_4 S_5 = C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z \tag{80}$$

$$S_4 S_5 = -S_1 a_x + C_1 a_y \tag{81}$$

and

$$\theta_4 = \tan^{-1}\frac{-S_1 a_x + C_1 a_y}{C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z} \tag{82}$$

if

$$\theta_5 > 0$$

and

$$Q_4 = \theta_4 + 180^\circ \quad \text{if } \theta_5 < 0. \tag{83}$$

Evaluating the elements of (35) we obtain

$$\begin{bmatrix} f_{41}(n) & f_{41}(o) & f_{41}(a) & 0 \\ f_{42}(n) & f_{42}(o) & f_{42}(a) & 0 \\ f_{43}(n) & f_{43}(o) & f_{43}(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_5 C_6 & -C_5 S_6 & S_5 & 0 \\ S_5 C_6 & -S_5 S_6 & -C_5 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{84}$$

where

$$f_{41} = C_4[C_{23}(C_1 x + S_1 y) - S_{23} z] + S_4[-S_1 x + C_1 y] \tag{85}$$

$$f_{42} = -S_{23}(C_1 x + S_1 y) - C_{23} z \tag{86}$$

$$f_{43} = -S_4[C_{23}(C_1 x + S_1 y) - S_{23} z] + C_4[-S_1 x + C_1 y]. \tag{87}$$

From the right side of (84), we can then obtain equations for S_5, C_5, S_6 and C_6 by inspection. When both sine and cosine are defined we obtain a unique value for the joint angle. We obtain a value for θ_5 by equating the 13 and 23 elements of (84):

$$S_5 = C_4[C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z] + S_4[-S_1 a_x + C_1 a_y] \tag{88}$$

$$C_5 = S_{23}(C_1 a_x + S_1 a_y) + C_{23} a_z \tag{89}$$

and obtain θ_5 as

$$\theta_5 = \tan^{-1}\frac{C_4[C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z] + S_4[-S_1 a_x + C_1 a_y]}{S_{23}(C_1 a_x + S_1 a_y) + C_{23} a_z} \tag{90}$$

While we have equations for both S_6 and C_6, the equation for S_6 is in terms of elements of the first column which involves the use of the $\boldsymbol{n}$ vector of T_6. The $\boldsymbol{n}$ vector of T_6 is not usually made available as it represents redundant information. It can always be computed by the vector cross product of the $\boldsymbol{o}$ and $\boldsymbol{a}$ vectors. By evaluating the elements of (36) we can obtain equations for S_6 and C_6 as a function of the $\boldsymbol{o}$ vector:

$$\begin{bmatrix} f_{51}(n) & f_{51}(o) & 0 & 0 \\ f_{52}(n) & f_{52}(o) & 0 & 0 \\ f_{53}(n) & f_{53}(o) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{91}$$

where

$$f_{51} = C_5\{C_4[C_{23}(C_1x + S_1y) - S_{23}z] + S_4[-S_1x + C_1y]\} + S_5\{-S_{23}(C_1x + S_1y) - C_{23}z\} \quad (92)$$

$$f_{52} = -S_4[C_{23}(C_1x + S_1y) - S_{23}z] + C_4[-S_1x + C_1y] \quad (93)$$

$$f_{53} = S_5\{C_4[C_{23}(C_1x + S_1y) - S_{23}z] + S_4[-S_1x + C_1y]\} + C_5\{S_{23}(C_1x + S_1y) + C_{23}z\}. \quad (94)$$

By equating the 12 and 22 elements we obtain expressions for S_6 and C_6:

$$S_6 = -C_5\{C_4[C_{23}(C_1o_x + S_1o_y) - S_{23}o_z] + S_4[-S_1o_x + C_1o_y]\} + S_5\{S_{23}(C_1o_x + S_1o_y) + C_{23}o_z\} \quad (95)$$

$$C_6 = -S_4[C_{23}(C_1o_x + S_1o_y) - S_{23}o_z] + C_4[-S_1o_x + C_1o_y]. \quad (96)$$

We obtain an equation for θ_6 as:

$$\theta_6 = \tan^{-1}\frac{-C_5\{C_4[C_{23}(C_1o_x + S_1o_y) - S_{23}o_z] + S_4[-S_1o_x + C_1o_y]\} + S_5\{S_{23}(C_1o_x + S_1o_y) + C_{23}o_z\}}{-S_4[C_{23}(C_1o_x + S_1o_y) - S_{23}o_z] + C_4[-S_1o_x + C_1o_y]}. \quad (97)$$

Even in the case where θ_4 is undefined because the manipulator configuration is degenerate, once a value is assigned to θ_4 the correct values for θ_5 and θ_6 are determined by these equations. This solution corresponds to 16 transcendental function calls, 38 multiplies, and 25 additions.

Extension to Other Manipulators

This solution technique, demonstrated with the PUMA manipulator, is valid for kinematically simple manipulators, including all commercially available manipulators for which solutions have been obtained. There are, however, some manipulators whose configurations mandate a slightly different approach to the solution. In the case of a manipulator with an offset at the hand, the problem was inverted and the solution to the kinematic problem to position the base at T_6^{-1} was solved.

There are two common pitfalls in obtaining solutions which should be avoided. One of these is division by the sine or cosine of an angle. The other is not maximizing the use of common expressions. For example, after solving for θ_4 from (82), a possible method to determine θ_5 would be to equate the 2,3 and 3,3 elements of (68). In order to do this, the 2,3 element (S_4S_5) would have to be divided by S_4. This leads to inaccuracy when S_4 is near or equal to zero. By extending the method one more step and premultiplying by A_3^{-1} both problems were avoided.

Summary

We have reviewed the method of assigning coordinate frames to the links of a manipulator. In terms of these coordinate frames the kinematic equations can be developed in a straightforward manner. These equations can be obtained for any manipulator. If the manipulator is kinematically "simple," the solution to the kinematic equations can be obtained in a very straightforward, error-free manner.

Appendix I

Given a homogeneous transformation represented by four vectors $\boldsymbol{l}$, $\boldsymbol{m}$, $\boldsymbol{n}$, and $\boldsymbol{p}$

$$T = \begin{bmatrix} l_x & m_x & n_x & p_x \\ l_y & m_y & n_y & p_y \\ l_z & m_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (98)$$

Its inverse is given by

$$T^{-1} = \begin{bmatrix} l_x & l_y & l_z & -\boldsymbol{p}\cdot\boldsymbol{l} \\ m_x & m_y & m_z & -\boldsymbol{p}\cdot\boldsymbol{m} \\ n_x & n_y & n_z & -\boldsymbol{p}\cdot\boldsymbol{n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (99)$$

where the terms of the right-hand column are obtained using vector dot product. That (99) represents the inverse is easily verified by forming the matrix product and checking that the result is an identity matrix:

$$T^{-1} * T = \begin{bmatrix} l_x & l_y & l_z & -\boldsymbol{p}\cdot\boldsymbol{l} \\ m_x & m_y & m_z & -\boldsymbol{p}\cdot\boldsymbol{m} \\ n_x & n_y & n_z & -\boldsymbol{p}\cdot\boldsymbol{n} \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} l_x & m_x & n_x & p_x \\ l_y & m_y & n_y & p_y \\ l_x & m_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (100)$$

$$T^{-1} * T = \begin{bmatrix} \boldsymbol{l}\cdot\boldsymbol{l} & \boldsymbol{l}\cdot\boldsymbol{m} & \boldsymbol{l}\cdot\boldsymbol{m} & 0 \\ \boldsymbol{m}\cdot\boldsymbol{l} & \boldsymbol{m}\cdot\boldsymbol{m} & \boldsymbol{m}\cdot\boldsymbol{n} & 0 \\ \boldsymbol{n}\cdot\boldsymbol{l} & \boldsymbol{n}\cdot\boldsymbol{m} & \boldsymbol{n}\cdot\boldsymbol{n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (101)$$

As the three vectors $\boldsymbol{l}$, $\boldsymbol{m}$, and $\boldsymbol{n}$ are orthogonal we have

$$\boldsymbol{l}\cdot\boldsymbol{l} = \boldsymbol{m}\cdot\boldsymbol{m} = \boldsymbol{n}\cdot\boldsymbol{n} = 1 \quad (102)$$

and

$$\boldsymbol{l}\cdot\boldsymbol{m} = \boldsymbol{l}\cdot\boldsymbol{n} = \boldsymbol{n}\cdot\boldsymbol{m} = 0 \quad (103)$$

and thus (101) reduces to an identity matrix.

Acknowledgment

Mickey Krebs was responsible for the document preparation and Marc Ream for preparing the drawings.

References

[1] R. A. Lewis, "Autonomous manipulations on a robot: Summary of manipulator software functions," Jet Propulsion Laboratories, Pasadena, CA, TM 33-679, 1974.
[2] C. Rosen *et al.*, "Exploratory research in advanced automation," 2nd Rep., Stanford Research Institute, Stanford Univ., Stanford, CA, Aug. 1974.
[3] R. Paul, "Modelling, trajectory calculation, and servoing of a computer controlled arm," Stanford Artificial Intelligence Lab., Stanford Univ., Stanford, CA, Memo. AIM-77, Nov. 1972.
[4] ——, "Advanced industrial robot control systems," 1st Rep. Purdue Univ., Lafayette, IN, Memo. EE 78-25, May 1978.
[5] D. L. Pieper, "The kinematics of manipulators under computer control," Stanford Artificial Intelligence Project, Stanford, CA, Memo. AIM-72, Oct. 1968.
[6] L. G. Roberts, "Homogeneous matrix representation and manipulation of *N*-dimensional constructs," M.I.T., Lincoln Labs., Document MS-1045, May 1965.
[7] B. Roth, "Performance evaluation of manipulators from a kinematic viewpoint," National Bureau Standards, Rep. SP-459, 1976.
[8] T. Binford *et al.*, "Exploratory studies of computer integrated assembly systems," National Science Foundation Progress Rep., Stanford Artificial Intelligence Lab., Standford, CA, Memo. AIM-285, July 1976.
[9] R. H. Taylor, "Planning and execution of straight-line manipulator trajectories," IBM Research Rep. RC 6657, July 1977.
[10] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, pp. 215–221, June 1955.

APPENDIX C

Conventional Controllers Design for Industrial Robots*

(A Tutorial)

J. Y. S. Luh
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

Industrial robots are serial link manipulators whose dynamic characteristics are highly nonlinear. By controlling each link or joint individually, it is possible to employ the conventional technique to design linear feedback controllers for the robot. The gravitational force and force interactions between joints are suppressed by precalculated feedforward compensation. To ease the computational burden, the compensating signals are either approximated or their computational formulas are simplified.

I. INTRODUCTION

The number of joints of industrial robots commercially available today varies from three to seven. Typically they have six joints, giving six degrees of freedom, with a gripper which is referred to as a hand or an end effector. Figures 1 and 2 show the commercially available Cincinnati Milacron Model T3, and Unimation PUMA 600. Figure 3 shows a Stanford manipulator [1] which is also an industrial robot existing among research institutes in the United States. Each joint of these robots is driven hydraulically, pneumatically or electrically with a feedback control loop. As an example, a block diagram for a joint control of the Stanford manipulator, which has a permanent magnet motor drive [2], is shown in Figure 4. It has an optical encoder for positional feedback with a tachometer feedback for damping. Thus, an industrial robot is a positioning device in that each of its joints

*Supported by NSF Grant DAR (APR 77-14533).

has a positional control system.

II. CARTESIAN AND JOINT COORDINATES

In reality a robot task is naturally specified in terms of its hand in Cartesian coordinates. It consists of position, described by a position vector $\underline{p}(t)$, and orientation, described by a unit approach vector $\underline{a}(t)$ and a unit sliding vector $\underline{s}(t)$, as indicated in Figure 5(a). All these vectors are defined with reference to the base coordinates. For convenience, a unit normal vector is defined as $\underline{n}(t) = \underline{s}(t) \times \underline{a}(t)$ where X denotes the "cross product." Thus the state of hand at time t in Cartesian coordinates can be represented by a 4 by 4 hand matrix

$$\underline{H}(t) = \begin{bmatrix} \underline{n}(t) & \underline{s}(t) & \underline{a}(t) & \underline{p}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The last row in $\underline{H}(t)$ is added for the convenience of future computation using homogeneous transformation [3]. The orientation may also be defined in terms of Euler angles with respect to the base coordinates. Initially at $t = t_0$, let $\underline{n}(t_0)$, $\underline{s}(t_0)$ and $\underline{a}(t_0)$ align with $\underline{x}_0$, $\underline{y}_0$ and $\underline{z}_0$, respectively, as shown in Figure 5(b). Any orientation $[\underline{n}(t)\ \ \underline{s}(t)\ \ \underline{a}(t)]$ may be obtained by a rotation of γ radians about $\underline{z}_0$ so that $\underline{s}(t_0)$ aligns with $\underline{s}(t_1)$; then a rotation of β radians about $\underline{s}(t_1)$ so that $\underline{a}(t_0) = \underline{a}(t_1)$ aligns with $\underline{a}(t)$, and finally a rotation of α radians about $\underline{a}(t)$ to obtain the required $\underline{n}(t)$ and $\underline{s}(t)$. This is equivalent to rotate the $[\underline{n}(t_0)\ \ \underline{s}(t_0)\ \ \underline{a}(t_0)]$ coordinate, which aligns with $[\underline{x}_0\ \ \underline{y}_0\ \ \underline{z}_0]$ originally, α radians about $\underline{z}_0$, then β radians about $\underline{y}_0$, and finally γ radians about $\underline{z}_0$ again. The relationship is thus

$$\underline{n}(t) = \begin{bmatrix} \cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma \\ \cos\alpha\cos\beta\sin\gamma + \sin\alpha\cos\gamma \\ -\cos\alpha\sin\beta \end{bmatrix} \tag{2}$$

$$\underline{s}(t) = \begin{bmatrix} -\sin\alpha\cos\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\alpha\cos\beta\sin\gamma + \cos\alpha\cos\gamma \\ \sin\alpha\sin\beta \end{bmatrix} \tag{3}$$

$$\underline{a}(t) = \begin{bmatrix} \sin\beta\cos\gamma \\ \sin\beta\sin\gamma \\ \cos\beta \end{bmatrix} \tag{4}$$

Consequently the state of hand at time t in Cartesian coordinates with reference to the base coordinates may also be represented by a 6-dimensional vector $\left[\underline{p}(t)' \quad \underline{\theta}(t)'\right]'$ where $\left[\underline{\theta}(t)\right]' = [\alpha \quad \beta \quad \gamma]$ and $(\ )'$ = transpose of ( ).

The hand, however, is driven by the actuators at the joints. Intuitively, if all the joint displacements are given, the position and orientation of the hand are determined. Let n be the number of joints. For $i = 1, 2, \ldots, n$, let q_i be the displacement of the i-th joint with respect to its own reference point. Then, for any given robot with known geometrical dimensions, there is a relation

$$\left[\underline{p}(t)' \quad \underline{\theta}(t)'\right]' = \underline{f}\left[q_1, q_2, \cdots, q_n\right] \tag{5}$$

where $\underline{f}(\cdot)$ is a 6 by 1 vector valued function. This relation is known, but almost always nonlinear which complicates the problem [4]. Since in reality, one specifies $\left[\underline{p}(t)' \quad \underline{\theta}(t)'\right]$ in Cartesian coordinates and desires to determine the corresponding $\left[q_1, \cdots, q_n\right]$ so that one may command the joint actuators to comply with the specification in Cartesian coordinates. The solution requires the inverse vector function $\underline{f}^{-1}(\cdot)$ of n-dimension. This solution, if it can be found, may not be unique. for the commercially available robots in operation, n is usually either 5 or 6. The geometrical configuration of these robots with proper definitions and ranges of q_i enables one to obtain a unique solution of equation (5) [4].

Knowing the transformation of position in Cartesian and joint coordinates, it is ready to examine the following simple task. As shown in Figure 6, the robot is required to go to the bolt magazine to fetch a bolt, and place it on top of the bolt hole on the beam. Intuitively one expects that the robot hand travels along the path of straight-line segments, as indicated in Figure 6, so that it will reach the bolt first and then arrive at the beam. The question is how would one control the joint actuators to accomplish the goal? Before one arrives with an answer, one may examine the following two possible specifications:

(a) Is the work space free from obstacles?

(b) Must the hand follow the specified path?

The answers to each of these two questions could be either yes or no. They combine to form four different classes of problems as follows:

		Is the work space free from obstacles?	
		Yes	No
Must the hand follow the specified path?	No	Class 1. Positional Control Problem	Class 4. Positional Control Plus On-line Collision Avoidance Travelling
	Yes	Class 2. Path Tracking Problem	Class 3. Off-line Collision-free Path Planning Plus On-line Path Tracking

Only the problems of Classes 1 and 2 will be analyzed briefly in the following sections.

III. POSITIONAL CONTROL

If there is no path constraints and if the work space is free, then the controllers only have to make sure that the hand passes through all the specified corner points of the path. It involves the coordinates transformation, which computes the corresponding joint coordinates $\left[q_1, \cdots, q_n\right]$ of the specified corner points in Cartesian coordinates by means of $\underline{f}^{-1}(\cdot)$, and then positionally control the robot in joint coordinates from point to point. In practice, a positional servo is used for each joint. If the robot is allowed to move only one joint at a time alternatively by locking all the other joints, then each of the joint controllers is very simple. When the number of joints in simultaneous motion is more than one, force interactions among the joints create couplings which complicate the control system. These topics will be discussed as follows.

A. Single-Joint Controller

In the following discussion, the robot is considered to have a rigid body structure. Refer to Figure 7(a), the schematic representation of an actuator-gear-load assembly for a single joint, in which

J_a = actuator inertia of one joint (oz-in-sec^2 / rad),

J_m = manipulator (robot) inertia of the joint,

J_L = inertia of the load,

B_m = damping coefficient at actuator side (oz-in-sec / rad),

B_L = damping coefficient at load side,

f_m = average friction torque (oz-in),

τ_g = gravitational torque,

τ_m = generated torque at actuator shaft,

τ_L = load torque,

θ_m = angular displacement at actuator shaft (radians),

θ_s = angular displacement at load side.

Let

N_m, N_s = number of teeth of the gear at the actuator, shaft and load shaft, respectively,

r_m, r_s = pitch radii of the gears at the actuator shaft and load shaft, respectively.

Then

$$n = r_m/r_s = N_m/N_s \leq 1. \tag{6}$$

is the gear ratio. As indicated in Figure 7(b), the force F is transmitted from the actuator to the load at the contacting point of the mating gears. Thus

$$\begin{aligned} \tau_L' &= \text{equivalent load torque at actuator shaft} \\ &= Fr_m \end{aligned}$$

and

$$\tau_L = Fr_s \tag{7}$$

which leads to

$$\tau_L'/\tau_L = r_m/r_s = n \tag{8}$$

or

$$\tau_L' = n\ \tau_L \tag{9}$$

From Figure 7(c),

$$\theta_m = 2\pi/N_m \tag{10}$$

$$\theta_s = 2\pi/N_s \tag{11}$$

so that

$$\theta_s = \theta_m\ N_m/N_s = n\ \theta_s \tag{12}$$

By taking time derivatives on both sides of (12), one obtains the relations between angular velocities and accelerations as

$$\dot{\theta}_s = n\,\dot{\theta}_m \tag{13}$$

$$\ddot{\theta}_s = n\,\ddot{\theta}_m \tag{14}$$

From Figure 7(a), it is seen that the load torque τ_L is required to overcome the load inertia effect $J_L\,\ddot{\theta}_s$ and damping effect $B_L\,\dot{\theta}_s$. Using D'Alembert's principle "Σ torque = $\Sigma(J\ddot{\theta})$", one obtains

$$\tau_L - B_L\dot{\theta}_s = J_L\ddot{\theta}_s \tag{15}$$

Apply the same principle at the actuator shaft to yield

$$\tau_m - \tau_L' - B_m\dot{\theta}_m = (J_a + J_m)\ddot{\theta}_m \tag{16}$$

If we wish to express the torque relation at the actuator shaft, first substitute (13) and (14) into (15) and then combine the result with (9) to obtain

$$\tau_L' = n^2(J_L\ddot{\theta}_m + B_L\dot{\theta}_m) \tag{17}$$

Combining (16) and (17) yields

$$\tau_m = (J_a + J_m + n^2J_L)\ddot{\theta}_m + (B_m + n^2B_L)\dot{\theta}_m \tag{18}$$

where $J_{eff} = (J_a + J_m + n^2J_L)$ is the effective inertia and $B_{eff} = (B_m + n^2B_L)$ is the effective damping coefficient at the actuator shaft.

To express the torque relation at the load shaft, eliminate $\ddot{\theta}_m$ and $\dot{\theta}_m$ among (13), (14) and (18) to yield

$$\tau_m/n = \left[(J_a + J_m)/n^2 + J_L\right]\ddot{\theta}_s + (B_m/n^2 + B_L)\dot{\theta}_s \tag{19}$$

where $\left[(J_a + J_m)/n^2 + J_L\right]$ is the effective inertia and $(B_m/n^2 + B_L)$ is the effective damping coefficient at the load shaft. τ_m/n is the equivalent generated torque at the load shaft.

The actuators used in the industrial robots are either hydraulic, or pneumatic, or electrical. As an example, the Unimation PUMA and the Stanford manipulators have electrical system using permanent magnet dc motors. They are armature controlled and their schematic diagram is shown in Figure 8. In this figure, $v_b(t)$ is the back emf in volts in the armature winding which can be represented by

$$v_b(t) = K_b \dot{\theta}_m(t) \tag{20}$$

where K_b is the back emf constant in volts-second/radian. Let L and R be the inductance in Henry and resistance in ohms of the motor armature winding, respectively. Then by applying the Kirchhoff's voltage law to the armature circuit, one obtains

$$v(t) - v_b(t) = L\, di(t)/dt + Ri(t) \tag{21}$$

which has an equivalent equation in frequency domain through a Laplace transformation

$$V(s) - K_b s\, \theta_m(s) = (Ls + R)I(s) \tag{22}$$

where s is the complex frequency in radians per second. The dc motor is operated in its linear range so that the generated torque is proportional to the armature current. The relation in the frequency domain is

$$T_m(s) = K_I I(s) \tag{23}$$

where K_I is the torque constant in oz-in/ampere. The motor shaft is mechanically connected to an actuator-gear-load assembly, as indicated in Figure

8, with an effective inertia J_{eff} and effective damping coefficient B_{eff} at the actuator shaft. The relation among the mechanical components are described by (18) which has a Laplace transform equivalence

$$T_m(s) = \left[(J_a + J_m + n^2 J_L)s^2 + (B_m + n^2 B_L)s\right] \theta_m(s)$$

$$= (J_{eff}\, s^2 + B_{eff}\, s)\, \theta_m(s) \qquad (24)$$

Eliminating $T_m(s)$ and $I(s)$ among (22), (23) and (24) yields

$$\frac{\theta_m(s)}{V(s)} = \frac{K_I}{s\left[LJ_{eff}s^2 + (RJ_{eff} + LB_{eff})s + (RB_{eff} + K_I K_b)\right]} \qquad (25)$$

which is the transfer function, or the feedforward gain, from the applied voltage to the dc motor (input) to the angular displacement of the motor shaft (output).

To construct a positional controller for the angular displacement of the load shaft, it is necessary to convert the displacement into electrical voltage to actuate the dc motor. For a feedback (or closed-loop) controller, the actuating signal is the error at time t between the desired and the actual displacements:

$$e(t) = \theta_d(t) - \theta_s(t) \qquad (26)$$

By means of a potentiometer or an optical encoder/counter assembly, the displacement error is converted into voltage as

$$v(t) = K_\theta\left[\theta_d(t) - \theta_s(t)\right] \qquad (27)$$

which has a transformed equivalence

$$V(s) = K_\theta E(s) = K_\theta \left[\theta_d(s) - \theta_s(s) \right] \tag{28}$$

where K_θ is the conversion constant in volts/radian. Combine all the physical apparatus together, one may construct a positional controller whose block diagram is shown in Figure 9(a). The feedforward gain, or the open-loop transfer function is

$$\frac{\theta_s(s)}{E(s)} = \frac{n K_\theta K_I}{s\left[LJ_{eff}s^2 + (RJ_{eff} + LB_{eff})s + (RB_{eff} + K_I K_b)\right]} \tag{29}$$

which is obtained either from the block diagram, Figure 9(a), or by combining equations (25) and (28), and the relation $\theta_s(s) = n\,\theta_m(s)$.

Physically the inductance of the armature winding is in the order of tenth of milli-henries while its resistance is in units of ohms. Thus L is practically zero so that (29) may be reduced to

$$\frac{\theta_s(s)}{E(s)} \simeq \frac{n K_\theta K_I}{S[RJ_{eff}s + (RB_{eff} + K_I K_b)]} \tag{30}$$

The unity feedback positional controller, Figure 9(a), then has a closed-loop transfer function

$$\frac{\theta_s(s)}{\theta_d(s)} = \frac{\theta_s/E}{1+\theta_s/E} = \frac{nK_\theta K_I}{RJ_{eff}} \cdot \frac{1}{S^2 + (RB_{eff} + K_I K_b)s/(RJ_{eff}) + K_\theta K_I/(RJ_{eff})} \tag{31}$$

It yields a second order system which is theoretically always stable. To increase the response time, one customarily increases the system gain, such as increasing K_θ, and injects some damping into the system to reinforce the effect of back emf by adding a negative feedback of the motor shaft velocity. This can be done by either using a tachometer, or computing the difference in angular displacements of the shaft during a fixed time interval. The block diagram of the resulting controller is shown as in Figure 9(b) in which K_t is the tachometer constant in volt-sec/radian, and K_1 is the gain of amplifier in volts/volt. Since the feedback voltage at the motor armature circuit is now $K_b\dot{\theta}_m(t) + K_1K_t\dot{\theta}_m(t)$ instead of $K_b\dot{\theta}_m(t)$ alone, the circuit equation (21) is modified as

$$v(t) - (K_b + K_1K_t)\dot{\theta}_m(t) = L\,di/dt + Ri(t) \tag{32}$$

which has a Laplace transform

$$V(s) - (K_b + K_1K_t)s\,\theta_m(s) = (Ls + R)I(s) \tag{33}$$

Thus the revised open-loop and closed-loop transfer functions can be obtained simply by replacing K_b by $(K_b + K_1K_t)$ in (29) and (31), respectively. Consequently,

$$\frac{\theta_s(s)}{E(s)} = \frac{nK_\theta}{s}\,\frac{s\theta_m(s)}{K_\theta E(s)}$$

$$= \frac{nK_\theta K_I}{RJ_{eff}s^2 + [RB_{eff} + K_I(K_b + K_1K_t)]s} \tag{34}$$

$$\frac{\theta_s(s)}{\theta_d(s)} = \frac{\theta_s(s)/E(s)}{1 + \theta_s(s)/E(s)}$$

$$= \frac{nK_\theta K_I}{RJ_{eff}S^2 + [RB_{eff} + K_I(K_b + K_1K_t)]S + nK_\theta K_I} \tag{35}$$

For a specific robot, the numerical values of the parameters n, K_I, K_t, K_b, R, J_{eff} and B_{eff} are either specified (by the component manufacturer), or determined by experiments. As an example, Joints 1 and 2 assemblies of the Stanford manipulator contain respectively a U9M4T and a U12M4T dc motor with an integral tachometer 030/105 by Photocircuits Corp. The parametric data for the motor-tachometer unit are listed as follows [5]:

Model	U9M4T	U12M4T
K_I (oz-in/amp.)	6.1	14.4
J_a (oz-in-sec^2/rad)	0.008	0.033
B_m (oz-in-sec /rad)	0.01146	0.04297
K_b (volts-sec/rad)	0.04297	0.10123
L (μ-henries)	100.	100.
R (ohms)	1.025	0.91
K_t (volts-sec/rad)	0.02149	0.05062
f_m (oz-in)	6.0	6.0
n	0.01	0.01

The second to the last line is the average friction torque f_m exists in each assembly which must be overcome. The effective inertia of each joint of the Stanford-JPL (Jet Propulsion Laboratory) manipulator is listed as [6]:

Joint No.	Min. Value/No Load	Max. Value/No Load	Max. Value/Full Load
1	1.417 $kg\text{-}m^2$	6.176 $kg\text{-}m^2$	9.570 $kg\text{-}m^2$
2	3.590	6.950	10.300
3	7.257	7.257	9.057
4	0.108	0.123	0.234
5	0.114	0.114	0.225
6	0.040	0.040	0.040

The given values of the effective inertia is in $kg\text{-}meter^2$. From Newton's second law of motion, $F = ma$, where mass m is in kg and acceleration a is in $meters/sec^2$; the force F is in $kg\text{-}meter/sec^2$ which is defined as the unit of force "Newton." The unit for torque τ is Newton-meter, or $kg\text{-}meter^2/sec^2$. But $\tau = J\ddot{\theta}$ where the angular acceleration $\ddot{\theta}$ is in $radians/sec^2$ so that J is in $Newton\text{-}meter\text{-}sec^2/radian$ (or $kg\text{-}meter^2$). Now one Newton-meter (or $kg\text{-}m^2/sec^2$) of work is equivalent to 0.73756 ft-lb., or $0.73756 \times 12 \times 16 = 141.61$ oz-in. Consequently one $kg\text{-}meter^2$ (or $Newton\text{-}meter\text{-}sec^2$ / radian) of inertia is the same as 141.61 $oz\text{-}in\text{-}sec^2/radian$. Note that the conversion constant K_θ and amplifier gain K_1, however, must be determined from the parameters corresponding to the structural resonant frequency and the damping ratio of the robot, which will be discussed in the following section.

As listed above, an average friction torque f_m of the motor-tachometer assembly must be overcome by the motor. Of course, the motor must also compensate the load torque τ_L and gravitational torque τ_g. These quantities represent the reaction from the physical burden to the robot. Schematically they are inserted in the block diagram of the positional controller, Figure 9(b), at the point where the torque is generated from the motor. Figure 9(c) shows the revised block diagram of the positional controller, in which

$F_m(s)$, $T_L(s)$ and $T_g(c)$ are the Laplace transformed variables of f_m, τ_L and τ_g, respectively.

B. Determination of K_θ and K_1

From (35), the closed-loop transfer function can be written as

$$\frac{\theta_s(s)}{\theta_d(s)} = \frac{nK_\theta K_I}{RJ_{eff}} \cdot \frac{1}{S^2 + \left[RB_{eff} + K_I(K_b + K_1K_t)\right]S/(RJ_{eff}) + nK_\theta K_I/(RJ_{eff})} \quad (36)$$

Thus the characteristic equation for the closed-loop controller is

$$S^2 + \left[RB_{eff} + K_I(K_b + K_1K_t)\right]S/(RJ_{eff}) + nK_\theta K_I/(RJ_{eff}) = 0 \quad (37)$$

which is conventionally expressed as

$$S^2 + 2\zeta\omega_n S + \omega_n^2 = 0 \quad (38)$$

where ζ is the damping ratio and ω_n the undamped natural frequency. From (37) and (38), one obtains

$$\omega_n = \sqrt{nK_\theta K_I/(RJ_{eff})} > 0 \quad (39)$$

and

$$2\zeta\omega_n = \left[RB_{eff} + K_I(K_b + K_1K_t)\right]/(RJ_{eff}) \quad (40)$$

so that

$$\zeta = \left[RB_{eff} + K_I(K_b + K_1K_t)\right]/\left[2\sqrt{nK_\theta K_I RJ_{eff}}\right] \quad (41)$$

Refer to Figure 10 in which k_{eff} is the effective stiffness (in oz-in/rad.) of the joint of the robot. The restoring torque due to stiffness is $-k_{eff}\theta$. Thus, by D'Alembert's principle

$$-k_{eff}\theta = J_{eff}\ddot{\theta} \tag{42}$$

so that the structural resonant frequency in radians/sec. is:

$$\omega_r = \sqrt{k_{eff}/J_{eff}} \tag{43}$$

Although k_{eff} for the joint is fixed, J_{eff} varies as the load varies so that ω_r changes accordingly. Let ω be the measured structural resonant frequency of the same joint corresponding to the effective inertia J, then

$$\omega = \sqrt{k_{eff}/J} \tag{44}$$

Thus, by (43) and (44),

$$\omega_r = \omega\sqrt{J/J_{eff}} \tag{45}$$

The measured ω and its corresponding J for the Stanford manipulator are listed as follows [1]:

Joint No.	J	f	$\omega(=2\pi f)$
1	5 kg-m2	4 Hz	25.1327 rad/sec
2	5	6	37.6991
3	7	20	125.6636
4	0.1	15	94.2477
5	0.1	15	94.2477
6	0.04	20	125.6636

For a conservative design with a safety factor of 200%, one set the undamped natural frequency ω_n no more than one-half of the structural resonant frequency ω_r [7]. Thus by (39) and (45), one obtains

$$\sqrt{nK_\theta K_I/(RJ_{eff})} \leq \omega \sqrt{J/J_{eff}} \;/2 \qquad (46)$$

which reduces to

$$K_\theta \leq (J\omega^2)R/(4nK_I) \qquad (47)$$

Relation (47) establishes the upper bound of K_θ. It remains to determine the bound on K_1. For practical reasons, one avoids the underdamped positional controller for the robot. Thus $\zeta \geq 1$, and from (41) one obtains:

$$RB_{eff} + K_I(K_b + K_1K_t) \geq 2\sqrt{nK_\theta K_I RJ_{eff}} > 0 \qquad (48)$$

Again for conservative design, K_θ at the right side of (48) is replaced by its upper bound which is given by (47) as $(J\omega^2)R/(4nK_I)$. Thus (48) reduces to

$$K_1 \geq R(\omega\sqrt{JJ_{eff}} - B_{eff})/(K_IK_t) - K_b/K_t \qquad (49)$$

Since J_{eff} varies as the load changes, the lower bound on K_1 changes accordingly. To simplify the design of the controller, the amplifier gain should be fixed. Thus the maximum value of J_{eff} should be used in (49) to avoid any possibility of resulting in an underdamping system.

