

# Contents

---

<b>I Introduction to Robotics</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Introduction . . . . .	3
1.2 A Little about the History . . . . .	3
1.3 The Technological Roots of Robotics . . . . .	5
1.4 First Approach . . . . .	6
1.4.1 Science Fiction Robots . . . . .	7
1.4.2 Toy Robots . . . . .	7
1.4.3 Real Robots . . . . .	8
1.5 Basic Terminology in Robotics . . . . .	15
1.5.1 Kinematic Chain . . . . .	15
1.5.2 Mechanism/Manipulator . . . . .	16
1.5.3 Degree of Freedom (DoF) . . . . .	17
1.5.4 End Point . . . . .	17
1.5.5 Cartesian Space vs. Joint Space . . . . .	18
1.5.6 Workspace . . . . .	18
1.5.7 Accuracy, Precision, Repeatability, Resolution . . . . .	19
1.6 Some Things to Think About . . . . .	21
<b>2 Introduction to Industrial Robotics</b>	<b>23</b>
2.1 Some Definitions of Industrial Robots . . . . .	23

2.1.1	Robotics Industry Association (RIA)'s Industrial Robot definition . . . . .	23
2.1.2	French Standards Association's Industrial Robot definition . . . . .	23
2.1.3	International Federation of Robotics (IFR)'s Industrial Robot definition . . . . .	24
2.2	Robot generations . . . . .	24
2.3	Industrial Applications . . . . .	25
2.3.1	Welding . . . . .	28
2.3.2	Surface Treatment and Painting/Spraying . . . . .	28
2.3.3	Cutting/Machining . . . . .	29
2.3.4	Assembly . . . . .	29
2.3.5	Disassembly . . . . .	30
2.3.6	Machine Tending/Handling . . . . .	30
2.3.7	Packaging/Palletizing . . . . .	30
<b>II</b>	<b>Industrial Robotics</b>	<b>31</b>
<b>3</b>	<b>Industrial Robot Programming</b>	<b>33</b>
3.1	Introduction . . . . .	33
3.2	Levels of Robot Programming . . . . .	35
3.3	Robot Programming Methods . . . . .	36
3.3.1	On-line Programming Methods . . . . .	37
3.3.2	Off-line Programming Methods . . . . .	40
3.3.3	Robot Programming Architecture . . . . .	41
3.4	Examples of Robot Programming Languages . . . . .	41
3.5	Robot programming Language Architecture . . . . .	42
3.6	Robot Program Development Process: the six steps . . . . .	43
3.6.1	Step 1: Task Analysis . . . . .	44
3.6.2	Step 2: Identify Placements for $P_e$ . . . . .	46
3.6.3	Step 3: Identify Subroutines . . . . .	48
3.6.4	Step 4: Robot Programming and Documentation . . . . .	50

<b>CONTENTS</b>	<b>vii</b>
3.6.5 Step 5: Program Testing . . . . .	50
3.6.6 Step 6: Test the program on the real robot. . . . .	50
3.7 Some Things to Think About . . . . .	51
<b>4 Robot Geometry</b>	<b>53</b>
4.1 Dimensions and Degrees of Freedom (DoF) . . . . .	53
4.2 Types of Joints . . . . .	54
4.3 The Geometry of Robot Manipulators . . . . .	58
4.3.1 Serial Geometries . . . . .	58
4.3.2 Parallel Geometries . . . . .	63
4.3.3 Wrists . . . . .	66
4.3.4 Some Things to Think About . . . . .	69
<b>5 Basic Components of Robots</b>	<b>71</b>
5.1 Introduction . . . . .	71
5.2 Links . . . . .	71
5.3 Actuators and Motors . . . . .	73
5.3.1 Pneumatic Actuators . . . . .	73
5.3.2 Hydraulic Actuators . . . . .	74
5.3.3 Electric Actuators . . . . .	74
5.4 Reduction mechanisms . . . . .	83
5.4.1 Harmonic-Drive . . . . .	84
5.4.2 Cyclo-Drive . . . . .	85
5.4.3 Other reduction and Transmission mechanisms . . . . .	86
5.5 Sensors . . . . .	86
5.5.1 Internal Sensors . . . . .	90
5.5.2 External Sensors . . . . .	99
5.6 End-Effectors and Terminal Devices . . . . .	101
5.7 Some Things to Think About . . . . .	102

<b>III Robot Mathematical Modelling and Control</b>	<b>103</b>
<b>6 Coordinate frames and homogeneous transformation</b>	<b>105</b>
6.1 Introduction . . . . .	105
6.2 The Representation of Position in 3D Space . . . . .	106
6.3 The Representation of Orientation in Space . . . . .	108
6.3.1 Rotation Matrices . . . . .	108
6.3.2 Properties of the Rotation Matrix . . . . .	110
6.3.3 Basic 3D rotation matrices . . . . .	111
6.3.4 The Composition of Basic Rotation Matrices . . . . .	113
6.3.5 Formulation Singularity in Euler Angles . . . . .	114
6.3.6 Rotation Axis- Rotation Angle Representation . . . . .	116
6.3.7 Quaternions or Euler Parameters . . . . .	116
6.4 Homogeneous Coordinates . . . . .	117
6.4.1 Position and Orientation in Space . . . . .	117
6.4.2 Homogeneous Coordinates . . . . .	118
6.4.3 Homogeneous Transformation . . . . .	120
6.4.4 Inverse Homogeneous Transformation Matrix . . . . .	122
6.4.5 Basic Translation Homogeneous Transformation Matrix . . . . .	122
6.4.6 Basic Rotation Homogeneous Transformation Matrix . . . . .	125
6.4.7 Composition of Translation and Rotation Matrices . . . . .	128
6.4.8 Composition of Homogeneous Transformation Matrices . . . . .	131
<b>7 Kinematics of Manipulators</b>	<b>137</b>
7.1 Introduction . . . . .	137
7.2 Forward and Inverse Kinematics . . . . .	137
7.3 Geometric Parametres of Robotic Mechanisms . . . . .	138
7.3.1 Link Parameters . . . . .	138
7.3.2 Joint Parameters . . . . .	140
7.4 Denavit and Hartenberg method . . . . .	141

7.5	Forward Kinematics Examples . . . . .	143
7.6	Inverse Kinematics . . . . .	156
7.6.1	Multiple Solutions . . . . .	157
7.6.2	Inverse Kinematics of a Serial Robot . . . . .	158
<b>8</b>	<b>The Jacobian Matrix</b>	<b>169</b>
8.1	Calculating the Jacobian Matrix . . . . .	170
8.2	Speed Propagation Method to Compute the Jacobian Matrix . . . . .	173
8.3	Static of Robotic Manipulators . . . . .	173
8.4	Force/Torque Propagation Method to Compute the Jacobian Matrix . . . .	175
8.5	Singularities and Singular Configurations . . . . .	176
<b>9</b>	<b>Path planning</b>	<b>179</b>
9.1	Introduction . . . . .	179
9.2	Types of Trajectories in Kinematic Control . . . . .	184
9.3	Trajectory interpolation in joint-space . . . . .	186
9.3.1	Linear interpolation . . . . .	187
9.3.2	Cubic interpolation (spline interpolation) . . . . .	188
9.3.3	Trapezoidal interpolator . . . . .	190
<b>10</b>	<b>Control of robot manipulators</b>	<b>193</b>
10.1	Introduction . . . . .	193
10.1.1	Analogy between Mechanical and Electrical Physis . . . . .	194
10.1.2	Types of robotic control loops . . . . .	195
10.1.3	Sources of Instability in Robot Control Loops . . . . .	197
10.2	Linear Position Control of a Single Joint . . . . .	198
10.2.1	Linear Dynamic Model of 1-DoF joint . . . . .	198
10.2.2	Proportional error control of a single joint . . . . .	200
10.2.3	The Steady State Error Problem . . . . .	204
10.2.4	The Overshoot Problem . . . . .	206

10.2.5	Speed Sensing and Control of a Joint System . . . . .	207
10.2.6	Feedforward and Feedback Compensations . . . . .	208
10.3	Computed Torque Algorithm . . . . .	211
10.3.1	Linear Dynamic Model of n-DoF Joint Manipulator . . . . .	211
10.3.2	Computed Torque Algorithm . . . . .	211
10.4	Adaptive Position Control of Joint Systems . . . . .	214
10.4.1	Gain Scheduling . . . . .	214
10.4.2	Model Reference Adaptive Control (MRAC) . . . . .	216
10.4.3	Computed Torque Adaptive Control . . . . .	217
10.5	Some Things to Think About . . . . .	217
<b>IV</b>	<b>Exercises</b>	<b>219</b>
i	<b>Kinematic structures and programming</b>	<b>221</b>
ii	<b>Transformations in space (homogeneous transformation)</b>	<b>233</b>
iii	<b>Forward Kinematics</b>	<b>245</b>
iv	<b>Inverse Kinematics</b>	<b>271</b>
v	<b>Jacobian</b>	<b>287</b>
<b>V</b>	<b>Appendix</b>	<b>297</b>
<b>A</b>	<b>Robot History</b>	<b>299</b>
A.1	Introduction . . . . .	299
A.2	Greek Mythology . . . . .	299
A.3	Middle ages . . . . .	301
A.4	XVIII-XIX centuries . . . . .	302
A.5	XX Century: Robot Timeline . . . . .	304

CONTENTS	xi
<b>B Mobile Robot Samples</b>	<b>309</b>
B.1 Microbots . . . . .	309
B.2 Legged Robots . . . . .	309
B.2.1 AIBO . . . . .	309
B.2.2 SDR . . . . .	309
B.2.3 ASIMO . . . . .	310
<b>C Robotic Wrists</b>	<b>313</b>
<b>D Glossary</b>	<b>315</b>



# List of Figures

---

1.1	A representation of a Rossum's Universal Robot. . . . .	4
1.2	First teleoperated device. It was designed for radioactive material handling. . . . .	6
1.3	Samples of Science Fiction Robots: HAL, R2D2, C3PO, T1000 (Terminator). . . . .	7
1.4	Samples of Toy Robots: Bender, R2D2. . . . .	7
1.5	Samples of Industrial Robots: an old Puma 7000; HAZBOT II, a robot for handling explosives. . . . .	8
1.6	Samples of Service Robots: a tank filling robot (Reis Robotics, Germany); a cleaning robot,(HACOmatic, Hakoberke, Germany), pet robot QRIO (Sony), pet robot Aibo (Sony). . . . .	9
1.7	Samples of biomedical robots for planning (left) and performing (right) surgery. . . . .	10
1.8	A picture of: Mars PATHFINDER and SOJOUNER on Mars; ROBICEN, a climbing robot (CEIT), KISMET (MIT). . . . .	11
1.9	Sample of Teleoperation device: SIMANTEL, CEIT. . . . .	11
1.10	Teleoperation architecture. . . . .	12
1.11	The bidirectionality of the haptic channel. . . . .	13
1.12	Example of a haptic system. . . . .	13
1.13	Two commercial haptic devices with tactile feedback through vibrations. . . . .	14
1.14	PHANTOM haptic devices. . . . .	15
1.15	Exoskeletons designed by PERCRO (left) and the University of Washington (right) for medical rehabilitation. . . . .	15
1.16	Open kinematic chain (left) vs. closed kinematic chain (right). . . . .	16
1.17	Serial mechanism/manipulator (left) vs. parallel mechanism/manipulator (right). . . . .	16

1.18 Hybrid mechanism/manipulator. . . . .	17
1.19 A <b>manipulator</b> and the human body <b>analogy</b> (left); schematic diagram of an arm (right). . . . .	19
1.20 Accuracy versus precision. . . . .	20
1.21 Accuracy, repeatability and resolution. . . . .	20
2.1 Robot applications. . . . .	26
2.2 Number of robots used in car manufacturing vs other sectors. . . . .	26
2.3 Distribution of the Spanish robots depending on economical sectors. . . . .	27
2.4 Different examples of welding processes. . . . .	28
2.5 Painting . . . . .	29
2.6 Cutting. . . . .	29
2.7 Assembly . . . . .	29
2.8 Handling. . . . .	30
2.9 Palletizing. . . . .	30
3.1 Control loop of a robot. . . . .	34
3.2 Kawasaki's teach pendant (left); motoman's teach pendant (right). . . . .	38
3.3 Programming by passive guiding vs. programming by teach-pendant. . . . .	39
3.4 Relationship between On-line and Off-line programming. . . . .	41
3.5 Common symbols used in flowcharts. . . . .	43
3.6 A simple assembly task. . . . .	44
3.7 Flowchart sample. . . . .	45
3.8 Definition of a simple assembly task. . . . .	47
3.9 Get-Put subroutine, written using pseudocode. . . . .	48
3.10 Flowchart of the Get-Put subroutine. . . . .	49
4.1 Reasons to use extra DoFs or redundant robots. . . . .	55
4.2 Basic types of DoFs. . . . .	56
4.3 Types of joints . . . . .	57
4.4 Cartesian robot . . . . .	59

4.5	Cylindrical robot . . . . .	60
4.6	Spherical robot. . . . .	61
4.7	Anthropomorphic robot. . . . .	61
4.8	Scara robot . . . . .	62
4.9	2D parallel robot with only 1-DoF . . . . .	63
4.10	Samples of 2-DoF parallel robots: two prismatic. . . . .	63
4.11	Samples of 3-DoF parallel robots: Star Robot (left); Mianouwski's . . . . .	64
4.12	The Stewart Platform. . . . .	64
4.13	Hybrid robots: Hybrid parallel robot (left), Hybrid serial robot (right) . . . . .	65
4.14	The drilling task needs the placement of robot's end-effector and orientation as well. . . . .	67
4.15	Wrist sample: gripper with two fingers. . . . .	67
4.16	Conventional Aeronautics Terminology . . . . .	68
4.17	RPY-wrist. . . . .	68
4.18	RP-wrist. . . . .	68
4.19	RPR-wrist. . . . .	69
5.1	Mechanical structure of the PA10 . . . . .	72
5.2	Compressed-air preparation. . . . .	73
5.3	Types of pneumatic actuators. . . . .	74
5.4	A model of a DC motor. . . . .	75
5.5	A PWM signal. . . . .	77
5.6	A PFM signal. . . . .	78
5.7	An H-bridge. . . . .	78
5.8	Full-stepper motor. . . . .	80
5.9	Half-stepper motor. . . . .	80
5.10	Stepper motor: rotor types. . . . .	81
5.11	Bipolar windings vs. Unipolar windings. . . . .	81
5.12	A RC servomotor: internal parts (left), external appearance (right) . . . . .	82
5.13	Commanding a RC servomotor. . . . .	82

5.14	Robotic redutor mechanism simulation. . . . .	84
5.15	Components of a Harmonic-Drive. . . . .	85
5.16	Harmonic-drive working sequence. . . . .	85
5.17	Schematic of a Cyclo-drive. . . . .	86
5.18	Cyclo-drive animation. . . . .	86
5.19	Sensitivity and range for a given sensor . . . . .	88
5.20	Sensor resolution . . . . .	89
5.21	Accuracy, offset and linearity for a given sensor . . . . .	89
5.22	Sensor hysteresis . . . . .	91
5.23	Dead zone and saturation for a given sensor . . . . .	91
5.24	Sensor response time . . . . .	91
5.25	Sensor bandwidth . . . . .	92
5.26	Resolver and Synchro. . . . .	93
5.27	Scott-T transformer. . . . .	94
5.28	Resolver-to-Digital converter. . . . .	94
5.29	LVDT . . . . .	95
5.30	LVDT output signal conditioner (left) and signal characteristic response. .	96
5.31	Encoder's working principle . . . . .	97
5.32	Incremental Encoder: one channel (left), two channel (quadrature encoder, right). . . . .	97
5.33	Time diagram of a quadrature encoder. . . . .	98
5.34	Gray code used in absolute encoders. . . . .	99
6.1	Relationship among different coordinate systems related to robot. . . . .	106
6.2	Cartesian coordinates of a point. . . . .	107
6.3	Cylindrical coordinates of a point. . . . .	107
6.4	Spherical coordinates of a point. . . . .	107
6.5	Two different frames with different orientation, but the same origin. . . . .	108
6.6	Sample of rotation in 2-D. . . . .	109
6.7	Rotation around $\hat{x}_A$ axis. . . . .	111

6.8	Rotation around $\hat{y}_A$ axis. . . . .	112
6.9	Rotation around $\hat{z}_A$ axis. . . . .	112
6.10	<i>Rotation Axis- Rotation Angle</i> Representation of orientation. . . . .	116
6.11	The relation between frames $\{A\}$ and $\{B\}$ involves both a translation plus a rotation. . . . .	118
6.12	Translation of a frame. . . . .	124
6.13	Translation of a position with a fixed frame. . . . .	124
6.14	Rotation $\pi/2$ around axis $\hat{z}$ . . . . .	126
6.15	Translation plus rotation of a frame. . . . .	128
6.16	Translation plus rotation of a frame: Rotation and translation are not commutative. . . . .	129
7.1	Forward Kinematics vs. Inverse Kinematics. . . . .	138
7.2	Link parameters: Link Length ( $a_i$ ) and Link Twist ( $\alpha_i$ ). . . . .	139
7.3	Joint parameters: Link Offset and Joint Angle. . . . .	140
7.4	Example 1: 3 DoF planar robot. . . . .	144
7.5	Forward Kinematics. First two steps: numbering the joints and numbering the links. . . . .	144
7.6	Forward Kinematics. Third step: defining reference systems. . . . .	145
7.7	Example 2: a three DoF robot with one prismatic joint. . . . .	148
7.8	Example 3: a 3DoF manipulator: three rotational joints, 2 axes intersecting axes and 2 parallel axes. . . . .	150
7.9	First two solutions for example 3. . . . .	150
7.10	Second two solutions for example 3. . . . .	151
7.11	Example 4: a cylindrical robot. . . . .	152
7.12	Reference systems for example 4. . . . .	153
7.13	Example 5: a 4 DoF SCARA robot. . . . .	154
7.14	Reference systems for example 5. . . . .	155
7.15	Example 1: an 5 DoF anthropomorphical robot. . . . .	158
7.16	Example 1: Denavit frames associated with a 5 DoF anthropomorphical robot. . . . .	159

7.17 Example 1: Kinematic decoupling of a 5 DoF anthropomorphical robot. . . . .	160
7.18 Example 1: one solution for the inverse kinematics of an anthropomorphic positioner. . . . .	162
7.19 All inverse kinematic solutions of an antropomorphic positioner. . . . .	165
7.20 Wrist inverse kinematics of an 5DoF anthropomorphic manipulator. . . . .	166
7.21 Wrist inverse kinematics of an 5DoF anthropomorphic manipulator. . . . .	167
8.1 Jacobian relationship. . . . .	169
8.2 Calculation of the Jacobian Matrix via velocity propagation method. . . . .	174
8.3 Calculation of the Jacobian Matrix via force/torque propagation method. .	177
9.1 Trajectory Generation vs. Joint Control. . . . .	180
9.2 Definition of a straight line trajectory (cartesian space). . . . .	181
9.3 Sampling (in space) of the straight line trajectory (cartesian space). . . . .	181
9.4 Sampled trajectory (joint space). . . . .	182
9.5 Interpolated trajectory (joint space). . . . .	182
9.6 Sampling, in time, the trajectory (joint space). . . . .	183
9.7 Joint reference values for the every joint. . . . .	183
9.8 Specified trajectory vs. actual robot's trajectory. . . . .	183
9.9 Point to point trajectories and joint by joint motion. . . . .	184
9.10 Point to point trajectories and simultaneous joint motion. . . . .	185
9.11 Coordinated or synchronous trajectories. . . . .	185
9.12 Continuous trajectories. . . . .	186
9.13 Time sampling in the Joint Space. . . . .	187
9.14 Linear interpolator. . . . .	188
9.15 Cubic interpolator. . . . .	189
9.16 Trapezoidal interpolator. . . . .	191
9.17 Trapezoidal interpolator: parabolic link between to adjacent interpolators.	192
10.1 Position and Force control loops. . . . .	194
10.2 Mechanical-electrical analogy. . . . .	194

10.3 Position and Force control loops. . . . .	198
10.4 1-DoF Proportional Error position control. . . . .	201
10.5 Step response types of systems: $\omega_i^2 > 0 \rightarrow$ overdamped system, $\omega_i^2 = 0 \rightarrow$ critically damped system, and $\omega_i^2 < 0 \rightarrow$ underdamped/oscillatory system. . . . .	203
10.6 1-DoF PID position control. . . . .	207
10.7 1-DoF PID position control. . . . .	208
10.8 Basic PID control algortimh. . . . .	209
10.9 Feedforward compensation. . . . .	210
10.10 Feedback compensation. . . . .	211
10.11 Computed-Torque control algorithm. . . . .	213
10.12 Gain Scheduling control algorithm. . . . .	215
10.13 Model Reference Adaptive Control. . . . .	216
10.14 Computed-Torque Adaptive Control. . . . .	218
i.1 Fur classification. . . . .	221
i.2 Photo of Scrbot Robot. . . . .	222
i.3 Dimension plot of Scrbot Robot. . . . .	222
i.4 RT3300 Robot. . . . .	223
i.5 Which geometries?. . . . .	224
i.6 Moving blocks. . . . .	224
i.7 Scrbot's DoFs. . . . .	226
i.8 Scrbot's kinematic chain. . . . .	226
i.9 RT3300's DoFs. . . . .	227
i.10 RT3300's kinematic chain. . . . .	228
i.11 Anthropomorphic Robot, sample 1. . . . .	228
i.12 SCARA Robot. . . . .	229
i.13 Anthropomorphic Robot, sample 2. . . . .	229
i.14 Programming example points. . . . .	230
ii.1 Robot with camera: initial configuration. . . . .	235

iii.1	Peg-in-hole task.	245
iii.2	2-DoF manipulator.	246
iii.3	3-DoF manipulator with multiple D-H solutions.	247
iii.4	PUMA manipulator.	248
iii.5	3R non-planar robot.	248
iii.6	Planar robot RPR.	248
iii.7	RRP manipulator.	249
iii.8	RRR manipulator.	249
iii.9	SCARA manipulator.	250
iii.10	Cylindrical robot.	250
iii.11	MALIBA manipulator.	251
iii.12	SCORBOT robot.	252
iii.13	RT3300 robot.	253
iii.14	D-H coordinate systems for a 2-DoF manipulator.	254
iii.15	3-DoF manipulator 1 <sup>st</sup> and 2 <sup>nd</sup> solutions.	255
iii.16	3-DoF manipulator 3 <sup>rd</sup> and 4 <sup>th</sup> solutions.	256
iii.17	Solution to an unknown robot.	257
iii.18	PUMA DoFs and D-H coordinate systems.	257
iii.19	D-H coordinate systems for a 3R non-planar robot.	258
iii.20	D-H coordinate systems for a planar robot RPR.	260
iii.21	D-H coordinate systems for a RRP manipulator.	261
iii.22	D-H coordinate systems for a RRR manipulator.	262
iii.23	D-H coordinate systems for a SCARA manipulator.	264
iii.24	D-H coordinate systems for a cylindrical manipulator.	265
iii.25	D-H coordinate systems for MALIBA manipulator.	267
iii.26	D-H coordinate systems for SCORBOT robot.	268
iii.27	D-H coordinate systems for RT3300 robot.	269
iv.1	RP manipulator.	271

iv.2	RPR manipulator.	272
iv.3	Planar arm with two joints.	273
iv.4	SCARA robot.	273
iv.5	D-H coordinate systems of a RP manipulator.	275
iv.6	D-H coordinate systems of a RPR manipulator.	277
iv.7	Inverse kinematic equations of a RPR manipulator.	278
iv.8	Workspace of a planar arm with two joints.	279
iv.9	D-H coordinate systems of a SCARA robot.	279
iv.10	Inverse kinematic equations of a SCARA robot.	281
iv.11	D-H coordinate systems of a anthropomorphic robot.	282
iv.12	Inverse kinematic equations of a anthropomorphic robot.	283
iv.13	D-H coordinate systems of a spherical robot.	284
iv.14	Inverse kinematic equations of a spherical robot: right-arm solution.	284
iv.15	Inverse kinematic equations of a spherical robot: left-arm solution.	285
v.1	RR planar manipulator.	288
v.2	RR planar manipulator with an end-point frame.	289
v.3	RRR planar manipulator.	290
A.1	The return of Hephaestus to Olympus.	300
A.2	Death of Talus.	301
A.3	Fryer Bacon's speaking head.	301
A.4	Cock of the Strasbourg Cathedral.	302
A.5	Vauncanson's duck: a diagram of its interior and a replica of it at <b>Le Musée de Automates de Grenoble</b> .	302
A.6	Von Kempelen's chess player.	303
A.7	Droz's dolls at the Art and History Museum in Neuchâtel, Switzerland.	303
A.8	Maillert's automaton and two of its drawings	303
B.1	Samples of Micro-robots: Sumo Fighter (MAZO),line-tracking (PIONERO), Microrobotics Club of TECNUN.	309

B.2	AIBO (SONY): <a href="http://www.sony.com.au/aibo">http://www.sony.com.au/aibo</a>	309
B.3	SDR 3x (SONY): <a href="http://au.playstation.com/technology/sonyrobot.jhtml">http://au.playstation.com/technology/sonyrobot.jhtml</a>	310
B.4	SDR 4x (SONY): <a href="http://au.playstation.com/technology/sonyrobot.jhtml">http://au.playstation.com/technology/sonyrobot.jhtml</a>	310
B.5	ASIMO (HONDA): <a href="http://world.honda.com/ASIMO">http://world.honda.com/ASIMO</a>	311
B.6	Specifications of ASIMO	311
B.7	Evolution of ASIMO	312

# List of Tables

---

1.1	Mechanical features of two PHANToM haptic devices.	14
3.1	Computer programming vs. robot programming.	34
3.2	Specification of the positions needed in the program.	46
4.1	Comparison between serial and parallel robots.	65
5.1	The transistor only dissipates power in the linear region.	77
5.2	Transmission Mechanisms	87
5.3	Internal sensors.	92
7.1	List of variable parameters according to the type of DoF.	141
7.2	DH parameters for example 1.	145
7.3	DH parameters for example 1, including the end-effector frame.	147
7.4	DH parameters for example 2.	148
7.5	First set DH parameters for example 3.	149
7.6	Second set DH parameters for example 3.	150
7.7	Third set DH parameters for example 3.	151
7.8	Fourth set DH parameters for example 3.	151
7.9	DH parameters for example 4.	153
7.10	DH parameters for example 5.	155
7.11	Number of solutions for inverse kinematics.	157
10.1	Summary of the analogy between mechanics and electricity.	195

iii.1 D-H table of an unknown robot. . . . .	247
--	-----

# Notation

---

## Coordinate Systems

$\{0\}$ , fixed universal coordinate system, used by default.

$\{e\}$ , coordinate system attached to  $p_e$  (Robot's end-point).

$\{i\}$ , coordinate system attached to joint  $i$ .

$\{1\dots n\}$ , coordinates system attached to joint from 1 to  $n$ .

$[\hat{x}_0, \hat{y}_0, \hat{z}_0]$ , axis of the coordinate system  $\{0\}$ .

$[\hat{x}_e, \hat{y}_e, \hat{z}_e]$ , axis of the coordinate system  $\{e\}$ .

$[\hat{x}_i, \hat{y}_i, \hat{z}_i]$ , axis of the coordinate system  $\{i\}$ .

## Cylindrical Coordinates

$r'$ , the magnitude of a vector projected onto the plane  $\hat{x}_0\hat{y}_0$ .

$\theta$ , the angle between the axis  $\hat{x}_0$  and  $r'$ .

$z$ , the magnitude of the projection of a vector onto the axis  $\hat{z}_0$ .

## Spherical Coordinates

$r$ , the magnitude -or distance- of a vector.

$\theta$ , the angle between the axis  $\hat{x}_0$  and  $r'$  (see Cylindrical coordinates).

$\phi$ , the angle between the axis  $\hat{z}_0$  and a vector.

## Rotation Matrices

${}^0 p_e$ , the vector  $p_e$  is referenced to the system  $\{0\}$ . If leading superscript is missing, this is the reference system by default.

${}^e p_e$ , the vector  $p_e$  is referenced to the system  $\{e\}$ .

${}^i p_e$ , the vector  $p_e$  is referenced to the system  $\{i\}$ .

${}^0 R_e$ , the rotation matrix of the system  $\{e\}$  with respect to the system  $\{0\}$ .

This rule is always applied when a vector is rotated:

$${}^0 p = {}^0 R_e {}^e p$$

### Basic Rotations

$Rot(\hat{x}, \psi_1)$  Rotation around  $\hat{x}$ ,  $\psi_1$  degrees:

$$Rot(\hat{x}, \psi_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 \\ 0 & \sin \psi_1 & \cos \psi_1 \end{bmatrix}$$

$Rot(\hat{y}, \psi_2)$  Rotation around  $\hat{y}$ ,  $\psi_2$  degrees:

$$Rot(\hat{y}, \psi_2) = \begin{bmatrix} \cos \psi_2 & 0 & \sin \psi_2 \\ 0 & 1 & 0 \\ -\sin \psi_2 & 0 & \cos \psi_2 \end{bmatrix}$$

$Rot(\hat{z}, \psi_3)$  Rotation around  $\hat{z}$ ,  $\psi_3$  degrees:

$$Rot(\hat{z}, \psi_3) = \begin{bmatrix} \cos \psi_3 & -\sin \psi_3 & 0 \\ \sin \psi_3 & \cos \psi_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

${}^0 \omega_e$ , the angular speed of the system  $\{e\}$  with respect to the system  $\{0\}$ :

$${}^0 \omega_e = \omega_e = \begin{bmatrix} \omega_{xe} \\ \omega_{ye} \\ \omega_{ze} \end{bmatrix}$$

$\omega \wedge, \tilde{\omega}$  skew-symmetric matrix associated to cross product by the angular speed vector:

$$\tilde{\omega}_e = {}^0 \dot{R}_e {}^0 R_e^T = \begin{bmatrix} 0 & -\omega_{ze} & \omega_{ye} \\ \omega_{ze} & 0 & -\omega_{xe} \\ -\omega_{ye} & \omega_{xe} & 0 \end{bmatrix}$$

## Euler Angles

$\psi_e$ , orientation of the end point of the robot using Euler Angles.

$$\psi_e = \begin{bmatrix} \psi_{1e} \\ \psi_{2e} \\ \psi_{3e} \end{bmatrix}$$

If roll, pitch and yaw angles are used, then . . .

$$\psi_e = \begin{bmatrix} \psi_{1e} \\ \psi_{2e} \\ \psi_{3e} \end{bmatrix} = \begin{bmatrix} \psi_{xe} \\ \psi_{ye} \\ \psi_{ze} \end{bmatrix} = \begin{bmatrix} \psi_{yaw} \\ \psi_{pitch} \\ \psi_{roll} \end{bmatrix}$$

$B_\phi$ , the matrix that relates angular velocity with respect to the time-derivative of Euler angles. If we use the roll-pitch-yaw set, we will have:

$$\omega = B_\psi \dot{\psi} = \begin{bmatrix} 1 & 0 & -\sin \psi_y \\ 0 & \cos \psi_z & \sin \psi_z \cos \psi_y \\ 0 & -\sin \psi_z & \cos \psi_z \cos \psi_y \end{bmatrix} \begin{bmatrix} \dot{\psi}_x \\ \dot{\psi}_y \\ \dot{\psi}_z \end{bmatrix}$$

## Rotated Angle/ Rotation Axis Notation

$\varphi$ , rotated angle.

$u$ , rotation axis:

$$u = [u_x \ u_y \ u_z]^T$$

The rotation matrix can be obtained as below:

$${}^0 R_e = \begin{bmatrix} - (u_z^2 + u_y^2) A + 1 & u_x u_y A - u_z B & u_x u_z A + u_y B \\ u_x u_y A + u_z B & - (u_x^2 + u_z^2) A + 1 & u_y u_z A - u_x B \\ u_x u_z A - u_y B & u_y u_z A + u_x B & - (u_x^2 + u_y^2) A + 1 \end{bmatrix}$$

$$A = 1 - \cos \varphi \quad B = \sin \varphi$$

The angular speed is:

$$\omega_e = \begin{bmatrix} u_x & B & -Au_z & Au_y \\ u_y & Au_z & B & -Au_x \\ u_z & -Au_y & Au_z & B \end{bmatrix} \begin{bmatrix} \dot{\varphi}_e \\ \dot{u}_{xe} \\ \dot{u}_{ye} \\ \dot{u}_{ze} \end{bmatrix}$$

## Quaternion (Euler Parameters)

$[e_0, e_1, e_2, e_3]$ , Euler Parameters.

$$\begin{aligned} e_0 &= \cos\left(\frac{\varphi}{2}\right) \\ e_1 &= u_x \sin\left(\frac{\varphi}{2}\right) \\ e_2 &= u_y \sin\left(\frac{\varphi}{2}\right) \\ e_3 &= u_z \sin\left(\frac{\varphi}{2}\right) \end{aligned}$$

$e$ , the vector composed of  $e_0, e_1, e_2, e_3$ :

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}$$

$B_q$ , the matrix that relates angular velocity with respect to the time-derivative of Euler parameters.

$$\omega = B_q \begin{bmatrix} \dot{e}_0 \\ \dot{e} \end{bmatrix} = \begin{bmatrix} -e_1 & e_0 & -e_3 & e_2 \\ -e_2 & e_3 & e_0 & -e_1 \\ -e_3 & -e_2 & e_1 & e_0 \end{bmatrix} \begin{bmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix}$$

## Homogeneous Coordinates

$$p^{4D} = \begin{bmatrix} p \\ 1 \end{bmatrix} = \begin{bmatrix} wp_x \\ wp_y \\ wp_z \\ w \end{bmatrix}$$

vector  $P$  represented in 4D space as homogeneous vector.

${}^0T_e$ , the homogeneous transform matrix of the system  $e$  with respect to the system 0. Its normal definition:

$${}^0T_e = \begin{bmatrix} ({}^0R_e)_{3x3} & ({}^0p_e)_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} \text{rotation} & \text{translation} \\ \text{perspective} & \text{scaling} \end{bmatrix}$$

$P_{0e} = P_e = [p_{x0e} \ p_{y0e} \ p_{z0e}]^T$ , translation vector between two reference systems.

$({}^0R_e)_{3x3}$ , rotation matrix in 3D space.

$f_{1x3}$ , perspective transformation.

$w_{1x1}$ , global scaling factor.

But in robotics, the common definition is:

$${}^0T_e = \begin{bmatrix} {}^0R_e & {}^0p_e \\ [0 & 0 & 0] & 1 \end{bmatrix}$$

This rule is always applied when a vector is rotated and translated using homogeneous vectors:

$${}^0p_e^{4D} = {}^0T_e {}^ep_e^{4D}$$

### **Basic transformations:**

*Trans* ( $p_{0e}$ ), Translation around  $p_{0e}$ . The homogeneous transformation that involves it is:

$$\text{Trans} (p_{0e}) = \begin{bmatrix} 1 & 0 & 0 & p_{0e} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*Rot* ( $\hat{x}, \psi_1$ ), Rotation around  $\hat{x}$ ,  $\psi_1$  degrees . The homogeneous transformation that involves it is:

$$\text{Rot} (\hat{x}, \psi_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 & 0 \\ 0 & \sin \psi_1 & \cos \psi_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*Rot* ( $\hat{y}, \psi_2$ ), Rotation around  $\hat{y}$ ,  $\psi_2$  degrees. The homogeneous transformation that involves it is:

$$\text{Rot} (\hat{y}, \psi_2) = \begin{bmatrix} \cos \psi_2 & 0 & \sin \psi_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \psi_2 & 0 & \cos \psi_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*Rot* ( $\hat{z}, \psi_3$ ), Rotation around  $\hat{z}$ ,  $\psi_3$  degrees. The homogeneous transformation that involves it is:

$$= \text{Rot} (\hat{z}, \psi_3) = \begin{bmatrix} \cos \psi_3 & -\sin \psi_3 & 0 & 0 \\ \sin \psi_3 & \cos \psi_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## **Position/Orientation**

$n$ , robot degrees of freedom (DoF).  $q$ , joint space coordinate (rotating or linear).

$$q = [q_1 \ q_2 \ \dots \ q_n]^T$$

$\theta_i$ , joint space coordinate (rotating DoF).

$d_i$ , joint space coordinate (translating DoF).

$p_e$ , (lowercase) position of the end point of the robot:

$$p_e = [p_{xe} \ p_{ye} \ p_{ze}]^T$$

$P_e$ , (uppercase) position and orientation of the end point of the robot and in the case of using Euler Angles we have:

$$P_e = \begin{bmatrix} p_e \\ \psi_e \end{bmatrix}$$

$$P_e = [P_{xe} \ P_{ye} \ P_{ze} \ \psi_{1e} \ \psi_{2e} \ \psi_{3e}]^T$$

$\dot{q}$ , rate of change of location of  $q$

$$\dot{q} = [\dot{q}_1 \ \dot{q}_2 \ \dots \ \dot{q}_n]^T$$

$\dot{P}_e$ , rate of change of location (position+orientation) of  $P_e$ . If we use the Roll-Pitch-Yaw notation, we have:

$$\dot{P}_e = \begin{bmatrix} (\dot{P}_e)_{3x1} \\ (\dot{\psi}_e)_{3x1} \end{bmatrix}$$

$$\dot{P}_e = [\dot{p}_{xe} \ \dot{p}_{ye} \ \dot{p}_{ze} \ \dot{\psi}_{xe} \ \dot{\psi}_{ye} \ \dot{\psi}_{ze}]^T$$

$v_e$ , translational speed of robot's end-point.

$$v_e = [v_{xe} \ v_{ye} \ v_{ze}]^T = [\dot{p}_{xe} \ \dot{p}_{ye} \ \dot{p}_{ze}]^T$$

$\omega_e$ , angular speed of robot's end-point.

$$\omega_e = [\omega_{xe} \ \omega_{ye} \ \omega_{ze}]^T \neq [\dot{\psi}_{xe} \ \dot{\psi}_{ye} \ \dot{\psi}_{ze}]^T$$

$V_e$ , translational+rotational speed of robot's end-point.

$$V_e = \begin{bmatrix} v_e \\ \omega_e \end{bmatrix}$$

$$V_e = [\dot{p}_{xe} \quad \dot{p}_{ye} \quad \dot{p}_{ze} \quad \omega_{xe} \quad \omega_{ye} \quad \omega_{ze}]^T$$

$\dot{v}_e$ , translational acceleration of robot's end-point.

$\dot{\omega}_e$ , angular acceleration of robot's end-point.

### Forward and Inverse Kinematics (Position)

$fkin$ , Forward Kinematics.

$ikin$ , Inverse Kinematics.

### Denavit-Hartenberg parameters

$a_{i-1}$ , the length of link  $i - 1$ .  $\alpha_{i-1}$ , the twist of link  $i - 1$ .  $d_i$ , the offset of link  $i$ .  $\theta_i$ , the angle of joint  $i$ . Then, the homogeneous transform between the coordinate frames  $\{i - 1\}$  and  $\{i\}$  is:

$${}^{i-1}T_i = Trans(\hat{x}_{i-1}, a_{i-1}) Rot(\hat{x}_{i-1}, \alpha_{i-1}) Trans(\hat{z}_i, d_i) Rot(\hat{z}_i, \theta_i)$$

### Static Forces/Torques

$f_e$ , force exerted on the origin of frame  $\{e\}$ .

$$f_e = [f_{xe} \quad f_{ye} \quad f_{ze}]^T$$

$n_e$ , torque exerted on the origin of frame  $\{e\}$ .

$$n_e = [n_{xe} \quad n_{ye} \quad n_{ze}]^T$$

$F_e$ , force and torque exerted on the origin of frame  $\{e\}$ .

$$F_e = \begin{bmatrix} f_e \\ n_e \end{bmatrix}$$

$\tau$ , force/torque exerted on every joint, that is to say, the force/torque exerted in the joint-space.

$$\tau = [\tau_1 \quad \tau_2 \quad \dots \quad \tau_n]^T$$

## Jacobian and Singularities

$J_q$ , jacobian of a serial manipulator or direct jacobian matrix ( $J_q = \text{Identity}$  in case of parallel robots).

$J_x$ , jacobian of a parallel manipulator or inverse jacobian matrix ( $J_x = \text{Identity}$  in case of serial robots).

The jacobian relationships are:

$$\begin{aligned} J_x V_e &= J_q \dot{q} \\ \tau = J_q^T J_x^{-T} F_e &\Leftrightarrow \text{in the case of } \exists J_x^{-1} \end{aligned}$$

In case of  $\exists J_x^{-1} \Rightarrow$  we can write:

$$J = J_x^{-1} J_q$$

In this case, the jacobian relationships are simplified into:

$$V_e = J \dot{q}$$

$$\tau = J^T F_e$$

$\sigma_i$ , the  $i - th$  singular value of the jacobian matrix.

$\sigma_{max}$ , the maximum singular value of the jacobian matrix.

$\sigma_{min}$ , the minimum singular value of the jacobian matrix.

$cond$ , condition number.

$$cond = \frac{\sigma_{max}}{\sigma_{min}}$$

$\det(J)$ , the determinant of the jacobian matrix is the jacobian.

$\kappa = \sqrt{(\det(J^T J))}$ , Yoshikawa's manipulability index.

$\kappa = \det(J)$ , Yoshikawa's manipulability index for square jacobian matrices.

## Dynamics of Manipulators

$I$ , moments of inertia.

$I_{xx}, I_{yy}, I_{zz}$ ; mass moments of inertia.

$I_{xy}, I_{xz}, I_{yx}, I_{yz}, I_{zx}, I_{zy}$ ; mass products of inertia.

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

$m$ , mass.

$b$ , damper.

$k$ , stiffness.

$\mathcal{K}$ , potential energy.

$\mathcal{U}$ , kinetic energy.

$\mathcal{L} = \mathcal{K} - \mathcal{U}$ , Lagrange function.

## Dynamics in joint-space

$M_q(q)$ , the actual robot mass matrix (nxn).

$V_{cor_q}(q, \dot{q})$ , the actual robot coriolis force(nx1).

$V_{cen_q}(q, \dot{q})$ , the actual robot centrifugal force (nx1).

$V_{iner_q}(q, \dot{q}) = V_{cen_q}(q, \dot{q}) + V_{cen_q}(q, \dot{q})$ , the actual robot coriolis and centrifugal forces (nx1).

$F_{fricV_q}(\dot{q})$ , the actual robot dynamic friction (damping) (nx1).

$F_{fricC_q}(q, \text{sgn}(\dot{q}))$ , the actual coulomb friction (nx1).

$F_{fric_q} = F_{fricC_q}(q, \text{sgn}(\dot{q})) + F_{fricV_q}(\dot{q})$ , the actual friction (nx1).

$G(q)$ , the actual gravity force (nx1).

The actual dynamic equation in joint space is:

$$\tau = M_q(q)\ddot{q} + V_{iner_q}(q, \dot{q}) + F_{fricV_q}(\dot{q}) + F_{fricC_q}(q, \text{sgn}(\dot{q})) + G_q(q)$$

The estimated dynamic equation in joint-space is:

$$\hat{\tau} = \hat{M}_q(q)\ddot{q} + \hat{V}_{iner_q}(q, \dot{q}) + \hat{F}_{fricV_q}(\dot{q}) + \hat{F}_{fricC_q}(q, \text{sgn}(\dot{q})) + \hat{G}_q(q)$$

## Dynamics in cartesian-space

$$F_e = M_x(q)\dot{V}_e + V_{iner_x}(q, \dot{q}) + F_{fricV_x}(V_e) + F_{fricC_x}(q, \text{sgn}(\dot{q})) + G_x(q)$$

## Control of Manipulators

### Control of 1 DoF

$k_{p_i}, k_{I_i}, k_{d_i}$ , the proportional, integral and derivative gains of a PID controller of the  $i - th$  joint.

$t_{I_i}, t_{d_i}$ , the integral and derivate time constants of a PID controller of the  $i - th$  joint.

$q_{ref_i}$ , the reference signal for the magnitude  $q_i$ .

$q_{real_i}$ , the actual signal for the magnitude  $q_i$ .

$q_{meas_i}$ , the measured signal for the magnitude  $q_i$ .

$q_{err_i} = q_{ref_i} - q_{meas_i}$ , the error signal for the magnitude  $q_i$ .

## Control of n DoFs

$K_p, K_I, K_d$ , the proportional, integral and derivative matricial gains of a PID controller.

$T_I, T_d$ , the integral and derivate time constants of a PID controller.

$q_{ref}$ , the reference signal for the magnitude  $q$ .

$q_{real}$ , the actual signal for the magnitude  $q$ .

$q_{meas}$ , the measured signal for the magnitude  $q$ .

$q_{err} = q_{ref} - q_{meas}$ , the error signal for the magnitude  $q$ .

## Electrical Analogy

### Electrical system

$e$ , electrical voltage.

$e_1$ , input voltage in a two-port network.

$e_2$ , output voltage in a two-port network.

$i$ , electrical current.

$i_1$ , input current in a two-port network.

$i_2$ , output current in a two-port network.

### Mechanical system

$f$ , mechanical force.

$f_1$ , input force in a two-port network.

$f_2$ , output force in a two-port network.

$v$ , mechanical speed.

$v_1$  input speed in a two-port network.

$v_2$ , output speed in a two-port network.

### Analogy

Electrical system		Mechanical system	
Voltage	$e_1, e_2$	Force	$f_1, f_2$
Current	$i_1, i_2$	Velocity	$v_1, v_2$
Inductance	$L$	Mass	$m$
Resistance	$R$	Damping	$b$
Reciprocal capacitance	$\frac{1}{C}$	Stiffness	$k$

### Two-port Network Matrices

$Z$ , impedance matrix:

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = Z \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} \quad \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = Z \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$Y$ , admittance matrix:

$$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = Y \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \quad \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = Y \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

$H$ , hybrid matrix:

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = H \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} \quad \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = H \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$G$ , 2<sub>nd</sub> hybrid matrix:

$$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = G \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \quad \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = G \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Transmission matrix:

$$\begin{bmatrix} e_1 \\ i_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} e_2 \\ i_2 \end{bmatrix} \quad \begin{bmatrix} f_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} f_2 \\ v_2 \end{bmatrix}$$



# **Part I**

# **Introduction to Robotics**



## Chapter 1

# Introduction

---

### 1.1 Introduction

The word **robot** represents a very general concept that is hard to define precisely, since this word can be used to name a great amount of machines.

The aim of this chapter is to give an approximation to the concept of robot. Thus a brief history of robotics, the two technological roots of robotics, and finally an explanation to basic robotic jargon are presented.

Let's start with a little riddle: which the following examples are robots?

- An automatic camera?
- An automatic washing machine?
- An automatic dishwasher?
- An automatic car (a car with an automatic gearbox)?
- A crane<sup>1</sup>?

For some people, some or all of these examples are kinds of robots, for others none of them are. It all depends on what our own idea of what a robot is, and this can vary a lot from person to person. The word robot is often related with some kind of machine that has similarities to some part of human (or animal), body. But this is not always true ...

### 1.2 A Little about the History

Although in appendix A a review is made of remarkable milestones in **Robot history** from the Ancient Greek culture to the present, let's see only the beginning of modern robotics.

---

<sup>1</sup>grúa

The word robot was used for the first time in 1920 in a play entitled **R.U.R. (Rossum's Universal Robots)**. This play was written by **Karel Čapek**, a Czech playwright. In the original Czech, **Robota** means forced servitude. The name Rossum is an allusion to the Czech word rozum, which means reason, intellect. After the production of R.U.R. opened, the word robot displaced older words such as automaton or android in most languages. R.U.R stands for "Rossumovi Umělí Roboti" (Mr. Reasson's Artificial Robots) and when the play was first put on in English-speaking countries, this title was translated from the Czech as "Rossum's Universal Robots" in order to fit the initials R.U.R (Figure 1.1).

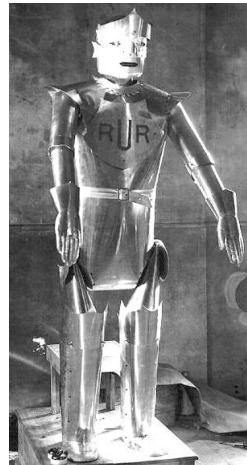


Figure 1.1: A representation of a Rossum's Universal Robot.

In Čapek's play, robots were human-like (humanoid) machines that were created by Rossum and his son to become servants of real human beings. But robots end up not accepting this role and rebel against humans. This resulted in a violent conflict between the people and the robots. However, it is the writer **Isaac Asimov** to whom we owe the survival of the word robot. He is regarded as the father of Science Fiction Robots, since he was the first to refer to the science of robotics in a series of short stories that were collected in the book **I, Robot** (1950).

One of these stories is about a robot named Robbie <sup>2</sup>:

Robbie was a non-vocal robot. He couldn't speak. He was made and sold in 1996. Those were the days before extreme specialization, so he was sold as a nursemaid.

Dr Susan Calvin, U.S. Robotics, 2058

In the story, Dr Calvin was an expert on robot-psychologists, and who was talking about the history of the company of the occasion of her retirement.

The Asimov's vision is different from that of Čapek. The latter envisages a pessimistic scenario for the relationship among humans and robots. The former defines in *I, Robot*

---

<sup>2</sup>Text from the story Robbie that was first published as Strange Playfellow in Super Science Stories; 1940, Fictioners, Inc; 1968, Isaac Asimov

robots as intelligent machines that have positronic brains. These positronic brains are programmed by humans, who stamp into them the **three laws of robotics**, namely:

**First Law** A robot must not harm a human being or, through inaction, allow one to come to harm;

**Second Law** A robot must always obey human beings unless that is in conflict with the First Law;

**Third Law** A robot must protect itself from harm unless that is in conflict with the First or Second Law.

Latter, the **Zeroth Law**, was added: A robot may not injure humanity, or, through inaction, allow humanity to come to harm.

Asimov's robot stories are a kind of exploration of the implications of implementing these laws in robots. However, the works of most other authors ignore or even contradict them. So, these laws have not prevailed as Asimov intended.

Thus these three Laws are sometimes seen as a future design directives that should be considered by the people that would work in artificial intelligence, once an artificial intelligence has reached the stage where it can comprehend these laws.

## 1.3 The Technological Roots of Robotics

The origins of robotics are linked to industrial robotics and can be traced in two separate, but related, technological developments. The first source came from the development of the **Computer Numerically Controlled Machine Tool** or **CNC Machine**. The first CNC machine was developed at the MIT Servomechanisms Lab, USA, in 1952. It was the first programmable industrial machine tool.

Secondly, working at the Aragonne National Labs, USA, R. C. Goertz demonstrated the first mechanical **telemanipulator** (teleoperated device) in 1948 (Figure 1.2). Seven years later, in 1954, he produced the first electric powered telemanipulator, which also had bilateral control.

The term **bilateral control** is used to define a kind of control in which two robots are considered: the **master** and the **slave robot**. The operator grasps and moves the master robot, and the slave robots tracks. But the interaction forces/torques between the slave robot and its environment are also fed back to the master unit. This was a kind of haptic interface, of the sort that was later to be developed for more 'realistic' virtual reality systems.

These two different technologies were subsequently combined in 1954, by George C. Devol in his **Programmable Object Transfer Device**, a kind of programmable robot manipulator as the result of a combination of the electric telemanipulator and CNC control technologies.

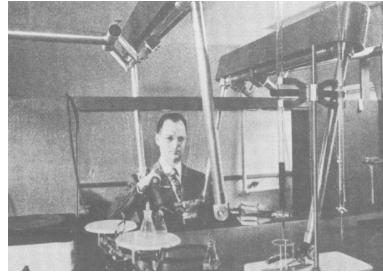


Figure 1.2: First teleoperated device. It was designed for radioactive material handling.

Then, in 1956, George C. Devol and Joseph F. Engelberger founded Consolidated Controls Corporation, which produced the first industrial robot to be installed in a General Motors factory in New Jersey. It was called the Unimate robot (universal automation robot). The name of the firm was later changed to Unimation Inc.. The Unimate was thus the first Industrial Robot, and Joseph Engelberger became known as the father of the Industrial Robot.

## 1.4 First Approach

A **robot** is a group of several **subsystems** each with its own function:

**Mechanical system** By which the robot interacts with the surrounding environment. It usually performs one particular task. It consists of actuators, joints, wrists, tools, etc. . .

**Electrical system** Consisting of sensors, electrical/pneumatic/hydraulic actuators, computers, etc. . .

**Control system** This system receives high level orders and translates them into commands for actuators.

**Sensor system** It measures different physical magnitudes so that control system is able to perform the correct action.

The main feature for a robot is the availability of being reprogrammable. So it can be said that a robot is. . .

. . . robot is a machine which can be programmed to do a variety of tasks, in the same way that a computer is an electronic circuit which can be programmed to do a variety of tasks.

–Introduction to Robotics, Mac Kerrow

Another way of defining it is. . .

... a computer-controlled mechanical device that can be programmed to do a variety of tasks without human supervision.

In this first approach, we will divide robots into three different categories: Science Fiction Robots, Toy Robots, and Real Robots.

### 1.4.1 Science Fiction Robots

Since Robbie, in Asimov's story, there have been a lot more **Science Fiction robots**: R2D2, C3P0 (Star Wars); HAL 9000 (2001: A Space Odissey); T1000 (Terminator), but to name a few (Figure 1.3). The vast majority of these characters are baddies who typically act violently against people. In a sense, it is a little surprising that robots have been so often described as bad characters in Science Fiction stories.

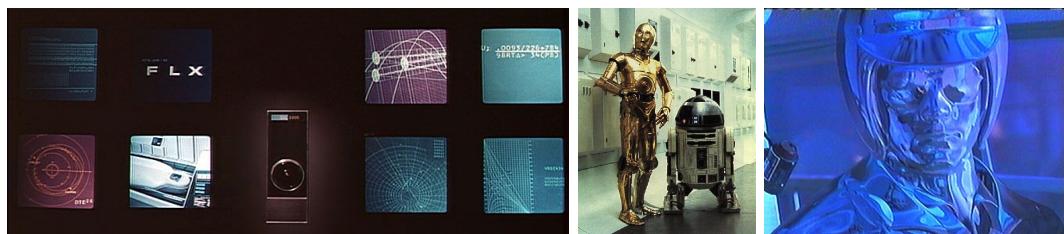


Figure 1.3: Samples of Science Fiction Robots: HAL, R2D2, C3PO, T1000 (Terminator).

Given that Since Science Fiction stories, films, and television programmes, are the source of most people's images and ideas about robots, it is hardly surprising that for many people robots are not bad or even frightening things. However, one of the possible answers to our question about what is a robot is precisely that a robot is a science fiction character. This is not the kind of robot we will consider in this course.

### 1.4.2 Toy Robots

As the second category, a robot could also be defined as a toy (Figure 1.4). There are many examples of **toy robots**. Several of them are essentially models of science fiction robots, others are mere toys that we consider them as robots, or imitate life, for example Sony's Aibo pet robot dog.



Figure 1.4: Samples of Toy Robots: Bender, R2D2.

Then, a question is arisen: when can a toy be thought as a robot? Not all toys that move around and make noises are robots. For most people, to be a robot, even a toy one, it is necessary to have arms, maybe legs, a head and eyes. In other words it is necessary to have a more or less humanoid form. This idea, the humanoid form, is important, and comes from the images of science fiction robots, most of which are also humanoid. Therefore, robots can be also toys, and they normally have a humanoid form.

### 1.4.3 Real Robots

The last category groups the robots that operate in the real world. They can be further subdivided into four different types:

**Industrial robot** these are the vast majority of existing robots;

**Service robots** there are hardly any yet;

**Biomedical robot** a quite new and promising application field to robotics;

**Experimental or Scientific robots** the second most popular type of real robot.

**INDUSTRIAL ROBOTS** The industrial robots can also be further divided into robot manipulator arms and mobile robots.

**Industrial mobile robots** are often called **automatic guided vehicles (AGVs)**, and they are used to transport components and materials in factories (Figure 1.5). The vast majority of industrial robots are, however, manipulator arms fixed in position, and used to spray paint, make solder joints, assemble components, and manipulate and transport objects. They can adopt several forms depending upon the particular application for which they are used.



Figure 1.5: Samples of Industrial Robots: an old Puma 7000; HAZBOT II, a robot for handling explosives.

Most industrial robots simply do what they are programmed to do, and cannot change, modify, or adapt what they do to deal with changes, variations, or unexpected events in their surrounding environments. Thus they only work in very “carefully controlled environments” (work cells, structured environments) in which there is very little uncertainty, variation, and no unexpected events. This is also a reason why the work cells are well isolated from workers while the robots are in action. Moreover, the

industrial robots are typically large powerful machines that can easily seriously injure or kill people if something goes wrong (safety).

We now need to go back to our search for the definition of robot. Once again, it all depends on what you are happy to call a robot. For most people, a robot is a machine that can make its own decisions about what it ought to do, and that can react to internal and external events and respond to changes in the surrounding environments. This could be quite a good way of defining what a robot is. The problem is that most industrial robot manipulator arms, the most widely spread robot, do not fit in this definition.

**SERVICE ROBOTS** The Service robots are robots that mainly work outside the factory, and are normally supposed to provide direct services to people (Figure 1.6). There are still very few real service robots in the world, but the most popular applications or proposed applications are floor cleaning, security patrol, (paper) mail delivery, and other kinds of delivery tasks.

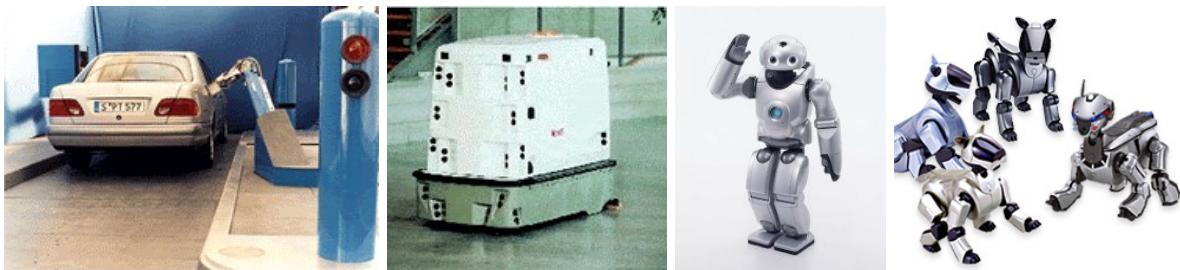


Figure 1.6: Samples of Service Robots: a tank filling robot (Reis Robotics, Germany); a cleannig robot,(HACOmatic, Hakoberke, Germany), pet robot QRIO (Sony), pet robot Aibo (Sony).

These and other service robot applications need mobile robots. There are, however, machines that exist today that we might also call service robots, though they are not mobile robots. The automatic car washing machines, the kind you drive into, might also be thought of as a kind of robot that provides a service to people.

**BIOMEDICAL ROBOTS** Strictly speaking, biomedical robots<sup>3</sup> are a type of service robot since they work mainly outside the factory. However, the number of medical robotic applications has been increasing so much in recent last years that they deserve a specific category and a different treatment. But notice that, despite this different treatment, they share many characteristics with service robots because biomedical robots also provide direct services to people. We can distinguish two main types of biomedical robots:

- Robots that assist doctors in planning, training and performing surgery (surgery robots, figure 1.7);
- Robots that assist patients in reducing the effects of impairments caused by a trauma. This effect reduction could be replacing a lost limb (mechatronic prothesis),

---

<sup>3</sup>This type of robot is sometimes referred to as biorobots. But rigorously speaking, biorobots are robots that try to mimic the behaviour of living beings and bear no relation to robots that are used in medicine.

helping the patient move the impaired limb (mechatronic orthosis) or as a part of a rehabilitation program for the impaired limb.



Figure 1.7: Samples of biomedical robots for planning (left) and performing (right) surgery.

Finally, we can also include autonomous wheelchairs in this category.

Regarding the search for a precise definition of robots, once again, it all depends on what you are happy to call a robot. For most people, a robot is a machine that can make its own decisions about what it ought to do, and that can react to internal and external events and respond to changes in the surrounding environments. But as the robotic reasoning capacity is very low, and the failure probability is high, with biomedical robots it is not possible to let the robot be autonomous. Compared to industrial robots, where they are supposed to replace human work, biomedical robots are supervised by humans and are only intended as assistive technology. Thus, for us, a biomedical robot will be a mechatronic, programmable device for surgery assistance (planning/training and performing) and/or rehabilitation purposes.

**EXPERIMENTAL ROBOTS** Experimental or scientific robots, as their name suggests, are robots used for robotic research purposes or other scientific applications. One of the most famous scientific robot in aeroespacial developments in XX century was Sojourner that was used to explore a small part of the surface of Mars in July 1997 (Figure 1.8). For some people, however this was not a robot, because it was teleoperated: it received instructions for what it had to do from Earth. These instructions were issued by a human operator who could watch images form the various on board cameras and other sensors and system parameters.

**Teleoperation** By **Teleoperation** we mean "work at a distance", thus the operator controls the robot from a (large) distance. The operator is in the local environment, while the robot is in the remote environment (Figure 1.9). Teleoperation provides the capacity of changing the working scale, as for example a surgeon may use micromanipulator technology to conduct surgery on a microscopic level. By teleoperation, we aim the **Telepresence** which means "feeling as if you were operating directly in the remote environment".

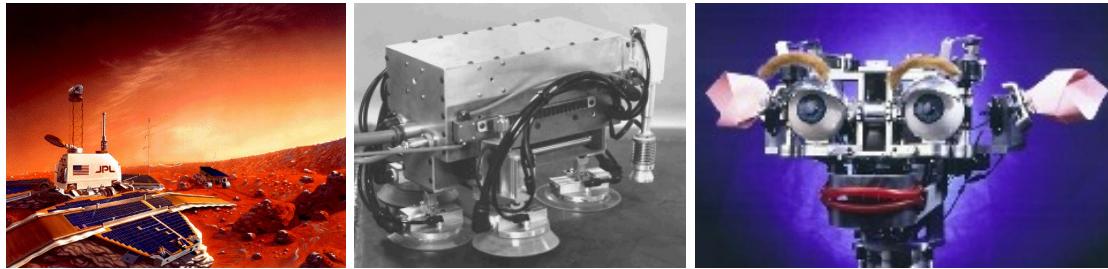


Figure 1.8: A picture of: Mars PATHFINDER and SOJOUNER on Mars; ROBICEN, a climbing robot (CEIT), KISMET (MIT).



Figure 1.9: Sample of Teleoperation device: SIMANTEL, CEIT.

A teleoperated system usually consists of two robots: the **Master robot** and the **Slave robot**. The former is controlled directly by the human operator; the latter tracks the master's trajectory. Some systems are capable of force feedback <sup>4</sup>; that is, the operator can feel the forces and torques exerted by the remote environment on the slave (Figure 1.10).

These systems are used to undertake tasks in hostile environments for humans such as in mines, outer space, areas of high radioactivity, bomb disposal sites, underwater, etc. In these situations, the master robot is kept in a safe zone while the slave performs the desired task in the remote hazardous environment.

**Haptic** Haptic devices were developed in the early 90s. The word haptic comes from the Greek “haptethai” and means “related to touch” or “tactile”. It refers to the manual interaction with environments, either for exploration or manipulation. A haptic device is a mechanism designed to interact with humans, allowing a tactile connection with a remote or virtual environment. These devices allow providing the user with additional information of the working environment through the sense of touch. This information, for example, could consist of the physical attributes of the objects being manipulated, their hardness, texture, or inertia. That is to say, haptic devices do for the sense of touch what TV screens do for vision.

<sup>4</sup>this force rendering capacity is also known as haptic feedback and the devices capable of haptic feedback are haptic devices.

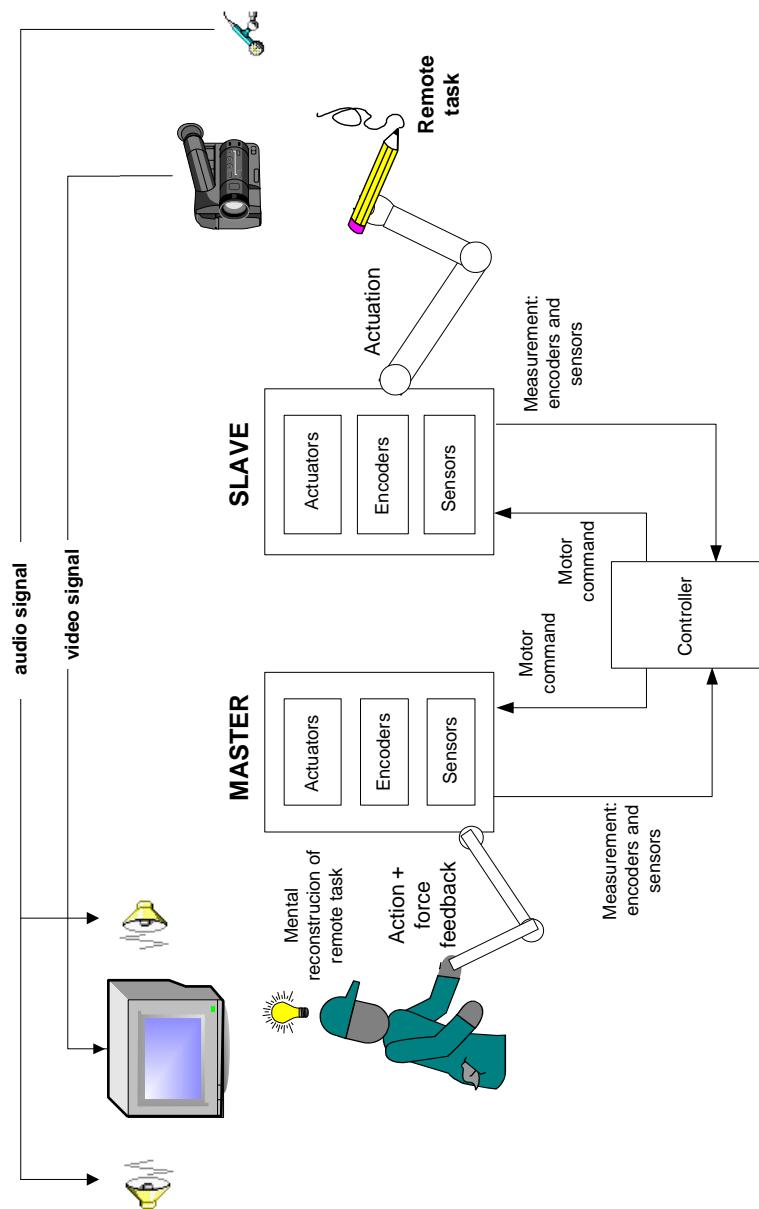


Figure 1.10: Teleoperation architecture.