C. Compensation for Steady-State Errors

In the preceding section, the block diagram of the positional controller for a single-joint robot was presented in Figure 9(c). Because of an addition of the physical burden f_m, τ_L and τ_g to the motor, the closed-loop transfer function of the controller is not the same as given by (35), and it must be modified to include the additions. From Figure 9(c), it is seen that

$$(J_{eff}S^2 + B_{eff}S)\theta_m = T_m(s) - F_m(s) - T_g(s) - nT_L(s) \tag{50}$$

But

$$T_m(s) = K_I\left[V(s) - s(K_b + K_1K_t)\theta_s(s)/n\right]/R \tag{51}$$

and

$$V(s) = K_\theta\left[\theta_d(s) - \theta_s(s)\right] \tag{52}$$

Thus, after some algebraic manipulation,

$$\theta_s(s) = \{nK_\theta K_I\theta_d(s) - nR\left[F_m(s) + T_g(s) + nT_L(s)\right]\}/\Omega(s) \tag{53}$$

where

$$\Omega(s) = RJ_{eff}s^2 + \left[RB_{eff} + K_I(K_b + K_1K_t)\right]s + nK_\theta K_I \tag{54}$$

Whenever $F_m(s)$, $T_g(s)$, and $T_L(s)$ vanish, equation (53) reduces to (35). Since the position error e(t) is defined as

$$e(t) = \theta_d(t) - \theta_s(t) \tag{55}$$

then (53) can be written as

$$\begin{aligned} E(s) &= \theta_d(s) - \theta_s(s) \\ &= (\{RJ_{eff}s^2 + \left[RB_{eff} + K_I(K_b + K_1K_t)\right]s\}\theta_d(s) + \\ &\quad nR\left[F_m(s) + T_g(s) + nT_L(s)\right])/\Omega(s) \end{aligned} \tag{56}$$

where E(s) is the Laplace transform of e(t). For a constant load, $\tau_L = C_L$.

Since $f_m = C_f$, and $\tau_g = C_g$ are also constant, then $T_L(s) = C_L/S$, $F_m(s) = C_f/s$ and $T_g(s) = C_g/s$. Consequently equation (56) becomes

$$E(s) = (\{RJ_{eff}s^2 + \left[RB_{eff} + K_I(K_b + K_1K_t)\right]s\}X(s) + nR\left[C_f + C_g + nC_L\right]/s)/\Omega(s) \tag{57}$$

where X(s) replaces $\theta_d(s)$ to represent a generalized input command.

The steady-state error e_{ss} may be determined by the use of the Final Value Theorem which states that

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) \tag{58}$$

provided the limits exist.

(a) Steady-State Position Error and Compensation

If the input is a constant displacement C_θ, then

$$X(s) = \theta_d(s) = C_\theta/s \tag{59}$$

By (58), one obtains a steady-state position error

$$e_{ssp} = R(C_f + C_g + nC_L)/(K_\theta K_I) \tag{60}$$

Since K_θ has an upper bound given by (47), the error may not be reduced to an arbitrary small value by merely adjusting the parameter K_θ. However, if one knows the value of τ_L, f_m and τ_g in advance, it is possible to feed forward these quantities into the controller to anticipate the burden. Based on this idea, an anticipated gravitational torque signal τ_a and a desired compensating torque signal τ_d are fed to the controller as an additional input, as shown in Figure 11(a) where $T_a(s)$ and $T_d(s)$ are Laplace transforms of τ_a and τ_d, respectively. With this arrangement, the error given by (56) is modified to become

$$E(s) = (\{RJ_{eff}s^2 + \left[RB_{eff}s^2 + \left[RB_{eff} + K_I(K_b + K_1K_t)\right]s\}X(s)$$

$$+ nR\{F_m(s) + T_g(s) + nT_L(s) - K_IK_R\left[T_a(s) + T_d(s)\right]/R\})/\Omega(s) \quad (61)$$

where X(s) replaces $\theta_d(s)$ to represent a generalized input signal. For an input of a constant displacement $X(s) = \theta_d(s) = C_\theta/s$, the steady-state position error now becomes

$$e_{ssp} = \lim_{s \to 0} s\{R\left[F_m(s) + T_g(s) + nT_L(s)\right] - K_IK_R\left[T_a(s) + T_d(s)\right]\}/(K_\theta K_I) \quad (62)$$

If $T_a(s) = RT_g(s)/(K_IK_R)$ and $T_d(s) = R\left[F_m(s) + nT_L(s)\right]/(K_IK_R)$, then the steady-state error would be zero. In practice, one may set:

$$\tau_a = (R/K_IK_R)\hat{\tau}_g \quad (63)$$

and

$$\tau_d = (R/K_IK_R)(\hat{f}_m + n\hat{\tau}_L) \quad (64)$$

to reduce the error, where $\hat{\tau}_g$, $\hat{f}_m$ and $\hat{\tau}_L$ are the estimates of τ_g, f_m and τ_L, respectively. For a given task, the value τ_L which includes the compliant torque, is usually known. Thus $\hat{\tau}_L$ can be estimated directly. The measured value of f_m through experimentation is usually used for $\hat{f}_m$. The value of $\hat{\tau}_g$ is normally computed which will be discussed in the section on Multiple Joint Controller.

(b) Tracking Error and Compensation

Quite often the robot is required to follow a moving conveyer with a constant speed, and perform a certain task on the conveyer. In this case the input signal of the desired position $\theta_d(t)$ must be updated very frequently, say every 1/60 second, to synchronize the moving conveyer. In essence the input is a ramp signal $C_v t$ with a constant slope so that $X(s) = C_v/S^2$. Applying the Final Value Theorem to (61), one obtains the steady-state tracking error

$$e_{sst} = \left[RB_{eff} + K_I(K_b + K_1K_t)\right]C_v/(nK_\theta K_I) + e_{ssp} \tag{65}$$

Since the controller requires $\zeta \geq 1$ to avoid underdamping, (48) holds so that (65) becomes

$$e_{sst} \geq 2C_v \sqrt{RJ_{eff}/(nK_\theta K_I)} + e_{ssp} \tag{66}$$

If the controller is designed conservatively to require $\omega_n \leq \omega_r/2$, then (47) holds and (66) reduces to

$$e_{sst} \geq 4(C_v/\omega) \sqrt{J_{eff}/J} + e_{ssp} \tag{67}$$

which gives a lower bound for the steady-state tracking error.

To reduce the steady-state tracking error, additional feedforward signal v_d, corresponding to the desired constant slope of the ramp signal is fed to the controller at the same place where τ_a and τ_d are injected. This is indicated in Figure 11(b) where $V_d(s)$ is the Laplace transform of v_d. As a result, E(s) given by (61) is modified to become

$$E(s) = (\{RJ_{eff}s^2 + \left[RB_{eff} + K_I(K_b + K_1K_t)\right]s\}X(s) - nK_IK_RV_d(s)$$
$$+ nR\{F_m(s) + T_g(s) + nT_L(s) - K_IK_R\left[T_a(s) + T_d(s)\right]/R\})/\Omega(s) \quad (68)$$

so that e_{sst} in (65) now becomes

$$e_{sst} = \{\left[RB_{eff} + K_I(K_b + K_1K_t)\right]C_v - nK_IK_R \lim_{s\to D} sV_d(s)\}/(nK_\theta K_I) + e_{ssp} \quad (69)$$

Consequently the first term of e_{sst} vanishes if

$$V_d(s) = s(C_v/s^2)\left[RB_{eff} + K_I(K_b + K_1K_t)\right]/(nK_IK_R) \quad (70)$$

But $C_v/s^2 = X(s)$ represents the ramp input signal, hence

$$v_d = (dx/dt)\left[RB_{eff} + K_I(K_b + K_1K_t)\right]/(nK_IK_R) \quad (71)$$

which can be obtained directly from the input terminal of the controller [7] as indicated in Figure 11(c). Since $x(t)$ is a ramp input C_vt, or $X(s) = C_v/S^2$, then dx/dt is the constant slope C_v, or $sX(s) = C_v/s$. Of course one may obtain dx/dt by computing the quotient $\left[x(t_i) - x(t_{i-1})\right]/(t_i-t_{i-1})$, where $x(t_i)$ and $x(t_{i-1})$ are values of two consecutive input signals. These arrangements will automatically take care of the steady state error in the original positional control mode. When the controller leaves the tracking mode and enters the positional control mode, $x(t)$ is a step input C_θ, or $X(s) = \theta_d(s) = C_\theta/s$. Then $dx/dt = C_\theta\delta(t)$, or $sX(s) = C_\theta$ (an impulse which is absorbed by the energy storing elements of the system), so that the compensation for the tracking error vanishes.

(c) Acceleration Error and Compensation

To minimize the travelling time of the robot, one often desires to accelerate it from one desired velocity to the other in a shortest possible time. To do so, one usually uses the constant, maximum possible acceleration. In this mode, then x(t) is a parabolic input signal $C_a t^2/2$ so that $X(s) = C_a/s^3$ where C_a is a constant. By the Final Value Theorem, (61) yields the steady-state acceleration error

$$e_{ssa} = (\{RJ_{eff} + \left[RB_{eff} + K_I(K_b + K_1 k_t)\right]/s\}C_a - nK_I K_R \lim_{s \to o} sV_d(s))/(nK_\theta K_I) + e_{ssp} \tag{72}$$

If a feedforward signal v_d is provided as indicated before and the controller is connected as shown in Figure 11(c), then

$$\begin{aligned} V_d(s) &= sX(s)Q(s) \\ &= (C_a/s^2)Q(s) \end{aligned} \tag{73}$$

where

$$Q(s) = \left[RB_{eff} + K_I(K_b + K_1 K_t)\right]/(nK_I K_R) \tag{74}$$

Consequently

$$e_{ssa} = RJ_{eff}C_a/(nK_\theta K_I) + e_{ssp} \tag{75}$$

For a conservatively designed controller with $\omega_n \leq \omega_r/2$, (47) holds so that (75) becomes

$$e_{ssa} \geq 4\ J_{eff}C_a/(J\omega^2) + e_{ssp} \tag{76}$$

This establishes the lower bound for the steady-state acceleration error.

As before, to reduce the error, additional feedforward signal α_d is required to be fed to the controller at the same point where v_d is applied, as shown in Figure 11(d) with $A_d(s)$ be the Laplace transform of α_d. With this addition, (68) reduces to:

$$E(s) = \Big[RJ_{eff}s^2X(s) - nK_IK_RA_d(s) + nR\{F_m(s) + T_g(s) + nT_L(s) - K_IK_R\big[T_a(s) + T_d(s)\big]/R\}\Big] / \Omega(s) \tag{77}$$

and hence

$$e_{ssa} = \Big[R\ J_{eff}C_a - nK_IK_R \lim_{s \to o} sA_d(s)\Big] / (nK_\theta K_I) + e_{ssp} \tag{78}$$

Again, if one chooses

$$A_d(s) = s^2(C_a/s^3)RJ_{eff}/(nK_IK_R)\ , \tag{79}$$

the first term of e_{ssa} will be eliminated. Since $C_a/s^3 = X(s)$, the parabolic input signal, $A_d(s)$ may be obtained from the input terminal directly [7] as shown in Figure 11(e). Since $x(t)$ is a parabolic signal $C_at^2/2$, then $dx(t)/dt = C_at$ is a ramp signal with constant slope, or $sX(s) = C_a/s^2$; and $d^2x(t)/dt^2 = C_a$ is a constant, or $s^2X(s) = C_a/s$. Again, the value of dx/dt may be obtained by computing the quotient $\big[x(t_i) - x(t_{i-1})\big]/(t_i - t_{i-1})$ and that of d^2x/dt^2 by

$$\{\big[x(t_i) - x(t_i - t_{i-1})\big]/(t_i - t_{i-1}) - \big[x(ti-1) - x(t_{i-1})\big]/(t_{i-1} - t_{i-2})\}/(t_i - t_{i-1}) \tag{80}$$

These arrangements also automatically handle the errors of other control modes. When the controller is in positional control mode, $x(t)$ is a step signal C_θ and $X(s) = C_\theta/s$. Then $dx/dt = C_\theta\delta(t)$, and $sX(s) = C_\theta$,

$d^2x/dt^2 = C_\theta\delta_d(t)$, a doublet or double impulse;

$s^2X(s) = sC_\theta$ so that the compensations for both the tracking error and acceleration error vanish. When the controller is in tracking mode, x(t) is a ramp signal $C_v t$ and $X(s) = C_v/s^2$. Then dx/dt is a step input C_v, $d^2x/dt^2 = C_v\delta(t)$; or $sX(s) = C_v/s$, and $s^2X(s) = C_v$ which give a correct amount of compensation for tracking error, but no compensation for acceleration error is provided.

Also, in computing $A_d(s)$, one needs the value of J_{eff}. This value must be estimated or approximated if it is not available. To avoid an over compensation which may result in an overshooting of the output of the positional controller, one normally uses the minimum value of J_{eff} [7] since $A_d(s)$ is a positive feedforward signal.

D. Multiple Joint Controller

Intuitively, it is not efficient to restrict the robot moving one of its joints at a time alternatively by locking all the other joints. In doing so the execution time of a given task is unnecessarily long and hence the operation is not economical. But to allow more than one joint moving simultaneously, force and moment interactions among the moving joints result which cause the inadequacy of the use of the above presented positional controller for each joint. Thus an additional compensation is needed to overcome the interaction. To determine the compensation for interaction, it is necessary to analyze the dynamic behavior of the robot.

(a) Lagrangian Formulation of Dynamic Equation

The discussion on Lagrangian equation can be found in most of physics textbooks (e.g. [8]). It represents the dynamic behavior of a system of rigid bodies, and it has the form

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \quad , \quad i = 1,2,\cdots,n. \tag{81}$$

where q_i = generalized coordinates.

$L = L(q_1,\cdots,q_n,\dot{q}_1,\cdots,\dot{q}_n)$ = Lagrangian

τ_i = generalized forcing function .

The generalized coordinate q_i represents the position θ_s of joint i. The Lagrangian is also defined as

L = (Kinetic energy of the system) - (Potential energy of the system).

By applying the Lagrangian equation to a robot with n joints (or n links), one obtains [6,7,9]:

$$\tau_i = \sum_{j=1}^{n} D_{ij}\ddot{q}_j + J_{ai}\ddot{q}_i + \sum_{j=1}^{n} D_{ijj}(\dot{q}_j)^2 + \sum_{\substack{j=1 \\ j\neq k}}^{n}\sum_{k=1}^{n} D_{ijk}\dot{q}_j\dot{q}_k + D_i \tag{82}$$

where

$$D_{ij} = \sum_{p=\max(i,j)}^{n} Tr\left[\underline{U}_{pj}\ \underline{J}_p(\underline{U}_{pi})'\right] \tag{83}$$

$$D_{ijk} = \sum_{p=\max(i,j,k)}^{n} Tr\left[\underline{U}_{pjk}\ \underline{J}_p(\underline{U}_{pi})'\right] \tag{84}$$

$$D_i = -\sum_{p=i}^{n} m_p\ \underline{\hat{g}}'\ \underline{U}_{pi}\ \underline{\hat{r}}_p \tag{85}$$

Tr = trace operator,

$(\ )'$ = transpose of $(\ )$,

τ_i = input generalized force for joint i,

m_p = mass of link p,

$\underline{\bar{r}}_p$ = a vector describing the center of mass of link p with respect to p-th coordinate system,

$\underline{\hat{g}}'$ = [0, 0, 9.8 m/sec2, 0] is a gravitational acceleration vector at a sea level base,

$\underline{J}_p$ = inertia matrix for link p,

$$\underline{U}_{pj} = \frac{\partial T_o^p}{\partial q_j} = \begin{cases} (\underline{T}_o^{j-1})\ \underline{Q}_j\ (\underline{T}_{j-1}^p), & \text{for } p \geq j, \\ \underline{0}, & \text{otherwise,} \end{cases} \tag{86}$$

$$\underline{U}_{pjk} = \frac{\partial^2 T_o^p}{\partial q_j \partial q_k} = \begin{cases} (\underline{T}_o^{j-1})\ \underline{Q}_j\ (\underline{T}_{j-1}^{k-1})\ \underline{Q}_k (\underline{T}_{k-1}^p), & \text{for } p \geq k \geq j, \\ (\underline{T}_o^{k-1})\ Q_k\ (\underline{T}_{k-1}^{j-1})\ \underline{Q}_j\ (\underline{T}_{j-1}^p), & \text{for } p \geq j \geq k \\ 0, & \text{otherwise,} \end{cases} \tag{87}$$

$$\underline{Q}_j = \begin{cases} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & \text{if joint j is rotational,} \\ \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & \text{if joint j is translational,} \end{cases} \tag{88}$$

$\underline{T}_j^k$ = 4 x 4 matrix which transforms any vector expressed in k-th coordinate system to the same vector expressed in j-th coordinate system.

q_k = generalized coordinate (i.e., joint displacement).

(b) Coupling Between Joints and Compensation

For each joint i, the required torque or force is divided into five groups as shown in (82). The first group represents the contribution from inertias of all the joints. Unlike the single-joint case in which all other joints are locked and inertias of all joints are lumped together, now there are contributions from coupling inertias between joints. These torque terms $\sum_{\substack{j=1 \\ j\neq i}}^{n} D_{ij}\ddot{q}_j$ must be fed forward in the controller for joint i, as shown in Figure 12, to compensate the interaction between joints. The second term in (82) represents the inertia torque of the actuator of joint i which has already been included in J_{eff}-term as outlined during the discussion on the single-joint controller. The last term is resulted from the gravitational acceleration, which has also been compensated by the feedforward term τ_a. This is the anticipated gravitational torque signal which must be computed by (63), i.e., $\tau_a = (R/K_I K_R)/\hat{\tau}_g$ where $\hat{\tau}_g$ is the estimate of gravitational torque τ_g. Intuitively, one uses D_i for the best estimate for τ_g for joint i controller. Thus by (85), one sets

$$\hat{\tau}_g = D_i = -\sum_{p=i}^{n} m_p \underline{\hat{g}}' \underline{U}_{pi} \underline{\hat{r}}_p \tag{89}$$

for joint i.

The third and fourth groups in (82) represent the contributions from, respectively, the centrifugal and the Coriolis forces. Again these torque terms must be fed forward in the controller for joint i to compensate the physical interactions between joints as shown in Figure 12. This figure depicts the complete block diagram of the controller for joint i of an industrial robot, $i=1,2,\cdots,n$. To implement these n controllers, the values of the feedforward elements D_{ij}, D_{ijk} and D_i must be computed for the specific robot, which will be discussed in the following sections.

(c) Computation of Compensation for Coupling Inertia

The computation of terms D_{ij} is, unfortunately, very complicated and time consuming. To illustrate the difficulty, equation (82) is expanded for a six-joint robot, n=6, as follows:

$$\begin{aligned}\tau_i = {} & D_{i1}\ddot{q}_1 + D_{i2}\ddot{q}_2 + \cdots + D_{i6}\ddot{q}_6 + J_{ai}\ddot{q}_i \\ & + D_{i11}\dot{q}_1^2 + D_{i22}\dot{q}_2^2 + \cdots + D_{i66}\dot{q}_6^2 \\ & + D_{i12}\dot{q}_1\dot{q}_2 + D_{i13}\dot{q}_1\dot{q}_3 + \cdots + D_{i16}\dot{q}_1\dot{q}_6 \\ & + \cdots \\ & + D_{i45}\dot{q}_4\dot{q}_5 + \cdots + D_{i56}\dot{q}_5\dot{q}_6 + D_i, \quad i=1,2,\cdots,6\end{aligned} \tag{90}$$

For i=1, the term $D_{i1} = D_{11}$ is further expended as shown in Figure 13 in which $\theta_i = q_i, i=1,2,\cdots,6$. Obviously it is not a simple computational task especially when the position-dependent and orientation-dependent parameters change as the robot moves. Therefore it warrants the effort of search for methods of simplifying the computation. There are three known approaches of simplification [10,11,7], viz. geometric/numeric, composite, and differential transformation. The geometric/numeric evaluation deals with the nature of joints whether it is revolute or prismatic. Thus the $\underline{T}_j^k$ matrices in (86), (87) and (88) can be simplified in advance. Since many elements in the 4 by 4 matrices are zeros, the resulting expressions for D_i, D_{ij} and D_{ijk} are less complicated [10]. The composite technique [11] involves the comparison of all the terms in Newton-Euler formulation of the dynamic equation [12] in a computer. Some of the terms may be eliminated under various criteria. The remaining terms are then rearranged in a Lagrangian formulation. The upshot is a computer output of a simplified equation in symbolic form. The differential transformation [7] which converts the partial derivatives of the matrix transformation $\partial \underline{T}_o^p/\partial q_j$ into the matrix product of the transformation and a differential matrix which reduces to a much simpler form. The third approach will be discussed in the following. To facilitate the discussion, it is necessary to introduce the homogeneous transformation which is a 4 by 4 matrix that represents the rotation and translation of vectors in some coordinate systems.

Refer to Figure 14(a) which shows two aligned coordinate systems $(\underline{x},\underline{y},\underline{z})$ and $(\underline{x}',\underline{y}',\underline{z}')$. A point P is fastened upon $(\underline{x}',\underline{y}',\underline{z}')$, i.e., when $(\underline{x}',\underline{y}',\underline{z}')$ moves with respect to $(\underline{x},\underline{y},\underline{z})$, point P moves with it. Supposing $(\underline{x}',\underline{y}',\underline{z}')$ is rotated γ radians about z-axis as shown in Figure 14(b). Since P moves together with $(\underline{x}',\underline{y}',\underline{z}')$, its location in that coordinates remains unchanged. However, the location of P in $(\underline{x},\underline{y},\underline{z})$ coordinates changes. Refer to Figure 14(c),

$$\begin{aligned} a_1 &= \overline{OP}\cos(\theta+\gamma) = \overline{OP}[\cos\theta\cos\gamma - \sin\theta\sin\gamma] \\ b_1 &= \overline{OP}\sin(\theta+\gamma) = \overline{OP}[\sin\theta\cos\gamma + \cos\theta\sin\gamma] \end{aligned} \tag{91}$$

But

$$\begin{aligned} a &= \overline{OP}\cos\theta \\ b &= \overline{OP}\sin\theta \end{aligned} \tag{92}$$

Thus

$$a_1 = a \cos\gamma - b \sin\gamma$$
$$b_1 = a \sin\gamma + b \cos\gamma \tag{93}$$
$$c_1 = c$$

which can be written as

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \tag{94}$$

The reason for the addition of the fourth row and fourth column in the matrix of the above equation will be clear when the translation of the position is introduced. In the mean time the matrix in (94) is the homogeneous transformation which rotates the vector or point P, and hence the coordinates $(\underline{x}', \underline{y}', \underline{z}')$, γ radians about z-axis. For convenience, the matrix is denoted by $\underline{R}(\underline{z},\gamma)$ so that (94) may be written as

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ 1 \end{bmatrix} = \underline{R}(\underline{z},\gamma) \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \tag{95}$$

Likewise,

$$\underline{R}(\underline{x},\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \underline{R}(\underline{y},\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{96}$$

Now supposing $(\underline{x}', \underline{y}', \underline{z}')$ is rotated β radians about y-axis. Again, the location of point P in $(\underline{x}',\underline{y}',\underline{z}')$ does not change but in $(\underline{x},\underline{y},\underline{z})$ changes from (a_1,b_1,c_1) to (a_2,b_2,c_2) as

$$\begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ 1 \end{bmatrix} = \underline{R}(\underline{y},\beta) \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ 1 \end{bmatrix} = \underline{R}(\underline{y},\beta)\underline{R}(\underline{z},\gamma) \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix}, \tag{97}$$

or

$$\begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta\ \cos\gamma & -\cos\beta\ \sin\gamma & \sin\beta & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ -\sin\beta\ \cos\gamma & \sin\beta\ \sin\gamma & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \tag{98}$$

Thus, (97) and (98) show that coordinates $(\underline{x}',\underline{y}',\underline{z}')$ is first rotated γ radians about z-axis, and then rotated β radians about y-axis. If it is further rotated α radians about x-axis, then one will have

$$\begin{bmatrix} a_3 \\ b_3 \\ c_3 \\ 1 \end{bmatrix} = \underline{R}(\underline{x},\alpha) \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ 1 \end{bmatrix} = \underline{R}(\underline{x},\alpha)\underline{R}(\underline{y},\beta)\underline{R}(\underline{z},\gamma) \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \tag{99}$$

or

$$\begin{bmatrix} a_3 \\ b_3 \\ c_3 \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \tag{100}$$

where

$$\begin{aligned} R_{11} &= \cos\beta \cos\gamma \\ R_{12} &= -\cos\beta \sin\gamma \\ R_{13} &= \sin\beta \\ R_{21} &= \cos\alpha \sin\gamma + \sin\alpha \sin\beta \cos\gamma \\ R_{22} &= \cos\alpha \cos\gamma - \sin\alpha \sin\beta \sin\gamma \\ R_{23} &= -\sin\alpha \cos\beta \\ R_{31} &= \sin\alpha \sin\gamma - \cos\alpha \sin\beta \cos\gamma \\ R_{32} &= \sin\alpha \cos\gamma + \cos\alpha \sin\beta \sin\gamma \\ R_{33} &= \cos\alpha \cos\beta \end{aligned} \tag{101}$$

Now supposing $(\underline{x}', \underline{y}', \underline{z}')$ is translated t_x, t_y and t_z units, respectively, along x-axis, y-axis, and z-axis. Then

$$\begin{bmatrix} a_4 \\ b_4 \\ c_4 \\ 1 \end{bmatrix} = \begin{bmatrix} a_3 \\ b_3 \\ c_3 \\ 1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \\ 0 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \tag{102}$$

Let

$$\underline{L}(t_x,t_y,t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{103}$$

denotes the above mentioned linear translation, then the 4 by 4 matrix in (102) may be written as

$$\underline{L}(t_x,t_y,t_z)\underline{R}(\underline{x},\alpha)\underline{R}(\underline{y},\beta)\underline{R}(\underline{z},\gamma) = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{104}$$

which represents a rotation of γ radians about z-axis, then a rotation of β radians about y-axis followed by a rotation of α radians about x-axis, and finally a translation of t_x, t_y, t_z, respectively along the x, y, and z-axes. Thus the 4 by 4 matrix which is called the homogeneous transformation, includes rotation as well as translation of the coordinates $(\underline{x}',\underline{y}',\underline{z}')$. Since the matrix multiplications do not commute, the order of multiplying the matrices $\underline{L}$ and $\underline{R}$'s in (104) cannot be interchanged. Should, for example, $\underline{R}(\underline{x},\alpha)$ and $\underline{R}(\underline{y},\beta)$, or $\underline{L}(t_x,t_y,t_z)$ and $\underline{R}(\underline{z},\gamma)$ interchange their places in (104), the resulting matrix, and hence the physical order of rotations and translation would be different.

Refer to Figure 15 and suppose that the p-th joint of the robot is originally at the point P. Then

$$\underline{T}_o^p = \begin{bmatrix} \underline{x}_p & \underline{y}_p & \underline{z}_p & \underline{\ell}_p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{105}$$

represents the coordinate frame with respect to the base coordinates. Now rotate the joint in the following order: γ radians about z_o-axis, β radians about y_o-axis, α radians about x_o-axis; and finally translate $\underline{t}$ with respect to $(\underline{x}_o, \underline{y}_o, z_o)$ coordinates. Supposing the resulting orientation and position of the joint is, respectively, $(\underline{x}_p^1, \underline{y}_p^1, \underline{z}_p^1)$ and $\underline{\ell}_p^1$ with reference to $(\underline{x}_o, \underline{y}_o, \underline{z}_o)$. Thus the current state of the hand with reference to base coordinates can be represented by:

$$\begin{bmatrix} \underline{x}_p^1 & \underline{y}_p^1 & \underline{z}_p^1 & \underline{\ell}_p^1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \underline{T}_o^p \, . \tag{106}$$

Now if $\underline{T}_o^p$ is perturbed by a small translation and rotation with respect to the base coordinates then $t_x \leftarrow \delta_o x$, $t_y \leftarrow \delta_o y$, $t_z \leftarrow \delta_o z$, $\alpha \leftarrow \delta_o \alpha$, $\beta \leftarrow \delta_o \beta$, $\gamma \leftarrow \delta_o \gamma$. But $\cos(\delta_o \alpha) \approx 1$, $\sin(\delta_o \alpha) \approx \delta_o \alpha$, $\cdots$, $\delta_o \alpha \delta_o \beta \approx 0$, $\cdots$, etc. Hence by (101),

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\delta_o \gamma & \delta_o \beta & \delta_o x \\ \delta_o \gamma & 1 & -\delta_o \alpha & \delta_o y \\ -\delta_o \beta & \delta_o \alpha & 1 & \delta_o z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{107}$$

On the other hand, one may express

$$\begin{bmatrix} \underline{x}_p^1 & \underline{y}_p^1 & \underline{z}_p^1 & \underline{\ell}_p^1 \\ o & o & o & 1 \end{bmatrix} = \underline{T}_o^p + \delta_o \underline{T}_o^p \tag{108}$$

Combining (106), (107) and (108) yields the perturbation

$$\delta_o \underline{T}_o^p = \begin{bmatrix} 0 & -\delta_o\gamma & \delta_o\beta & \delta_o x \\ \delta_o\gamma & 0 & -\delta_o\alpha & \delta_o y \\ -\delta_o\beta & \delta_o\alpha & 0 & \delta_o z \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{T}_o^p \tag{109}$$

The matrix in (109) is the variational operator with respect to the base coordinates. If the variation is referred to p-th joint's own coordinates, then it must be premultiplied by a 4 by 4 transformation matrix

$$\underline{T}_p^o = (\underline{T}_o^p)^{-1} \tag{110}$$

which transforms any vectors or coordinate frames with reference to base coordinates $(\underline{x}_o, \underline{y}_o, \underline{z}_o)$ to the p-th joint coordinates $(\underline{x}_p, \underline{y}_p, \underline{z}_p)$. Thus the perturbation on the p-th joint coordinate frame with reference to its own coordinates is

$$\delta_p \underline{T}_o^p = (\underline{T}_o^p)^{-1} \begin{bmatrix} 0 & -\delta_o\gamma_p & \delta_o\beta_p & \delta_o x_p \\ \delta_o\gamma_p & 0 & -\delta_o\alpha_p & \delta_o y_p \\ -\delta_o\beta_p & \delta_o\alpha_p & 0 & \delta_o y_p \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{T}_o^p \tag{111}$$

Since

$$(\underline{T}_o^p)^{-1} = \begin{bmatrix} x_{px} & x_{py} & x_{pz} & -\underline{x}_p'\underline{\ell}_p \\ y_{px} & y_{py} & y_{pz} & -\underline{y}_p'\underline{\ell}_p \\ z_{px} & z_{py} & z_{pz} & -\underline{z}_p'\underline{\ell}_p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{112}$$

Then

$$\delta_p\underline{T}_o^p = \begin{bmatrix} \underline{x}_p'(\underline{\Delta}_p X \underline{x}_p) & \underline{x}_p'(\underline{\Delta}_p X \underline{y}_p) & \underline{x}_p'(\underline{\Delta}_p X \underline{z}_p) & \underline{x}_p'[(\underline{\Delta}_p X \underline{\ell}_p) + \underline{d}_p] \\ \underline{y}_p'(\underline{\Delta}_p X \underline{x}_p) & \underline{y}_p'(\underline{\Delta}_p X \underline{y}_p) & \underline{y}_p'(\underline{\Delta}_p X \underline{z}_p) & \underline{y}_p'[(\underline{\Delta}_p X \underline{\ell}_p) + \underline{d}_p] \\ \underline{z}_p'(\underline{\Delta}_p X \underline{x}_p) & \underline{z}_p'(\underline{\Delta}_p X \underline{y}_p) & \underline{z}_p'(\underline{\Delta}_p X \underline{z}_p) & \underline{z}_p'[(\underline{\Delta}_p X \underline{\ell}_p) + \underline{d}_p] \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{113}$$

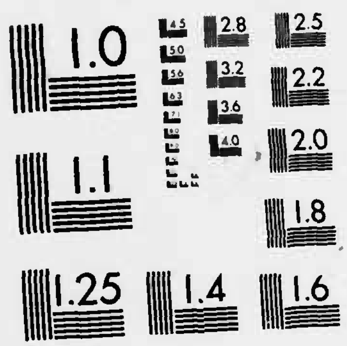
where ( )' = transpose of ( ), $\underline{\Delta}_p = \left[\delta_o\alpha_p \ \ \delta_o\beta_p \ \ \delta_o\gamma_p\right]'$, $\underline{d}_p = \left[\delta_o x_p \ \ \delta_o y_p \ \ \delta_o z_p\right]'$, and X = cross product. By the vector identities $\underline{a}'(\underline{b}X\underline{a}) = 0$, $\underline{a}'(\underline{b}X\underline{c}) = \underline{b}'(\underline{c}X\underline{a})$, $\underline{x}_p X \underline{y}_p = \underline{z}_p = -\underline{y}_p X \underline{x}_p$, $\cdots$, etc., (113) reduces to

$$\delta_p\underline{T}_o^p = \begin{bmatrix} 0 & -\underline{z}_p'\underline{\Delta}_p & \underline{y}_p'\underline{\Delta}_p & \underline{x}_p'(\underline{\Delta}_p X \underline{\ell}_p + \underline{d}_p) \\ \underline{z}_p'\underline{\Delta}_p & 0 & -\underline{x}_p'\underline{\Delta}_p & \underline{y}_p'(\underline{\Delta}_p X \underline{\ell}_p + \underline{d}_p) \\ -\underline{y}_p'\underline{\Delta}_p & \underline{x}_p'\underline{\Delta}_p & 0 & \underline{z}_p'(\underline{\Delta}_p X \underline{\ell}_p + \underline{d}_p) \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{114}$$

Since $\delta_p\underline{T}_o^p$ is the perturbation on the p-th joint coordinate frame with reference to its own coordinates, then based on the results in (109), it can be written as

AD-A134 852 TUTORIAL WORKSHOP ON ROBOTICS AND ROBOT CONTROL(U) ARMY TANK-AUTOMOTIVE COMMAND WARREN MI 26 OCT 82 3/4

UNCLASSIFIED F/G 14/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

$$\delta_p \underline{T}_o^p = \begin{bmatrix} 0 & -\delta_p\gamma & \delta_o\beta & \delta_p x \\ \delta_p\gamma & 0 & -\delta_p\alpha & \delta_p y \\ -\delta_p\beta & \delta_p\alpha & 0 & \delta_p z \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{T}_p^p \tag{115}$$

where $\underline{T}_p^p$ is an identity matrix, so that

$$\begin{aligned} \delta_p\alpha &= \underline{x}_p^{\prime}\underline{\Delta}_p \\ \delta_p\beta &= \underline{y}_p^{\prime}\underline{\Delta}_p \\ \delta_p\gamma &= \underline{z}_p^{\prime}\underline{\Delta}_p \\ \delta_p x &= \underline{x}_p^{\prime}(\underline{\Delta}_p X \underline{\ell}_p + \underline{d}_p) \\ \delta_p y &= \underline{y}_p^{\prime}(\underline{\Delta}_p X \underline{\ell}_p + \underline{d}_p \\ \delta_p z &= \underline{z}_p^{\prime}(\underline{\Delta}_p X \underline{\ell}_p + \underline{d}_p) \end{aligned} \tag{116}$$

Equations (115) and (116) serve as the basis for numerical computation because of their reduced form. Now, if the perturbation on the p-th joint coordinate frame is expressed with reference to j-th joint coordinates $(\underline{x}_j, \underline{y}_j, \underline{z}_j)$ then it has a form

$$\delta_j \underline{T}_o^p = \underline{T}_j^p(\delta_p \underline{T}_o^p) \tag{117}$$

As a limit,

$$\frac{\partial \underline{T}_o^p}{\partial q_j} dq_j = \underline{T}_j^p(\delta_p \underline{T}_o^p) \tag{118}$$

Through a lengthy algebraic manipulation, reference 7 shows that (with the condition that all the cross inertia terms are ignored since they are relatively insignificant by experiments [6,7]):

$$D_{ij} = \sum_{p=\max(i,j)}^{n} m_p \Big\{ \Big[(\delta_p \alpha_i) k_{pxx}^2 (\delta_p \alpha_j) + (\delta_p \beta_i) k_{pyy}^2 (\delta_p \beta_j)$$

$$+ (\delta_p \gamma_i) k_{pzz}^2 (\delta_p \gamma_j) + \Big[(\underline{T}_i^o \underline{d}_p)' (\underline{T}_j^o \underline{d}_p) \Big] \tag{119}$$

$$+ \Big[\hat{r}_p' ((\underline{T}_i^o \underline{d}_p) X (\underline{T}_j^o \underline{\Delta}_p) + (\underline{T}_j^o \underline{d}_p) X (\underline{T}_i^o \underline{\Delta}_p)) \Big] \Big\}$$

where $\delta_p \alpha_i$ is a small rotation of coordinates frame of joint p about x-axis with respect to i-th joint's coordinates, and k_{pxx} is the radius of gyration "xx" of joint (or link) p about the origin of p-th joint's coordinates, ··· etc.

(d) Computation of Compensation for Gravity, Centrifugal and Coriolis Terms.

From (85) and (86) one obtains

$$D_i = - \sum_{p=i}^{n} m_p \, \underline{g}' \, \frac{\partial \underline{T}_o^p}{\partial q_i} \, \underline{\hat{r}}_p \tag{120}$$

By some algebraic manipulation, reference 7 also shows that

$$D_i = \underline{r}' \sum_{p=i}^{n} m_p \, (\underline{\hat{r}}_p^{i-1}) \tag{121}$$

where $\underline{\hat{r}}_p^{i-1}$ = a vector describing the center of mass of link p with respect to (i-1)th coordinates, and

$$\underline{r}' = \begin{cases} [-\underline{g}' \underline{y}_{i-1} \quad \underline{g}' \underline{x}_{i-1} \quad 0 \quad 0] & \text{if joint p is revolute,} \\ [0 \quad 0 \quad 0 \quad -\underline{g}' \underline{z}_{i-1}] & \text{if joint p is prismatic.} \end{cases}$$

Since the term D_{ijk} contains a second partial derivative $\partial^2 \underline{T}_o^p/(\partial q_j \partial q_k)$, it is not able to simplify (84) for computation. Conventionally, one often ignores the centrifugal and Coriolis terms. The justification is that these two terms are velocity-dependent. When the robot starts to move from its initial location and approaches its goal location, the velocities are usually low and hence the contributions from these two terms are insignificant. Once it picks up the velocity, the robot is travelling in the space and normally the travelled path is not of importance. Should the path is important, such as avoiding collision with obstacles, then these two terms may not be ignored. They must be computed either by equation (84), or using Newton-Euler formulation approach [12] which is a computational scheme. This scheme has been proven to be computationally efficient [13]. Also, Bejczy [6,10] used geometric/numeric approach to show that for the last four joints of the Stanford-JPL manipulator, which has six joints (n=6), the following terms are identically zero:

D_{333} D_{334} D_{335} D_{336} D_{344} D_{345} D_{346} D_{356} D_{366}

D_{433} D_{434} D_{435} D_{436} D_{444} D_{446} D_{455} D_{466}

D_{533} D_{534} D_{535} D_{536} D_{555} D_{556} D_{566}

D_{633} D_{634} D_{635} D_{636} D_{644} D_{666}

Thus it is possible to reduce the computational task if the geometrical configuration of the robot is known and if an analysis is carried out.

IV. PATH TRACKING

Refer to the questions raised at the end of Section III. If the robot is required to travel along a prescribed path, the controller must keep up with path tracking. With positional controllers the path tracking can be accomplished by dividing the Cartesian path into a number of segments. Each end point of the segments is transformed into joint coordinates, and then the positional control is applied from point to point in joint coordinates. This approach was briefly mentioned previously in Section II-C(b) on tracking errors. A number of facts related to this approach should be mentioned. By transforming all the end points of segments of Cartesian path, one essentially constructs n corresponding paths in joint coordinates, one for each of the n joints. If these segments, $\left[d\underline{p}(t)' \quad d\underline{\theta}(t)'\right]$, are very short, the increments of joint displacement, dq_i, between adjacent points are very small so that sin $dq_i \approx dq_i$ and cos $dq_i \approx 1$. Thus the transformation $\underline{f}(\cdot)$ defined by (5) becomes a differential transformation which is usually linear. This transformation is the Jacobian matrix of the displacement, which contains trigonometric functions of the joint displacement with respect to the joint coordinates before the differential increment takes place [14]. Analytically, the solution dq_i in terms of $d\underline{p}$ and $d\underline{\theta}$ can be obtained simply by inventing the Jacobian matrix. Although it is sometimes possible, it is usually difficult since the Jacobian is quite complicated. Numerical solution is also possible but usually requires long computing time. Moreover, the Jacobian matrix becomes singular when the robot reaches a degenerate position at which the solution dq_i is not unique (i.e., more than one value of dq_i yields a same $d\underline{p}$ and $d\underline{\theta}$.) An alternative method proven to work successfully is to differentiate the solution of equation (5) directly [14]. This is possible since for a given robot with fixed dimensions, the transformation $\underline{f}^{-1}$ is known. Using this approach, one must set dq_i to zero if it is physically impossible due to constraints, or if it is undetermined. It usually results in a simpler expression [14].

The desired Cartesian corner points of the path may be fed into the control system either numerically, or through the teaching by doing procedure. The intermediate points of the small segments of the path are usually specified, or interpolated by the computer-controller, or sampled and recorded through teaching by doing.

By controlling the robot from point to point using the positional controller, it has a natural tendency to stop at each point. This undesirable phenomenon may be eliminated by specifying a nonzero velocity at each Cartesian point, which can be transformed into the corresponding nonzero joint velocities by a Jacobian matrix of the velocity. The joint velocities are often referred to as the "resolved rate" [15]. For implementation a velocity control loop must be added to each controller. Alternatively, one may take advantage of the associated digital computer to compute the required joint torques which would yield appropriate velocities at fixed accelerations. The computation requires the knowledge of the dynamical equation of the robot. Since the robot is a nonlinear mechanism with couplings among its joints, as shown previously in Section III-D(a) on Lagrangian formulation, the computation is time consuming.

A possible way to avoid the storage of numerous pre-computed points of joints paths or the computation of these points on-line is to determine the functional representation of n joint paths. However, the transformation $\underline{f}$ in (5) is pointwise. Since no transformation of a function is known, the curve fitting procedure may be adopted as follows [16,17,18]. The corner points of the Cartesian path are first transformed into joint coordinates.

A cubic function is assigned to each segment between every two adjacent joint coordinate points. The adjacent cubic functions are then splined together with the continuity conditions of desired velocity and, if required, desired acceleration. Curve fitting is done by using a large number of points on the path, after they are transformed into joint coordinates, and applying the least square error fit to obtain the coefficients of the spline function. Thus one needs to store only the coefficients. The points on the joint paths can be obtained by a simple computation when needed.

V. CONCLUSION

For industrial robots whose assignments are limited to usual tasks involving mainly synchronization, their controllers can be designed by conventional methods based on two factors: damping factor and structural resonant frequency. It is possible to reduce the steady-state position error and to eliminate the tracking and acceleration errors by feedforward compensations. For multiple joint controllers, further feedforward compensation is necessary to encounter the force interactions between joints. However, it introduces the computational difficulties so that methods of approximation or simplification are needed.

REFERENCES

1. Scheinman, V. D., Design of a Computer Manipulator, A. I. Memo 92, Artificial Intelligency Laboratory, Stanford University, June 1969.

2. Paul, R., J. Luh, et al., Advanced Industrial Robot Control Systems, Seventh Report, TR-EE 81-8, School of Electrical Engineering, Purdue University, March 1981, p. 23.

3. Roberts, L. G., Homogeneous Matrix Representation and Manipulation of N-dimensional Constructs, Lincoln Laboratory Document, No. MS 1045, MIT, 1965.

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

5. Luh, J. Y. S., W. D. Fisher and R. P. C. Paul, "Joint Torque Control by a Direct Feedback for Industrial Robots," Proceedings of 20th IEEE Conference on Decision and Control, December 16-18, 1981, San Diego, California, pp. 265-270. Also to appear in IEEE Transactions on Automatic Control, Vol. 28, No. 3, March 1983.

6. Bejczy, A. K., Robot Arm Dynamics and Control, Technical Memorandum 33-669, Jet Propulsion Laboratory, February 1974.

7. Paul, R. P., Robot Manipulators: Mathematics, Programming and Control, MIT Press, 1981.

8. Goldstein, H. Classical Mechanics, Addison-Wesley, 1959.

9. Lewis, R. A., Autonomous Manipulation on a Robot: Summary of Manipulator Software Functions, Tech Memo. 33-679, Jet Propulsion Laboratory, March 1974.

10. Bejczy, A. K. and R. P. Paul, "Simplified Robot Arm Dynamics for Control," Proceedings of 20th IEEE Conference on Decision and Control, December 16-18, 1981, San Diego, California, pp. 261-262.

11. Luh, J. Y. S. and C. S. Lin, "Automatic Generation of Dynamic Equations for Mechanical Manipulators," Proceedings of Joint Automatic Control Conference, June 17-19, 1981, Charlottesville, Virginia, pp. TA-2D.

12. Luh, J. Y.S., M. W. Walker and R. P. C. Paul, "On-line Computational Scheme for Mechanical Manipulators," ASME Transactions, Journal of Dynamic Systems, Measurement and Control, Vol. 102, No. 2, June 1980, pp. 69-76.

13. Hollerbach, J. M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Cooperative Study of Dynamics Formulation Complexity," IEEE Transactions on Systems, Man and Cybernetics, Vol. 10, No. 11, November 1980, pp. 730-736.

14. Paul, R. P., B. Shimano and G. E. Mayer, "Differential Kinematic Control Equations for Simple Manipulators," IEEE Transactions on Systems, Man and Cybernetics, Vol. 11, No. 6, June 1981, pp. 456-460.

15. Whitney, D. E., "Resolved Motion Rate Control of Manipulators and Human Prostheses," IEEE Transactions on Man-Machine Systems, Vol. 10, No. 2, June 1969, pp. 47-53.

16. Paul, R. C., Modeling, Trajectory, Calculation and Servoing of a Computer Controlled Arm, A. I. Memo 177, Artificial Intelligence Laboratory, Stanford University, September 1972.

17. Finkel, R. A., Constructing and Debugging Manipulator Programs, A. I. Memo 284, Artificial Intelligence Laboratory, Stanford University, August 1976.

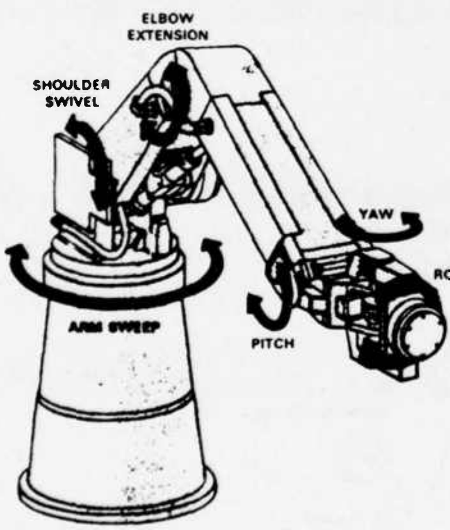
18. Luh, J. Y. S. and C. S. Lin, "Approximate Joint Paths for Control of Mechanical Manipulators," Proc. IEEE Conference on Pattern Recognition and Image Processing, Los Vegas, June 14-17, 1982, pp. 622-623.

WHAT DID WE NEGLECT TO TELL YOU ABOUT THE TOMORROW TOOL?

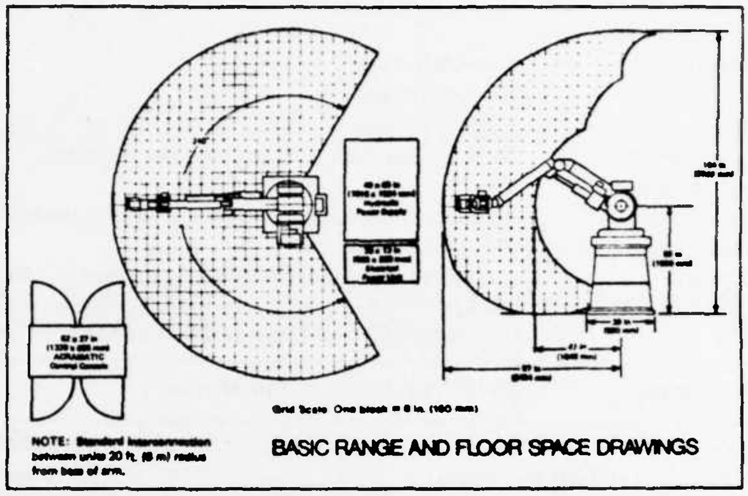
There are plenty of facts we left out. And there are plenty of details about the case histories* presented. Then there are the questions in your mind that we never even thought of.

We'd like to answer them for you. We'd like to show you how The Tomorrow Tool might make your plant more productive, more profitable. And even safer. Phone or write us with any question.

Industrial Robot Division,
Cincinnati Milacron,
4701 Marburg Avenue,
Cincinnati, Ohio 45209.
Phone 513/841-8386.



THE TOMORROW TOOL DOES WORK PEOPLE CAN'T DO.



BASIC SPECS FOR THE TOMORROW TOOL.

Floor space and net weight
T^3 Industrial robot . . . 9 sq. ft. (0.8 sqm) 5,000 lb (2267 kg)
Hydraulic power supply. 17 sq. ft. (1.5 sqm) 1,200 lb (544 kg)
Electrical power unit . . . 3.4 sq. ft. (0.3 sqm) 700 lb (317 kg)
ACRAMATIC computer
control . . . . 10 sq. ft. (0.9 sqm) 1,100 lb (498 kg)

Load Capacity
Load 10" (254 mm) from tool mounting plate . . 120 lbs (54 kg)
Load at tool mounting plate, max. velocity . . . . 175 lbs (79 kg)
Load at tool mounting plate, reduced velocity, depends on arm and wrist attitude* . . . . 300 lbs (136 kg)

Positioning accuracy, axis drive
Accuracy to any programmed point . ± 0.050-in. (± 1.27mm)
Drive for each of 6 axes . . . . Direct, electrohydraulic

Jointed-arm motions, range, velocity
Max. horizontal sweep . . . . 240°
Max. velocity of tool center point . . . . . . 50 ips (1270 mmps)
Pitch . . . . 190°
Roll . . . . 240°
Yaw . . . . 180°

Power requirements . . 230/460 volts, 3 phase, 60 Hz, 22 KVA

Environmental temperature* . . . . . 40 to 120° F (5 to 50° C)

Minicomputer memory capacity . . . . 400 points std. (additional storage optionally available)

*Consult factory for special applications

Figure 1. Cincinnati Milacron T3.

Unimation® Inc. PUMA™ Robot Specifications

600 Series

General	
Configuration	6 revolute axes
Drive	Electric DC servos
Controller	System computer
Teaching method	By teach pendant and/or computer terminal
Program language	VAL™
Program capacity	As required by application (4K RAM user memory std.)
External program storage	Floppy-disk (optional)
Gripper control	Computer-controlled, pneumatic, 0.5 cfm @ 100 psi.
Optional accessories	CRT or TTY, floppy-disk, pneumatic grippers (w/o fingers) I/O module (115 VAC compatible), continuous path
Power requirement	95-130 Vac, 50-60 Hz, 750 VA max.

Performance	
Repeatability	±0.004 in. (.1mm)
Load capacity	5.0 lbs. (including gripper) (2.27KG)
Tip velocity	3.3 ft./sec. max. with max. load
Static force at tip	13.2 lbs. max.

Physical characteristics	
Arm weight	120 lbs.
Controller size	16.5 in. x 19.0 in. x 20.0 in. (H x W x D)
Controller weight	75 lbs.
Controller cable length	15 ft. max.

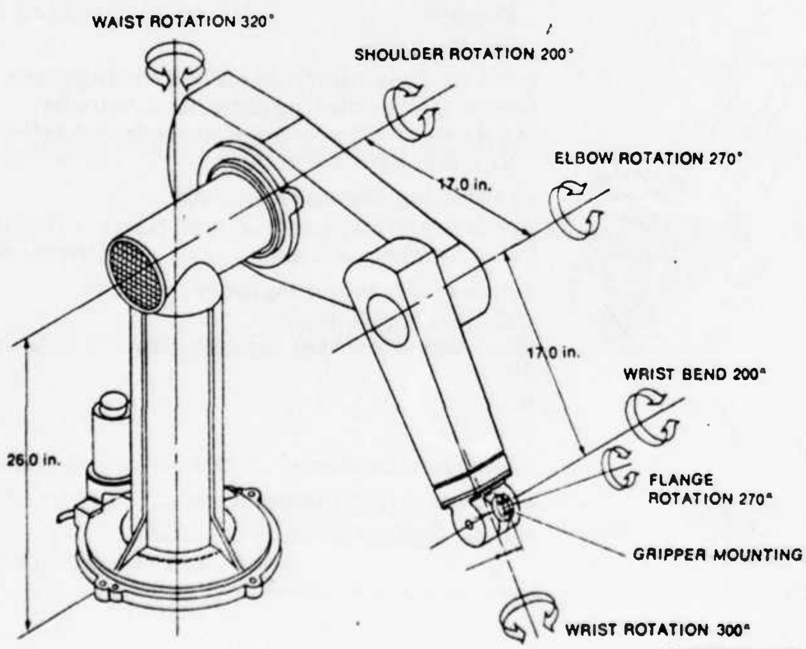


Figure 2. PUMA 600

Unimation Inc.
A CONDEC Company
Shelter Rock Lane, Danbury, Connecticut 06810
(203) 744-1800

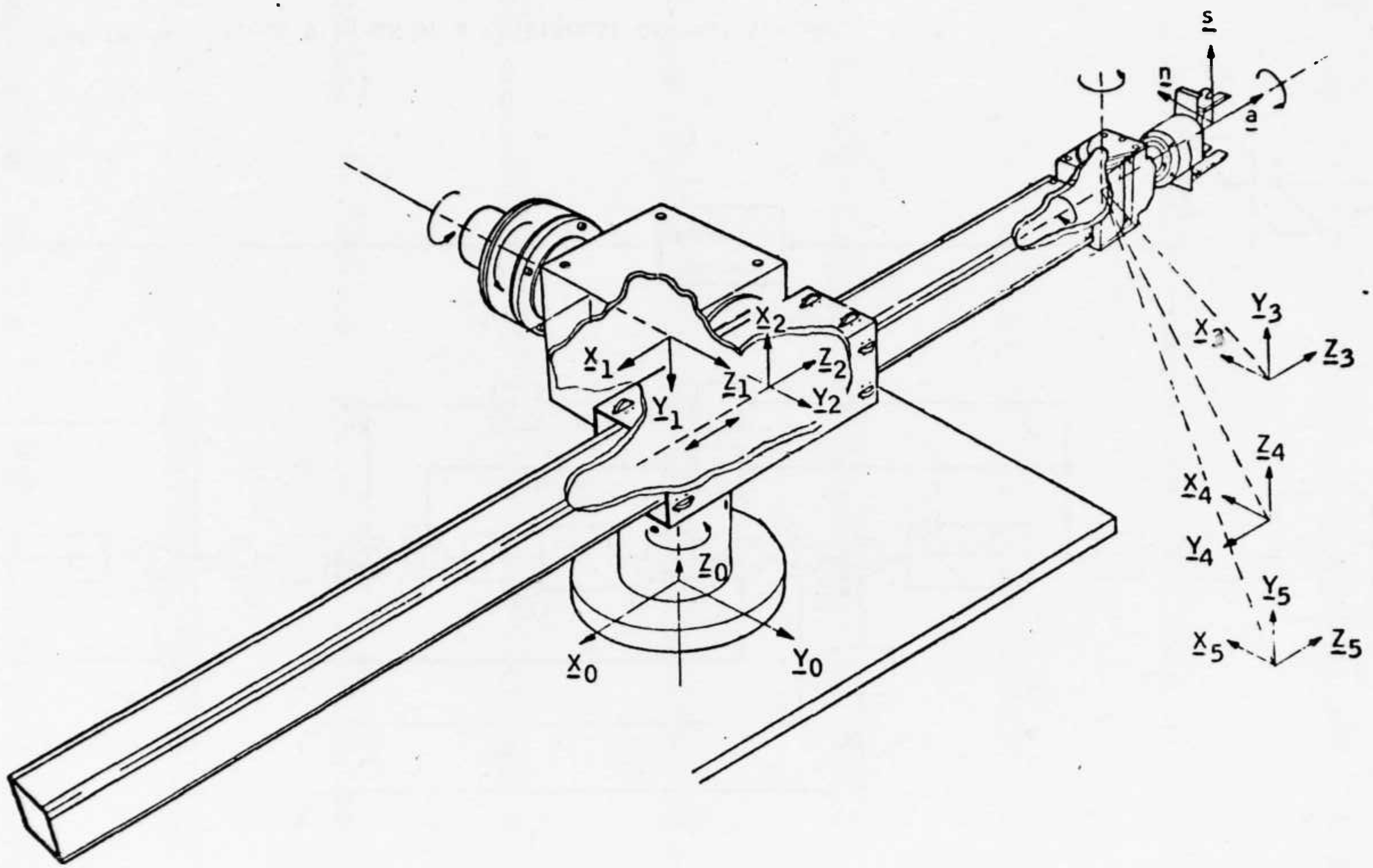


Figure 3. The Stanford Arm

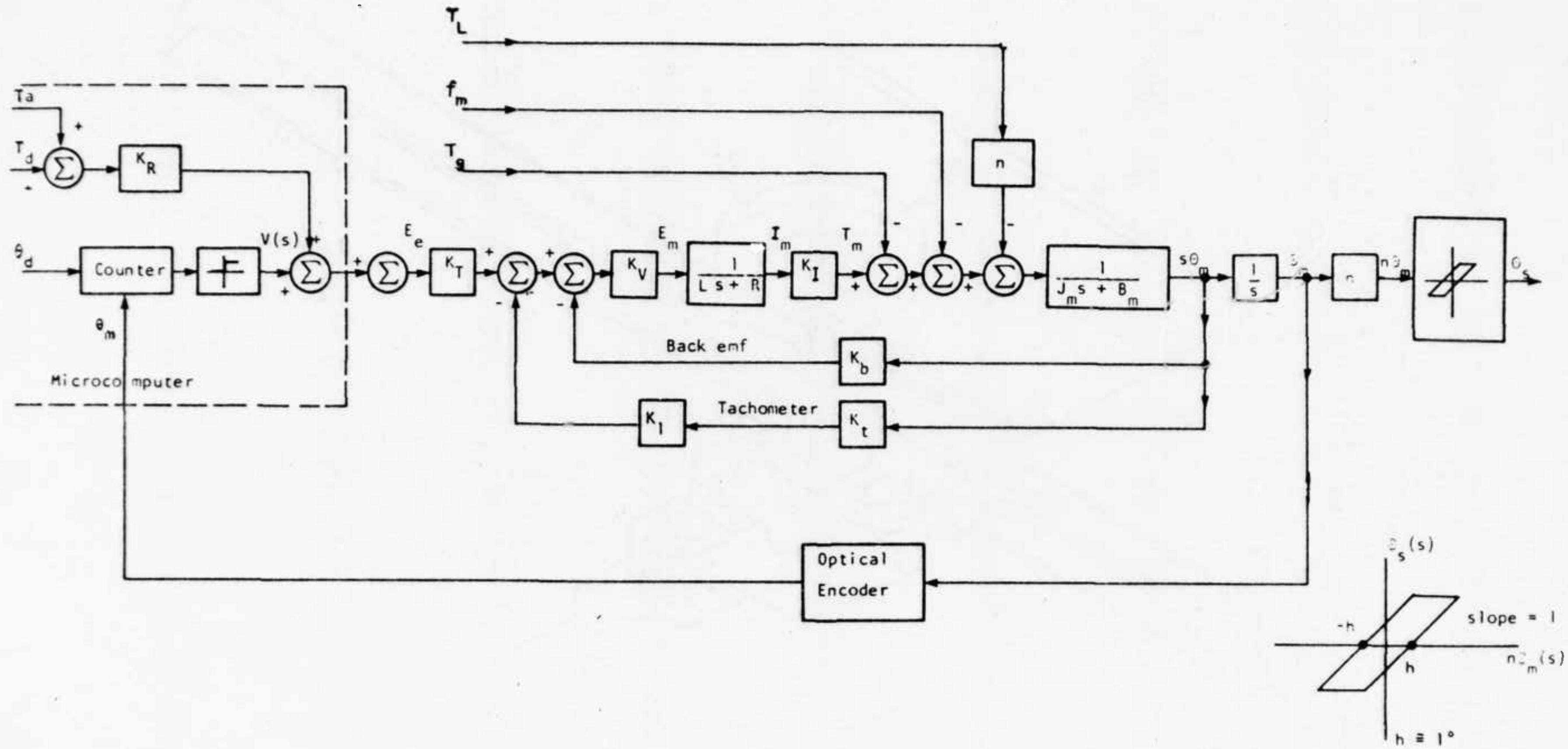


Figure 4. Block Diagram of a Positional Control System

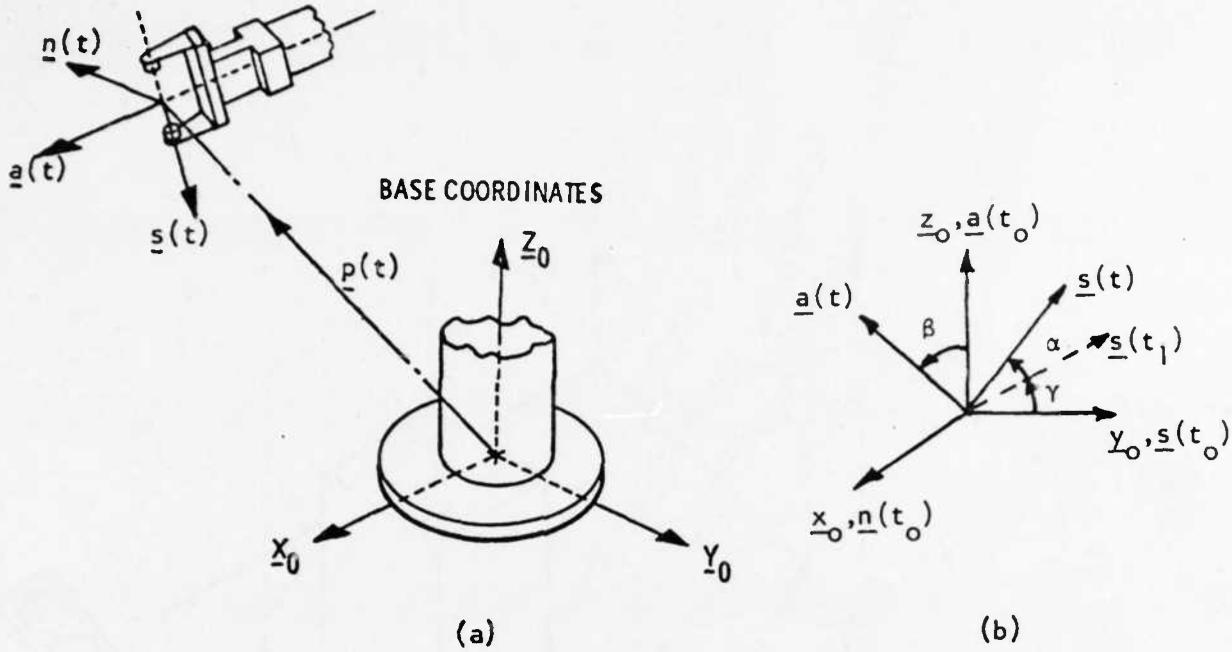


Figure 5. (a) Position and Orientation Vectors of the Hand
(b) Euler Angles of Orientation