The vast majority of human-machine interfaces have a unidirectional information flow. The user receives information through the senses of sight and hearing, and may interact with the environment through the use of a peripheral device (e.g. a mouse). This interaction is unidirectional: from the user to the environment. In contrast, in case of haptic devices, it is bidirectional (Figure 1.11). The inclusion of the sense of touch allows the interaction with the environment to be more intuitive and real, giving the user additional information and improving the sense of immersion.

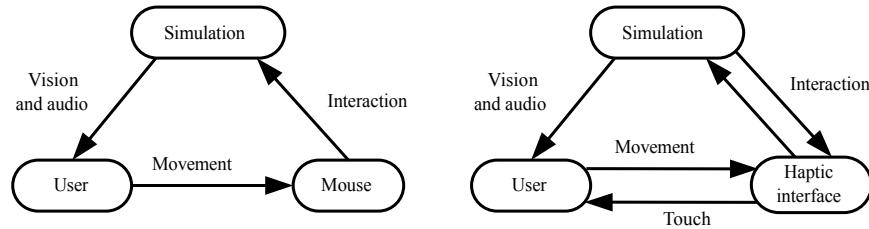


Figure 1.11: The bidirectionality of the haptic channel.

Figure 1.12 shows the most common components that comprise a haptic system. On one side is the application or scenario on which the user wishes to interact or complete a task. Typically, the scenario is virtual and it is visually represented by specific hardware (monitors, stereoscopic projections, etc.). Additionally, the system can also transmit information through the hearing channel. On the other hand, the haptic device is the mechanism that allows the user to physically interact with the scenario. With this device the user can manipulate a virtual object of the scenario, called **avatar**, controlling its movements and receiving tactile information due to the interaction of the avatar with the other objects.

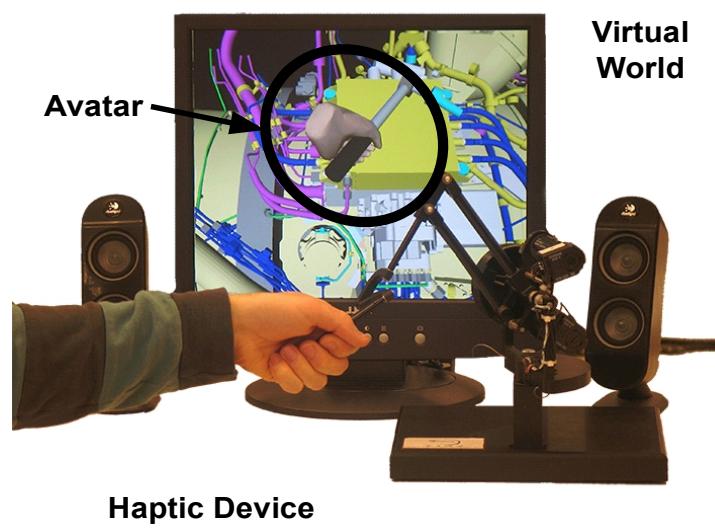


Figure 1.12: Example of a haptic system.

Haptic devices are classified into kinesthetic devices and vibratory devices. The former can be desktop devices or they can be fixed to the floor, and they exert

forces/torques on user's limbs causing the modification of the body position or simply varying the strength of user's muscles, tendons and joints<sup>5</sup>. Thus, kinesthetic haptic devices enable the user the perception of virtual objects' stiffness and weight mainly, causing an input for the proprioceptive system<sup>6</sup>. Vibratory tactile devices are placed in the area of the body with which we want to interact. They usually have vibratory parts that excite cutaneous nerve endings, thus, by means of these devices we can mimic textures and basic relief shapes.

Among the different simplest mechanical force-reflection devices we can highlight some such as the "Logitech WingMan" or the "CyberGlove" system from Immersion Corporation (Figure 1.13). The first one works like a simple mouse, but it is able to provide the user with tactile stimuli in the form of vibrations. Its main application is in the field of video gaming. The second system is capable of providing tactile stimuli feedback on the fingers and palms, also through vibrations, and is useful for interacting with the hand in virtual scenarios.



Figure 1.13: Two commercial haptic devices with tactile feedback through vibrations.

Possibly the best known variety of commercial haptic devices is PHANTOM, from the SensAble Technologies Company (Figure 1.14). From the first prototype developed by Massie and Salisbury in 1994, various devices have been developed which vary in the number of active degrees of freedom, their workspace and their ability to provide forces and torque feedback. They are desktop devices where the user picks up a pen-shaped tool, with which the position and orientation of a virtual tool can be controlled. Figure 1.1 shows the characteristics of two PHANTOM haptic devices.

<b>Feature</b>	<b>Omni</b>	<b>Premium 1.5</b>
Sensed Degree of Freedom (DoF)	6	6
Actuated Degree of Freedom (DoF)	3	6
Workspace	16x12x7 cm	38x26x19cm
Accuracy (translation)	0.055mm	0.03mm
Maximum Force (peak)	3.3 N	8.5 N
Maximum Force (continuous)	0.88 N	1.4 N
Apparent inertia	45 g	136 g

Table 1.1: Mechanical features of two PHANTOM haptic devices.

An special example of kinesthetic haptic device is the so called exoskeleton (Fig-

<sup>5</sup>That is what the term kinesthetic refers to.

<sup>6</sup>The sense that detects bodily position, weight, or movement of the muscles, tendons, and joints.



Figure 1.14: PHANToM haptic devices.

ure 1.15). These devices—unlike the previous ones, that only interact with user's hand—act directly on the concerned complete body limb. These devices are designed primarily for rehabilitation tasks in medicine or for military use as body extenders to amplify soldier's force, and can be anchored to the ground or carried by the user.



Figure 1.15: Exoskeletons designed by PERCRO (left) and the University of Washington (right) for medical rehabilitation.

## 1.5 Basic Terminology in Robotics

Before starting with the next chapter, we need to establish some terminology.

### 1.5.1 Kinematic Chain

A **Kinematic Chain** is a collection of elements, bodies, bonds, and links interconnected by joints (or articulations). There are two types:

**Open Chain** when the last link is not connected to the first link.

**Closed Chain** when the last link is connected to the first link.

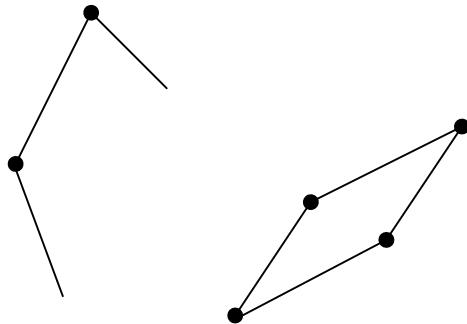


Figure 1.16: Open kinematic chain (left) vs. closed kinematic chain (right).

### 1.5.2 Mechanism/Manipulator

A **mechanism** is a **kinematic chain** in which one of the links is fixed. This link is the reference for the remaining links.

A manipulator/arm is a mechanism. Then we distinguish two categories of manipulator:

**Serial manipulator** a mechanism based on an open kinematic chain.

**Parallel manipulator** a mechanism based on a closed kinematic chain.

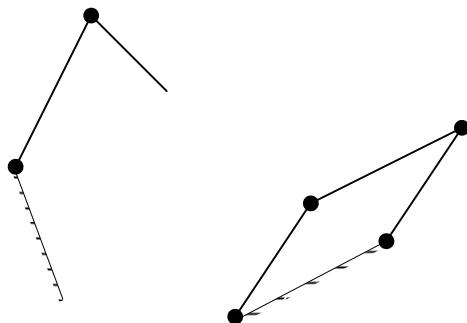


Figure 1.17: Serial mechanism/manipulator (left) vs. parallel mechanism/manipulator (right).

We can find **hybrid manipulators** that are combination of parallel/serial manipulators:

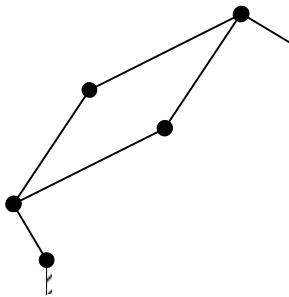


Figure 1.18: Hybrid mechanism/manipulator.

### 1.5.3 Degree of Freedom (DoF)

It is the minimum **number of independent coordinates** with which we can determine the arrangement of a mechanism. If serial manipulators are considered, it will correspond with the number of joints. Then, it will be expressed as:

$$q = [q_1 \quad q_2 \quad \dots \quad q_n]^T \quad (1.1)$$

where  $n$  is the number of degrees of freedom.

Usually industrial robot arms have between 4 and 6 degrees of freedom, one at each joint.

### 1.5.4 End Point

It is the point of the mechanism/manipulator that we want to place in a specific location. Mathematically, we will define the end-point as  $p_e$ . It is also where the robot's end-effector (the gripper or the tool) is attached.

$$p_e = [p_{x_e} \quad p_{y_e} \quad p_{z_e}]^T \quad (1.2)$$

If, for example, the robot has a two-finger **gripper**, to pick things up with, we usually define  $p_e$  to be a point between the two **fingers** (when they are open), so that when this point is geometrically inside some object to be picked up, all the robot has to do is to close the fingers of its gripper to grasp the object. It can then move away with the object between its fingers.

It is not sufficient for  $P_e$  just to be defined as a point. We also need to attach or (conceptually) fix a coordinate system to it, so that we can define both the position of  $p_e$  in space, and its orientation ( $\psi_e$ ). In this way we are able to define the position and orientation of the robot's gripper in terms of the position and orientation of  $P_e$ .

$$P_e = \begin{bmatrix} p_e \\ \psi_e \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \end{bmatrix} \quad (1.3)$$

### 1.5.5 Cartesian Space vs. Joint Space

The robot's **end-point** can be determined by the values of the joint positions of the arm ( $q_1, q_2, q_3$ , etc.) and the geometry of the elements of the robot arm that connect each pair of joints. Then we say that the robot's end point is defined in the joint space.

The position and orientation of the end point can also be defined with respect to some global Cartesian frame of reference, some global coordinate system. For this, we usually use a frame of reference fixed to the base of the robot, which should not move. Thus, we determine the position and orientation of the end-point in the Cartesian space.

Any particular position and orientation of  $P_e$  in space, and so any particular set of joint values, is called a **configuration of the robot arm**.

### 1.5.6 Workspace

It is the locus that contains the reachable space by the robot's end-point. But this is different from **workspace envelope**. The **workspace limitations**:

- Actuators endstroke.
- Working range of joints.
- Collisions among manipulator's link.

We can define the robot workspace in two ways:

1. The **Dexterous Workspace**,  $WS_D$ , is the volume of the theoretical workspace in which  $P_e$  can be oriented in any way.
2. The **Reachable Workspace**,  $WS_R$ , is the volume of the theoretical workspace to which  $P_e$  can be moved in at least one orientation.

Clearly  $WS_D$  is a subset (sub-volume) of  $WS_R$ .

It is also desirable that both  $WS_D$  and  $WS_R$  have no 'holes' in them, internal sub-volumes that are not reachable by  $P_e$ .

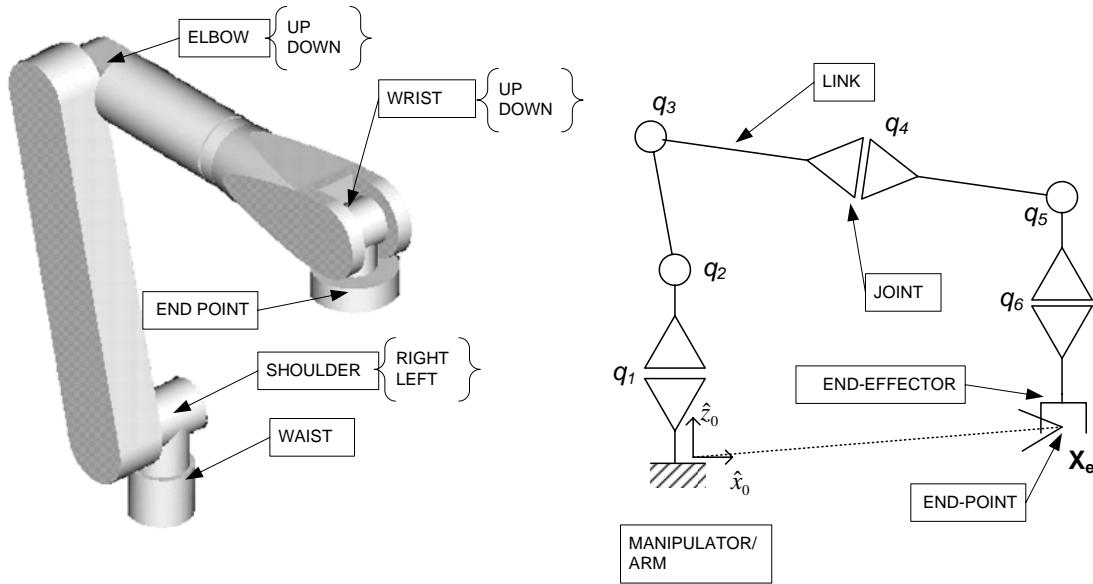


Figure 1.19: A **manipulator** and the human body **analogy** (left); schematic diagram of an arm (right).

### 1.5.7 Accuracy, Precision, Repeatability, Resolution

**Accuracy** **Accuracy** is the degree of closeness of a measured (position) compared to its true value. It is related with the **absolute error** that is given by the difference between the desired position and the actual position (see figure 1.20). In other words, it is when the robot does not locate  $P_e$  exactly where it is programmed to do so. This kind of error is often not constant over the workspace of the robot arm. It depends upon the geometry of the robot arm and other aspects of its control. Often absolute error is smaller if  $P_e$  is closer to the body of the robot, than if it is defined to be near to the limits of the reach of the arm. Defining movements to be made near to the robot can thus minimise the absolute error, but they can often also be more restricted, again depending upon the geometry of the robot arm. Arranging for most of the robot movements to be at or near the mid-range of the robots workspace, is a good way to keep the absolute error small without suffering (too many) restrictions on the kinds of movements that can be made.

**Repeatability/Precision** **Precision** or **Repeatability** or **Reproducibility** is the degree to which a repeated motion commanded to the robot produces the same or similar results in each trial (see figure 1.20). This is related to the **Repeatability error** that occurs when the robot is commanded to do the same movement repeatedly, move  $P_e$  to the same position and orientation, in other words, the same location. Usually the robot will not move  $P_e$  to exactly the  $P_e$  defined (**absolute error**) nor will it go to exactly the same location each time. There will be some sphere, of which centre is the specified (commanded)  $P_e$ , and it includes all the actual locations the robot moves  $P_e$  to each time the robot repeats the same operation. The repeatability is measured by means of the radius of that sphere.

**Resolution** When a **continuous signal** is **digitalized**, it can only reach a set of pre-established values that depends on the number of bytes of the digital system. The resolution is the minimum value between two valid positions due to the digitalization process. It depends on the number of bytes of the digital system and the robot's position as well.

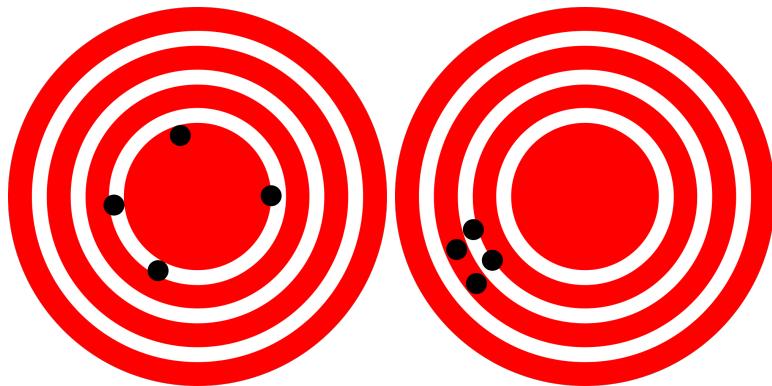


Figure 1.20: Accuracy versus precision.

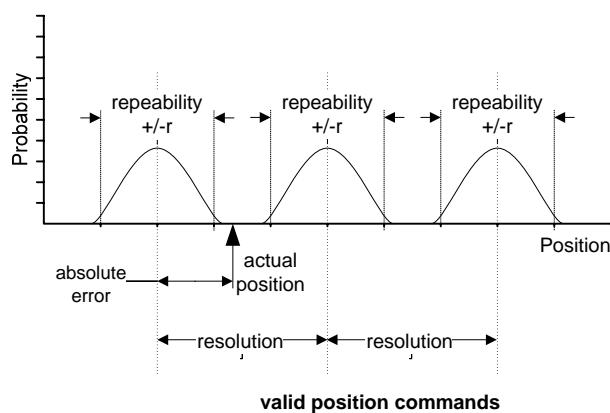


Figure 1.21: Accuracy, repeatability and resolution.

Both **absolute error** and **repeatability error** can usually be made smaller by using slower robot movements. Sometimes it is also possible to specify the **level of accuracy** required for each movement command. The higher accuracy movements usually mean that they are made with lower accelerations. Those slower movements mean that the robot program takes longer to execute. A balance between the accuracy needed and the **execution time** of the program thus needs to be established for each movement in each program.

## 1.6 Some Things to Think About

1. Identify one or two good examples of science fiction robots, toy robots, and real robots. Explain for each case why you think it is a good example. And, in the case of the real robot examples, think about how they are controlled and programmed.



## Chapter 2

# Introduction to Industrial Robotics

---

### 2.1 Some Definitions of Industrial Robots

The aim of this chapter is to provide the characteristic concepts and particularities of robotics applied to industry.

To specify in a bit more detail what kind of robot we will be considering in this course, it is useful to look at several official definitions of Industrial Robots.

#### 2.1.1 Robotics Industry Association (RIA)'s Industrial Robot definition

The first definition comes from the Robotics Industry Association (RIA) of the USA, 1979. This states that:

"An **Industrial Robot** is a multifunctional, reprogrammable **manipulator** that can move materials, pieces, tools and special devices, following variable programmed trajectories, to carry out different tasks."

Multifunctional here means that it can be used to do more than one kind of task, but not at the same time, not normally at least.

The problem with this definition is that it depends on knowing what a manipulator is. We also need to distinguish between manipulators, such as building site cranes and industrial robots.

#### 2.1.2 French Standards Association's Industrial Robot definition

Another definition, from the French Standards Association, does make the distinction between robot and manipulator. First it defines what a **manipulator** is:

"A **manipulator** is formed, in general, from a series of connected elements which can pick up and move objects or tools. It is multifunctional and can be controlled directly or via a program."

It then defines an **Industrial Robot** as follows:

"A robot is a manipulator which is reprogrammable, servocontrolled, and versatile that can position and orient pieces and tools and other devices, following variable trajectories."

Comparing these two definitions, it can be seen that there is not clear distinction between what is a manipulator and what is a robot. As example, the difference between them cannot lie in the fact that they both are programmable. However, it can be noticed that one difference lie in the fact that a robot is a kind of manipulator which are servocontrolled, whereas manipulators may or may not be servocontrolled. The term servocontrol means negative feedback control, normally of joint positions, but it could also be of joint speed. It is an old fashioned term for feedback control that is usually only used in the context of the control of mechanical systems.

Thus, according to this French definition, industrial robots are (re)programmable servocontrolled manipulators.

It further describes them as normally formed by one multi-jointed arm with a wrist at the end (to orient pieces); as having controllers endowed with program memory, and which may be able to receive signals from external sensors. It also says that they are used for highly repetitive tasks.

### **2.1.3 International Federation of Robotics (IFR)'s Industrial Robot definition**

The International Federation of Robotics gives the most detailed explanation about what an **Industrial Robot** is . . .

" . . . an automatically controlled, reprogrammable, multipurpose, manipulative machine with three or more reprogrammable axes (that can position and orient materials, pieces, tools and special devices to realise different tasks), which users may either fix in place or make mobile for industrial automation applications".

From now, and to keep the text simple, the word robot will implicitly refers to industrial robot.

## **2.2 Robot generations**

Robots can be classified in many ways based on their evolution -or generation-. Here the most known.

**First Generation** It executes the programmed task in a never-ending loop. It does not sense the disturbances in its environment.

**Second Generation** It acquires limited data from its environment and then makes simple decisions.

**Third Generation** It can be programmed using natural language.

But there are more classifications. Lets see the robot generations considered by the French Association for Industrial Robotics (AFRI)

**Type A** Manipulator with manual control.

**Type B** Automatic manipulator with preset working cycles; with end-strokes; controlled by PLC; electrical, pneumatic o hydraulic actuators.

**Type C** Programmable robot; continuous trajectory or point-to-point; without knowledge about its environment.

**Type D** The robot is able to acquire data from its environment, and thus adapts to its task.

## 2.3 Industrial Applications

The industrial robots are used in many applications, mainly for repetitive, boring and heavy tasks. The advantages of using a robot are: the safety is increased (no operators can be injured) and a robot never misses any operation and performs equally well throughout the day.

The industrial applications are divided into two groups: processing and handling.

**Processing** In these applications the robot performs some task on a piece. The robot is a working part inside the process:

- Welding: Spot welding, Arc welding, Laser Welding,...
- Surface treatment: painting, spraying, metal coating, sealing,...
- Machining: drilling, grinding, deburring,...
- Cutting: plasma cutting, acetylene cutting, oxy cutting, water jet cutting,...

**Handling** The robot only handles the parts.

- Assembly.
- Machine tending.
- Transportation.
- Palletizing.
- Packing.
- Testing.

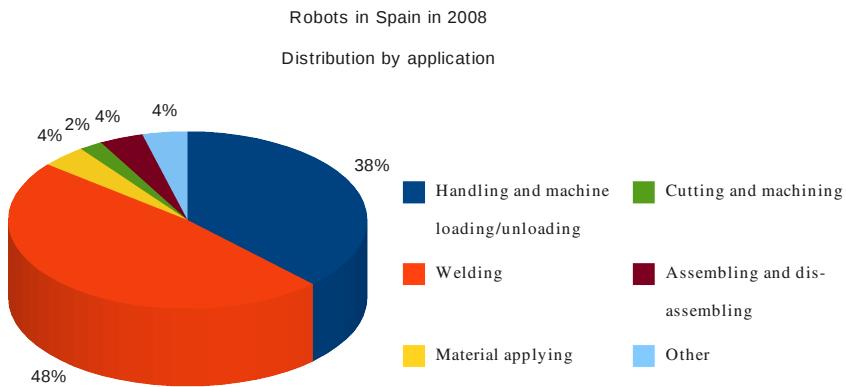


Figure 2.1: Robot applications.

However, the main applications are welding and handling. Chart 2.1 summarizes the approximated percentage.

In Spain, *welding* was the predominant application at the end of 2008 (48%). The second largest application area was *handling and machine tending* (38%). The *motor vehicle industry* is by far the largest user, accounting for as much as 64'7% (see figure 2.2). The second largest user was *rubber and plastic manufacturing* (8'96%), see figure 2.3. The metal product industry only accounted for 4'45%<sup>1</sup>.

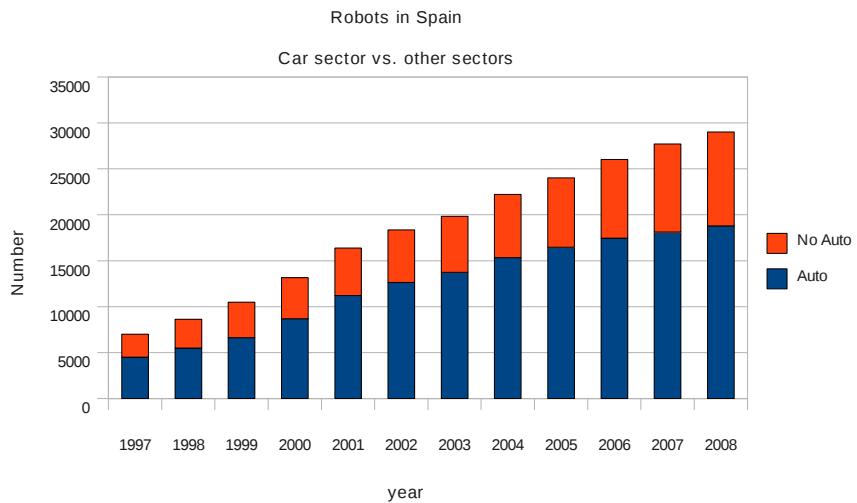


Figure 2.2: Number of robots used in car manufacturing vs other sectors.

<sup>1</sup>Sources: Estudio Estadísticas de Robótica 2009, Asociación Española de Robótica y automatización tecnologías de la producción.

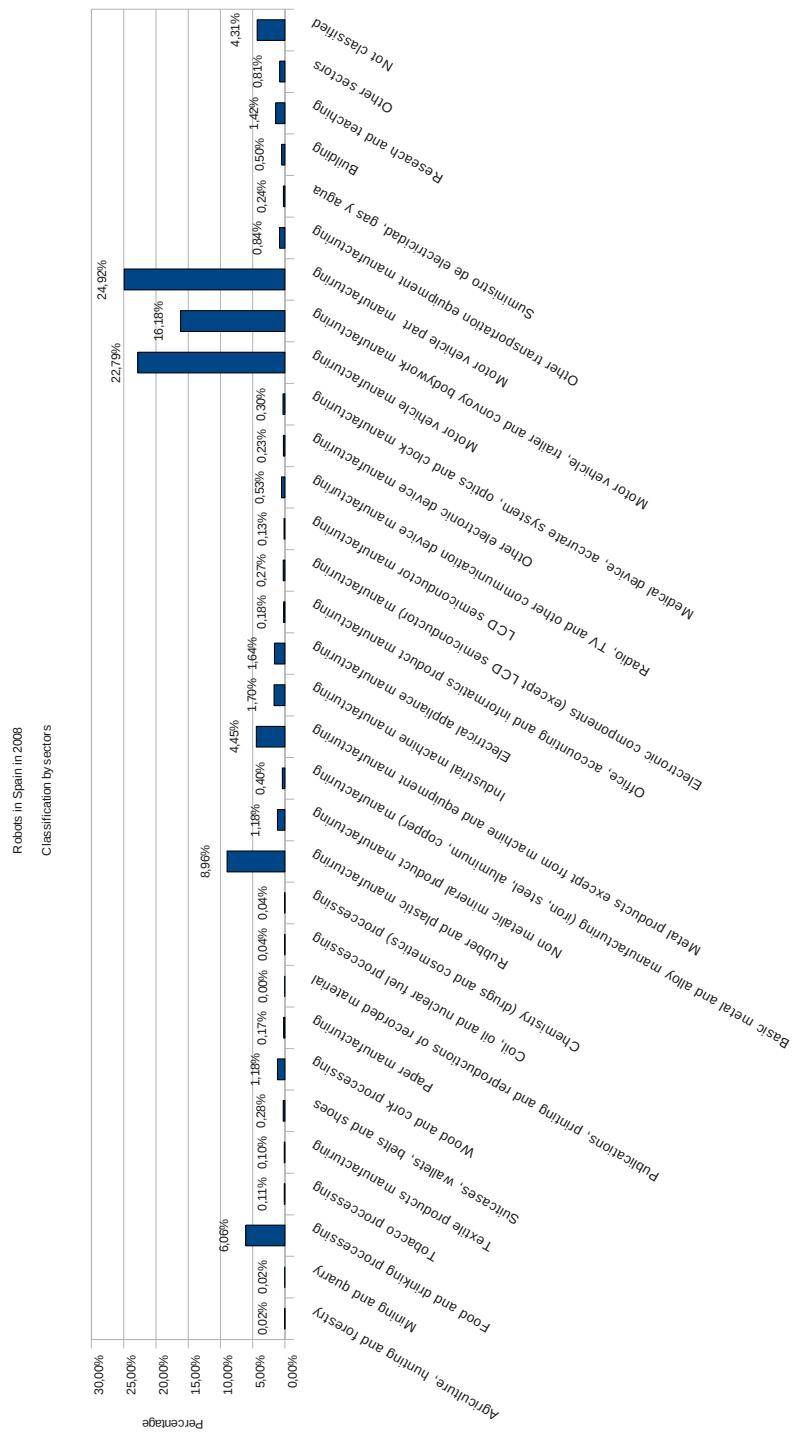


Figure 2.3: Distribution of the Spanish robots depending on economical sectors.

### 2.3.1 Welding

One of the most common uses for industrial robots is welding. Welding robots have five or even six axes, since two or three wrist axes are necessary for proper orientation of the welding tool. Welding robots are mostly used in welding car bodies, and electrical household appliances. There are two types of welding: spot welding and arc welding.

In spot welding process, the robot will follow a programmed path (trajectory) repeatedly. This process can work in two ways, depending whether the robot handles the welding tool or not:

- Fixed workpiece: The piece being welded is fixed and the robot carries the welding tool. So the robot has to have a high payload in order that it can support the weight of the tool. Moreover, a large workspace can be needed for complete the task.
- Fixed welding tool: If the workpiece is not too big. Then the robot grips and places it in the adequate position.

The welding tool is like a pincer of a crab. When it (or the workpiece) is placed in the proper location, the welding tool sparks and produces a welding spot. The precision of the robot is not too critical.

There are several types of arc welding: MIG, TIG, CO<sub>2</sub>. But all of them require that the robot handles a torch and tracks a specific path. High precision and repeatability is required in order to create a good weld.



Figure 2.4: Different examples of welding processes.

### 2.3.2 Surface Treatment and Painting/Spraying

This application is very common in robotics because a robot is highly efficient and can save up to 30% of the paint/material. It usually needs complex trajectories but it can be tracked with low precision. Robots are even more necessary in this area, because the paint (or other material concerned) to be applied could be toxic to humans. Samples of surface treatments are: sealing, coating, water cleaning, ...



Figure 2.5: Painting

### 2.3.3 Cutting/Machining

Cutting can be done by using several technologies such as plasma cutting, water jet cutting, laser cutting, oxycutting. Among the machining processes, we can mention drilling, grinding, deburring,...



Figure 2.6: Cutting.

All of them (cutting and machining) need very high precision in tracking trajectories.

### 2.3.4 Assembly

Assembly accounts for approximately 33% of the applications of the world's robot stock. Many of these robots can be found in the automotive and electronics industries.



Figure 2.7: Assembly

It could be said that assembly is an easy task for humans, but quite complex for robots. One sample is the peg-in-hole operation. In this case, it is very important to have a good sensor system. The SCARA robots are the most commonly used robots for this application.

### 2.3.5 Disassembly

This application is quite new. It is focused on dismantling goods in order to recycle.

### 2.3.6 Machine Tending/Handling

This is another interesting application for industrial robots: it is very repetitive and, in some cases, hazardous for humans. One example is furnace tending.



Figure 2.8: Handling.

### 2.3.7 Packaging/Palletizing

Packaging/palletizing is still a minor application area for industrial robots, accounting for only 4'24%. This application area is expected to grow as robots become easier to handle. The requirements are big workspace and heavy payload.



Figure 2.9: Palletizing.

## **Part II**

# **Industrial Robotics**



## Chapter 3

# Industrial Robot Programming

---

### 3.1 Introduction

When we **program**, we identify and specify a series of **basic actions** which, when executed in the specified order, achieve some **specific task** or realise some **specific process**. So, before we begin to program anything we need to clearly identify the task or process to be programmed. Another term for basic actions is operations or commands.

We need to understand how programming and control are related in the context of industrial robots. We can begin to do this by considering a kind of programming we are all familiar with: **computer programming**. When we program a computer, the basic actions depend the level at which we are programming the computer. If we are using C or C++, the basic operations will include things like addition, subtraction, division and the multiplication of numbers, reading and writing operations, etc. If we are programming using the Assembler language or even the Machine Code of the computer, then the basic operations are much more primitive, they are binary or bit string operations, operations in 8, 16, or 32 bits.

The different kinds or levels of computer programming can all be translated (**compiled** or **interpreted**) into the **Machine Code** of the computer that executes the program. We will call the **Machine Code instructions** the most basic actions of the computer.

One thing we have to notice is that the way a program is executed definitely needs some kind of control, to make sure that the right operations (basic actions) are performed in the right order, and at the right time, perhaps, as specified by our program. But this is control of the program. It is not control of the basic actions in computer programming.

In other words, in the case of the computer, no **feedback control** is used to make sure that the effects or results of the basic actions are always the same, and as they should be. This means that none of the basic actions need or have specified reference values that are used as input to a controller.

What about robots? Do the basic actions of a robot need control? As before, to answer this question we first need to ask what are the basic actions of a robot? In the context of

this course, we will consider two types of **basic robot actions**: on the one hand, some calculus as the computers; 1) typical calculations as with any computer program, and 2) robot specific operations, i.e. individual joint movements. The end-point motion of the robot is composed (in some way) of all the movements of the individual joints of the entire robot arm.

Our question then becomes, do the joint movements (the robot specific actions) need explicit feedback control? The answer is 'yes'; in general, the joint motions of an Industrial robot arm do need control. There are, however, exceptions to this, as we will see. We will also see that the kind of control needed, when it is needed. In other words, the control of robot joint movements and positions usually need the action from some controller. We will see the reasons for this when we deal with different **control algorithms** in Chapter 10.

programming a...	<table border="0" style="width: 100%;"> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;"><b>computer</b></td><td style="border-bottom: 1px solid black; padding-bottom: 5px;"><i>calculations</i></td></tr> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;"></td><td style="border-bottom: 1px solid black; padding-bottom: 5px;"><i>data input/output</i></td></tr> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;"></td><td style="border-bottom: 1px solid black; padding-bottom: 5px;">...</td></tr> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;"><b>robot</b></td><td style="border-bottom: 1px solid black; padding-bottom: 5px;"><i>same as computer</i></td></tr> <tr> <td></td><td><i>digital/analog input/output</i></td></tr> <tr> <td></td><td><i>end-effector motion</i> → needs control</td></tr> </table>	<b>computer</b>	<i>calculations</i>		<i>data input/output</i>		...	<b>robot</b>	<i>same as computer</i>		<i>digital/analog input/output</i>		<i>end-effector motion</i> → needs control
<b>computer</b>	<i>calculations</i>												
	<i>data input/output</i>												
	...												
<b>robot</b>	<i>same as computer</i>												
	<i>digital/analog input/output</i>												
	<i>end-effector motion</i> → needs control												

Table 3.1: Computer programming vs. robot programming.

In a robot program, just as in a computer program, it is usually more convenient to specify actions at a higher level than that of the most basic actions. In particular, it is better if we can specify positions, orientations, and movements of the robot end-point, rather than have to specify the positions and movements of all the individual joints.

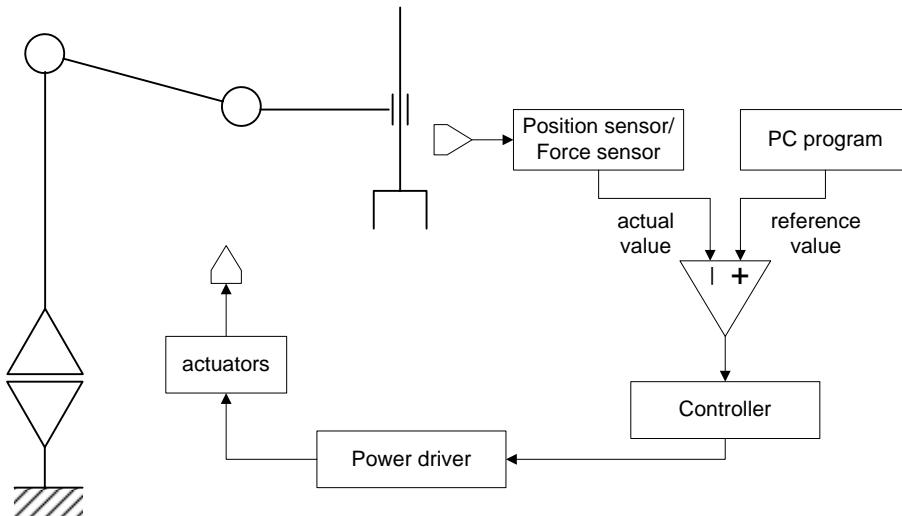


Figure 3.1: Control loop of a robot.

The execution of robot programs that specify whole-robot movements thus requires some means of composing these specified movements from the most basic actions available, individual joint positions and motions. Again, just as in computer programs, which are compiled or interpreted into the machine code instructions of the computer,

the generation of the individual joint movements needed to realize some specified whole-arm motion or some specified end-point motion or action, also needs some kind of control.

## 3.2 Levels of Robot Programming

We can distinguish four different types, or levels (as they are sometimes called) of robot programming, based on the different types of basic actions that are specified in the programs that are developed at each of these four levels. The four different levels are:

- **Low-level programming**, which can be further subdivided into:

**Joint-level programming** in which the basic actions are defined in terms of positions of the individual joints of the robot arm: joint angles in the case of rotational joints and linear positions in the case of linear or prismatic joints. At this level, basic actions can also be defined in terms of velocities of movement of individual joints, but this is unusual. Example:

```
moveTO q_ini
moveTO q_approx
openGRIPPER
moveTO q_get
closeGRIPPER
moveTO qf
```

**Robot-level programming** in which the basic actions are defined in terms of positions and orientations of  $P_e$ , in other words, in terms of the position and orientation in space of the frame of reference fixed to  $P_e$ . At this level basic actions can also be defined in terms of particular kinds of movement of  $P_e$  in space, straight lines, or circular arc movements, etc., and these definitions may also include a specification of the speed of the movement, such as a constant speed straight line movement of  $P_e$ . Example:

```
move to P1
circle to P2 by P3
line to P4
open gripper
line to P5
close gripper
line to P4
```

- High-level programming, which can be further subdivided into:

**Object-level programming** in which the basic actions are operations to be performed on the parts or objects to be manipulated or assembled, or relationships that must be established between parts. To illustrate this idea, here is a short, very simple example, which involves putting one block on top of another. Example:

```

pickup BLOCK-A by SIDE-A1 and SIDE-A3
move BLOCK-A to LOCATION-2
pickup BLOCK-B by SIDE-B1 and SIDE-B3
putdown BLOCK-B on-top-off BLOCK-A
with SIDE-A5 coplanar-with SIDE-B6 and
with SIDE-A1 coplanar-with SIDE-B1 and
with SIDE-A2 coplanar-with SIDE-B2

```

Here BLOCK-A and BLOCK-B are both the same size, with the sides numbered as 1 to 4, the top face as number 5, and the bottom face as number 6. The coplanar-with relationships specified in the last command of this short Object-level program are the three spatial relationships needed to completely locate BLOCK-B with respect to BLOCK-A in space. For this program to be a complete program that could be compiled or interpreted into actual robot commands, we would need to add geometric descriptions of the two objects involved, BLOCK-A and BLOCK-B, which include definitions of the features such as SIDE-A1, SIDE-A3, etc., together with definitions for LOCATION-2 and any other locations used.

**Task-level programming** in which the basic actions specified by the program are complete tasks or subtasks, such as:

```

put BLOCK-B on top of BLOCK-A
paint-the car-body red
assemble the gear-box

```

For these kinds of programs to work, at the Task-level, the robot system would somehow need to know about blocks, car bodies and paint colours, and gearboxes and the parts that make them up. Providing the system with this kind of information is not easy. Joint-level programming and Robot-level programming are still the most widely used kinds of programming in industrial robots. There have so far been no successful Object-level programming languages, and these remain the subject of research, mostly in University research labs. There are also no industrially used Task-level programming languages, and these too remain the subject of research.

Both Object-level and Task-level programming requires that the robot system have some kind of intelligent control.

### 3.3 Robot Programming Methods

We usually divide robot programming methods into two different kinds:

- On-line programming methods, and
- Off-line programming methods.

**On-line programming** uses the robot to be programmed to generate the program. **Off-line programming** does not need access to the robot to develop the program, at

least, not until the final testing of the program. It involves writing a program using a text-based robot programming language.

### 3.3.1 On-line Programming Methods

**Programming by guiding** or **programming by teaching**. This involves physically guiding (moving) the robot arm through the movements and actions that the robot has to perform latter, while the robot system is recording the positions (and perhaps trajectories).

#### 3.3.1.1 Point Recording vs. Trajectory Recording

If the controller only records (in its memory) static configurations of the robot arm, and not movement trajectories, it is called **point-to-point programming** (PTP), and each configuration is identified by a number or symbolic name. (Note that the term "point-to-point" is not a very good one here, since what is really recorded are the configurations of the robot arm, in other words, the positions of all the joints). Execution of the program, therefore, only involves the robot control system moving the robot arm to each of the recorded configurations in the sequence specified by the program, at some rate which may not be specified by the programmer (trajectory programming).

If the controller can record the trajectories that the end-point of the robot arm is made to move through, it is called **trajectory programming**. This is in fact similar to point-to-point programming except that many more "points" (robot configurations) are recorded. The number needed, and how far apart they are, is also determined automatically by the robot control system. It essentially samples the motion of the robot arm while the person programming it moves it in the desired way. Execution of the program is then just a matter of repeating the programmed trajectories in the order they were programmed in, again, at some speed that may or may not be specified by the programmer.

Other kinds of actions are not usually possible to program with these methods of programming, although there is usually some way of programming the opening and closing of a gripper, or the starting and stopping of a paint spray gun, or a spot welding gun, etc.

#### 3.3.1.2 Guiding Techniques

During the on-line programming, the motors of the robot can either be off or on. If the motors are off, this kind of programming is called **passive on-line programming**.

The main advantage of passive on-line programming is that it is quite easy to do, and it does not need any special programming skills or training.

The main disadvantages are:

- It is not practical for large or heavy robots;

- It cannot be used in hazardous situations;
- High accuracy and straight-line movements are difficult to achieve, as are any other kind of geometrically defined trajectory, such as circular arcs, etc.
- It is difficult to incorporate external sensor data;
- Synchronisation or coordination with other robots or other machines in the work cell is difficult.

For robots that are not too large and which are only used to do relatively simple tasks that do not involve coordination with other machines or robots, this method of programming remains the most convenient and low-cost.

In **master-slave guiding** or **guiding by dummy/mannequin**, the operator manually moves a **dummy robot arm** through the motion sequence of the work cycle. The dummy arm has the same configuration as the robot arm. The joints of the dummy arm are equipped with position sensors that are used to sample robot's motion. After recording the program the real robot arm is positioned in place of the dummy arm. The advantage of the dummy is that it is lighter than the actual robot, so the operator's fatigue is reduced. The disadvantage is the cost of the dummy and the space required for storing it.

Another guiding technique is **Teach-box Programming**. This uses what is called either a **teach-box**, or **programming pendant**, or a **teach pendant**, and **active on-line programming**.



Figure 3.2: Kawasaki's teach pedant (left); motoman's teach pedant (right).

A Teach-box (or programming pendant, or teach pendant) is a control panel connected to the robot control system which can be used to instruct the robot arm to move to particular configurations, or perform other operations, such as opening and closing the gripper fingers, or move a particular joint by a specified amount. Samples of teach pendants are shown in Figure 3.2: The left one is from a Kawasaki Robot <sup>1</sup> and the right one is from a Motoman Robot <sup>2</sup>.

<sup>1</sup><http://www.kawasakirobot.com/>

<sup>2</sup><http://www.jcdrobotics.com/>

Some teach pendants may also have a Joystick that allows the programmers to move the robot arm, either in terms of individual joint movements, or in terms of the end-point,  $P_e$ .

The Teach-box typically has a set of buttons and allows particular configurations or operations to be recorded as steps in the program. It may also be possible to store particular movements that use defined trajectories.

Using this method of programming requires that the motors of the robot are on! It is thus sometimes called active on-line programming. It also means that it is potentially dangerous, since there is a great temptation for the programmer to get close to the robot while it is being programmed to better see what is happening or to better adjust or align its gripper position and orientation, for example. This should NEVER be done! The robot must always be put into safe mode before moving into its workspace.

This method of programming is sometimes called ***Extended Teach-box programming***, if it is possible, via the Teach-box interface, to include movement speed specifications, and/or to specify other conditions of the robot movements and actions, such as the type of trajectory to use between two configurations.

### 3.3.1.3 Programming by Pasive Guiding vs. Programming by Teach Pendant

The Figure 3.3<sup>3</sup> shows clearly the difference between these two programming methods: whe using the first method, the programmer moves directly the robot arm that is not powered, and the second one, the programmer controls robot movements by the teach pendant.

All of these methods, programming by guiding and programming using a teach-box, are methods for On-line programming. Programming by guiding is essentially a method for Joint-level programming.

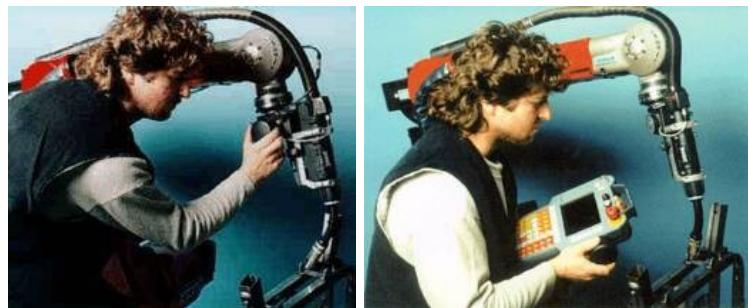


Figure 3.3: Programming by pasive guiding vs. programming by teach-pendant.

Teach-box programming is a method that usually allows both Joint-level and Robot-level programming to be mixed in the same program. It is sometimes more convenient to move just one joint, to establish some desired robot configuration, than to try to

---

<sup>3</sup><http://www.reisrobotics.de/>

command the necessary movement of the end-point. At other times, it is much easier to move the end-point,  $P_e$ , of the robot arm.

Although both these methods are still much in use, they both suffer from two important disadvantages:

- On-line programming needs access to the robot while it is being programmed. This means that the robot cannot be doing anything else while it is being programmed, and this may also mean that no other robot or related machine can be doing anything, since they depend upon each other. If there is more than one robot, each one has to be individually programmed. This means that a multi-robot production line or multi-robot system has to be taken out of productive work for a long period of time. This is expensive!
- The programs generated by on-line programming methods only exist in the memory of the robot control system, and it may not be possible to transfer these to any other system, a PC, for example, or to even print out the programs. This makes the programs difficult to document, maintain, and modify. This too introduces extra costs into the programming and re-programming of robots using On-line programming methods.

On-line programming methods can only be used for Joint-level or Robot level programming. They cannot be used for High-level programming.

### 3.3.2 Off-line Programming Methods

**Off-line robot programming** methods involve preparing a text file containing the robot instructions and other declarations that form the program the robot is to execute.

In this sense, it is much like programming a computer, in C, C++, Java, or some other programming language. When programming, it is recommended to use good programming and software engineering practices. However, some robot programming interfaces are so simple that they allow to write code in a way that the programs will be hard to maintain. For example, many robot programming languages still have goto statements, a well-known source of bugs and errors in any kind of program. The programming environments, that support the program development process, are also typically somewhat primitive in comparison to modern computer programming environments.

As in computer programs, the program text file specifies the sequence of the basic actions to be performed, together with other information needed to control the maximum speed of movement (of the end-point), or types of trajectories to follow between two end-point configurations.

The main advantages of Off-line programming are:

- Programs can be developed without needing to use the robot,

- The sequence of operations and robot movements can be optimised or easily improved, once the basic program has been developed,
- Previously developed and tested procedures and **subroutines** can be re-used,
- External sensor data can be incorporated, though this typically makes the programs more complicated, and so more difficult to modify and maintain, but more powerful,
- Programs can be tested and evaluated using simulation techniques, though this can never obviate the need to do final testing of the program using the real robot;
- Programs can more easily be maintained and modified; and
- Programs can be properly documented and commented, at least more easily.

Off-line programming can be used to develop Robot-level programs and High-level programs. It is possible to use Off-line methods for Joint-level programming, but it is very seldomly done in practice.

### 3.3.3 Robot Programming Architecture

The relationship between On-line and Off-line programming and the robot control loop in an Industrial Robot system is illustrated below.

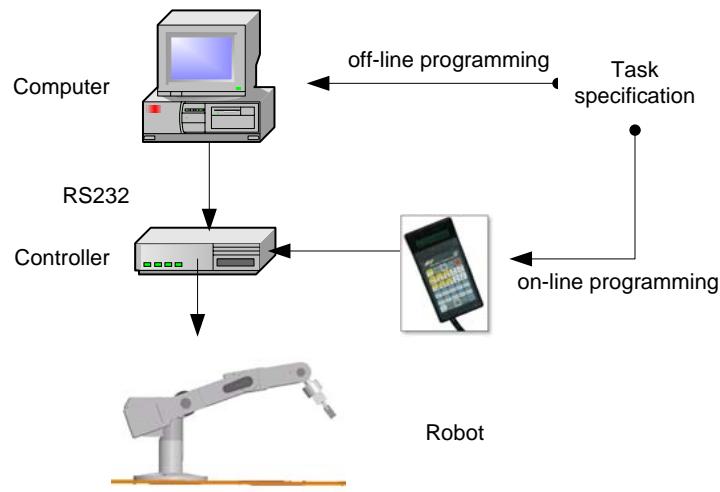


Figure 3.4: Relationship between On-line and Off-line programming.

## 3.4 Examples of Robot Programming Languages

Some examples of Robot Programming languages are:

- Robot-Level programming languages:

- AL, University of Stanford, USA (1974);
- AML, IBM (1982);
- LM; University of Grenoble (1989);
- VAL-II, Univation (1983);
- V+, Adept (1989);
- RAPID, ABB (1994);
- ARLA, ASEA (1979).
- Object-Level programming languages:
  - LAMA, MIT AI Lab (1976);
  - AUTOPASS, IBM (1977);
  - RAPT, University of Edinburgh (1978-86).

### 3.5 Robot programming Language Architecture

This section is focused on depicting the structure of robot programming languages. Many similarities with respect to computer programming languages can be found:

- Variables: integers, floats, booleans, characters, arrays, etc...
- Flow control: for loop, while loop, do...while loop, if...else if...else statements, etc...
- Input/output: keyboard/ screen
- Execution policies & synchronization: processes, threads, interrupts/triggers, wait statements, events, etc...
- Operating system utilities: directory, copy/delete/move, help, etc...

However, we can find another commands that are specific for robot programming languages:

- Points: stored in cartesian/joint space
- Movement control/ speed control/trajectory profile
- Gripper: open/close
- Input/Output: sensors/actuators (both of them can be analog or digital)

Now, it is time for the design phase of a program. During this phase, the programmer can help himself by means of pseudocode and flowcharts. The **pseudocode** is a kind of robot program language but invented by the programmer. Samples of this type of code can be seen at thissection . However, the programmer would like to 'plot' his

program in order to have a graphical representation of how it works. This can be done by **flowcharts**. So a flowchart is a collection of symbols, connected by arrows, which show the relationship among the different commands in the program: when and in which order they are executed. A sample of these symbols is given below.

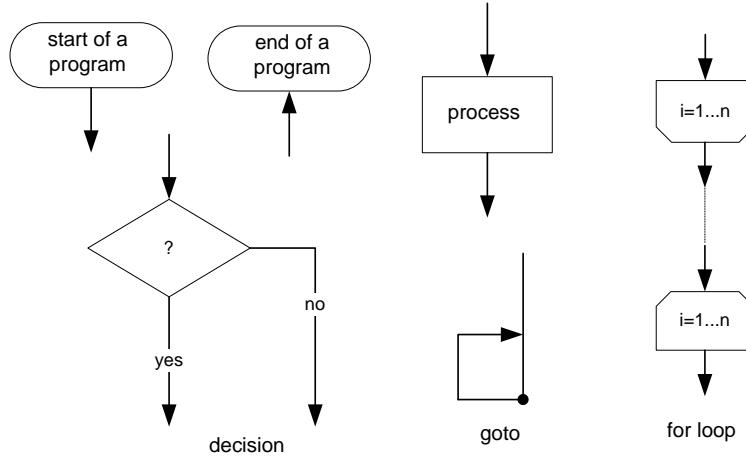


Figure 3.5: Common symbols used in flowcharts.

## 3.6 Robot Program Development Process: the six steps

In this section we are looking at the six steps in the process of developing good and reliable robot programs. The Basic Steps are:

- Step 1: Analyse and decompose the task into a series of operations on the objects involved, and specify the order in which the operations must be executed.
- Step 2: Identify and specify all the situations needed to program all the movements and actions of the robot.
- Step 3: Identify any type of repeated actions and specify them as **subroutines** with parameters.
- Step 4: Design and develop the complete robot program and its documentation.
- Step 5: Test and debug the program using a simulator of the robot and its workspace.
- Step 6: Test the program on the real robot.

To illustrate the program development process with a concrete (though simple) example, we will use the assembly task defined in the figure below.

**Task:** Move Block A and Block B to form a tower on top of Block C, in the order C, A, B, from the bottom. They are all the same sizes. All the Blocks must be well aligned in the final tower assembly.

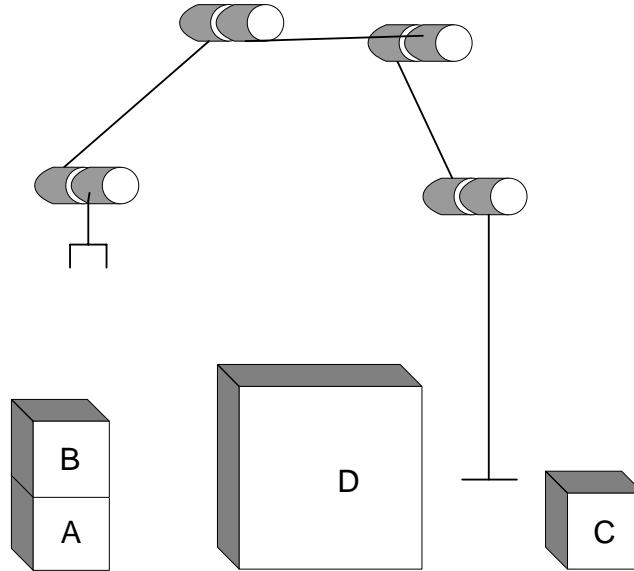


Figure 3.6: A simple assembly task.

### **3.6.1 Step 1: Analyze and decompose the task into a series of operations on the objects involved**

The goal of this first step is to analyse and decompose the task into a series of operations on the objects involved, and specify the order in which the operations must be executed.

For example, to perform the task defined above we might decompose the task into the following sequence of operations on the three Blocks (**pseudocode**):

```
Remove Block-B from on top of Block-A,  
Put block-B down by the side of Block-A;  
Put Block-A on top of Block-C;  
Put Block-B on top of Block-A;  
Align Block-A with Block-C;  
Align Block-B with Block-A
```

The three operations for block placement are obviously necessary. The two alignment operations are needed to make sure that each block is correctly aligned with the one below after it has been put down. Just putting one block down on top of another typically is not enough to achieve accurate alignment, so these extra operations are needed.

Knowing that these two alignment operations will be needed is a matter of experience, and this illustrates the fact that each step of this programming process needs good experience of the whole process before it can be done well. The sequence of five operations defined above is not the only possibility here, nor is it the best one! For example, it is probably better to align Block-A with Block-C immediately after Block-A has been put on top of Block-C, since the robot is already at Block-C after doing this.

This, of course, depends upon there being no change to this alignment when Block-B is put on top of Block-A. If this is done slowly, this will probably be the case. But we can define the sequence of operations by a flowchart:

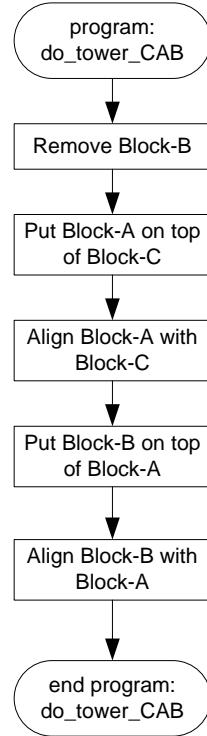


Figure 3.7: Flowchart sample.

If the fingers of the gripper are wide enough, and if the robot has sufficient degrees of freedom, another possibility is to pick up Block-A and Block-B both together -from the side, rather than from the top- and to put them on top of Block-C directly. This would significantly reduce the total number of operations needed, and so reduce the execution time of the program, but it is also a more risky kind of operation to do, it may be more likely to fail.

If moving Block-A and Block-B together is considered not safe enough, or not possible with the gripper the robot has, or the robot does not have sufficient degrees of freedom, another possibility is to move Block-C. There is nothing in the task definition that says that this is not possible!

Moving Block-C to be near Block-A means that all the operations on Block-A and Block-B involve shorter trajectories, they will therefore be quicker. Nor will they need to pass over the big Block-D, which also means there is less risk of hitting it or other things on the way.

This illustrates that robot tasks are not usually completely specified. It is then the responsibility of the programmer to identify and consider all the possibilities for how the task may be completed, what operations will be needed in each case, and how reliable and efficient (in time) each one may be. It is very important that all these aspects are

properly considered at this stage of the program development process. Changing the way the task is to be done later on is always more expensive and more prone to errors.

### **3.6.2 Step 2: Identify and specify all the situations needed to program all the movements and actions of the robot.**

A Situation is a particular configuration of the robot arm and the distribution of elements in the robot workspace, where the parts currently are and how they are oriented. A configuration of the robot is defined by the position and orientation of the frame of reference attached to the end-point,  $P_e$ , of the robot, or by a set of all the joint positions of the robot arm. So, we might specify the following situations in the case of our example assembly task:

<b>Location name</b>	<b>Description</b>
$P_0$	The initial configuration of the robot and location of the parts.
$P_{A0}$	A situation in which $P_e$ coincides with the centre of Block-A, used to define the <b>grasp</b> location of $P_e$ needed for taking Block-A in the gripper, in its initial location.
$P_{B0}$	A situation in which $P_e$ coincides with the centre of Block-B, used to define the <b>grasp</b> location of $P_e$ needed for taking Block-B in the gripper, in its initial location.
$P_{C0}$	A situation in which $P_e$ coincides with the centre of Block-C, used to define the <b>grasp</b> location of $P_e$ needed for taking Block-C in the gripper, in its initial location).
$P_1$	A situation in which $P_e$ is vertically above the centre of the top face of Block-B (in its initial location, $P_{B0}$ ), from which the robot can perform a reliable <b>get-part</b> action on Block-B
$P_2$	A situation in which $P_e$ is vertically above the centre of the top face of Block-A (in its initial location, $P_{A0}$ ), from which the robot can perform a reliable <b>get-part</b> action on Block-A
$P_{B1}$	A situation in which $P_e$ is to one side of Block-A, at a height equal to half the height of Block-A, and at a sufficient distance from Block-A for the robot to have room to put down Block-B, used to define the <b>put-down</b> action of Block-B after it has been taken off the top of Block-A.
$P_3$	A situation in which $P_e$ is vertically above $P_{B1}$ , the centre of the top face of Block-B (in its intermediate location), from which the robot can perform a reliable <b>get-part</b> action on Block-B.
$P_{A1}$	A situation in which $P_e$ coincides with the final location of Block-A, used to define the <b>put-down</b> action of Block-A, on the top of Block-C.
$P_4$	A situation in which $P_e$ is vertically above $P_{A1}$ , the final location for Block-A, in which the robot can perform a reliable <b>put-down</b> action on Block-A.
$P_{B2}$	A situation in which $P_e$ coincides with the final location of Block-B, used to define the <b>put-down</b> action of Block-B, on the top of Block-B.
$P_5$	A situation in which $P_e$ is vertically above $P_{B2}$ , the final location for Block-B, in which the robot can perform a reliable <b>put-down</b> action on Block-B.
$P_6$	A situation in which $P_e$ is above Block-D, and the parts are in any location, used as an intermediate situation in the definition of <b>collision free trajectories</b> from one side of the table to the other.

Table 3.2: Specification of the positions needed in the program.

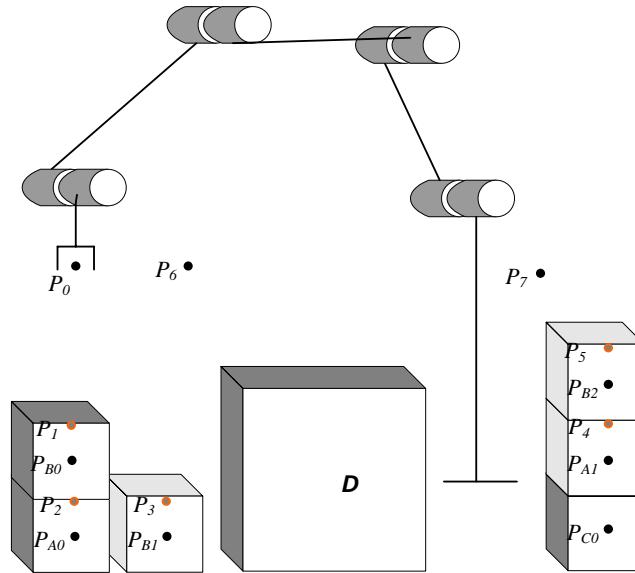


Figure 3.8: Definition of a simple assembly task.

It is a good idea to try to identify and specify situations that can be used to program different robot actions.  $P_6$  is an example of this: it is used to define a ***via-point*** for all the robot motions that have to pass over Block-D, a location through which the trajectory of the motion of  $P_e$  must pass. The end-point of the robot has to pass through a via-point (ex.  $P_2$ ) to approach, or move away from, a ***target-point*** (ex.  $P_{A0}$ ) when we want to grasp, or leave, a workpiece at this target point. The situations of a via-point is always vertically above the related target-point. When the robot has to move parts/workpieces large distances, it can go fast. But when some put-down or get-part action must be taken, the displacements from, or to, the ***approximation-point*** and to, or from, the target point are done very slowly in order to get more accurate positioning.

Considerable care must also be taken in defining situations such as  $P_{B1}$ , used to define the put-down action for Block-B. It is best to add a very small amount of height to the location of end-point defined by this situation so that the robot actually lets go of Block-B when it is a very small distance above the table, but not actually touching it. When the robot opens its gripper, Block-B will then fall a very small distance onto the table.

We do this because there is always some error in the real position and orientation of  $P_e$ , and if we define  $P_{B1}$  to locate  $P_e$  exactly half the height of Block-B above the table, there is a chance that when the robot actually performs this action, it will try to move Block-B further down than the table will allow, due to a small error in the real location of  $P_e$ . This will result in a reaction force from the table on the robot, and if this is too large, the robot will detect this as a surge in motor current demanded by its position controllers, and typically stop. The definitions of the ***situations*** thus need to take into account the possible consequences or effects of the small errors that always occur in the robot actions.

### 3.6.3 Step 3: Identify any types of repeated actions and specify them as subroutines with parameters.

Just as in any kind of computer program, if the robot needs to perform the same sequence of movements and actions at different points in a program, it should use the same program code to do this. This is best done by defining **subroutines**. This both significantly improves the maintainability of the program and reduces the probability of programming errors.

In our example task, we can identify block grasping, a get operation, and block placement, a put operation, as both requiring similar sequences of movements, and that are needed several times. So, for example, we might define a **subroutine**, called Get-Put, for both picking up a block and for putting a block down from or in some specified location. This **subroutine** takes three arguments  $Q_{init}$ , which defines the location of  $P_e$  at the start of the pick-up or put-down operation,  $Q_{getput}$ , which defines the location of  $P_e$  when the robot either closes the gripper fingers to get a part or opens them to put down a part, and Action, which defines whether it is a pick-up or put-down action that is to be performed.

```
DefProc GET-PUT(Q_init, Q_getput, Action)
{
    If Current_Situation = Q_init Do
    {
        If Action="GET" Do Open Gripper
        Else Do
        {
            If Action="PUT" Do
            {
                If Gripper=NOT CLOSED Do
                    FAIL with Write(Nothing in gripper!);
                }
                Else Do FAIL with Write(Action specified not known!);
            }
            Move_to Q_getput with Speed=LOW and Accuracy=HIGH;
            If Action="GET" Do Close Gripper
            Else Do
                If Action="PUT" Do Open Gripper;
            Move_to Q_init with Speed=LOW and Accuracy=HIGH;
        }
        Else Do FAIL with Write(Robot NOT currently at Q_init!);
    }
    EndDef;
```

Figure 3.9: Get-Put subroutine, written using pseudocode.