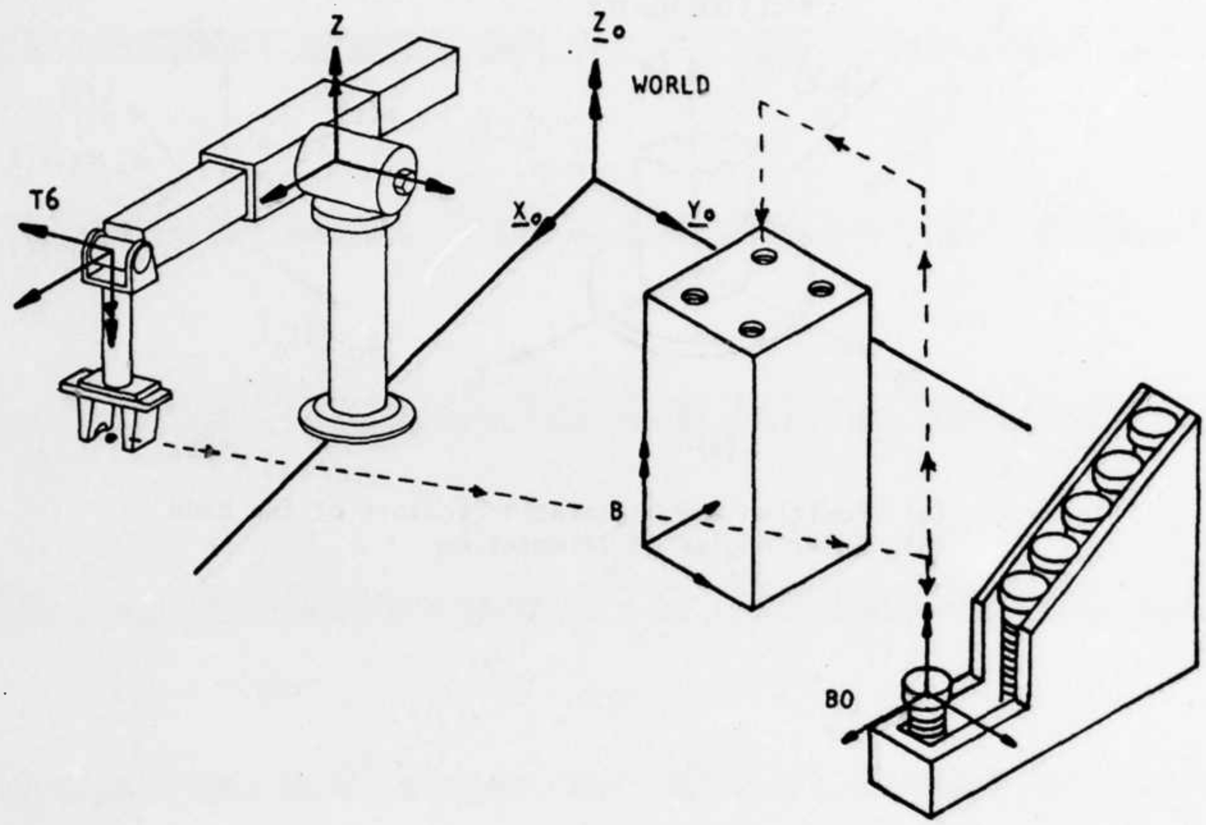


Figure 6. Simple Robot Task for Illustration

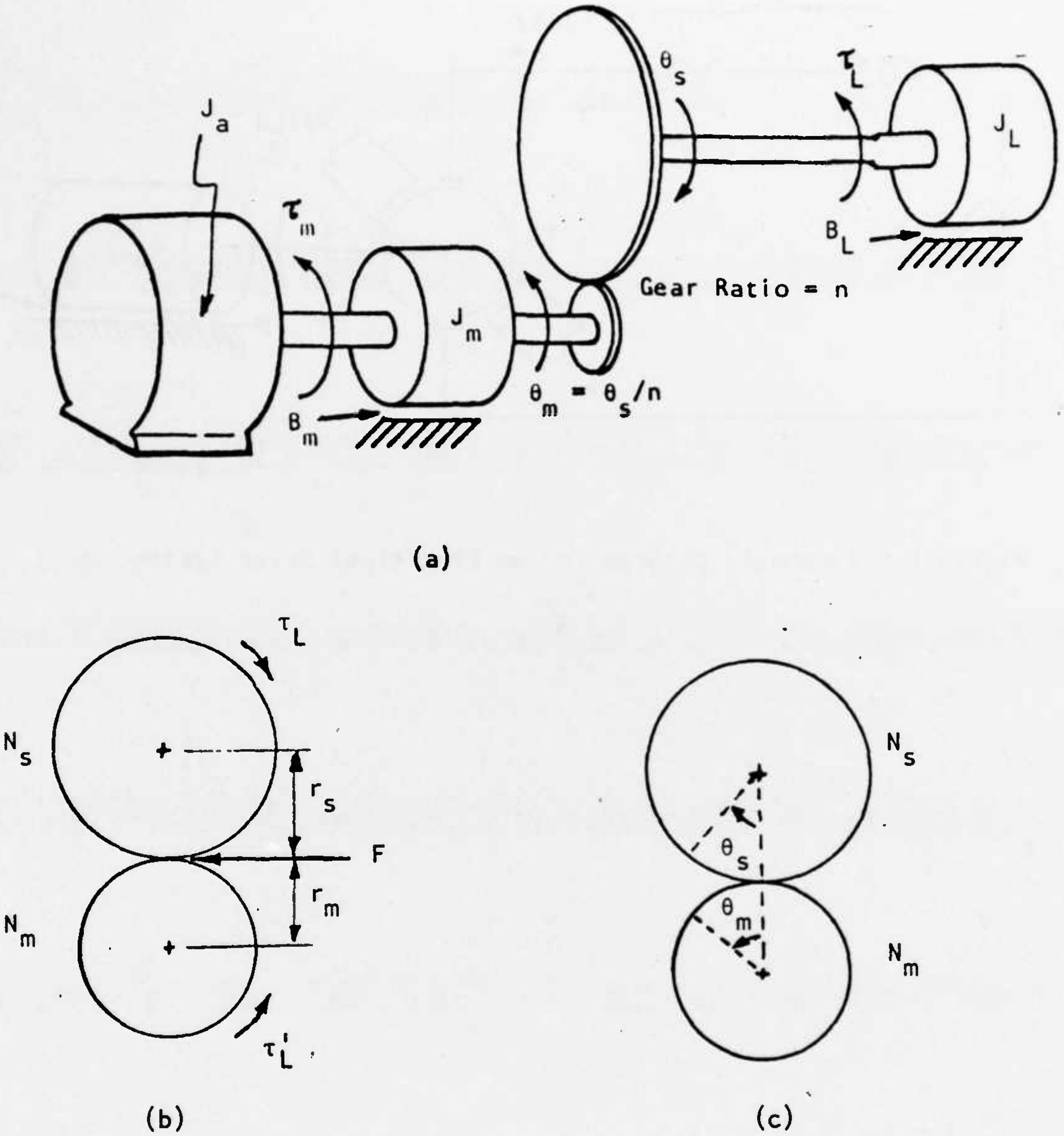


Figure 7. Schematic Representation of an Actuator-Gear-Load Assembly for One Joint.

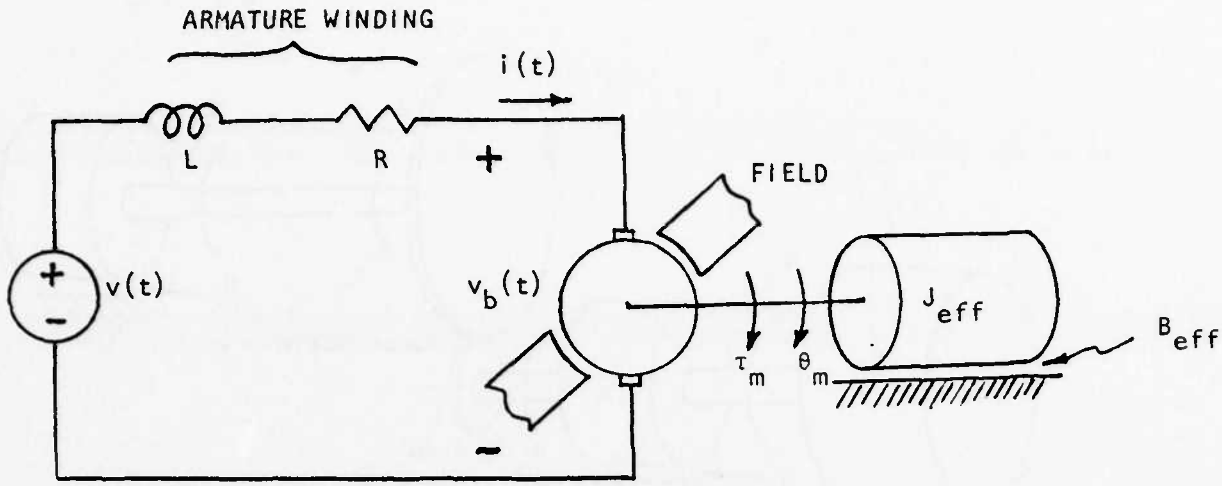


Figure 8. Schematic Diagram for an Electrical Drive System.

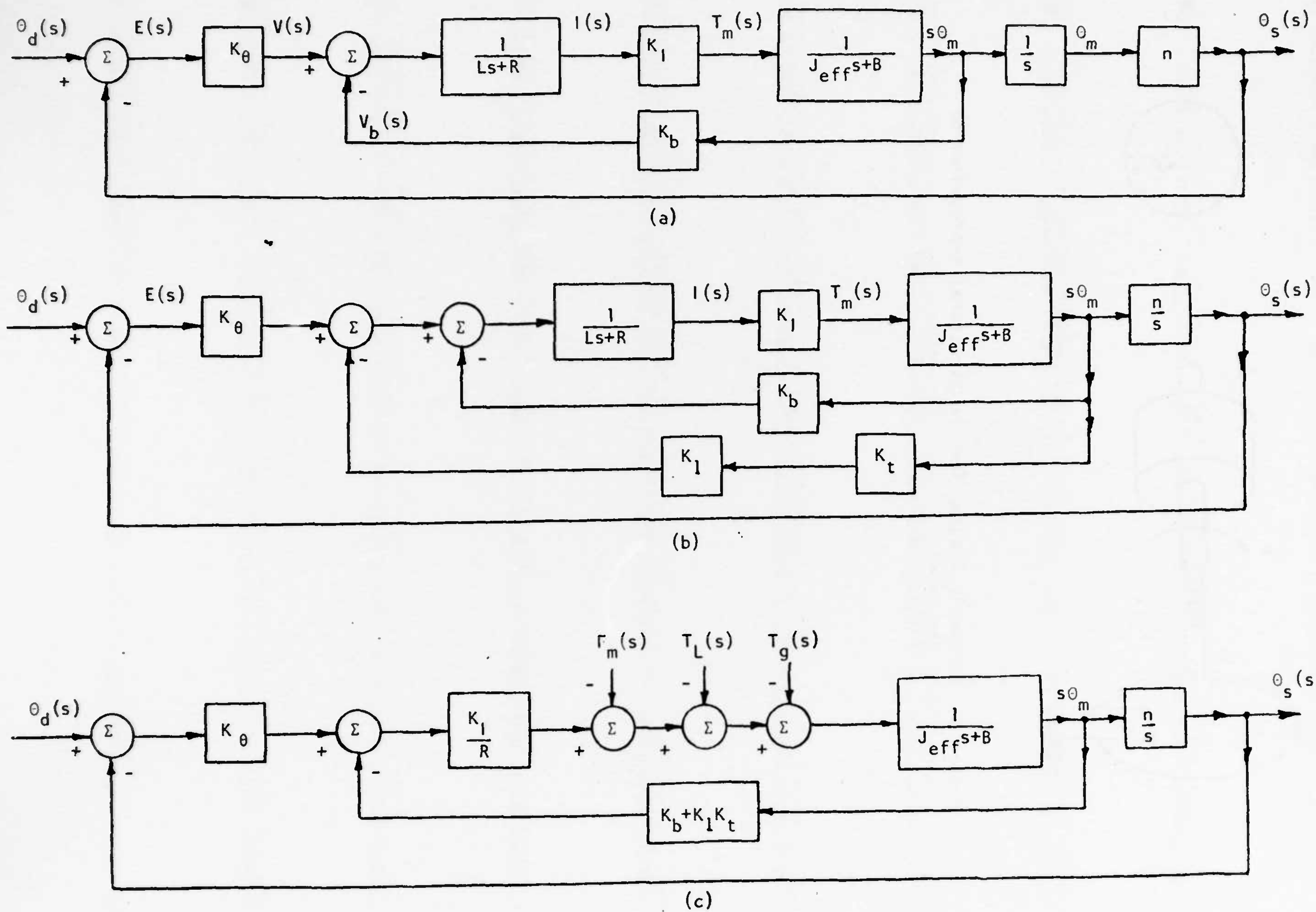


Figure 9. Block Diagram of a Positional Controller.

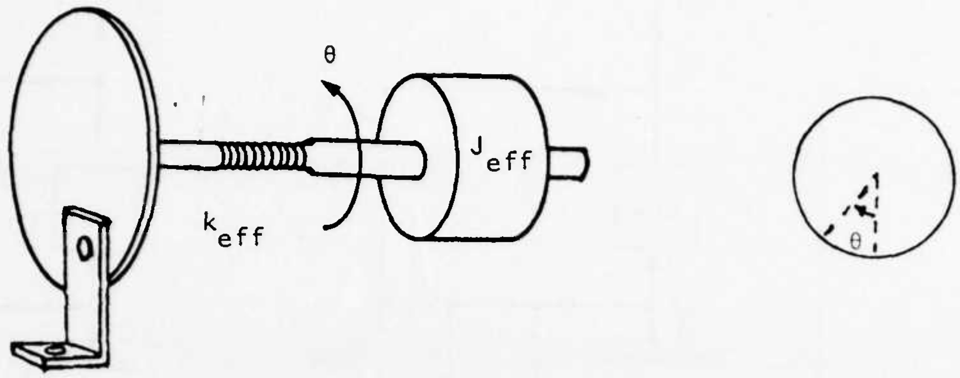


Figure 10. Schematic Diagram for the Structural Representation of One Joint.

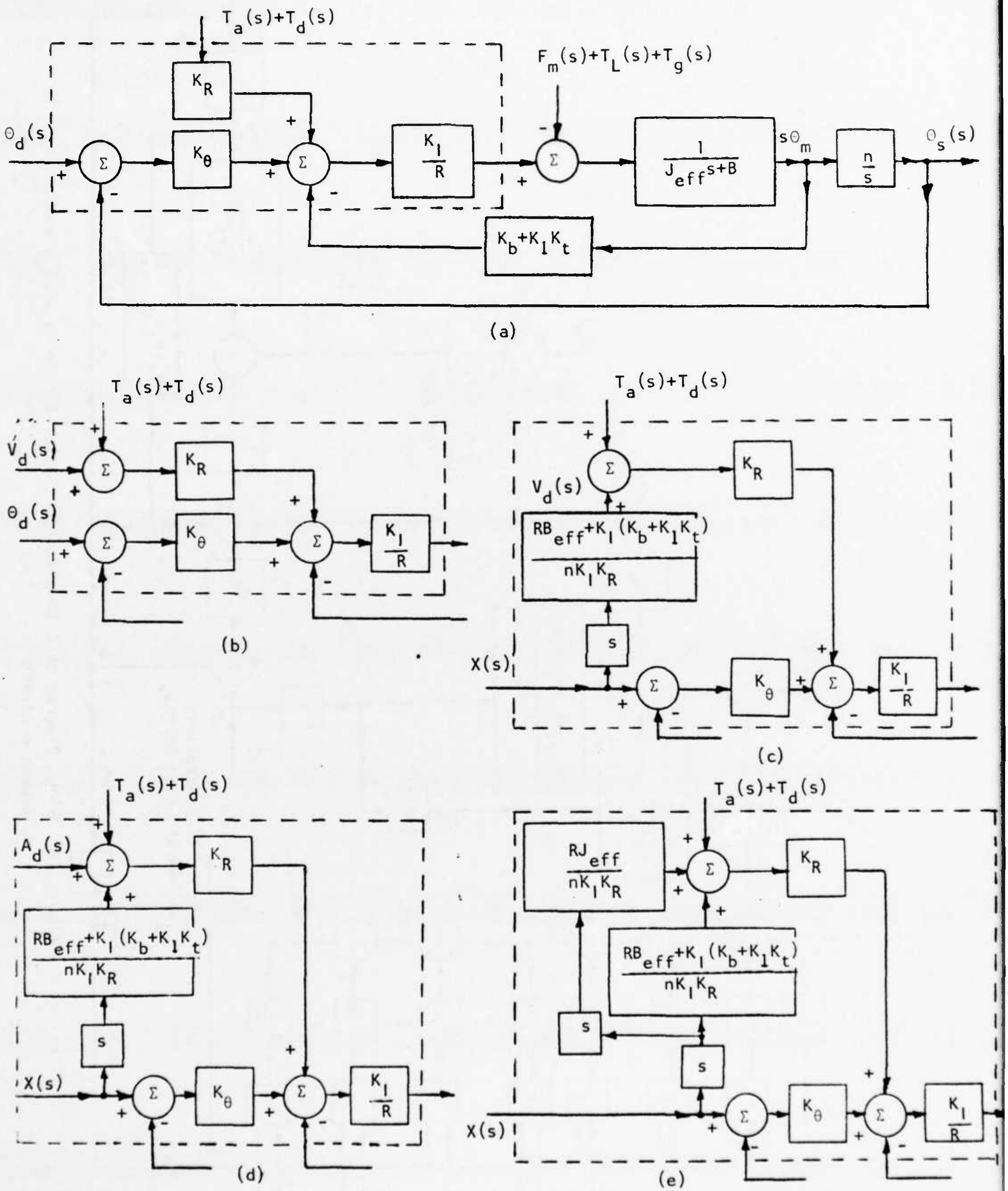


Figure 11. Controller with Anticipated Burden and Feedforward Compensation

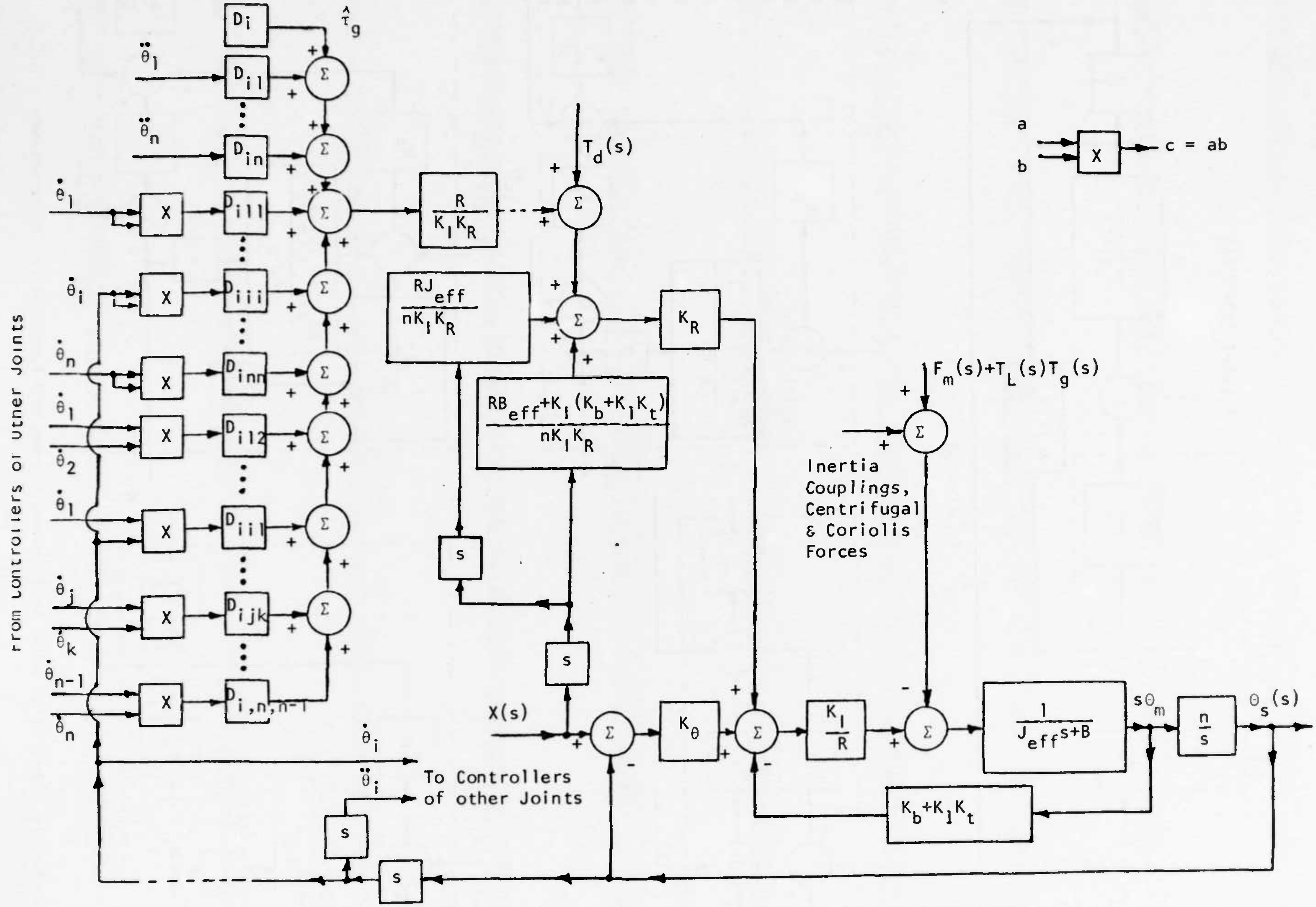


Figure 12. Block Diagram of a Complete Controller for Joint i of a Robot Having n Joints.

$$
\begin{aligned}
D_{11} = {} & m_1 k_{122}^2 \\
& + m_2 \left[k_{211}^2 s^2\theta_2 + k_{233}^2 c^2\theta_2 + r_2(2\bar{y}_2 + r_2) \right] \\
& + m_3 \left[k_{322}^2 s^2\theta_2 + k_{333}^2 c^2\theta_2 + r_3(2\bar{z}_3 + r_3) s^2\theta_2 + r_2^2 \right] \\
& + m_4 \Big\{ \tfrac{1}{2} k_{411}^2 \left[s^2\theta_2(2s^2\theta_4 - 1) + s^2\theta_4 \right] + \tfrac{1}{2} k_{422}^2 (1 + c^2\theta_2 + s^2\theta_4) \\
& \qquad + \tfrac{1}{2} k_{433}^2 \left[s^2\theta_2(1 - 2s^2\theta_4) - s^2\theta_4 \right] + r_3^2 s^2\theta_2 + r_2^2 - 2\bar{y}_4 r_3 s^2\theta_2 + 2\bar{z}_4(r_2 s\theta_4 + r_3 s\theta_2 c\theta_2 c\theta_4) \Big\} \\
& + m_5 \Big\{ \tfrac{1}{2}(-k_{511}^2 + k_{522}^2 + k_{533}^2) \left[(s\theta_2 s\theta_5 - c\theta_2 s\theta_4 c\theta_5)^2 + c^2\theta_4 c^2\theta_5 \right] \\
& \qquad + \tfrac{1}{2}(k_{511}^2 - k_{522}^2 + k_{533}^2)(s^2\theta_4 + c^2\theta_2 c^2\theta_4) \\
& \qquad + \tfrac{1}{2}(k_{511}^2 + k_{522}^2 - k_{533}^2) \left[(s\theta_2 c\theta_5 + c\theta_2 s\theta_4 s\theta_5)^2 + c^2\theta_4 s^2\theta_5 \right] + r_3^2 s^2\theta_2 + r_2^2 \\
& \qquad + 2\bar{z}_5 \left[r_3(s^2\theta_2 c\theta_5 + s\theta_2 s\theta_4 c\theta_4 s\theta_5) - r_2 c\theta_4 s\theta_5 \right] \Big\} \\
& + m_6 \Big\{ \tfrac{1}{2}(-k_{611}^2 + k_{622}^2 + k_{633}^2) \left[(s\theta_2 s\theta_5 c\theta_6 - c\theta_2 s\theta_4 c\theta_5 c\theta_6 - c\theta_2 c\theta_4 s\theta_6)^2 + (c\theta_4 c\theta_5 c\theta_6 - s\theta_4 s\theta_6)^2 \right] \\
& \qquad + \tfrac{1}{2}(k_{611}^2 - k_{622}^2 + k_{633}^2) \left[(c\theta_2 s\theta_4 c\theta_5 s\theta_6 - s\theta_2 s\theta_5 s\theta_6 - c\theta_2 c\theta_4 c\theta_6)^2 + (c\theta_4 c\theta_5 s\theta_6 + s\theta_4 c\theta_6)^2 \right] \\
& \qquad + \tfrac{1}{2}(k_{611}^2 + k_{622}^2 - k_{633}^2) \left[(c\theta_2 s\theta_4 s\theta_5 + s\theta_2 c\theta_5)^2 + c^2\theta_4 s^2\theta_5 \right] \\
& \qquad + \left[r_6 c\theta_2 s\theta_4 s\theta_5 + (r_6 c\theta_5 + r_3) s\theta_2 \right]^2 + (r_6 c\theta_4 s\theta_5 - r_2)^2 \\
& \qquad + 2\bar{z}_6 \Big[r_6(s^2\theta_2 c^2\theta_5 + c^2\theta_4 s^2\theta_5 + c^2\theta_2 s^2\theta_4 s^2\theta_5 + 2s\theta_2 c\theta_2 s\theta_4 s\theta_5 c\theta_5) \\
& \qquad\qquad + r_3(s\theta_2 c\theta_2 s\theta_4 s\theta_5 + s^2\theta_2 c\theta_5) - r_2 c\theta_4 s\theta_5 \Big] \Big\}
\end{aligned}
$$

Figure 13. Coefficient of Inertia Term for Joint 1.

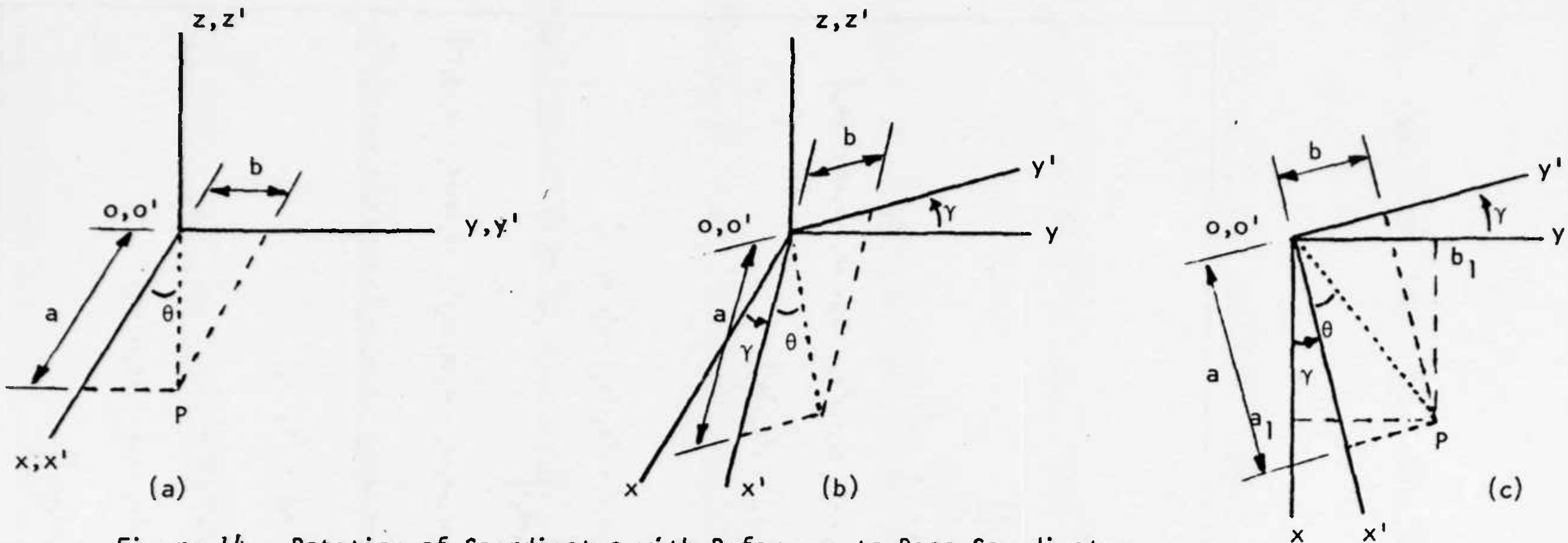


Figure 14. Rotation of Coordinates with Reference to Base Coordinates.

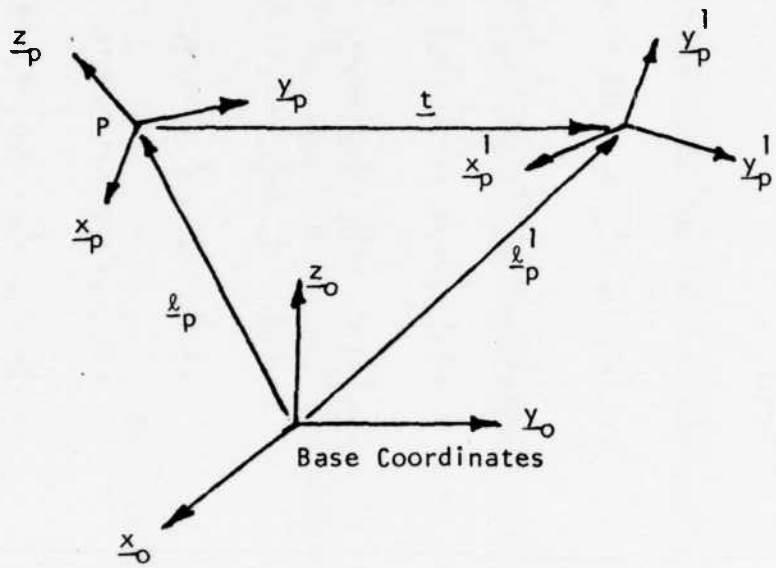


Figure 15. Rotation and Translation of Coordinates with Reference to Base Coordinates.

Differential Kinematic Control Equations for Simple Manipulators

RICHARD P. PAUL, SENIOR MEMBER, IEEE, BRUCE SHIMANO, AND GORDON E. MAYER

Abstract—**The Jacobian representing the differential change in position and orientation of the manipulator end-effector in terms of differential changes in joint coordinates will be presented together with the inverse Jacobian.**

Introduction

A general method for obtaining the kinematic equations for any manipulator has been developed and demonstrated for the PUMA manipulator [2]. The kinematic equations have the homogeneous transformation representing the position and orientation of the end-effector as the dependent variable and the joint coordinates as the independent variables. In the case of simple manipulators, such as the PUMA arm, it is possible to solve these equations to obtain the joint coordinates given the position and orientation of the end-effector [2].

In this correspondence we develop a general method for obtaining the Jacobian with differential change in position and orientation as dependent variables and differential change in joint coordinates as independent variables. A method of obtaining the Jacobian was developed by Uicker in terms of the differential change of transform elements [4]. Groome [1] developed the Jacobian in terms of a differential translation and rotation, based on vector methods. Whitney and Klumpp [3] adapted this method to the standard form of assigning manipulation coordinates. Their results were obtained in a coordinate frame located at the manipulator's end-effector but aligned with the base coordinate frame. A further matrix multiplication was required to obtain the manipulator Jacobian. In this correspondence we combine the matrix methods of Uicker with the vector representation of Groome to provide a straightforward algorithm to obtain the Jacobian directly from the kinematic equations. The method leads almost directly to equations resulting in the minimum number of arithmetic operations.

If the manipulator has a solution obtained by the methods developed in [2], then the Inverse Jacobian can be obtained directly by differentiating the solution. This task is enormously simplified by performing this differentiation in terms of the change in the transformation elements and in terms of the differential change of each joint coordinate as it is obtained. This results in very simple expressions for the differential change. The equations representing the kinematic equations and their solution for the PUMA arm are included in Appendix I for reference. It is assumed that the reader is aware of their derivation and significance, if not, it is suggested that [2] be reviewed.

The Jacobian

The differential change dT, of any transformation T can be expressed in terms of the differential change of its elements or in terms of a differential translation $d_x i + d_y j + d_z k$ and a differential rotation $\delta_x i + \delta_y j + \delta_z k$ with respect to T. The relationship between the two forms is given by

$$T + dT = T \cdot (I + \Delta_T) \tag{1}$$

where

$$dT = \begin{bmatrix} dn_x & do_x & da_x & dp_x \\ dn_y & do_y & da_y & dp_y \\ dn_z & do_z & da_z & dp_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

and

$$\Delta_T = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3}$$

Thus

$$dT = T \cdot \Delta_T. \tag{4}$$

In the case of a manipulator, T_6 describes the end of the manipulator. We will evaluate Δ_{T6} as a function of changes of joint coordinates, and if dT_6 is desired then we simply premultiply Δ_{T6} by T_6 to obtain dT_6. If in the case of a manipulator we were to make a change with respect to a link coordinate frame $n-1$ of Δ_n we could find an equivalent change in T_6, Δ_{T6} expressed as

$$T_6 \cdot \Delta_{T6} = A_1 \cdot A_2 \cdot \cdots \cdot A_{n-1} \cdot \Delta_n \cdot A_n \cdot \cdots \cdot A_6 \tag{5}$$

or on simplifying

$$\Delta_{T6} = (A_n \cdot \cdots \cdot A_6)^{-1} \cdot \Delta_n \cdot (A_n \cdot \cdots \cdot A_6). \tag{6}$$

If link n follows a revolute joint then a change of joint coordinate $d\theta_n$ corresponds to a rotation about the z axis of the link $n-1$ coordinate frame or

$$\Delta_{\text{revolute}} = \begin{bmatrix} 0 & -d\theta & 0 & 0 \\ d\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{7}$$

If the link follows a prismatic joint then the change of joint coordinate dd_n, corresponds to a translation along the z axis of the link_{n-1} coordinate frame:

$$\Delta_{\text{prismatic}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & dd \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{8}$$

If we define U_n as $U_n = (A_n \cdot A_{n+1} \cdot \cdots \cdot A_6)$ with elements

$$U_n = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{9}$$

Then

$$\Delta_{T6} = U_n^{-1} \cdot \Delta_{\text{revolute}} \cdot U_n \tag{10}$$

and

$$\Delta_{T6} = \begin{bmatrix} 0 & o_x n_y - o_y n_x & a_x n_y - a_y n_x & p_x n_y - p_y n_x \\ n_x o_y - n_y o_x & 0 & a_x o_y - a_y o_x & p_x o_y - p_y o_x \\ n_x a_y - n_y a_x & o_x a_y - o_y a_x & 0 & p_x a_y - p_y a_x \\ 0 & 0 & 0 & 0 \end{bmatrix} d\theta_n. \tag{11}$$

Manuscript received December 10, 1980; revised March 30, 1981. This material is based upon research supported by the National Science Foundation under Grants APR77-14533, APR75-13074, and APR74-01390. Any opinions, findings, and conclusions or recommendations in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

R. P. Paul is with the Advanced Technology Laboratory, GTE Laboratories Inc., 40 Sylvan Road, Waltham, MA 02254, on leave from the Department of Electrical Engineering, Purdue University, West Lafayette, IN 47905.