As we can see from the code for this subroutine, most of it is to test the validity of the values of the parameters; to make sure that the get or put action can really be

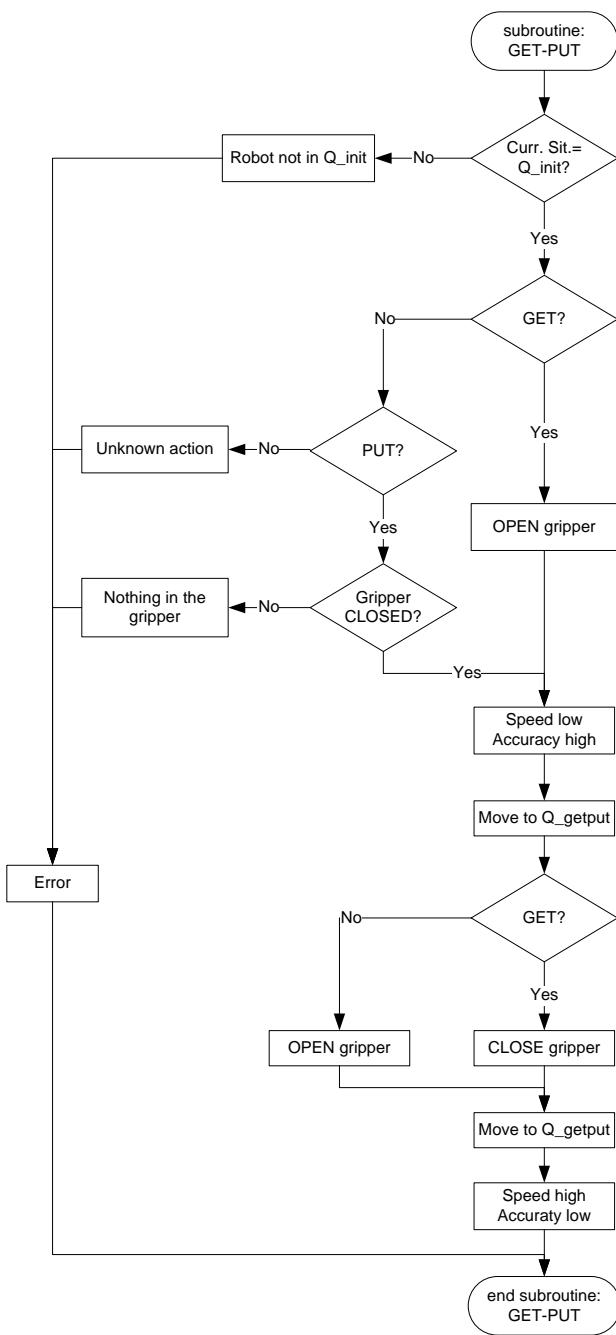


Figure 3.10: Flowchart of the Get-Put subroutine.

performed.

The Move to action is also defined to be a low speed high accuracy movement. Other large motions, to bring  $P_e$  to  $Q_{init}$ , for example, can be made with faster lower accuracy movements.

This subroutine is not completely defined here, since it also needs proper commenting and documentation, and testing!

### **3.6.4 Step 4: Design and develop the complete robot program and its documentation.**

Even after the operations, situations, and any subroutines have been specified, developed, and tested, there is usually still a lot more work to be done, and decisions to be made, to develop and document a complete robot program. This development, and the decisions made, needs to be consistent with the decisions and results of the previous three steps.

### **3.6.5 Step 5: Test and debug the program using a simulator of the robot and its workspace.**

Most modern off-line robot programming environments provide simulators that can be used to model and to simulate the robot to be programmed, as well as any other equipment and objects in the workspace of the robot.

Using a simulator can make testing and debugging a program much easier, but this benefit is only gained if the simulator is based upon a good model of the actual robot and its workspace and contents. Developing a good simulation model can often be a difficult and time consuming process in itself.

If you are not, or were not, the person who developed the robot simulator model you use, it is very important that you know or find out how good a model it is of the real robot or robots you are programming. This step obviously only applies to Off-line programming methods.

### **3.6.6 Step 6: Test the program on the real robot.**

After a program has been developed, debugged, and tested using a simulator, it must be tested on the real robot! A simulator can never be a sufficient test of a robot program.

If, as is often the case, you are developing a program to be executed by more than one robot, for example, there may be ten robots in the production line which all have to do the same task at the same time, then you will need to decide on how many of these robots you will need to test the program on.

Testing the program on only one of them probably is not enough. Testing it on all of them may be more than is really needed. In order to be able to decide, you will need to

know a lot about the history and performance of each of the robots you are programming. The less you know about the robots being programmed, the more robots you will need to test the program on.

Usually you can divide a group of robots (ten, in this example) into subgroups, and test the program on one robot from each subgroup. Subgroups may be defined in terms of the age or number of operating hours of the robots, and/or the frequency and type of maintenance they have received, and/or the number and type of failures they have suffered.

As we have seen, the complete robot programming process, involves many decisions and choices, and each one has consequences and implications for the subsequent steps. It is therefore very important to properly and adequately record and document all these decisions and choices, together with the reasons for them, during the program development process. The design and rational of the program needs to be recorded for others to be able to use them. Doing this, just like documenting the program, and doing it well, is a matter of good professional practice. Failure to do it, or do to it well enough could result in expensive failures or accidents, or unnecessary production costs.

### **3.7 Some Things to Think About**

1. In what important ways is programming a robot different from programming a computer?
2. Why are robot programming languages and robot programming environments not as sophisticated as programming languages and programming environment for computers, C++ and Java on PCs, for example?
3. How can you know when a robot program is working properly?



## Chapter 4

# Robot Geometry

---

### 4.1 Dimensions and Degrees of Freedom (DoF)

In this part of the course we will consider the different robot geometries that can be obtained by connecting different numbers of rigid elements with different numbers and types of joints. The relationship between the geometry of a robot and its **workspace** will then be introduced.

We will begin by considering Dimensions and **Degrees of Freedom** (DoF).

There are six spatial dimensions and we can associate a **Degree of Freedom** with each one. There are three dimensions of position in space, plus three dimensions of orientation in space, giving six in total. Thus any object has, in principle, up to six **Degrees of Freedom** or **Degrees of Movement**.

Industrial robot manipulators, like other machines, must have some number of **DoFs** in order to be able to move and orient objects or other tools, to be able to position and orient the end point  $P_e$ .

An important geometric property of the real world is that...

- ... to completely position anything in 3D space we only need linear DoFs, and ...
- ... to completely orient anything in 3D space we only need rotational DoFs

So, the definition of the position of an object in space does not depend on its orientation, and the definition of the orientation of an object in space, does not depend upon its position.

Thus, the position and orientation of an object in the real world are independent properties. This is an important and remarkable property that we will make much use of here.

An important consequence of this independence of position and orientation is that the orientation of an object can be completely defined with respect to a local frame of reference, whereas, the position of the same object needs a global frame of reference.

In 3D space, we thus need:

- 3 coordinates to completely define position, and
- 3 coordinates to completely define orientation, giving a total of 6 coordinates in total.

In 2D space, we need two coordinates of position and one of orientation, giving a total of three in total, and in 1D space, we need one coordinate of position and one of orientation, giving a total of two.

Thus, to position and orient (and move) objects or tools etc., in 3D space, without (theoretical) restrictions, a robot manipulator needs at least 6 DoFs (3 DoFs of position and 3 DoFs of orientation).

In reality, many industrial robot manipulators have 4, 5 or 6 DoFs. Less than 6 DoFs is usually enough to perform most of the industrial tasks.

The first three DoFs guarantee a correct positioning and the followings are used to orient.

Robot arms that have fewer DoFs are common, are easier to control, and are generally cheaper and more accurate, but they may not be easier to program.

Sometimes robots have more than 6 DoFs. They are called ***redundant robots***. Three reasons for having an extra/redundant DoF can be:

**Obstacle avoidance** : when the robot manipulator has to work in space occupied by other objects or equipment, or in a constrained space, having an extra DoF allows the robot to position and orient its end-point,  $P_e$ , while also avoiding contact or collisions with other things in its space, which it would not be able to do with only 6 DoFs.

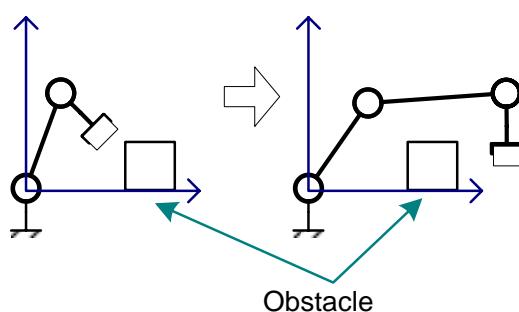
**Increasing workspace** : an extra DoF can also be used to significantly increase the volume of the accessible workspace of the robot.

**Avoiding Singularities** : a singular position is a ***robot configuration*** in which its kinematics/dynamics performance become poor. A more detail description of ***singularities*** is detailed latter. An extra DoF can also be used to avoid singularities.

## 4.2 Types of Joints

A ***joint*** is needed for each DoF that a robot manipulator has. We will consider as a ***joint***, a mechanism that provides the type and amount of movement needed to realise the particular DoF.

There are two types of DoFs in the 3D space of the real world:



... to avoid an obstacle in a 2D space

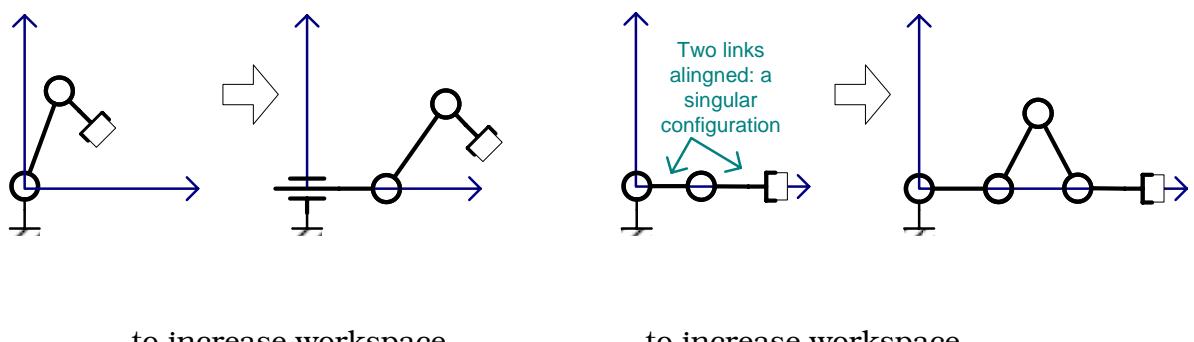


Figure 4.1: Reasons to use extra DoFs or redundant robots.

- Linear or displacement DoF, also known as translational or prismatic DoF (P).
- Rotational or orientation DoF (R).



Figure 4.2: Basic types of DoFs.

There are seven different basic types of joint that could be used, at least in principle, to form a robot manipulator's arm. These are:

**Spherical joint**, which has 3 DoFs, all rotational (RRR).

**Universal joint**, which has 2 DoFs, all rotational (RR).

**Cylindrical joint**, which has 2 DoFs, one translational and one rotational (PR).

**Planar joint**, which has 2 DoFs, one translational and one rotational (PR).

**Prismatic joint**, which has one translational DoF (P).

**Rotational joint**, which has one rotational DoF (R).

**Screw joint**, which has one translational DoF (or, it is better to say, it is a transmission mechanism that converts one rotational DoF into a translational DoF).

To produce actual movement in the directions of the DoFs in a specific joint, each DoF needs to be motorized and usually somehow controlled. In practice, it is difficult to motorize more than one DoF in a single joint: it is possible, but expensive. As a consequence, robot manipulators mostly use types of joints that have only one DoF. Or, if they use joints with more than one DoF, only one DoF is motorized and controlled, the other DoFs are left free, and are usually called the **passive DoFs** of a joint.

Almost all of the robots we will consider in this course will use single DoF joints that are motorized and controlled.<sup>1</sup>

A **prismatic joint** is robust, controlled easily, but a **rotational joint** has a bigger **workspace** and, from a technological point of view, is easier to build.

---

<sup>1</sup>Note that this is not like the joints in animals, which typically have multiple DoF all of which are motorized by muscles and controlled by the nervous system.

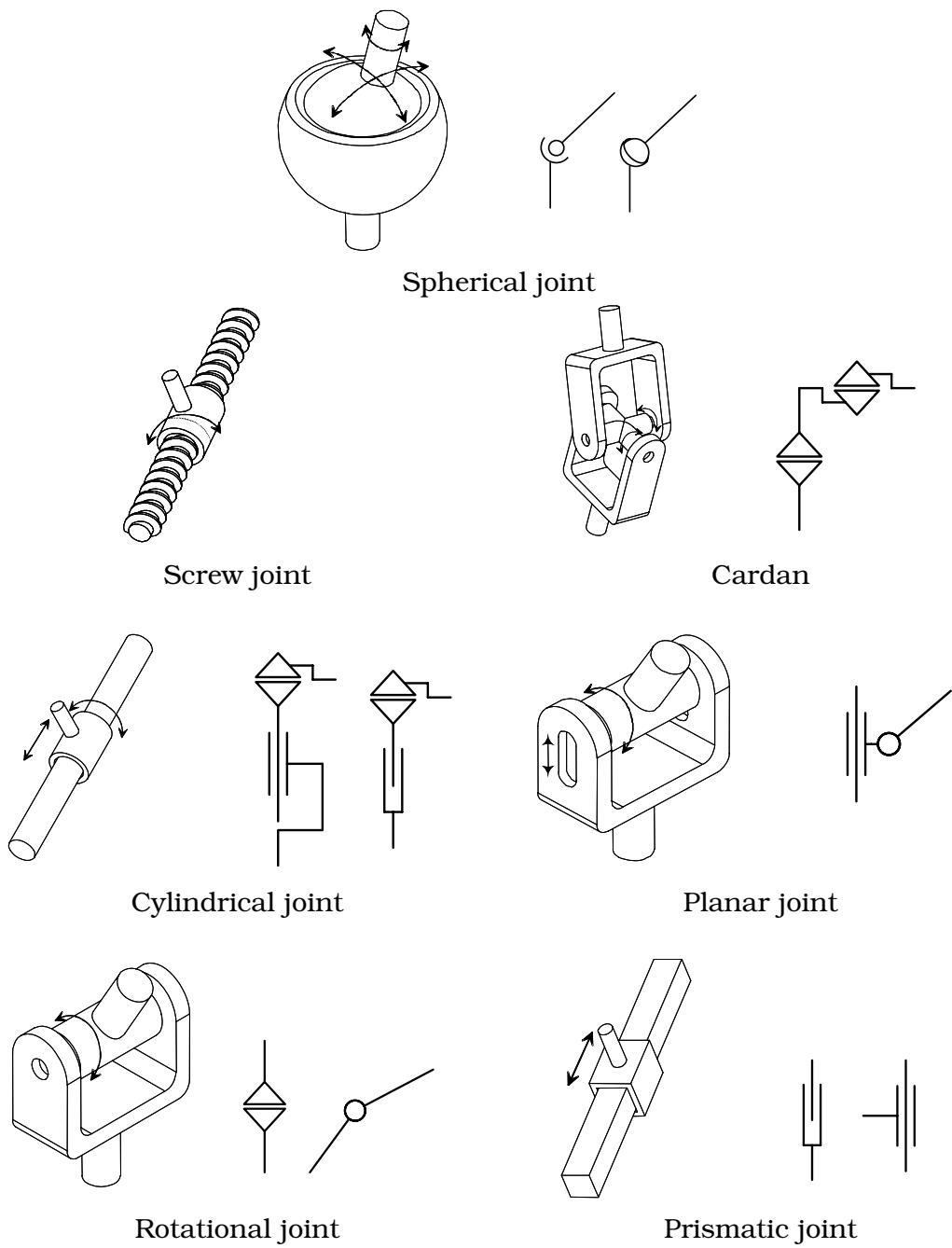


Figure 4.3: Types of joints

## 4.3 The Geometry of Robot Manipulators

Robot manipulators are formed from combinations of rigid elements, called **links**, connected by **joints** that allow two connected links to move relative to each other in one direction, or possibly more than one direction.

By using different types of joints to connect pairs of links, it is possible to construct robot manipulators that have different basic geometries with different numbers of DoFs.

There are two different ways to combine links and joints to form robot manipulators (and other mechanisms): in series, or in parallel.

In the case of manipulator, geometries formed of **chains** of links and joints, i.e., serial geometries, and given the independence of the degrees of freedom needed for translation (position) and orientation, it is conventional to dedicate the first three (or perhaps two) DoFs to the positioning of  $P_e$  in space, the **positioner**, and to dedicate the last three (or perhaps two or one, in the case of robots with 5 and 4 DoFs), to the orientation of  $P_e$  in space, the **wrist**.

Strictly speaking, we say a robot is a **serial robot** when its **positioner** is composed of an **open kinematic chain**, and a **parallel robot** when its **positioner** is composed by a **closed kinematic chain**.

### 4.3.1 Serial Geometries

Serial combinations are the most common, but we will consider parallel combinations at the end of this chapter.

The most common serial geometries for robot manipulators with 3 DoFs for positioning  $P_e$  have the following forms:

- The Cartesian robot,
- The Cylindrical robot,
- The Spherical or Polar robot,
- The Angular or Anthropomorphic robot, and
- The SCARA robot

#### 4.3.1.1 The Cartesian robot

It has three prismatic joints, PPP. They are aligned with the Cartesian axis (figure 4.4).

Advantages:

- It can be programmed and controlled easily.

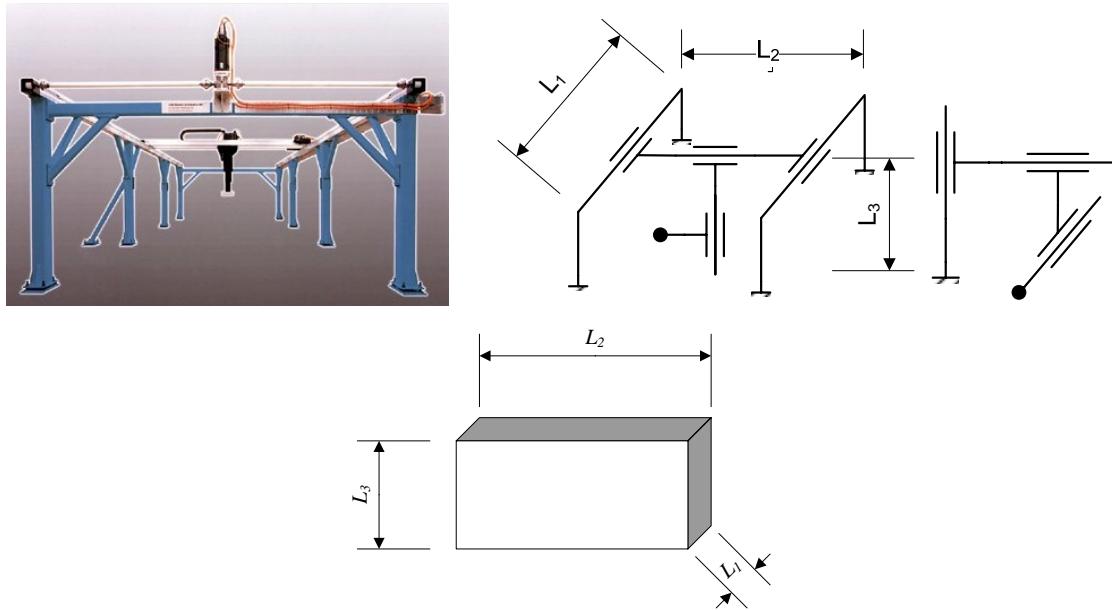


Figure 4.4: Cartesian robot

- High accuracy in linear movements can be obtained.
- It has a simple kinematics model.
- There are no singular configurations.
- Its parameters (stiffness, accuracy, repeatability, etc) are constant within its workspace.
- It can be built using inexpensive pneumatic actuators for pick and place operations.

Disadvantages:

- Its accuracy is low in tracking curved trajectories.
- The ratio between workspace and volume occupied is low.
- The robot's accessibility is low (unable to reach areas under objects).
- The maintenance of linear actuators is expensive.

#### 4.3.1.2 The Cylindrical robot

It has a rotational joint and two prismatic joints, RPP. To obtain this robot, we can start from a cartesian robot in which the first prismatic joint is replaced by a rotational joint. That is to say, the robot's DoFs are aligned with a cylindrical coordinate system (figure 4.5).

Advantages:

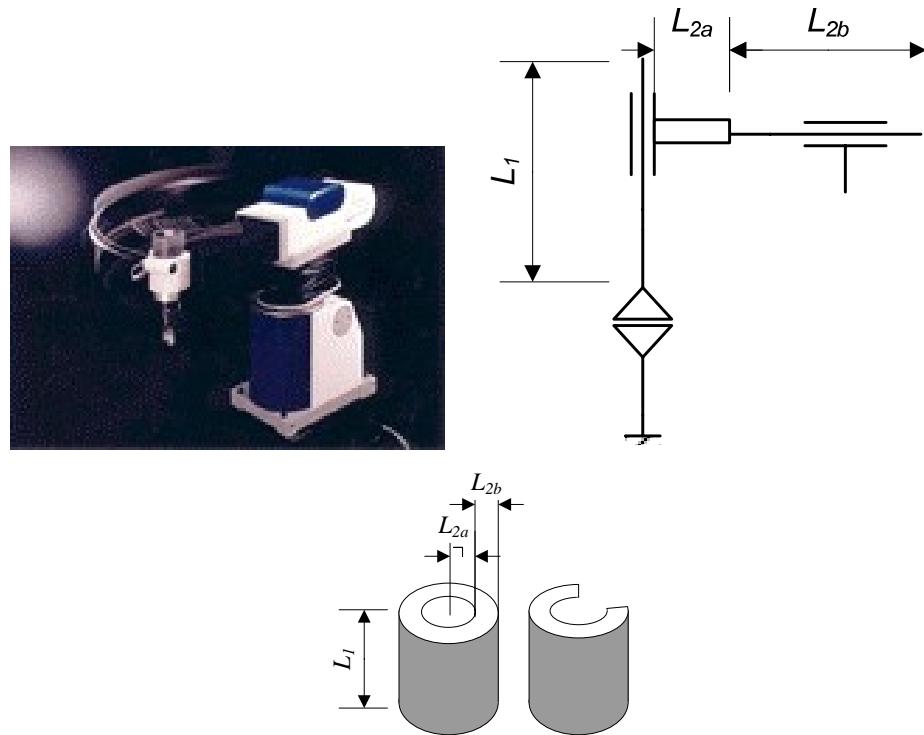


Figure 4.5: Cylindrical robot

- It can be programmed and controlled easily if cylindrical coordinates are used.
- High accuracy in linear movements can be obtained.
- It has a simple kinematics model.
- It is suitable when there are no obstacles facing it and where other machines are in a radial arrangement.
- The workspace is bigger than the workspace of cartesian robots.

Disadvantages:

- Its accuracy is low in tracking curved trajectories, but better than the cartesian robot.
- The robot's accessibility is not quite large.
- The maintenance of linear actuator is expensive.

#### 4.3.1.3 The Spherical or Polar robot

It has two rotational joints and a prismatic joint, RRP. They are aligned with the Spherical coordinate system axis (figure 4.6).

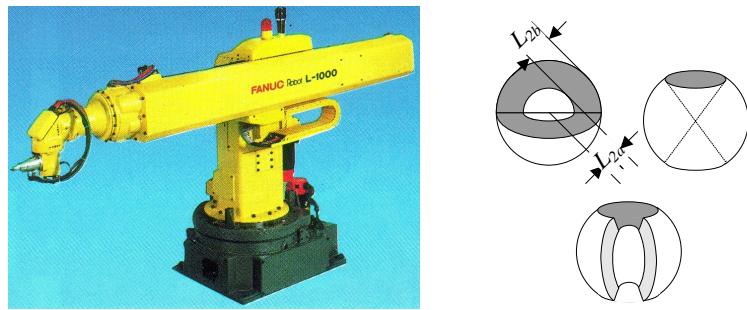


Figure 4.6: Spherical robot.

Advantages:

- It can be programmed and controlled easily if spherical coordinates are used.
- The workspace is big compared to the workspace of cylindrical and cartesian robots.
- It can pick up parts from the floor.
- It is suitable when there are no obstacles facing it and where other machines are in a radial arrangement.

Disadvantages:

- The kinematics is complex.
- The maintenance of the linear actuator is expensive.

#### 4.3.1.4 The Angular or Anthropomorphic robot

It has three rotational joints, RRR. Its appearance is similar to a human arm. The theoretical workspace is given by figure 4.7

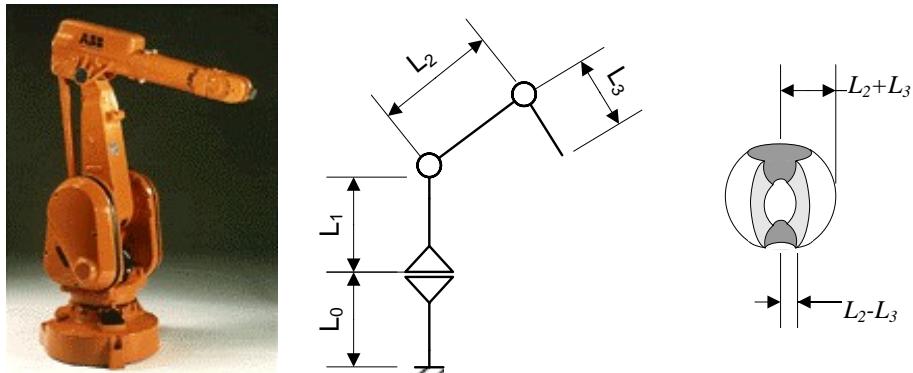


Figure 4.7: Anthropomorphic robot.

Advantages:

- It has high dexterity and great manoeuvrability. It can access zones behind obstacles.
- It can follow complex trajectories.
- It covers a large workspace compared to the small volume of this type of robots.
- The rotational joints are maintained easily.
- It can pick up parts from the floor.
- This is the most commonly used robot in industrial applications.

Disadvantages:

- The kinematics is very complex.
- The control of linear movements is difficult.
- The robot can reach many singular configurations.

#### 4.3.1.5 The SCARA robot.

It has two rotational joints and a prismatic joint, RRP. That is to say, the same DoFs as those of Cylindrical robots, but with a different arrangement. SCARA stands for **Selective Compliance Assembly Robot Arm**. This geometry was invented by Professor Makino in Japan, in 1982, and is specifically designed for performing vertical action (or top-down) assembly operations, such as "chip stuffing". The theoretical workspace is given by figure 4.8.

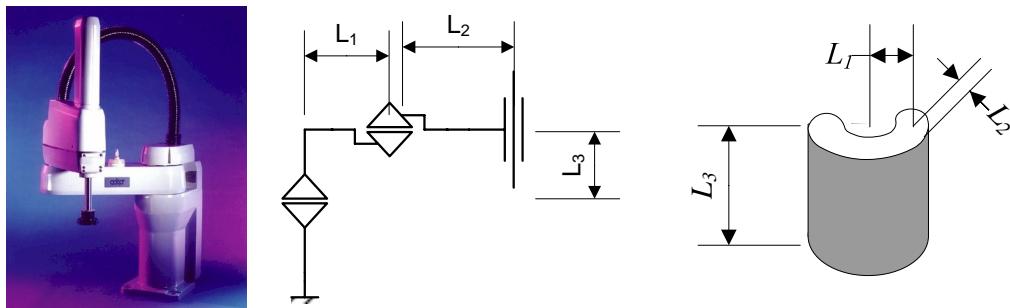


Figure 4.8: Scara robot

Advantages:

- The kinematics is simple.
- The repeatability is high.
- It is well adapted for tasks that must be undertaken on a flat horizontal surface.

Disadvantages:

- It is only suitable for tasks on horizontal surfaces.

### 4.3.2 Parallel Geometries

Serial manipulators usually have big workspaces with great dexterity. However, their load capacity is quite small.

To overcome this problem, it is also possible to define geometries formed by connecting links in parallel. For example, in 2D we can define a parallel geometry as shown below.

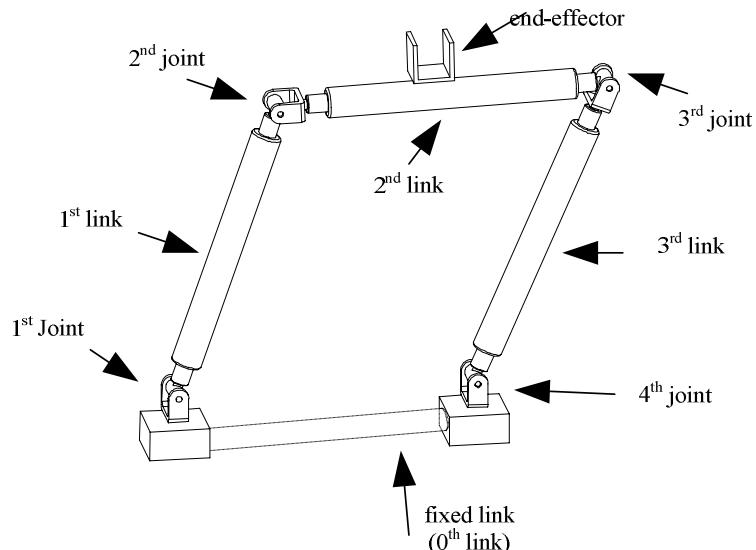


Figure 4.9: 2D parallel robot with only 1-DoF

This robot has one motorized degrees of freedom, Joint 1, which are rotating joint, and three passive degrees of freedom, Joint 2, Joint 3 and Joint 4. This gives 1-DoF. Other samples of 2-DoF parallel robots are shown below:

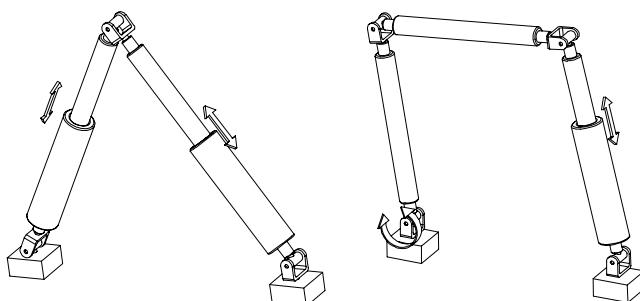


Figure 4.10: Samples of 2-DoF parallel robots: two prismatic.

For a 3-DoF parallel robot, we could use the geometry shown below.

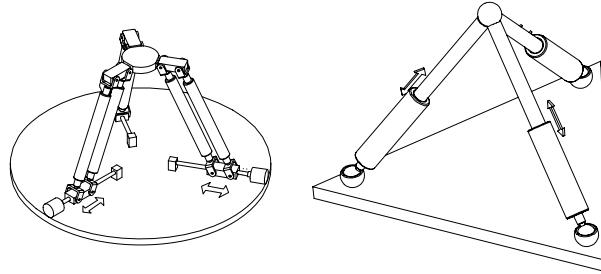


Figure 4.11: Samples of 3-DoF parallel robots: Star Robot (left); Mianouwski's

These robots have three motorized joints, which are again built as three prismatic joints. The remaining joints are passive, which are each implemented as passive rotational joints.

These geometries give us 3-DoF parallel robots, but how could we extend this to have a 6-DoF parallel robot? You should think about this and decide for yourself.

At this point, the question is: how can we distinguish a serial robot from a parallel robot? The answer is not completely clear. Notice that in all robots we can identify two main bodies: the first one is fixed (the base of the robot, or  $0^{th}$  link) and the second one is mobile and is considered as the robot's end-effector/end-point. We classify a robot as a real parallel robot when these two bodies are connected by several kinematic chains of **only one link**. The most common sample of parallel robots is the Stewart platform:

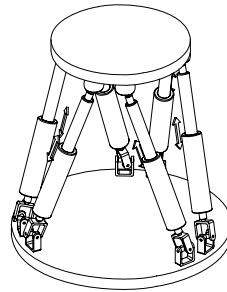


Figure 4.12: The Stewart Platform.

This structure owes its name to D. Stewart<sup>2</sup> who designed in 1965 a similar 6-DoF mechanism for flight simulators. The robot we know as Stewart platform is the design made by V. E. Gough<sup>3</sup> around the same period of time.

The chart below summarizes the differences between serial and parallel robots.

The main advantages of parallel geometry robots are that all the motors can be

<sup>2</sup> D. Stewart, A Platform with Six Degrees of Freedom, UK Institution of Mechanical Engineers Proceedings 1965-66, Vol 180, Pt 1, No 15.

<sup>3</sup> Gough, V. E., Contribution to discussion of papers on research in Automobile Stability, Control and Tyre performance, Proc. Auto Div. Inst. Mech. Eng., pages 392-394, 1956-1957.

Feature	Parallel robot	Serial robot
Workspace	↓	↑
Payload and weight of the robot	↑	↓
Price of components	↓	↑
State of art	Research	Very developed, many industrial applications
Accuracy and repeatability	↑	↓
Control	complex	simple
Applications	Increasing	Mainly, many industrial applications
Stiffness	↑	↓
Dexterity	↓	↑
Direct kinematics $(q_i \rightarrow P_e)$	Complex, multiple solutions.	Easy, one solution.
Inverse kinematics $(P_e \rightarrow q_i)$	Easy, one solution.	Complex, multiple solutions.
Direct statics $(\tau_i \rightarrow F_e)$	Easy, one solution.	Complex, many solutions.
Inverse statics $(F_e \rightarrow \tau_i)$	Complex, many solutions. (It is like a hyperstatic structure).	Easy, one solution.
In singular configuration	It gains DoFs	It loses DoFs.

Table 4.1: Comparison between serial and parallel robots.

mounted on the base of the robot, and not at each joint of the sequential chain, as in serial geometry robots. The links can thus be smaller and lighter, but they also form a stiff structure.

The main disadvantage of parallel geometries is that the effective workspace of the robot is quite small in comparison with the overall size of the robot.

These features are quite suitable for machining-tools. So we can find parallel robots as a machine-tool, but they are called **hexapods**.

It is also possible to mix serial and parallel geometries in one robot. Then we get a robot with a balance between the features of a parallel robot and a serial robot. The following figure shows two samples of hybrid robots:

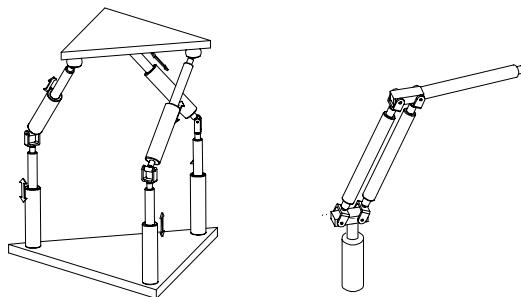


Figure 4.13: Hybrid robots: Hybrid parallel robot (left), Hybrid serial robot (right)

This second one is the most common geometry of hybrid robots. The parallelogram combination is used to replace the second link in an anthropomorphic robot arm. This combination increases the robustness and the stiffness of the robot without increasing its weight. Thus the payload of the robot is bigger. It also offers the possibility to mount the motor to drive the 2<sup>nd</sup> DoF at the base of the robot. One negative side effect, however, is that the workspace is decreased.

#### 4.3.2.1 How to determine the number of DoFs in a parallel geometry (Grübler criteria)

The number of DoFs of a serial robot is very easy to count because there is a one-to-one mapping between the DoFs and the joints. But this statement is false when we are dealing with parallel or hybrid robots.

Thus, if we want to count the number of DoFs in a parallel robot we have to use another method, such as the one described below (known as Grübler criteria).

A free link, under no constraints, has 6 DoFs (three position + three orientation). If the movement is constrained in one direction, i.e., the link must touch a surface, the link will have 5 DoFs because one DoF has been blocked. In this case, the link can be moved along the tangents to the surface and can be rotated in three directions.

We can conclude that the constraints restrict the DoFs of the links in such a way that the number of DoFs allowed ( $f$ ) plus the number of DoFs restricted ( $u$ ) by the constraint must be equal to six:

$$f + u = 6 \quad (4.1)$$

The number of DoFs of a robot can be obtained by applying the 4.1 to every link of the robot. The number of DoFs is also known as mobility ( $M$ ), and it can be calculated by the 4.2 (Grübler criteria).

$$M = 6(n - 1) - \sum_{i=1}^g u_i = u + f \quad (4.2)$$

Where  $n$  is the number of links,  $g$  is the number of constraints and  $u_i$  is the number of DoFs restricted by the  $i^{th}$  constraint.

#### 4.3.3 Wrists

We can place the robot using 3-DoF geometries. Now the attention is focused on the orientations we need when a task must be performed. Let us see as an example the drilling task in figure 4.14.

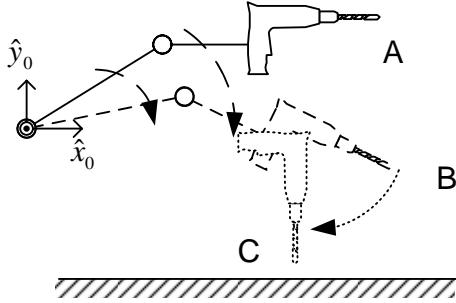


Figure 4.14: The drilling task needs the placement of robot's end-effector and orientation as well.

The robot will be at the right location only when it manages to place and orient the end-effector (the drill). In previous sections, we see we can place the end-point of the robot correctly if it has 3-DoF. Now, if we also want to orient it, we will need 3 extra DoFs.

To convert any one of those 3-DoF geometries into a 6-DoF robot manipulator, with 3-DoF of position and 3-DoF of orientation, we need to add a 3-DoF wrist.

In most robots, the wrist mechanisms have three axes that are joined in series by very short links.

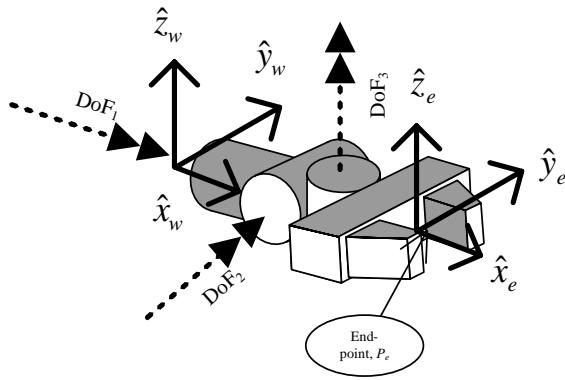


Figure 4.15: Wrist sample: gripper with two fingers.

There are many types of wrists. They are classified depending on the direction of the wrist-axis. As can be seen in figure 4.15, the first DoF is aligned with the  $\hat{x}_w$ ; the second one, with the  $\hat{y}_w$ ; and the third one, with the  $\hat{z}_w$ .

These rotating directions are often referenced using the conventional aeronautics terminology; that is to say, **roll** is a rotation aligned with the  $\hat{x}_w$ , **pitch** is with respect to the  $\hat{y}_w$ ; and **yaw** is a rotation around the  $\hat{z}_w$ .

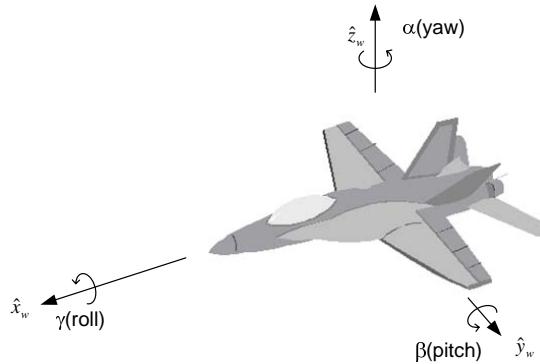


Figure 4.16: Conventional Aeronautics Terminology

Thus we have a **Roll-Pitch-Yaw**-wrist or RPY-wrist. A simplified representation of that wrist is given in figure 4.17:

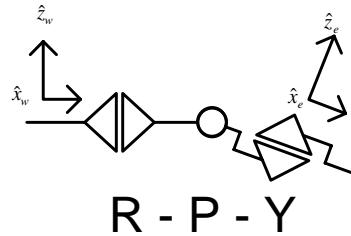


Figure 4.17: RPY-wrist.

In figure 4.18 samples of 3-DoF wrists can be found. Nevertheless, samples of 1-DoF or 2-DoF wrists can be found in real applications. In this case the robot will have more restricted movements, but will be cheaper. The figure 4.18 shows a sample of 2DoF wrist.

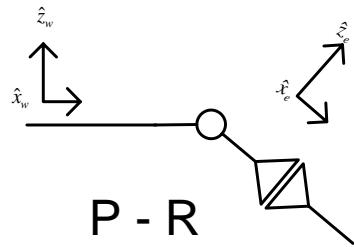


Figure 4.18: RP-wrist.

We can find samples of wrists based on parallel mechanisms as well. In this case, the wrist will have properties similar to parallel robots (high load capacity, but small workspace).

Ideally, the axes of the wrist should coincide, so that rotation around any axis does

not produce a change of position of  $P_e$ . That is to say, the rotation and the translation will not be coupled. In practice, however, this is hard to realize (and expensive). However, other samples of cheaper wrists, of which three axes intersect, can be found. This is the case of **Roll-Pitch-Roll**-wrists (figure 4.19)

However, an RPR-wrist has other disadvantages, such as lower dexterity and singular configurations.

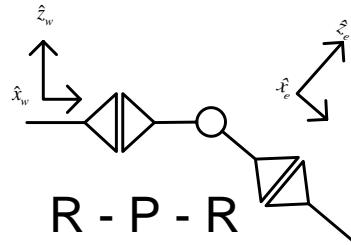


Figure 4.19: RPR-wrist.

#### 4.3.4 Some Things to Think About

1. Are all seven types of joints, presented in section above, all the different types of possible joints? If not, what other types are possible, at least in theory?
2. Are all five different geometries, presented in section above, all the different serial geometries that are possible to form? If not, what others are possible?
3. How many DoFs do you have in one of your arms?
4. What is the geometry of your arm, and how does it compare to the five basic robot manipulator geometries presented in this chapter?



## Chapter 5

# Basic Components of Robots

---

## 5.1 Introduction

In this part of the course we will briefly look at the technology and the basic components used in robotic arms, but without going into much detail. As we have seen from our consideration of the geometry of robot manipulators, they are formed from combinations of links and motorized joints that each typically has one DoF. These joints are an assembly of different components, such as a motor or actuator, a reduction mechanism, a transmission mechanism, and an internal sensor that measures the actual DoF position (see the parts of a PA-10 robot in figure 5.1. We can also find external sensors that let the robot sense the world around it. And last, but not least, the end-effector, or tool, attached to the end-point of the robot.

## 5.2 Links

The links of a robot arm should:

- have low mass, and
- be very stiff.

These two requirements are not easily satisfied together. If an element is heavy and not stiff enough, it will deflect under its own weight. If the robot is also holding something in its gripper, for example, this deflection will be even larger.

To increase the stiffness of the links they are usually made bigger and stronger, but this also makes them heavier. Heavier links need more powerful motors to drive them. More powerful motors are also heavier than less powerful versions. If the motors are placed at the joints, this extra weight of the motors also tends to deflect the link.

A deflection of the links of a robot arm results in position errors at  $P_e$ , an error that accumulates along a serial geometry arm. It is therefore very important that deflections are kept minimal, though it is impossible to avoid them completely.

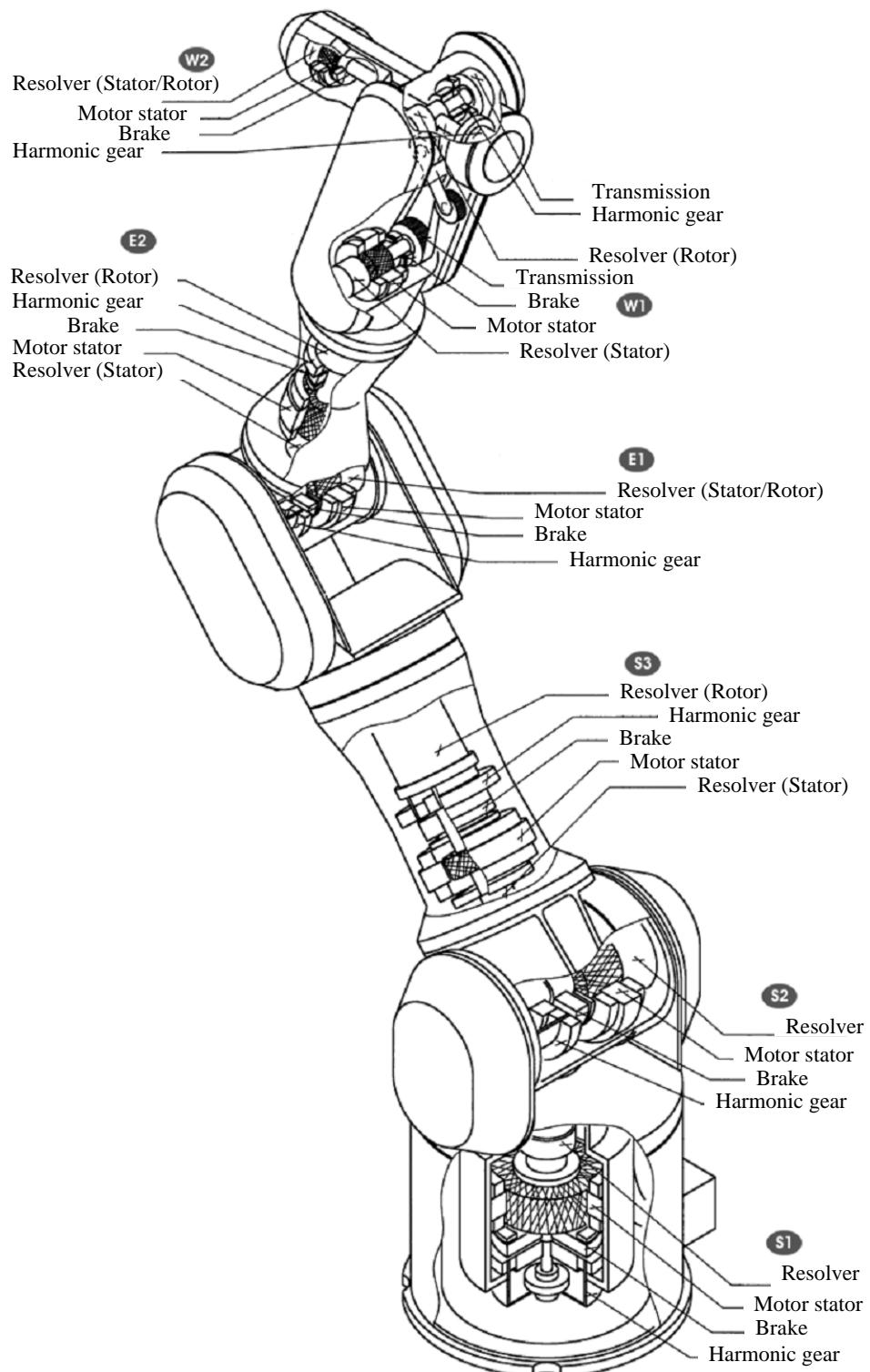


Figure 5.1: Mechanical structure of the PA10

The links of robots are normally made of steel.

## 5.3 Actuators and Motors

There are basically three different kinds of actuators or motors used in robots: Pneumatic, hydraulic, and electric.

### 5.3.1 Pneumatic Actuators

The pneumatic actuators use compressed air between 5 and 10 bar. They need a special installation composed of a compressor, pressure regulators, dust filters, lubricators, drains, etc, up to the compressed-air socket (figure 5.2).

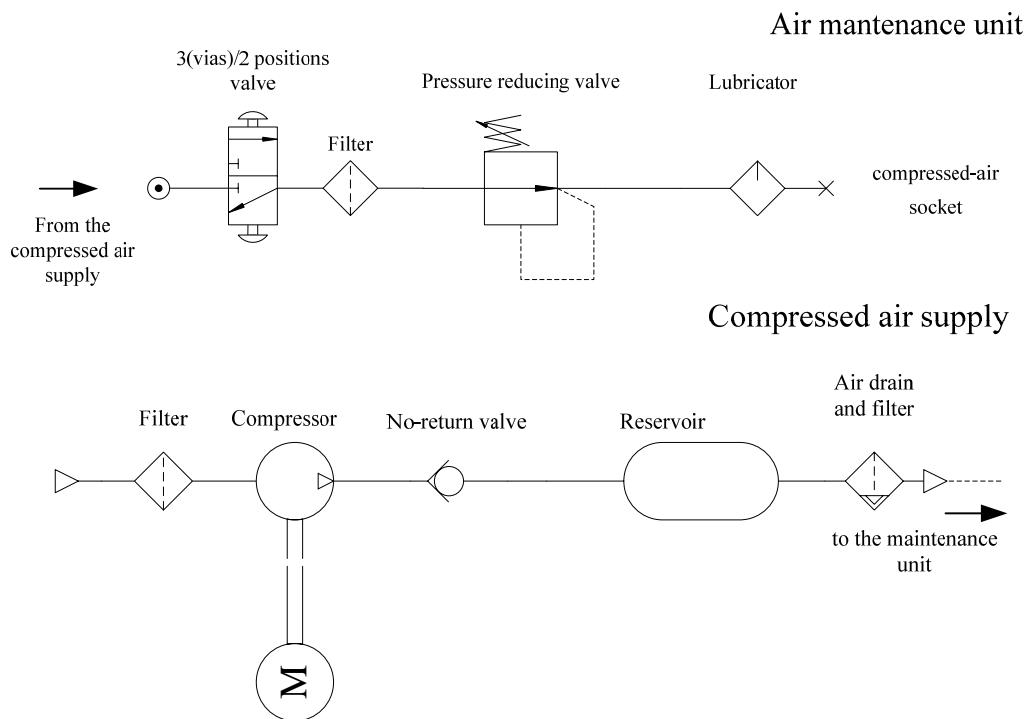


Figure 5.2: Compressed-air preparation.

There are two types of pneumatic actuators (figure 5.3) that can be connected to the air compressed socket: linear actuators (pneumatic cylinders) and rotary actuators (pneumatic motors).

The precision of pneumatic actuators is low. Moreover, they are not stiff, since the air is compressible. However, they do not break if they are overloaded.

The pneumatic actuators are noisy but cheap, robust and clean. They do not need an air-return line. That is to say, when the compressed-air is used in the actuators, it is released by an air-vent valve.

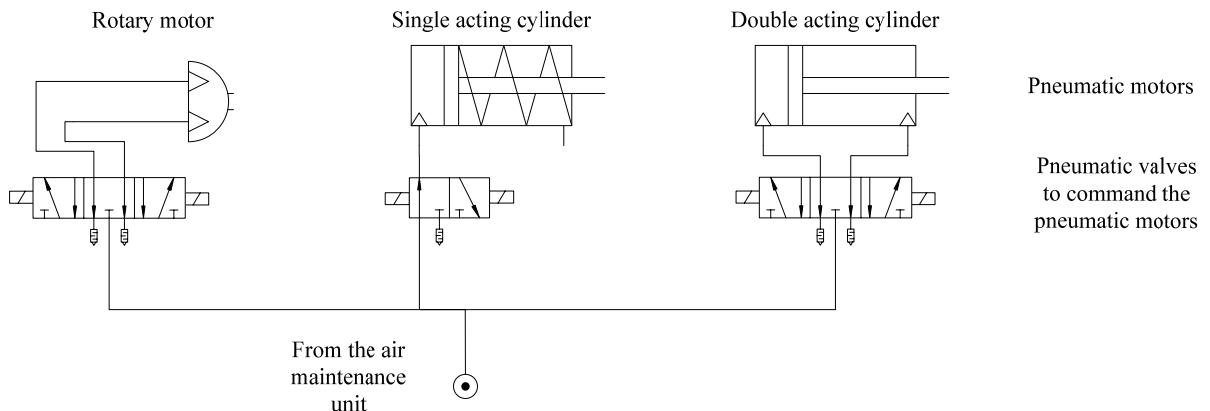


Figure 5.3: Types of pneumatic actuators.

Due to their characteristics, pneumatic actuators are often used as drivers for robots' end-effectors.

### 5.3.2 Hydraulic Actuators

The hydraulic actuators are quite similar to their pneumatic counterparts. However, we can find many differences, since the hydraulic actuators work with mineral oil, under a pressure of between 50 and 100 Bar. Hydraulic installations have pumps instead of compressors (oil is uncompressible). A return-line is needed for the oil flow through a closed loop.

Since oil is less compressible than air, hydraulic actuators have more precision and they are stiffer. However, they have other problems due to the fact that certain properties of oil, such as viscosity, depend on temperature.

Because of the high working pressures, the hydraulic actuators can exert higher forces, but they are more expensive.

The maintenance of hydraulic installations is more complex because there is often leakage.

The use of oil leads to auto-lubricated but dirty systems.

Similar to pneumatic actuators, the hydraulic ones are very robust and they do not break in stall conditions originated by an excessive load.

### 5.3.3 Electric Actuators

Electric actuators are most commonly used due to their simplicity. They can have high precision and they are very reliable. They are also noiseless.

As a disadvantage, we can find the maximum power that they can exert is quite

low. Generally speaking, their output speed is too high compared to the output force (or torque), which is too low. For that reason, reduction mechanisms are required and, in many cases, an integral geared speed reducer is incorporated.

They can burn up under overload conditions.

There are many types of electric actuators, among them, the most important ones are listed below:

- Conventional DC motors.
- Conventional AC motors:
  - Asynchronous motors.
  - Synchronous motors.
- Stepper motors.
- RC Servomotors.
- Brushless motors.
- Direct Drive motors:
  - DC Direct Drive Motors.
  - AC Direct Drive Motors.

### 5.3.3.1 Conventional DC Motors

They are used many times because they are the easiest to use. Their speed can be controlled smoothly from zero to the steady state value. They also respond quickly to control-signal changes. A model of the motor is shown in figure 5.4.

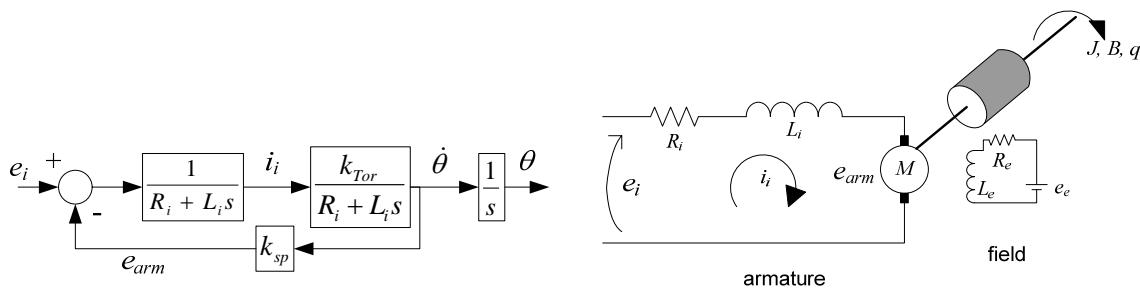


Figure 5.4: A model of a DC motor.

The motor speed can be controlled by the voltage of the armature:

$$e_{arm} = k_{sp}\dot{\theta} \quad (5.1)$$

Where  $e_{arm}$  is the voltage of the armature,  $k_{sp}$  is the speed-constant of the motor, and  $\dot{\theta}$  is the output speed.

But the motor torque can be controlled as well. In this case, we have to control the current of the armature:

$$T = k_{tor} i_i \quad (5.2)$$

Where  $T$  is the output torque,  $k_{tor}$  is the torque-constant of the motor, and  $i_i$  is the current of the armature.

We can also control torque through the current of the field. By weakening the field current, the motor speed will increase, but the torque will be reduced. In fact, varying the current field, we are changing the values of motor constants. However, this method of control is less common.

For a given motor  $k_{tor}$  and  $k_{sp}$  are equal, since the energy conservation law must be fulfilled, that is to say, the input power must be equal to output power.

$$P_{in} = P_{out} \quad (5.3)$$

If we consider the electrical power as input power, and the mechanical power as output power, we have:

$$e_{arm} i_i = T \dot{\theta} \quad (5.4)$$

Rearranging the equations, we see that the two motor constants must be equals:

$$\frac{e_{arm}}{\dot{\theta}} = k_{sp} = \frac{T}{i_i} = k_{tor} \quad (5.5)$$

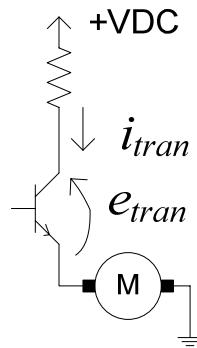
The disadvantages of DC motors are:

- The output speed is too high, and the output torques are too low, so a reduction mechanism is always required.
- The speed control is an open loop, so a position sensor is required.
- The brushes are mechanical sliding parts under friction, so they need maintenance. Moreover, there is usually brush sparking, so the brush-motors are forbidden in flammable environments.

**PWM AND PFM CONTROL SIGNALS** Traditional analog speed controllers regulate the speed by lowering the voltage to the motor through a resistor or a transistor working at its linear region. This is an inefficient technique because the transistor/resistor consumes power that is lost as heat.

In Table 5.1, we can see the transistor will not consume power if it is short or open. Thus we can avoid the power loss if the transistors of the power unit do not work in their linear region.

There is a technique called **PWM**, which tries to apply this idea. PWM stands for *Pulse Width Modulation* and it refers to the method of applying a square control signal



Condition	$i_{tran}$	$e_{tran}$	$P_{tran} = i_{tran}e_{tran}$
Linear region	$i_{tran}$	$e_{tran}$	$i_{tran} \neq 0$
Open	0	$e_{tran}$	$0.e_{tran} = 0$
Short	$i_{tran}$	0	$i_{tran}0 = 0$

Table 5.1: The transistor only dissipates power in the linear region.

to the actuator. This signal has a constant frequency. Thus, the actuator receives the current/voltage in an ON-OFF manner.

Changing the ratio of the ON time to the OFF time changes the speed of the motor. For example, at 80% speed, the current pulse is on 80% of the time and off 20% of the time. In this case, we say the signal has a **duty cycle** of 80%.

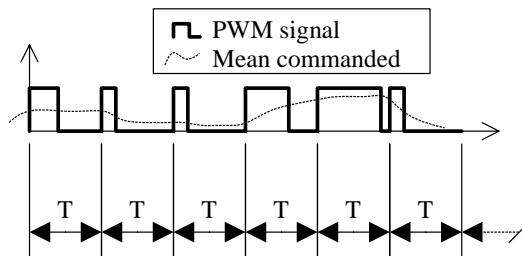


Figure 5.5: A PWM signal.

If the frequency of the PWM signal is high enough, the inductor of the motor will integrate the signal and the movement will be smooth.

Since the motor is a mechanical device, a frequency above 1 kHz would be enough if a man could not hear frequencies up to 20 kHz. Thus, a frequency of 25 kHz or higher could be a good choice for a PWM signal.

There is another technique, but less common, called **PFM**. In this case, PFM stands for *Pulse Frequency Modulation*. In this case, the pulse-width is constant and the speed is controlled by the frequency of the voltage signal.

These techniques are used in combination with an H-bridge (figure 5.7). An H-bridge

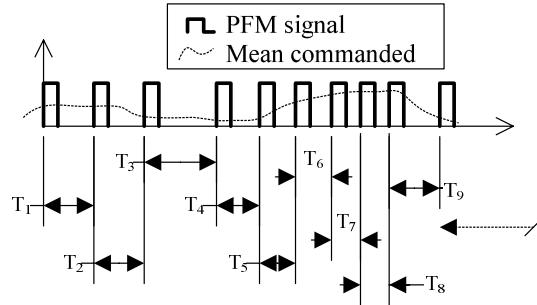


Figure 5.6: A PFM signal.

is a very popular circuit for driving DC motors. Its name is due to the fact that it looks like the capital letter 'H' in classic schematics. The great advantage of an H-bridge circuit is that the motor can be driven forward or backward at any speed, optionally using a completely independent power source.

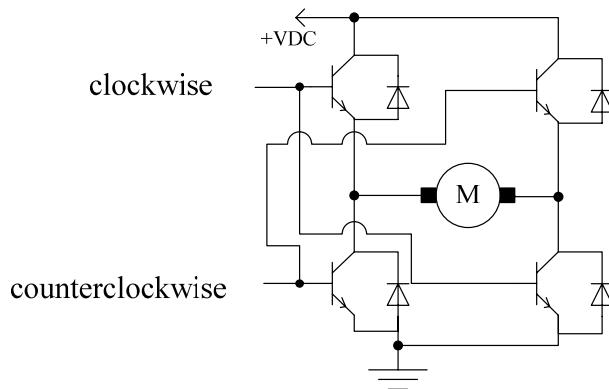


Figure 5.7: An H-bridge.

An H-bridge can be implemented with various kinds of transistors (common bipolar transistors, FET transistors, MOSFET transistors, power MOSFETs, or even chips). The diodes in figure 5.7 are included to protect the transistors against high voltages induced by the motor's coils, during switching.

### 5.3.3.2 Synchronous AC Motors

Synchronous AC motors have windings in the stator and in the rotor. The stator windings are powered by a three-phase power supply and, as a consequence of this, a rotating magnetic field is generated. The rotor windings are powered by a DC source, so it creates a constant magnetic field.

Synchronous motors are constant speed motors; they operate in absolute synchro-

nism with power line frequency. So, the speed can be controlled by varying the frequency. Varying the speed by the frequency is not as easy as varying the speed by voltage.

The advantages of this type of motor are:

- There are no brushes (no-sparks, no-wearing).
- It has a better power to weight ratio than the DC motors.

### **5.3.3.3 Asynchronous AC Motors**

Induction motors are the most common form of asynchronous motors. They are basically AC transformers. The primary coil of the transformer is called the stator and is connected to a power supply (normally, a three-phase power supply). The secondary winding of the transformer is shorted in the rotor of the motor. The torque is produced by the interaction of two magnetic fields: the first one is generated by the stator current, the second one is created by the induced rotor currents.

Thanks to a special configuration of the stator windings, the stator magnetic field rotates and drags the rotor magnetic field, which is fixed to the rotor.

The main feature of this type of motor is that the synchronous speed is the absolute limit of motor speed. At synchronous speed, there is no difference between rotor's speed and rotating stator field speed, so the voltage induced in the rotor is equal to zero. Thus there is no output torque. Therefore, the rotor must rotate slower than the magnetic field. The motor speed is just slow enough to balance the output torque and the resisting torque.

From the previous paragraph, we can conclude that the speed control of asynchronous motors is more complex. For that reason, the asynchronous motor has been less used in robotics. However, nowadays this problem is being overcome thanks to the new voltage/frequency regulators.

### **5.3.3.4 Stepper Motor**

The actuators described above require a position sensor in order to control the position. Stepper motors are devices that do not require position sensor since they convert electrical pulses into a mechanical rotation in fixed angular increments (steps).

Thanks to this, stepper motors are often used in open loop control systems. A position sensor can be used to confirm positioning accuracy.

Step size is determined by the construction of the motor. We can find models that have steps of 1.8 degrees or less.

The working principle of stepper motor is very simple. Their windings are grouped into two phases. They move in steps when their phases are sequentially energized (full-stepper motor, figure 5.8): when the current flows through phase 1, the rotor will align

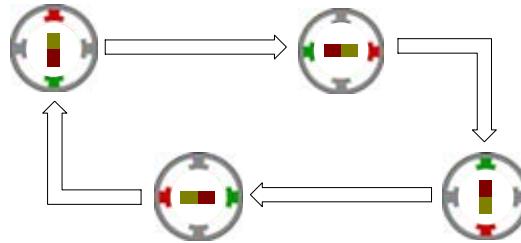


Figure 5.8: Full-stepper motor.

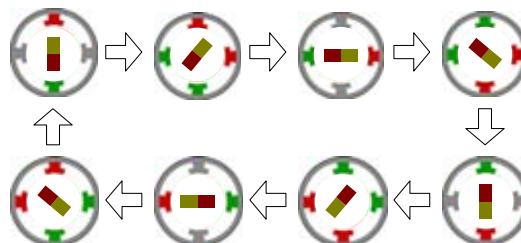


Figure 5.9: Half-stepper motor.

with it. Then, the current is switched from phase 1 to phase 2 and the rotor will move to be aligned with phase 2, turning one step.

But, in some cases, both phases can be energized simultaneously. Then the rotor will establish its equilibrium midway between two steps. In this case, the motor is said to be half-step (figure 5.9).

There are several types of stepper motor depending on the rotor characteristics (figure 5.10). The most popular are:

**Permanent Magnet (PM) stepper motor** This type of stepper motor is also referred to as a synchronous inductor motor. This motor has a magnet in the rotor. This magnet will align with the energized phase.

**Variable Reluctance (VR) stepper motor** While this motor has a non magnet rotor, it is notched<sup>1</sup> instead of having a magnet.

**Hybrid (HB) stepper motor** This motor combines both of the above types, i.e. magnet and notches.

We can also use windings as a criteria of classification:

**A bipolar stepper motor** with two phases has one winding/phase.

<sup>1</sup>Notched=con muesca

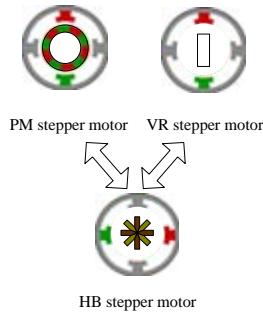


Figure 5.10: Stepper motor: rotor types.

**A unipolar stepper motor** has one winding, with a centre tap per phase. Sometimes the unipolar stepper motor is referred to as a "four-phase motor", even though it only has two phases.

This difference comes from the fact that we have to have a bipolar supply to power the bipolar windings if we want to complete one turn (figure 5.11).

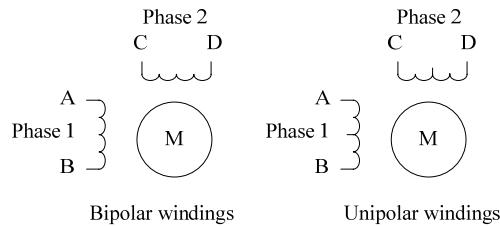


Figure 5.11: Bipolar windings vs. Unipolar windings.

The main disadvantages of stepper motors are:

- The maximum output power is low.
- At low speeds, a ripple torque can appear. The movement will not be smooth.
- Since the control is in an open loop, the motor can lose one stepper as a consequence of a disturbance. Without a position sensor, there is no way to detect this type of event.

### 5.3.3.5 RC Servomotor

Let us consider the motors used in radio controlled models as **RC servomotors**. The term **servo** comes from the fact that this kind of motor has implemented a rudimentary, but powerful, position loop. The servomotor itself has a built-in motor, a gearbox, and a position feedback controller.

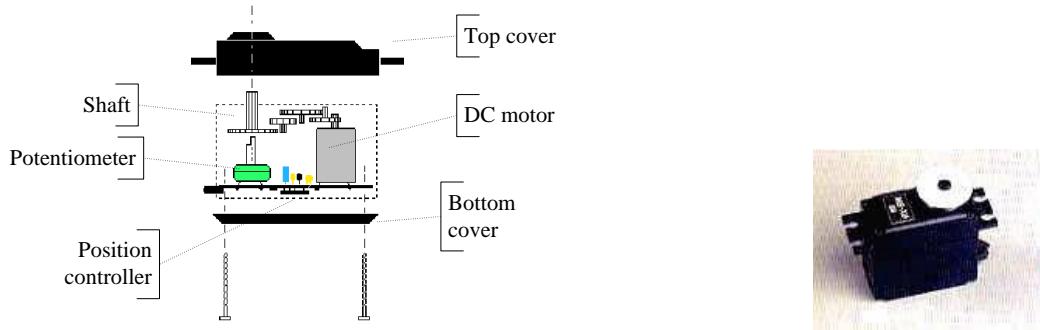


Figure 5.12: A RC servomotor: internal parts (left), external appearance (right)

They are very useful in many kinds of small robotics experiments because they are small, compact and inexpensive. Servomotors usually have a three-wired connector. The standard color code is black for the ground, white for the control pin and red for the power supply (5 Volts).

The control signal that must be sent to the servo is a periodical pulse of 1.5-2.5 ms around every 20 ms approximately (figure 5.13). The period of the signal is not critical, but if the period is too high, the motor will lose/forget the command and if it is too low, the controller can become unstable.

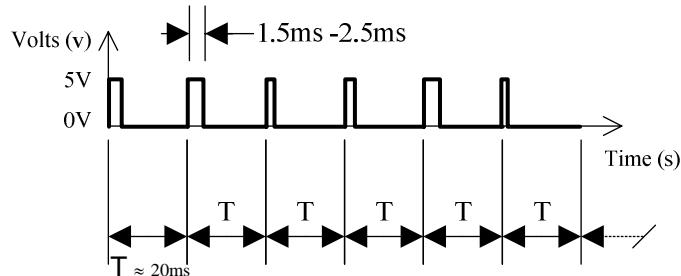


Figure 5.13: Commanding a RC servomotor.

If the pulse lasts 1.5 ms that will set the motor to the position of 0 degrees, and if it lasts 2.5 ms, that will set it to the other end-stroke limit. The other values in the interval (from 1.5 ms to 2.5 ms) set the motor position to one that is interpolated linearly.

### 5.3.3.6 Brushless Motor

A brushless motor resembles a common DC motor, but upside down. That is to say, the permanent magnets are located on the rotor while the powered coils are on the stator. So there are no brushes or mechanical sliding parts to power the rotor.

The rotating field is obtained by placing three stator windings that are connected to a special driver. This driver can sense the rotor position and powers the proper winding in order to get a rotating stator magnetic field that must be perpendicular to the rotor magnetic field.

Its use is similar to DC motor but, in this case it can be used in flammable environments (no brushes, no sparks).

### 5.3.3.7 Direct Drive Motor

The biggest disadvantage of every electrical motor is their output speed is too high and the torque is very low. The interesting combination is to get high torques at low speeds. So, the use of a gear is always required.

We will see later that the introduction of a gear has several disadvantages, such as increased inertia, friction, backlash or greater loss of backdrivability.

The direct drive motors can overcome this problem. We can find AC Direct Drive Motors and DC Direct Drive Motors as well. The advantage of those motors is that they have a special magnetic circuit that allows them to work at low speeds while exerting high torques without any gearing.

In this case, we can connect the load directly to the shaft of the motor.

The main advantages of these motors are summarized below:

- Fast and accurate positioning.
- Simple mechanical system (noiseless and frictionless).
- High acceleration ranges (1g for lead screws, 5g for DD motors with conventional bearings, 10g for DD motors with air bearings).
- High speed ranges.

But we can also find disadvantages:

- They are very expensive.
- Ultra-high resolution position sensors are required.
- The system has a high backdrivability. Thus, if the motor is unpowered, the mechanism can not support its own weight and, it will fall to a stable position.
- The motors are big and heavy.

## 5.4 Reduction mechanisms

When the actuators' output speed is too high and/or the output torque/force is too low, it is necessary to convert them in order to achieve lower speeds and higher torques (or

forces). This typically means that some kind of reduction mechanism is placed between the motor or actuator and the joint it drives.

For robot arms, reduction mechanisms should have special characteristics, since the speeds are quite high with many peaks of acceleration. Their features are listed below:

- low in weight,
- small in size,
- low friction, and
- have very little backlash or play,

Sometimes, backdrivable actuators are desirable, but most often, high reduction actuators are used.

In robotics, the most used reduction mechanisms are the **harmonic-drive** and the **cyclo-drive**. Both of them work under the same basic principle. In order to simplify the explanation, let us carry out a little experiment (figure 5.14): Grab a coin as is seen in the figure below and follow the steps. For example, if you describe clockwise circles, you should notice how the smaller coin rotates counterclockwise and vice-versa.

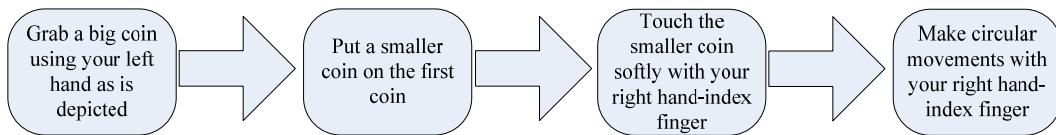


Figure 5.14: Robotic redutor mechanism simulation.

#### 5.4.1 Harmonic-Drive

Now, we can start to analyze the working principle of a **harmonic drive**<sup>2</sup>.

The harmonic drive is composed of three elements (figure 5.15):

- The first element is the **wave generator**, it is connected to the actuator's shaft. It has an elliptical shape.
- The second element is the **flexispline**. It's made of some kind of elastic matter. It can adapt its shape to the shape of the wave generator.
- The last one is the **circular spline**.

<sup>2</sup> <http://www.harmonicdrive.de>

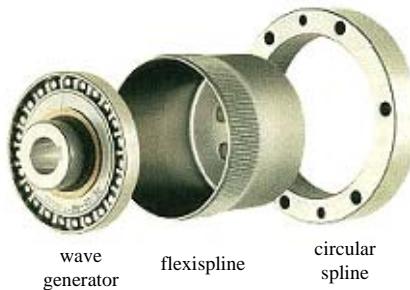


Figure 5.15: Components of a Harmonic-Drive.

The number of teeth of the **flexispline** is two or three less than the teeth of the **circular spline**.

When the harmonic drive is assembled, the wave generator deforms the flexispline in the way that only a few teeth are engaged with the circular spline (only where it corresponds at the major axis of the wave generator).

The figure 5.16 shows the working sequence of a harmonic drive when we rotate the wave generator clockwise. The **flexispline** will rotate counterclockwise an angle that corresponds to the teeth difference.

This ‘rotation delay’ of the **flexispline** is due to the small difference between the number of teeth of the **flexispline** and the number of teeth of the **circular spline**.

The reduction ratio of this reduction mechanism is commonly about 400:1.

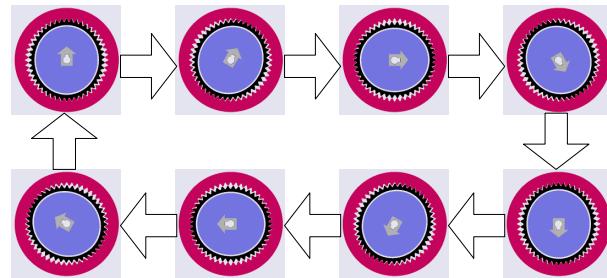


Figure 5.16: Harmonic-drive working sequence.

### 5.4.2 Cyclo-Drive

The second reduction mechanism commonly used in robotics is the **cyclo drive** (figure 5.17).

Although the shape looks very different, the working principle of the **cyclo drive** (figure 5.18) is similar to the working principle of the **harmonic drive**. But the performance of the **cyclo drive** is not as smooth as the performance of the harmonic drive. Another problem of **cyclo drives** is the eccentricity between the input and the output shafts, so it is necessary to assemble it always in pairs.

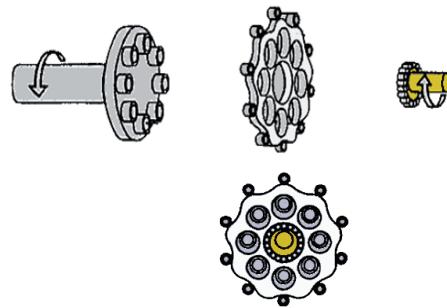


Figure 5.17: Schematic of a Cyclo-drive.

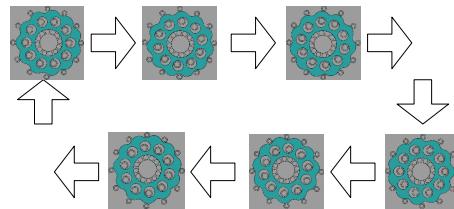


Figure 5.18: Cyclo-drive animation.

#### 5.4.3 Other reduction and Transmission mechanisms

The next table summarizes the most typical transmission mechanisms we can find. Most of them can also be used as reduction mechanisms.

### 5.5 Sensors

Robots need sensors in order to complete their task with sufficient accuracy and speed, and to be able to interact with their environment. Thus we can distinguish two types of sensors:

**Internal sensors** Since every active DoF is motorised and controlled, they need, at least, a position or speed sensor.

**External sensors** The robot uses them to interact with its environment.

The sensors are also divided into digital sensors and analog sensors. The first ones are sensitive to noise, and the second ones have problems related to the resolution and transmission speeds.

Given the characteristic curve, which relates an output voltage with the measured variable, there are many important aspects of sensors which need to be considered: if, how, and when to use them in any robot application. The most important aspects are summarized in below:

<b>Picture</b>	<b>Input</b>	<b>Output</b>	<b>Name</b>	<b>Advantages</b>	<b>Disadvantages</b>
	circular	circular	gears	-High torque; -Non-slip; -Low-cost.	-Backlash; -Heavy, not suitable for high speeds; -Noisy; -Increased friction.
	circular	circular	cogged belt	-Allows great distances between axes; -Non-slip; -Low-cost.	-Backlash; -Heavy, not suitable for high speeds; -Noisy; -Increased friction.
	circular	<u>circular</u> <u>linear</u>	belt	-Allows great distances between axes; -Non-slip; -Low-cost.	-Backlash; -Heavy, not suitable for high speeds; -Noisy; -Increased friction.
	circular	<u>circular</u> <u>linear</u>	wire (transmission)	-Backdrivable; -Low friction; -Low cost; -Low weight.	-Low torques; -May slip.
	circular	circular (continuous or and-fro motion)	four bar mechanism	-High torques; -Non-slip.	-Small stroke.
	circular	linear	lead screw	-Light-weight; -Translate circular motion to linear.	-Low speeds; -Backlash; -High friction; -Not backdrivable.
	circular	<u>linear</u> <u>circular</u>	rack and pinion	-High torques; -Non slip.	-Low speeds; -Backlash; -Increased friction.

Table 5.2: Transmission Mechanisms

**Sensitivity** Slope of the characteristic curve. If the sensitivity is constant, it is also known as the gain of the sensor (figure 5.19).

**Sensitivity error** The variation of the real slope from the ideal (theoretical) slope.

**Range** The maximum and the minimum values that can be measured.

**Dynamic range** The total range of the sensor (from the minimum to the maximum).

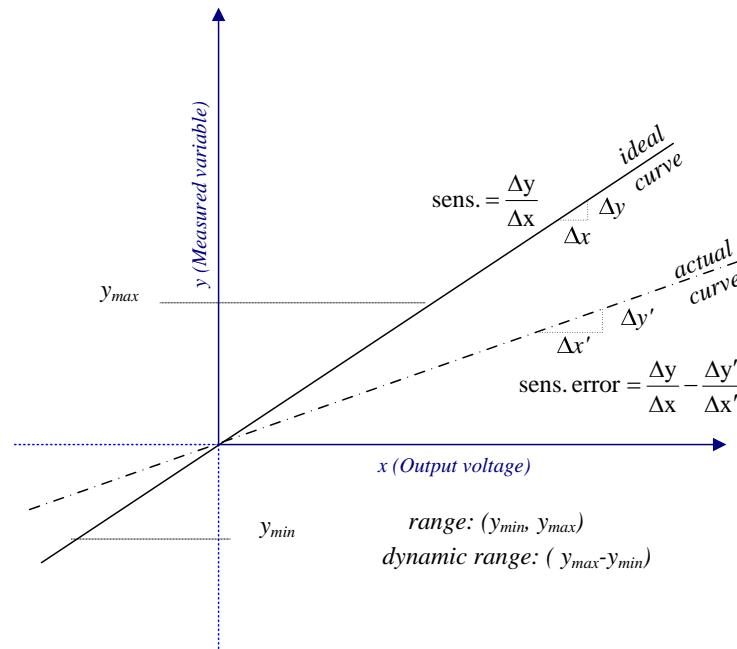


Figure 5.19: Sensivity and range for a given sensor

**Resolution** Smallest detectable incremental change of the measured variable (figure 5.20). This concept is more related to digital sensors.

**Accuracy** The maximum difference between the actual value and the measured value by the sensor (figure 5.21).

**Offset** The output that appears when it should be zero.

**Linearity** The maximum deviation between the theoretical linearized curve and the actual curve. The most common types of non linearities are: hysteresis, dead zone, saturation.

**Hysteresis** Ideally, the sensor response should not depend on the direction in which the measured variable is changing, that is to say if the measure variable is increasing or if it is decreasing. The hysteresis is the measure of this dependency (figure 5.22).

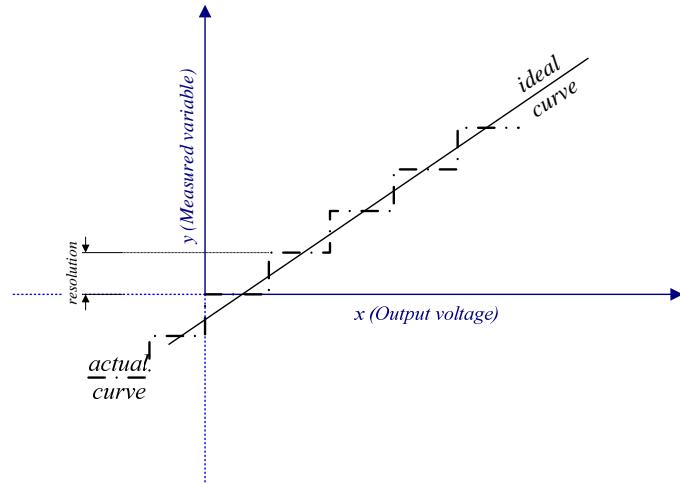


Figure 5.20: Sensor resolution

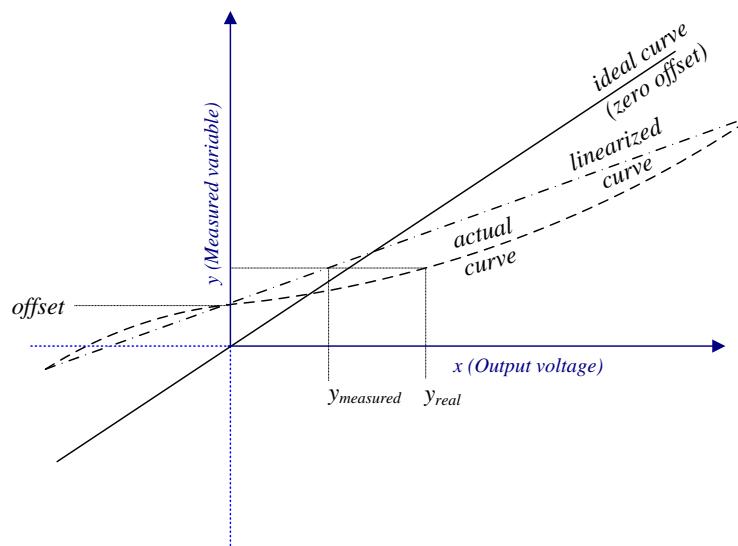


Figure 5.21: Accuracy, offset and linearity for a given sensor

**Dead zone** A band close to zero in which there is no output change even if there is change in the output value (figure 5.23).

**Saturation** If the measured value travels beyond the sensor ranges, the output will not follow the changes of the measured values. In this case, the sensor get saturated.

**Response time** Time required for a sensor output to change from its previous state to a final settled value within a tolerance band. If we are considering that the final state is lower than the previous, we use the term **decay time** (figure 5.24).

**Bandwidth** The maximum frequency for the measured variable that can be followed by the sensor. It is given by the frequency in which the sensitivity of the sensor is degraded by a factor of 1.41. A proper choice of this parameter is important for noise rejection (figure 5.25).

It is desirable that all of the properties described above be constant over time and with use. But the sensor should not be "better" than is needed for the particular application with we are dealing.

Another important topic, related to the use of the sensor but not to the measure itself, is the sensor calibration: the calibration of the sensors should not take very long to do, nor need special expertise, nor special equipment to perform, nor should it need to be done very often.

### 5.5.1 Internal Sensors

Each joint (DoF) that is motorized and controlled usually needs some kind of internal sensor of position or speed. Only in the case of joints that use stepper motors, the internal sensors are not required.

The table 5.3 summarises the internal sensors described in following sections:

#### 5.5.1.1 Potentiometer

A **potentiometer** is a variable resistor that acts as a voltage divider. We can find linear and rotating versions.

The advantages are:

- Low-cheap.
- Easy to use.
- Absolute measurement.

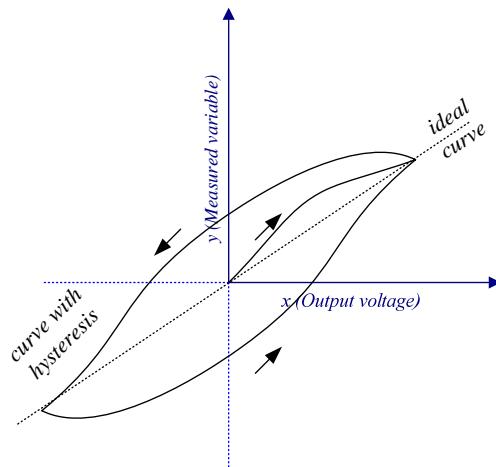


Figure 5.22: Sensor hysteresis

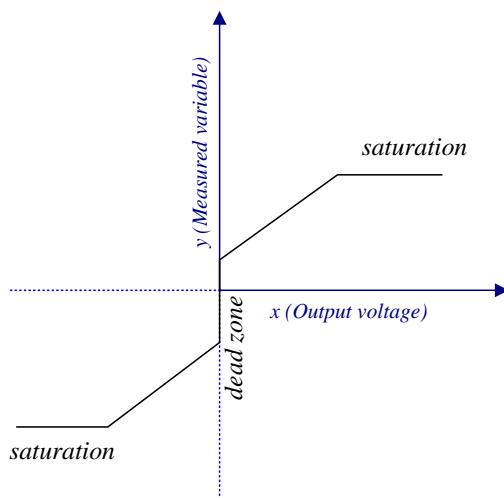


Figure 5.23: Dead zone and saturation for a given sensor

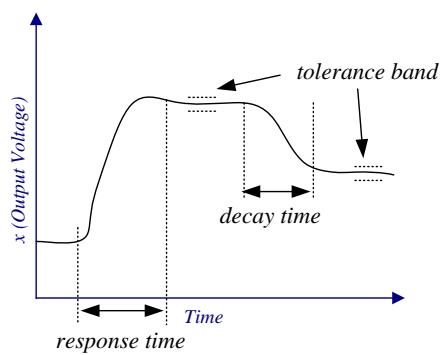


Figure 5.24: Sensor response time

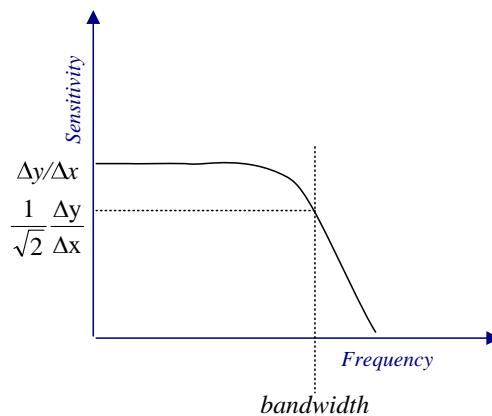


Figure 5.25: Sensor bandwidth

	analog	<ul style="list-style-type: none"> <li>• Potentiometer</li> <li>• Resolver</li> <li>• Synchro</li> <li>• Inductosyn</li> <li>• LVDT</li> </ul>
Position		
	digital	<ul style="list-style-type: none"> <li>• Optical Encoder (incremental and absolute)</li> <li>• Optical Ruler</li> </ul>
Speed	analog	<ul style="list-style-type: none"> <li>• tachometer</li> </ul>
End-stroke	digital	<ul style="list-style-type: none"> <li>• Mechanical switch</li> </ul>

Table 5.3: Internal sensors.

- Robust.

The disadvantages are:

- Low precision.
- The measurement depends on the temperature (i.e., it is unstable from the thermal point of view).

### 5.5.1.2 Resolver and Synchro

Diagrams of a typical **resolver** and a typical **synchro** are shown in the figure below. Both of them are based on single-winding rotors inside fixed stators. In the case of the resolvers, the stator has two windings at  $90^\circ$ . In the case of the synchros, the stator has three windings oriented at  $120^\circ$ , connected in a Y-connection, as we can see in the primary of a tri-phase transformer (figure 5.26). The performance of the **resolvers** and

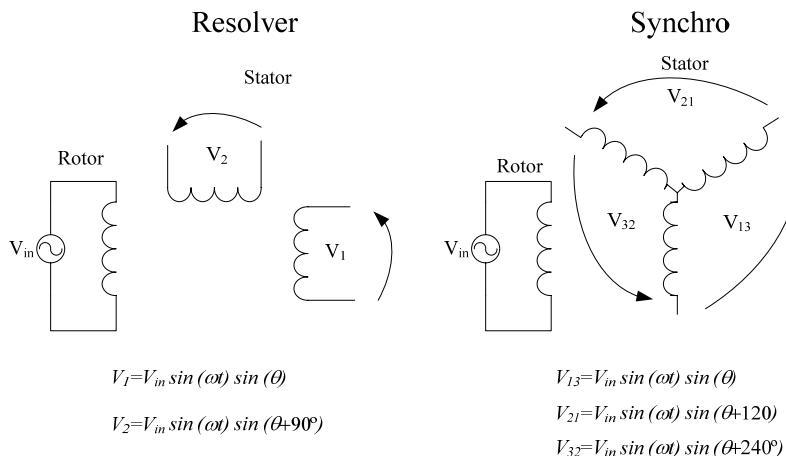


Figure 5.26: Resolver and Synchro.

the **synchros** is similar to that of rotating transformers: the rotor winding is driven by AC voltage ( $V_{in}$ ), which induces voltages in the stator windings. These induced voltages depend on the amplitude and frequency of the voltage of the rotor's windings and the relative angle,  $\theta$ , between the stator and the rotor (shaft angle).

As can be seen in figure 5.26, in the case of the **resolver**, the format of the output signal differs from the output signal in the case of the **synchro**. However, the **synchro** output can be easily converted into the resolver-equivalent format using a Scott-T transformer 5.27. Now, the remaining problem is to connect the sensor to our digital controller. The conversion is not easy but it can be resolved using a R/D (Resolver-to-Digital) converter depicted in figure 5.28. The **resolvers** and the **synchros** are rugged sensors as well, suitable for difficult conditions and accurate. As there are windings in the rotor that must be energized, there are samples of this type of sensor with brushes, but modern **resolvers** and **synchros** are available in a brushless format. They have

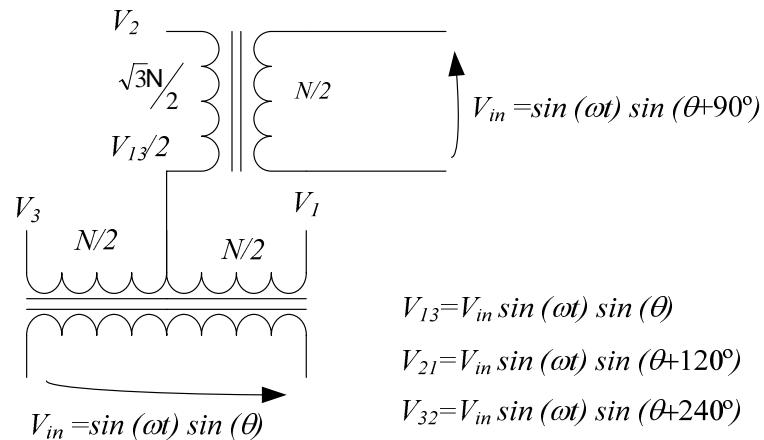


Figure 5.27: Scott-T transformer.

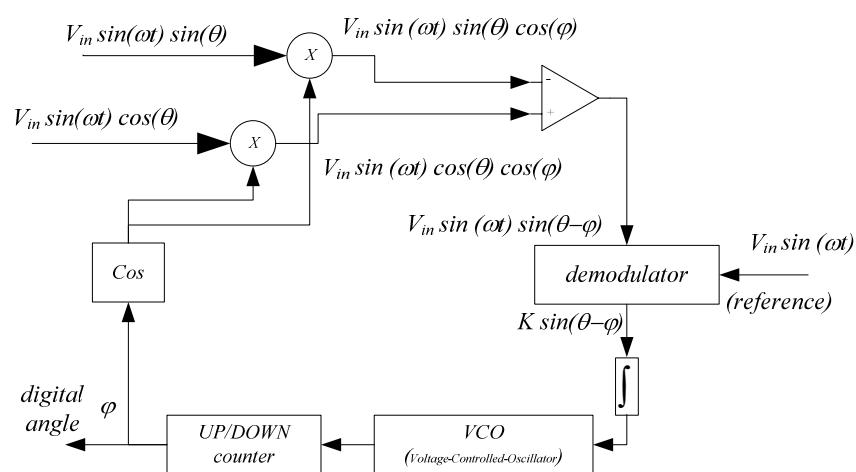


Figure 5.28: Resolver-to-Digital converter.

low inertia and good dynamic response. Finally, notice they provide us with an absolute measurement.

### 5.5.1.3 Inductosyn

The **inductosyn** resembles a **resolver**. In fact, the operation is the same: one winding energized with a sine wave; there are two windings where two  $90^\circ$ -phase voltages are induced. But in this case, the **inductosyn**<sup>3</sup> senses linear displacements, not rotary position. The energized winding is in the stator, and the induced windings are in the slider.

The remaining description about its working principle is similar to the description of the **resolver**, given above, so it is not necessary.

### 5.5.1.4 LVDT

**LVDT** is an acronym and stands for Linear Variable Differential Transformer. The wiring diagram is shown in figure 5.29. An LVDT resembles a simple transformer with one

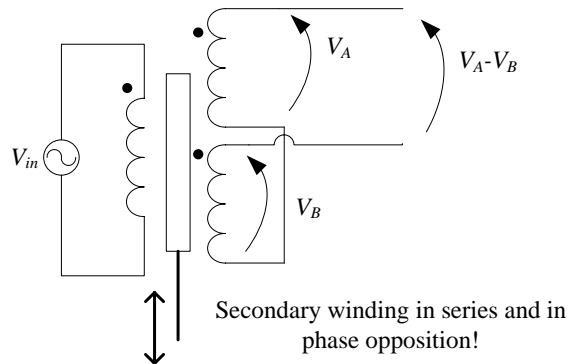


Figure 5.29: LVDT

primary winding, two secondary windings and a magnetic core. However, the LVDT differs from common transformers in that its core is a slider. The primary winding is driven by an AC reference voltage. The voltages induced in the secondary windings ( $V_A$ ,  $V_B$ ) depend on the position of the slider because the magnetic coupling, between the primary and secondary windings, varies:

- When the core is centred, the voltages in the secondary windings ( $V_A, V_B$ ) are equal in amplitude and phase but with opposite sign. Then the output voltage ( $V_A - V_B$ ) is equal to zero.
- When the core is moved off centre, the induced voltages ( $V_A, V_B$ ) are different and the output voltage ( $V_A - V_B$ ) will be different from zero.

<sup>3</sup>Trademark of Farrand Controls, Inc.

- The induced voltage increases in the secondary winding that is nearest to the slider and decreases in the secondary winding that is furthest.
- The result is a differential output voltage which varies linearly with the position of the core.

The LVDT offers good accuracy, linearity, sensitivity, good dynamic response, frictionless operation and ruggedness. However, it is quite expensive, not easy to use and the output signal must be processed as it is described in figure 5.30. Another disadvantage is that LVDT is only available for small strokes.

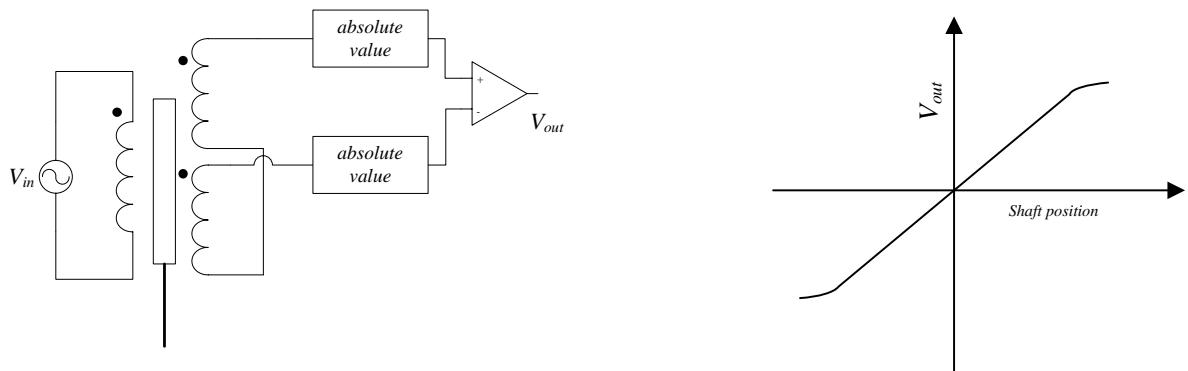


Figure 5.30: LVDT output signal conditioner (left) and signal characteristic response.

### 5.5.1.5 Optical encoder

#### *Incremental Encoder*

The **incremental encoder** is one of the most popular position sensor used in robotics. It is composed of a LED (Light Emitter Diode), a phototransistor and a slotted disk (5.31). The disk's slots are arranged in a circular fashion. The disk is fixed to the rotating part of which the displacement is being measured. The phototransistor detects when a light beam, from the LED, crosses the slotted disk. The encoder output is a stream of square pulses of which frequency depends on the rotational speed of the slotted disk. As is described in the previous paragraph, this type of encoder does not provide an absolute position, that is to say, we can only start to count pulses during specific time intervals. Knowing the number of slots in the disk, the actual displacement can be computed:

$$\text{encoder}_{\text{resolution}} = \frac{\# \text{slots}}{2\pi} \quad (5.6)$$

Now, we must address one problem with this type of encoder: with this design we can only detect that the shaft (encoder) is rotating but we have no information about the direction of rotation. This problem can be solved with a second set of slots at a 90°

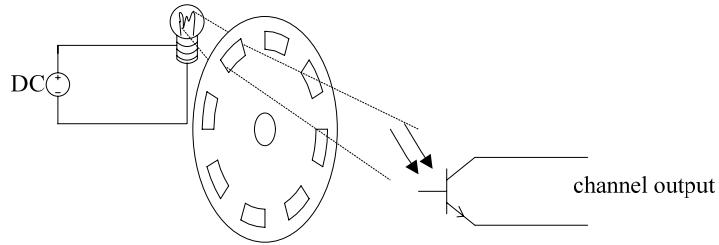


Figure 5.31: Encoder's working principle

angle of phase displacement (figure 5.32). Each set of slots is called 'channel'. So we distinguish the outer set (channel A) from the inner set (channel B). This type of encoder is called ***quadrature encoder***. The direction of rotation can be obtained by analyzing



Figure 5.32: Incremental Encoder: one channel (left), two channel (quadrature encoder, right).

which signal is delayed, thus if the channel A is delayed, then one direction is assumed and if the channel B is delayed, then the opposite direction is assumed (figure 5.33). Thanks to the inner (or second) channel, the resolution of the ***quadrature encoder*** is multiplied by four compared to the resolution of a ***single-channel encoder*** with the same number of slots per revolution and per channel.

At this point, it is obvious the encoder is a digital sensor. The standard encoder resolutions are about 500x4 slots per spin.

There are also models with a third channel called index . This channel only give one pulse every  $2\pi$  radians.

In systems with an actuator-gear train, we can easily enhance virtually the encoder resolution if we assemble the encoder between the actuator and the gear train (and not after the set of gears). The actual encoder resolution is virtually multiplied by N (where N is the reduction ratio). In this case, the index-channel becomes useless because it send N pulses every spin.

To summarize, the encoder offers good resolution, without sliding parts, and good thermal stability. But they are unsuitable for dusty environments. Other disadvantages are the price, the performance at low speeds, the breakability and the needed of a signal

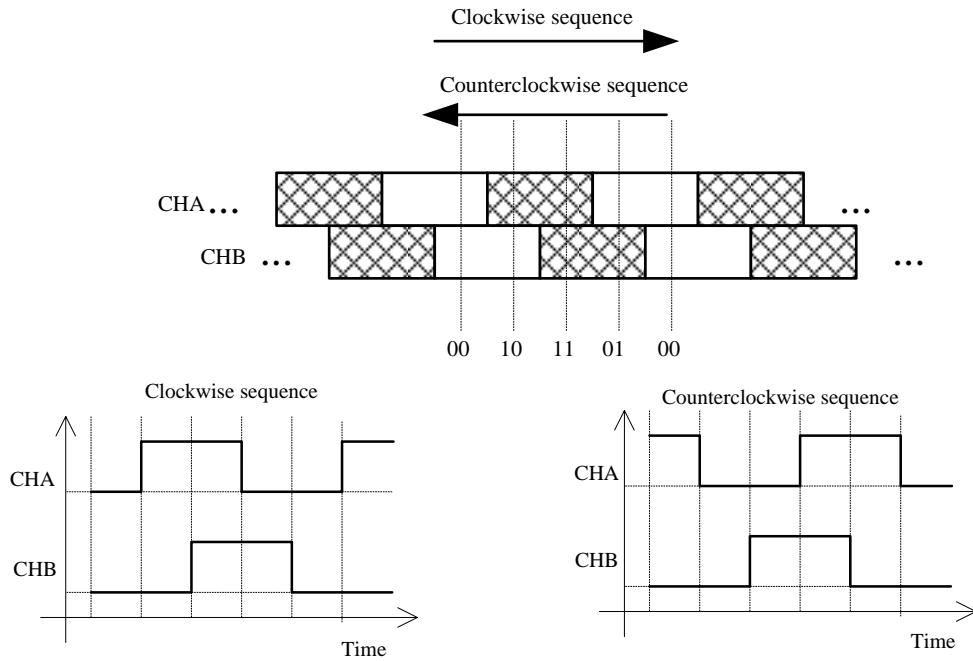


Figure 5.33: Time diagram of a quadrature encoder.

conditioner for the output signal.

### Absolute Encoder

One of the disadvantages of incremental encoders is the impossibility to know the absolute position. Due to this, a type of initialization is needed. Another way to overcome this problem is the use of **absolute encoders**. The working principle of **absolute encoders** is quite the same than **incremental encoders**, but here, the number of channels is higher. The channels of an absolute encoder must be interpreted as a set of bits in which the shaft position is coded (so they should be read in a fixed order).

The figure 5.34 shows the disk of a four-bit absolute encoder (left) and how the output channels must be interpreted as a nibble (right). Notice that the code used is the **gray code**<sup>4</sup>.

The last remarkable feature of absolute encoders is that they are more expensive than their incremental counterparts.

#### 5.5.1.6 Optical Ruler

The **optical ruler** is based on the same working principle as the incremental encoder, but it is designed to measure linear displacements.

<sup>4</sup> gray code: a means of counting in digital systems. The main feature is the next count only differs from the previous one by one bit.

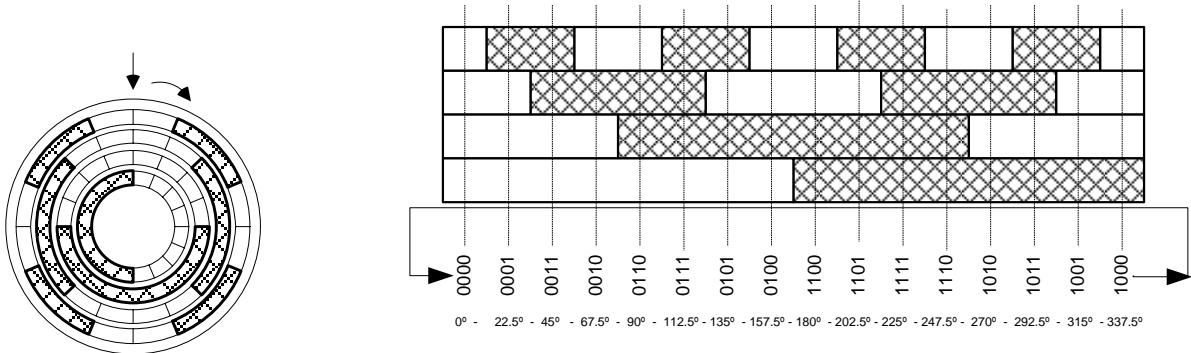


Figure 5.34: Gray code used in absolute encoders.

### 5.5.1.7 Tachometer

The **tachometer** is a DC motor but used as DC voltage generator. The tachometer is coupled to the shaft of which speed is measured. The tachometer outputs is a voltage proportional to shaft speed.

The tachometer offers durability, and is easy to use. But it is noisy, heavy and may have brushes.

## 5.5.2 External Sensors

External sensors are widely used in robot applications. A common classification is based on the measured variable:

- Proximity
  - Inductive
  - Capacitive
  - Hall effect
  - Optical
  - Ultrasonic
- Force/torque
  - Strain gauges
  - Piezoelectric
- Vision
  - CCD camera
- Temperature
  - PTC/NTC

- Thermocouple...

However, in robotics, the most common classification is based on the architecture:

- Complex or multi-value sensors:
  - multi-axis force sensors,
  - touch or pressure sensitive arrays,
  - cameras and vision systems,
  - laser range finders,
  - arrays of ultrasonic sensors,
  - etc.
- Simple or single value sensors
  - touch or contact sensors,
  - single axis force sensors,
  - make-and-break light beams,
  - etc.

Simple sensors (and every axis of multi-value sensors) can be further divided into:

- single-bit or binary sensors, and
- multi-bit or analogue sensors, which need A/D converters and produce 8bit or 16bit values, for example.

Complex sensors may be really complete subsystems in themselves that typically need special purpose hardware and software and interfaces to robot control systems. They are also expensive, in general.

Simple sensors can often be connected directly to the robot control system, which today typically have a number of binary and analogue input channels. Simple sensors are also typically low-cost.

The best sensors to use, with respect to these aspects of reliability, calibration, and precision, and with respect to cost, are binary sensors. They are typically very reliable because they are mechanically and electrically simple and easy to make. They do not need calibration or, at times, only a simple threshold setting. They also have absolute precision, assuming they are working properly.

Complex sensors are often used as a complete subsystem to provide data about objects involved in the task or the end-point of the robot, or the tool it is using: global position and orientation, for example. This data can then be used in a robot-level program directly, through sometimes a special subroutine or function is provided so that the sensor can be called when new data is required from it.

Simple sensors are more usually used to provide values in conditional statements in a program. These are either movement commands or other action commands, which might take the following forms:

```
move to CONFIG-1 while SENSOR-1 = 1; }
```

or

```
move in direction DIRECTION-1 until SENSOR-1 = 0; }
```

These kind of commands are sometimes called **guarded motions** because they depend upon real-time sensor data.

Simple sensor data can also be used in the conditions parts of if-statements:

```
if SENSOR-f = 0 do write(''Failed to acquire part'') and FAIL;
```

These sensors are commonly used to detect end-strokes, count parts and/or synchronize the program with the arrival and/or departure of parts.

In general, the more sensors that are used, the more complex the programming becomes, the more difficult it becomes to properly test and debug the program, and the more difficult becomes the prediction of its execution time.

All or any of these consequences of using even simple binary sensors, may be reason enough for not using sensors at all, or for using fewer of them, and/or using more robust robot movements and actions, even if this makes the program take longer to complete. A constant but slower execution time is often better than a faster program, but one of which execution time can vary depending upon run-time sensor data. This is especially true when the robot has to work in coordination with other robots or other machine in the same **work cell**.

## 5.6 End-Effectors and Terminal Devices

The last basic type of basic component of robots that we will consider here are called **end-effectors** or **terminal devices**. These are the tools, grippers, spot welding guns, paint sprayers, glue guns, etc. that are attached to the end of the robot manipulator arm.

What type of end-effector a particular robot has will depend upon the type of task it is programmed to carry out. There are no completely general robot end-effectors<sup>5</sup>, and often they are specially designed for the particular task.

End-effectors can sometimes have their own degrees of freedom. Mostly these are rotational, but they can also be translational, or combinations of rotational and translational degrees of freedom.

---

<sup>5</sup> The most general gripper is the human hand.

These extra end-effector DoFs have the effect of increasing the total number of DoFs of the robot manipulator arm, but they are not usually counted in the total. So, a robot manipulator with 6 DoFs that is fitted with an end-effector which itself has 2 DoFs is still referred to as a 6-DoF robot manipulator, though it effectively forms a 8-DoF system.

End-effectors or terminal devices should be:

- very reliable,
- low weight,
- as small and compact as possible, and
- easy to change and maintain.

The most popular grippers have pneumatic actuators, because this type of actuator has a good behaviour under a stall condition and the force/torque they exerted can be limited easily.

## 5.7 Some Things to Think About

1. Why are electric motors most often used as the actuators in the joint systems of robots?
2. What is the effect of any bending that occurs in the elements of a serial geometry robot manipulator arm?
3. What is the effect of any play or backlash in the transmission systems of the joints in a serial geometry robot manipulator arm?
4. What are the advantages of adding degrees of freedom in a robot end-effector?
5. What are the disadvantages of adding degrees of freedom in a robot end-effector?
6. What is the most universal end-effector?
7. Try to find in internet samples of end-effectors.

## **Part III**

# **Robot Mathematical Modelling and Control**



## Chapter 6

# Coordinate frames and homogeneous transformation

---

### 6.1 Introduction

Before going on to consider robot control, we need to be able to formally describe and model the geometry and **kinematics** (movement) of robot manipulator arms. For this we first need some kind of coordinate system.

In particular, we need a formal and precise way of relating the position and orientation of the end-point,  $P_e$ , of the robot to the base of the robot, which is fixed to the ground. We also need a way of relating both of these to the position and orientation of objects in the robot's workspace.

In this part of the course we will review several different kinds of coordinate systems for describing position and rotation in three dimensional space. How these are combined to form **homogeneous coordinates** will then be explained, together with their use in the context of robot arms.

Here it is useful to introduce some terminology which we will use throughout this part of the course. We will use the term **location** to mean both the **position and orientation** of something in space, for example: the robot base, the point  $P_e$ , or an object in the workspace.

The figure 6.1 shows some coordinate systems that can be defined around a robot:

- $\{0\}$ , the universal frame,
- $\{B\}$ , the robot's base coordinate frame,
- $\{w\}$ , the wrist's coordinate frame,
- $\{e\}$ , the end-effector's coordinate frame,
- $\{O\}$ , the target object's coordinate frame,
- $\{T\}$ , the task's coordinate frame.

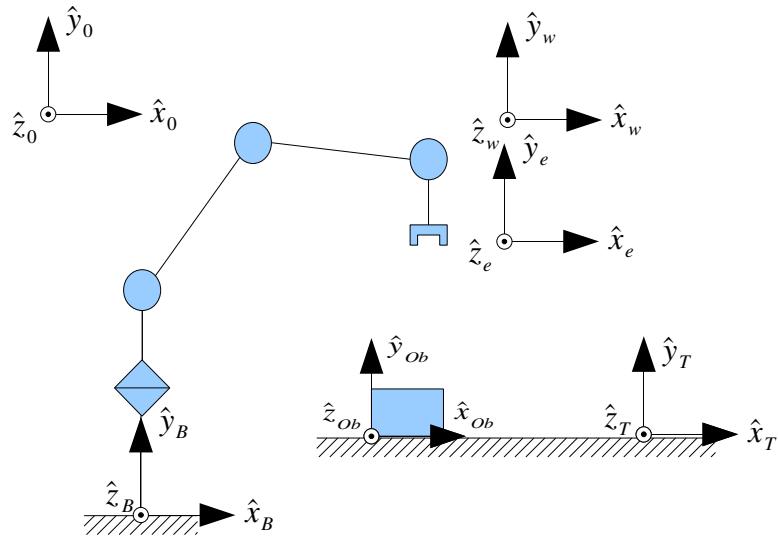


Figure 6.1: Relationship among different coordinate systems related to robot.

## 6.2 The Representation of Position in 3D Space

To completely and uniquely define the position of a point in 3D space, we need to specify particular values that fix the three degrees of freedom of position.

If we want to define the position of an object in space, we must first define some point associated with this object (for example, its centre of mass, or some other convenient point on its surface) and then define the position in space of this **object reference point**.

There are three common coordinate systems used to define position in 3D space.

**Cartesian Coordinates** (figure 6.2), in which the position of a point,  $P_e$ , with respect to the coordinate system  $\{A\}$ , is defined by the three values  ${}^A P_{e_x}$ ,  ${}^A P_{e_y}$  and  ${}^A P_{e_z}$ , or by the vector  ${}^A P_{e_{3x1}} = [{}^A P_{e_x} \ {}^A P_{e_y} \ {}^A P_{e_z}]^T$

**Cylindrical Coordinates** (figure 6.3), in which the position of a point,  $P_e$ , with respect to the coordinate system  $\{A\}$ , is defined by the three values:  ${}^A P_{e_r}$ , the magnitude of the projection of the vector  $P_e$  onto the plane  $\hat{x}_A \hat{y}_A$ ;  ${}^A P_{e_\theta}$ , the angle between the axis  $\hat{x}_A$  and the projection of the vector  $P_e$  onto the plane  $\hat{x}_A \hat{y}_A$ ; and  ${}^A P_{e_z}$ , the magnitude of the projection of the vector  $P_e$  onto the axis  $\hat{z}_A$ .

**Spherical Coordinates** (figure 6.4), in which the position of a point,  $P_e$ , with respect to the coordinate system  $\{A\}$ , is defined by the three values:  ${}^A P_{e_R}$ , the magnitude -or distance- of  $P_e$  from the origin  $\{A\}$ ;  ${}^A P_{e_\theta}$ , the angle between the axis  $\hat{x}_A$  and the projection of the vector  $P_e$  onto the plane  $\hat{x}_A \hat{y}_A$ ; and  ${}^A P_{e_\phi}$ , the angle between the axis  $\hat{z}_A$  and the vector  $P_e$ .

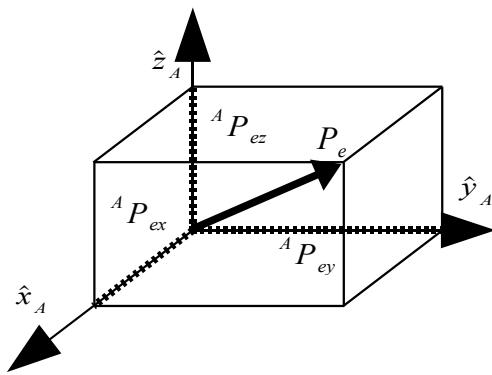


Figure 6.2: Cartesian coordinates of a point.

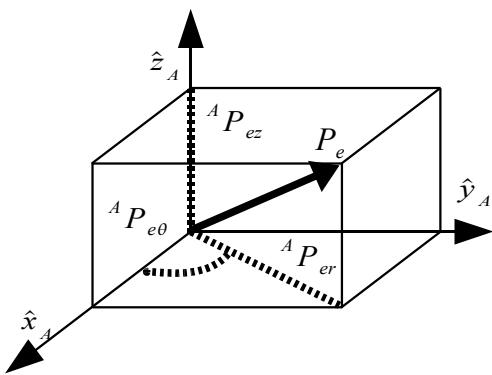


Figure 6.3: Cylindrical coordinates of a point.

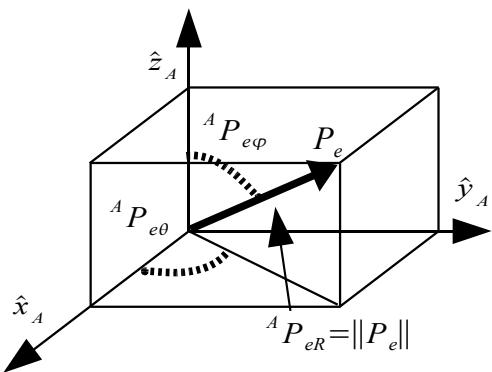


Figure 6.4: Spherical coordinates of a point.

## 6.3 The Representation of Orientation in Space

To be able to define, in a simple way, the orientation of an object, or the end-point of a robot arm,  $P_e$  with respect to some reference system of coordinate axes, we attach a new coordinate reference system to the object or point, and then define the orientation relationship between the two reference systems.

In order to keep things simple, we will assume that the origins of the two frames of reference, the two coordinate systems, are coincident at the same point in 3D space. We will therefore have no translation component in the relationship between the two coordinate systems, only orientation.

### 6.3.1 Rotation Matrices

Suppose we have two coordinate reference systems,  $\{A\}$  and  $\{B\}$ , which have coincident origins, and that the system  $\{A\}$  is fixed with respect to the world, and that the system  $\{B\}$  is rigidly attached (fixed to) the object whose orientation we want to define with respect to  $\{A\}$ , see figure below.

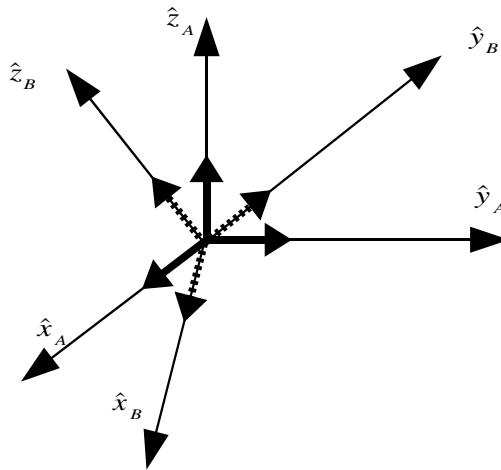


Figure 6.5: Two different frames with different orientation, but the same origin.

Then, we can define **unit vectors** in the directions of the three axes of the reference system  $\{A\}$ :  $\hat{x}_A$ ,  $\hat{y}_A$ , and  $\hat{z}_A$ ; and we can define **unit vectors** in the directions of the three axes of the reference system  $\{B\}$ :  $\hat{x}_B$ ,  $\hat{y}_B$ , and  $\hat{z}_B$ .

A vector  $P_e$  in space can then be defined with respect to either the system  $\{A\}$

$${}^A P_e = \begin{bmatrix} {}^A P_{e_x} \\ {}^A P_{e_y} \\ {}^A P_{e_z} \end{bmatrix} = P_e (\hat{x}_A + \hat{y}_A + \hat{z}_A) \quad (6.1)$$

or the system  $\{B\}$ .

$${}^B P_e = \begin{bmatrix} {}^B P_{e_x} \\ {}^B P_{e_y} \\ {}^B P_{e_z} \end{bmatrix} = P_e (\hat{x}_B + \hat{y}_B + \hat{z}_B) \quad (6.2)$$

We can relate the two vectors  ${}^A P_e$  and  ${}^B P_e$  by:

$${}^A P_e = {}^A R_B {}^B P_e \quad (6.3)$$

where

$${}^A R_B = \begin{bmatrix} \hat{x}_A \hat{x}_B & \hat{x}_A \hat{y}_B & \hat{x}_A \hat{z}_B \\ \hat{y}_A \hat{x}_B & \hat{y}_A \hat{y}_B & \hat{y}_A \hat{z}_B \\ \hat{z}_A \hat{x}_B & \hat{z}_A \hat{y}_B & \hat{z}_A \hat{z}_B \end{bmatrix} = \begin{bmatrix} {}^A \hat{x}_B & {}^A \hat{y}_B & {}^A \hat{z}_B \end{bmatrix} \quad (6.4)$$

is the **rotation matrix** that defines the orientation of the coordinate system  $\{B\}$  with respect to the coordinate system  $\{A\}$ .

### 6.3.1.1 Example of rotation in 2D space

Let's suppose two 2D-coordinate systems:  $\{A\}$  and  $\{B\}$ . System  $\{B\}$  is rotated an angle  $\theta$  equal to  $\pi/2$  radians (figure 6.6). If the coordinates of a point  $P$  with respect to frame  $\{B\}$  are  ${}^B P = [2 \ -3]^T$ , we can calculate the coordinates of that point referred to system  $\{A\}$ , that is,  ${}^A P$ .

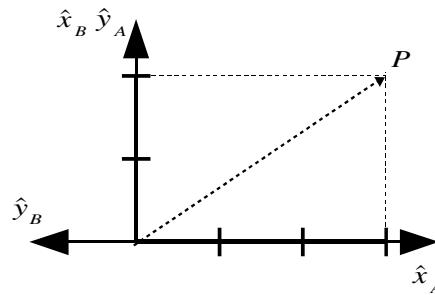


Figure 6.6: Sample of rotation in 2-D.

The solution can be obtained by simple inspection of the axis of the system  $\{B\}$ ,  $\hat{x}_B$ ,  $\hat{y}_B$  and how they can be represented in system  $\{A\}$ :

$${}^A \hat{x}_B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$${}^A\hat{y}_B = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

Then, according to equation 6.4, the rotation matrix is:

$${}^A R_B = [{}^A \hat{x}_B \quad {}^A \hat{y}_B] = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Applying equation 6.3, we finally get:

$${}^A P = {}^A R_B {}^B P = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Having a rotation angle of  $\theta$ , proves that the general expression for the 2D-rotation matrix is:

$${}^A R_B = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (6.5)$$

### 6.3.2 Properties of the Rotation Matrix

The matrix  $R$  is also sometimes called the ***direct cosine matrix***, and it is an ***orthogonal matrix***, which means that its inverse is equal to its transpose, i.e.:

$${}^A R_B^{-1} = {}^A R_B^T = {}^B R_A \quad (6.6)$$

But as the rotation matrix is built with unitary vectors, all its rows/columns have a unitary modulus. Thus, the rotation matrix is also a ***normal matrix***.

The matrices that are normal and orthogonal are ***orthonormal matrices***.

The rotation matrices are applied to solve three kind of transformation/mapping problems:

- quantifying the orientation of a frame with respect to another,
- rotating a vector by a given angle, and
- coordinate transformation (this implies a rotation of the coordinate system itself).

### 6.3.3 Basic 3D rotation matrices

The principle use of this rotation matrix is in the definition how to represent the orientation of an object (or coordinate system) that is rotated around just one of the axes of the fixed reference frame,  $\{A\}$ .

Applying this condition, we can define three different cases, one each for rotation with respect to one of the axes of  $\{A\}$ , and thus get three different rotation matrices.

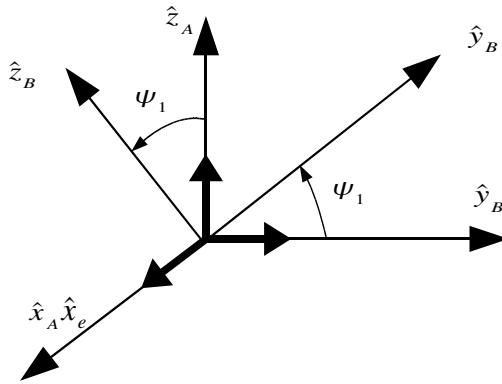


Figure 6.7: Rotation around  $\hat{x}_A$  axis.

#### Rotation around $\hat{x}_A$

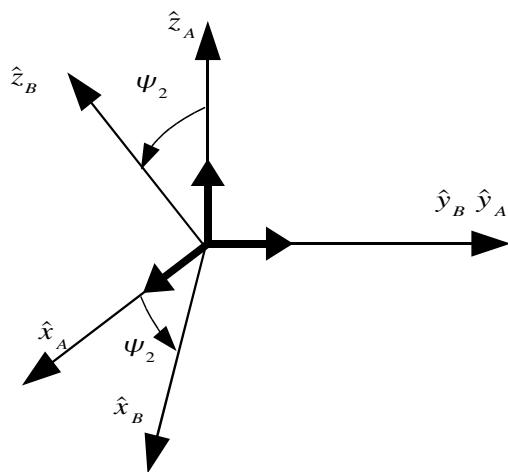
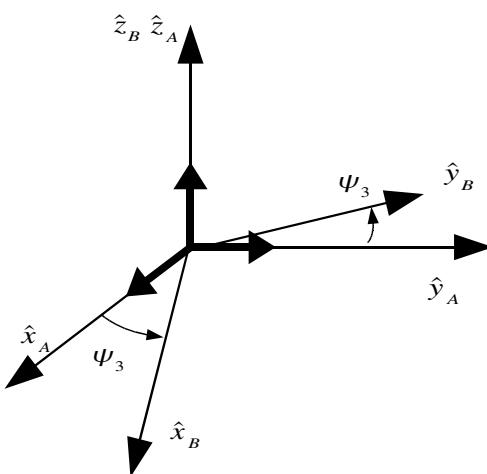
$$\text{Rot } (\hat{x}, \psi_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 \\ 0 & \sin \psi_1 & \cos \psi_1 \end{bmatrix} \quad (6.7)$$

#### Rotation around $\hat{y}_A$

$$\text{Rot } (\hat{y}, \psi_2) = \begin{bmatrix} \cos \psi_2 & 0 & \sin \psi_2 \\ 0 & 1 & 0 \\ -\sin \psi_2 & 0 & \cos \psi_2 \end{bmatrix} \quad (6.8)$$

#### Rotation around $\hat{z}_A$

$$\text{Rot } (\hat{z}, \psi_3) = \begin{bmatrix} \cos \psi_3 & -\sin \psi_3 & 0 \\ \sin \psi_3 & \cos \psi_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.9)$$

Figure 6.8: Rotation around  $\hat{y}_A$  axis.Figure 6.9: Rotation around  $\hat{z}_A$  axis.

### 6.3.4 The Composition of Basic Rotation Matrices

We can combine the three basic rotation matrices defined above [Rot ( $\hat{x}, \psi_1$ ), Rot ( $\hat{y}, \psi_2$ ) and Rot ( $\hat{z}, \psi_3$ )], to represent any orientation of the coordinate system  $\{B\}$  with respect to the coordinate system  $\{A\}$ .

For example, let's suppose coordinate system  $\{B\}$  coincides with system  $\{A\}$  in the begining and if we apply to  $\{B\}$  a rotation  $\psi_1$  around  $\hat{x}_A$ , followed by a rotation  $\psi_2$  around  $\hat{y}_A$ , and then a rotation  $\psi_3$  around  $\hat{z}_A$ , we can represent the final orientation of  $\{B\}$  with respect to  $\{A\}$  by:

$${}^A R_B = \text{Rot} (\hat{z}, \psi_3) \text{ Rot} (\hat{y}, \psi_2) \text{ Rot} (\hat{x}, \psi_1) \quad (6.10)$$

and thus,

$${}^A R_B = \begin{bmatrix} \cos \psi_3 & -\sin \psi_3 & 0 \\ \sin \psi_3 & \cos \psi_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \psi_2 & 0 & \sin \psi_2 \\ 0 & 1 & 0 \\ -\sin \psi_2 & 0 & \cos \psi_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 \\ 0 & \sin \psi_1 & \cos \psi_1 \end{bmatrix} \quad (6.11)$$

which gives:

$${}^A R_B = \begin{bmatrix} c_3c_2 & c_3s_2s_1 - s_3c_1 & c_3s_2c_1 + s_3s_1 \\ s_3c_2 & s_3s_2s_1 + c_3c_1 & s_3s_2c_1 - c_3s_1 \\ -s_2 & c_2s_1 & c_2c_1 \end{bmatrix} \quad (6.12)$$

where  $s_i = \sin \psi_i$ ,  $c_i = \cos \psi_i$  with  $i = 1, 2, 3$

Therefore we can say that any measured orientation between two coordinate systems can be decomposed in a series of three basic rotations. These basic rotation are named **Euler Angles**. When these basic rotations are a rotation  $\psi_1$  around  $\hat{x}_A$ , followed by a rotation  $\psi_2$  around  $\hat{y}_A$ , and then a rotation  $\psi_3$  around  $\hat{z}_A$ , then they are called **YAW-PITCH-ROLL** angles.

This decomposition can be done provided that the nine elements of a rotating matrix are not independent due to the fact that the rotating matrices are orthonormal. This means that given:

$${}^A R_B = [{}^A \hat{x}_B \quad {}^A \hat{y}_B \quad {}^A \hat{z}_B] \quad (6.13)$$

the following equations are true:

$$\left. \begin{array}{l} \left. \begin{array}{l} \|\hat{x}_B\| = 1 \\ \|\hat{y}_B\| = 1 \\ \|\hat{z}_B\| = 1 \end{array} \right\} \rightarrow \text{normal matrix} \\ \left. \begin{array}{l} \hat{x}_B \hat{y}_B = 0 \\ \hat{x}_B \hat{z}_B = 0 \\ \hat{z}_B \hat{y}_B = 0 \end{array} \right\} \rightarrow \text{orthogonal matrix} \end{array} \right\} \rightarrow \text{orthonormal matrix} \quad (6.14)$$

#### 6.3.4.1 Rotation around the Fixed Frame vs. Rotation around the Mobile Frame

Notice that, up to now, the basic rotations are always in reference to frame  $\{A\}$ , also named “the fixed frame”. But there is no restriction to refer them to frame  $\{B\}$ , “the mobile frame”. Then, if we repeat the last exercise, but change the reference, we have a coordinate system  $\{B\}$  that coincides with the system  $\{A\}$  at the beginning and then we apply to  $\{B\}$  a rotation  $\psi_1$  around  $\hat{x}_e$ , followed by a rotation  $\psi_2$  around  $\hat{y}_e$ , and then a rotation  $\psi_3$  around  $\hat{z}_e$ . In this case, instead of pre-multiplying the basic rotation matrices, we have to post-multiply, giving the following expression:

$${}^A R_B = \text{Rot}(\hat{x}, \psi_1) \text{ Rot}(\hat{y}, \psi_2) \text{ Rot}(\hat{z}, \psi_3) \quad (6.15)$$

This result will be different from equation 6.12. It is important to note the order in which this combination is formed. The product of matrices is not commutative, in general, so changing the order would result in a different representation of the final orientation of  $\{B\}$  with respect to  $\{A\}$ .

If the first rotation we apply to  $\{B\}$  defined with respect to  $\{A\}$  is a rotation  $\psi_3$  around  $\hat{z}_A$ , followed by a rotation  $\psi_2$  around  $\hat{y}_A$ , and then a rotation  $\psi_1$  around  $\hat{x}_A$ , then we obtain a rotation matrix equal to the equation 6.15.

To sum up, the composition of rotation matrices depends on the frame we use as reference. If we set as reference the fixed frame, we have to pre-multiply and if the reference is the the mobile frame, then we have to post-multiply.

Due to the number of combinations, 24 different sets of Euler Angles can be defined about the fixed reference and another 24 sets when the rotations are in reference to the mobile frame.

In robotics, the most common sets are **Roll-Pitch-Yaw mobile Euler Angles** and **Roll-Pitch-Roll mobile Euler Angles**. Do you think it is a coincidence? The answer is no. Think why. Bear in mind that: Yaw=  $\text{Rot}(\hat{x}, \psi_1)$ , Pitch=  $\text{Rot}(\hat{y}, \psi_2)$  and Roll=  $\text{Rot}(\hat{z}, \psi_3)$ .

#### 6.3.5 Formulation Singularity in Euler Angles

From equation 6.12 it can be easily proven that the rotation matrix can always be obtained from a given set of Euler Angles, without any restriction.