B. Shimano is with the West Coast Division of Unimation Inc., 5841A Uplander Way, Culver City, CA 90230

G. E. Mayer is with the Wright-Patterson AFB, AFWAL/MLTC, OH 45433.

0018-9472/81/0600-0456$00.75

Treating $\boldsymbol{n}$, $\boldsymbol{o}$, and $\boldsymbol{a}$ as vectors we can rewrite this in terms of the vector cross product:

$$\Delta_{T6}=\begin{bmatrix} 0 & (\boldsymbol{o}\times\boldsymbol{n})_z & (\boldsymbol{a}\times\boldsymbol{n})_z & (\boldsymbol{p}\times\boldsymbol{n})_z \\ (\boldsymbol{n}\times\boldsymbol{o})_z & 0 & (\boldsymbol{a}\times\boldsymbol{o})_z & (\boldsymbol{p}\times\boldsymbol{o})_z \\ (\boldsymbol{n}\times\boldsymbol{a})_z & (\boldsymbol{o}\times\boldsymbol{a})_z & 0 & (\boldsymbol{p}\times\boldsymbol{a})_z \\ 0 & 0 & 0 & 0 \end{bmatrix} d\theta_n. \tag{12}$$

The cross products of orthogonal unit vectors are equal to other unit vectors:

$$\Delta_{T6}=\begin{bmatrix} 0 & -a_z & o_z & (p\times n)_z \\ a_z & 0 & -n_z & (p\times o)_z \\ -o_z & n_z & 0 & (p\times a)_z \\ 0 & 0 & 0 & 0 \end{bmatrix} d\theta_n. \tag{13}$$

In the case of a prismatic joint

$$\Delta_{T6}=U_n^{-1}*\Delta_{\text{prismatic}}*U_n \tag{14}$$

and

$$\Delta_{T6}=\begin{bmatrix} 0 & 0 & 0 & n_z \\ 0 & 0 & 0 & o_z \\ 0 & 0 & 0 & a_z \\ 0 & 0 & 0 & 0 \end{bmatrix} dd_n. \tag{15}$$

If we write ΔT_6 in the form of a column vector representing a differential translation and rotation we may write:

$$\begin{bmatrix} T_6 d_x \\ T_6 d_y \\ T_6 d_z \\ T_6 \delta_x \\ T_6 \delta_y \\ T_6 \delta_z \end{bmatrix} = \begin{bmatrix} J \end{bmatrix} * \begin{bmatrix} dq_1 \\ dq_2 \\ dq_3 \\ dq_4 \\ dq_5 \\ dq_6 \end{bmatrix} \tag{16}$$

where $dq_i = d\theta_i$ if the joint is revolute and $dq_i = dd_i$ if the joint is prismatic. Therefore the matrix J, the Jacobian, consists of six columns of the form:

revolute	prismatic
$(p\times n)_z$	n_z
$(p\times o)_z$	o_z
$(p\times a)_z$	a_z
n_z	0
o_z	0
a_z	0

where n, o, a, and p are the columns of U where

$$U_n = \left(A_n * A_{n+1} * \cdots * A_6\right). \tag{17}$$

For example, using the values of U_n (62)–(79) developed for the PUMA arm in [2], we may evaluate the expressions given for the columns of the Jacobian and obtain the Jacobian matrix for this arm:

$$\begin{bmatrix} dT_{6x} \\ dT_{6y} \\ dT_{6z} \\ \delta T_{6x} \\ \delta T_{6y} \\ \delta T_{6z} \end{bmatrix} = \begin{bmatrix}
(d_4S_{23}+a_3C_{23}+a_2C_2)(S_4C_5C_6+C_4S_6)-d_3[C_{23}(C_4C_5C_6-S_4S_6)-S_{23}S_5C_6] & (d_4S_3+a_3C_3+a_2)(S_5C_6)-(-d_4C_3+a_3S_3)(C_4C_5C_6-S_4S_6) & a_3(S_5C_6)+d_4(C_4C_5C_6-S_4S_6) & 0 & 0 & 0 \\
(d_4S_{23}+a_3C_{23}+a_2C_2)(-S_4C_5S_6+C_4C_6)-d_3[-C_{23}(C_4C_5S_6+S_4C_6)+S_{23}S_5S_6] & -(d_4S_3+a_3C_3+a_2)(S_5S_6)+(-d_4C_3+a_3S_3)(C_4C_5S_6+S_4C_6) & -a_3(S_5S_6)-d_4(C_4C_5S_6+S_4C_6) & 0 & 0 & 0 \\
(d_4S_{23}+a_3C_{23}+a_2C_2)(S_4S_5)-d_3(C_{23}C_4S_5+S_{23}C_5) & -(d_4S_3+a_3C_3+a_2)C_5-(-d_4C_3+a_3S_3)(C_4S_5) & -a_3C_5+d_4(C_4S_5) & 0 & 0 & 0 \\
-[S_{23}(C_4C_5C_6-S_4S_6)+C_{23}S_5C_6] & S_4C_5C_6+C_4S_6 & S_4C_5C_6+C_4S_6 & -S_5C_6 & S_6 & 0 \\
S_{23}(C_4C_5S_6+S_4C_6)+C_{23}S_5S_6 & -S_4C_5S_6+C_4C_6 & -S_4C_5S_6+C_4C_6 & S_5S_6 & C_6 & 0 \\
S_{23}C_4S_5-C_{23}C_5 & S_4S_5 & S_4S_5 & C_5 & 0 & 1
\end{bmatrix} * \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ d\theta_3 \\ d\theta_4 \\ d\theta_5 \\ d\theta_6 \end{bmatrix} \tag{18}$$

Assuming that the solution has been evaluated, the computation of the Jacobian requires 51 multiplies and 24 additions. Each solution of (16) requires 36 multiplies and 30 additions. The earlier method of Whitney and Klumpp [3] involves an additional 108 multiplies and 72 additions to compute the Jacobian.

Differential Change in Position

In manipulation we frequently need a solution to (16), that is, given a desired differential change in position and orientation d_x, d_y, d_z, δ_x, δ_y, and δ_z what is the required differential change in the joint coordinates dq_i:

$$\begin{bmatrix} dq_1 \\ dq_2 \\ dq_3 \\ dq_4 \\ dq_5 \\ dq_6 \end{bmatrix} = \begin{bmatrix} J \end{bmatrix}^{-1} \cdot \begin{bmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix}. \tag{19}$$

It is sometimes possible to invert the Jacobian symbolically but this is difficult as the expression of the elements of the Jacobian (18) are quite complicated. A numeric solution to (16) can be obtained, but this is typically far too slow and complicated by the Jacobian becoming singular whenever the manipulator becomes degenerate. An even worse approach is to invert the Jacobian numerically as this requires six solutions to (16). The approach we will follow is to differentiate the solution obtained for the joint coordinates given a value of T_6 [2]. This method gives us expressions for differential changes in the same order as we obtained the joint coordinates. Manipulator degeneracies are easily identified. The expressions for each differential change in joint coordinates are a function of change in T_6, and also of the differential joint angle changes already obtained. This frequently results in simpler expressions for the differential changes and if, due to joint constraints, a change is not possible, then the change may be set to zero, resulting in a correct solution for the following joints. We will assume that a solution has been obtained and that the sines and cosines of the joint angles are available.

The equations relating to the solution for the PUMA arm are included in Appendix I, and we will illustrate the method by obtaining the inverse Jacobian for this arm. Equation (80) specifies θ_1, for the PUMA arm which we may differentiate to obtain an expression for $d\theta_1$:

$$d\theta_1 = \frac{C_1 dp_y - S_1 dp_x}{C_1 p_x + S_1 p_y}. \tag{20}$$

If the denominator is zero the manipulator is degenerate and $d\theta_1$ may be assigned arbitrarily. We check, as we obtain each differential change, in case $\theta + d\theta$ exceeds the joint motion limit. If this happens we set $d\theta$ to

$$d\theta = \text{limit} - \theta \tag{21}$$

then $\theta + d\theta$ equals the joint motion limit. As we obtain each differential change $d\theta_i$, we also evaluate $d(\sin\theta_i)$ and $d(\cos\theta_i)$ as

$$dS_i = C_i\, d\theta_i \tag{22}$$

$$dC_i = -S_i\, d\theta_i. \tag{23}$$

We have θ_3 specified by (81) through (83), and we define

$$v = 2a_2 d_4 S_3 + 2a_2 a_3 c_3, \tag{24}$$

differentiating, we obtain

$$d\theta_3 = \frac{d_v}{2a_2(d_4 C_3 - a_3 S_3)} \tag{25}$$

where

$$d_v = 2f_{11}(p)\, df_{11}(p) + 2f_{12}(p)\, df_{12}(p) \tag{26}$$

and $d_{f11}(p)$ and $d_{f12}(p)$ may be obtained by differentiating (82) and (83).

We now solve for $d\theta_{23}$ using (84)–(86), and define

$$-v_1 = w_2 f_{11}(p) - w_1 p_z \tag{27}$$

$$v_2 = w_1 f_{11}(p) + w_2 p_z. \tag{28}$$

Therefore

$$S_{23} v_2 + C_{23} v_1 = 0. \tag{29}$$

Differentiating, we obtain

$$d\theta_{23} = \frac{-S_{23} dv_2 - C_{23} dv_1}{C_{23} v_2 - S_{23} v_1} \tag{30}$$

where dv_1, dv_2, dw_1, and dw_2 may be explicitly obtained by direct differentiation.

Thus

$$d\theta_2 = d\theta_{23} - d\theta_3. \tag{31}$$

In general, if

$$\tan\theta = \frac{N\sin\theta}{N\cos\theta} \tag{32}$$

where N is any variable then

$$d\theta = \frac{N\cos\theta\, d(N\sin\theta) - N\sin\theta\, d(N\cos\theta)}{(N\sin\theta)^2 + (N\cos\theta)^2}. \tag{33}$$

From (87) and (88), we obtain expressions for NS_4 and NC_4 and their derivatives

$$NS_4 = -S_1 a_x + C_1 a_y \tag{34}$$

$$d(NS_4) = -dS_1 a_x - S_1 da_x + dC_1 a_y + C_1 da_y \tag{35}$$

$$NC_4 = C_{23} D_{41} - S_{23} a_z \tag{36}$$

where

$$D_{41} = C_1 a_x + S_1 a_y \tag{37}$$

$$dD_{41} = dC_1 a_x + C_1 da_x + dS_1 a_y + S_1 da_y \tag{38}$$

so,

$$d(NC_4) = dC_{23} D_{41} + C_{23} dD_{41} - dS_{23} a_z - S_{23} da_z \tag{39}$$

and $d\theta_4$ is then obtained from (33) as

$$d\theta_4 = \frac{NC_4 d(NS_4) - NS_4 d(NC_4)}{(NS_4)^2 + (NC_4)^2}. \tag{40}$$

Once again a zero denominator indicates a manipulator degeneracy and $d\theta_4$ may then be assigned arbitrarily.

In the case of θ_5 we have expressions for both sine and consine and the expression for the derivative becomes

$$d\theta = C\theta\, d(S\theta) - S\theta\, d(C\theta). \tag{41}$$

When we have expressions for both the sine and cosine the manipulator cannot become degenerate. We have from (89) and (90):

$$S_5 = C_4 NC_4 + S_4 NS_4 \tag{42}$$

$$dS_5 = dC_4 NC_4 + C_4 d(NC_4) + dS_4 NS_4 + S_4 d(NS_4) \tag{43}$$

$$C_5 = S_{23} D_{41} + C_{23} a_z \tag{44}$$

$$dC_5 = dS_{23} D_{41} + S_{23} dD_{41} + dC_{23} a_z + C_{23} da_z. \tag{45}$$

$d\theta_5$ is then obtained from (33) as

$$d\theta_5 = C_5 dS_5 - S_5 dC_5. \tag{46}$$

In this case we do not need to evaluate dS_5 and dC_5 as they are given above.

Finally, for θ_6 we have from (91) and (92):

$$S_6 = -C_5N_{61} - S_5N_{612} \tag{47}$$

$$C_6 = -S_4N_{611} + C_4N_{6112} \tag{48}$$

where

$$N_{6111} = C_1o_x + S_1o_y \tag{49}$$

$$dN_{6111} = dC_1o_x + C_1do_x + dS_1o_y + S_1do_y \tag{50}$$

$$N_{6112} = -S_1o_x + C_1o_y \tag{51}$$

$$dN_{6112} = -dS_1o_x - S_1do_x + dC_1o_y + C_1do_y \tag{52}$$

$$N_{611} = C_{23}N_{6111} - S_{23}o_z \tag{53}$$

$$dN_{611} = dC_{23}N_{6111} + C_{23}dN_{6111} - dS_{23}o_z - S_{23}do_z \tag{54}$$

$$N_{612} = -S_{23}N_{6111} - C_{23}o_z \tag{55}$$

$$dN_{612} = -dS_{23}N_{6111} - S_{23}dN_{6111} - dC_{23}o_z - C_{23}do_z \tag{56}$$

$$N_{61} = C_4N_{611} + S_4N_{6112} \tag{57}$$

$$dN_{61} = dC_4N_{611} + C_4dN_{611} + dS_4N_{6112} + S_4dN_{6112} \tag{58}$$

then

$$dS_6 = -dC_5N_{61} - C_5dN_{61} - dS_5N_{612} - S_5dN_{612} \tag{59}$$

$$dC_6 = -dS_4N_{611} - S_4dN_{611} + dC_4N_{6112} + C_4dN_{6112} \tag{60}$$

and $d\theta_6$ is obtained from (40) as

$$d\theta_6 = C_6dS_6 - S_6dC_6. \tag{61}$$

The differential solution $d\theta/dT_6$ represents 91 multiplies and 55 additions and no transcendental function calls, assuming that the solution has been evaluated.

Summary

The Jacobian can be developed in a straightforward manner from kinematic equations. These equations and the Jacobian can be obtained for any manipulator. Further if the solution to the kinematic equations has been obtained, the differential change in joint coordinates for a differential change in position and orientation can also be determined.

Appendix

The kinematics of any manipulator are defined by the A matrices [2]. From the matrices we develop the U matrices for the PUMA arm as follows:

$$U_6 = A_6 \tag{62}$$

$$U_5 = \begin{bmatrix} C_5C_6 & -C_5S_6 & S_5 & 0 \\ S_5C_6 & -S_5S_6 & -C_5 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{63}$$

$$U_4 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 & 0 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & 0 \\ -S_5C_6 & S_5S_6 & C_5 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{64}$$

$$U_3 = \begin{bmatrix} C_3(C_4C_5C_6 - S_4S_6) - S_3S_5C_6 & -C_3(C_4C_5S_6 + S_4C_6) + S_3S_5S_6 & C_3C_4S_5 + S_3C_5 & d_4S_3 + a_3C_3 \\ S_3(C_4C_5C_6 - S_4S_6) + C_3S_5C_6 & -S_3(C_4C_5S_6 + S_4C_6) - C_3S_5S_6 & S_3C_4S_5 - C_3C_5 & -d_4C_3 + a_3S_3 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{65}$$

$$U_2 = \begin{bmatrix} C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6 & -C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6 & C_{23}C_4S_5 + S_{23}C_5 & d_4S_{23} + a_3C_{23} + a_2C_2 \\ S_{23}(C_4C_5C_6 - S_4S_6) + C_{23}S_5C_6 & -S_{23}(C_4C_5S_6 + S_4C_6) - C_{23}S_5S_6 & S_{23}C_4S_5 - C_{23}C_5 & -d_4C_{23} + a_3S_{23} + a_2S_2 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{66}$$

$$U_1 = T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{67}$$

where

$$n_x = C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_1[S_4C_5C_6 + C_4S_6] \tag{68}$$

$$n_y = S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + C_1[S_4C_5C_6 + C_4S_6] \tag{69}$$

$$n_z = -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6 \tag{70}$$

$$o_x = C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] - S_1[-S_4C_5S_6 + C_4C_6] \tag{71}$$

$$o_y = S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] + C_1[-S_4C_5S_6 + C_4C_6] \tag{72}$$

$$o_z = S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6 \tag{73}$$

$$a_x = C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5 \tag{74}$$

$$a_y = S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5 \tag{75}$$

$$a_z = -S_{23}C_4S_5 + C_{23}C_5 \tag{76}$$

$$p_x = C_1(d_4S_{23} + a_3C_{23} + a_2C_2) - S_1d_3 \tag{77}$$

$$p_y = S_1(d_4S_{23} + a_3C_{23} + a_2C_2) + C_1d_3 \tag{78}$$

$$p_z = -(-d_4C_{23} + a_3S_{23} + a_2S_2). \tag{79}$$

The solution of the equations represented by $T6$ is presented in [2], the key equations in the solution are

$$-S_1p_x + C_1p_y = d_3 \tag{80}$$

$$f_{11}^2(p) + f_{12}^2(p) - d_4^2 - a_3^2 - a_2^2 = 2a_2d_4S_3 + 2a_2a_3C_3 \tag{81}$$

$$f_{11}(p) = C_1p_x + S_1p_y \tag{82}$$

$$f_{12}(p) = -p_z \tag{83}$$

$$\frac{S_{23}}{C_{23}} = \frac{w_2f_{11}(p) - w_1p_z}{w_1f_{11}(p) + w_2p_z} \tag{84}$$

$$w_1 = a_2c_3 + a_3 \tag{85}$$

$$w_2 = d_4 + a_2S_3 \tag{86}$$

$$C_4S_5 = C_{23}(C_1a_x + S_1a_y) - S_{23}a_z \tag{87}$$

$$S_4S_5 = -S_1a_x + C_1a_y \tag{88}$$

$$S_5 = C_4[C_{23}(C_1a_x + S_1a_y) - S_{23}a_z] + S_4[-S_1a_x + C_1a_y] \tag{89}$$

$$C_5 = S_{23}(C_1a_x + S_1a_y) + C_{23}a_z \tag{90}$$

$$S_6 = -C_5\{C_4[C_{23}(C_1o_x + S_1o_y) - S_{23}o_z] + S_4[-S_1o_x + C_1o_y]\} + S_5\{S_{23}(C_1o_x + S_1o_y) + C_{23}o_z\} \tag{91}$$

$$C_6 = -S_4[C_{23}(C_1o_x + S_1o_y) - S_{23}o_z] + C_4[-S_1o_x + C_1o_y]. \tag{92}$$

Acknowledgment

The solution for a differential change in manipulator position was based on a suggestion by T. Binford. Mickey Krebs was responsible for the document preparation and Marc Ream for preparing the drawings. Richard Gray and Gordon Mayer programed the solutions.

References

[1] R. C. Groome, Jr., "Force feedback steering of a teleoperator system," S.M. thesis, Massachusetts Institute of Tech., Cambridge, MA, 1972.

[2] R. P. Paul, B. Shimano, and G. E. Mayer, "Kinematic control equations to simple manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 449–455, June 1981, this issue.

[3] D. W. Whitney, "The mathematics of coordinated control of prosthetic arms and manipulators," *Trans. ASME, J. Dynamic Syst., Measurement, Contr.*, pp. 303–309, Dec. 1972.

[4] J. J. Uicker, Jr., "Dynamic force analysis of spatial linkages," ASME Paper No. 66-Mech-1, Mechanisms Conf., Lafayette, IN, Oct. 1966.

THIS PAGE LEFT BLANK INTENTIONALLY

SESSION III

DISCLAIMER

This paper has been printed from a xeroxed copy of viewgraphs.

The original viewgrahs are produced in several colors for ease of presentation.

Their original clarity is not reproduced in the xeroxed copy.

The printer takes no responsibility for the quality of this reproduction.

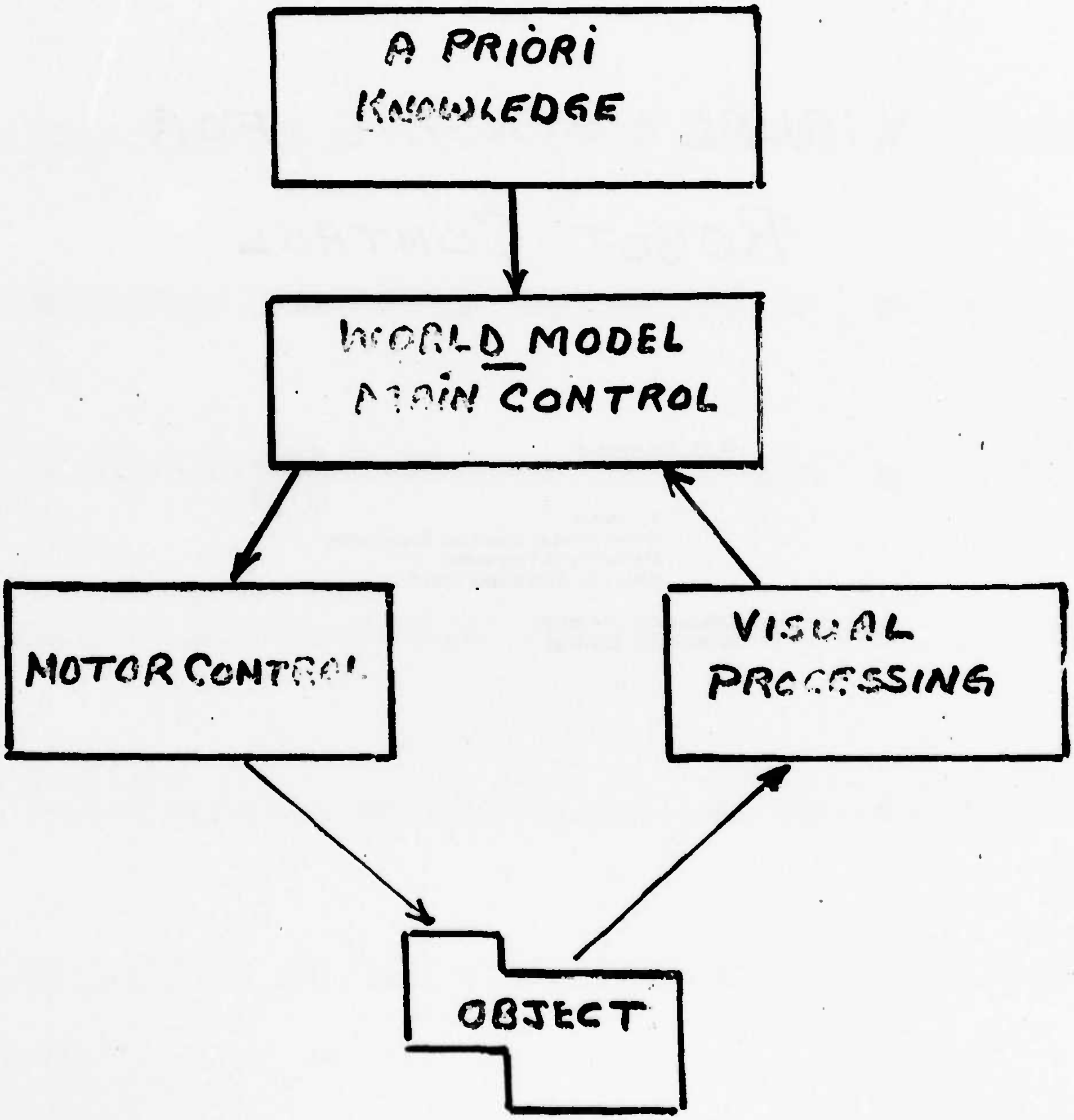
VISUAL SENSING FOR ROBOT CONTROL

R.C. Gonzalez

Professor
Department of Electrical Engineering
University of Tennessee
Knoxville, Tennessee 37916

Office (615) 974-2579
Home (615) 966-1028

STRUCTURE OF A VISUALLY CONTROLLED MANIPULATOR



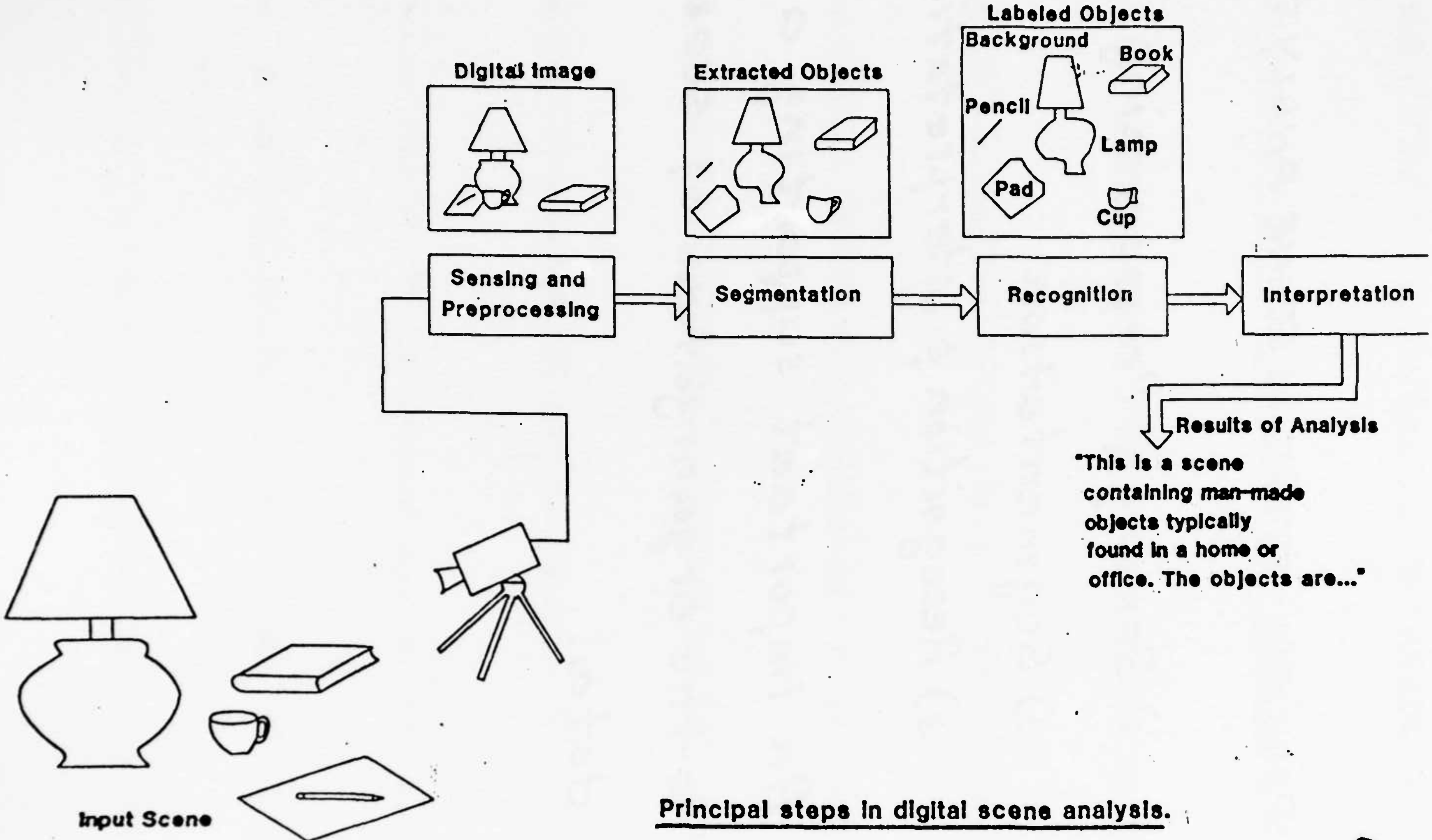
VISUAL Processing for robotics applications is a problem in digital scene analysis.

Digital SCENE ANALYSIS MAY BE DEFINED AS THE PROCESS OF USING A DIGITAL COMPUTER TO EXTRACT, CHARACTERIZE, AND INTERPRET INFORMATION FROM IMAGES OF A THREE-DIMENSIONAL WORLD.

PRINCIPAL STEPS IN SCENE ANALYSIS

1) Sensing & Preprocessing
2) Segmentation
3) Recognition & interpretation

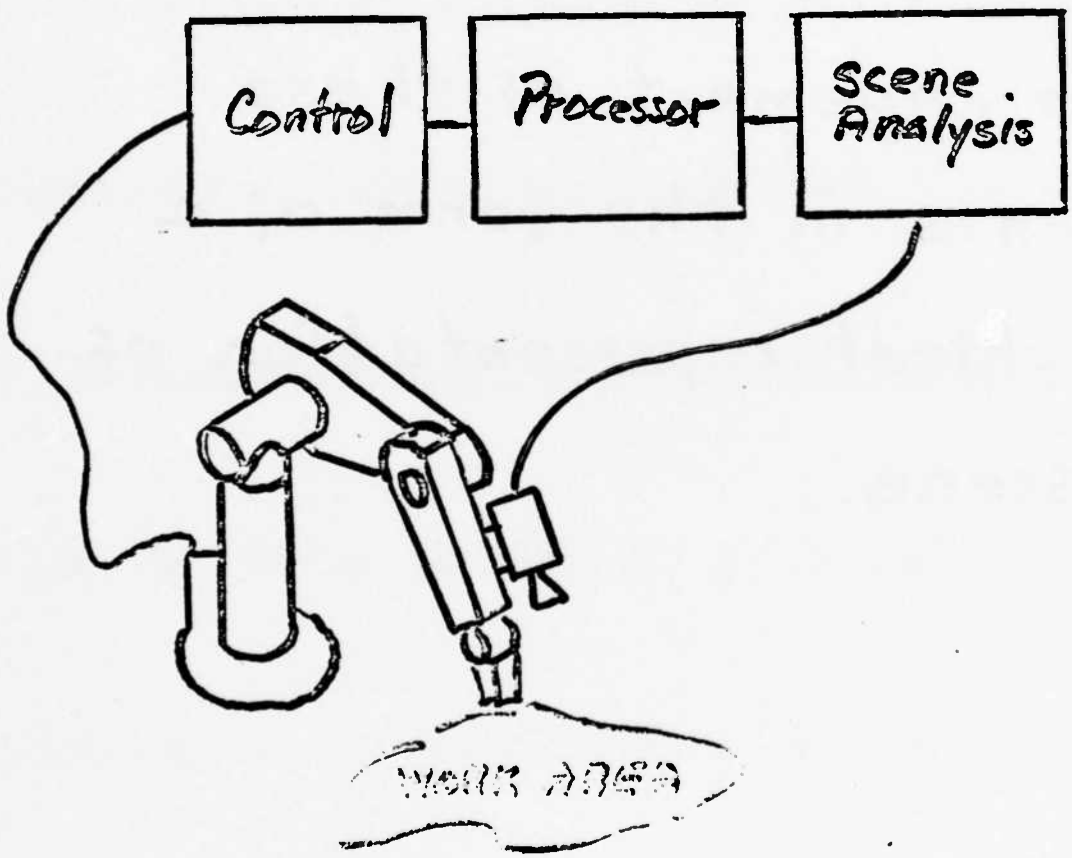
An important supporting area is the organization of scene data.



Principal steps in digital scene analysis.

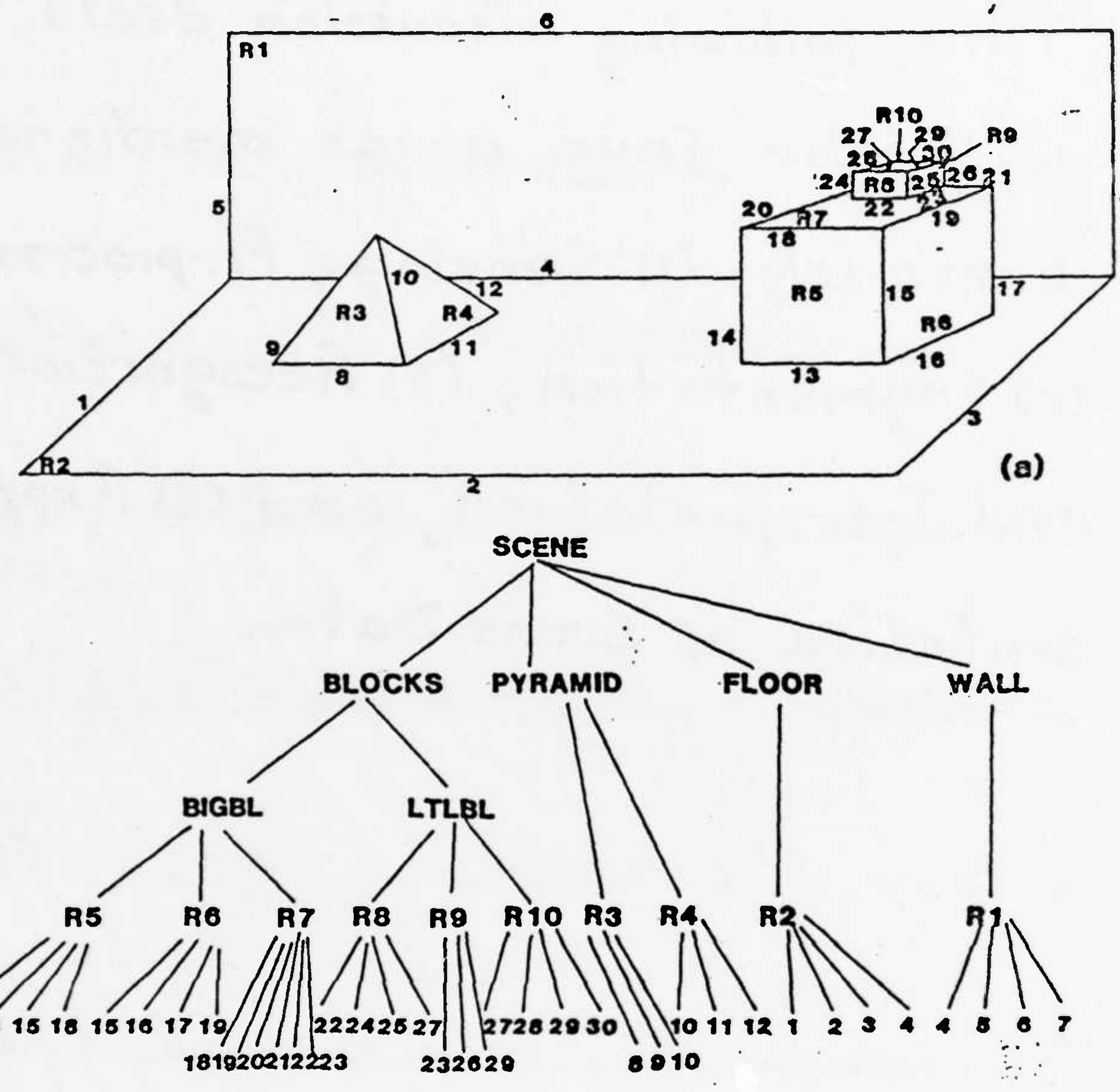
6

In terms of Robotics applications:



HIERARCHICAL MODEL OF SCENE ANALYSIS

The sequence of steps just discussed leads to a decomposition of a scene into simpler elements and the arrangement of these elements in the form of a hierarchical representation of the scene.



(a)

(b)

A simple scene and its hierarchical representation.

The following discussion deals with the four areas mentioned previously: (1) Sensing & Preprocessing, (2) Segmentation, (3) Recognition and Interpretation, and (4) Representation of Scene Data.

SENSING AND PREPROCESSING

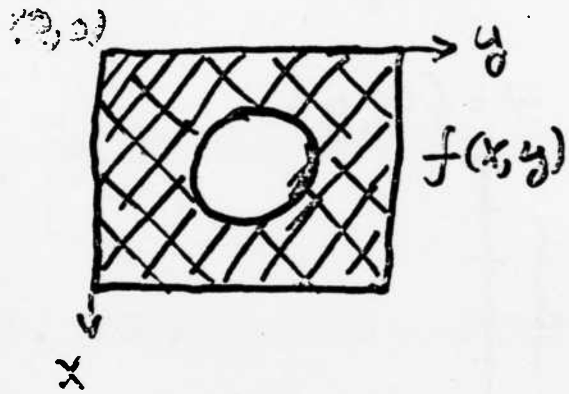
The sensing problem is one of converting a physical scene into a form suitable for computer processing. Typical sensors used in this area include:

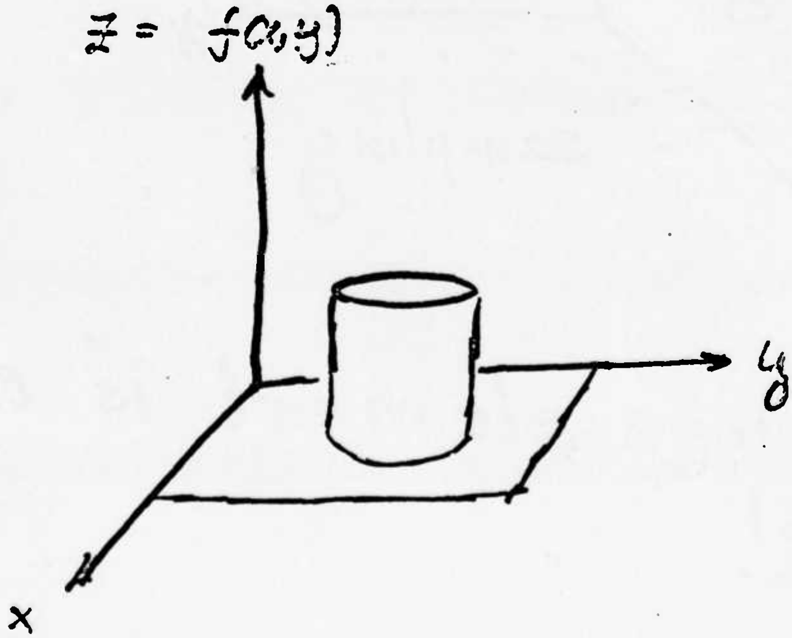
- Optical,
- X-ray,
- Infrared,
- Microwave, and
- Ultrasonic sensors.

The principal factor affecting the choice of a sensor is usually the application. When more than one type of sensor will do the job, additional constraints such as resolution, size, weight, and cost, play a deciding factor.

An important point: Scene analysis is a 3-D problem, but most of the present work is with planar (image) views. Reason: limitations in both sensors and algorithms.

Digital Image Representation



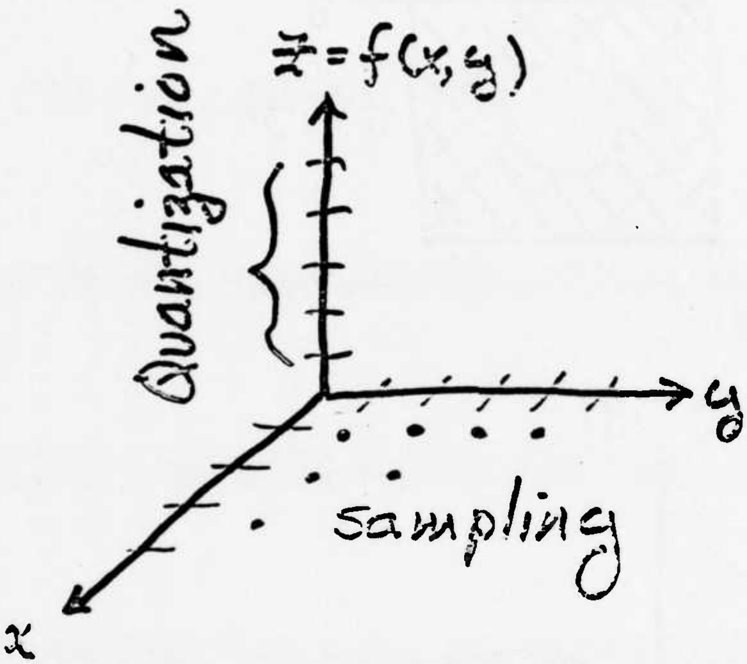


z = amplitude coordinate (Intensity)

(x,y) = spatial coordinates

Example: Lake scene

Digital image is obtained by <u>sampling</u> and <u>quantization</u>



Each image element is called a <u>pixel</u>

Uniform Sampling & Quantization

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \cdots & f(i,j) & \cdots \\ f(M-1,0) & & \cdots & f(M-1,N-1) \end{bmatrix}$$

i,jth pixel

$M \times N$ digital image array

Typically, $N, M = 2^n$, $G = 2^m$

For ~~square~~ $M \times N$ image array,

$\#\text{bits} = b = M \times N \times m$

Example: (a) 512 x 512 (b) 256 x 256 (c) 128 x 128 (d) 64 x 64 (e) 32 x 32 (f) 16 x 16. All are displayed in 256 levels.

Example: (a) 256 levels. (b) 128. (c) 64

(d) 32 (e) 16 (f) 8 (g) 4 (h) 2.

All are 512 x 512

False Contouring

Preprocessing in scene analysis deals with topics such as noise reduction, enhancement, and restoration.

SEGMENTATION

Segmentation is the process that breaks up a scene into its constituent parts or objects.

Most segmentation algorithms employ one of two basic principles:

(1) Discontinuity (e.g., edge detection)

(2) Similarity (e.g., region growing)

Algorithms for segmentation may be divided into three principal categories

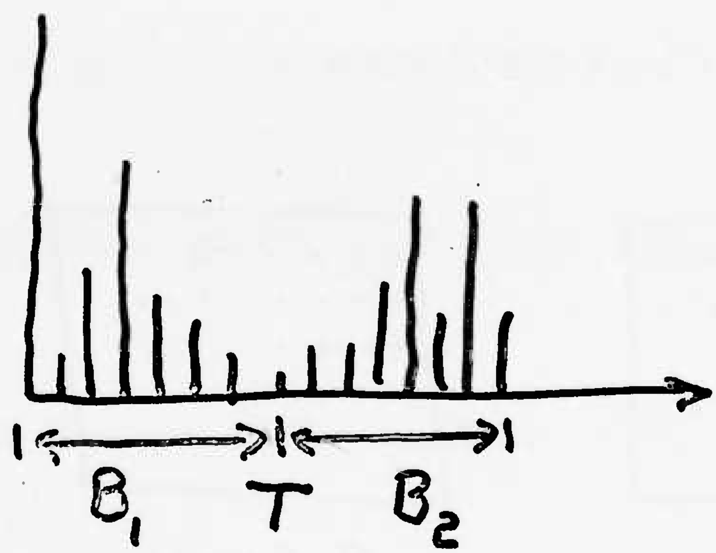
(1) Thresholding

(2) Edge detection

(3) Region growing

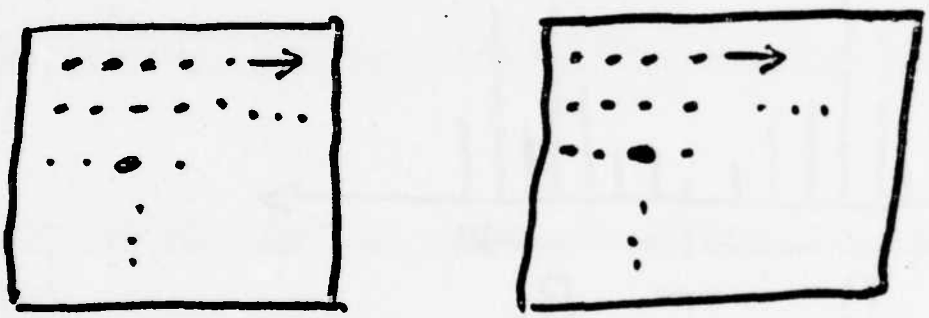
GRAY-LEVEL THRESHOLDING

1. A simple, two-pass procedure for partitioning the histogram of gray levels.



Approach: Select T so that B_1 corresponds to background intensity levels and B_2 to object intensity levels

Pass 1: Process rows

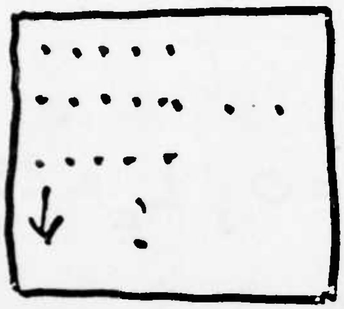


$f(x,y)$ $\qquad$ $g_1(x,y)$

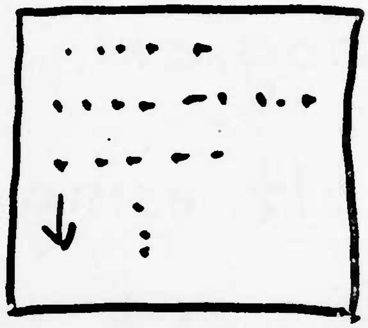
For any (x,y)

$$g_1(x,y) = \begin{cases} L_E & \text{if } f(x,y) \text{ and } f(x,y-1) \\ & \text{are not in same} \\ & \text{gray level band} \\ L_B & \text{otherwise} \end{cases}$$

Pass 2: Process Columns



$f(x, y)$



$g_2(x, y)$

$$g_2(x,y) = \begin{cases} L_E & \text{if } f(x,y) \text{ \& } f(x-1,y) \text{ are not in same band} \\ L_B & \text{otherwise} \end{cases}$$

Form Thresholded image

$$g(x,y) = \begin{cases} L_E & \text{if } g_1(x,y) \text{ or } g_2(x,y) = L_E \\ L_B & \text{otherwise} \end{cases}$$

Example:

(a) Original image.

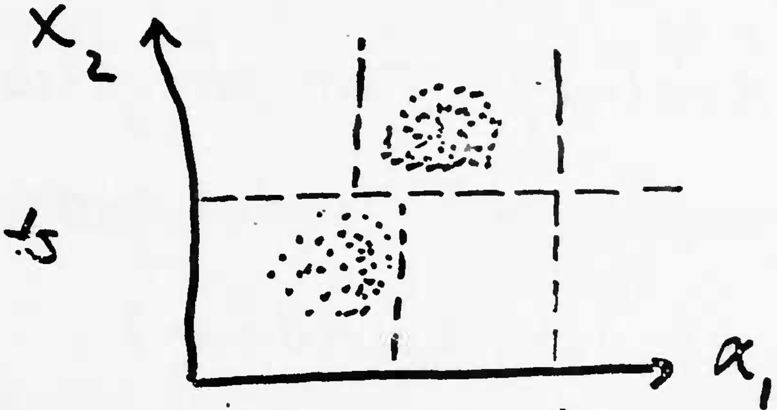
(b) Histogram.

(c) Result using $L_B = 0$, $L_E = 255$, $B_1 = B_2$ (i.e., $T = 128$).

(d) (c) superimposed on (a).

Use of several variables

Approach: find clusters of points and assign a different value to each cluster.



Histogram bins are 2-dimensional

A pixel closer to the ith cluster will receive value K_i.

Example: (a) Original (b) Points closer to "flesh" cluster. (c) Points close to a cluster that was near the red axis. The original was a color picture.

EDGE DETECTION

Based on the concept of discontinuity. The most-widely used approach is to employ templates (mask, window).

A template is an array designed to detect some invariant regional property

POINT TEMPLATE

-1	-1	-1
-1	8	-1
-1	-1	-1

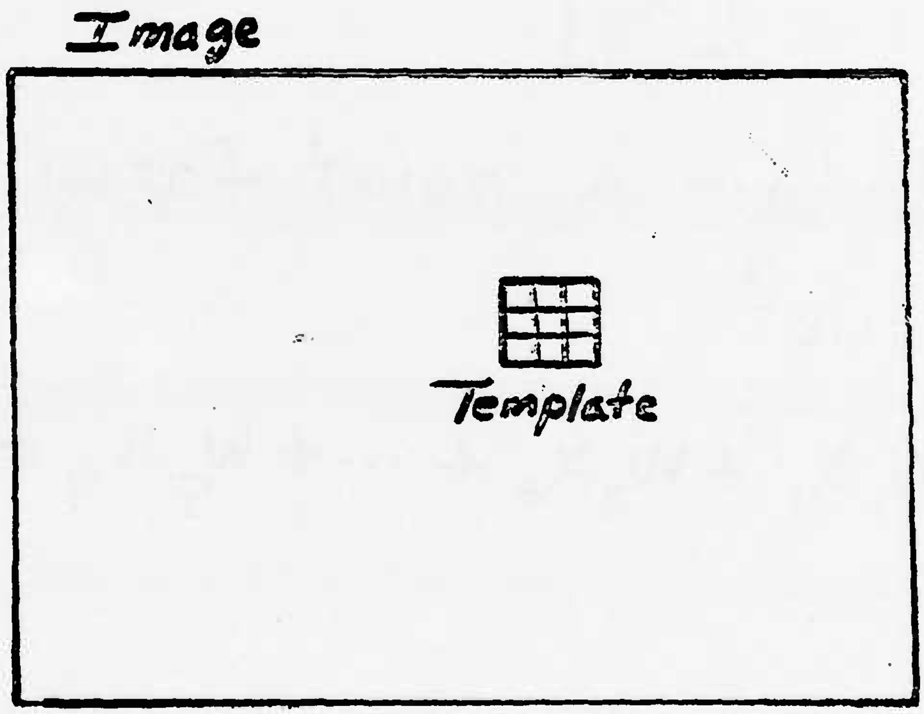
coefficients may be arranged in vector form

$$\underline{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_9 \end{bmatrix}$$

$w_5 = 8$

all others $= -1$

Running the template through an image:



The idea is to get a template response at every pixel location by centering the mask at each pixel in the image.

Image points in 3×3 region

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_9 \end{bmatrix}$$

To detect a point form the product

$$w_1 x_1 + w_2 x_2 + \cdots + w_9 x_9 = \underline{w}^T \underline{x}$$

If

$$\underline{w}^T \underline{x} > T$$

Then the pixel located in the center of the mask is an isolated point.

Line templates

HORIZONTAL

-1	-1	-1
2	2	2
-1	-1	-1

$\underline{w}_1$

+45°

-1	-1	2
-1	2	-1
2	-1	-1

$\underline{w}_2$

VERTICAL

-1	2	-1
-1	2	-1
-1	2	-1

$\underline{w}_3$

−45°

2	-1	-1
-1	2	-1
-1	-1	2

$\underline{w}_4$

The responses of these templates to a 3×3 image array are

$$\underline{w}_1^T \underline{x} \qquad \underline{w}_2^T \underline{x} \qquad \underline{w}_3^T \underline{x} \qquad \underline{w}_4^T \underline{x}$$

If, for example,

$$\underline{w}_1^T \underline{x} > \underline{w}_j^T \underline{x} \qquad j = 2, 3, 4$$

we say that $\underline{x}$ contains a horizontal line

Edge Templates (Gradient)

Consider the 3x3 image region

a	b	c
d	e	f
g	h	i

Gradient in x-direction

$$G_x = (g + 2h + i) - (a + 2b + c)$$

in y-direction

$$G_y = (c + 2f + i) - (a + 2d + g)$$

Gradient

$$G = \sqrt{G_x^2 + G_y^2}$$

or

$$G = |G_x| + |G_y|$$

Templates

$\underline{W}_1$

-1	-2	-1
0	0	0
1	2	1

$\underline{W}_2$

-1	0	1
-2	0	2
-1	0	1

(Also called Sobel operators)

REGION GROWING

A region is an area of an image whose pixels share a common set of properties.

Central to region-growing procedures are the selection of a <u>similarity criterion</u> and the formulation of a <u>growth termination rule</u>.

The selection of these factors is strongly influenced by the type of scene to be segmented.

The basic principles involved in region-growing will be illustrated by an application in FLIR image target segmentation. This is an interesting problem because different regions are usually not separated by abrupt transitions, thus making edge detection less than an ideal approach. Thresholding is also not well-suited to this problem because of typical variations within a region.

Similarity Criterion

Adjacency

Stopping Criteria

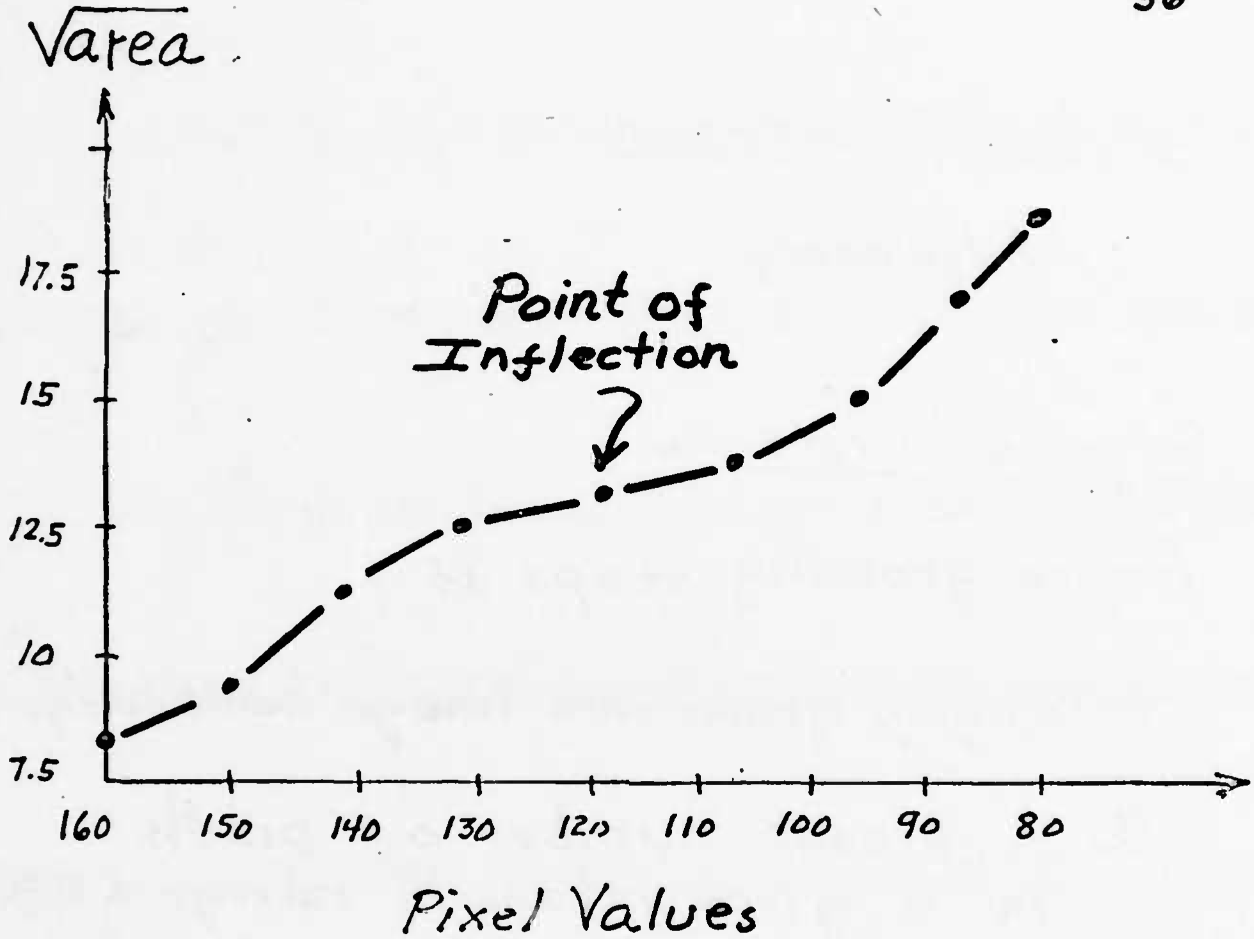
Region growing stops if

(a) Region grows into image boundary.

(b) A preset number of pixels in a given intensity range (e.g. top 25% of intensity scale) have been grown.

(c) "Optimum" target size has been reached

Optimum target size is defined as the size reached when growth rate is minimum. This concept is illustrated in the following figure.

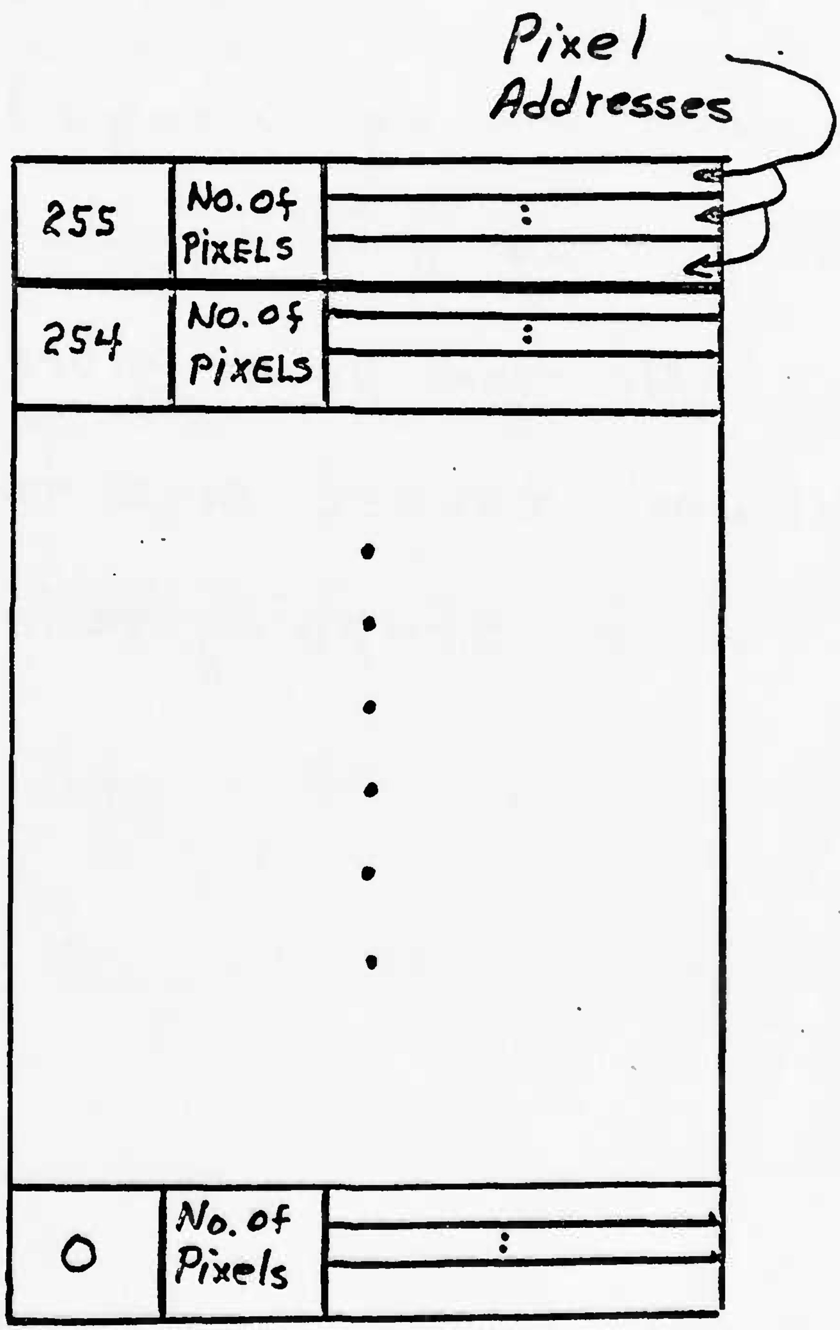


After an erratic start, the area increases rapidly at first, reaches a plateau at the edge of the target, and then increases rapidly again, as background pixels begin to contribute to the area.

Inflection point may be extracted by smoothing the curve (e.g. with a 5-point moving average), differentiating, and finding the minimum. The inflection point gives the "optimum" target size to be used in the stopping rule.

Basic structure of the algorithm

① Construct a histogram/pixel-address table



(2) BEGIN TARGET GROWING BY DECLARING THE FIRST PIXEL IN THE TABLE AN OBJECT AND ASSIGNING AN OBJECT NO. TO IT. [TABLE IS SCANNED TOP DOWN].

(3) SCAN THE TABLE IN ORDER OF DECREASING BRIGHTNESS VALUES. SUBSEQUENT PIXELS ARE ASSIGNED A NEW OBJECT NO. IF THEY ARE NOT ADJACENT TO ANY EXISTING OBJECT. OTHERWISE, THEY ARE ASSIGNED THE APPROPRIATE OBJECT NO.

(4) WHEN THE NUMBER OF SEPARATE OBJECTS REACHES A PRE-ESTABLISHED MAXIMUM, THE CREATION OF NEW OBJECTS IS INHIBITED. NEW PIXELS WHICH WOULD HAVE OTHERWISE BEEN NEW OBJECTS ARE IGNORED.

(5) IF A PIXEL IS ADJACENT TO MORE THAN ONE OBJECT, THE OBJECTS ARE MERGED TOGETHER, PROVIDED THEIR COMBINED AREA DOES NOT EXCEED THE MAXIMUM EXPECTED TARGET SIZE.

(6) STOP WHEN ANY OF THE STOPPING rules is SATISFIED FOR ALL REGIONS BEING GROWN.

Examples of FLIR target segmentation.

(a) Original FLIR image

(b) Blurred and noisy image

(c) Example of segmentation on "clean" and noisy images

(d) Segmentation results on original after it was smoothed with a 3x3 averaging window.

(e) Segmentation results on blurred, noisy image. (It was also smoothed).

RECOGNITION AND INTERPRETATION

Recognition is basically a labeling process. That is, the function of recognition algorithms is to identify each segmented object in a scene and to assign a label to it (e.g., road, vehicle, etc.)

Interpretation is closely linked with recognition – its function is to assign meaning to a set of recognized (labeled) scene elements.

The design of recognition procedures for scene analysis consists of two basic steps: (1) selection of a set of features or descriptors, and (2) selection of a classification strategy.

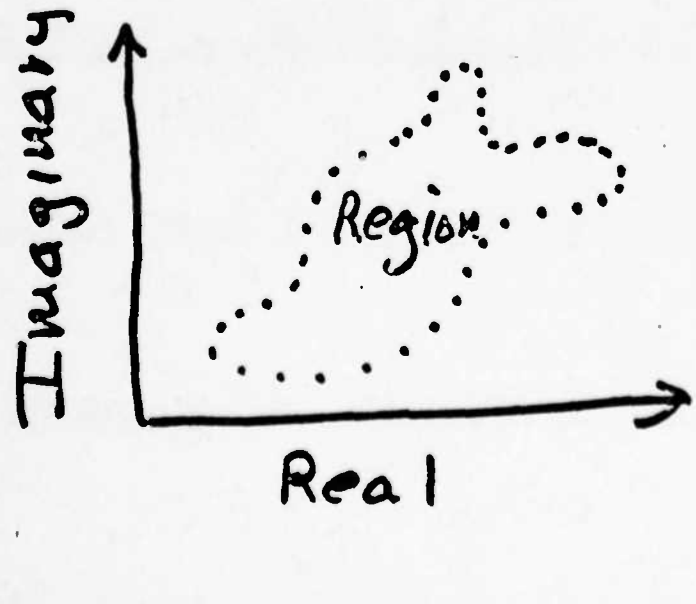
Will consider the problem of selecting descriptors in the following discussion, and then will consider techniques for recognition

Regional Descriptions

should be as independent as possible to variations in size, position, and rotation

Fourier Descriptors

Suppose points on boundary of a region have been determined. May be viewed as



Take FFT of contour points.

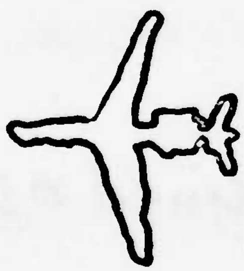
Changes in size are simply a scale factor in terms of the FT.

Rotation is just multiplication by $e^{j\theta}$, θ = angle of rotation.

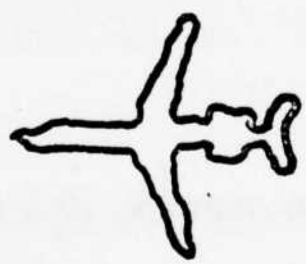
Magnitude of transform is independent of object location.

Example: 512-point boundaries but only 32 lowest freq. components retained.

210 Caravelle



BAC ONE-ELEVEN



NOTE THAT BASIC SHAPE WAS RETAINED

DC-10

DC-8

Moments

Given $f(x,y)$, the moment of order $p+q$ is defined as

$$m_{pq} = \iint_{-\infty}^{\infty} x^p y^q f(x,y)\,dx\,dy$$

The <u>central</u> moments are defined as

$$\mu_{pq} = \iint_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q f(x,y)\,dx\,dy$$

with

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

For discrete variables:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

The normalized central moments are given by

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}$$

$$\gamma = \frac{p+q}{2} + 1$$

A set of seven moments which are invariant to scale change, translation, and rotation is

$$\varphi_1 = \eta_{20} + \eta_{02}$$

$$\varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\varphi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\vdots$$

$$\varphi_7 = (3\eta_{12} - \eta_{30})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{12} - \eta_{03}) \cdot (\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

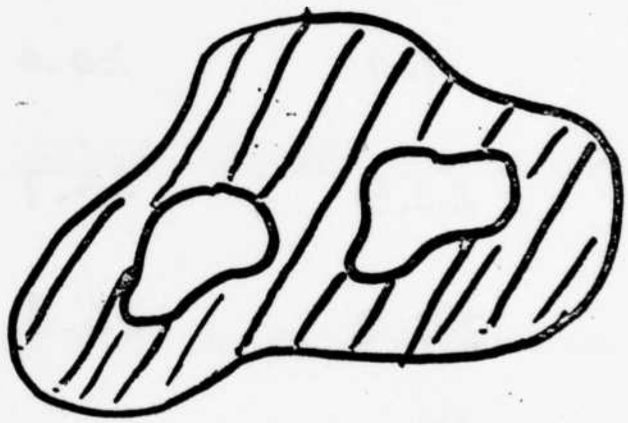
Example: (a) Original. (b) ½ size (c) Mirrored. (d) Rotated 2° and (e) 45°. Resulting moments:

Orig.	½ size	Mirror	2°	45°
6.2	6.2	6.9	6.3	6.3
17.2	17.0	20.0	17.3	16.8
22.7	23.5	26.7	22.8	19.7
22.9	24.2	26.9	23.1	20.4
45.7	48.3	53.7	46.1	40.5
31.8	32.9	37.1	32.0	29.3
45.6	48.3	53.6	46.0	40.5

Topological Descriptors

Topology: The study of properties of a figure which are unaffected by any deformation without tearing or joining.