The inverse problem, that is to say, to obtain the Euler Angles from a given rotation matrix, requires a little work. Given the rotation matrix of system  $\{B\}$ , in reference to a system  $\{A\}$ :

$${}^A R_B = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (6.16)$$

The values for the Roll-Pitch-Yaw mobile Euler Angles can be obtained equating the previous expression with equation 6.12:

$${}^A R_B = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c_3 c_2 & c_3 s_2 s_1 - s_3 c_1 & c_3 s_2 c_1 + s_3 s_1 \\ s_3 c_2 & s_3 s_2 s_1 + c_3 c_1 & s_3 s_2 c_1 - c_3 s_1 \\ -s_2 & c_2 s_1 & c_2 c_1 \end{bmatrix} \quad (6.17)$$

then, we have

$$\begin{aligned} \sin \phi_2 &= -r_{31} & \cos \phi_2 &= \pm \sqrt{1 - \sin^2 \phi_2} \\ \sin \phi_3 &= r_{21} / \cos \phi_2 & \cos \phi_3 &= r_{11} / \cos \phi_2 \\ \sin \phi_1 &= r_{32} / \cos \phi_2 & \cos \phi_2 &= r_{33} / \cos \phi_2 \end{aligned} \quad (6.18)$$

If  $\cos \phi_2 = 0$  ( $\phi_2 = 2\pi n + \pi/2$  with  $n \in \mathbb{N}$ ) the problem has no solution. There is a **formulation singularity**.

Every set of Euler Angles has a formulation singularity for a given value of  $\phi_2$ . The value presented above for which this singularity occurs is specific for this set of Euler Angles (that is, the Roll-Pitch-Yaw Euler Angles) since we have used the rotation matrix for this set.

The major consequence of the singularity is that there is no unique solution for the Euler Angles that produces the given matrix. It can only be said that  $\phi_1 + \phi_3 = \pi/2$ .

The ways to avoid the formulation singularity are:

- Constrain robot movements to prevent crossing the region of the formulation singularity.
- Use another set of Euler Angles. In this case, the formulation singularity will be moved to another region in the robot workspace.
- Don't use the Euler Angles to quantify the orientation and try other techniques that do not present the formulation singularity.

The formulation singularity has no relationship with mechanical singularity (see chapter 8).

### 6.3.6 Rotation Axis- Rotation Angle Representation

Due to the formulation singularity of the Euler Angles, other representation models are defined that are based on 4 parameters. The fourth parameter is normally redundant. But when the Euler Angle representation fails due to the formulation singularity, this fourth (redundant) parameter becomes no-redundant and solves the ambiguity.

This is the case of the Rotation Axis- Rotation Angle Representation (figure 6.10). This representation consists of defining a unitary vector ( $\vec{u}$ , with  $\|\vec{u}\| = 1$ ) that points the rotation axis and the fourth parameter is the radians that must be rotated ( $\varphi$ ).

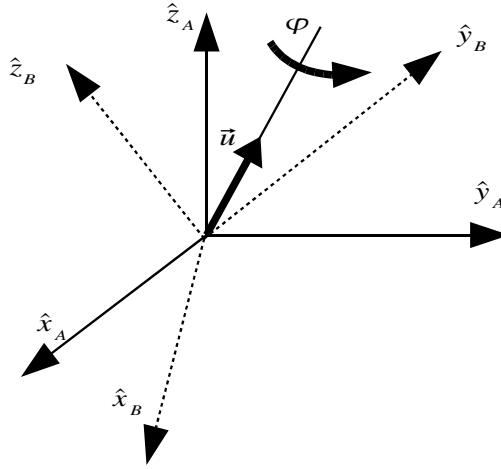


Figure 6.10: *Rotation Axis- Rotation Angle* Representation of orientation.

The expression for the rotation matrix is:

$${}^A R_B = Rot(\vec{u}, \varphi) = \begin{bmatrix} -(u_y^2 + u_z^2)\alpha + 1 & u_x u_y \alpha - u_z \beta & u_x u_z \alpha + u_y \beta \\ u_x u_y \alpha + u_z \beta & -(u_x^2 + u_z^2)\alpha + 1 & u_y u_z \alpha - u_x \beta \\ u_x u_z \alpha - u_y \beta & u_y u_z \alpha + u_x \beta & -(u_x^2 + u_y^2)\alpha + 1 \end{bmatrix} \quad (6.19)$$

where:  $\alpha = 1 - \cos \varphi$ ,  $\beta = \sin \varphi$ .

### 6.3.7 Quaternions or Euler Parameters

The *Rotation Axis-Rotation Angle* Representation has the problem that not all the parameters have the same dimension. This inconvenience has the side effect that the order of the parameter values are not the same. This problem is overcome by the **Euler Parameters** (also called as **Quaternions**).

The definition of the Quaternions is based on the Rotation Axis-Rotation Angle

Representation:

$$\begin{aligned} e_0 &= \cos(\varphi/2) \\ e_1 &= u_x \sin(\varphi/2) \\ e_2 &= u_y \sin(\varphi/2) \\ e_3 &= u_z \sin(\varphi/2) \end{aligned} \quad (6.20)$$

Quaternions behave as a vector in the 4D space and have the interesting property that:

$$\|e\| = \sqrt{e_0^2 + e_1^2 + e_2^2 + e_3^2} = 1 \quad (6.21)$$

Quaternions can also be envisioned as an extension of complex numbers:

$$\vec{e} = e_0 + e_1 \vec{i} + e_2 \vec{j} + e_3 \vec{k} \quad (6.22)$$

that verify:  $\vec{i}\vec{i} = \vec{j}\vec{j} = \vec{k}\vec{k} = \vec{i}\vec{j}\vec{k} = -1$

## 6.4 Homogeneous Coordinates

### 6.4.1 Position and Orientation in Space

Now it is time to combine the translation and rotation transformations into one. So, let us consider two coordinate systems ( $\{A\}$  and  $\{B\}$ ) in which the origins do not coincide and the coordinated axes of frame  $\{B\}$  are rotated with respect to the frame  $\{A\}$  (figure 6.11)

The vector  ${}^A P_e$  locates the point  $P_e$  with respect to the origin of the coordinate system  $\{A\}$ , that is  $O_A$ , and the vector  ${}^B P_e$  locates the point  $P_e$  with respect to the origin of the coordinate system  $\{B\}$ , that is  $O_B$ . Finally, the vector  ${}^A O_{AB}$  connects the two origins.

It can easily be seen that the vector  ${}^A P_e$  is the summation of  ${}^B P_e$  and  ${}^A O_{AB}$ . Thus,

$${}^A P_e = \underbrace{{}^A R_B {}^B P_e}_{\text{rotation}} + \underbrace{{}^A O_{AB}}_{\text{translation}} \quad (6.23)$$

Notice that to formally write the equation, we have to premultiply the rotation matrix because we cannot sum two different sets of coordinates that do not relate to the aligned axes.

From equation 6.23, we can conclude that the algebra used to rotate is different than the algebra needed to translate. This makes the operations, when several transformation

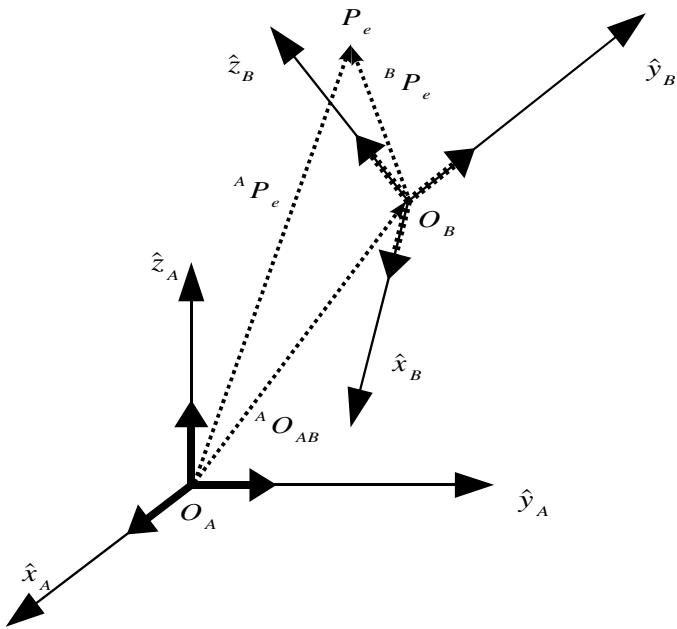


Figure 6.11: The relation between frames  $\{A\}$  and  $\{B\}$  involves both a translation plus a rotation.

series should be combined, extremely difficult. This problem can be overcome thanks to the **homogeneous transformation**, but first we have to understand the concept of **homogeneous coordinates**.

### 6.4.2 Homogeneous Coordinates

So far we have seen different ways of representing the position and the orientation in space of objects or systems of reference. But none of these methods allow us to represent both position and orientation at the same time. None of them can be used to represent the **location**<sup>1</sup> of an object or reference system in space.

To present the location of something in space we need to use **homogeneous coordinates**. These are defined using a  $4 \times 4$  matrix, and are a representation of coordinates in a four dimensional space.

In other words, the 3D space is embedded in a 4D space in such a way that a position vector in 3D space,  $P_e$ , is represented as:

$$P_e = \begin{bmatrix} w \cdot P_{e_x} \\ w \cdot P_{e_y} \\ w \cdot P_{e_z} \\ w \end{bmatrix} \quad (6.24)$$

<sup>1</sup>The term *location* will be used throughout these notes to mean the position and orientation of something (an object, the endpoint, a frame, etc) in space.

where  $w$  has any arbitrary nonzero value and, in effect, represents a scaling factor of the embedding of the 3D space in the 4D space.

For example, a vector

$$P_e^{3D} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (6.25)$$

can be represented in homogeneous coordinates as a column vector:

$$P_e^{4D} = \begin{bmatrix} w \cdot a \\ w \cdot b \\ w \cdot c \\ w \end{bmatrix} \quad (6.26)$$

given that  $w$  can be any arbitrary nonzero value.

For example, the vector

$$P_e^{3D} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \quad (6.27)$$

can be represented in homogeneous coordinates as:

$$P_e^{4D} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 1 \end{bmatrix} \quad (6.28)$$

where  $w = 1$  or as:

$$P_e^{4D} = \begin{bmatrix} 4 \\ 6 \\ 8 \\ 2 \end{bmatrix} \quad (6.29)$$

where  $w = 2$  or as:

$$P_e^{4D} = \begin{bmatrix} 6 \\ 6 \\ 12 \\ 3 \end{bmatrix} \quad (6.30)$$

where  $w = 3$ , etc.

In homogeneous coordinates, null vectors are represented as  $P_e^{4D} = [0 \ 0 \ 0 \ w]$  with  $w \neq 0$ .

The vectors of type  $[a \ b \ c \ 0]^T$  with  $a, b, c \neq 0$  can be used to represent directions, since they represent vectors with infinite magnitude.

### 6.4.3 Homogeneous Transformation

As well as defining homogeneous coordinates, we can also define the **homogeneous transformation matrix**,  $T$ , which is a  $4 \times 4$  matrix which represents the transformation of a vector in homogeneous coordinates with respect to some reference system, or the relationship between two coordinate reference systems.

The homogeneous transformation matrix  $T$  is composed of four submatrices:

- a submatrix  $R_{3x3}$  which corresponds to a typical rotation matrix in 3D space,
- a submatrix  $O_{3x1}$  which corresponds to a translation vector in 3D space,
- a submatrix  $f_{1x3}$  which represents a perspective transformation, and
- a submatrix  $w_{1x1}$  which represents a global scaling factor.

The general form of the homogeneous transformation matrix is thus given by:

$$T = \begin{bmatrix} R_{3x3} & O_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} \text{rotation} & \text{translation} \\ \text{perspective} & \text{scaling} \end{bmatrix} \quad (6.31)$$

In the field of robotics we only need to represent rotation,  $R_{3x3}$ , and position or translation,  $O_{3x1}$ . We can thus define the perspective transformation to be a null (sub)matrix and the scaling factor to be one.

The form of the homogeneous transformation matrix that we will use here is thus given by:

$$T = \begin{bmatrix} R_{3x3} & O_{3x1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.32)$$

and this can be used to represent the position and orientation of an object in 3D space, with respect to some reference system, or the relationship between two reference systems,  $\{B\}$  and  $\{A\}$ , for example, in 3D space.

If the matrix  ${}^A T_B$  represents the relationship between two coordinate reference systems,  $\{A\}$  and  $\{B\}$ , it can be used to calculate the coordinates  ${}^A P_e = [{}^A P_{e_x} \ {}^A P_{e_y} \ {}^A P_{e_z}]^T$  of a vector  $P_e$  with respect to the reference system  $\{A\}$ , given the coordinates of the same vector,  $P_e$ , specified with respect to a reference system  $\{B\}$ ,  ${}^B P_e = [{}^B P_{e_x} \ {}^B P_{e_y} \ {}^B P_{e_z}]^T$ , as follows:

$${}^A P_e = \begin{bmatrix} {}^A P_{e_x} \\ {}^A P_{e_y} \\ {}^A P_{e_z} \\ 1 \end{bmatrix} = {}^A T_B {}^B P_e = {}^A T_B \begin{bmatrix} {}^B P_{e_x} \\ {}^B P_{e_y} \\ {}^B P_{e_z} \\ 1 \end{bmatrix} \quad (6.33)$$

In this example,  ${}^A T_B$  represents the *relation between two coordinate reference systems* in 3D space. But now we have to use the homogeneous coordinates.

We can also use the homogeneous transformation matrix  ${}^A T_B$  to represent the translation and rotation of a vector with respect to some fixed coordinate reference system,  $\{A\}$ .

In which case we have a vector  $P_e$ , which is rotated by  $R_{3x3}$  and translated by  $O_{3x1}$ , will be converted to the vector  $P'_e$ , which is given by:

$${}^A P'_e = {}^A T_B {}^A P_e \quad (6.34)$$

In this example  ${}^A T_B$  represents the transformation operation of translation and rotation applied to the vector  $P_e$ , defined with respect to the reference system  $\{A\}$ , to move it to the new vector  $P'_e$ .

#### 6.4.3.1 Example

Given  $O_{3x1} = [5 \ -4 \ 2]^T$ ,  ${}^B P_e = [-1 \ 2 \ -3]^T$ , and  $\psi_x = 0$ ,  $\psi_y = 0$ ,  $\psi_z = -\pi/2$  (<sup>2</sup>),  ${}^B P_e$  can be obtained as following:

- Calculate the rotation matrix  $\rightarrow {}^A R_B = rot(\hat{z}, -\pi/2) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

---

<sup>2</sup>That is to say  $\psi_1 = 0$ ,  $\psi_2 = 0$ ,  $\psi_3 = -\pi/2$  when the Y-P-R fixed Euler Angles convention is used.

- Construct the homogeneous transformation matrix  $\rightarrow {}^A T_B = \begin{bmatrix} 0 & 1 & 0 & 5 \\ -1 & 0 & 0 & -4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Calculate  ${}^A P_e = {}^A T_B {}^B P_e \rightarrow {}^A P_e = \begin{bmatrix} 7 \\ -3 \\ -1 \\ 1 \end{bmatrix}$

#### 6.4.4 Inverse Homogeneous Transformation Matrix

In this case, the homogeneous matrix does not fulfill the properties of an orthonormal matrix, so the inverse matrix is not the transpose matrix as the case of rotation matrix. However, it can easily inverted as is shown in the following:

$${}^A T_B = \begin{bmatrix} {}^A R_B & O_{AB} \\ 0 & 0 & 1 \end{bmatrix} \rightarrow {}^A T_B^{-1} = {}^B T_A = \begin{bmatrix} {}^A R_B^T & -{}^A R_B^T O_{AB} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.35)$$

Try to demonstrate this equality. It is quite easy! .

#### 6.4.5 Basic Translation Homogeneous Transformation Matrix

Suppose a reference system  $\{B\}$  is translated by a vector  $O_{AB} = [O_{AB_x} \ O_{AB_y} \ O_{AB_z}]^T$  with respect to a reference system  $\{A\}$ . The matrix  ${}^A T_B$  will then correspond to a **homogeneous translation matrix** defined as follows:

$${}^A T_B = \text{Trans}(O_{AB}) = \begin{bmatrix} 1 & 0 & 0 & O_{AB} \\ 0 & 1 & 0 & O_{AB} \\ 0 & 0 & 1 & O_{AB} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & O_{AB_x} \\ 0 & 1 & 0 & O_{AB_y} \\ 0 & 0 & 1 & O_{AB_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.36)$$

and is known as a **basic translation matrix**.

Thus, for example, a vector  ${}^B P_e$  represented in a reference system  $\{B\}$  will have vector components with respect to a reference system  $\{A\}$  given by:

$${}^A P_e = \begin{bmatrix} P_{e_x} \\ P_{e_y} \\ P_{e_z} \\ 1 \end{bmatrix} = {}^A T_B {}^B P_e = \begin{bmatrix} 1 & 0 & 0 & O_{AB_x} \\ 0 & 1 & 0 & O_{AB_y} \\ 0 & 0 & 1 & O_{AB_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^B P_{e_x} \\ {}^B P_{e_y} \\ {}^B P_{e_z} \\ 1 \end{bmatrix} = \begin{bmatrix} O_{AB_x} + {}^B P_{e_x} \\ O_{AB_y} + {}^B P_{e_y} \\ O_{AB_z} + {}^B P_{e_z} \\ 1 \end{bmatrix} \quad (6.37)$$

which we can write as:

$${}^A P_e = {}^A T_B {}^B P_e \quad (6.38)$$

where  ${}^A T_B$  represents a relation between the reference system  $\{B\}$  and the reference system  $\{A\}$ .

Similarly, a vector  $P_e$  transformed by a basic translation matrix  ${}^A T_{transf}$  with respect to reference system  $\{A\}$  will have vector components given by:

$${}^A P_e^I = \begin{bmatrix} P_{e_x}^I \\ P_{e_y}^I \\ P_{e_z}^I \\ 1 \end{bmatrix} = {}^A T_{transf} {}^A P_e = \begin{bmatrix} 1 & 0 & 0 & O_{AB_x} \\ 0 & 1 & 0 & O_{AB_y} \\ 0 & 0 & 1 & O_{AB_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{e_x} \\ P_{e_y} \\ P_{e_z} \\ 1 \end{bmatrix} = \begin{bmatrix} O_{AB_x} + P_{e_x} \\ O_{AB_y} + P_{e_y} \\ O_{AB_z} + P_{e_z} \\ 1 \end{bmatrix} \quad (6.39)$$

which we can write as

$${}^A P'_e = {}^A T_{transf} {}^A P_e \quad (6.40)$$

where  ${}^A T_{transf}$  represents an operation on the vector  $P_e$  in the reference system  $\{A\}$ .

#### 6.4.5.1 Example 1

If the reference system  $\{B\}$  is translated by a vector  $O_{AB} = [6 \ -3 \ 8]^T$  with respect to reference system  $\{A\}$ , calculate the components  ${}^A P_e = [P_{e_x} \ P_{e_y} \ P_{e_z}]^T$  of the vector  $P_e$  of which the components with respect to  $\{B\}$  are  ${}^B P_e = [{}^B P_{e_x} \ {}^B P_{e_y} \ {}^B P_{e_z}]^T = [-2 \ 7 \ 3]^T$ .

$${}^A P_e = \begin{bmatrix} P_{e_x} \\ P_{e_y} \\ P_{e_z} \\ 1 \end{bmatrix} = {}^A T_B {}^B P_e = \begin{bmatrix} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -2 \\ 7 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 11 \\ 1 \end{bmatrix} \quad (6.41)$$

and thus  ${}^A P_{e_x} = 4$ ,  ${}^A P_{e_y} = 4$ , and  ${}^A P_{e_z} = 11$ .

#### 6.4.5.2 Example 2

Calculate the vector  ${}^A P_e^I$  which is the result of translating the vector  ${}^A P_e = [4 \ 4 \ 11]^T$  by using the transformation matrix  ${}^A T_{transf}$  with  $O_{transf} = [6 \ -3 \ 8]^T$ .

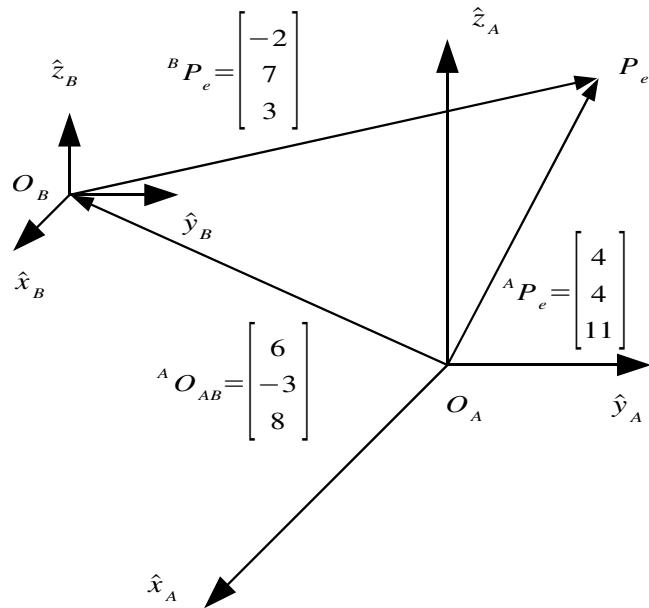


Figure 6.12: Translation of a frame.

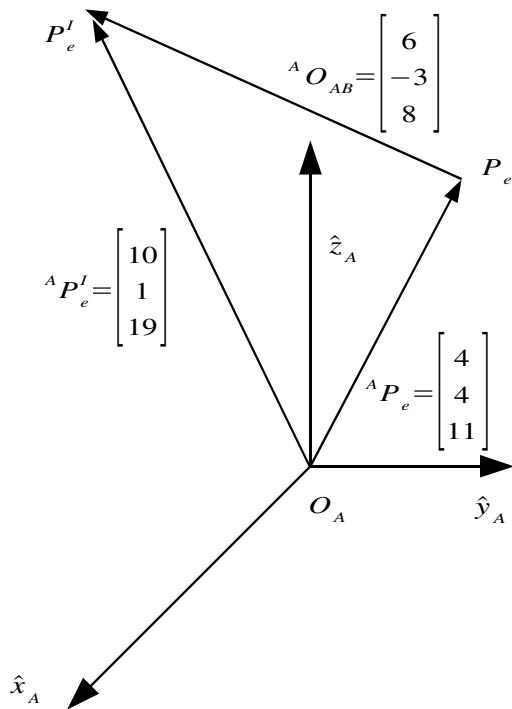


Figure 6.13: Translation of a position with a fixed frame.

$${}^A P_e^I = \begin{bmatrix} {}^A P_{e_x}^I \\ {}^A P_{e_y}^I \\ {}^A P_{e_z}^I \\ 1 \end{bmatrix} = {}^A T_{transf} {}^A P_e = \begin{bmatrix} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \\ 11 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 1 \\ 19 \\ 1 \end{bmatrix} \quad (6.42)$$

and thus  ${}^A P_{e_x}^I = 10$ ,  ${}^A P_{e_y}^I = 1$ , and  ${}^A P_{e_z}^I = 19$ .

#### 6.4.6 Basic Rotation Homogeneous Transformation Matrix

Assumed that the coordinate reference system  $\{B\}$  is rotated with respect to the reference system  $\{A\}$ . If the submatrix of rotation  $R_{3x3}$  takes the form of any one of the three basic rotation matrices introduced above, then we will have what we call a **basic rotation homogeneous transformation matrix**. This can take three different forms corresponding to the three basic rotation matrices.

$${}^A T_B = \text{Rot } (\hat{x}, \psi_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \psi_x & -\sin \psi_x & 0 \\ 0 & \sin \psi_x & \cos \psi_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.43)$$

$${}^A T_B = \text{Rot } (\hat{y}, \psi_y) = \begin{bmatrix} \cos \psi_y & 0 & \sin \psi_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \psi_y & 0 & \cos \psi_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.44)$$

$${}^A T_B = \text{Rot } (\hat{z}, \psi_z) = \begin{bmatrix} \cos \psi_z & -\sin \psi_z & 0 & 0 \\ \sin \psi_z & \cos \psi_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.45)$$

A vector  ${}^B P_e$  represented in a reference system  $\{B\}$  which is rotated by  ${}^A T_B$  will have components defined with respect to reference system  $\{A\}$ ,  ${}^A P_e$ , given by

$${}^A P_e = \begin{bmatrix} P_{e_x} \\ P_{e_y} \\ P_{e_z} \\ 1 \end{bmatrix} = {}^A T_B {}^B P_e = {}^A T_B \begin{bmatrix} {}^B P_{e_x} \\ {}^B P_{e_y} \\ {}^B P_{e_z} \\ 1 \end{bmatrix} \quad (6.46)$$

Here  ${}^A T_B$  represents the relation between the reference system  $\{B\}$  and the reference system  $\{A\}$ .

Similarly, a vector  ${}^A P_e$  which is transformed by  ${}^A T_{transf}$  to a new vector  ${}^A P'_e$  will be defined by:

$${}^A P'_e = \begin{bmatrix} P'_{e_x} \\ P'_{e_y} \\ P'_{e_z} \\ 1 \end{bmatrix} = {}^A T_{transf} {}^A P_e = {}^A T_{transf} \begin{bmatrix} P_{e_x} \\ P_{e_y} \\ P_{e_z} \\ 1 \end{bmatrix} \quad (6.47)$$

Here  ${}^A T_{transf}$  represents a rotation operation on the vector  ${}^A P_e$  in the reference system  $\{A\}$ .

#### 6.4.6.1 Example 1

If the reference system  $\{B\}$  is rotated  $\pi/2$  around the axis  $\hat{z}$  of reference system  $\{A\}$ , calculate the components of the vector  ${}^A P_e$ , given that  ${}^B P_e = [4 \ 8 \ 12]^T$ .

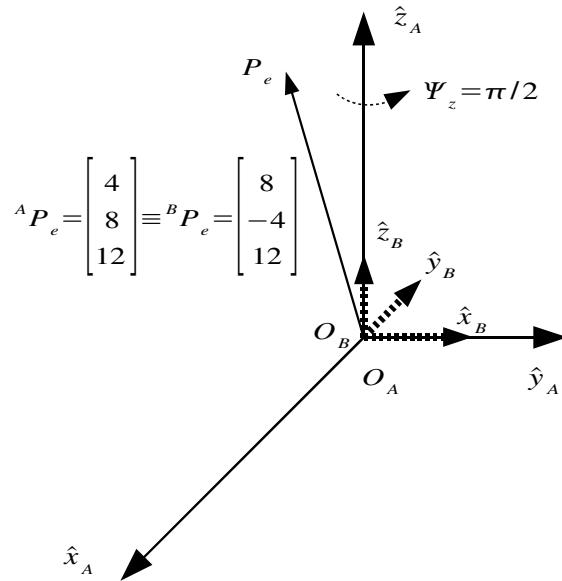


Figure 6.14: Rotation  $\pi/2$  around axis  $\hat{z}$ .

$${}^A P_e = \begin{bmatrix} P_{e_x} \\ P_{e_y} \\ P_{e_z} \\ 1 \end{bmatrix} = {}^A T_B {}^B P_e = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 8 \\ 12 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \\ -4 \\ 12 \\ 1 \end{bmatrix} \quad (6.48)$$

and thus  $P_{e_x} = 8$ ,  $P_{e_y} = -4$ , and  $P_{e_z} = 12$ .

#### 6.4.6.2 Composition of Basic Rotation Homogeneous Transformation Matrix

For example, a transformation matrix that represents a rotation of  $\psi_1$  around  $\hat{x}$ , followed by a rotation of  $\psi_2$  around  $\hat{y}$ , and finally a rotation of  $\psi_3$  around  $\hat{z}$ , can be obtained from combining three basic homogeneous transformation matrices in the following way:

$${}^A T_B = \text{Rot}(\hat{z}, \psi_3) \text{Rot}(\hat{y}, \psi_2) \text{Rot}(\hat{x}, \psi_1) \quad (6.49)$$

which gives:

$${}^A R_B = \begin{bmatrix} c_3c_2 & c_3s_2s_1 - s_3c_1 & c_3s_2c_1 + s_3s_1 & 0 \\ s_3c_2 & s_3s_2s_1 + c_3c_1 & s_3s_2c_1 - c_3s_1 & 0 \\ -s_2 & c_2s_1 & c_2c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.50)$$

where  $s_i = \sin \psi_i$ ,  $c_i = \cos \psi_i$  with  $i = 1, 2, 3$

which, due to the non-commutativity of matrix products (in general) is not the same as the homogeneous transformation matrix which results from applying the same three rotation operations in the reverse order. In other words,

$${}^A T_B = \text{Rot}(\hat{x}, \psi_1) \text{Rot}(\hat{y}, \psi_2) \text{Rot}(\hat{z}, \psi_3) \quad (6.51)$$

gives an

$${}^A T_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 & 0 \\ 0 & \sin \psi_1 & \cos \psi_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \psi_2 & 0 & \sin \psi_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \psi_2 & 0 & \cos \psi_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \psi_3 & -\sin \psi_3 & 0 & 0 \\ \sin \psi_3 & \cos \psi_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.52)$$

and thus

$${}^A T_B = \begin{bmatrix} c_3c_2 & -s_3c_2 & s_2 & 0 \\ c_3s_2s_1 + s_3c_1 & -s_3s_2s_1 + c_3c_1 & -c_2s_1 & 0 \\ -c_3s_2c_1 + s_3s_1 & s_3s_2c_1 + c_3s_1 & c_2c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.53)$$

where  $s_i = \sin \psi_i$ ,  $c_i = \cos \psi_i$  with  $i = 1, 2, 3$

In conclusion, we can see that the basic rotation homogeneous transformation matrices behave as the basic rotation matrices and no further explanation should be needed. Despite that, in the following subsections we are going to analyse this behaviour thoroughly.

### 6.4.7 Composition of Basic Translation and Rotation Homogeneous Transformation Matrices

The main advantages of using homogeneous coordinates and homogeneous transformation matrices is that they can be used to represent both translation (or position) and rotation (or orientation) of an object or reference system in 3D space.

If we want to represent the position and orientation of some coordinate reference system  $\{B\}$ , which is initially completely coincident with reference system  $\{A\}$ , but which has been subsequently rotated and translated with respect to  $\{A\}$ , we first need to know in what order the rotation and translation operations have been applied. This follows from the fact that basic homogeneous transformation matrices are not commutative in combination.

For example, if initially the reference system  $\{B\}$  is completely coincident with reference system  $\{A\}$ , and we then apply a translation of  $O_{AB}$  (we get  $\{B_1\}$ ) and then a rotation of  $\pi$  radians around the axis  $\hat{z}$  of reference system  $\{A\}$ ,  $\text{Rot}(\hat{z}_A, \pi)$ , we will have a new reference system  $\{B_2\}$  (figure 6.15).

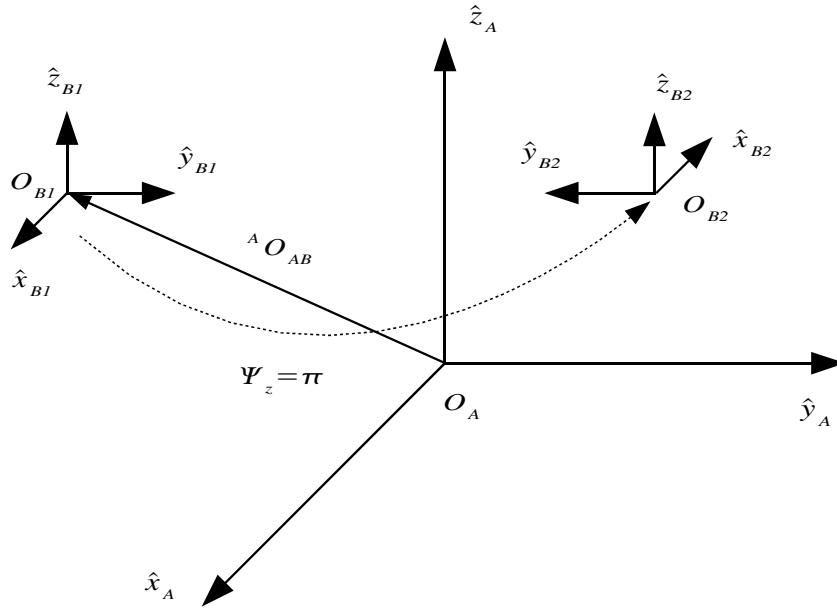


Figure 6.15: Translation plus rotation of a frame.

If, on the other hand, we first apply the rotation  $\text{Rot}(\hat{z}, \pi)$  (we get  $\{B_3\}$ ) and then we apply the translation  $O_{AB}$  with respect to the  $\{A\}$  coordinate system, we will get a different transformed reference system  $\{B_4\}$  (figure 6.16). Notice that  $\{B_2\} \not\sim \{B_4\}$ .

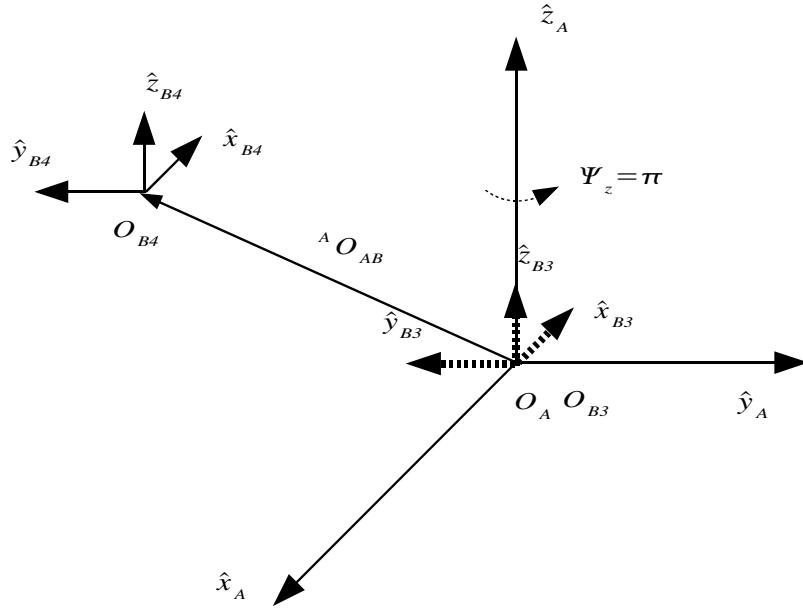


Figure 6.16: Translation plus rotation of a frame: Rotation and translation are not commutative.

#### 6.4.7.1 Rotation and then Translation

In the case where we first apply a rotation around one of the principle axes of the fixed reference system  $\{A\}$  followed by a translation, also defined with respect to the fixed reference system  $\{A\}$ , the homogeneous transformation matrix will have one of the following forms.

For a rotation  $\psi_1$  around the  $\hat{x}$  axis of  $\{A\}$ , followed by a translation defined by  $O_{AB}$

$$\begin{aligned} {}^A T_B &= \text{Trans}(O_{AB}) \text{Rot}(\hat{x}, \psi_1) = \begin{bmatrix} 1 & 0 & 0 & O_{ABx} \\ 0 & 1 & 0 & O_{ABy} \\ 0 & 0 & 1 & O_{ABz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 & 0 \\ 0 & \sin \psi_1 & \cos \psi_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} 1 & 0 & 0 & O_{ABx} \\ 0 & \cos \psi_1 & -\sin \psi_1 & O_{ABy} \\ 0 & \sin \psi_1 & \cos \psi_1 & O_{ABz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (6.54)$$

which we can write, in terms of basic homogeneous transformation matrices, as

$${}^A T_B = \text{Trans}(P_{3x1}) \text{Rot}(\hat{x}, \psi_1) \quad (6.55)$$

The two other forms will thus be given by

$${}^A T_B = \text{Trans}(P_{3x1}) \text{Rot}(\hat{y}, \psi_2) \quad (6.56)$$

and

$${}^A T_B = \text{Trans}(P_{3x1}) \text{Rot}(\hat{z}, \psi_3) \quad (6.57)$$

#### 6.4.7.2 Translation and then Rotation

In the case where we first apply a translation followed by a rotation around one of the principle axes of the fixed reference system  $\{A\}$ , the homogeneous transformation matrix will take one of the following forms:

$${}^A T_B = \text{Rot}(\hat{x}, \psi_1) \text{Trans}(P_{3x1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 & 0 \\ 0 & \sin \psi_1 & \cos \psi_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.58)$$

multiplying both matrices gives:

$${}^A T_B = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & \cos \psi_1 & -\sin \psi_1 & (P_y \cos \psi_1 - P_z \sin \psi_1) \\ 0 & \sin \psi_1 & \cos \psi_1 & (P_y \sin \psi_1 + P_z \cos \psi_1) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.59)$$

The two other forms will thus be given by:

$${}^A T_B = \text{Rot}(\hat{y}, \psi_2) \text{Trans}(P_{3x1}) \quad (6.60)$$

and

$${}^A T_B = \text{Rot}(\hat{z}, \psi_3) \text{Trans}(P_{3x1}) \quad (6.61)$$

#### 6.4.7.3 Example 1

A reference system  $\{B\}$ , which is initially completely coincident with a fixed reference system  $\{A\}$ , is first rotated by  $\pi/2$  around the axis  $\hat{x}$ , and then translated by a vector  $O_{AB} = [8 \ -4 \ 12]^T$ , defined with respect to  $\{A\}$ . Calculate the components of  ${}^A P_e = [P_{e_x} \ P_{e_y} \ P_{e_z}]^T$  of the vector  $P_e$ , given that  ${}^B P_e = [-3 \ 4 \ -11]^T$ .

$${}^A P_e = \text{Trans}(P_{3x1}) \text{Rot}(\hat{x}, \psi_1) {}^B P_e = \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 4 \\ -11 \\ 1 \end{bmatrix} \quad (6.62)$$

which gives,

$${}^A P_e = \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 0 & -1 & -4 \\ 0 & 1 & 0 & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 4 \\ -11 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \\ 16 \\ 1 \end{bmatrix} \quad (6.63)$$

and thus  $P_{e_x} = 8$ ,  $P_{e_y} = -4$ , and  $P_{e_z} = 12$ .

Now, we consider applying the transformations in the reverse order: a reference system  $\{B\}$  which is initially completely coincident with a fixed reference system  $\{A\}$ , is first translated by a vector  $O_{AB} = [8 \ -4 \ 12]^T$ , defined with respect to the fixed reference system  $\{A\}$ , and then rotated by  $\pi/2$  around the axis  $\hat{x}$ . Calculate the components  ${}^0 r = [P_{e_x} \ P_{e_y} \ P_{e_z}]^T$  of the vector  ${}^A P_e$ , given that  ${}^B P_e = [-3 \ 4 \ -11]^T$ .

$${}^A P_e = \text{Rot}(\hat{x}, \psi_1) \text{Trans}(P_{3x1}) {}^B P_e = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 4 \\ -11 \\ 1 \end{bmatrix} \quad (6.64)$$

which gives,

$${}^A P_e = \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 0 & -1 & -12 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 4 \\ -11 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ -1 \\ 0 \\ 1 \end{bmatrix} \quad (6.65)$$

and thus  $P_{e_x} = 5$ ,  $P_{e_y} = -1$ , and  $P_{e_z} = 0$ .

#### 6.4.8 Composition of Homogeneous Transformation Matrices

We have seen that the homogeneous transformation matrix can be used to represent combinations of translations and rotations applied to one movable reference system and defined with respect to another fixed reference system.

We have also seen that when the complete or final transformation is composed of a series of applied basic transformations, the composition of the total transformation matrix depends upon the order in which the basic operations are applied.

Using a more concrete example, if a reference system  $\{B\}$  is rotated  $\pi/2$  around  $\hat{x}$ , then translated  $O_{AB} = [5 \ 5 \ 10]^T$  with respect to the fixed reference system  $\{A\}$ , and finally rotated  $\pi/2$  around  $\hat{z}$ , what is the resulting homogeneous transformation matrix?

Composing the complete transformation matrix from basic transformation matrices, we have

$${}^A T_B = \text{Rot}(\hat{z}, \pi/2) \text{Trans} \begin{pmatrix} 5 \\ 5 \\ 10 \end{pmatrix} \text{Rot}(\hat{x}, -\pi/2) \quad (6.66)$$

thus

$${}^A T_B = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.67)$$

which gives

$${}^A T_B = \begin{bmatrix} 0 & 0 & -1 & -5 \\ 1 & 0 & 0 & 5 \\ 0 & -1 & 0 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.68)$$

In the examples we have seen so far, all the basic operations and transformations have been defined with respect to the fixed reference system  $\{A\}$ . However, we can also define basic operations and transformations with respect to the reference system that moves  $\{B\}$ , and combine these to form complete or total homogeneous transformation matrices. In this case, the order of composition is reversed!

For example, if the reference system  $\{B\}$  is first rotated an angle  $\psi_1$  around the axis  $\hat{x}_B$ , then rotated an angle  $\psi_2$  around axis  $\hat{y}_B$  of the reference system  $\{B\}$ , and finally rotated an angle  $\psi_3$  around the axis  $\hat{z}_B$  of the reference system  $\{B\}$ , then the complete homogeneous transformation matrix will be given by:

$${}^A T_B = \text{Rot}(\hat{x}_B, \psi_1) \text{Rot}(\hat{y}_B, \psi_2) \text{Rot}(\hat{z}_B, \psi_3) \quad (6.69)$$

This gives:

$${}^A T_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \psi_1 & -\sin \psi_1 & 0 \\ 0 & \sin \psi_1 & \cos \psi_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \psi_2 & 0 & \sin \psi_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \psi_2 & 0 & \cos \psi_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \psi_3 & -\sin \psi_3 & 0 & 0 \\ \sin \psi_3 & \cos \psi_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.70)$$

and thus

$${}^A T_B = \begin{bmatrix} c_3 c_2 & -s_3 c_2 & s_2 & 0 \\ c_3 s_2 s_1 + s_3 c_1 & -s_3 s_2 s_1 + c_3 c_1 & -c_2 s_1 & 0 \\ -c_3 s_2 c_1 + s_3 s_1 & s_3 s_2 c_1 + c_3 s_1 & c_2 c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.71)$$

where  $s_i = \sin \psi_i$ ,  $c_i = \cos \psi_i$  with  $i = 1, 2, 3$

Notice that

$$\begin{aligned} \text{Rot}(\hat{x}_B, \psi_1) \rightarrow \text{Rot}(\hat{y}_B, \psi_2) \rightarrow \text{Rot}(\hat{z}_B, \psi_3) &\neq \\ \neq \text{Rot}(\hat{z}_B, \psi_3) \rightarrow \text{Rot}(\hat{y}_B, \psi_2) \rightarrow \text{Rot}(\hat{x}_B, \psi_1) & \end{aligned} \quad (6.72)$$

but that

$$\begin{aligned} \text{Rot}(\hat{x}_B, \psi_1) \rightarrow \text{Rot}(\hat{y}_B, \psi_2) \rightarrow \text{Rot}(\hat{z}_B, \psi_3) &\equiv \\ \equiv \text{Rot}(\hat{z}, \psi_3) \rightarrow \text{Rot}(\hat{y}, \psi_2) \rightarrow \text{Rot}(\hat{x}, \psi_1) & \end{aligned} \quad (6.73)$$

Thus, in general, when we form complete or total homogeneous transformation matrices from combinations of basic transformation matrices, we must carefully note the following points:

- If the fixed reference system  $\{A\}$  and the transformed (movable) reference system  $\{B\}$  are completely coincident, then the homogeneous transformation matrix will be the  $4 \times 4$  identity matrix  $I_{4 \times 4}$ .
- If the transformed reference system  $\{B\}$  is obtained following a series of basic rotation and translation operations, all defined with respect to the fixed reference system  $\{A\}$ , then the complete (or total) homogeneous transformation matrix is obtained by premultiplying the result of the previous operations by the basic homogeneous transformation matrix of the next operation. In other words, the basic transformation matrices must be written down from **right to left** in the order in which they are applied.

- If the transformed reference system  $\{B\}$  is obtained following a series of basic rotation and translation operations, all defined with respect to the movable reference system  $\{B\}$ , then the complete (or total) homogeneous transformation matrix is obtained by postmultiplying the result of the previous operations by the basic homogeneous transformation matrix of the next operation. In other words, the basic transformation matrices must be written down from **left to right** in the order in which they are applied.

Thus any homogeneous transformation matrix can be composed of combinations of basic transformation matrices defined with respect to a fixed reference system or the transformed reference system, and we can even mix these in the composition, though this is not advisable, since it can more easily lead to construction errors.

In the next part of the course, on kinematic control, we will often use certain basic homogeneous transformation matrices which are defined with respect to a movable reference system, in order to compose the relationship between coordinate reference systems rigidly attached to the link or element that make up a robot manipulator.

#### 6.4.8.1 Example 1: postmultiplication

The transformation matrix that represents the following transformations:

1. Translation  $[-3 \ 10 \ 10]$  with respect to  $\{B\}$ ,
2. Rotation around  $\hat{x}_B$  at an angle of  $-\pi/2$  radians, and
3. Rotation around  $\hat{y}_B$  at an angle of  $\pi/2$  radians, is

$$\begin{aligned} {}^A T_B &= \text{Trans}(\{B\}, [-3 \ 10 \ 10]^T) \text{ rot}(\hat{x}_B, -\pi/2) \text{ rot}(\hat{y}_B, \pi/2) = \\ &= \begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & -3 \\ -1 & 0 & 0 & 10 \\ 0 & -1 & 0 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

#### 6.4.8.2 Example 2: premultiplication

The transformation matrix that represents the following transformations:

1. Rotation around  $\hat{x}_A$  at an angle of  $-\pi/2$  radians
2. Translation  $[5 \ 5 \ 10]$  with respect to  $\{A\}$ , and
3. Rotation around  $\hat{z}_A$  at an angle of  $\pi/2$  radians, is

$$\begin{aligned} {}^A T_B &= \text{rot}(\hat{z}_A, \pi/2) \text{Trans}(\{A\}, [5 \ 5 \ 10]^T) \text{rot}(\hat{x}_A, -\pi/2) = \\ &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -5 \\ 1 & 0 & 0 & 5 \\ 0 & -1 & 0 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$



## Chapter 7

# Kinematics of Manipulators

---

## 7.1 Introduction

In this part, we will study the kinematics of robot manipulator arms. In particular it will introduce the problems of the forward and inverse kinematics of robot arms. It will also introduce and illustrate a systematic method for describing robot arms.

First, we should be clear what **kinematics** is: it is the study of movement or motion without regard to the force or forces that produce the movement. Kinematics therefore includes the study of position, velocity, acceleration, and all higher derivatives of position. **Dynamics** is the study of movement or motion with regard to the forces that produce it.

We can study the kinematics of robot manipulator arms, without needing to study their dynamics, because all the forces involved can be generated by the controller.

We will begin by introducing the two basic problems of the kinematics of manipulator arms: the **forward kinematics problem**, and the **inverse kinematics problem**. We will then introduce a method for describing the geometry of robot manipulator arms, which we will need in order to study the above two problems.

## 7.2 Forward and Inverse Kinematics

The **forward kinematics problem** consists in *determining what the position and orientation of the reference system,  $\{e\}$ , at the end-point,  $P_e$ , of the robot is, with respect to some fixed global system of reference, given the values of the positions of each joints, plus information about the types of joints and the geometry of the elements that connect the joints.*

The **inverse kinematics problem** consists in *determining what values are needed for each of the joint positions, given a particular position and orientation of  $\{e\}$ , plus information about the types of joints and the geometry of the elements that connect them.*

From these definitions we can see that the forward kinematics problem, for any particular serial robot manipulator arm, has a unique solution, whereas, the inverse kinematic problem for that same robot arm may have more than one solution. In the case of parallel manipulators, the forward kinematics has more than one solution and the inverse kinematics has only one solution.

We can illustrate these two problems and the relationship between them in the following diagram.

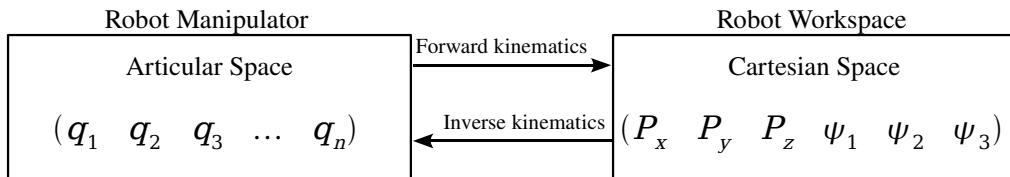


Figure 7.1: Forward Kinematics vs. Inverse Kinematics.

### 7.3 Geometric Parametres of Joint-Link Robotic Mechanisms

As we have seen, we can consider a serial manipulator as consisting of a set of links connected by single degree of freedom joints, where a link may be thought of as a rigid body which connects the axes of the two adjacent joints. Joint axes are simply represented as lines in space, the line about which we have rotation, in the case of a rotational joint, or the line along which we have linear movement, in the case of a linear joint.

The **links** that make up a robot arm are numbered starting with the fixed base of the robot, which we will call *link 0*. The first movable link will then be numbered as link 1, and so on, out to the end-point of the robot arm, which is attached to the last link in the chain, numbered *link n*.

The first and last links, *link 0* and *link n*, are special links, in the sense that they do not connect two joint axes. *Link 0* connects the first joint axis to the fixed world, and *link n* joins the last joint axis to the end-point,  $P_e$ , of the robot.

We use four parameters to describe the geometry of robotic mechanisms. Two are called the **Link Parameters**, and define the geometry of a link, and the other two are called the **Joint Parameters** or **Connection Parameters**, and describe the geometry of a joint.

#### 7.3.1 Link Parameters

A joint *axis i* will be defined by a direction vector, where the positive direction is used to define the positive (counterclockwise) rotation direction of a rotational joint, or the positive direction of movement in the case of a linear joint.

Thus, joint *axis i* is defined by a direction vector, around which *link i* rotates relative to *link i - 1*, and we can specify the relative location of the two axes involved, *axis i* and *axis i - 1*, in terms of two parameters.

In the same way joint *axis i + 1* is defined by a direction vector, around which *link i + 1* rotates relative to *link i*, and we can specify the relative location of the two axes involved, *axis i + 1* and *axis i*, in terms of two parameters.

Taking into account the relationships described in the last two paragraph, we can define the two link parameters.

The first link parameter is the distance between the two axes, called the **link length**. For any two (joint) axes in 3D space, there exists a well defined measure of the distance between them given by the length of the line which connects them and is mutually perpendicular to both axes (*axis i* and *axis i + 1*). This mutually perpendicular line always exists and is unique, except when the two axes are parallel, in which case there is an infinite number of them, but all of the same length. If the two axes meet at some point in 3D space, then this line has zero length. The length of this line between the two joint axes that intersects them, is the **link length**, and is denoted by  $a_i$  (figure 7.2).

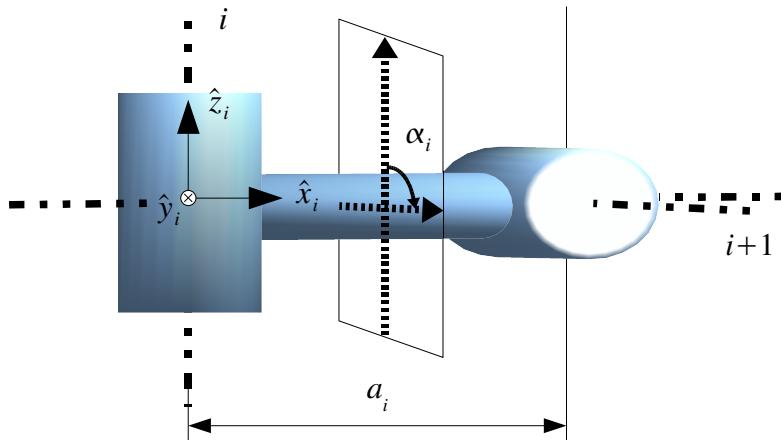


Figure 7.2: Link parameters: Link Length ( $a_i$ ) and Link Twist ( $\alpha_i$ ).

The second link parameter is the relative rotation, or *twist* between the two axes (*axis i* and *axis i + 1*), called the **link twist**. Imagine a plane whose normal vector is the mutually perpendicular line just constructed for the link distance. We then project the *axis i* onto this plane, and then we project *axis i + 1* onto the same plane. The *link twist* is then defined as the angle between these two projected axes, measured in the right-hand sense around the line of the link length. If the two joint axes intersect at some point, then the plane is the one that contains the two axes. However, in this case we have no definition of the positive rotation direction, so in this case we can choose it arbitrarily<sup>1</sup>. The link twist for *link i* is denoted by  $\alpha_i$  (figure 7.2).

The two link parameters,  $a_i$  and  $\alpha_i$ , for a particular link have fixed values which are

<sup>1</sup>The advice in this case is to choose the direction that coincides with the direction of the cross product between the *axis i* and the *axis i + 1*.

determined by the physical form of the link.

### 7.3.2 Joint Parameters

Two links are connected by a joint. They therefore have a common joint axis (joint axis  $i$ ). The **link offset** parameter is then defined as the distance along the common joint axis from the end of the first link (*link*  $i - 1$ ) to the beginning of the next link (*link*  $i$ ). The offset of *link*  $i$  with respect to *link*  $i - 1$ , at joint axis  $i$ , is denoted by  $d_i$  (figure 7.3).

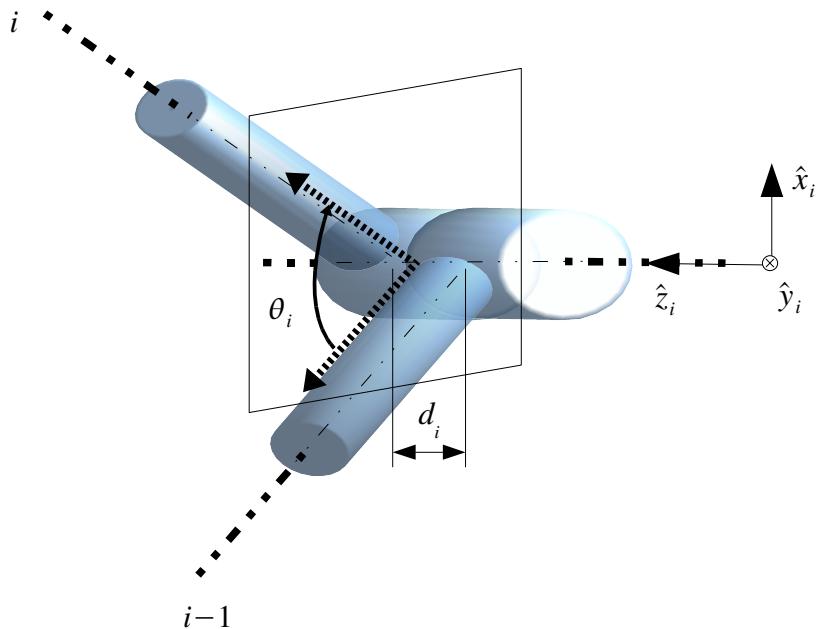


Figure 7.3: Joint parameters: Link Offset and Joint Angle.

The second joint parameter describes the relative rotation of *link*  $i$  with respect to *link*  $i - 1$ , around the common joint axis  $i$ . It is called the **Joint Angle**, and is denoted by  $\theta_i$  (figure 7.3).

The two joint parameters have either a fixed value or are variable, depending upon the type of joint.

In the case of a prismatic or linear joint connecting *link*  $i - 1$  to *link*  $i$ , the joint angle,  $\theta_i$ , will have a fixed value, and the link offset,  $d_i$ , will vary according to the position of the prismatic joint.

In the case of a rotational joint connecting *link*  $i - 1$  to *link*  $i$ , the joint angle,  $\theta_i$  will vary according to the position of the rotational joint and the link offset,  $d_i$  will have a fixed value.

Joint Type	Link Offset ( $d_i$ )	Joint Angle ( $\theta_i$ )
prismatic	variable	constant
rotational	constant	variable

Table 7.1: List of variable parameters according to the type of DoF.

## 7.4 Defining the Link Reference Systems (Denavit and Hartenberg method)

To be able to represent the relationship between any two connected links in a robot manipulator arm, we will fix (rigidly attach) a coordinate reference system to each link, and then construct the homogeneous transformation matrix that represents the geometric relationship between them, using the four geometric parameters introduced above.

Normally, we are interested in the relationship between a *link*  $i$  and the link immediately preceding it, i.e., *link*  $i - 1$ .

Then, by combining the relationships between pairs of connected links we can construct the relationship between the last *link*  $n$ , which has the robot gripper/end-effector attached to it, and the first link, *link* 0, which is the robot base. Once we have this whole arm relationship defined, we can use it to calculate where the end-point  $P_e$  of the robot is with respect to the robot base, knowing what the  $n$  joint values are. Or we can use it to solve the inverse kinematics problem: calculate what joint values are needed for a particular position and orientation of  $P_e$ .

To do this in a convenient and accurate manner, we will always define these link reference systems in the same way, using the following convention (also called the Denavit-Hartenberg convention):

1. Link reference systems are numbered according to the number of the link to which they are fixed. Thus, reference system  $\{i\}$  is fixed to *link*  $i$ .
2. For intermediate links: link with joints at both ends →
  - (a) The origin of  $\{i\}$  is defined as the intersection of the joint *axis*  $i$  and the line of the link length,  $a_i$ .
  - (b) The axis  $\hat{z}_i$  of  $\{i\}$  is defined to be along the joint *axis*  $i$ . In this case we can choose which direction to draw as the positive  $\hat{z}_i$ , but the best is always to choose the same criteria for each of the link reference systems.
  - (c) The axis  $\hat{x}_i$  of  $\{i\}$  is defined to be along the line of the link length,  $a_i$ , with the positive direction being from the joint *axis*  $i$  to the joint *axis*  $i + 1$ .
  - (d) Once we have the axes  $\hat{x}_i$  and  $\hat{z}_i$  of  $\{i\}$  defined, we define the axis  $\hat{y}_i$  so that the reference system  $\{i\}$  is a right-handed reference system.
3. For the first link: number zero, the fixed base of the robot arm →
  - (a) The direction of the axis  $\hat{z}_0$  of  $\{0\}$  is defined to be along the axis of *joint* 1.

- (b) The reference system  $\{0\}$  is then defined to be completely coincident with the reference system  $\{1\}$ , when the variable joint parameter,  $d_1$  or  $\theta_1$ , is zero.
  - (c) Following this convention for *link 0*, we will always have  $a_0 = 0$  and  $\alpha_0 = 0$ , and,
    - in the case of *joint 1* being prismatic,  $\theta_1 = 0$  or,
    - in the case of *joint 1* being rotational,  $d_1 = 0$ .
4. For the last link: *link n*, to which the gripper/end-effector is attached →
- (a) In the case that *joint n* is prismatic (which is rather rare), the  $\hat{x}_n$  axis of  $\{n\}$  is defined so that  $\theta_n = 0$ , and the origin of  $\{n\}$  is defined to be at the intersection of the axis  $\hat{x}_{n-1}$  and the axis of joint  $n$  ( $\hat{z}_n$ ), when  $d_n = 0$ .
  - (b) In the case that *joint n* is rotational (most common) the axis  $\hat{x}_n$  of  $\{n\}$  is defined to have the same direction as  $\hat{x}_{n-1}$  when  $\theta_n = 0$ , and the origin of  $\{n\}$  is defined so that  $d_n = 0$ .

The four geometric parameters that we need to construct the relationship between the reference system  $\{i\}$ , fixed to *link i*, and the reference system  $\{i-1\}$ , fixed to *link i-1*, are:

- $a_{i-1}$ , the length of *link i - 1* and the distance between the origin of  $\{i-1\}$  and the origin of  $\{i\}$  in the direction of  $\hat{x}_{i-1}$
- $\alpha_{i-1}$ , the twist of *link i - 1* and the angle between  $\hat{z}_{i-1}$  and  $\hat{z}_i$  around  $\hat{x}_{i-1}$  of  $\{i-1\}$ .
- $d_i$ , the offset of *link i* with respect to *link i - 1* and the distance between the origin of  $\{i-1\}$  and  $\{i\}$  in the direction of  $\hat{z}_i$  of  $\{i\}$ .
- $\theta_i$ , the angle of joint *joint i*, the angle between  $\hat{x}_{i-1}$  and  $\hat{x}_i$  around  $\hat{z}_i$  of  $\{i\}$ .

This convention or method for defining link reference systems is called the **Denavit and Hartenberg method** or **D-H method**, after the two people who first proposed it in 1955.

Today there are numerous versions of this method. The one presented here is sometimes called the **modified Denavit and Hartenberg method** or the **Denavit and Hartenberg method using Craig's convention**. It is distinguished by the reference system number scheme used. In the methods presented here, and which we will always use, reference system  $\{i\}$  is fixed to *link i* and has its origin lying on the axis of *joint i*, while in the **original Denavit and Hartenberg method**, the reference system  $\{i\}$  is fixed to *link i* and has its origin lying on the axis of *joint i + 1*<sup>(2)</sup>.

It should, however, always be remembered that the reference system  $\{i\}$  is fixed to the *link i*, and not to the axis of *joint i*. It therefore moves in space as *link i* is moved in space as a result of changes in the position of *joint i*.

---

<sup>2</sup>Trying to study both conventions at the same time is a good source of confusion. So the advice is to study only one.

Applying the Denavit and Hartenberg method to two connected links, *link*  $i - 1$  and *link*  $i$ , connected by *joint*  $i$ , gives us the following definition of the reference systems  $\{i - 1\}$  and  $\{i\}$  fixed to the *link*  $i - 1$  and *link*  $i$ , respectively.

The homogeneous transformation matrix that represents the relationship between  $\{i - 1\}$  and  $\{i\}$  is then given by:

$${}^{i-1}T_i = \text{Trans}(\hat{x}_{i-1}, a_{i-1}) \text{Rot}(\hat{x}_{i-1}, \alpha_{i-1}) \text{Trans}(\hat{z}_i, d_i) \text{Rot}(\hat{z}_i, \theta_i) \quad (7.1)$$

where all operations are defined with respect to the movable system of reference. Thus, we have that:

$${}^{i-1}T_i = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_{i-1} & -\sin \alpha_{i-1} & 0 \\ 0 & \sin \alpha_{i-1} & \cos \alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.2)$$

$${}^{i-1}T_i = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & \cos \alpha_{i-1} & -\sin \alpha_{i-1} & 0 \\ 0 & \sin \alpha_{i-1} & \cos \alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.3)$$

$${}^{i-1}T_i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \sin \theta_i \cos \alpha_{i-1} & \cos \theta_i \cos \alpha_{i-1} & -\sin \alpha_{i-1} & -\sin \alpha_{i-1} d_i \\ \sin \theta_i \sin \alpha_{i-1} & \cos \theta_i \sin \alpha_{i-1} & \cos \alpha_{i-1} & \cos \alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.4)$$

## 7.5 Forward Kinematics Examples

### 7.5.0.1 Example 1

A two dimensional robot with three Degrees of Freedom and three planar elements (figure 7.4).

The first step is to number the joint axes of the robot geometry in strict sequence, starting with the joint nearest to the robot base and numbering it as *joint* 1.

The second step is to number the links, again in strict sequence, starting with the robot base, which is given the number zero.

Following these two steps, we will have the situation in figure 7.5.

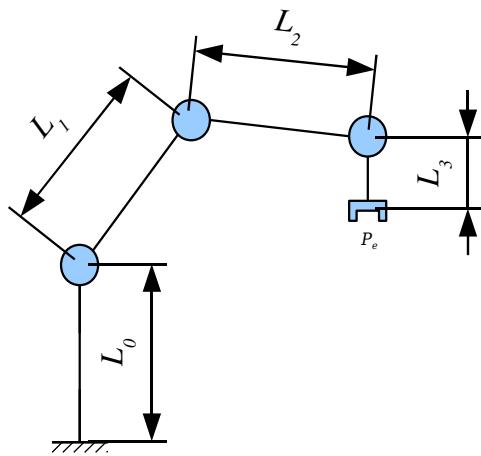


Figure 7.4: Example 1: 3 DoF planar robot.

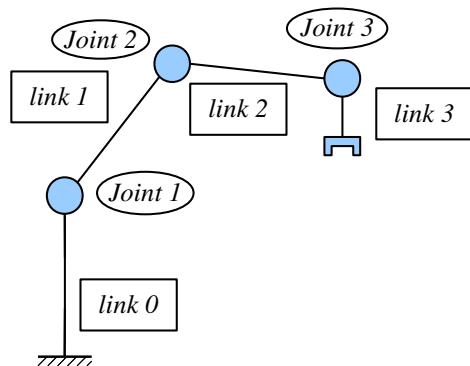


Figure 7.5: Forward Kinematics. First two steps: numbering the joints and numbering the links.

<i>link i</i>	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1		$L_0$	0	0	$\theta_1$
2		$L_1$	0	0	$\theta_2$
3		$L_2$	0	0	$\theta_3$

Table 7.2: DH parameters for example 1.

Having numbered the joint axes and the elements, the third step is to define the reference systems fixed to each element, following the method of Denavit and Hartenberg presented in the previous section. We will then have the following set of reference systems fixed to the three elements of this planar robot arm (figure 7.6).

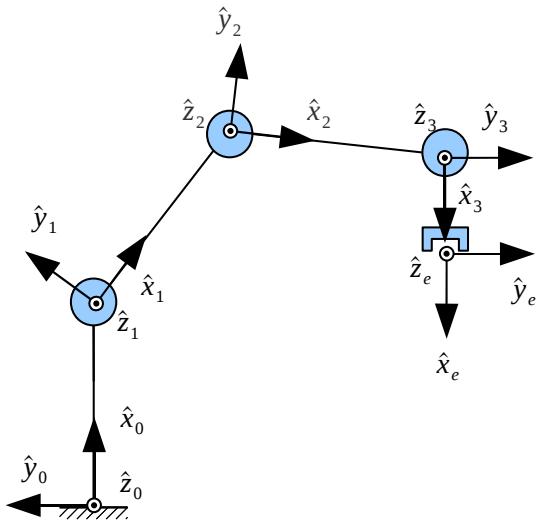


Figure 7.6: Forward Kinematics. Third step: defining reference systems.