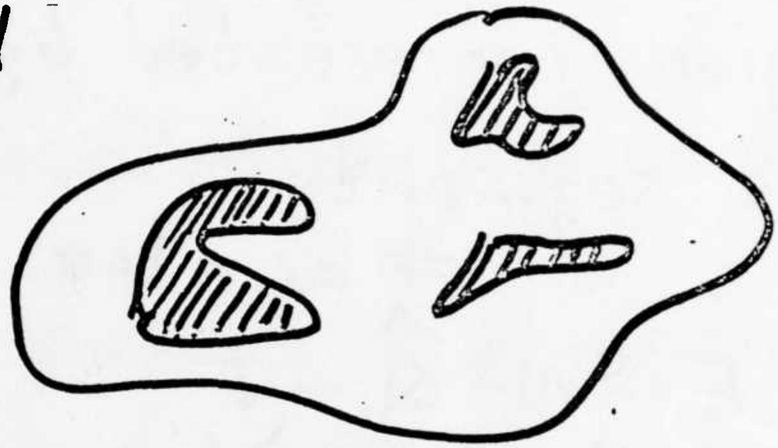
<u>Holes</u>



Not affected by scale change, rotation, location, stretching, etc.

<u>Connected</u> <u>component</u> – a region with the property that any two of its points can be joined by a connected curve lying entirely in the region.

Region with
3 connected
components

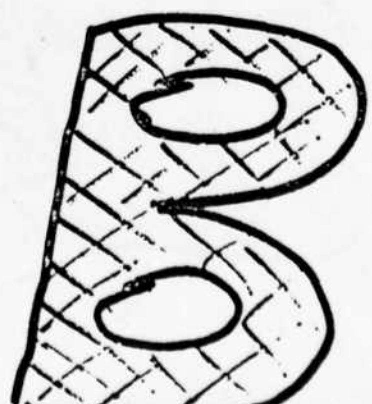


Euler Number

$$E = C - H$$

$E = 0$

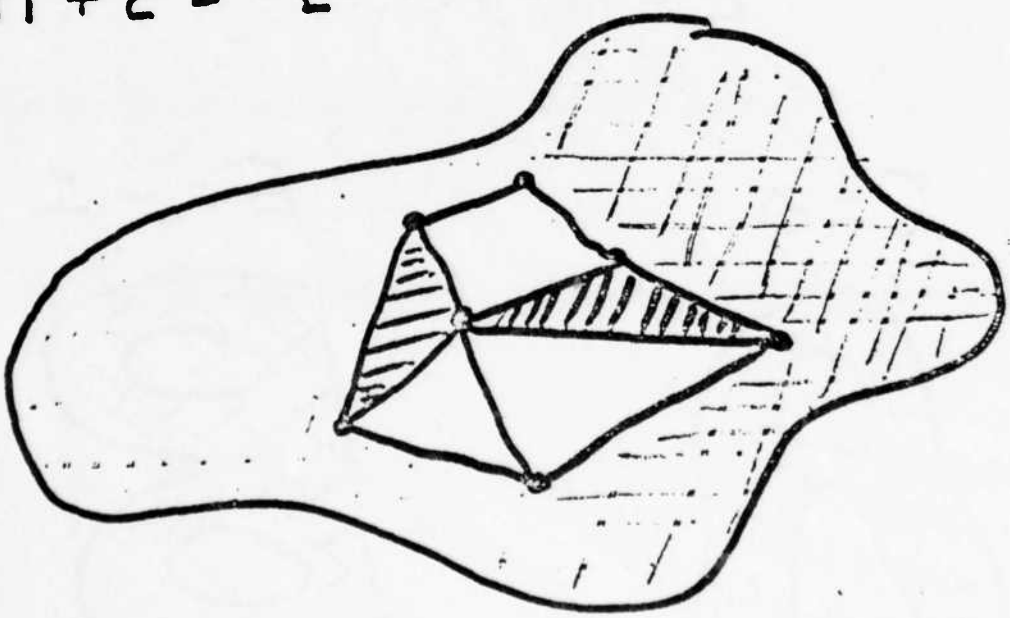
$E = -1$



For a polygonal network (i.e., region represented by straight line segments)

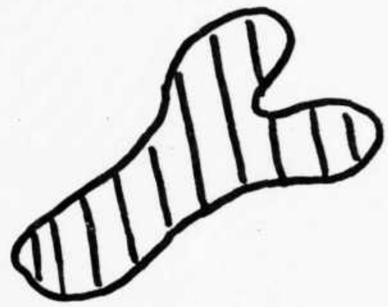
$$E = \underbrace{W}_{\#\text{ of vertices}} - \overset{\#\text{ of edges}}{\hat{Q}} + \underbrace{F}_{\#\text{ OF FACES}}$$

$$E = 7 - 11 + 2 = -2$$

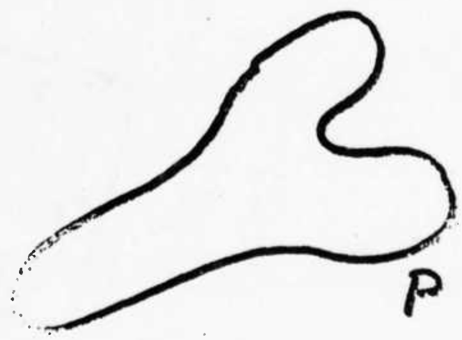


Other commonly-used descriptors of size and shape.

1. Area (A)

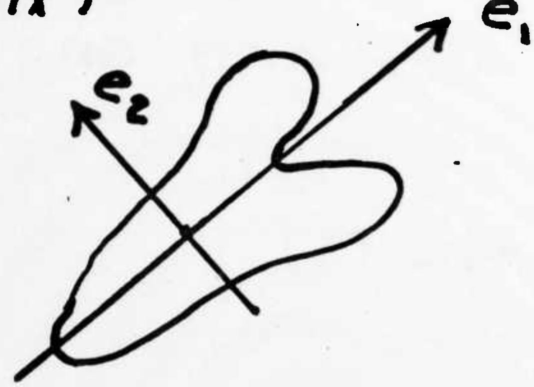


2. Perimeter Length (P)

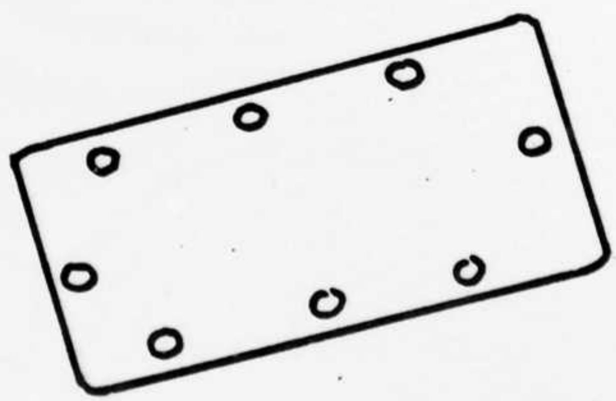


3. P^2/A - A measure of compactness.

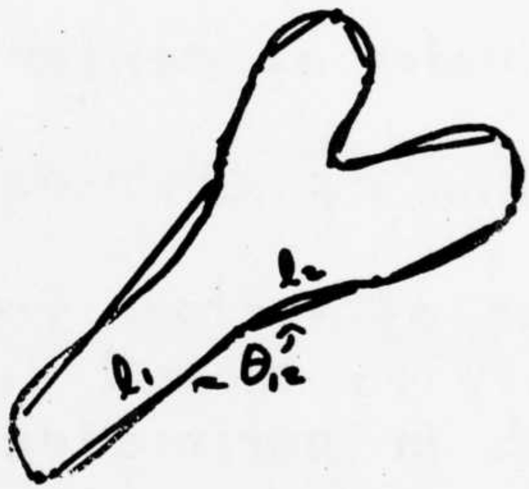
4. Axes of principal data spread (eigenvalues and eigenvectors of covariance matrix)



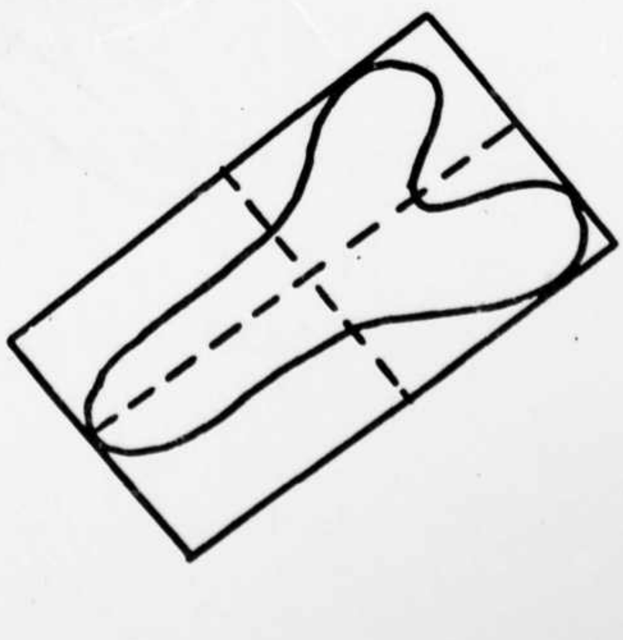
5. Number of holes, ratio of hole area to object area



6. Relationships between elements of a polygonal approximation.



7. Bounding box dimensions (Related to 1, 2, and 4).



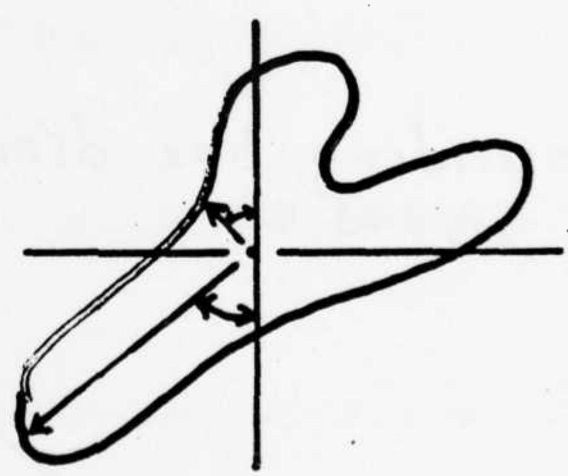
Some descriptors of position and orientation

1. Coordinates of center of gravity

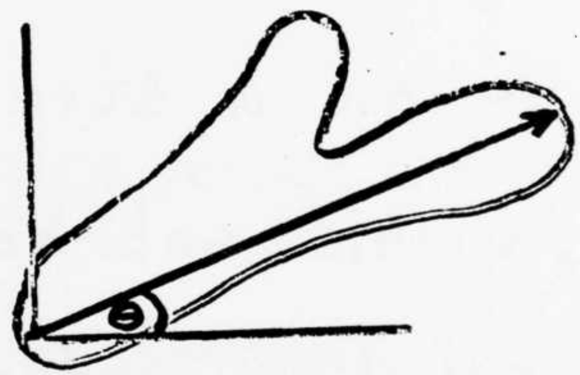
2. Coordinates of center of bounding box

3. Direction of eigen-axes

4. Angles of vectors from center of gravity to closest and furthest points in perimeter



5. Direction of vector connecting the two most distant points in the boundary of the object.



HOUGH TRANSFORM

Given n points in the x,y (image) plane, wish to find subsets that lie on a straight line.

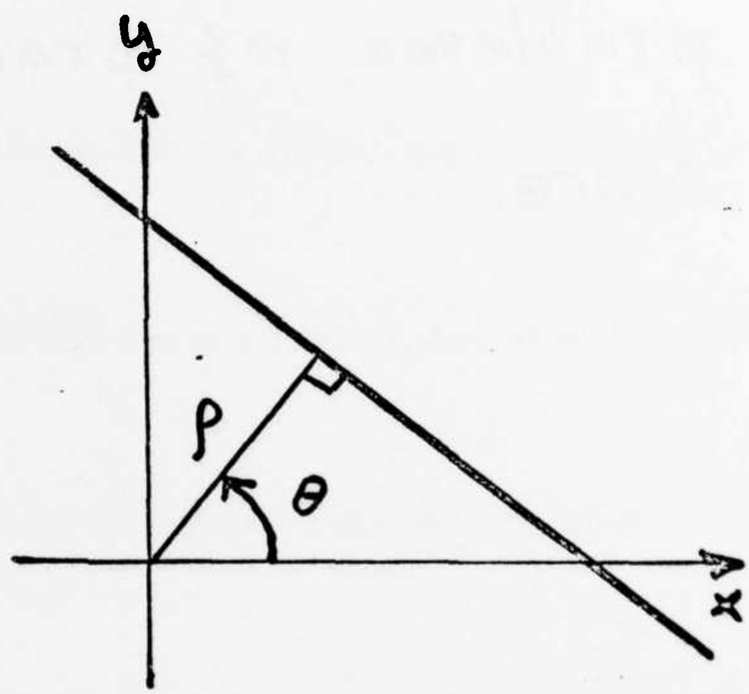
One possible solution is to first find all lines determined by every pair of points and then find all subsets that are "close" to particular lines.

The problem with this procedure is that it involves finding $n(n-1)/2 \sim n^2$ lines and then performing $(n)\, n(n-1)/_2 \sim n^3$ comparisons of every point to all the lines. This is computationally prohibitive for most problems of practical significance.

The approach proposed by Hough, and modified by Duda & Hart, gets around this problem by using the following normal representation of a line:

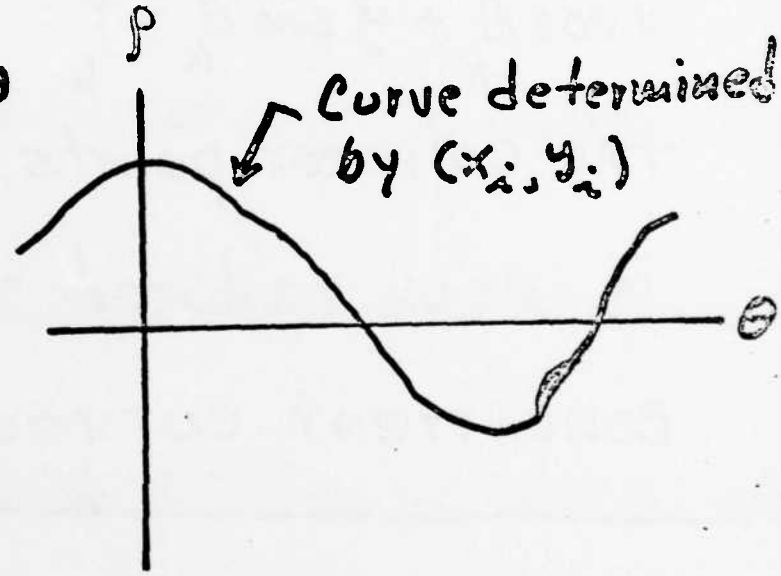
$$x\cos\theta + y\sin\theta = \rho$$

$$0 \le \theta < \pi$$



Then, each point (x_i, y_i) describes a sinusoidal curve in ρ-θ plane

$$\rho = x_i \cos\theta + y_i \sin\theta$$



The key is that the curves corresponding to colinear points in x-y plane, have a common point of intersection in θ-ρ plane

Thus, the point (θ_k, ρ_k) in the θ-ρ plane defines the line $x\cos\theta_k + y\sin\theta_k = \rho_k$ passing through the colinear points. The problem is now reduced to finding concurrent curves.

Similarly, by duality, a set of points $\{(\theta_1, \rho_1), (\theta_2, \rho_2), \ldots, (\theta_m, \rho_m)\}$ in the θ-ρ plane lying on the curve

$$\rho = x_0 \cos\theta + y_0 \sin\theta$$

determine all lines in x-y plane passing through the point (x_0, y_0).

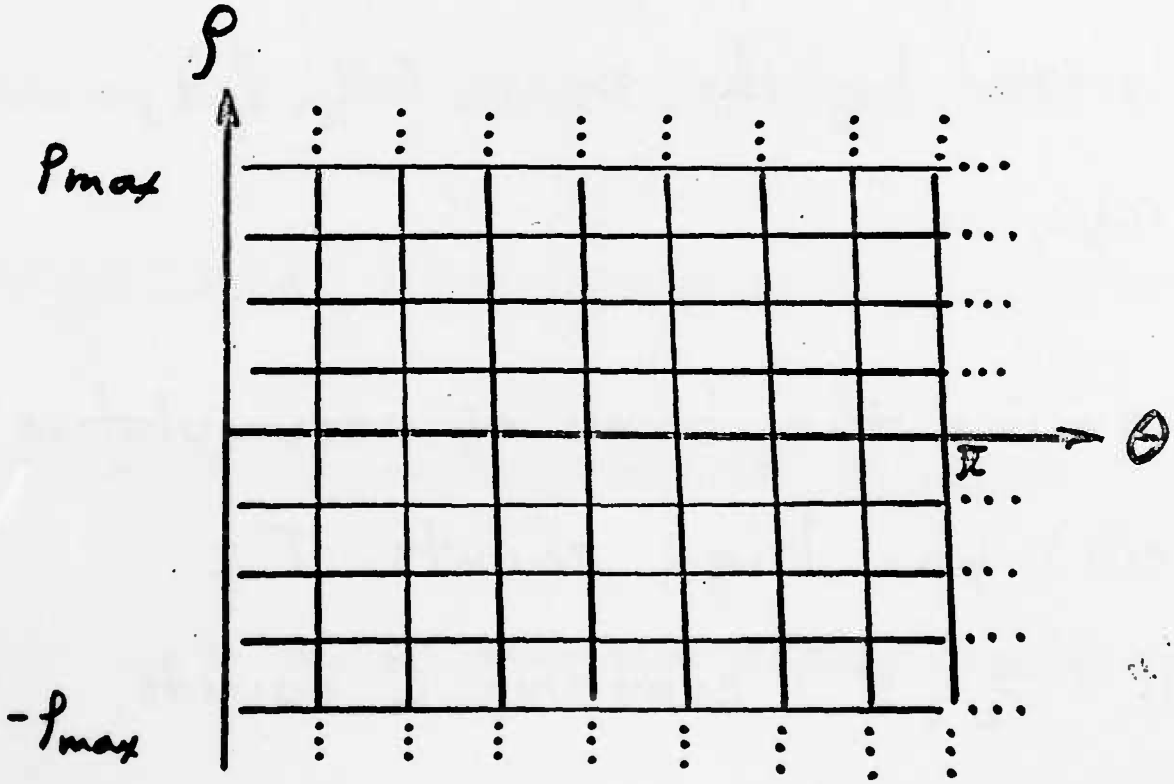
Summary of basic properties of the Hough transform.

1. A point in the image plane corresponds to a sinusoidal curve in the parameter plane.
2. A point in the parameter plane corresponds to a straight line in the image plane
3. Points on a straight line in the image plane correspond to curves through a common point in the parameter plane.

(4.) Points on the same curve in the parameter plane correspond to lines through the same point in the image plane.

IMPLEMENTATION

1. Quantize parameter plane into discrete cells (accumulators). Set all accumulator counts to zero.



2. For every point (x_i, y_i) compute

$$\rho_k = x_i \cos k\Delta\theta + y_i \sin k\Delta\theta$$

$$k = 0, 1, 2, \ldots$$

and increment by one the cells addressed by the pairs (θ_k, ρ_k), $k = 0, 1, 2, \ldots$

3. Examine the bank of accumulators (cells) for high counts. If cell (θ_u, ρ_v) contains C counts, then precisely C image points lie along the line with parameters θ_u, ρ_v (to within the error of the cells)

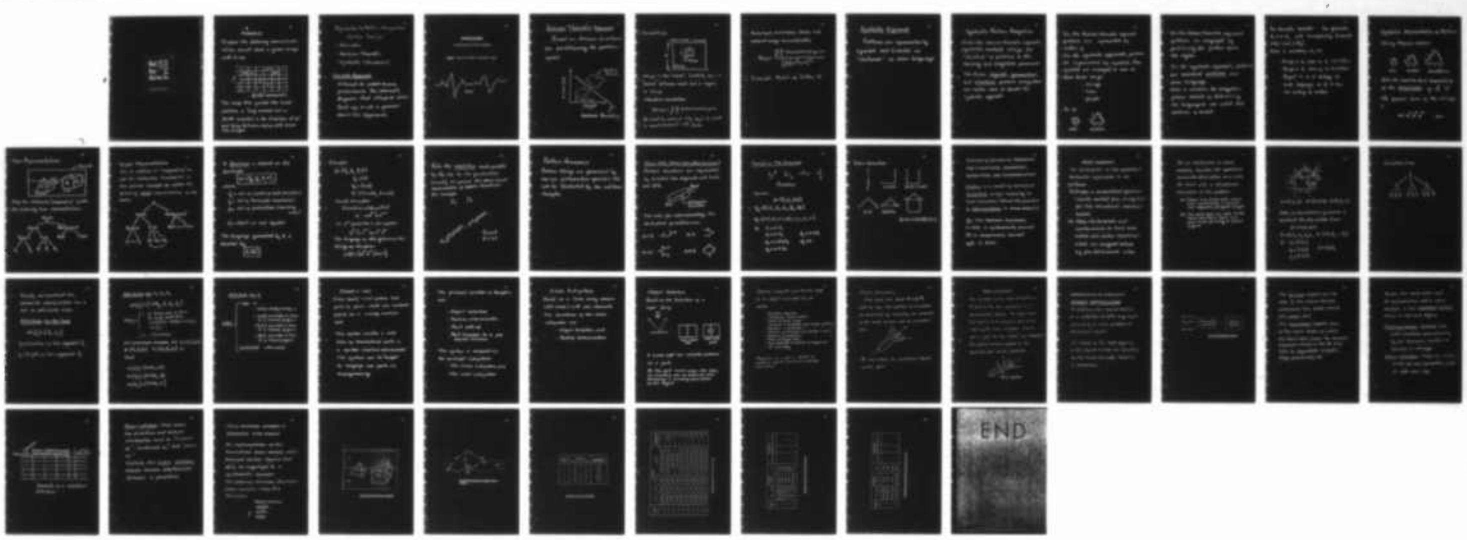
COMPUTATIONAL CONSIDERATIONS

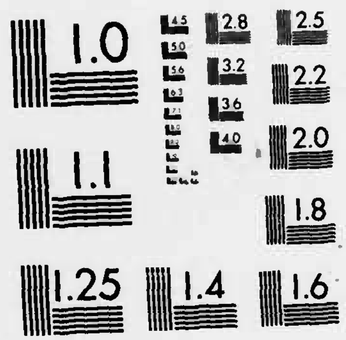
If the θ axis is divided into K_1 increments and the ρ axis into K_2 increments, then for every (x_i, y_i) we obtain K_1 values of ρ, corresponding to the K_1 possible values of θ.

Since there are n image points, this involves nK_1 computations, which is linear in n

AD-A134 852 TUTORIAL WORKSHOP ON ROBOTICS AND ROBOT CONTROL(U) ARMY TANK-AUTOMOTIVE COMMAND WARREN MI 26 OCT 82 4/4

UNCLASSIFIED F/G 14/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

EXAMPLE

Suppose the following accumulator values result from a given image with $n=150$.

ρ						
20 – 30				4		
10 – 20	199	5			30	
0 – 10			1			
-10 – 0			3			
-20 – -10				7		
-30 – -20						

θ (30° increments)

The image that yielded this table contains a "long" vertical set of points oriented in the direction 0°-30° and lying between 10 & 20 units from the origin

Approaches to Pattern Recognition System Design

- Heuristic
- Decision-Theoretic
- Syntactic (Structural)

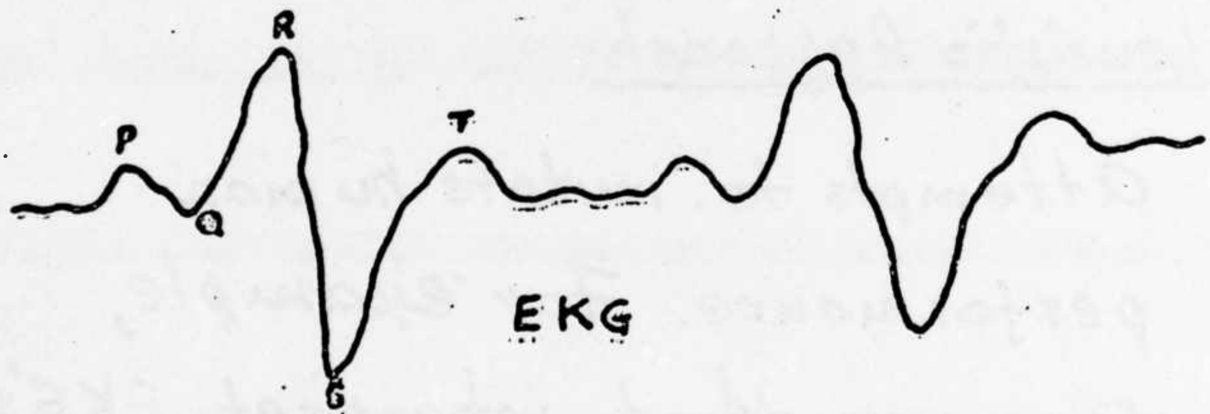
Heuristic Approach

Attempts to imitate human performance. For example, Programs that interpret EKG's. Can't say much in general about this approach.

THE HEURISTIC APPROACH

SYSTEM DESIGN BASED ON MAN'S EXPERIENCE

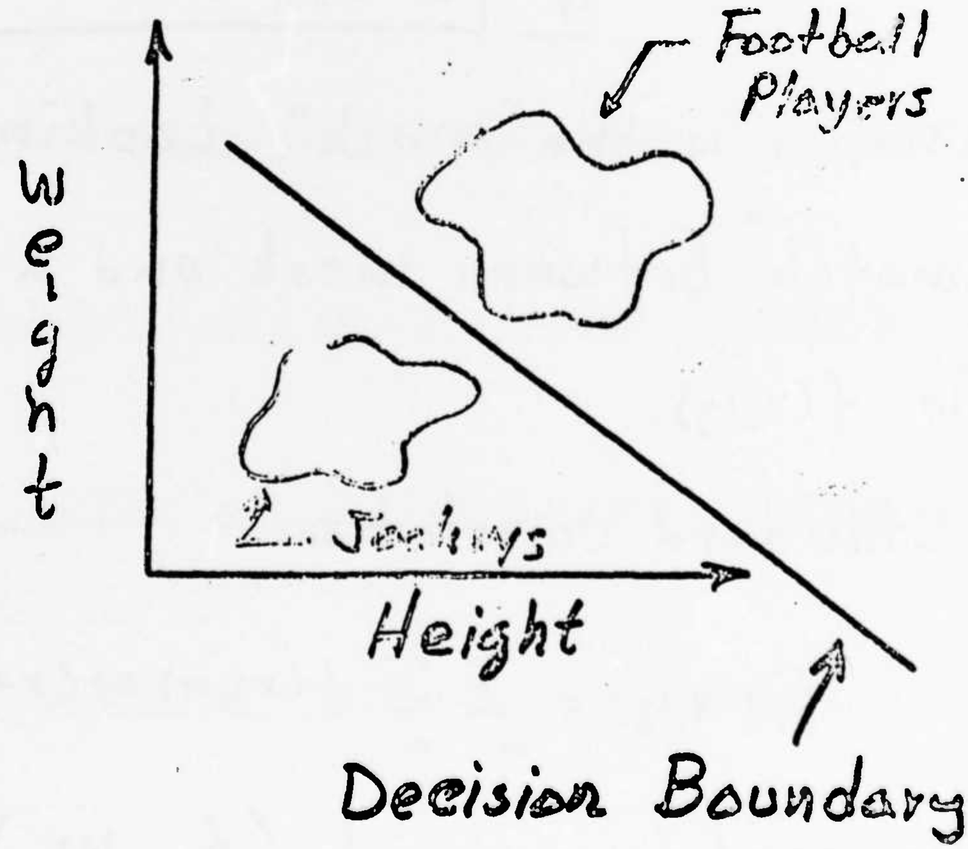
EXAMPLE: Design of an Automatic EKG Analysis System



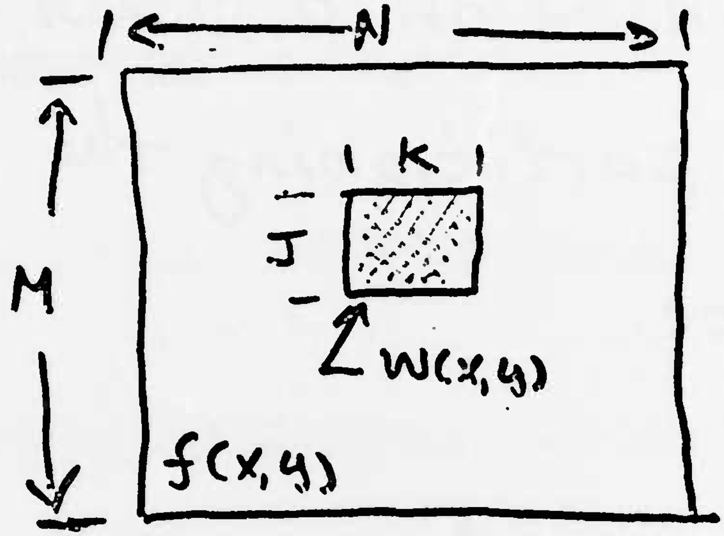
EKG

Decision Theoretic Approach

Based on decision functions for partitioning the pattern space.



Correlation



w(x,y) is the "mask". Looking for a "match" between mask and a region in f(x,y).

Standard correlation

$$R(x,y) = \sum_m \sum_n f(m,n)\, w(x+m, y+n)$$

No need to extend (f_e, w_e) if mask is small compared with f(x,y)

- Normalized correlation – takes into account image carracteristics

$$R(x,y) = \frac{\sum_m \sum_n f(m,n)\, w(x+m, y+m)}{\left[\sum_m \sum_n f^2(m,n)\right]^{1/2}}$$

- Example: Match of letter M

Syntactic Approach

Patterns are represented by symbols and treated as "sentences" of some language

Syntactic Pattern Recognition

Unlike the decision-theoretic approach, syntactic methods utilize the "structure" of patterns in the learning and recognition processes.

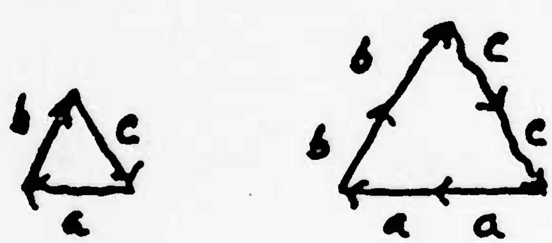
The terms *linguistic*, *grammatical*, and *structural* pattern recognition are often used to denote the syntactic approach.

In the decision-theoretic approach patterns are represented by vectors, $\underline{x}$.

In the syntactic approach, patterns are represented by symbols. These symbols are arranged in one of three basic ways:

- strings
- trees
- graphs

For ex:



abc aabbcc

In the decision-theoretic approach patterns are recognized by partitioning the pattern space into regions.

In the syntactic approach, patterns are considered sentences over some language.
Given a sentence, the recognition process consists of determining the language(s) over which that sentence is correct.

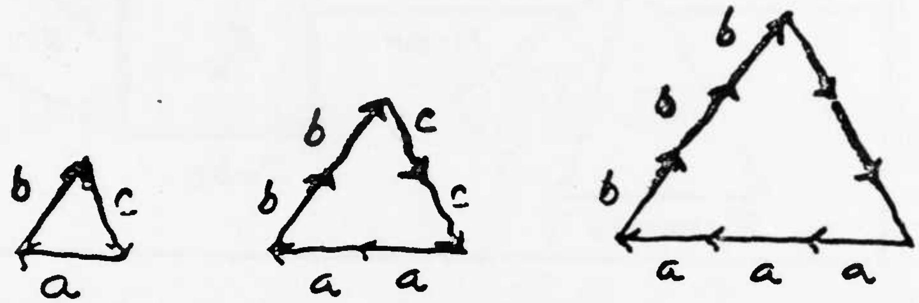
For example, consider two grammars G_1 and G_2 with corresponding languages $L(G_1)$ and $L(G_2)$.

Given a sentence, α, we

- Assign α to class ω_1 if $\alpha \in L(G_1)$
- Assign α to class ω_2 if $\alpha \in L(G_2)$
- Reject α if it belongs to both languages of if it does not belong to either.