The  $\hat{z}_i$  axes are all defined to be positive out of the page, and the reference system fixed to the base,  $\{0\}$  is defined to be completely coincident with the reference system fixed to element 1,  $\{1\}$ , when  $\theta_1$  is zero (figure 7.6).

The fourth step is then to construct the **table of D-H parameters** needed to define the relationships between subsequent reference systems (table 7.2).

All the link twist and link offset values must be zero because this is a 2D planar arm. Also, all the variable parameters are the  $\theta_i$  values since this robot arm only has rotational joints.

Using the values set out in *the D-H parameter table* 7.2, the fifth step is to define the homogeneous transformation matrices that represent the relationships between pairs of subsequent link reference systems, in terms of basic homogenous transformation matrices.

We thus have:

$${}^0T_1 = \text{Trans}(\hat{x}_0, 0) \text{Rot}(\hat{x}_0, 0) \text{Trans}(\hat{z}_1, 0) \text{Rot}(\hat{z}_1, \theta_1) = \text{Trans}(\hat{x}_0, L_0) \text{Rot}(\hat{z}_1, \theta_1) \quad (7.5)$$

$${}^1T_2 = \text{Trans}(\hat{x}_1, L_1) \text{Rot}(\hat{x}_1, 0) \text{Trans}(\hat{z}_2, 0) \text{Rot}(\hat{z}_2, \theta_2) = \text{Trans}(\hat{x}_1, L_1) \text{Rot}(\hat{z}_2, \theta_2) \quad (7.6)$$

$${}^2T_3 = \text{Trans}(\hat{x}_2, L_2) \text{Rot}(\hat{x}_2, 0) \text{Trans}(\hat{z}_3, 0) \text{Rot}(\hat{z}_3, \theta_3) = \text{Trans}(\hat{x}_2, L_2) \text{Rot}(\hat{z}_3, \theta_3) \quad (7.7)$$

The total relationship between the reference system fixed to element 3 and the reference system fixed to the base of the robot, is thus given by:

$${}^0T_3 = {}^0T_1 {}^1T_2 {}^2T_3 = \prod_{i=1}^3 {}^{i-1}T_i \quad (7.8)$$

Which, in turn, is given by:

$${}^0T_3 = \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & L_0 \\ \sin \theta_1 & \cos \theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 & L_1 \\ \sin \theta_2 & \cos \theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_3 & -\sin \theta_3 & 0 & L_2 \\ \sin \theta_3 & \cos \theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.9)$$

$${}^0T_3 = \begin{bmatrix} c_{123} & -s_{123} & 0 & P_{x_3} \\ s_{123} & c_{123} & 0 & P_{y_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.10)$$

where:

$$\begin{aligned} c_{123} &= \cos(\theta_1 + \theta_2 + \theta_3) \\ s_{123} &= \sin(\theta_1 + \theta_2 + \theta_3) \\ P_{x_3} &= L_2 c_{12} + L_1 c_1 + L_0 \\ P_{y_3} &= L_2 s_{12} + L_1 s_1 \end{aligned} \quad (7.11)$$

So, for example, if we want to define the position of the end-point of this robot,  $P_e$ , with respect to the fixed base system of reference we should use the following relation:

$${}^0P_e = {}^0T_3 {}^3P_e = {}^0T_3 \begin{bmatrix} L_3 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (7.12)$$

Notice that strictly speaking we have not arrived to the end effector yet. This fact is quite common after the use of the D-H method. If we really need to calculate the transform from the robot base to the end-effector, we have to modify the table 7.2

<i>link i</i>	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1		$L_0$	0	0	$\theta_1$
2		$L_1$	0	0	$\theta_2$
3		$L_2$	0	0	$\theta_3$
e		$L_3$	0	0	0

Table 7.3: DH parameters for example 1, including the end-effector frame.

slightly, including a new row (see table 7.3). This row is out of the scope of D-H method, but required to properly obtain the end-effector coordinates. As this last row is not a part of the D-H method, there is no variable parameter and the row has segregated from the other rows by a double line.

Then, the complete solution would be:

$${}^0T_e = {}^0T_3 {}^3T_e = \begin{bmatrix} c_{123} & -s_{123} & 0 & P_{x_3} \\ s_{123} & c_{123} & 0 & P_{y_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.13)$$

$${}^0T_e = \begin{bmatrix} c_{123} & -s_{123} & 0 & P_{x_e} \\ s_{123} & c_{123} & 0 & P_{y_e} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.14)$$

where:

$$\begin{aligned} c_{123} &= \cos(\theta_1 + \theta_2 + \theta_3) \\ s_{123} &= \sin(\theta_1 + \theta_2 + \theta_3) \\ P_{x_e} &= L_3 c_{123} + L_2 c_{12} + L_1 c_1 + L_0 \\ P_{y_e} &= L_3 s_{123} + L_2 s_{12} + L_1 s_1 \end{aligned} \quad (7.15)$$

### 7.5.0.2 Example 2

The second example is a three DoF robot with one prismatic joint. After first numbering the joints, numbering the elements, and then defining the reference systems fixed to each element, we have the set of reference frames shown in the figure below (figure 7.7).

The D-H table for this geometry is in table 7.4:

The three homogeneous transformation matrices that represent the relations between the three pairs of reference systems are thus given by:

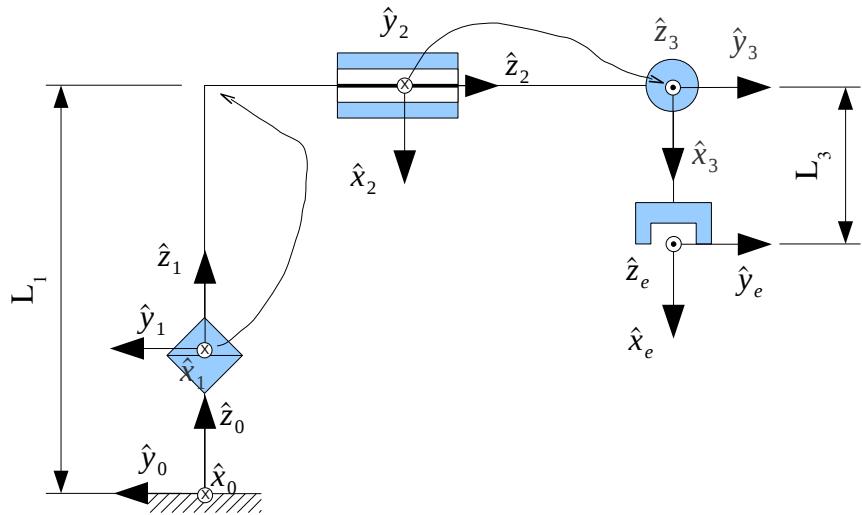


Figure 7.7: Example 2: a three DoF robot with one prismatic joint.

<i>link i</i>	DH par.			
	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1	0	0	$L_1$	$\theta_1$
2	0	$\pi/2$	$d_2$	$-\pi/2$
3	$\pi/2$	0	0	$\theta_3$
e	$L_3$	0	0	0

Table 7.4: DH parameters for example 2.

<i>link i</i>	DH par.			
	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1	0	0	0	$\theta_1$
2	0	$-\pi/2$	$L_1$	$\theta_2 - \pi/2$
3	$L_2$	0	0	$\theta_3$

Table 7.5: First set DH parameters for example 3.

$${}^0T_1 = \text{Trans}(\hat{z}_1, L_1) \text{Rot}(\hat{z}_1, \theta_1) \quad (7.16)$$

$${}^1T_2 = \text{Rot}(\hat{x}_1, \pi/2) \text{Trans}(\hat{z}_2, d_2) \text{Rot}(\hat{z}_2, -\pi/2) \quad (7.17)$$

$${}^2T_3 = \text{Rot}(\hat{x}_2, \pi/2) \text{Rot}(\hat{z}_3, \theta_3) \quad (7.18)$$

$${}^3T_e = \text{Trans}(\hat{z}_3, L_3) \quad (7.19)$$

The total relationship between the reference system fixed to element 3 and the reference system fixed to the base of the robot is thus given by:

$${}^0T_3 = {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_e = \quad (7.20)$$

$${}^0T_3 = \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -d_2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_3 & -\sin \theta_3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin \theta_3 & \cos \theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.21)$$

### 7.5.0.3 Example 3

Another 3 DoF manipulator with three rotational joints. The two first axes intersect each other and the two last ones are parallel (figure 7.8).

After numbering the joints and then the elements, we have four different possibilities for defining the reference systems fixed to each of the elements, all of which are compatible with the D-H method presented above.

The first two solutions are shown in figure 7.9. For the first solution, shown on the left figure 7.9, the D-H parameters are in table 7.5.

And for the second system, shown on the right figure 7.9, the D-H parameters are in table 7.6.

The second two systems are shown in figure 7.10, and their D-H parameters are in table 7.7 and in table 7.8 respectively.

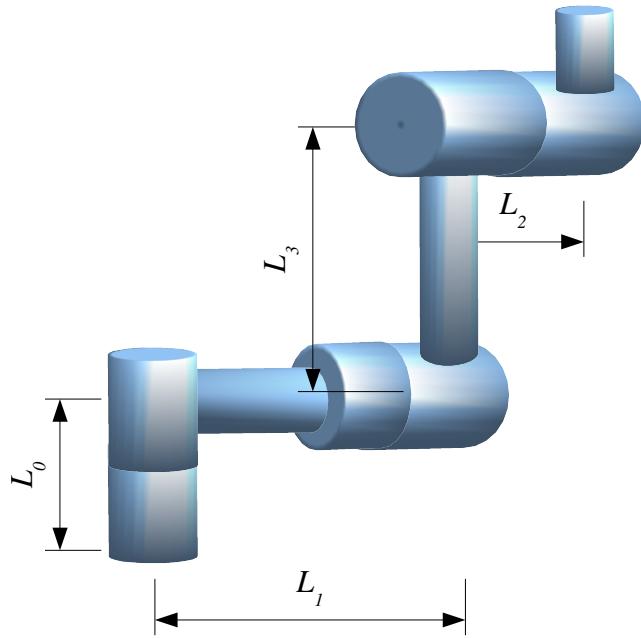


Figure 7.8: Example 3: a 3DoF manipulator: three rotational joints, 2 axes intersecting axes and 2 parallel axes.

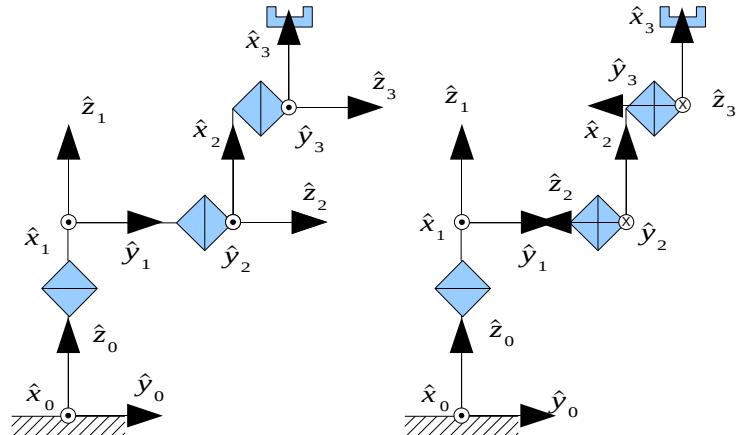


Figure 7.9: First two solutions for example 3.

<i>link i</i>	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1		0	0	0	$\theta_1$
2		0	$\pi/2$	$-L_1$	$\theta_2 + \pi/2$
3		$L_2$	0	0	$\theta_3$

Table 7.6: Second set DH parameters for example 3.

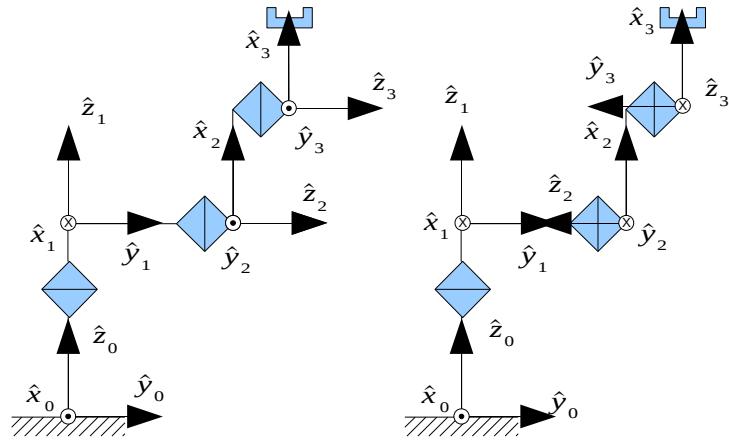


Figure 7.10: Second two solutions for example 3.

<i>link</i> <i>i</i>	DH par.	<i>a</i> <sub><i>i</i>-1</sub>	<i>α</i> <sub><i>i</i>-1</sub>	<i>d</i> <sub><i>i</i></sub>	<i>θ</i> <sub><i>i</i></sub>
1	0	0	0	0	<i>θ</i> <sub>1</sub>
2	0	$\pi/2$	$L_1$	0	$\theta_2 + \pi/2$
3	$L_2$	0	0	0	<i>θ</i> <sub>3</sub>

Table 7.7: Third set DH parameters for example 3.

<i>link</i> <i>i</i>	DH par.	<i>a</i> <sub><i>i</i>-1</sub>	<i>α</i> <sub><i>i</i>-1</sub>	<i>d</i> <sub><i>i</i></sub>	<i>θ</i> <sub><i>i</i></sub>
1	0	0	0	0	<i>θ</i> <sub>1</sub>
2	0	$-\pi/2$	$-L_1$	0	$\theta_2 - \pi/2$
3	$L_2$	0	0	0	<i>θ</i> <sub>3</sub>

Table 7.8: Fourth set DH parameters for example 3.

#### 7.5.0.4 Example 4

The next example is a cylindrical robot with 4 DoF (figure 7.11).

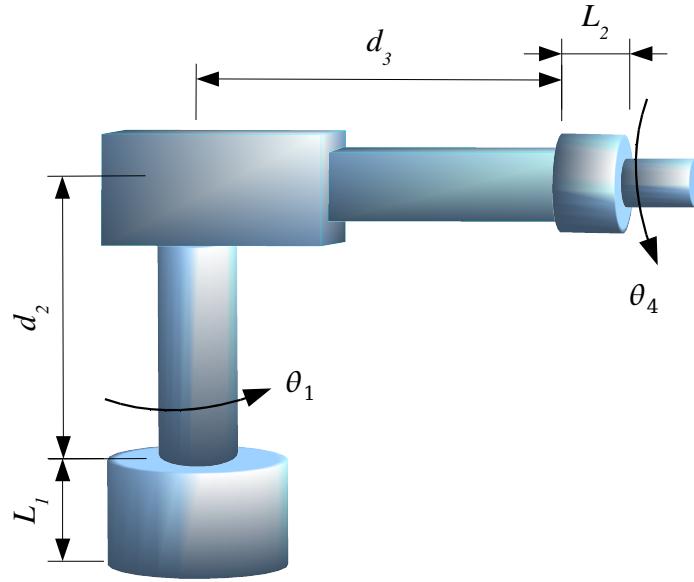


Figure 7.11: Example 4: a cylindrical robot.

Notice that we have put the reference system {1} at the bottom of element 1, so that it is coincident with system {0} when  $\theta_1$  is zero. We could have equally well put it at the top of element 1, so that its origin is coincident with the origin of system {2} (figure 7.12).

The table of D-H parameters for the set of reference systems shown above is in table 7.9.

The four homogeneous transformation matrices that represent the relations between the four pairs of reference systems are thus given by:

$${}^0T_1 = \text{Rot}(\hat{z}_1, \theta_1) \quad (7.22)$$

$${}^1T_2 = \text{Trans}(\hat{z}_2, d_2 + L_1) \quad (7.23)$$

$${}^2T_3 = \text{Rot}(\hat{x}_2, \pi/2) \text{Trans}(\hat{z}_3, d_3) \quad (7.24)$$

$${}^3T_4 = \text{Trans}(\hat{z}_4, L_2) \text{Rot}(\hat{z}_4, \theta_4) \quad (7.25)$$

The total relationship between the reference system fixed to element 4 and the reference system fixed to the base of the robot is thus given by:

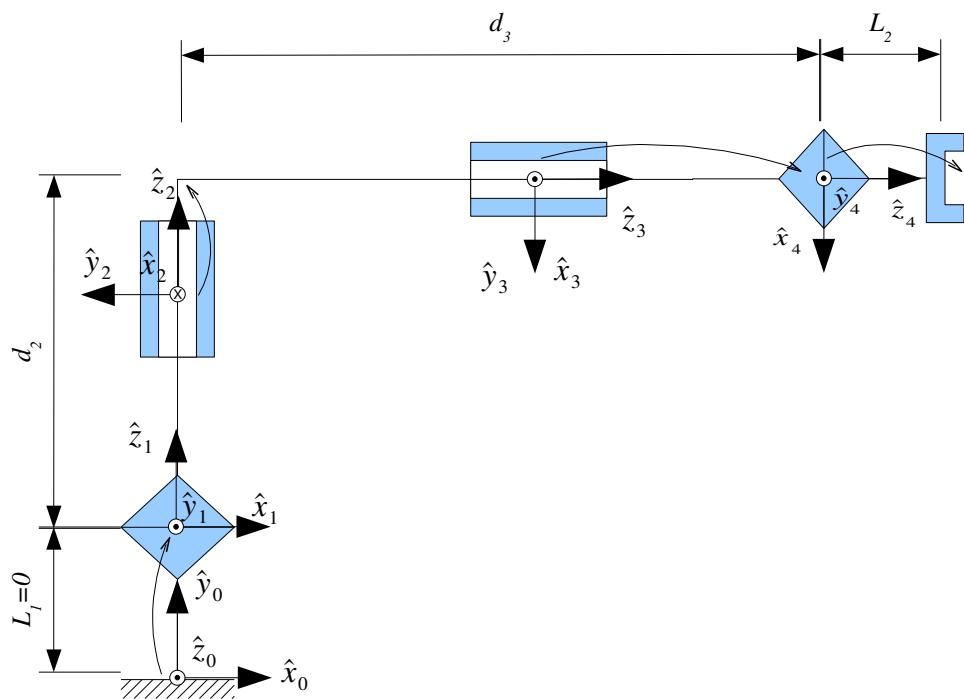


Figure 7.12: Reference systems for example 4.

link $i$	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1		0	0	0	$\theta_1$
2		0	0	$d_2 + L_1$	0
3		0	$\pi/2$	$d_3$	0
4		0	0	$L_2$	$\theta_4$
e		0	0	0	0

Table 7.9: DH parameters for example 4.

$$\begin{aligned}
 {}^0T_4 = & \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 + L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 & \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_4 & -\sin \theta_4 & 0 & 0 \\ \sin \theta_4 & \cos \theta_4 & 0 & 0 \\ 0 & 0 & 1 & L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned} \tag{7.26}$$

### 7.5.0.5 Example 5

The last example is a 4 DoF SCARA robot (figure 7.13).

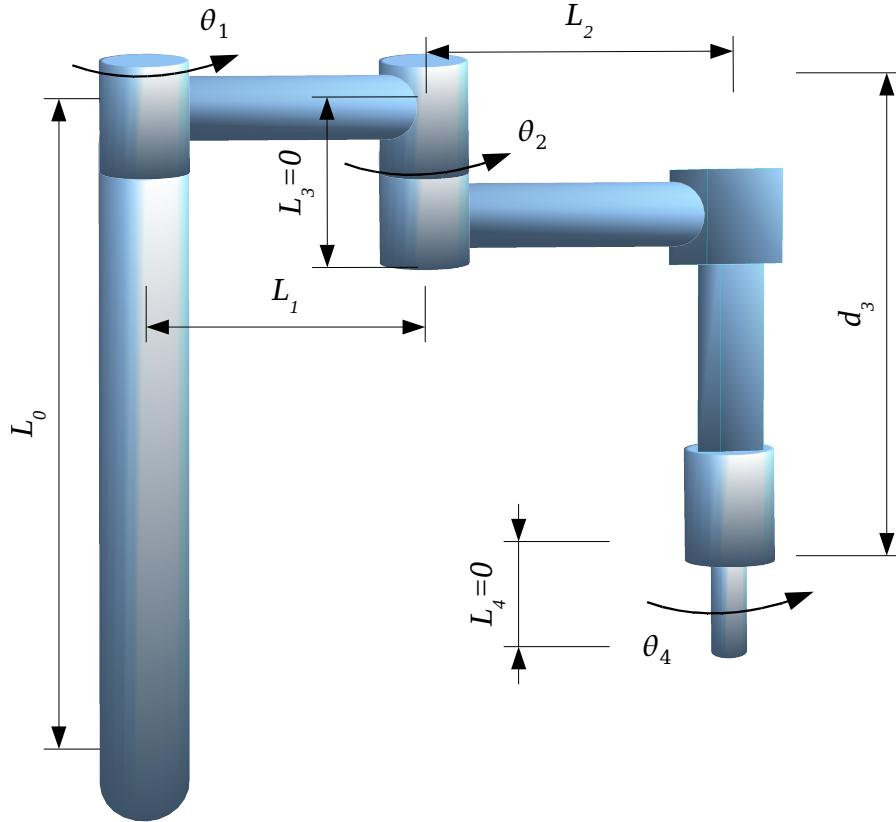


Figure 7.13: Example 5: a 4 DoF SCARA robot.

After numbering the joints, then the elements and assigning a suitable set of reference systems fixed to the elements of this robot, we will have the situation in figure 7.14.

The table of D-H parameters for the set of reference systems shown above is the following table 7.10.

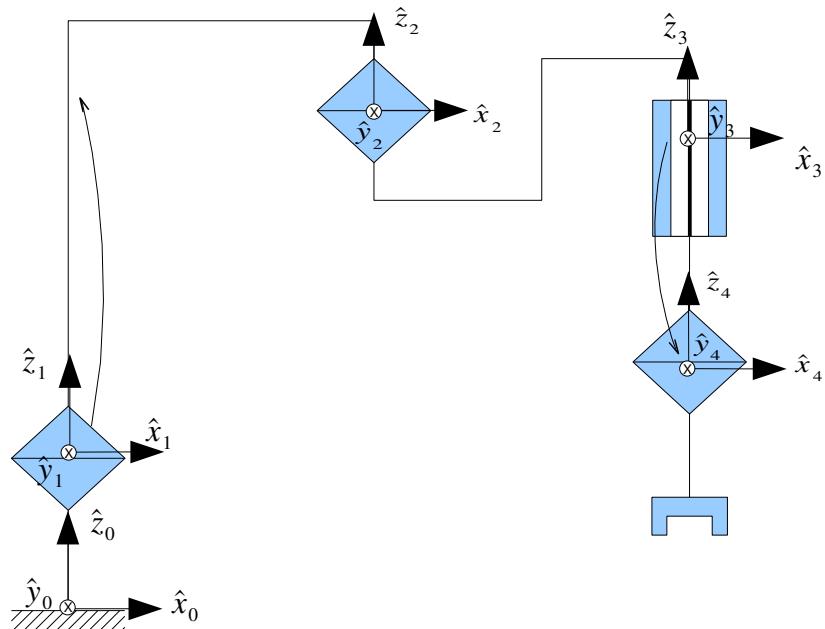


Figure 7.14: Reference systems for example 5.

<i>link i</i>	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1		0	0	$L_0$	$\theta_1$
2		$L_1$	0	$-L_3$	$\theta_2$
3		$L_2$	0	$-d_3$	0
4		0	0	$-L_4$	$\theta_4$

Table 7.10: DH parameters for example 5.

The four homogeneous transformation matrices that represent the relations between the four pairs of reference systems are thus given by:

$${}^0T_1 = \text{Trans}(\hat{z}_1, L_0) \text{Rot}(\hat{z}_1, \theta_1) \quad (7.27)$$

$${}^1T_2 = \text{Trans}\left(\begin{bmatrix} L_1 & 0 & -L_3 \end{bmatrix}^T\right) \text{Rot}(\hat{z}_2, \theta_2) \quad (7.28)$$

$${}^2T_3 = \text{Trans}\left(\begin{bmatrix} L_2 & 0 & -d_3 \end{bmatrix}^T\right) \quad (7.29)$$

$${}^3T_4 = \text{Trans}(\hat{z}_4, -L_4) \text{Rot}(\hat{z}_4, \theta_4) \quad (7.30)$$

The total relationship between the reference system fixed to element 4 and the reference system fixed to the base of the robot is thus given by:

$$\begin{aligned} {}^0T_4 = & \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 & L_1 \\ \sin \theta_2 & \cos \theta_2 & 0 & 0 \\ 0 & 0 & 1 & -L_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \\ & \cdot \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_4 & -\sin \theta_4 & 0 & 0 \\ \sin \theta_4 & \cos \theta_4 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (7.31)$$

## 7.6 Inverse Kinematics

We need the *inverse kinematics* of a robot manipulator arm in order to find the joint positions that are needed to put the reference system,  $\{e\}$ , fixed to  $P_e$ , in a specified position and orientation, with respect to the base reference system,  $\{0\}$ .

For the case of a 6 DoF robot arm, we have 12 equations and 6 unknown parameters: the 6 joint positions. However, we get 9 equations from the rotation submatrix of  ${}^0T_6$ , and only 3 are independent equations. These, together with the 3 equations we get from the position submatrix (vector) of  ${}^0T_6$ , give us a total of 6 equations with 6 unknowns. These six independent equations are, in general, nonlinear transcendental equations which are often hard to solve.

Unlike for systems of linear equations, there are no general methods for solving systems of nonlinear equations. One of the possible methods is based on iterably computing the forward kinematics, but the convergence is not guaranteed.

The set of all solutions to the inverse kinematics of any particular robot manipulator arm defines the ***theoretical workspace*** of the arm, the volume of space that the robot can reach with  $P_e$ .

link length $a_i$	Number of solutions
$a_1 = a_3 = a_5 = 0$	up to 4
$a_3 = a_5 = 0$	up to 8
$a_3 = 0$	up to 16
all $\neq 0$	up to 16

Table 7.11: Number of solutions for inverse kinematics.

### 7.6.1 Multiple Solutions

Unlike systems of linear equations, systems of nonlinear equations can have multiple solutions, and, in general, this is the case for the inverse kinematics equations for a serial robot manipulator arm.

The problem is then to choose between the two (or more) solutions.

A criterion commonly used for making this choice is to use the solution closest to the current configuration, since this minimises the distance the joints have to move to get to this next configuration.

However, in practice, the physical limitations on the ranges of movement of each of the joints must also be taken into account.

In moving from one configuration to another, no joint value can go out of range. If this is the case, then one of the multiple solutions must be chosen, though this might result in a substantial change from the configuration, and thus a large and perhaps fast and unexpected movement of the entire robot arm.

In general, the more nonzero link parameters there are in the robot geometry, the more ways there will be to reach a particular location with  $P_e$ , i.e., the more multiple solutions there will be in the inverse kinematics.

For example, in the case of a 6 DoF angular robot arm, the number of multiple solutions is related to the number of nonzero link length parameters,  $a_i$ , there are, see table 7.11.

There are basically two kinds of methods for obtaining solutions for the inverse kinematics:

- Analytical methods that give closed form solutions, and
- Numerical methods that give iterative solutions.

Only in special cases robot manipulator arms with 6 DoFs have analytical solutions for their inverse kinematics.

A sufficient condition for the existence of a closed form solution, for a 6 DoFs angular robot arm, is to have three consecutive joint axes that intersect at a point. An example of this case is all robots with RPR wrists.

Since computing closed form solutions is much cheaper, and also much faster, than computing iterative numerical solutions, many robot manipulator geometries satisfy this condition, though this typically make them mechanically and structurally more complicated and costly to build.

Keeping link twists to zero or plus or minus ninety degrees may also allow analytical solutions to exist.

### 7.6.2 Inverse Kinematics of a Serial Robot

The proposed method to compute the inverse kinematics consists of three steps:

1. The kinematic decoupling of the robot. By means of this method we divide the kinematic chain into two kinematic sub-chains: the positioner and the wrist.
2. Geometrically solving the inverse kinematics of the positioner.
3. Analytically solving the inverse kinematics of the wrist.

As mentioned above, there is no general method to solve inverse kinematics of serial manipulators. To keep a clear and simple explanation, the inverse kinematic method is presented with a sample of a 5 DoF robot (See figure 7.15). Figure 7.16 shows the Denavit-Hartenberg frames associated with this robot.

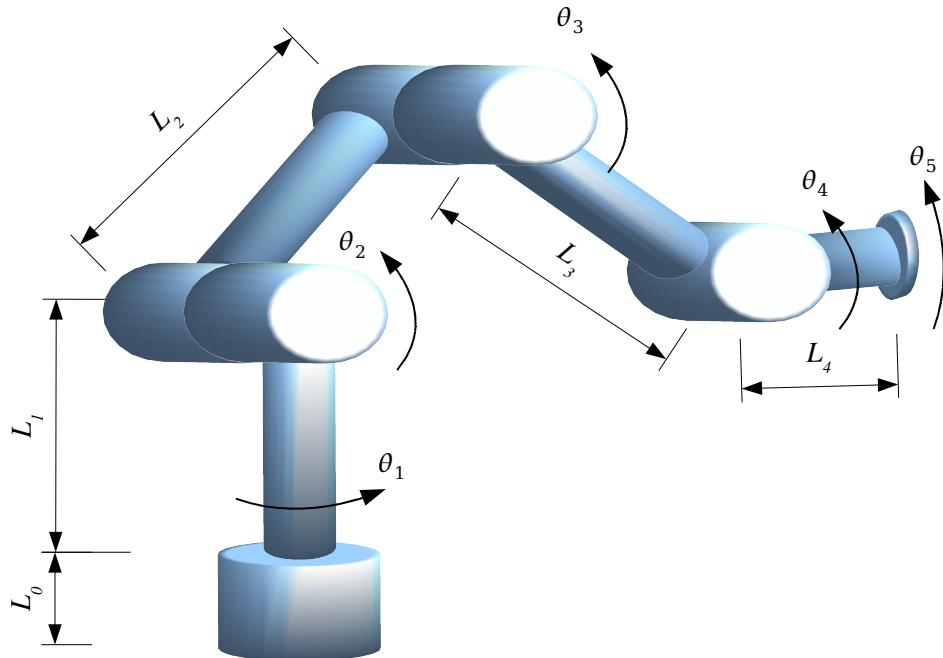


Figure 7.15: Example 1: a 5 DoF anthropomorphical robot.

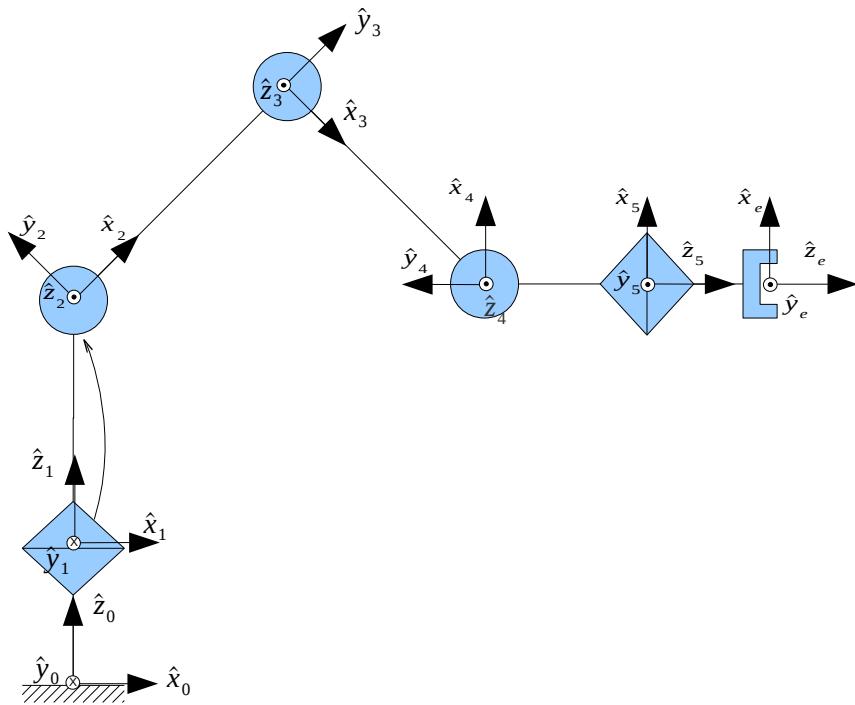


Figure 7.16: Example 1: Denavit frames associated with a 5 DoF anthropomorphical robot.

### 7.6.2.1 Kinematic Decoupling

When we have to solve the inverse kinematics of a 6 DoF robot, we ALWAYS have to test if its wrist axes intersect at a point. If this occurs, then by means of the kinematic decoupling method we can divide the kinematic chain of the wrist from the rest of the robot. Thus, we do not have one 6-DoF kinematic chain to solve, but 2 decoupled 3-DoF kinematic chains.

The method of decoupling involves two steps:

- Finding the point at which the wrist axes intersect (two in the case of robots with 2 DoF wrists). In figure 7.17, the figure on the left shows that the axes  $\hat{z}_4$  and  $\hat{z}_5$  intersect at  $P_4$ . Thus  $P_4$  is the point at which we will do the decoupling. After the decoupling we have two kinematic chains to solve isolated: the wrist (2 DoF), and the positioner (3 DoF).
- Calculating the cartesian coordinates of the point at which we have to decouple the robot. In the case of figure 7.17, we have to calculate the cartesian coordinates of  $P_4$ . To cope with that problem, first we have to notice that we know the numerical values of the homogeneous transformation between  $\{0\}$  and  $\{e\}$  since we are solving the inverse kinematic problem, that is, we know the numerical values of the  ${}^0T_e$  matrix:

$${}^0T_e = \begin{bmatrix} {}^0R_e & {}^0P_e \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (7.32)$$

Thus, we can easily obtain the coordinates of  ${}^0P_4$  from the robot's end point ( ${}^0P_e$ ) by translating the end-point coordinates a given distance along a given axis. In the example, the translation is  $L_4$  (figure 7.15), along the  $\hat{z}_e$  (figure 7.16)<sup>3</sup>. From the definition of the rotation matrices we also know that  ${}^0\hat{z}_e$  is the last column of  ${}^0R_e$ <sup>(4)</sup>. Therefore:

$${}^0P_4 = {}^0P_e - L_4 {}^0\hat{z}_e \quad (7.33)$$

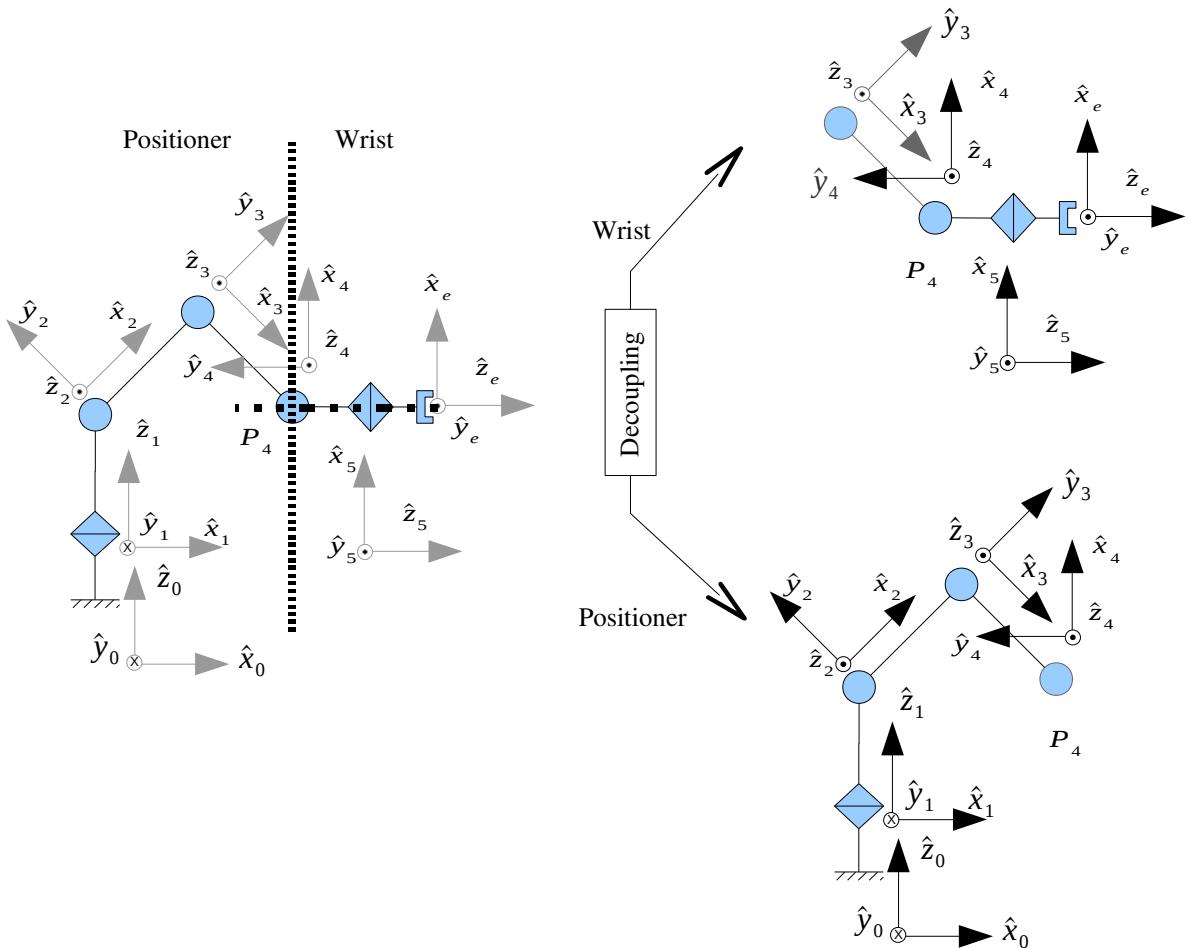


Figure 7.17: Example 1: Kinematic decoupling of a 5 DoF anthropomorphical robot.

Now we can solve the inverse kinematic problems of the positioner and the wrist independently.

<sup>3</sup>Notice that this is a particular solution for this robot. In other examples, the axis could be  $\hat{x}_e$  or  $\hat{y}_e$ , depending on the orientation of frame  $\{e\}$ .

### 7.6.2.2 Positioner's Inverse Kinematics, Geometric Method

This method is suitable for robots with few DoFs (3 or less) and is normally used to solve the positioner mechanism after decoupling the robot's wrist. It consists of finding geometric relationships in which there is a dependence on the cartesian coordinates of the robot's end point. If we are solving a 3-DoF robot, or with the cartesian coordinates of the point at which we are decoupling the robot ( $P_4$  in the case of the example in figure 7.17)<sup>5</sup>.

One of the most complex problems that can be solved using this method is shown below as example 1 (figure 7.18)<sup>6</sup>. That example is one of the possible solutions of the inverse kinematics of a 3-DoF anthropomorphic robot (or just the positioner of an anthropomorphic robot). In this kind of robot, the first DoF, also known as *waist DoF*, is solved and then the two DoFs grouped into the 'arm', that is, the *shoulder and the elbow DoFs*.

In order to present a more generic example and a simpler formulation, once the wrist has been splitted and calculated the coordinates of the origin of the 4<sup>th</sup> frame, frame  $e'$  is defined. The cartesian coordinates relationship between both frames is:

$${}^0P_{e'} = \begin{bmatrix} {}^0P_{e'x} \\ {}^0P_{e'y} \\ {}^0P_{e'z} \end{bmatrix} = \begin{bmatrix} -{}^0P_{y4} \\ -{}^0P_{z4} \\ {}^0P_{x4} \end{bmatrix} \quad (7.34)$$

#### **Waist DoF :**

The front view of figure 7.18 depicts the waist DoF. Notice its value can be calculated by:

$$\theta_1 = \arctan_2 \left( \frac{{}^0P_{e'y}}{{}^0P_{e'x}} \right) \quad (7.35)$$

Remember that  $P_{e'x}$ ,  $P_{e'y}$  and  $P_{e'z}$  are the cartesian coordinates of the positioner mechanism's end-point  $P_{e'}$ <sup>7</sup> and the function  $\arctan_2$  is the fourth-quadrant version of the arctan function.

#### **Shoulder and Elbow DoFs :**

To solve the Shoulder ( $\theta_2$ ) and Elbow ( $\theta_3$ ) DoFs, we have to analyze the front view of figure 7.18.

<sup>5</sup>The example in this section will be solved as if we were solving the inverse kinematics of a 3-DoF robot. In the case of solving the positioner of a decoupled robot, all references given to the robot's end-point should be changed to reference the point at which we have decoupled the robot.

<sup>6</sup>There are two views, the top and the front, to depict the three angles for each DoF in real magnitude. The top is for the first one and the front is for the last two.

<sup>7</sup>That coincides with the place where we have splitted the wrist.

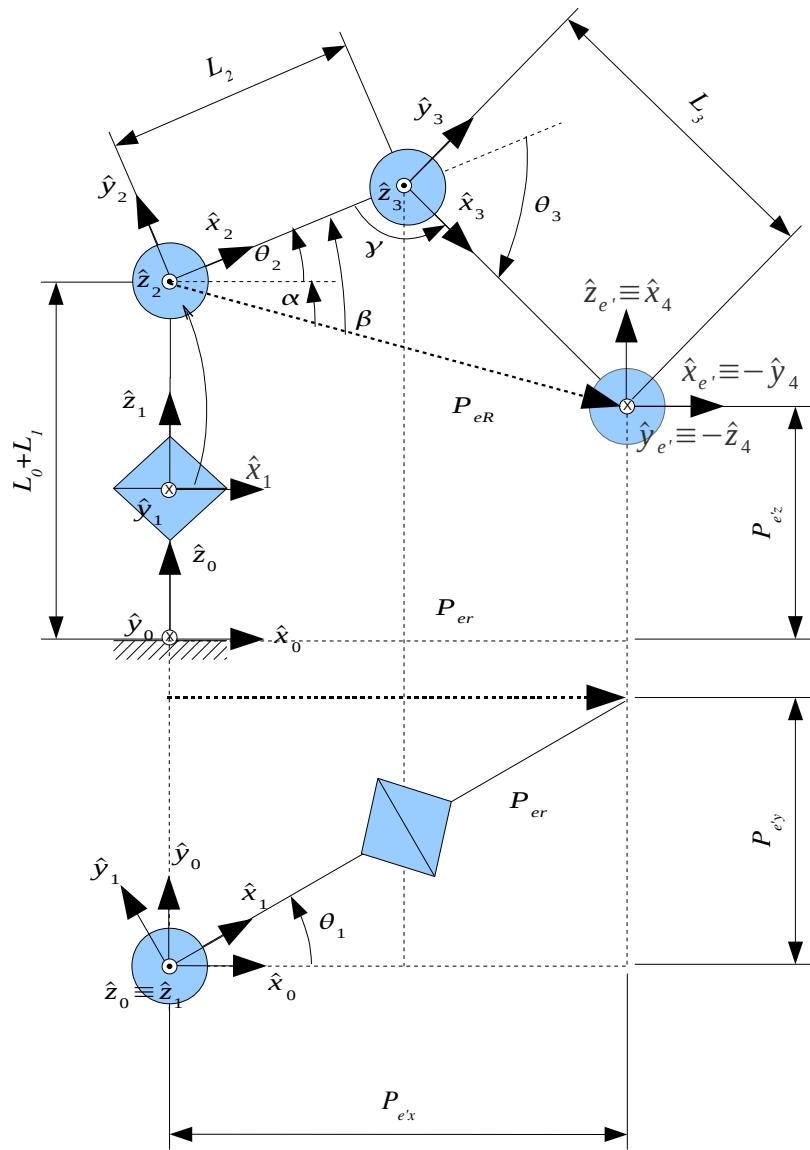


Figure 7.18: Example 1: one solution for the inverse kinematics of an anthropomorphic positioner.

Notice that the angles  $\theta_2$  and  $\theta_3$  can be expressed as below:

$$\theta_2 = \beta - \alpha \quad (7.36)$$

$$\theta_3 = \gamma - \pi \quad (7.37)$$

Where  $\alpha$ ,  $\beta$  and  $\gamma$  are auxiliary angles. Now let's see how to calculate them. But first we need to define a couple of auxiliary distances:

$$P_{e'r} = \sqrt{P_{e'x}^2 + P_{e'y}^2} \quad (7.38)$$

$$P_{e'R} = \sqrt{P_{e'x}^2 + P_{e'y}^2 + (P_{e'z} - (L_0 + L_1))^2} \quad (7.39)$$

Thanks to those distances we can easily define the three angles:

- $\alpha$  is an angle defined by a right triangle in which the hypotenuse is  $P_{e'R}$ , the opposite leg<sup>8</sup> is  $\|P_{e'z} - (L_0 + L_1)\|$  and the adjacent leg is  $P_{e'r}$ .  $\alpha$  can be obtained by the common trigonometric formula:

$$\alpha = \arccos\left(\frac{P_{e'r}}{P_{e'R}}\right) = \arctan2\left(\frac{\|P_{e'z} - (L_0 + L_1)\|}{P_{e'r}}\right) \quad (7.40)$$

- $\beta$  and  $\gamma$  are two of the angles of a triangle defined by  $L_2$ ,  $L_3$  and  $P_{e'R}$  distances. To calculate them we have to apply the cosine theorem<sup>9</sup>:

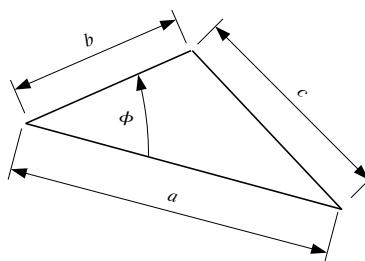
$$\beta = \arccos\left(\frac{P_{e'R}^2 + L_2^2 - L_3^2}{2P_{e'R}L_2}\right) \quad (7.41)$$

$$\gamma = \arccos\left(\frac{L_2^2 + L_3^2 - P_{e'R}^2}{2L_2^2L_3^2}\right) \quad (7.42)$$

---

<sup>8</sup>cathetus

<sup>9</sup>Cosine theorem  $\rightarrow c^2 = a^2 + b^2 - 2ab \cos \phi$  with:



Notice that there is not an ambiguity of the quadrant to which  $\alpha$ ,  $\beta$ , and  $\gamma$  angles belong. Since they are geometrical angles so they always belong to the first quadrant.

The next step is to combine these two latter expressions with the equations 7.40, 7.36 and 7.37, which yields the last two DoF values:

$$\theta_2 = \arccos\left(\frac{P_{e'R}^2 + L_2^2 - L_3^2}{2P_{e'R}L_2}\right) - \arctan2\left(\frac{\|P_{e'z} - (L_0 + L_1)\|}{P_{e'R}}\right) \quad (7.43)$$

$$\theta_3 = \arccos\left(\frac{L_2^2 + L_3^2 - P_{e'R}^2}{2L_2^2L_3^2}\right) - \pi \quad (7.44)$$

The solution we have just obtained is not the only valid solution of the inverse kinematics. It corresponds to the hypothesis the robot in the configuration depicted in figure 7.18. This solution is called *RIGHT-ARM & ELBOW-UP* configuration.

Then we have to guess which are the remaining configurations, as well as the remaining solutions. In this particular case, we can distinguish up to four configurations (figure 7.19).

Taking into account these possible extra configurations, the positioner inverse kinematics is not completed until we do not generalize the equations 7.35, 7.36 and 7.37 to cover all possible cases. After analyzing figure 7.19, it is easily proven that the general solution is:

$$\theta_1 = \begin{cases} \arctan2\left(\frac{P_{e'y}}{P_{e'x}}\right) & \rightarrow \text{RIGHT-ARM} \\ \arctan2\left(\frac{P_{e'y}}{P_{e'x}}\right) + \pi & \rightarrow \text{LEFT-ARM} \end{cases} \quad (7.45)$$

$$\theta_2 = \begin{cases} \beta - \alpha & \rightarrow \text{RIGHT-ARM \& ELBOW-UP} \\ -\beta - \alpha & \rightarrow \text{RIGHT-ARM \& ELBOW-DOWN} \\ \pi + \beta - \alpha & \rightarrow \text{LEFT-ARM \& ELBOW-UP} \\ \pi - \beta - \alpha & \rightarrow \text{LEFT-ARM \& ELBOW-DOWN} \end{cases} \quad (7.46)$$

$$\theta_3 = \begin{cases} \gamma - \pi & \rightarrow \text{RIGHT-ARM \& ELBOW-UP} \\ -\gamma + \pi & \rightarrow \text{RIGHT-ARM \& ELBOW-DOWN} \\ -\gamma + \pi & \rightarrow \text{LEFT-ARM \& ELBOW-UP} \\ \gamma - \pi & \rightarrow \text{LEFT-ARM \& ELBOW-DOWN} \end{cases} \quad (7.47)$$

I invite you to test the above results!!

### 7.6.2.3 Wrist's Inverse Kinematics

In the case of the wrist mechanism, an analytical method is preferred to a geometrical one. This method is based on the fact that the dot-product vectors depend on the cosinus of the angle between the above mentioned vectors. Since if we only calculate the cosinus

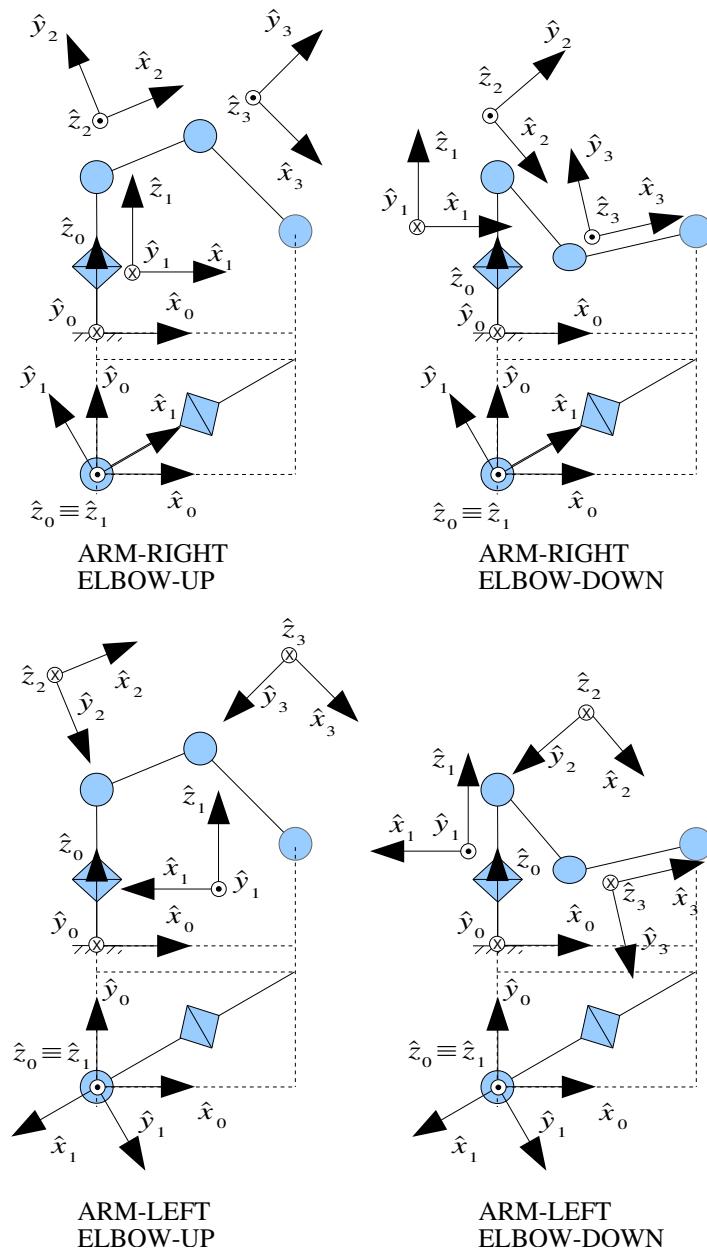


Figure 7.19: All inverse kinematic solutions of an anthropomorphic positioner.

we have an uncertainty about the quadrant angle, then we have to first calculate the cosinus and then the sinus for each DoF in the wrist.

The cosinus angle is calculated thanks to the dot product between  $\hat{x}_{i-1}$  and  $\hat{x}_i$ :

$$\hat{x}_i \cdot \hat{x}_{i-1} = \|\hat{x}_i\| \|\hat{x}_{i-1}\| \cos \theta_i = \\ = \hat{x}_{i_x} \hat{x}_{i-1_x} + \hat{x}_{i_y} \hat{x}_{i-1_y} + \hat{x}_{i_z} \hat{x}_{i-1_z} \quad (7.48)$$

The sinus angle should be calculated by the cross product. Since cross product formulae is more complex than dot product, the recommended method is the calculation of the cosinus (that is, dot product) by means of the vectors define the complementary angle of  $\theta_i$ .

Let us see the case of figure 7.20 in which we have a 5DoF anthropomorphic robot and provided the inverse kinematics of the positioner (first 3 DoF) we want to obtain the wrist inverse kinematics.

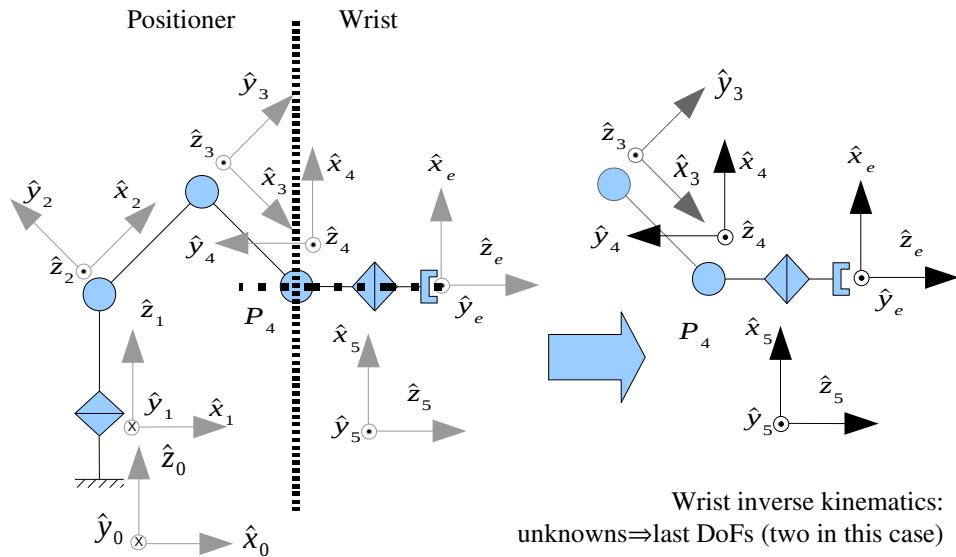


Figure 7.20: Wrist inverse kinematics of an 5DoF anthropomorphic manipulator.

Then we have to consider as input (known data) the transformation matrix from the robot base to the end-point (general input data for the inverse kinematics problem):

$${}^0T_e = \left[ \begin{array}{c|c} {}^0R_e & {}^0P_e \\ \hline [0 \ 0 \ 0] & 1 \end{array} \right] = \left[ \begin{array}{c|c} [{}^0\hat{x}_e \ {}^0\hat{y}_e \ {}^0\hat{z}_e] & {}^0P_e \\ \hline [0 \ 0 \ 0] & 1 \end{array} \right] \quad (7.49)$$

and the transformation matrix from the robot base to the 3<sup>rd</sup> DoF as it is supposed we have already solved the positioner inverse kinematics problem:

$${}^0T_3 = \left[ \begin{array}{c|c} {}^0R_3 & {}^0P_3 \\ \hline [0 \ 0 \ 0] & 1 \end{array} \right] = \left[ \begin{array}{ccc|c} 0\hat{x}_3 & 0\hat{y}_3 & 0\hat{z}_3 & {}^0P_3 \\ \hline [0 \ 0 \ 0] & & & 1 \end{array} \right] \quad (7.50)$$

Then, from D-H method, we know that  $\theta_4$  is the angle between  $\hat{x}_3$  and  $\hat{x}_4$  about  $\hat{z}_4$  (see figure 7.21) with  $\hat{x}_3$  known (the first column of  ${}^0R_3$ ):

$$\cos \theta_4 = \hat{x}_3 \cdot \hat{x}_4 \quad (7.51)$$

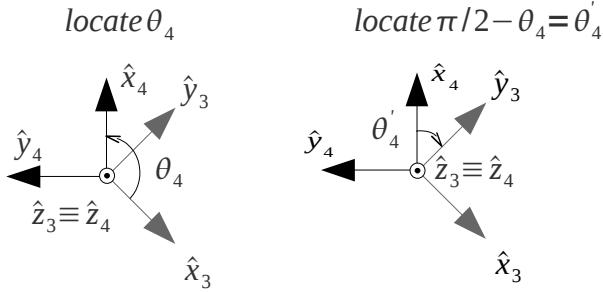


Figure 7.21: Wrist inverse kinematics of an 5DoF anthropomorphic manipulator.

We can calculate  $\hat{x}_4$  knowing that D-H method constraints  $\hat{x}_4$  to be perpendicular to  $\hat{z}_4$  and  $\hat{z}_5$ , thus:

$$\hat{x}_4 = (\hat{z}_4 \wedge \hat{z}_5) \quad (7.52)$$

From figure 7.21 we can also check that  $\hat{z}_4$  is parallel to  $\hat{z}_3$  and  $\hat{z}_3$  is known (as it is the third column of  ${}^0R_3$ ). In the case of  $\hat{z}_5$ , it is also easily proven that  $\hat{z}_5$  is equal to  $\hat{z}_e$  that is the last column of  ${}^0R_e$ .

Then, last equation become:

$$\hat{x}_4 = (\hat{z}_3 \wedge \hat{z}_e) = (\hat{z}_{y3}\hat{z}_{ze} - \hat{z}_{ye}\hat{z}_{z3} \quad \hat{z}_{xe}\hat{z}_{z3} - \hat{z}_{x3}\hat{z}_{ze} \quad \hat{z}_{x3}\hat{z}_{ye} - \hat{z}_{xe}\hat{z}_{y3}) \quad (7.53)$$

If we substitute this result in equation 7.51, we have that:

$$\cos \theta_4 = \hat{x}_3 \cdot \hat{x}_4 = \hat{x}_{x3}(\hat{z}_{y3}\hat{z}_{ze} - \hat{z}_{ye}\hat{z}_{z3}) + \hat{x}_{y3}(\hat{z}_{xe}\hat{z}_{z3} - \hat{z}_{x3}\hat{z}_{ze}) + \hat{x}_{z3}(\hat{z}_{x3}\hat{z}_{ye} - \hat{z}_{xe}\hat{z}_{y3}) \quad (7.54)$$

We obtain the sin as the cos of the complementary angle:

$$\cos \theta'_4 = \sin \theta_3 = \hat{y}_3 \cdot \hat{x}_4 = \hat{y}_{x3}(\hat{z}_{y3}\hat{z}_{ze} - \hat{z}_{ye}\hat{z}_{z3}) + \hat{y}_{y3}(\hat{z}_{xe}\hat{z}_{z3} - \hat{z}_{x3}\hat{z}_{ze}) + \hat{y}_{z3}(\hat{z}_{x3}\hat{z}_{ye} - \hat{z}_{xe}\hat{z}_{y3}) \quad (7.55)$$

Finally, we obtain the angle by means of  $\arctan_2^{10}$ :

$$\begin{aligned} \theta_4 &= \arctan(\sin \theta_4, \cos \theta_4) = & (7.56) \\ (\hat{y}_{x3}(\hat{z}_{y3}\hat{z}_{ze} - \hat{z}_{ye}\hat{z}_{z3}) + \hat{y}_{y3}(\hat{z}_{xe}\hat{z}_{z3} - \hat{z}_{x3}\hat{z}_{ze}) + \hat{y}_{z3}(\hat{z}_{x3}\hat{z}_{ye} - \hat{z}_{xe}\hat{z}_{y3}), \\ \hat{x}_{x3}(\hat{z}_{y3}\hat{z}_{ze} - \hat{z}_{ye}\hat{z}_{z3}) + \hat{x}_{y3}(\hat{z}_{xe}\hat{z}_{z3} - \hat{z}_{x3}\hat{z}_{ze}) + \hat{x}_{z3}(\hat{z}_{x3}\hat{z}_{ye} - \hat{z}_{xe}\hat{z}_{y3})) \end{aligned}$$

The last angle,  $\theta_5$  can be obtained following a similar procedure as the used for  $\theta_4$ . But in this case, we have:

$$\cos \theta_5 = \hat{x}_4 \cdot \hat{x}_5 = \hat{x}_4 \cdot \hat{x}_e = \hat{x}_{xe}(\hat{z}_{y3}\hat{z}_{ze} - \hat{z}_{ye}\hat{z}_{z3}) + \hat{x}_{ye}(\hat{z}_{xe}\hat{z}_{z3} - \hat{z}_{x3}\hat{z}_{ze}) + \hat{x}_{ze}(\hat{z}_{x3}\hat{z}_{ye} - \hat{z}_{xe}\hat{z}_{y3}) \quad (7.57)$$

$$\sin \theta_5 = \hat{x}_4 \cdot \hat{x}_5 = \hat{x}_4 \cdot \hat{x}_e = \hat{x}_{xe}(\hat{z}_{y3}\hat{z}_{ze} - \hat{z}_{ye}\hat{z}_{z3}) + \hat{x}_{ye}(\hat{z}_{xe}\hat{z}_{z3} - \hat{z}_{x3}\hat{z}_{ze}) + \hat{x}_{ze}(\hat{z}_{x3}\hat{z}_{ye} - \hat{z}_{xe}\hat{z}_{y3}) \quad (7.58)$$

---

<sup>10</sup>This function is the standard [arctan] but it takes into account the sign of the numerator (sin) and the denominator (cos) when guessing the right quadrant.

## Chapter 8

# The Jacobian Matrix

---

So far we have considered only the location (position and orientation) of the end-point,  $P_e$ , of the robot with respect to the fixed base reference system and the joint position needed to achieve it.

Sometimes we also need to consider and to control the rate of change of the location of  $P_e$ , not just its static location, to produce a constant speed straight line motion at  $P_e$ , for example.

We therefore need to know and to represent the relationship between the rates of change of the individual joint values ( $\dot{q}_1 \dot{q}_2 \dots \dot{q}_n$ ), and the rate of change of location of  $P_e$ , ( $\dot{P}_{xe} \dot{P}_{ye} \dot{P}_{ze} \dot{\psi}_{xe} \dot{\psi}_{ye} \dot{\psi}_{ze}$ ).

We call the matrix which represents this relationship the **Jacobian Matrix**,  $J$ , and this relationship is between the joint value rates of change and the rate of change of the location of  $P_e$ , is shown below.

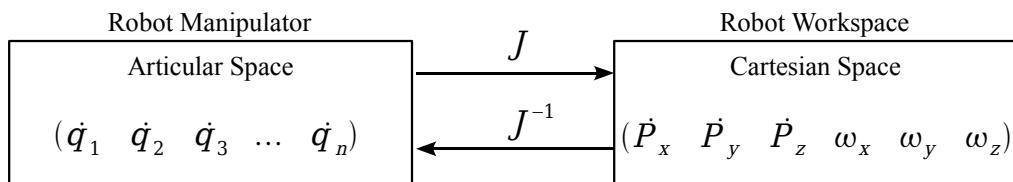


Figure 8.1: Jacobian relationship.

This relationship can be expressed more formally as follows:

$$\dot{P}_e = J\dot{q} \quad (8.1)$$

for the Forward Jacobian, and

$$\dot{q} = J^{-1}\dot{P}_e \quad (8.2)$$

for the Inverse Jacobian, where,

$$\dot{q} = [\dot{q}_1 \quad \dot{q}_2 \quad \dot{q}_3 \quad \cdots \quad \dot{q}_n]^T \quad (8.3)$$

and

$$\dot{P}_e = [\dot{P}_{x_e} \quad \dot{P}_{y_e} \quad \dot{P}_{z_e} \quad \dot{\psi}_{1_e} \quad \dot{\psi}_{2_e} \quad \dot{\psi}_{3_e}]^T \quad (8.4)$$

As  $J$  depends upon the instantaneous values of  $(\dot{q}_1 \ \dot{q}_2 \ \cdots \ \dot{q}_n)$ , so  $J$  will be different for each different set of values of  $(\dot{q}_1 \ \dot{q}_2 \ \cdots \ \dot{q}_n)$  in other words, for each different configuration of the robot arm.

To obtain the inverse Jacobian relation we need to invert  $J$ , which is, in general, hard. There are three different types of methods used to do this in the case of robot kinematic control:

- Invert  $J$  symbolically, which is only practical for very simple robot geometries.
- Numerically invert  $J$  for each configuration of the robot. This is computationally expensive, not always possible (when  $\|J\| = 0$ ,  $\not\exists J^{-1}$ ) and difficult.
- Derive  $J^{-1}$  directly from the Inverse Kinematics equations.

Note that, in general,  $J$  is not necessarily a square matrix. It depends upon the number of joints and thus on the number of degrees of freedom of the robot geometry. In this case it is not possible to compute the inverse Jacobian if it is required. However there are many techniques available to tackle with this problem. One of the most popular solution is the computation of the pseudoinverse.

$$J^+ = \begin{cases} (J^T J)^{-1} J^T & \text{when the Jacobian matrix has more rows than columns,} \\ & \text{as do most industrial robots} \\ J^T (J J^T)^{-1} & \text{when the Jacobian matrix has more columns than rows,} \\ & \text{which is the case of redundant robots that have more DoF} \\ & \text{than strictly needed} \end{cases} \quad (8.5)$$

## 8.1 Calculating the Jacobian Matrix

Formally speaking, a Jacobian Matrix is the matrix of which elements are the partial derivatives of a given function. Thus we can obtain the Jacobian Matrix of a robot,  $J$  by differentiating the equations we get for  $[\dot{q}_1 \ \dot{q}_2 \ \cdots \ \dot{q}_n]$ , from the Forward Kinematics.

Thus, for a  $n$ -DoF robot, the Forward Kinematics equations are:

$$\begin{bmatrix} P_{x_e} \\ P_{y_e} \\ P_{z_e} \\ \psi_{1e} \\ \psi_{2e} \\ \psi_{3e} \end{bmatrix} = \begin{bmatrix} \text{fkin}_x(q_1, q_2, q_3 \dots q_n) \\ \text{fkin}_y(q_1, q_2, q_3 \dots q_n) \\ \text{fkin}_z(q_1, q_2, q_3 \dots q_n) \\ \text{fkin}_{\psi_1}(q_1, q_2, q_3 \dots q_n) \\ \text{fkin}_{\psi_2}(q_1, q_2, q_3 \dots q_n) \\ \text{fkin}_{\psi_3}(q_1, q_2, q_3 \dots q_n) \end{bmatrix} \quad (8.6)$$

Differentiating these equations results in:

$$\begin{bmatrix} \dot{P}_{x_e} \\ \dot{P}_{y_e} \\ \dot{P}_{z_e} \\ \dot{\psi}_{1e} \\ \dot{\psi}_{2e} \\ \dot{\psi}_{3e} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n \frac{\partial \text{fkin}_x}{\partial q_i} \dot{q}_i \\ \sum_{i=1}^n \frac{\partial \text{fkin}_y}{\partial q_i} \dot{q}_i \\ \sum_{i=1}^n \frac{\partial \text{fkin}_z}{\partial q_i} \dot{q}_i \\ \sum_{i=1}^n \frac{\partial \text{fkin}_{\psi_1}}{\partial q_i} \dot{q}_i \\ \sum_{i=1}^n \frac{\partial \text{fkin}_{\psi_2}}{\partial q_i} \dot{q}_i \\ \sum_{i=1}^n \frac{\partial \text{fkin}_{\psi_3}}{\partial q_i} \dot{q}_i \end{bmatrix} \quad (8.7)$$

Which can be written in matrix form as:

$$\begin{bmatrix} \dot{P}_{x_e} \\ \dot{P}_{y_e} \\ \dot{P}_{z_e} \\ \dot{\psi}_{1e} \\ \dot{\psi}_{2e} \\ \dot{\psi}_{3e} \end{bmatrix} = \begin{bmatrix} \frac{\partial \text{fkin}_x}{\partial q_1} & \frac{\partial \text{fkin}_x}{\partial q_2} & \dots & \frac{\partial \text{fkin}_x}{\partial q_n} \\ \frac{\partial \text{fkin}_y}{\partial q_1} & \frac{\partial \text{fkin}_y}{\partial q_2} & \dots & \frac{\partial \text{fkin}_y}{\partial q_n} \\ \frac{\partial \text{fkin}_z}{\partial q_1} & \frac{\partial \text{fkin}_z}{\partial q_2} & \dots & \frac{\partial \text{fkin}_z}{\partial q_n} \\ \frac{\partial \text{fkin}_{\psi_1}}{\partial q_1} & \frac{\partial \text{fkin}_{\psi_1}}{\partial q_2} & \dots & \frac{\partial \text{fkin}_{\psi_1}}{\partial q_n} \\ \frac{\partial \text{fkin}_{\psi_2}}{\partial q_1} & \frac{\partial \text{fkin}_{\psi_2}}{\partial q_2} & \dots & \frac{\partial \text{fkin}_{\psi_2}}{\partial q_n} \\ \frac{\partial \text{fkin}_{\psi_3}}{\partial q_1} & \frac{\partial \text{fkin}_{\psi_3}}{\partial q_2} & \dots & \frac{\partial \text{fkin}_{\psi_3}}{\partial q_n} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dots \\ \dot{q}_n \end{bmatrix} \quad (8.8)$$

Which can be rewritten as:

$$\dot{P}_e = J^* \dot{q} \quad (8.9)$$

where  $J^*$  is the Jacobian Matrix. Notice that this Jacobian Matrix relates the temporal derivatives of joint positions, that is, joint speeds with the temporal derivatives of end-

point position. However, the temporal derivatives of end-point position do not always represent the end-point speed:

$$\begin{bmatrix} \dot{P}_{x_e} \\ \dot{P}_{y_e} \\ \dot{P}_{z_e} \\ \dot{\psi}_{1e} \\ \dot{\psi}_{2e} \\ \dot{\psi}_{3e} \end{bmatrix} = \begin{bmatrix} v_{x_e} \\ v_{y_e} \\ v_{z_e} \\ \dot{\psi}_{1e} \\ \dot{\psi}_{2e} \\ \dot{\psi}_{3e} \end{bmatrix} \neq \begin{bmatrix} v_{x_e} \\ v_{y_e} \\ v_{z_e} \\ \omega_{x_e} \\ \omega_{y_e} \\ \omega_{z_e} \end{bmatrix} \quad (8.10)$$

Equation 8.10 illustrates the fact that the temporal derivative of position corresponds to a translation speed, but neither the temporal derivative of Euler Angles nor of Quaternions corresponds to an angular speed. However, it is possible to find a relationship between the temporal derivatives of the representation of orientations and the angular speed. That relationship is verified by pre-multiplying the temporal derivative vector by a matrix  $B$ . This matrix  $B$  is unique and its expression depends on the used representation:

- In the case of Euler Angles →

$$\begin{bmatrix} \omega_{x_e} \\ \omega_{y_e} \\ \omega_{z_e} \end{bmatrix} = B_{\psi_{3x3}} \begin{bmatrix} \dot{\psi}_{1e} \\ \dot{\psi}_{2e} \\ \dot{\psi}_{3e} \end{bmatrix} \quad (8.11)$$

For the Roll-Pitch-Yaw convention, the expression for  $B_\psi$  is:

$$B_{\psi_{3x3}} = \begin{bmatrix} 0 & -\sin \psi_{3e} & \cos \psi_{3e} \cos \psi_{2e} \\ 0 & \cos \psi_{3e} & \sin \psi_{3e} \cos \psi_{2e} \\ 1 & 0 & -\sin \psi_{2e} \end{bmatrix} \quad (8.12)$$

- In the case of Rotation Axis- Rotation Angle representation →

$$\begin{bmatrix} \omega_{x_e} \\ \omega_{y_e} \\ \omega_{z_e} \end{bmatrix} = B_{u_{3x4}} \begin{bmatrix} \dot{\phi}_e \\ \dot{u}_{xe} \\ \dot{u}_{ye} \\ \dot{u}_{ze} \end{bmatrix} \quad (8.13)$$

where:

$$B_{u_{3x4}} = \begin{bmatrix} u_{xe} & \beta_e & -\alpha_e u_{ze} & \alpha_e u_{ye} \\ u_{ye} & \alpha_e u_{ze} & \beta_e & -\alpha_e u_{xe} \\ u_{ze} & -\alpha_e u_{ye} & \alpha_e u_{ze} & \beta_e \end{bmatrix} \quad (8.14)$$

where  $\alpha_e = 1 - \cos \varphi_e$  and  $\beta_e = \sin \varphi_e$ .

Therefore, the Jacobian Matrix that relates Cartesian velocity with articular speed is:

$$\begin{bmatrix} v_e \\ \omega_e \end{bmatrix} = J\dot{q} = B^{-1}J^*\dot{q} \quad (8.15)$$

## 8.2 Speed Propagation Method to Compute the Jacobian Matrix

This method only presents real advantages for serial manipulators. It involves a iterative process that starts with the hypothesis that the linear and angular speed at the base of the robot is known (which is normally equal to zero because the robot is attached to a static surface). All joint speed is computed from the base of the robot to the end-effector by the velocity propagation equations:

- angular speed propagation  $\rightarrow {}^{i+1}\omega_{i+1} = {}^{i+1}R_i{}^i\omega_i + \dot{\theta}_{i+1}{}^{i+1}z_{i+1}$
- linear speed propagation  $\rightarrow {}^{i+1}v_{i+1} = {}^{i+1}R_i({}^iv_i + {}^i\omega_i \wedge {}^iP_{i+1}) + \dot{d}_{i+1}{}^{i+1}z_{i+1}$

where  ${}^iP_{i+1}$  is the vector that connects the origins of frames  $\{i\}$  and  $\{i+1\}$ .

After finishing the iterative process, the jacobian matrix is obtained as the matrix form of the last given result ( ${}^ev_e$  and  ${}^e\omega_e$ ). The summary of the whole process is shown in figure 8.2.

## 8.3 Static of Robotic Manipulators

If we consider a static robotic manipulator and we exert a force/torque on the end-point, every robot actuators should exert forces/torques to keep the robot stopped. The analysis of this problem corresponds to the static analysis of the robot, or, rather the analysis of the forces/torque applied to the robot that keeps the manipulator static.

From a mathematical point of view, the relationship between the applied force/torque on the end-point of the robot and the forces/torque involved in the robot actuators to keep the manipulator stopped are also given by the Jacobian Matrix of the manipulator:

$$\tau = J^T \begin{bmatrix} f_e \\ n_e \end{bmatrix} \quad (8.16)$$

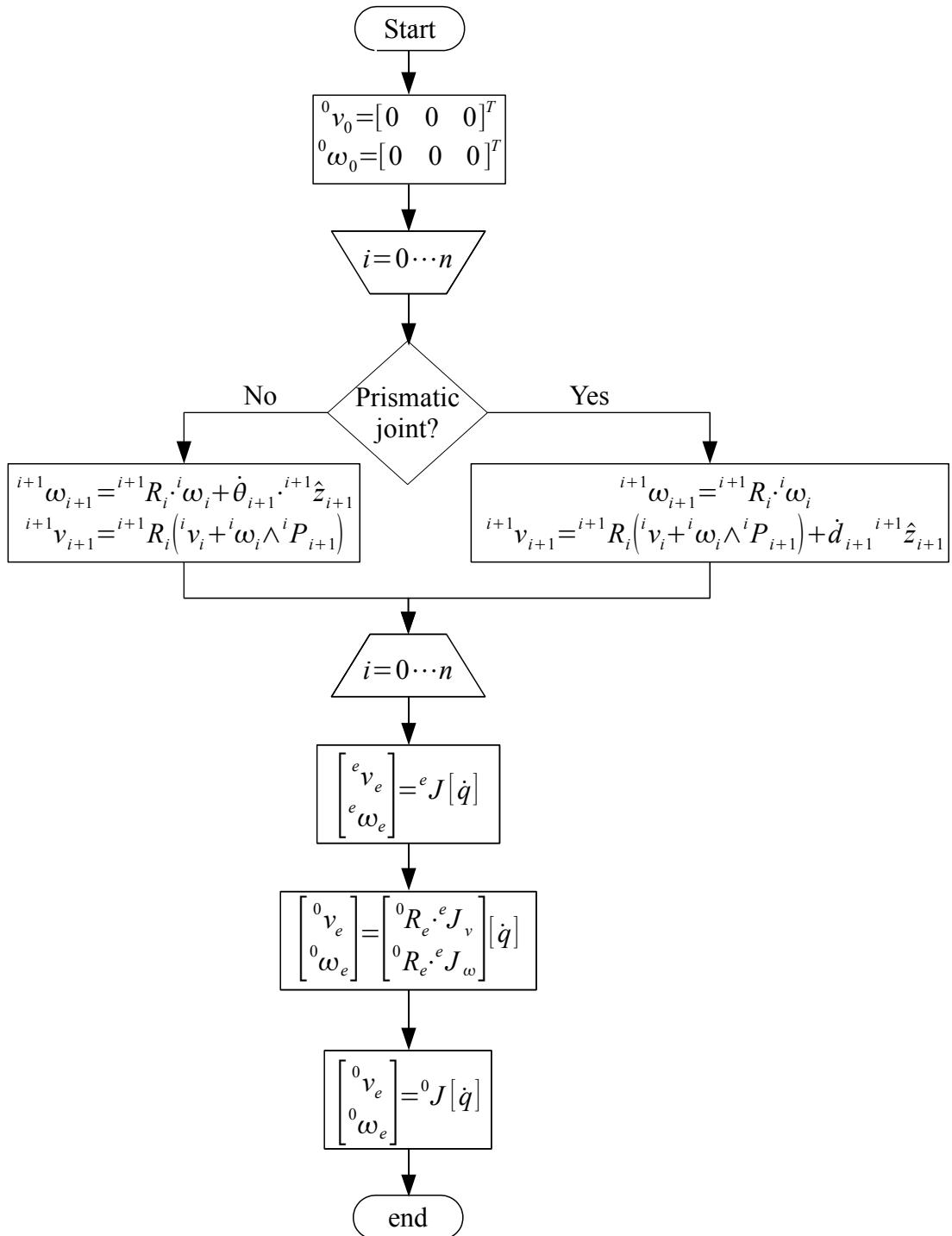


Figure 8.2: Calculation of the Jacobian Matrix via velocity propagation method.

where  $\tau$  is the vector of force/torque applied by every actuator, and  $f_e$  and  $n_e$  are the force and torque exerted on the end-point.

Notice that the Jacobian Matrix given in equation 8.16 is the same as the Jacobian Matrix in equation 8.15. This equality is easily proven by the Virtual Work Theorem:

$$Pot_{cartesian} = Pot_{joint} \quad (8.17)$$

$$\begin{aligned} [f_e^T \ n_e^T] \begin{bmatrix} v_e \\ \omega_e \end{bmatrix} &= \tau^T [\dot{q}] \\ [f_e^T \ n_e^T]^T J[\dot{q}] &= \tau^T [\dot{q}] \\ J^T \begin{bmatrix} f_e \\ n_e \end{bmatrix} &= \tau \end{aligned}$$

Thanks to this property, the Jacobian Matrix can also be calculated by performing a static equilibrium of the manipulator. In the case of the serial manipulators, an iterative method called Force/Torque propagation can be applied.

## 8.4 Force/Torque Propagation Method to Compute the Jacobian Matrix

As with the velocity propagation, this method only presents real advantages for serial manipulators. It involves a iterative process that starts with the hypothesis that the force and torque applied on the end-point is known (which is normally equal to a generic constant  $[f_x \ f_y \ f_z \ n_x \ n_y \ n_z]$ ), and then every torque/force exerted on each joint is computed from the end-point of the robot to the 1<sup>st</sup> DoF<sup>(1)</sup> by the force/torque propagation equations:

- force  $\rightarrow {}^i f_i = {}^i R_{i+1} {}^{i+1} f_{i+1} = {}^i f_{i+1}$
- torque  $\rightarrow {}^i n_i = {}^i R_{i+1} {}^{i+1} n_{i+1} + {}^i P_{i+1} \wedge {}^i f_{i+1}$

After finishing the iterative process, the force/torque exerted on every joint is obtained using the following equations:

- $\tau_i = {}^i f_i {}^i z_i \rightarrow$  in the case of prismatic DoF
- $\tau_i = {}^i n_i {}^i z_i \rightarrow$  in the case of rotational DoF

---

<sup>1</sup>Notice that neither  ${}^0 f_0$  nor  ${}^0 n_0$  need to be calculated

In other words, the above equations imply the extraction of the  $f_z$  value (if the DoF is prismatic) or  $n_z$  (if the DoF is rotational).

After obtaining  $\tau_1 \cdots \tau_n$ , the Jacobian matrix is obtained as the matrix form of the equations for  $\tau_1 \cdots \tau_n$ . The summary of the whole process is shown in figure 8.3.

## 8.5 Singularities and Singular Configurations

If, for any particular configuration of the robot arm that is in movement, the motion performance of a robot can be degraded abruptly, that configuration is a **singular configuration**. In those configurations, the Jacobian Matrix loses range and in the case of square Jacobian Matrices they become singular.

We can measure how far the manipulator is from a singular configuration thanks to the following **manipulability index**

$$\kappa = \sqrt{\det(J^T J)} \quad (8.18)$$

In the case of square matrices, that index becomes the determinant:

$$\kappa = \det(J) \quad (8.19)$$

In a singular configuration, the manipulability index becomes zero:

$$\kappa = 0 \rightarrow \text{singular configuration} \quad (8.20)$$

For a singular configuration,  $J^{-1}$  does not exist, it is not mathematically defined when  $\|J\| = 0$ , so, as a result, the relation between articular speed ( $\dot{q}$ ) and the cartesian speed ( $V_e$ ) is not defined. Consequently, neither is the relationship between articular force/torque ( $\tau$ ) and cartesian force/torque ( $f & n$ ).

Singular configuration occurs when two or more joint axes become aligned in space. When this happens, the serial robot geometry effectively loses one (or more) independent degrees of freedom: two or more of the degrees of freedom become mutually dependent.<sup>2</sup>

When that configurations near to a singular configuration, the elements of  $J^{-1}$  can have very large values. This means that, for even small values of  $V_e$  there will be large values in  $\dot{q}$ , which will most likely be too large for the joint motors to actually generate.

The loss of one or more effective degrees of freedom thus occurs not just at a singular configuration, but also in a region (a volume in joint space) around it.

According to Craig, singular configurations are classified into two different kinds:

---

<sup>2</sup>In the case of parallel robotics, the geometry gains one (or more) degrees of freedom and losses stiffness.

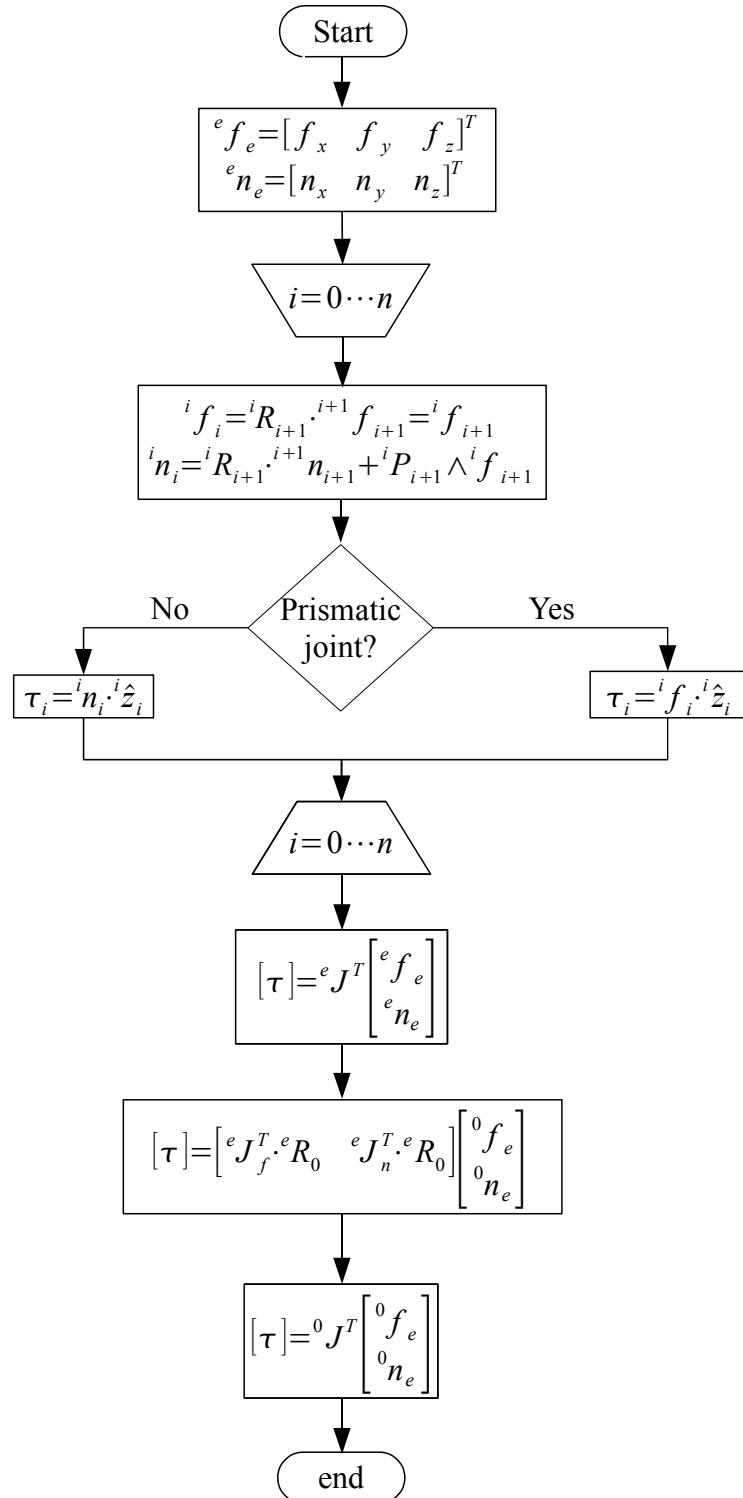


Figure 8.3: Calculation of the Jacobian Matrix via force/torque propagation method.

- **Workspace Boundary Singularities**, which occur when the robot manipulator is fully extended or folded onto itself so that  $P_e$  is at or near the boundary of the robot workspace. In such configurations, one or more joints will be at their limits of range of movement, so that they will not be able to maintain any movement at any given speed. This effectively makes  $J$  a singular matrix.
- **Workspace Interior Singularities**, which occur inside the robot workspace away from any of the boundaries and are generally caused by two or more joint axes become aligned.

Three different courses of action can be taken to avoid singular configurations:

- Increase robot manipulator's degrees of freedom, perhaps by attaching a tool or gripper to  $P_e$ , which has one or two degrees of freedom itself.
- Restrict the movements that the robot can be programmed to make so as to avoid any singular configurations.
- Dynamically modify  $J$  to remove the offending terms, and thus return  $\|J\| \neq 0$ . This means identifying the column and row of  $J$  that need to be removed.

## Chapter 9

# Path planning

---

### 9.1 Introduction

From the inverse kinematics of a robot manipulator arm we can know what joint values we need for any particular location of  $P_e$ . Thus, to make  $P_e$  follow some specified trajectory in Cartesian space (a straight line or circular arc, for example), we must first calculate how each joint must move, specify this joint motion in terms of a series of joint position reference values, and then input these to the controller of each joint at an appropriate rate in time. The process of turning a specified Cartesian space trajectory of  $P_e$  into sequences of appropriate joint position reference values, one sequence for each joint, is called **Trajectory Generation** (figure 9.1).

The relationship between the robot program, trajectory generation, and control level (joint control) of the robot is illustrated in figure 9.1.

The trajectory generation process of a robot manipulator arm consists of the following six steps:

1. Convert the movement specification from the program into an analytical and continuous Cartesian Space trajectory with respect to  $\{0\}$ , that is, defining a mathematical model, in Cartesian Space, that describes the desired trajectory:

$$P_e = [p_{e_x} \ p_{e_y} \ p_{e_z} \ \psi_{e_x} \ \psi_{e_y} \ \psi_{e_z}]^T \quad (9.1)$$

And do the same for the location change rate (speed), if necessary, to obtain:

$$\dot{P}_e = [\dot{p}_{e_x} \ \dot{p}_{e_y} \ \dot{p}_{e_z} \ \dot{\psi}_{e_x} \ \dot{\psi}_{e_y} \ \dot{\psi}_{e_z}]^T \quad (9.2)$$

2. Sample the Cartesian trajectory  $P_e$  to obtain a finite number,  $m$ , of sample points on the continuous trajectory so we have:

$$P_e^{(i)} = [p_{e_x}^{(i)} \ p_{e_y}^{(i)} \ p_{e_z}^{(i)} \ \psi_{e_x}^{(i)} \ \psi_{e_y}^{(i)} \ \psi_{e_z}^{(i)}]^T, i = 1 \dots m \quad (9.3)$$

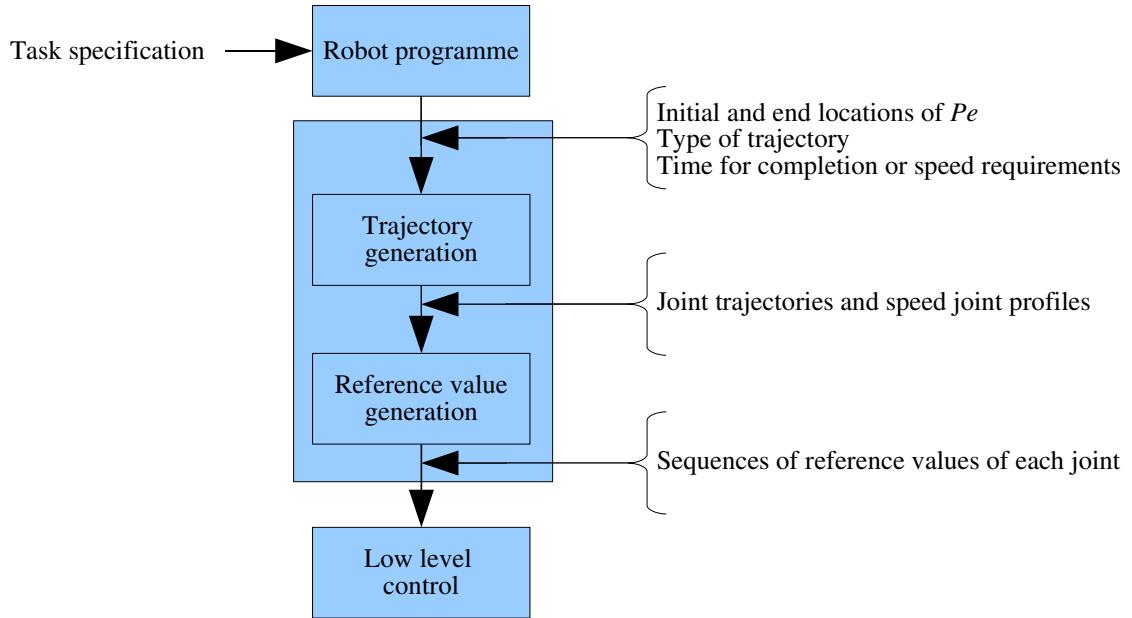


Figure 9.1: Trajectory Generation vs. Joint Control.

And do the same for the location change rate of the trajectory, if necessary, to produce:

$$\dot{P}_e^{(i)} = \begin{bmatrix} \dot{p}_{e_x}^{(i)} & \dot{p}_{e_y}^{(i)} & \dot{p}_{e_z}^{(i)} & \dot{\psi}_{e_x}^{(i)} & \dot{\psi}_{e_y}^{(i)} & \dot{\psi}_{e_z}^{(i)} \end{bmatrix}^T, i = 1 \dots m \quad (9.4)$$

3. Using the inverse kinematics relation of the robot manipulator arm, convert each Cartesian trajectory sample,  $P_e^{(i)}$ , into a corresponding joint space vector  $q_e^{(i)}$ .

$$q^{(i)} = \begin{bmatrix} q_1^{(i)} & q_2^{(i)} & q_3^{(i)} & \dots & q_n^{(i)} \end{bmatrix}^T, i = 1 \dots m \quad (9.5)$$

In this step, the possibility of multiple solutions to the inverse kinematics relation must be dealt with.

4. Using the inverse Jacobian relation, convert each instantaneous velocity vector,  $\dot{P}_e^{(i)}$ , into a corresponding joint speed vector,  $\dot{q}^{(i)}$ . In this step, the possibility of singular configurations must be dealt with.
5. Using the sequence of vectors  $q^{(i)}$  and  $\dot{q}^{(i)}$ ,  $i = 1 \dots m$ , generate continuous expressions  $q_j(t)$  and  $\dot{q}_j(t)$ ,  $j = 1 \dots n$  (for every joint), which pass through or sufficiently near to each joint space sample point,  $q^{(i)}$ , and velocity sample point,  $\dot{q}^{(i)}$ ,  $i = 1 \dots m$ , to produce continuous joint space trajectories for each joint.

6. Sample each continuous joint trajectory  $q_j(t)$  (and  $\dot{q}_j(t)$ ,  $j = 1 \dots n$ ), to generate a sequence of discrete reference values for each joint,  $q_j(kT)$  (and  $\dot{q}_j(kT)$ ,  $j = 1 \dots n$ ), where  $T$  is the sampling period used, and  $k$  is the sampling number.

To illustrate these six steps of the trajectory generation process, we will consider a simple 2D 2-DoF planar manipulator arm and a specified straight line motion of  $P_e$  (figure 9.2).

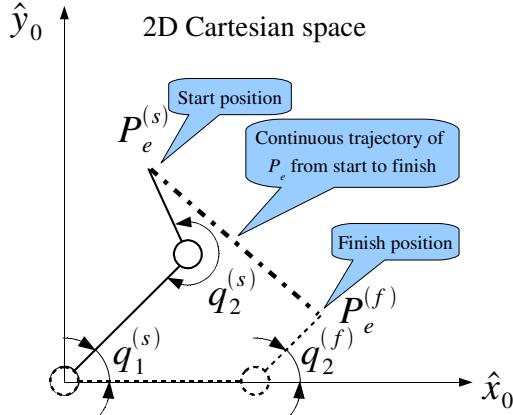


Figure 9.2: Definition of a straight line trajectory (cartesian space).

The continuous Cartesian space trajectory is therefore the straight line between  $P_e^{(s)}$  and  $P_e^{(f)}$ .

Step 2 is to sample the former continuous Cartesian space trajectory (figure 9.3).

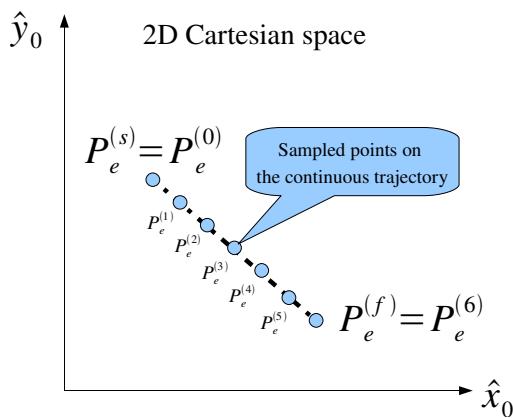


Figure 9.3: Sampling (in space) of the straight line trajectory (cartesian space).

Using the inverse kinematics relation, step 3 then converts each Cartesian space sample point into an  $n - \text{dimensional}$  joint space point (figure 9.4), where  $n$  is the number of DoF of the robot manipulator.

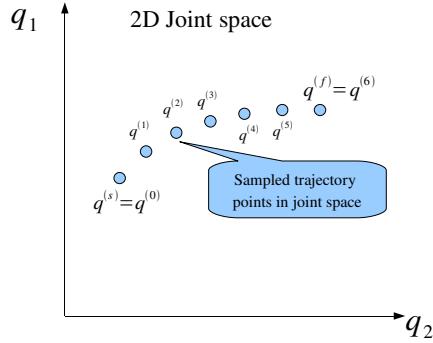


Figure 9.4: Sampled trajectory (joint space).

Step 4 involves fitting a smooth continuous curve to these joint space sample points (figure 9.5).

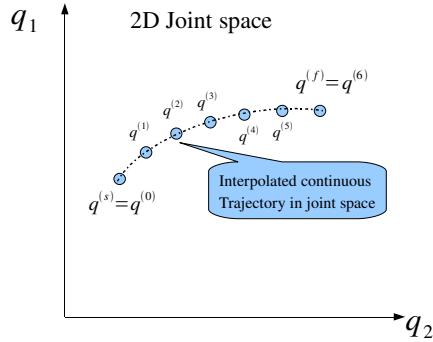


Figure 9.5: Interpolated trajectory (joint space).

And step 5 involves sampling, over time, this joint space trajectory (figure 9.6)...

...to generate a sequence of position reference values for the two robot arm joints (figure 9.7).

The Cartesian space sampling, inverse kinematic calculations, joint space fitting, and subsequent reference value generation, all introduce some error into the process. The resulting motion of  $P_e$  is therefore, in general, not exactly the same as the specified trajectory: there will always be some variation from this (figure 9.8).

By increasing  $T$  we can slow down the movement of  $P_e$ , and, similarly, by decreasing  $T$  we can accelerate (speed up) the movement of  $P_e$ . However, making  $T$  too small or too large may produce less precision in the movement of  $P_e$ , and it is important that  $T$  is never made too small compared to the slowest joint rate in the robot arm control loop.

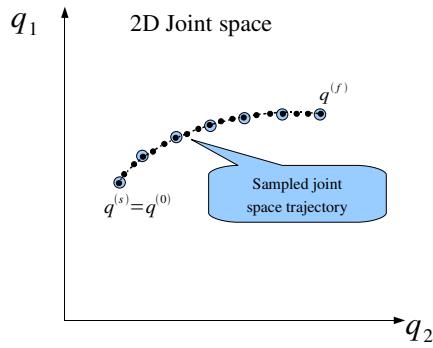


Figure 9.6: Sampling, in time, the trajectory (joint space).

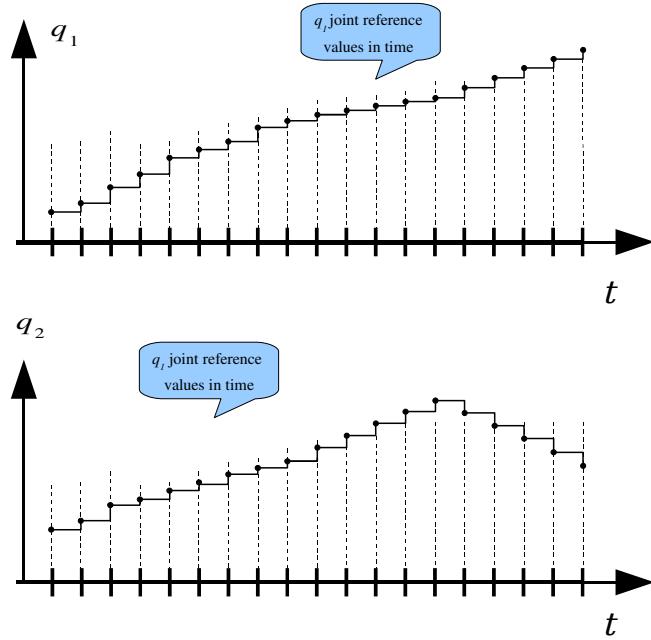


Figure 9.7: Joint reference values for every joint.

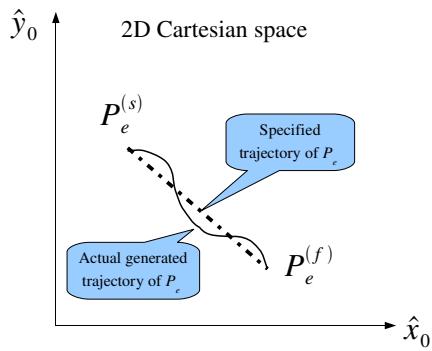


Figure 9.8: Specified trajectory vs. actual robot's trajectory.

## 9.2 Types of Trajectories in Kinematic Control

The trajectory of  $P_e$ , the motion of the Robot End-point location in Cartesian space, from  $P_e^{(s)}$  to  $P_e^{(f)}$ , can be generated by three different types of joint space trajectories. Only the last of which can be used to produce fully specified Cartesian space trajectories, such as straight lines or circular trajectories.

- Type 1: **Point-to-point trajectories**, in which each joint trajectory is generated completely independently of the other joint trajectories.

Each joint is moved from its starting position to its final position at some default rate.

In this type of trajectory generation we can distinguish two subtypes:

- **Joint-by-joint motion**, in which each joint is moved in turn in strict sequence (figure 9.9). Thus, for our simple 2D planar arm, we would have joint space trajectories as follows, together with the resulting Cartesian space trajectory of  $P_e$ . The total motion time is therefore the sum of each joint motion time, and the resulting motion of  $P_e$  is often rather jerky<sup>1</sup>.

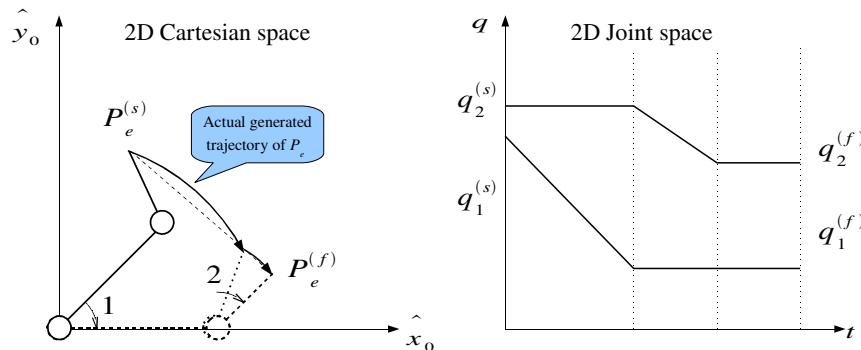


Figure 9.9: Point to point trajectories and joint by joint motion.

- **Simultaneous joint motion**, in which all joint motions are initiated together, thereby reducing the total motion time to the time taken by the slowest joint, or the joint which has the longest to move (figure 9.10). The motion of  $P_e$  is smoother, but as each joint finishes, the motion of  $P_e$  becomes jerky.

As can be seen, in this kind of trajectory generation, it is not easy to know exactly what the motion of  $P_e$  will be between its specified initial and final locations.

- Type 2: **Coordinated or Synchronous Trajectories**, in which all the individual joint motions are arranged to both start and finish together (figure 9.11). This produces much smoother  $P_e$  motion, and is also energy efficient, since the amount of acceleration and deacceleration is kept to a minimum, thus keeping to a minimum the amount of work needed to complete the arm motion.

<sup>1</sup>non-smooth.

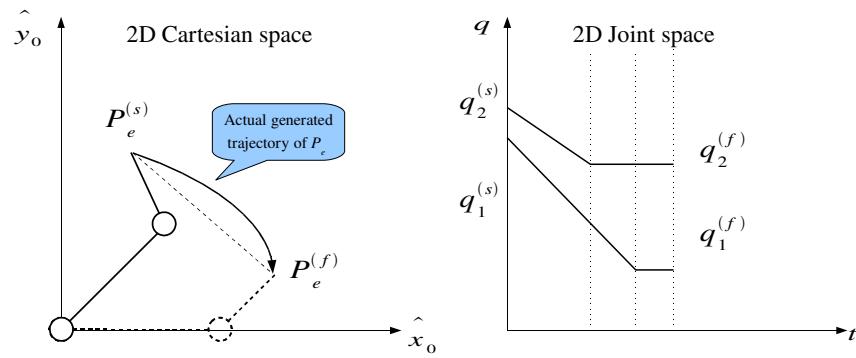


Figure 9.10: Point to point trajectories and simultaneous joint motion.

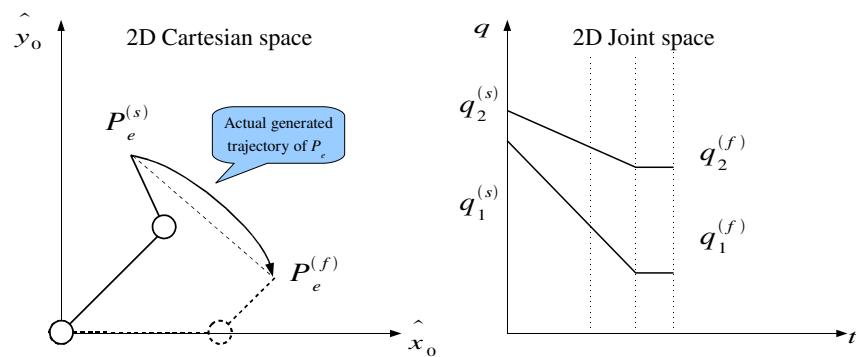


Figure 9.11: Coordinated or synchronous trajectories.

To achieve this type of coordinated synchronous trajectory, the kinematic controller has to first analyse the motion of all the joints needed to produce the motion of  $P_e$  to find which joint which needs the most time. All other joints are then programmed to take the same amount of time to complete their particular motions.

The total motion time is therefore, again, the time taken for the slowest joint or the joint which has the longest trajectory.

As for type one trajectories, although the resulting motion of  $P_e$  is smoother, it is once again difficult to know exactly what trajectory it will follow between the specified start and finish locations.

- Type 3: **Continuous Trajectories**, in which  $P_e$  must follow some specified trajectory in Cartesian space, rather than just move from the specified start location to the specified finish location (figure 9.12).

In this case the joint space trajectories, the motions of each joint, are typically quite complicated and will, in general, involve changes in the rate of movement and changes in the direction of movement.

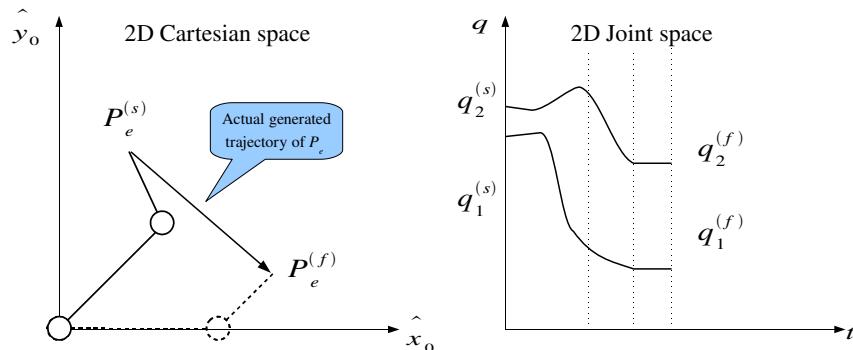


Figure 9.12: Continuous trajectories.

This is the type of trajectory generation we have seen described and illustrated in the previous section which introduced the six steps of the trajectory generation process.

### 9.3 Trajectory interpolation in joint-space

Given a set of sampled position vectors in the Joint Space (figure 9.4), that is, a series of points in Joint Space which a given joint has to pass through at a specific moment in time.

The aim is to obtain, by means of an interpolator, the intermediate reference values of which the reference signal is composed (figure 9.13).

The best interpolator would be the one that contains all sampled points using only one continuous function. That is, the ideal solution would be an interpolator with as

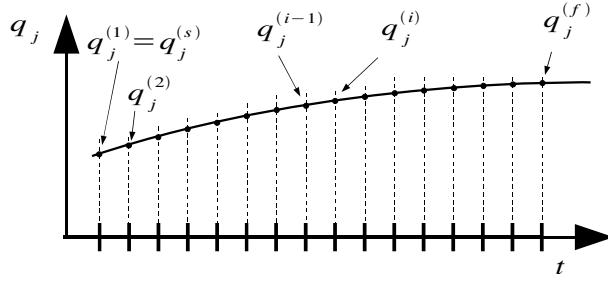


Figure 9.13: Time sampling in the Joint Space.

many parameters as samples. This option would lead to a very complicated (and complex) interpolator that would demand a lot of resources during its calculation.

In robotics, a low degree polynomial (with few coefficients, from now on referred to as parameters) is preferred. This will require the definition of intervals and the use of one interpolator for each interval.

Commonly used interpolators are:

- Linear Interpolators.
- Cubic Interpolators (Spline interpolation).
- Trapezoidal Interpolators (a linear interpolator between two intervals with parabolic interpolators)

### 9.3.1 Linear interpolation

Linear interpolation<sup>2</sup> consists of connecting the starting point and the ending point with a straight line. Thus, this interpolator only requires two parameters. Figure 9.14 depicts this (position) interpolator and its two derivatives (speed and acceleration). It is quick and easy, but it is not very precise.

This interpolator uses a linear function for each interval that is defined by two adjacent data points (sampled position vectors). Thus, the trajectory is defined as a series of straight lines that connects two consecutive samples. Notice that this interpolator leads to constant speeds for each section/interval.

Its equation is the following:

$$q_j(t) = \frac{(q_j^{(i)} - q_j^{(i-1)})}{T} (t - t^{(i-1)}) + q_j^{(i-1)} \quad (9.6)$$

---

<sup>2</sup>sometimes known as lerp

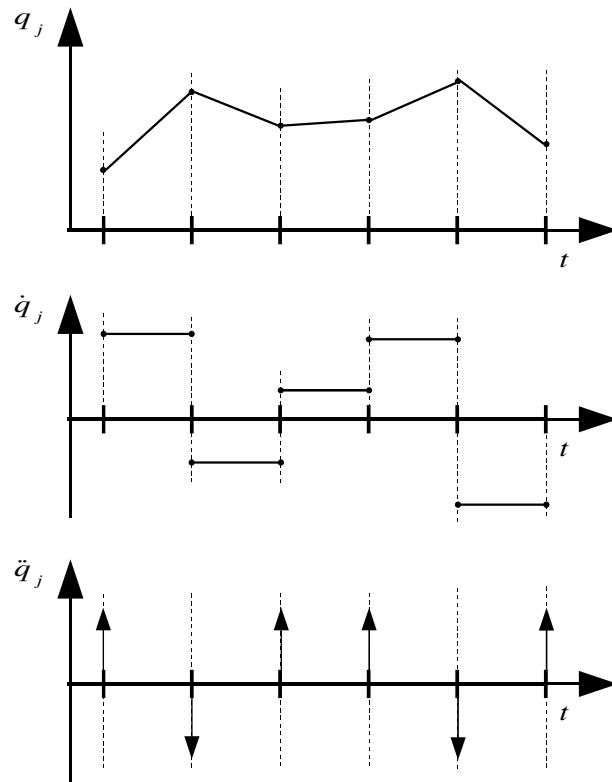


Figure 9.14: Linear interpolator.

where  $t^{(i-1)} < t < t^{(i)}$  and  $T = t^{(i)} - t^{(i-1)}$ .

The choice of  $t^{(i)}$ , that is, the travelling time for each  $q^{(i)}$ , can be made according to one of the following criteria:

- The concerned joint must arrive at its goal as fast as possible  $\rightarrow \dot{q} = cte = \dot{q}_{max}$ ,
- Tuning all joint speeds in order to synchronise the motion of all DoF.
- Customer specifications.

The main characteristic of this interpolator is its continuity even at the via-points (data points or samples), but this is not the case for its derivatives:

- velocity, first derivative  $\rightarrow$  there are step discontinuities (sudden speed changes),
- acceleration, second derivative  $\rightarrow$  there are impulses (step derivatives), that is to say, acceleration peaks with infinite amplitude and this is not reasonable.

### 9.3.2 Cubic interpolation (spline interpolation)

Instead of using a one degree polynomial (straight line, linear interpolator), the spline interpolation consists in using a low-degree polynomial. The commonly used degree is

three, so it is a cubic polynomial (interpolator) with 4 parameters (figure 9.15).

As happened with linear interpolators, a cubic interpolator is defined for each interval that is defined by two adjacent sampled points. The resulting curve is called spline.

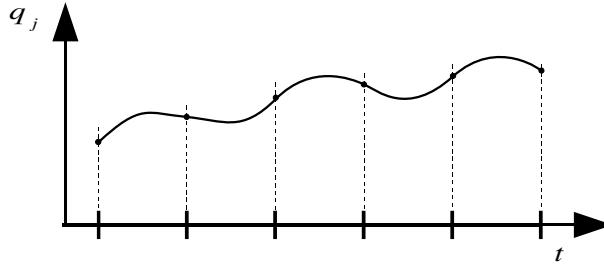


Figure 9.15: Cubic interpolator.

Due to the fact that the chosen interpolator is a cubic polynomial, the resulting spline is twice continuously differentiable, that is, its first and second derivatives (velocity and acceleration) are continuous functions.

The mathematical expression for a cubic interpolator is:

$$q_j(t) = a + b(t - t^{(i-1)}) + c(t - t^{(i-1)})^2 + d(t - t^{(i-1)})^3 \quad (9.7)$$

In order to guarantee a twice continuously differentiable spline, the following boundary conditions should be set to obtain the 4 parameters for each cubic interpolator:

- 2 position boundaries → the spline has to reach every sampled position vector (via-point), this means each interpolator has to pass through the corresponding via-points.
- 2 velocity boundaries → the time derivative of the spline curve must be continuous. This means that both one-side limits <sup>3</sup> (from the negative direction and from the positive direction) of interpolated velocity at each point must be equal. More specifically, this is also true for sampled points (via points) that are the boundaries at which two adjacent intervals (interpolators) are linked. That is, the interpolated values must be equal when one is obtained using the “interpolator on the left” (one-side limit from the negative direction) and when the other is obtained using the “interpolator on the right” (one-side limit from the positive direction) at each via point <sup>4</sup>.

Taking into account the former criteria, it is easily proven that the mathematical expression for the cubic interpolator polynomial is:

<sup>3</sup>limites laterales  $\lim_{x \rightarrow x_1^-} y \quad \lim_{x \rightarrow x_1^+}$

<sup>4</sup>Notice that, in the case of the first and last samples, the joint-speed must be equal to zero to guarantee that the robot is at rest before and after the motion covering all intervals

$$\begin{aligned}
a &= q_j^{(i-1)} \\
b &= \dot{q}_j^{(i-1)} \\
c &= \frac{3}{T^2} \left( q_j^{(i)} - q_j^{(i-1)} \right) - \frac{2}{T} \dot{q}_j^{(i-1)} - \frac{3}{T} \ddot{q}_j^{(i)} \\
d &= -\frac{2}{T^3} \left( q_j^{(i)} - q_j^{(i-1)} \right) + \frac{1}{T^2} \left( \dot{q}_j^{(i)} + \dot{q}_j^{(i-1)} \right) \\
T &= t^{(i)} - t^{(i-1)}
\end{aligned} \tag{9.8}$$

An additional criterion for via-point velocities is given by Craig in “Introduction to Robotics”:

$$\dot{q}_j^{(i)} = \begin{cases} 0 \rightarrow \text{if } \operatorname{sgn} \left( q_j^{(i)} - q_j^{(i-1)} \right) \neq \left( q_j^{(i+1)} - q_j^{(i)} \right) \\ \frac{1}{2} \left[ \frac{\left( q_j^{(i)} - q_j^{(i-1)} \right)}{T} + \frac{\left( q_j^{(i+1)} - q_j^{(i)} \right)}{T} \right] \rightarrow \text{in other case} \end{cases} \tag{9.9}$$

But this criterion allows acceleration discontinuities.

In any case, do not forget to set to zero the velocities at the beginning and in the end of the motion to guarantee the robot stops.

We also have the option of defining velocities in Cartesian Space and then obtaining the corresponding joint speeds, using the Jacobian matrix. In this case, we have to make sure that the robot is not crossing any singular configuration during motion.

### 9.3.3 Trapezoidal interpolator

The trapezoidal interpolator consists of a linear interpolator with parabolic connections between two consecutive interpolators to overcome discontinuities. Thus, the strategy is similar to splines but now we use linear interpolator instead of cubic polynomials.

As a consequence of this, in each interval, this interpolator has three sections (see figure 9.16):

- One parabolic link with constant acceleration from  $t^{(i-1)}$  to  $t^{(i-1)} + \tau$ , where  $\tau$  is the time length of this first section.
- A linear interpolator with constant speed from  $t^{(i-1)} + \tau$  to  $t^{(i)} - \tau$ .
- One parabolic link with constant deceleration from  $t^{(i)} - \tau$  to  $t^{(i)}$ .

Where  $t^{(i-1)}$  and  $t^{(i)}$  are the time instants at which the  $i^{th}$  interval starts and ends.

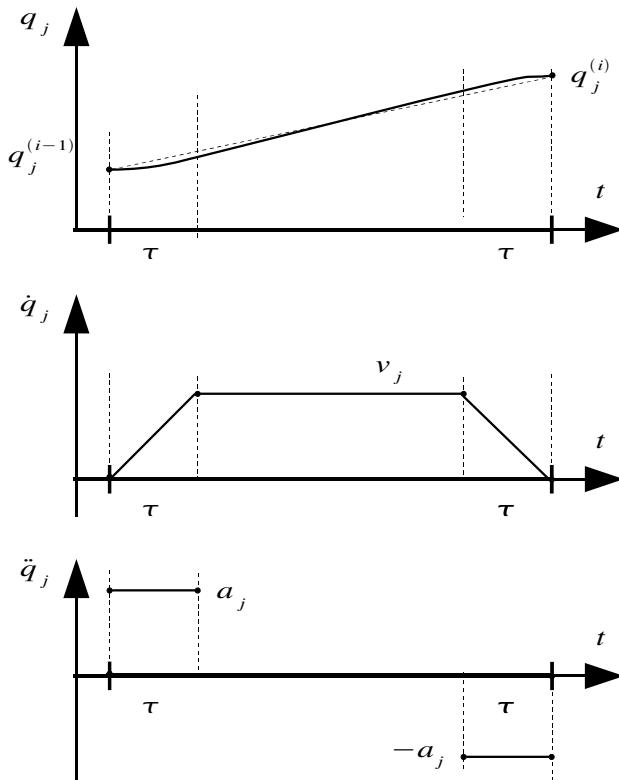


Figure 9.16: Trapezoidal interpolator.

It is called “trapezoidal” due to the shape of its speed profile (time derivative of the interpolator presented in the previous paragraph).

The mathematical expression for this type of interpolator is:

$$q_j(t) = \begin{cases} q_j^{(i-1)} + s_j^{(i)} \frac{a_j}{\tau} t^2 \rightarrow t^{(i-1)} \leq t^{(i-1)} + \tau \\ q_j^{(i-1)} - s_j^{(i)} \frac{v_j^2}{2a_j} + s_j^{(i)} v_j t \rightarrow t^{(i-1)} + \tau \leq t^{(i)} - \tau \\ q_j^{(i)} + s_j^{(i)} \left( -\frac{a_j T^2}{2} + a_j T t - \frac{a_j}{2} t^2 \right) \rightarrow t^{(i)} - \tau \leq t^{(i)} \end{cases} \quad (9.10)$$

Where:

$$\begin{aligned}\tau &= \frac{v_j}{a_j} \\ T &= s_j^{(i)} \frac{\left(q_j^{(i)} - q_j^{(i-1)}\right)}{V_j} + \tau \\ v_j &= \text{maximum speed} \\ a_j &= \text{maximum acceleration} \\ s &= \text{sgn}\left(q_j^{(i)} - q_j^{(i-1)}\right)\end{aligned}\tag{9.11}$$

Notice that the velocity at via points are zero (null) for the proposed interpolator. This feature is only useful when the complete trajectory is defined by only two samples (the starting and end positions). When the trajectory is composed of a series of more than two points, every two adjacent interpolators must be connected using the corresponding parabolic sections. These links have to be done following similar criteria as used with spline curves, that is, the robot acceleration must be continuous and the robot must not stop at an intermediate motion stage (figure 9.17).

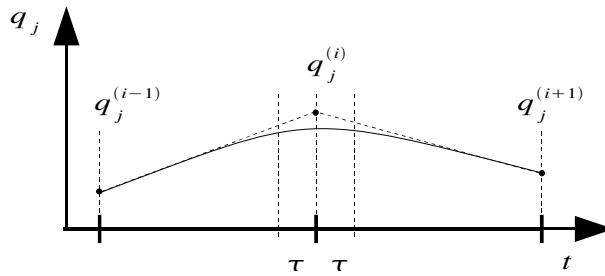


Figure 9.17: Trapezoidal interpolator: parabolic link between two adjacent interpolators.

Notice that the closer the parabola is from a via point, the greater the accuracy is. However, due to this interpolator behaviour, as the robot approaches to a via point, it has to deaccelerate and when it is getting far away, it has to accelerate. On the other hand, if robot maintain its trajectory far away from any via point, its velocity will be fairly constant.

Thus, the more accurate the trajectory is, the more jerky is due to the deacceleration-acceleration cycles the robot has to perform to slow down at via point vicinities.

## Chapter 10

# Control of robot manipulators

---

### 10.1 Introduction

Control when the consequence or result of an action is always be the same, despite any perturbations or disturbances that affect the action. Therefore, one of the concepts of control we will use in this course can be characterised as follows: Control is a process that causes a system variable to remain at or near to a specified value, which is called its reference value or set point value.

When more than one system variable needs to be kept at or near to a reference value, there are as many reference values as there are system variables to be controlled. We can thus define a controller or control system as:

an information-processing device which manipulates the flow of energy, material, or other resources in an associated physical system in such as a way as to make the overall system function in some specified manner in the face of arbitrary changes imposed on the system by its environment, and in the face of arbitrary changes in the physical system itself.

Prof. Alistair MacFarlane, 1976.

We will call this kind of control **Basic Control** in this course. It will be the first of different kinds of control we will see and use.

In this part of the course we will consider the basic level of control of a robot manipulator arm. For matters of clarity, we will consider a particular kind of robot, but all that we will see in this applies to all other kinds of industrial robots.

Suppose we have a 6 DoF robot manipulator arm with an Angular Robot geometry (i.e., six one-DoF rotational joints), which has an electric DC motors plus reduction and transmission mechanisms at each joint, together with an angular position sensor.

Then, to position and orient the coordinate frame fixed to the end-point,  $P_e$ , of the robot, we need to appropriately position each of the 6 rotational joints of the robot arm. To do this, we will need to explicitly control the position of each joint individually.

In robotics, there are two types of traditional controllers: one for position control and the second one to control the force that is applied by the end-effector. This will yield to controllers that have two feedback loops (figure 10.1).

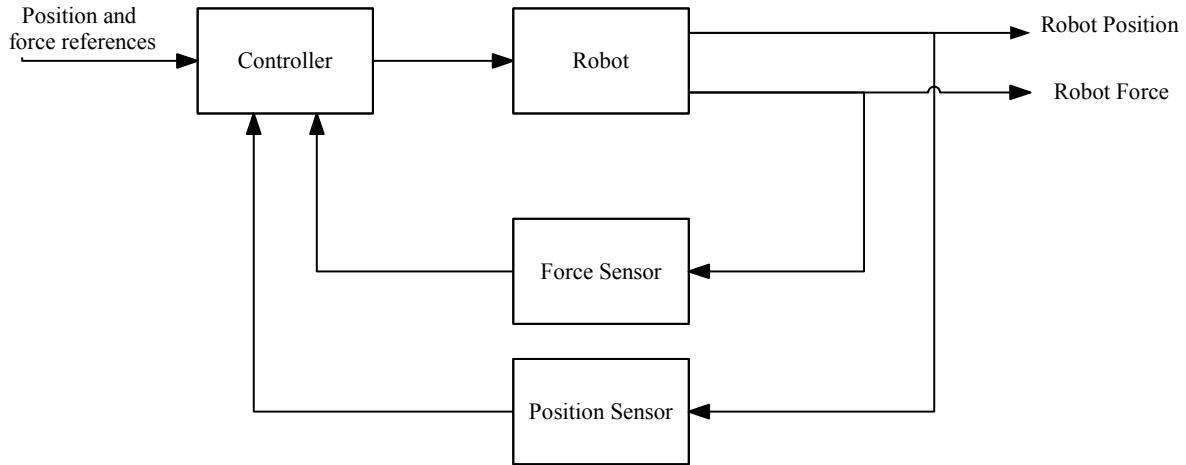


Figure 10.1: Position and Force control loops.

The first part of this chapter will be devoted to position control loops and the last will briefly describe the most important force control strategies.

### 10.1.1 Analogy between Mechanical and Electrical Physics

It is widely-known that the differential equations that predicts the dynamic behaviour of systems and the current flow in electrical circuits are formally equal. For example, the differential equation of the motion of a mass connected to a spring and a damper are the same as the equation of a simple electrical circuit composed of a voltage source connected to a resistor, inductance and capacitor in series (see figure 10.2).

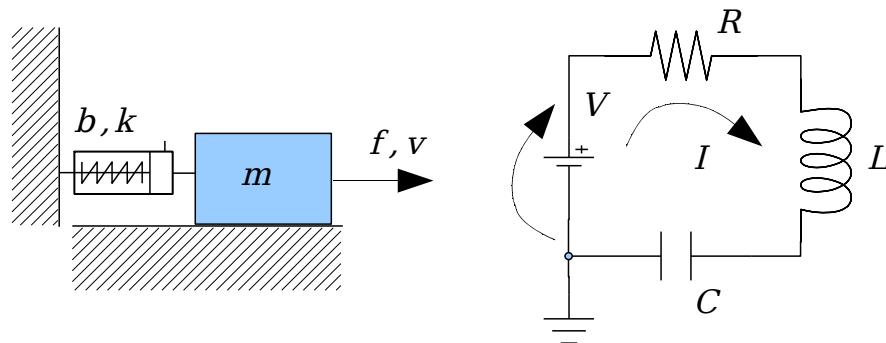


Figure 10.2: Mechanical-electrical analogy.

In the case of a mechanical system composed of a spring, damper and mass, the

differential equation is:

$$f = m\ddot{v} + b\dot{v} + k \int v dt \quad (10.1)$$

While for the resistor, inductance and capacitor circuit, the differential equation is:

$$V = L\dot{I} + RI + \frac{1}{C} \int I dt \quad (10.2)$$

When comparing these last equations, we have a formal correspondence between voltage and force, current and speed, etc. The complete reference to that correspondence is summarised in table 10.1.

Electricity	Mechanics
Voltage	Force/torque
Current	Speed
Resistor	Damper
Inductance	Mass
Capacitor inverse	Spring

Table 10.1: Summary of the analogy between mechanics and electricity.

Once the analogy is defined, we can exchange the nomenclature between these two worlds. For example, as the quotient between voltage and current is an (electrical) impedance, the quotient between force/torque and speed is known as mechanical impedance. In the same manner, the inverse of a mechanical/electrical impedance is a mechanical/electrical admittance.

This analogy will be very useful in the classification of the different kind of force/torque control strategies, as will be seen in the following section.

### 10.1.2 Types of robotic control loops

#### 10.1.2.1 Motion Control

This type of control forces the robot to follow a predefined position trajectory while the internal interaction forces within the robot's parts (mainly friction) and the externals (interaction with the environment) are compensated. Trajectory error compensation is achieved by high gains in the feedback paths. High forces are a direct consequence of these high gains, which lead to the saturation of actuators.

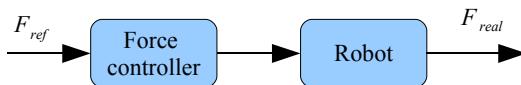
This kind of control is quite easy to implement and only needs position sensors (cheaper than force sensors). However, this controller is only suitable when the task to be performed does not require physical contact between the robot and its environment (ex. spraying).

### 10.1.2.2 Force Control

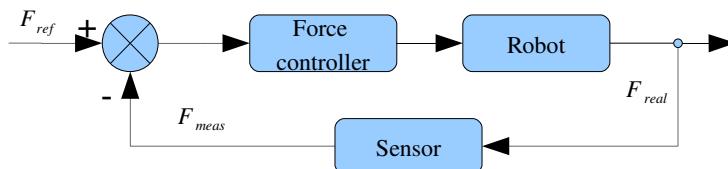
Force control is a new paradigm that is being introduced into industrial robot control loops when robots have to perform tasks that require physical interaction with their environment. Notice that this is the case of the majority of automated processes in industry.

On the other hand, this control requires the use of expensive force sensors, which are more complex than position controllers. Moreover some old robots are incompatible and this kind of control cannot be implemented.

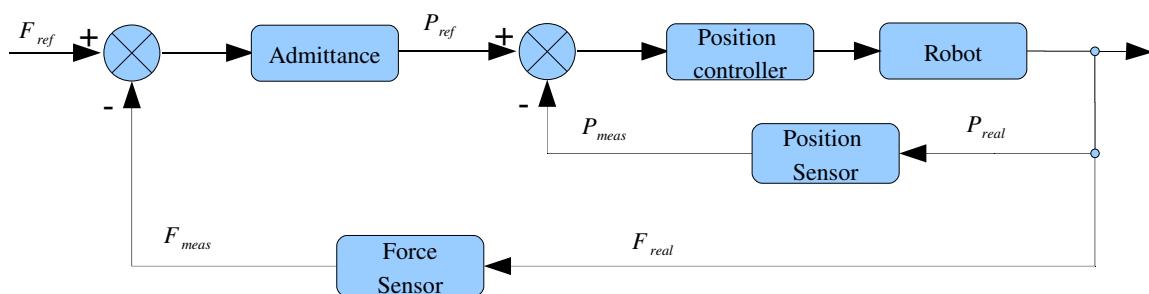
- Explicit force/torque control
  - Open loop: There is no force/torque feedback, so it is sensitive to disturbances.



- Closed loop: measured force/torque is compared to the reference. A force/torque sensor, which is normally expensive, is needed.



- Explicit force control: It is based on position measurement (**admittance control**). The computed error is the same as that of explicit control. Now, however, the error is transformed to a position reference by an admittance. This reference is the input for a position controller.

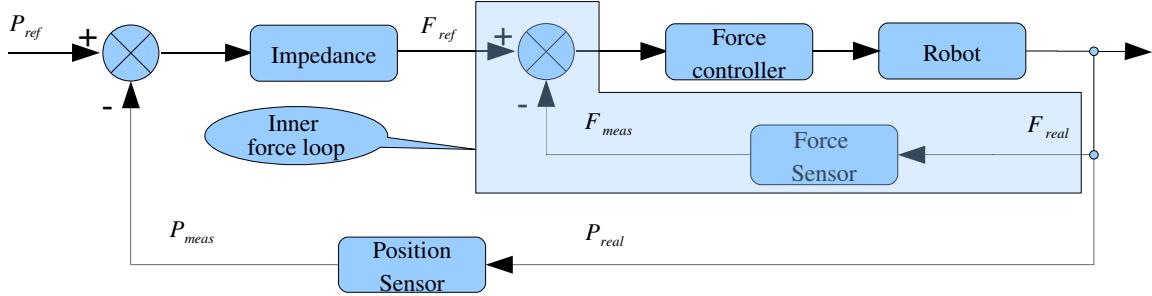


- Implicit force control (**Impedance control**)

Impedance Control is a strategy that controls the dynamic relation between the manipulator and the environment. The force exerted on the environment by the manipulator is dependent on its position and impedance. Thus, there is no force feedback and it may also be considered as a position control with low stiffness.