Syntactic Representation of Patterns

String Representation



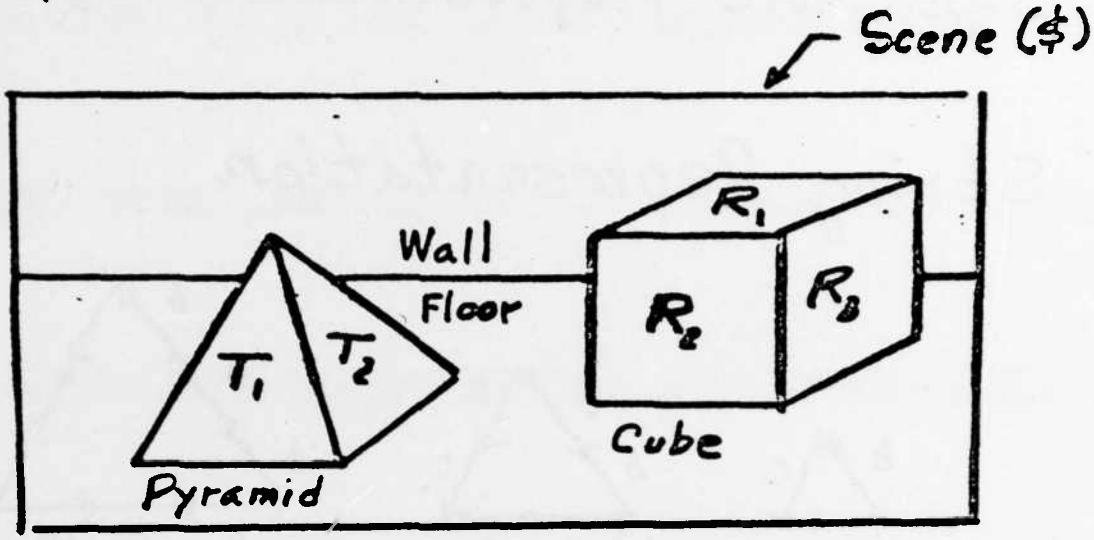
abc   aabbcc   aaabbbccc

Note the head-to-tail connectivity of the PRIMITIVES $\underset{a}{\longleftarrow}$ $\nearrow_{b}$ $\searrow^{c}$

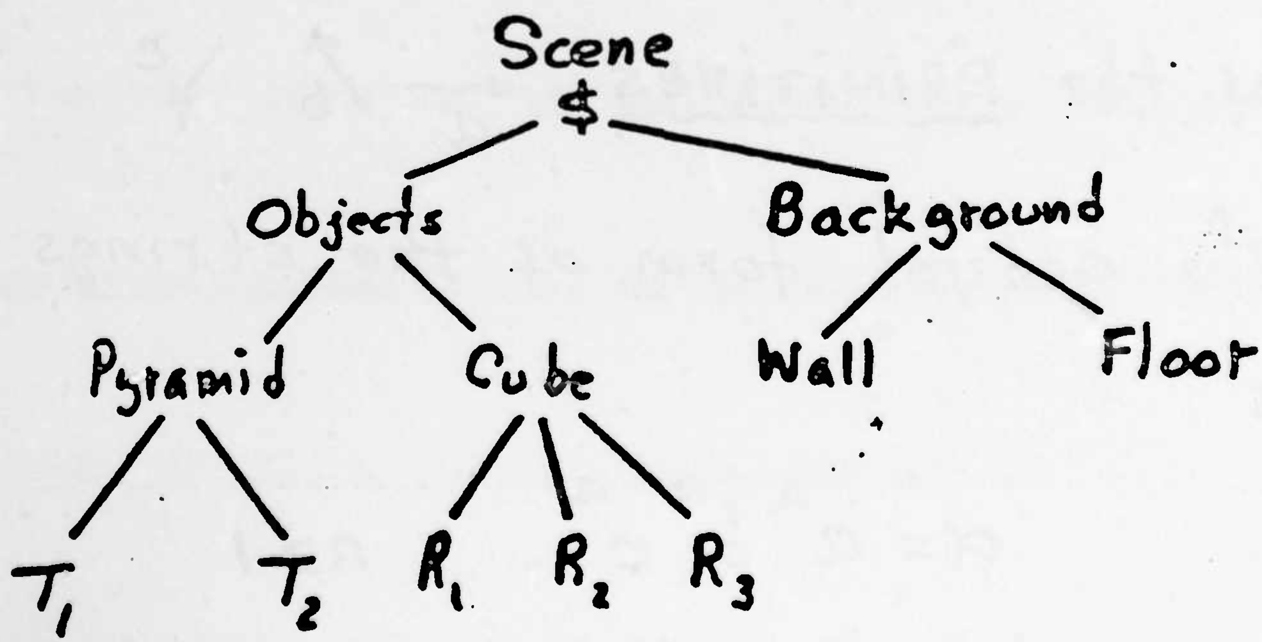
The general form of the strings is

$$\alpha = a^n b^n c^n \qquad n \geq 1$$

Tree Representation

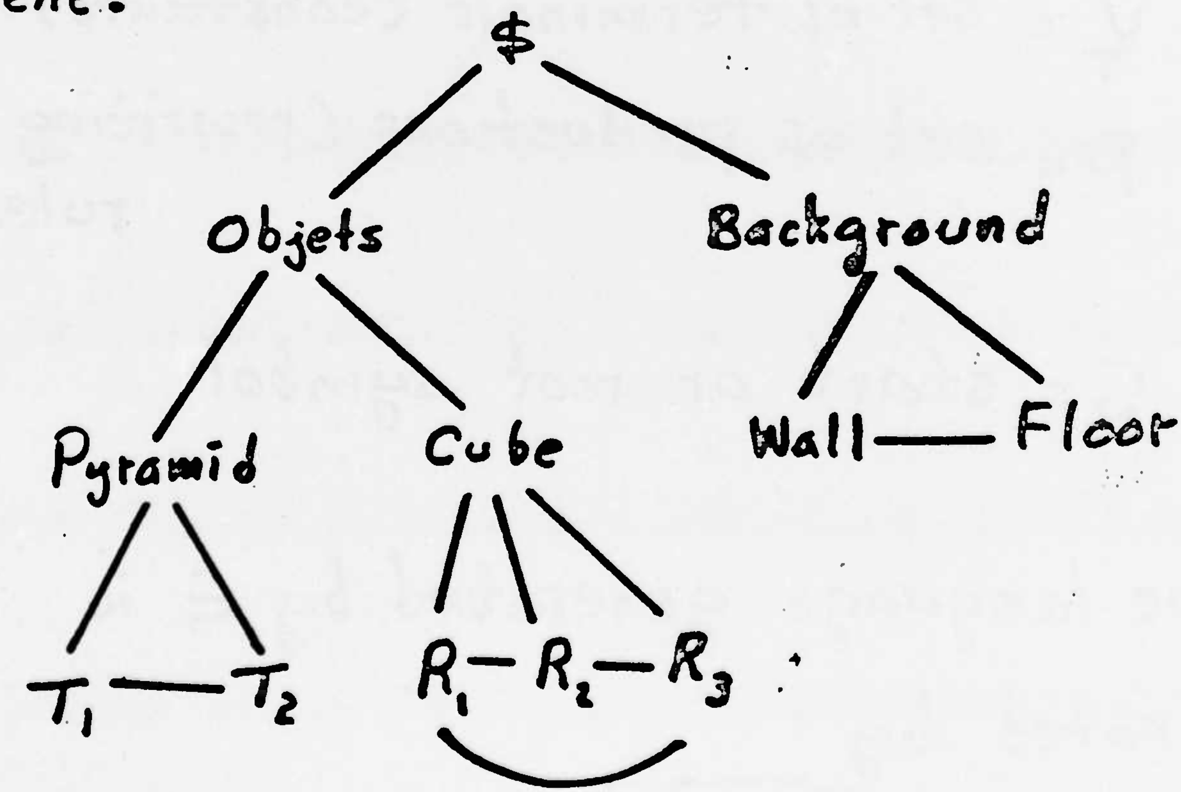


Using the relationship "composed of" yields the following tree representation:



Graph Representation

If, in addition to "composed of" we add the relationship "connected to" in the previous example, we obtain the following graph representation of the scene:



A *Grammar* is defined as the fourtuple:

$$\boxed{G = (V_N, V_T, P, S)}$$

where

V_N = set of nonterminals (variables)

V_T = set of terminals (constants)

P = set of productions (rewriting rules)

S = start or root symbol

The language generated by G is denoted by

$$\boxed{L(G)}$$

Example:

$G = (V_N, V_T, P, S)$

$$V_N = \{S\}$$

$$V_T = \{a, b\}$$

$$P = \{S \to aSb,\ S \to ab\}$$

Sample derivation:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow a^3 S b^3$$

$$\Rightarrow \cdots \Rightarrow a^{m-1} S b^{m-1}$$

If 2^{nd} production is now applied:

$$a^{m-1} S b^{m-1} \Rightarrow a^m b^m$$

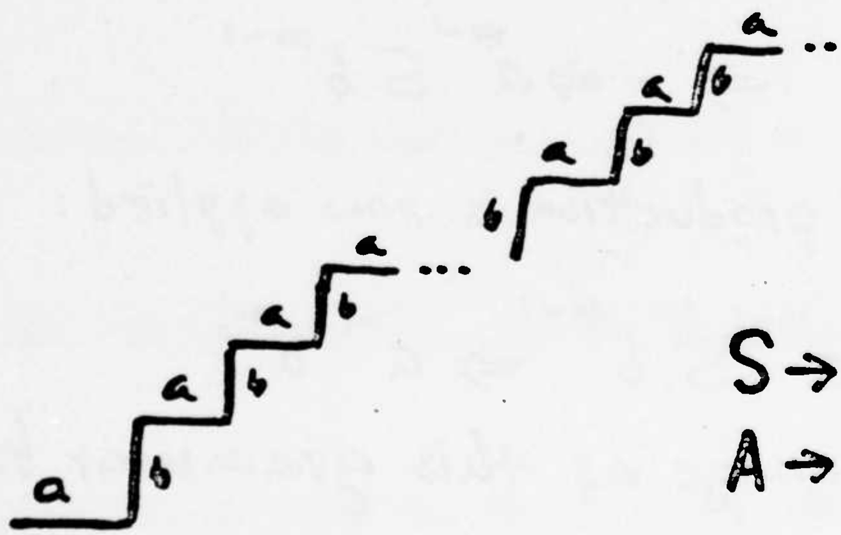
The language of this grammar has strings of the form:

$$L(G) = \{a^m b^m \,/\, m \geq 1\}$$

Note the repetition made possible by the use of the production $S \to aSb$. In general this allows simple representation of infinite structures.

For example:

$\xrightarrow{a}$ $\quad\quad$ $\uparrow b$



$S \to aA$

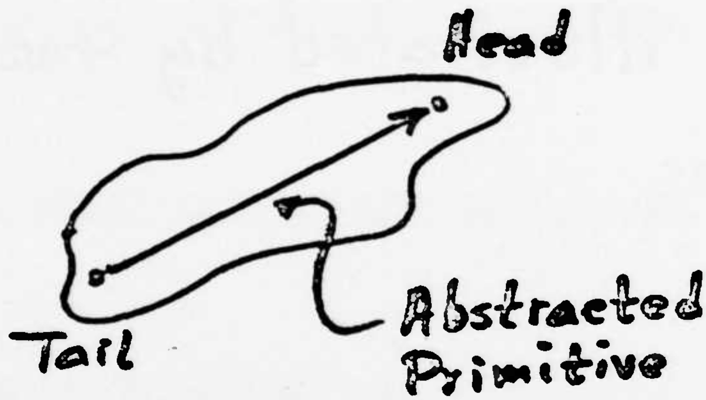
$A \to bS$

Pattern Grammars

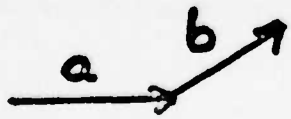
Pattern Strings are generated by the use juxtaposition operators. This will be illustrated by two well-known examples.

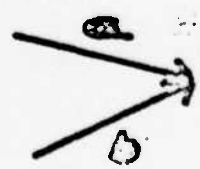
Shaw's PDL (Picture Description Language)

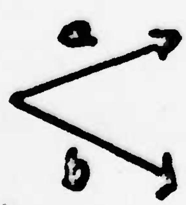
Pattern structures are represented by directed line segments with heads and tails.

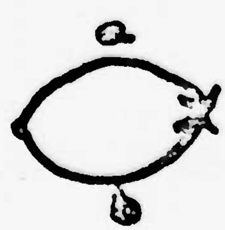


The rules for interconnecting the abstracted primitives are:

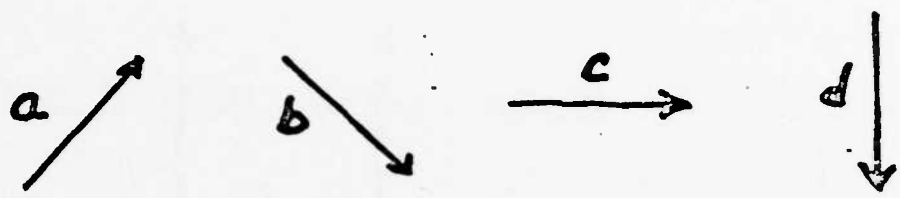
$a + b$ 

$a - b$ 

$a \times b$ 

$a * b$ 

Example of PDL Grammar



Primitives

Grammar:

$$G = (V_N, V_T, P, S)$$

$$V_N = \{S, A_1, A_2, A_3, A_4, A_5\}$$

$$V_T = \{a\nearrow, b\searrow, c\rightarrow, d\downarrow, +, \times, -, *, \sim\}$$

P:

$S \rightarrow d + A_1$

$A_1 \rightarrow c + A_2$

$A_2 \rightarrow \sim d * A_3$

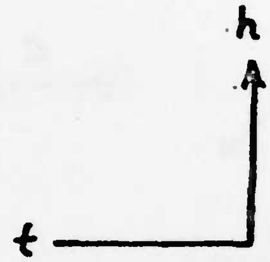
$A_3 \rightarrow a + A_4$

$A_4 \rightarrow b * A_5$

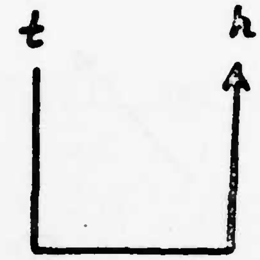
$A_5 \rightarrow c$

Pattern Generation

d

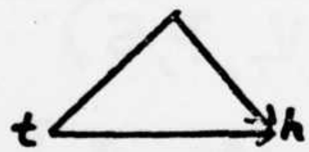


$c+(\sim d)$

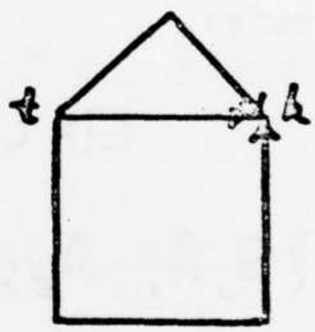


$d+(c+(\sim d))$

$a+b$



$(a+b)*c$



$(d+(c+(\sim d)))*((a+b)*c)$

SYNTACTIC/SEMANTIC TECHNIQUES FOR STRUCTURAL DESCRIPTION, RECOGNITION, AND INTERPRETATION

Syntax is a model of structure

Semantics assign meaning to that structure (Recall the function of interpretation in scene analysis)

Ex: The FORTRAN expression $C = A/B$ is syntactically correct. It is semantically correct only if $B \neq 0$.

BASIC CONCEPTS

The formulation of the syntactic/semantic approach is as follows:

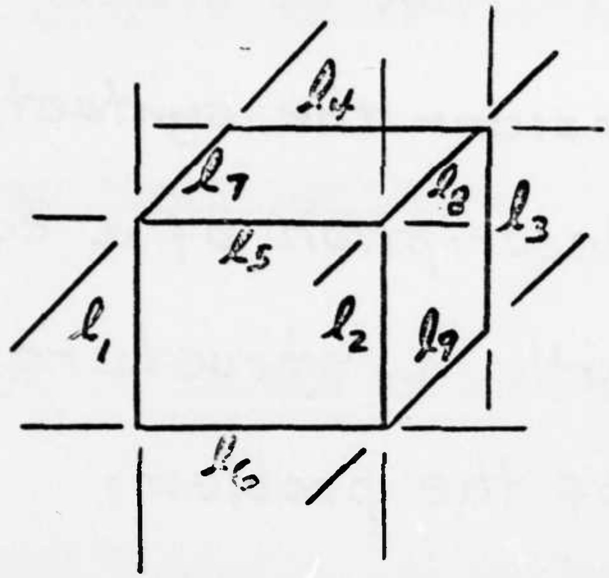
(1) Employ a conventional grammar (usually context free or regular) for the structural representation.

(2) Allow the terminals and nonterminals to have associated attributes (features) which are assigned values by pre-determined rules.

As an illustration in scene analysis, consider the syntactic/semantic description of a cube. We start with a structural description of the problem:

(a) There are three sets, A, B, C, each containing 3 parallel line segments of equal length.

(b) For each pair of sets (A,B), (A,C), and (B,C), there are four lines forming a closed figure.



$A = \{l_1, l_2, l_3\}$ $B = \{l_4, l_5, l_6\}$ $C = \{l_7, l_8, l_9\}$

Next, we formulate a grammar & construct the derivation tree

$$G = (N, \Sigma, P, S)$$

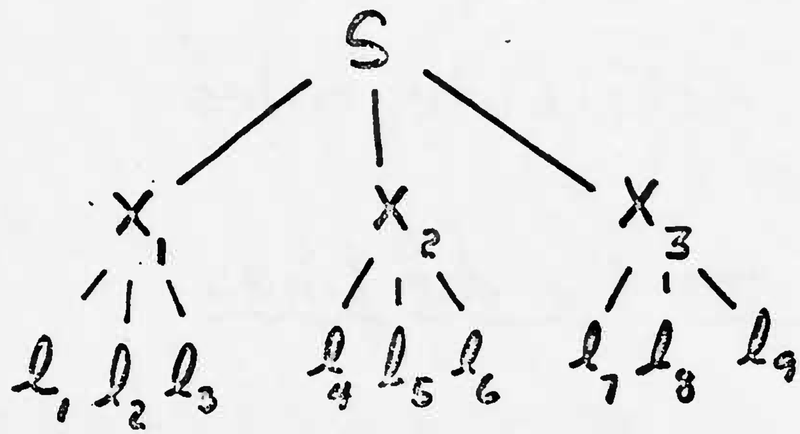
$N = \{S, X_1, X_2, X_3\}$, $\Sigma = \{l_1, l_2, \ldots, l_9\}$

P: $X_1 \to l_1\, l_2\, l_3$

$X_2 \to l_4\, l_5\, l_6$ $\quad S \to X_1 X_2 X_3$

$X_3 \to l_7\, l_8\, l_9$

Derivation tree:



Finally, we construct the semantic specification via a set of attribute rules:

Attributes for the lines

$$m[\ell_i] = [D_i, L_i]$$

D_i = direction of line segment ℓ_i

L_i = length of line segment ℓ_i

Attributes for X_1, X_2, X_3

$$m[X_j] = [FLAG_j, \ell_r, \ell_s, \ell_t]$$

$$FLAG_j = \begin{cases} 1 & \text{if there are 3 lines } \ell_r, \ell_s, \ell_t \text{ with the property } m(\ell_r) = m(\ell_s) = m(\ell_t) \\ 0 & \text{otherwise} \end{cases}$$

FOR NOTATIONAL PURPOSES, let $A = \{\ell_1, \ell_2, \ell_3\}$ $B = \{\ell_4, \ell_5, \ell_6\}$ $C = \{\ell_7, \ell_8, \ell_9\}$ so that

$$m[X_1] = [FLAG_1, A]$$

$$m[X_2] = [FLAG_2, B]$$

$$m[X_3] = [FLAG_3, C]$$

Attribute for S

$$m[S] = \begin{cases} \underline{\text{CUBE}} \text{ if} & \\ \quad \bullet\ FLAG_1 \cap FLAG_2 \cap FLAG_3 = 1 & \\ \quad \bullet\ (A,B) \text{ provides 4 lines in a closed figure} & \\ \quad \bullet\ (A,C) \text{ provides 4 lines in a closed figure} & \\ \quad \bullet\ (B,C) \text{ provides 4 lines in a closed figure} & \\ \underline{\text{undefined}} \text{ otherwise} & \end{cases}$$

CONSIGHT-I (GM)

Vision-based robot system that picks up parts which are randomly placed on a moving conveyor belt.

This system handles a wide class of manufactured parts in a non-ideal industrial environment. The system can be "taught" to recognize new parts via re-programming.

The principal functions of Consight-I are:

- Object detection
- Position determination
- Part pick-up
- Part transfer to a pre-defined location

The system is composed of two principal subsystems

- The vision subsystem, and
- The robot subsystem

Vision Subsystem

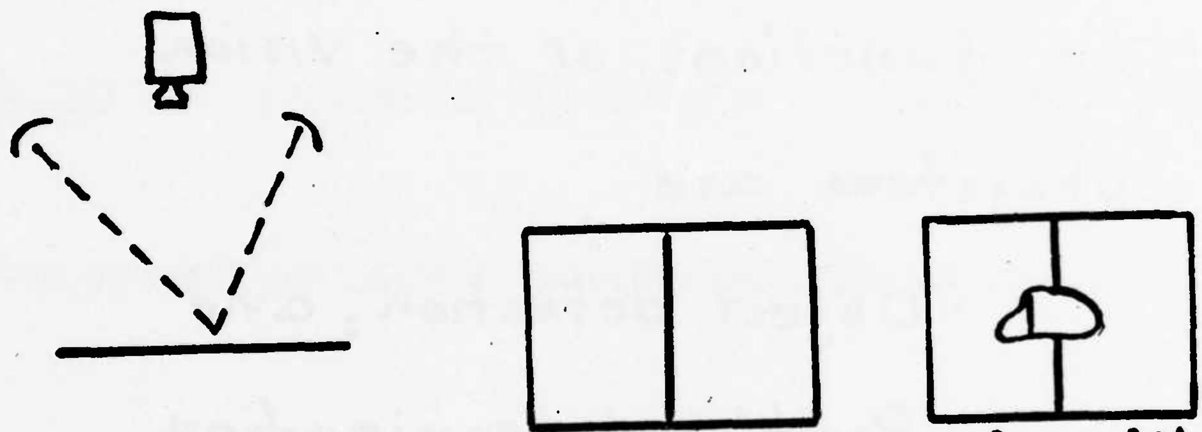
Based on a linear array camera (slit vision) with 256 elements.

The functions of the vision subsystem are

- Object detection, and
- Position determination

Object Detection

Based on the deflection of a light strip



VIEW WITHOUT PART   view with part

A broken light line indicates presence of a part.

As the part moves across the beam, its boundary can be followed, thus developing a boundary description of the object.

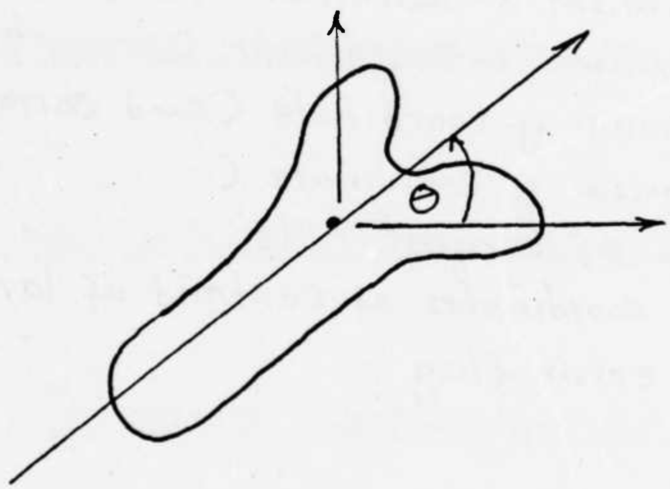
Features computed from the line image as an object moves past the slit include:

- Position reference
- Intensity (black & white)
- Count of pixels (area)
- Minimum x-coordinate (and corresp. y-coord.)
- Maximum x-coordinate (")
- Minimum y-coordinate (and corresp. x-coord.)
- Maximum y-coordinate (")
- Area of largest hole
- (x,y) coordinates of centroid of largest hole
- An error flag

Recognition of a part is achieved by matching against pre-stored prototype descriptions.

Position Determination

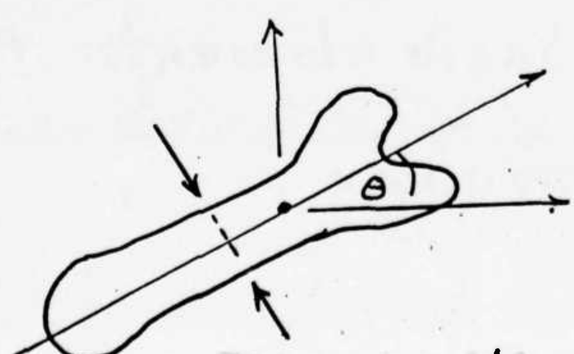
Once parts have passed through the field of view, their position and orientation are determined by computing the coordinates of the center of mass and an orientation axis



The axis chosen for orientation depends on the part.

ROBOT Subsystem

The function of the robot subsystem is to pick up the part and put it in a pre-determined location. The place where the part is to be pick-up is given to the robot by the vision subsystem. That is, once a part has been located and recognized, the pick-up procedure depends on the particular part and its orientation



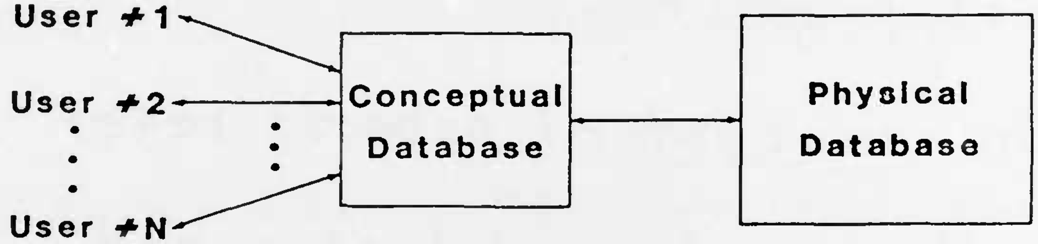
Pick-up position

ORGANIZATION OF SCENE DATA

DATABASE REPRESENTATIONS

A database (DB) may be defined as a collection of data organized according to some predefined structural model.

As shown in the next figure, a DB may be viewed as consisting of two basic elements: PHYSICAL & CONCEPTUAL.



General database model.

The <u>physical</u> aspects of a DB refer to the actual storage mechanism (e.g., fields, records, files, pages, etc.)

The <u>conceptual</u> aspects refer to the user's model of what the stored data mean. For example, different records in the DB may refer to segmented objects, shape parameters, etc.

By far, the most-often used DB representation used in scene analysis is the relational dabase, shown in the next figure.

Primitives column: Contains info. which identifies each primitive by, for example, number or location in storage.

Unary relations: These are attributes for each primitive, such as color and size.

PRIMITIVES \ RELATIONS	UNARY RELATIONS	BINARY RELATIONS	TERNARY RELATIONS	. . .	n-ARY RELATIONS
⋮					

Elements of a relational database

Binary relations: They involve two primitives and include relationships such as "in front of", "contained in," and "above of."

Similarly, the n-ary relations column involves relationships between n primitives

USE OF RELATIONAL DATABASES IN HIERARCHICAL SCENE ANALYSIS

An implementation of the hierarchical scene analysis model discussed earlier requires that data be organized in a systematic manner.
The following discussion illustrates these concepts using the hierarchy:

object clusters
|
objects
|
parts
|
edges

↑

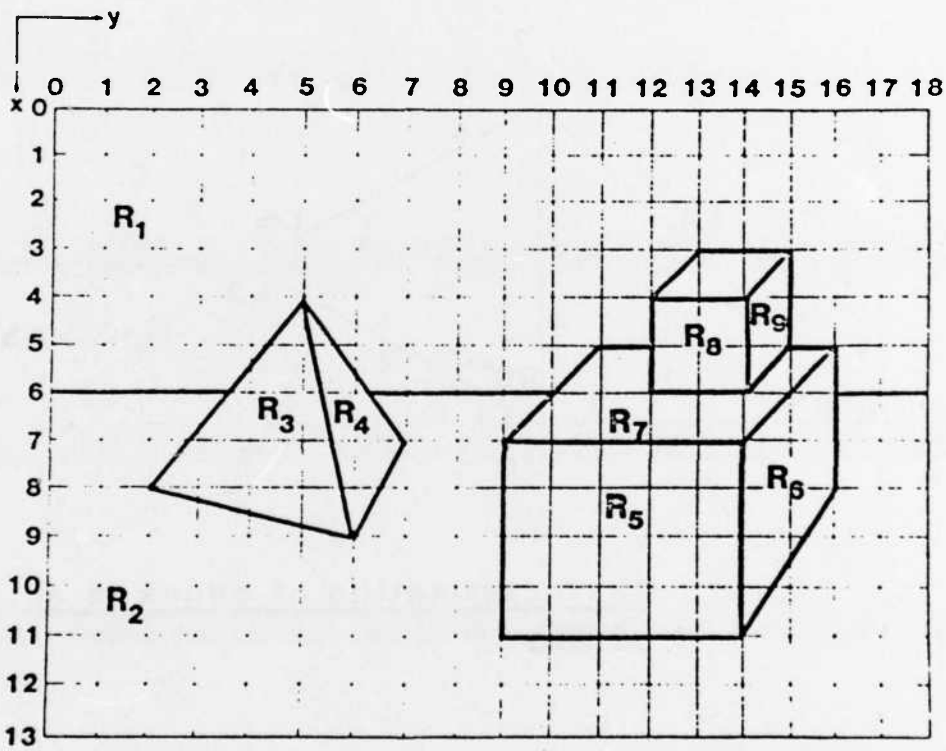
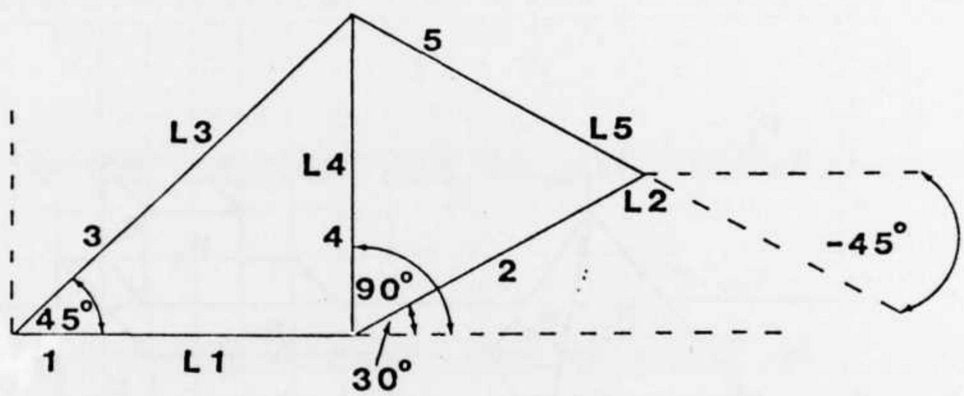


Image projection of a scene.



Characterization of edges in an object.

Element Number	Attributes: Direction	Attributes: Length	Relationship Connected to
1	0°	L1	2,3,4
2	30°	L2	1,4,5
3	45°	L3	1,4,5
4	90°	L4	1,2,3,5
5	-45°	L5	2,3,4

Relational table for the edges

Element No.	Attributes				Relationship
	Shape	Color	Texture	Boundary	Connected to
1	Rectangle	Black	Smooth	(0,0)-(0,18)-(6,18)-(6,0)	2
2	Rectangle	White	Smooth	(6,0)-(6,18)-(13,18)-(13,0)	1,3,4,5,6
3	Triangle	Blue	Smooth	(4,5)-(9,6)-(8,2)	2,4
4	Triangle	Blue	Smooth	(4,5)-(7,7)-(9,6)	2,3
5	Rectangle	Red	Smooth	(7,9)-(7,14)-(11,14)-(11,9)	2,6,7
6	Rectangle	Red	Smooth	(5,16)-(8,16)-(11,14)-(7,14)	2,5,7
7	Rectangle	Red	Rough	(5,11)-(5,16)-(7,14)-(7,9)	5,6,8,9
8	Rectangle	Red	Smooth	(4,12)-(4,14)-(6,14)-(6,12)	7,9,10
9	Rectangle	Red	Smooth	(3,15)-(5,15)-(6,14)-(4,14)	7,8,10
10	Rectangle	Red	Smooth	(3,13)-(3,15)-(4,14)-(4-12)	8,9

Relational table of parts for the scene

Element No.	Attributes				Relationships		
	Type	Color	Texture	Size	In front of	Right of	Resting on
1	Rectangle	Black	Smooth	Large			
2	Rectangle	White	Smooth	Large	1		
3	Pyramid	Blue	Smooth	Medium	1		2
4	Cube	Red	Smooth	Medium	1	3	2
5	Cube	Red	Smooth/Rough	Small	1	3	4

Relational table of objects for the scene

Element No.	Attributes				Relationships		
	Type	Color	Texture	Size	In front of	Right of	Resting on
1	Rectangle	Black	Smooth	Large			
2	Rectangle	White	Smooth	Large	1		
3	Pyramid	Blue	Smooth	Medium	1		2
4	Blocks	Red	Smooth/Rough	Medium	1	3	2

Relational table of object clusters for the scene

END

FILMED

12-83

DTIC

parts
edges