Impedance consists of two components, that which is physically intrinsic to the manipulator and that which is given to the manipulator by active control. It is the goal of Impedance Control to mask the intrinsic properties of the arm and replace them with the target impedance.

The following schematic illustrates the Impedance Control with Force Feedback.<sup>1</sup>



- Hybrid Control. This controller detects the directions in which the robot is having physical interaction with the environment. In these directions the hybrid controller switches to force/torque control and in the remaining direction, the controller acts as a normal position controller.

**Impedance Control vs. Admittance Control** Impedance Control is used when a force/torque direct control is desired without a compulsory need for a force sensor. Admittance control is implemented when the robot has an inner position controller which cannot be modified.

Impedance controllers with force feedback can compensate the robot dynamic. Impedance control without force feedback is mainly used in haptic devices with high backdrivability.

Admittance control is advised for devices with high non-linearities, high inertia or non-backdrivability.

### 10.1.3 Sources of Instability in Robot Control Loops

It is widely known that controller accuracy improves when gain is increased. However, gains cannot be as large as we want because, beyond a certain threshold instabilities can appear due to:

- sampling time in digital implementations,
- Coulomb friction,
- actuator wind-up and limited band-width,

---

<sup>1</sup>In this case, the topology of the impedance controller is equal to the topology of a position controller, but now the gain is tuned to have a compliant motion despite the fact that the trajectory tracking will have a high error rate.

- noise in sensors,
- flexible joints and links,
- sensor dynamics,
- environment dynamics,
- etc.

## 10.2 Linear Position Control of a Single Joint

The robot control algorithms usually have cope with multiple inputs-outputs of non-linear systems. This fact is because robots have more than one DoF which are usually coupled. This means that the classical control theories based on Laplace-transform cease to be valid, resulting in a complex controller design. That is why numerous efforts focused on simplifying the analysis. Let's see some of them.

### 10.2.1 Linear Dynamic Model of 1-DoF joint

Let's say a motor **m** that is connected to a reduction mechanism with a ratio  $\eta = \frac{r_L}{r_m} \gg 1$ , as it is in figure 10.3.

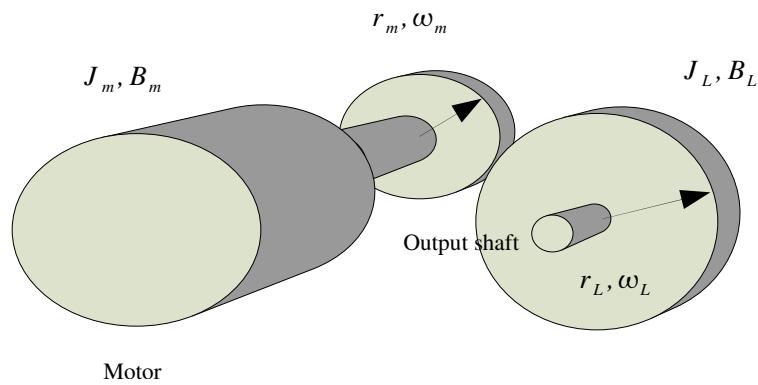


Figure 10.3: Position and Force control loops.

Where:

- $J_m$  is the motor inertia.
- $B_m$  is the motor damping.
- $J_L$  is the payload inertia that is attached to the output of the reductor.
- $B_L$  is the payload damping that is attached to the output of the reductor.
- $\omega_m$  is the angular motor speed.

- $\omega_L$  is the angular speed at the reductor output.
- $r_m$  is the ratio of the input gear.
- $r_L$  is the ratio of the output gear.

### 10.2.1.1 Resisting torques

According to figure 10.3, it is cleared depicted that there are only two resisting torques:  $\tau_m$ , the resisting torque due to the rotatory movement of the motor; and  $\tau_L$ , which is the resisting torque due to the link inertia and other payloads attached to that link.

The payload resisting torque can be computed by means of torque equilibrium at the output of the reduction mechanism:

$$(\tau_L)_L = J_L \dot{\omega}_L + B_L \omega_L \quad (10.3)$$

where:  $(\tau_L)_L$  is the payload resisting torque at the output shaft, measured at the output of the reductor.

The motor resisting torque can be found in a similar way. The equation is as follows:

$$(\tau_m)_m = J_m \dot{\omega}_m + B_m \omega_m \quad (10.4)$$

Now, we need the expressions that relate the input speed/torque with the output speed/torque in the reduction mechanism. A reductor can be envisioned as a speed/torque transformer where ideally, without any mechanical losses, the input power is equaled to the output power:

$$P_L = P_m \quad (10.5)$$

Where  $P_L$  is the transmitted power to the load and  $P_m$  is the power exerted by the motor on the reductor input.

By substituting the last equation the expressions for power yields:

$$(\tau_L)_L \omega_L = (\tau_L)_m \omega_m \quad (10.6)$$

Where  $(\tau_L)_m$  is the load-resisting torque transmited to the motor side.

Rearranging terms of 10.6 to have torque on one side and angular speed on the other side yields:

$$\frac{(\tau_L)_L}{(\tau_L)_m} = \frac{\omega_m}{\omega_L} = \eta \quad (10.7)$$

and

$$(\tau_L)_L = \eta(\tau_L)_m \quad (10.8)$$

$$\omega_m = \eta\omega_L \quad (10.9)$$

Substituting  $(\tau_L)_L$  and  $\omega_L$  in equation 10.3, using the equations 10.8 and 10.9 yields:

$$(\tau_L)_m = \frac{1}{\eta} \left( J_L \frac{\dot{\omega}_m}{\eta} + B_L \frac{\omega_m}{\eta} \right) \quad (10.10)$$

Now we have to notice that the total commanded motor torque  $(\tau_{com})$  required is divided between moving the rotor shaft and moving the payload:

$$\tau_{com} = \tau_m + (\tau_L)_m \quad (10.11)$$

Finally, in equation 10.11 the expressions for  $(\tau_m)_m$  and  $(\tau_L)_m$  (equations 10.4 and 10.10) should be substituted:

$$\tau_{com} = J_m \dot{\omega}_m + B_m \omega_m + \frac{1}{\eta} \left( J_L \frac{\dot{\omega}_m}{\eta} + B_L \frac{\omega_m}{\eta} \right) \quad (10.12)$$

To simplify the final expression, it is convenient to combine the common factors  $\dot{\omega}_m$  and  $\omega_m$ :

$$\tau_{com} = \left( J_m + \frac{J_L}{\eta^2} \right) \dot{\omega}_m + \left( B_m + \frac{B_L}{\eta} \right) \omega_m \quad (10.13)$$

Taking into account the hypothesis of high reduction ( $\eta = \frac{r_L}{r_m} \gg 1$ ), a very common fact in industrial robotics, yields:

$$\tau_{com} = \left( J_m + \frac{J_L}{\eta^2} \right) \dot{\omega}_m + \left( B_m + \frac{B_L}{\eta^2} \right) \simeq J_m \dot{\omega}_m + B_m \omega_m \quad (10.14)$$

In other words, the influence of the payload attached to the output shaft is negligible compared to the motor inertia. This is due to the scale factor of  $\eta^2$ .

### 10.2.2 Proportional error control of a single joint

After confirming that the high reduction hypothesis holds true, in a  $n - DoF$  robot, we can implement  $n$  decouple linear SISO <sup>(2)</sup> position controllers instead of one non-linear MIMO <sup>(3)</sup> position controller of  $n - DOF$ .

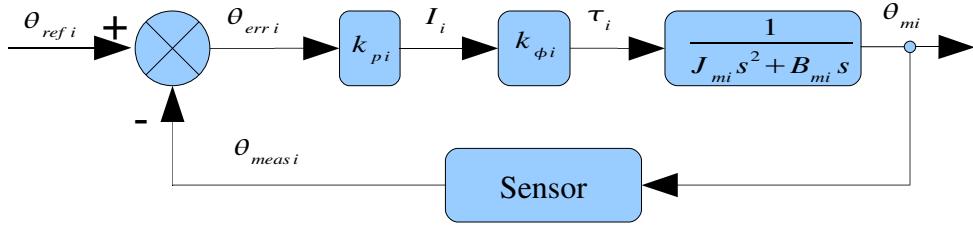


Figure 10.4: 1-DoF Proportional Error position control.

If we use Proportional Error (PE) control to control the position of each joint, we will have a negative feedback control loop of the kind shown in figure 10.4: one for each joint system.

The equation of  $i - th$  (rotational) joint motion is given by:

$$\tau_i = J_{m_i} \dot{\omega}_{m_i} + B_{m_i} \omega_{m_i} \quad (10.15)$$

which can also be expressed as follows:

$$\tau_i = J_{m_i} \ddot{\theta}_{m_i} + B_{m_i} \dot{\theta}_{m_i} \quad (10.16)$$

where:  $\tau_i$  is the torque applied to the  $i - th$  joint by the DC motor via reduction and transmission mechanisms;  $\theta_{m_i}$  is the (angular) position of the  $i - th$  actuator at time  $t$ ;  $J_{m_i}$  is the (rotational) inertia of the joint, due to the masses of the motor and reduction and transmission mechanisms for the joint, plus the masses<sup>4</sup> of the link connected by the joint, and all other subsequent links in the chain;  $B_{m_i}$  is the total friction constant coefficient of the joint, due to the motor and reduction and transmission mechanisms.

For a reasonable approximation (and to keep the model linear), the torque generated by a DC motor, together with its reduction and transmission mechanisms,  $\tau_i$ , can be taken to be directly proportional to the current,  $I_i$ , to the motor. This leave us with:

$$\tau_i = k_{\phi_i} I_i \quad (10.17)$$

where:  $k_{\phi_i}$  is the constant of proportionality; and  $I_i$  is the current to the motor. Therefore, equations 10.16 and 10.17 spawn the following result:

$$k_{\phi_i} I_i = J_{m_i} \ddot{\theta}_{m_i} + B_{m_i} \dot{\theta}_{m_i} \quad (10.18)$$

For proportional error control, the torque applied by the motor (via its reduction and transmission mechanisms) must be proportional to the position error of the joint,

$$\theta_{err_i} = \theta_{ref_i} - \theta_{meas_i} \quad (10.19)$$

From 10.18 we can see that we can make  $\tau_i$  proportional to the position error,  $\theta_{err_i}$ , by making the motor current,  $I_i$ , proportional to the  $\theta_{err_i}$ . Therefore, the following has resulted from equations 10.18 and 10.19:

<sup>4</sup>negligible if the hypothesis of high reduction

$$k_{\phi_i} K_{p_i} (\theta_{ref_i} - \theta_{meas_i}) = J_{m_i} \ddot{\theta}_{m_i} + B_{m_i} \dot{\theta}_{m_i} \quad (10.20)$$

where:  $k_{p_i}$  is the constant of proportionality between  $I_i$  and  $\theta_{err_i}$ .

We can combine the constants  $k_{\phi_i}$  and  $k_{p_i}$  to obtain  $kpe_i$ , and without loss of generality, we can substitute the expression  $(\theta_{ref_i} - \theta_{meas_i})$  with  $\Delta\theta_i$  to clarify the equations. Then this results the following:

$$k_{p_i} \Delta\theta_i = J_{m_i} \ddot{\theta}_{m_i} + B_{m_i} \dot{\theta}_{m_i} \quad (10.21)$$

Or, more conventionally, as:

$$J_{m_i} \ddot{\theta}_{m_i} + B_{m_i} \dot{\theta}_{m_i} - k_{p_i} \Delta\theta_i = 0 \quad (10.22)$$

Equation 10.22 is a second-order lineal differential equation, of which the general form solution is given by:

$$\Delta\theta_i = e^{\left(-\frac{B_{m_i}}{2J_{m_i}}t\right)} \left[ C_{i_1} e^{\left(\frac{\Omega_i}{2}t\right)} + C_{i_2} e^{\left(-\frac{\Omega_i}{2}t\right)} \right] \quad (10.23)$$

where  $C_1$  and  $C_2$  are constants of integration that depend upon the initial conditions, and where

$$\Omega_i = \sqrt{\left(\frac{B_{m_i}}{J_{m_i}}\right)^2 - \left(\frac{4k_{p_i}}{J_{m_i}}\right)} \quad (10.24)$$

This kind of controller is often called a servomechanism, servo-controller, servo-system or servo-motor. It is a negative feedback control system which controls one of five kind of mechanical variables: position, speed, acceleration, force, or torque.

The term servo-system is also sometimes applied to a control system, of which the output is supposed to follow some varying input reference value.

The damping or dynamic response term in equation 10.23 guarantees that, with time,  $t$ , the value of  $\theta_{meas_i}$  goes to the value of  $\theta_{ref_i}$ , the reference position, as long as the friction term,  $B_{m_i}$ , is not zero, which is never the case for real systems.

From the expression for  $\Omega$ , in equation 10.24, we can see that (figure 10.5):

- If

$$\left(\frac{B_{m_i}^2}{4k_{p_i}}\right) > J_{m_i} \quad (10.25)$$

$\Omega_i$  is positive, and therefore, the system has more damping. This is sometimes called an overdamped system.

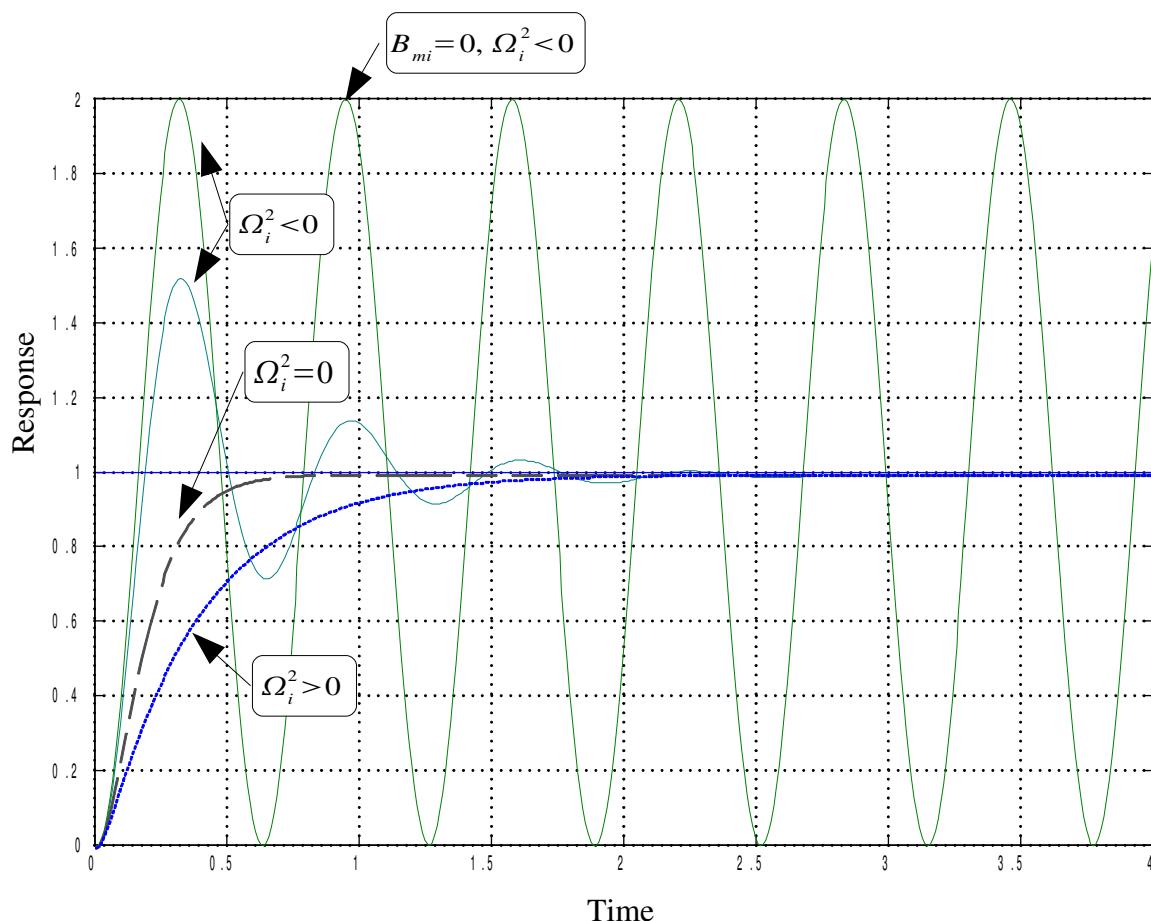


Figure 10.5: Step response types of systems:  $\omega_i^2 > 0 \rightarrow$  overdamped system,  $\omega_i^2 = 0 \rightarrow$  critically damped system, and  $\omega_i^2 < 0 \rightarrow$  underdamped/oscillatory system.

- If

$$\left( \frac{B_{m_i}^2}{4k_{p_i}} \right) = J_{m_i} \quad (10.26)$$

$\Omega_i$  is equal to zero, equation 10.23 becomes more symplified in,

$$\Delta\theta_i = C_{i_{12}} e^{\left(-\frac{B_{m_i}}{2J_{m_i}} t\right)} \quad (10.27)$$

In this case, the joint system is said to be critically damped, and it means that its response time,  $t_r$ , has reached its minimum value. (This may not, however, be the optimal response time for the system). The form of this response curve is shown below.

- If

$$\left( \frac{B_{m_i}^2}{4k_{p_i}} \right) < J_{m_i} \quad (10.28)$$

$\Omega_i^2$  is negative, therefore,  $\Omega_i$  is defined as a complex number,  $\Omega_i j$ , and in this case equation 10.23 takes the following form:

$$\Delta\theta_i = e^{\left(-\frac{B_{m_i}}{2J_{m_i}} t\right)} \left[ (C_{i_1} + C_{i_2}) \cos\left(\frac{\Omega_i}{2}t\right) - j(C_{i_1} - C_{i_2}) \sin\left(\frac{\Omega_i}{2}t\right) \right] \quad (10.29)$$

The joint system is said to be *under-damped*, and it will have oscillatory motion with  $\frac{\Omega_i}{2\pi}$ Hz frequency of oscillation. In other words, the system will have complex poles off the real axis in the  $s$ -plane of its root locus diagram. This means that the values of  $B_{m_i}$  and  $k_{p_i}$  can influence the stability of the joint system. If  $B_{m_i}$  is small and  $k_{p_i}$  is large, the joint system may be unstable.

### 10.2.3 The Steady State Error Problem

If we include a static force, such as gravity or a payload,  $g_i$ , in the equation of the single joint motion (equation 10.22), in our mathematical model of the controlled system, we will have a new version which will now have the form:

$$\tau_{m_i} = -k_{p_i}\Delta\theta_i = J_{m_i}\ddot{\theta}_{m_i} + B_{m_i}\dot{\theta}_{m_i} + g_i \quad (10.30)$$

where  $g_i$  is the payload/gravity that the  $i^{th}$  DoF is supporting. As we can see, it is a nonlinear second order differential equation! <sup>5</sup> It is not a homogeneous equation, since

---

<sup>5</sup>the gravity terms normally depend on trigonometric functions of  $q_i$ .

the total torque,  $\tau_{m_i} - g_i$  does not depend only on time,  $t$ , but the angular position of joint. (How do we convert this model into a linear model, which is what we need?)

We can also see from equation 10.30 that when the control torque  $\tau_{m_i} = k_{p_i}\Delta\theta_i$  is zero, which it will be if there is no position error, there will still be torque acting on the joint, produced by  $g_i$ . This means that joint cannot sustain a zero position error. Instead, it will move away from its reference position enough to produce an error sufficient to induce a control torque high enough to counteract  $g_i$ .

In other words, the joint  $i^{th}$  will only be able to sustain a position at which  $\tau_{m_i} = g_i$ . This means that there will be some amount of position error, which results in a non-zero  $\tau_{m_i}$ .

This error is called the **steady state error** or **static error**. How big it is depends on the value of  $g_i$ , which, as we have seen, depends on the actual configuration of the robot, whether the robot is carrying an object or not, and, if so, what mass the object has.

We can solve this problem by adding some **lag compensation** to the basic **PE controller**.

This we do by adding an **integral term** to the basic proportional error law, which integrates the position error of the joint over time. We therefore have that:

$$\tau_{m_i} = k_{p_i}(\theta_{ref_i} - \theta_{meas_i}) + k_{I_i} \int_0^t (\theta_{ref_i} - \theta_{meas_i}) dt \quad (10.31)$$

where  $k_{I_i}$  is the gain of the integral term.

This results in a new controller which we call a **PI controller**, for **proportional-integral controller**. By varying the value of the integration constant,  $k_{I_i}$ , we can arrange for the joint system to converge on its reference value more or less quickly.

However, if we make  $k_{I_i}$  too large, oscillations in the behaviour of the joint will occur, also known as **overshoot**. This is another problem that a PE controller can suffer from, and, as we can see, PI controllers can too.

Before looking at how we can solve this overshoot problem, it is worth stopping and asking ourselves if there is any other way in which we might solve the steady state problem associated with using PE controllers. In other words, is adding lag compensation, using an error integration term, the only way to solve this problem?

As is most often the case in engineered systems, the answer is no. We can also avoid this problem, or at least reduce it to just one joint, by adopting a different geometry for the robot arm.

In particular, we could use a SCARA type geometry, in which the first two joints have vertical rotational axes. This means that these joints do not suffer the effects of gravity on either the masses of the links they connect, the configuration of the rest of the arm, or whether the robot is carrying an object (or a tool) or not.

In a SCARA-type geometry, the third (vertical prismatic) joint would need a PI controller to deal with the effects of the mass of the third element, wrist, and any object. But in this case the direction of movement is always vertical, which means the same value of  $k_{I_i}$  is always good. This is not the case in other geometries, and in particular Angular Robot geometries, in which the ideal value (or optimal value) of  $k_{I_i}$  varies with different configurations.

We can also try to reduce the static error by adding **Feedforward Compensation** and **Feedback Compensation** control strategies. These strategies will be explained in following sections.

#### 10.2.4 The Overshoot Problem

As we have seen, if the friction in the joint system is low and the inertia of the joint system is high, which are often both the case, especially for the first one or two degrees of freedom of a robot arm, then the torque needed to move the joint rapidly to its reference position may be too large to move it. When that happens, the robot cannot move directly to the reference position without presenting **overshoot**.

To accelerate a high inertia joint, we need large control torques, but if there is little friction, there will be little to stop the movement of such a joint system once it has been accelerated.

The joint will thus tend to oscillate about the reference value until it finally settles on this value.

We can also see that the integral term of a PI controller, is no help here, since it effectively introduces extra inertia into the joint system.

Solving the steady-state problem using a PI controller can thus make the overshoot problem worse.

To solve the overshoot problem we need to introduce some **lead compensation** into the joint system and its control. This we can do by adding a differential term to the control law, which depends on the position error.

This results in:

$$\tau_{m_i} = k_{p_i} (\theta_{ref_i} - \theta_{meas_i}) + k_{I_i} \int_0^t (\theta_{ref_i} - \theta_{meas_i}) dt - k_{d_i} \dot{\theta}_{meas_i} \quad (10.32)$$

where  $k_{d_i}$  is the gain of the differential term.

This gives us a new controller, which is called a **PID controller**, which stands for proportional, integral, differential controller.

The effect of the differential term in the controller is to add (artificial) friction to the joint system, which, like the natural friction term,  $B\dot{\theta}_i$ , varies linearly with joint's speed.

The block diagram of a PID-controlled joint system thus takes the form shown in figure 10.6.

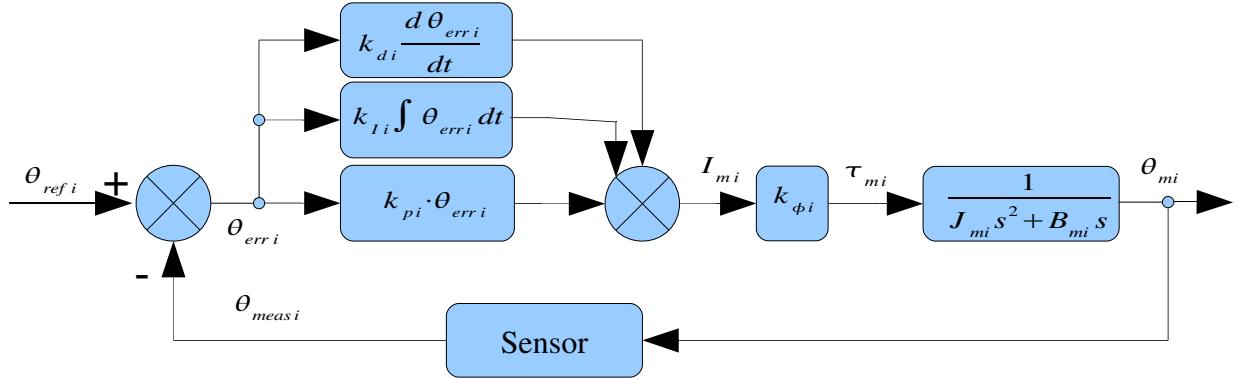


Figure 10.6: 1-DoF PID position control.

### 10.2.5 Speed Sensing and Control of a Joint System

A serious problem that can arise in the implementation of a PD or PID controller, in other words, any controller with a derivative component, is that it requires the differentiation of the signal from the joint position sensor.

If there is any noise in the sensor signal, which there will normally be, the differentiation will amplify this noise.

This can result in not just inaccurate speed measurements, but incorrect speed values as well, which sometimes may have even the opposite sign to the correct value.

In this case, the derivative component of the controller acts to accelerate joint movement, rather than reducing it.

The practical solution to this problem is to measure the speed of the joint using a speed sensor, rather than the joint position. We can then obtain a position value by integrating the speed measurement.

Integration of a signal with noise has the effect of reducing the relative proportion of the noise in the integrated signal. Integration thus effectively suppresses any noise in the signal.

This makes it a much more accurate and reliable way of measuring the position of a joint, and now that cheap rotation speed sensors are available, this is the most commonly used sensor in robot arm controllers.

If the basic sensor in the joint system is a speed sensor, we can take advantage of this to introduce a further improvement in the control of the joint system, by introducing a **speed control loop** inside the basic position control loop (figure 10.7).

In this control system, the speed reference value input to the controller is now taken

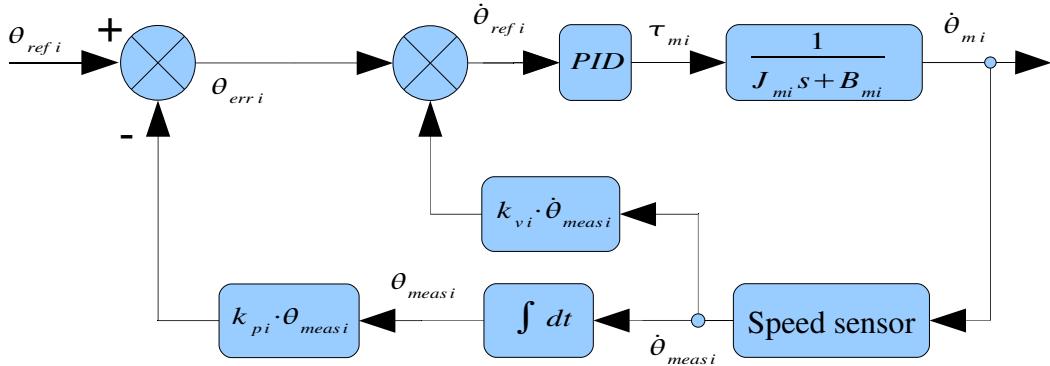


Figure 10.7: 1-DoF PID position control.

to be defined by the actual position error value.

In other words, when position error is large, the reference value for the controller has a high speed value. Conversely, when the position error is low, the reference value input to the controller has a low speed value. When there is no position error, the speed reference value is, of course, also zero.

By adjusting the constants  $k_{v_i}$  and  $k_{p_i}$  on the speed and position control loops respectively, we can change the behaviour of the joint system:

- $k_{v_i} > k_{p_i}$  results in a joint system with effectively more damping.
- $k_{v_i} < k_{p_i}$  results in a joint system with effectively less damping.

It is common to make the  $k_{d_i}$  gain of the PID controller low, or even zero, in this kind of controller, so as not to have the same problem as before, caused by the noise amplifying effect of the derivation of the speed signal. If  $k_{d_i}$  is kept low or at zero, we can use the a larger  $k_{v_i}$  value to improve the dynamic behaviour.

As this kind of speed control is relatively easy to implement, and can be an effective and cheap way of improving the overall dynamic behaviour of the joint system being controlled. That is why this kind of strategy is often found in the control of the larger joints of industrial robot arms.

### 10.2.6 Feedforward and Feedback Compensations

Previous sections focused on how to eliminate the steady state error and overshoot problems. At this moment, the only proposed solution is the use of a PID controller (figure 10.8). From a theoretical point of view, high PID gains ( $k_{p_i}$ ,  $k_{I_i}$  and  $k_{d_i}$ ) should solve all these problems and yield highly accurate positioning. In practice, this is a utopian goal that cannot be reached due to the common presence of a gain boundaries that, when trespassed, cause the robot's behaviour to become unstable.

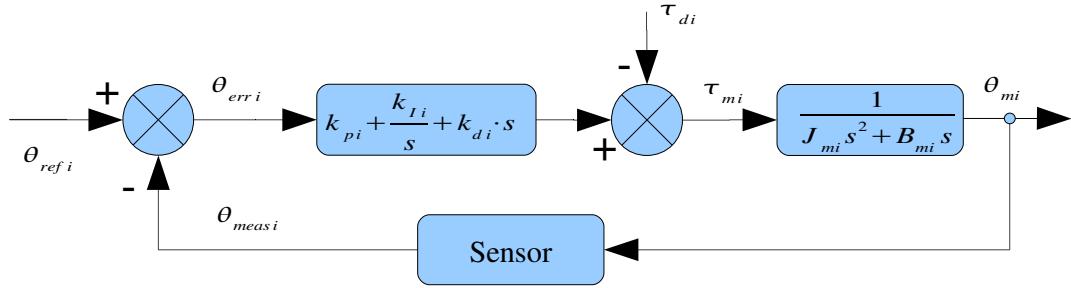


Figure 10.8: Basic PID control algorithm.

Thus, the proposed approach in this section is to keep the robot's behaviour stable despite having low PID gains. Motion accuracy will be then improved by modifying the basic PID control strategy.

However, we should first analyse the error signal in the Laplace domain of the basic PID controller. This will give us some design cues for our controller.

$$\theta_{err_i} = \frac{\theta_{ref_i} (J_{mi}s^2 + B_{mi}s) + \tau_{di}}{PID_i + (J_{mi}s^2 + B_{mi}s)} \quad (10.33)$$

where  $\tau_{di}$  is the disturbance force/torque exerted on the  $i^{th}$  DoF. That perturbation/disturbance can include noise, non-linearities and controller tuning inaccuracies.

The previous equation denotes that the tracking error of the PID controller depends on:

- the set-point → the bigger the set-point, the higher error rate is. However, we cannot control the set-point because the set-value is the position reference and it depends upon where we want the robot to go.
- controller stiffness → the higher the stiffness is, the lower the error. This issue was previously discussed at the beginning of this section. Now, we assume that we have already tuned the controller to have the maximum PID stiffness that guarantees the robot's stability.
- robot impedance → there is an interplay between the impedance values and the error, but clearly, the lower the impedance is, the lower the effects on the error the set-point has. If we have a good dynamical model, we might be able to use it to improve our controller.
- perturbations → the more perturbations there are, the bigger the error rate is. If we analyse better the source of the perturbations better, we can try to compensate for them.

Now, the goal is to reduce as many terms as we can in equation 10.33. The first try can be the compensation of the robot's dynamics. This can be done by adding

a compensation block to the basic PID control scheme (figure 10.9). This kind of compensation is known as ***feedforward compensation***<sup>6</sup>.

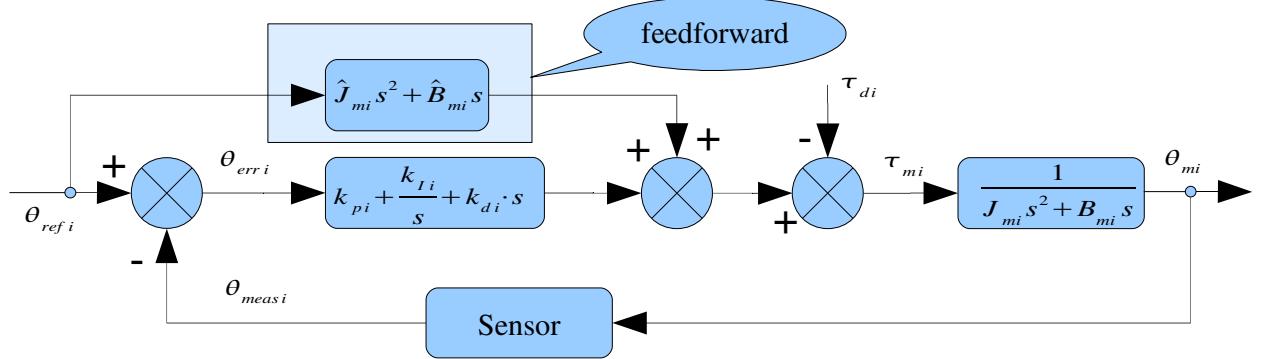


Figure 10.9: Feedforward compensation.

If we consider that the estimated parameters (parameters with hat) are equal to the real robot parameters (without hat), the equation for the error signal is:

$$\theta_{err_i} = \frac{\tau_{d_i}}{PID_i + (J_{mi}s^2 + B_{mi}s)} \quad (10.34)$$

If we compare the latter equation with the equation 10.33, we see that now there is no dependency on the set-point and the influence of the robot dynamics is much lower. This means that we have improved the transitory behaviour.

According to equation 10.34, in the case of zero-perturbations, the tracking should be error free. Remember that this is true only if we have a good model in the feedforward block that perfectly cancels the robot dynamic.

As we saw in the section about steady state error, one of the most typical perturbations is gravity. In that section, we saw that that perturbation can be eliminated by introducing an integral term in the controller. However, this increases the overshoot. Moreover, the gravity is not a linear disturbance and if we want to guarantee the performance of the system we need to introduce non-linear strategies.

One way to reduce the effects of gravity is to compensate for them as shown in figure 10.10. This compensation is known as ***feedback compensation*** and the expression for  $G(\theta_i)$  can be as general<sup>7</sup> as we need to estimate the force/torque we have to introduce to cancel the effects of gravity. This approach is so general that we can compensate other non-linearities as static friction.

<sup>6</sup>The hat over some parameters denotes that these parameters are estimations introduced in the controller.

<sup>7</sup>even as a non-linear expression

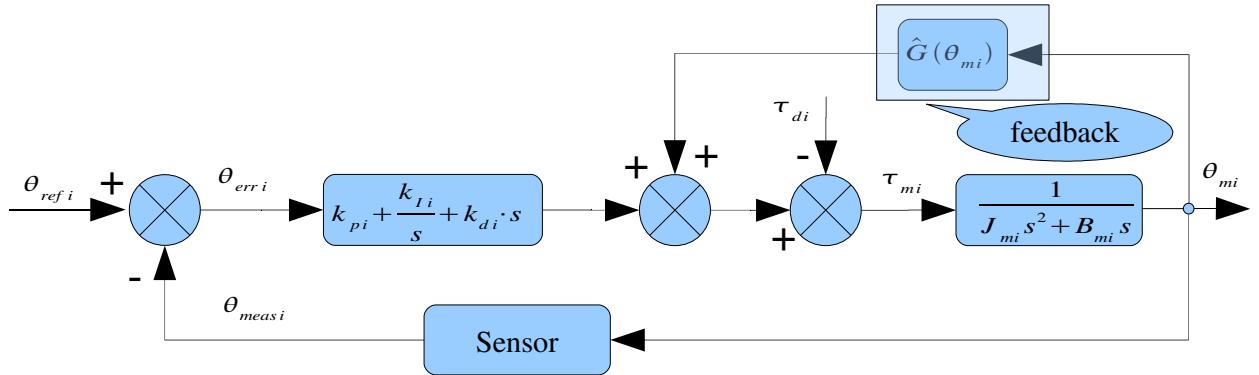


Figure 10.10: Feedback compensation.

### 10.3 Nonlinear Position Controller: Computed Torque Algorithm

#### 10.3.1 Linear Dynamic Model of n-DoF Joint Manipulator

Without trying to demonstrate any terms, the general expression for the actual dynamic equation in joint space is:

$$\tau = M(q)\ddot{q} + V_{iner}(q, \dot{q}) + F_{fricV}(\dot{q}) + F_{fricC}(q, \text{sgn}(\dot{q})) + G(q) \quad (10.35)$$

The estimated dynamic equation in joint-space is:

$$\hat{\tau} = \hat{M}(q)\ddot{q} + \hat{V}_{iner}(q, \dot{q}) + \hat{F}_{fricV}(\dot{q}) + \hat{F}_{fricC}(q, \text{sgn}(\dot{q})) + \hat{G}(q) \quad (10.36)$$

where:

$M(q)$ , the actual robot mass matrix (nxn).

$V_{cor}(q, \dot{q})$ , the actual robot coriolis force(nx1).

$V_{cen}(q, \dot{q})$ , the actual robot centrifugal force (nx1).

$V_{iner}(q, \dot{q}) = V_{cen}(q, \dot{q}) + V_{cor}(q, \dot{q})$ , the actual robot coriolis and centrifugal forces (nx1).

$F_{fricV}(\dot{q})$ , the actual robot dynamic friction (damping) (nx1).

$F_{fricC}(q, \text{sgn}(\dot{q}))$ , the actual coulomb friction (nx1).

$F_{fric} = F_{fricC}(q, \text{sgn}(\dot{q})) + F_{fricV}(\dot{q})$ , the actual friction (nx1).

$G(q)$ , the actual gravity force (nx1).

#### 10.3.2 Computed Torque Algorithm

As we have seen, the basic control level of a robot manipulator arm combines as many PID controllers as there are joints in the robot arm. These all function independently of

each other. In other words, the basic control of a 6 DoF robot arm is not treated as a 6-input/6-output control problem, and we do not use multi-input/multi-output controller design principles to design and implement this basic level of control.

In this type of control, the idea is to use a model of the dynamics of the controlled joint system to calculate, in real time, what additional torque should be applied to the joint system to remove or to minimise the effects of the changes in configuration and object mass.

This kind of control can be understood as a kind of feedforward and feedback compensated PID-control, since it attempts to correct for the effects of changing inertia at the joints by generating an additional control torque.

The computed torque algorithm is illustrated in figure 10.11.

Notice that as this controller is not linear, we cannot use Laplace. Moreover, the robotic system is treated as a coupled multi-DoF.

The equation for the error signal is given by<sup>8</sup>:

$$\ddot{q}_{err} + K_d \dot{q}_{err} + K_p q_{err} = \hat{M}^{-1} [\tau_d + (V_{iner} - \hat{V}_{iner}) + (F_{fricV} - \hat{F}_{fricV}) + (F_{fricC} - \hat{F}_{fricC}) + (G - \hat{G})] \quad (10.37)$$

When properly estimated parametres (with hat) are equal to the real parametres, yielding:

$$\ddot{q}_{err} + K_d \dot{q}_{err} + K_p q_{err} = \hat{M}^{-1} \tau_d \quad (10.38)$$

When the perturbations are equal to zero:

$$\ddot{q}_{err} + K_d \dot{q}_{err} + K_p q_{err} = 0 \quad (10.39)$$

As the last equation is a set of  $n$  linear differential equations decoupled, we can use classic control tools to tune the *PD* controller and to analyse the stability. Remember, however, that this is only possible if two conditions are fulfilled: accurate estimation of the robot's dynamics and zero-perturbations.

When it comes to constant perturbations, we can add an integral part in the algorithm, resulting the following equation:

$$\ddot{q}_{err} + K_d \dot{q}_{err} + K_p q_{err} + K_I \int q_{err} dt = \hat{M}^{-1} \tau_d \quad (10.40)$$

---

<sup>8</sup>Note that now the parametres are not escalar, but matricial and the signals are vectorial

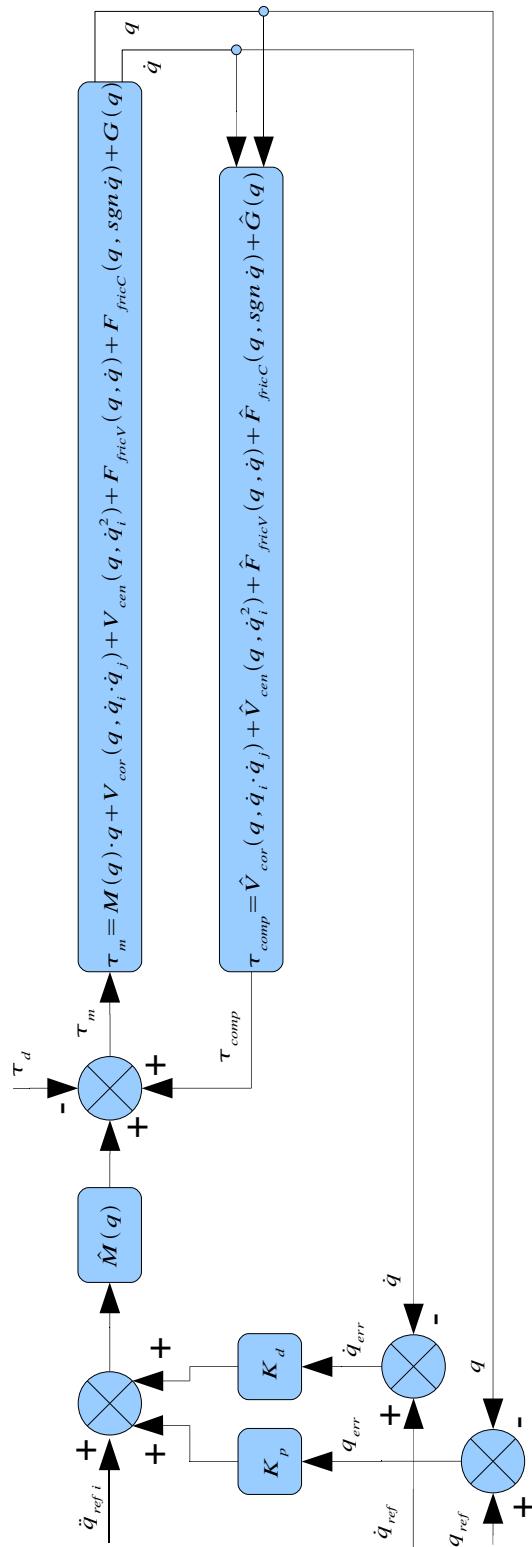


Figure 10.11: Computed-Torque control algorithm.

Deriving both sides of the equation we have:

$$\ddot{q}_{err} + K_d \ddot{q}_{err} + K_p \dot{q}_{err} + K_I q_{err} = 0 \quad (10.41)$$

This equation is again a set of  $n$  linear differential equations decoupled.

## 10.4 Adaptive Position Control of Joint Systems

In this section, we will briefly look at three different strategies for implementing **adaptive control** of joint systems in robot manipulator arms. The basic idea of adaptive control is the real-time modification of the controller parameters that in classic control are supposed to be constant. This parameter modification is applied to the instant system behaviour.

In robotics, three types of adaptive control are traditionally used:

- **Gain Scheduling,**
- **Model Reference Adaptive Control,** and
- **Dynamic Model-based Adaptive Control**

### 10.4.1 Gain Scheduling

The main causes of variation of the values of robot inertia at the joint of a robot arm are the changes in configuration of the manipulator arm and the mass of any object the arm is carrying in its gripper.

As the positions of each of the joints is known from the joint position sensors, we can always know the current configuration of the robot arm.

It may also be possible to know, perhaps by measurement, what the mass the robot is carrying, especially if there is only a predefined set of objects or tools that the robot has to carry.

By knowing the configuration and mass of any object being carried, it is theoretically possible to define good values of the PID gain constants  $k_{p_i}$ ,  $k_{I_i}$  and  $k_{d_i}$ .

Of course, if we defined values for these three gain constants for all possible configurations and object masses, we would have an infinite number of sets of three values.

In practice, it is often possible to divide the workspace of the robot into regions, or sub-volumes, for which we can usefully define particular sets of good values for the PID gains of each joint of the robot arm as a whole.

We can therefore generate a table, which defines the three gain values,  $k_{p_i}$ ,  $k_{I_i}$  and  $k_{d_i}$ , for each joint, for each work space region and for each object.

If it is important to have different PID gains for different kinds of movements (one set for slow, accurate movements, and a different set for fast, less accurate movements), we can extend the table to include this extra dimension.

Using such a table of sets of PID gains to adjust the controllers of each of the joints of a robot arm is called **Gain Scheduling**. It is one of the oldest kinds of adaptive control and was first developed and used in the aerodynamic control of airplanes, for which flying slowly and close to the ground requires different control parameters than those needed for flying fast and well above the ground.

The basic strategy for Gain Scheduling of joint  $i^{th}$  is illustrated in figure 10.12. This is repeated for each of the gain scheduled joint systems.

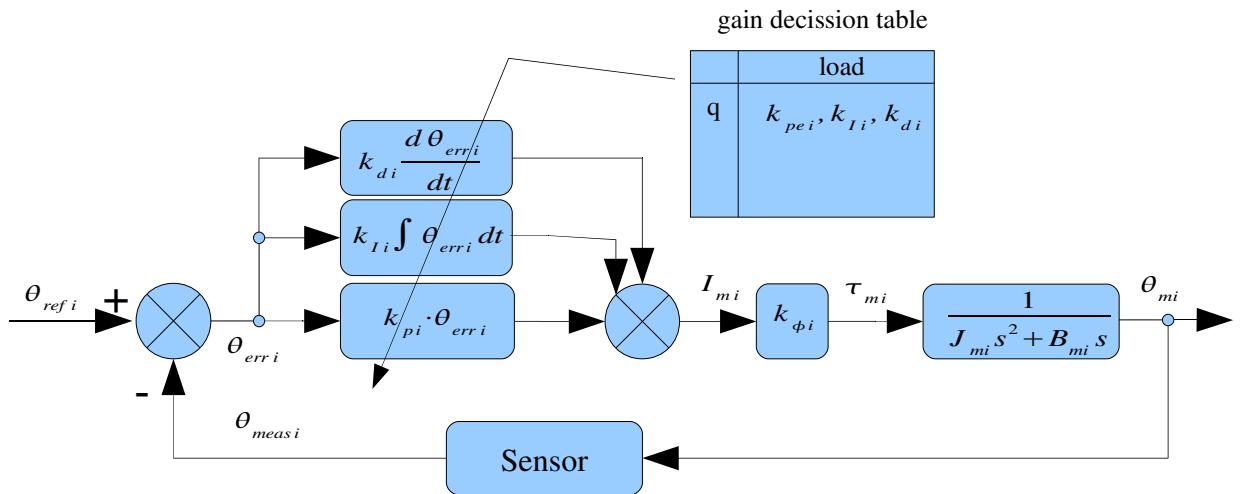


Figure 10.12: Gain Scheduling control algorithm.

If the table needed to achieve good overall performance of the robot arm is not too big, then Gain Scheduling can be a very effective way of implementing adaptive control.

It is also relatively easy and cheap to implement. It is not difficult to maintain, modify, or to change the table, as long as it is well-documented.

It does not need explicit knowledge of the kinematics or dynamics of the robot arm being controlled: the PID gains values can be found experimentally, rather than analytically.

Nonetheless, this control strategy has two particular disadvantages. Firstly, the number of table entries increases exponentially according to the number of degrees of freedom and how fine the discretization of the robot's workspace is. Secondly, the stability of the controller is not guaranteed if we use an interpolator for the gains.

### 10.4.2 Model Reference Adaptive Control (MRAC)

In **Model Reference Adaptive Control (MRAC)**, we use a mathematical model of the joint system to define the ideal behaviour of the joint. The output of this model is then compared, in real time, with the actual behaviour of the PID controlled joint. Then, the difference between the (model-based) ideal behaviour and the actual behaviour is used to adjust the three gain values of the PID controller.

The basic strategy for this kind of adaptive control, for a single-joint system controller, is shown in figure 10.13.

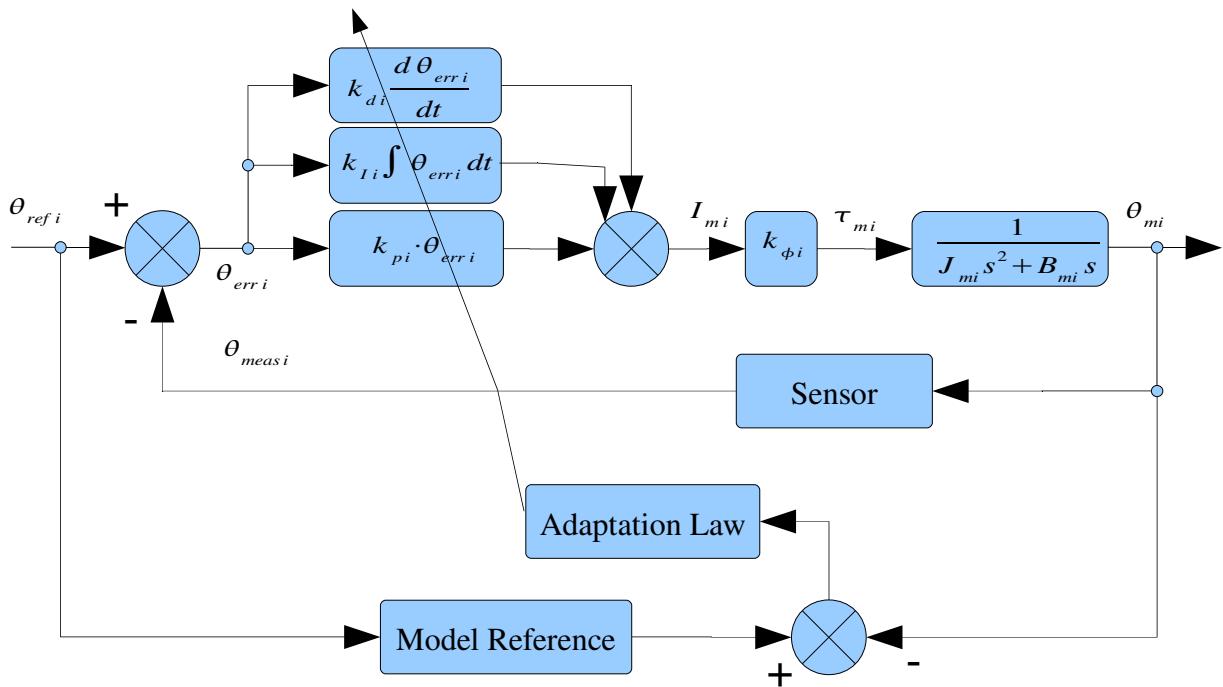


Figure 10.13: Model Reference Adaptive Control.

This kind of adaptive control can be used to adjust the three gain values of the PID controller continuously, rather than discontinuously over discrete work space regions, as we saw in the Gain Scheduling strategy.

The frequency of the adaptive loop, the rate at which the three PID gains values are changed, must, however, be at least an order of magnitude slower than the sampling frequency of the basic negative feedback loop.

The effectiveness of this technique depends upon the quality of the model used to define the ideal behaviour of the joint system. Developing and maintaining such a model is not an easy task, and if it is not a reliable model, or it becomes inaccurate, due to changes in the real joint system, this kind of control can become worse than ordinary (fixed gain) PID control.

Another difficulty with implementing this kind of adaptive control is in the implemen-

tation of the algorithm that decides how each of the three PID gains should be modified. This clearly has to calculate appropriate changes to make, but it has to do this in real time.

Until a few years ago, this typically required more computational power (faster microprocessors) than was economically feasible. This is currently much easier to do, thanks to faster microprocessors and digital signal processors (DSPs) and this kind of MRAC has become more widely used.

#### **10.4.3 Dynamic Model-based Adaptive Control (Computed Torque Adaptive Control)**

The biggest challenge faced when implementing computed torque control is finding an accurate model estimation of the robot's dynamic. The adaptive version of computed torque control is designed to improve, in real time, the offline model estimation.

This method of adaptive control is thus different from the two previous methods, since it does not attempt to adjust the three PID gain values. Rather, it calculates an additional control torque to compensate for the fact that the actual PID gains may not be the best at any moment: it adapts the control torque, rather than the PID gains.

The basic algorithm for this type of adaptive control applied to the control of one joint system is shown in figure 10.14.

## **10.5 Some Things to Think About**

1. Why is it acceptable to achieve effective multi-input/multi-output control using completely independent controllers and what are the restrictions that must be accepted as a result of doing so?
2. Why is it difficult, if not impossible, to use any of the two conventional empirical methods for adjusting the gain values of the PID controllers used in the basic level control of a robot manipulator arm? What are these methods?
3. When would we need to consider using adaptive control techniques in the design and implementation of the basic level of control of a robot manipulator arm and why?

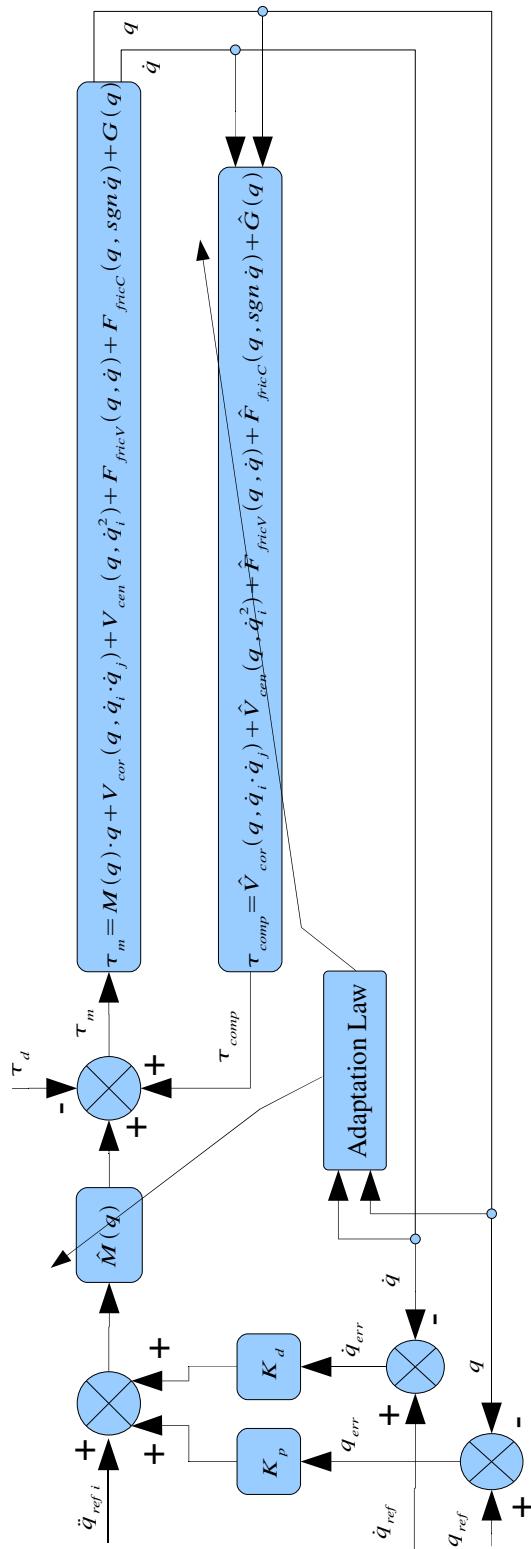


Figure 10.14: Computed-Torque Adaptive Control.

## **Part IV**

# **Exercises**



## **Exercises i**

# **Kinematic structures and programming**

---

### **Exercise i.1**

Figure i.1 shows a table with fur pieces on top, and nearby there are boxes to classify and store them.

How many degrees of freedom are required to do this work? Which robot geometry would be appropriate to do this? (There maybe be more than one solution).

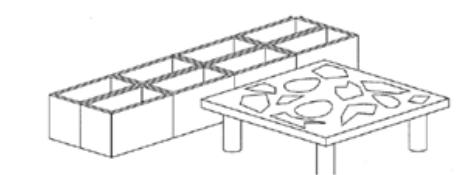


Figure i.1: Fur classification.

### Exercise i.2

Scorbot ER VII robot (Figure i.2) is a versatile system for educational use. Due to its speed and repeatability, it is highly suited for both stand-alone operations and integrated use in automated work cell applications such as robotic welding, machine vision, CNC machine tending and other FMS operations. (See figure i.3 to see robot dimension drawing).

Identify the kinematic chain on this SCORBOT ER VII robot.

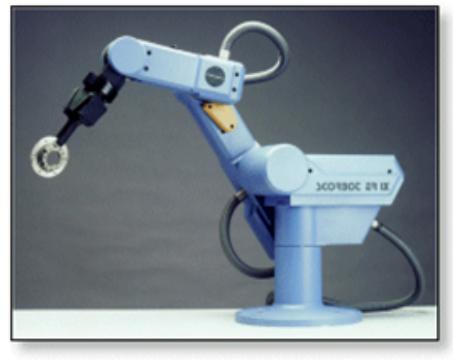


Figure i.2: Photo of Scorbot Robot.

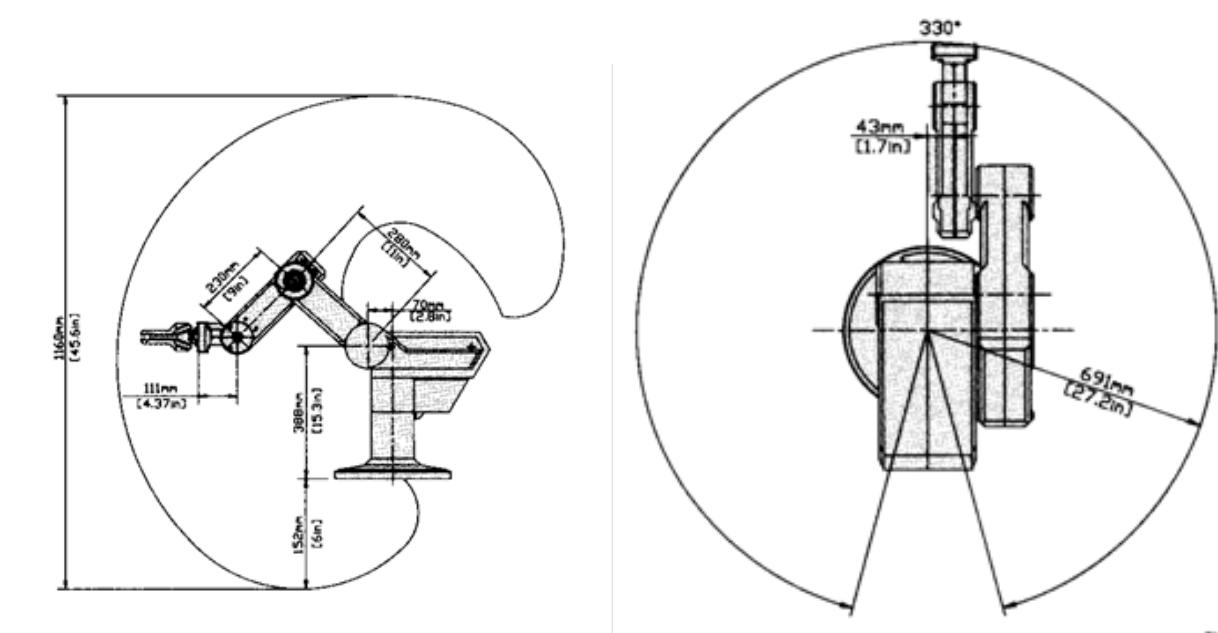


Figure i.3: Dimension plot of Scorbot Robot.

### Exercise i.3

RT3300 (Seiko, inc) cylindrical robots have been geometrically designed to reach into and out of constrained workspaces. Notice that this robot is a common example of the SCARA geometry. The fast rotary motion of cylindrical robots is combined with the precision linear motion of cartesian robots.

Identify the kinematic chain on this Seiko RT3300 robot.

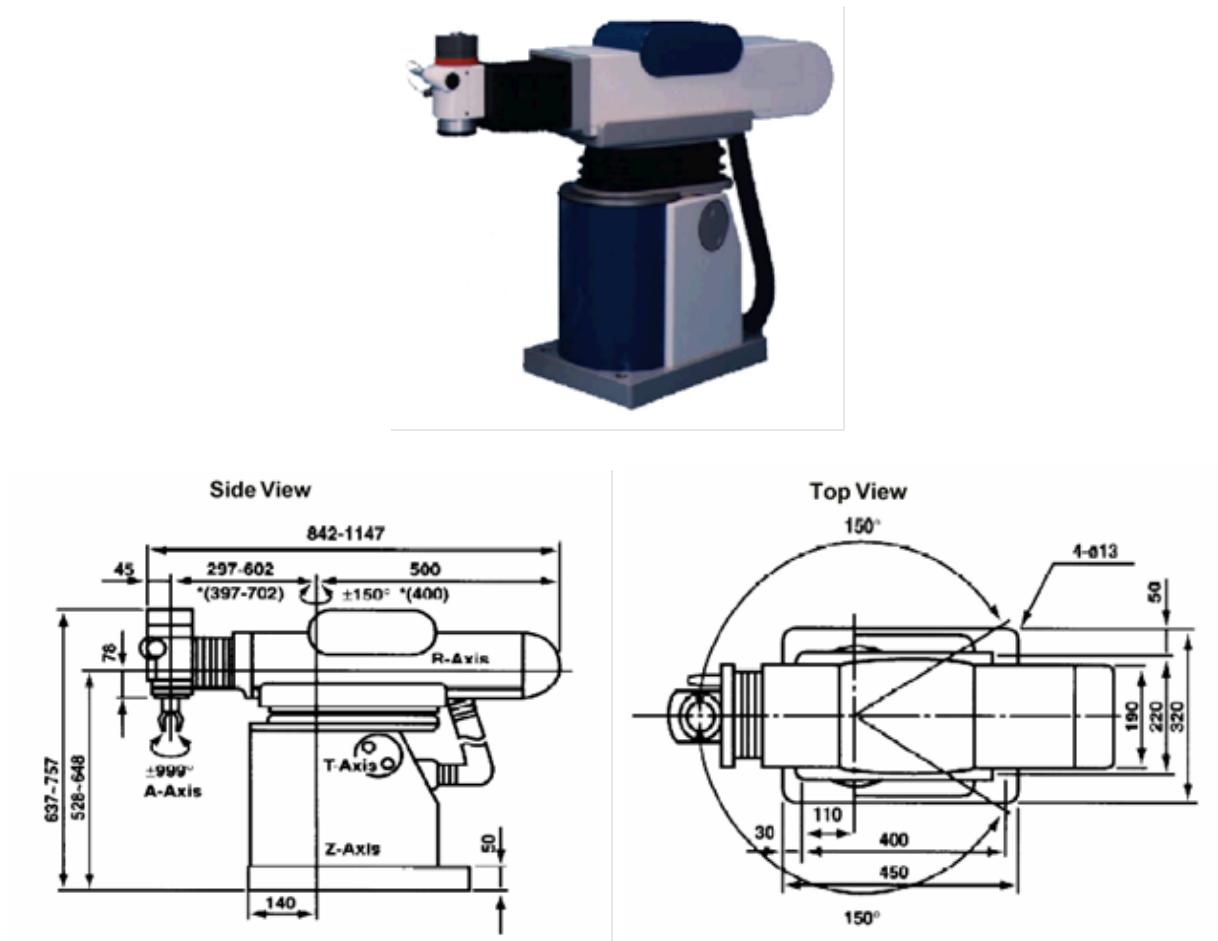


Figure i.4: RT3300 Robot.

**Exercise i.4**

Which geometry do these robots have?

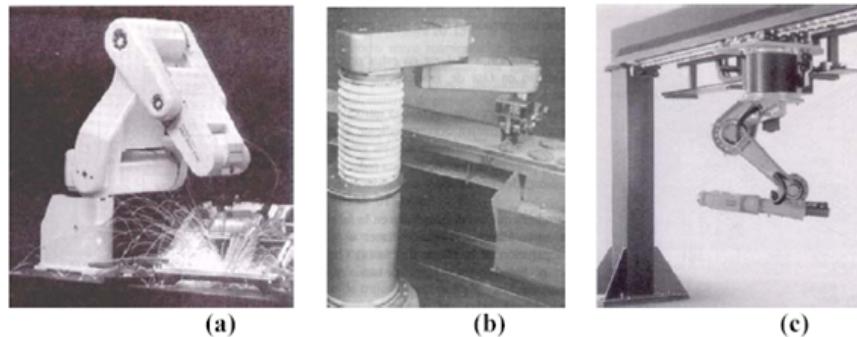


Figure i.5: Which geometries?.

**Exercise i.5**

Develop and specify a program for the last 3 programming levels (robot level, object level, and task level) of a robot manipulator with 6 DoF. This robot has to do the work specified in figure i.6. Use your own programming pseudo languages for each program, and specify all the information used. If the robot needs additional tools or external sensors, specify them and briefly explain its necessity and use.

**Task to do by the robot:** Move blocks A and B and build a tower at the top of block C. The blocks order should be C, A, B from the bottom. The program should check that all blocks are properly aligned.

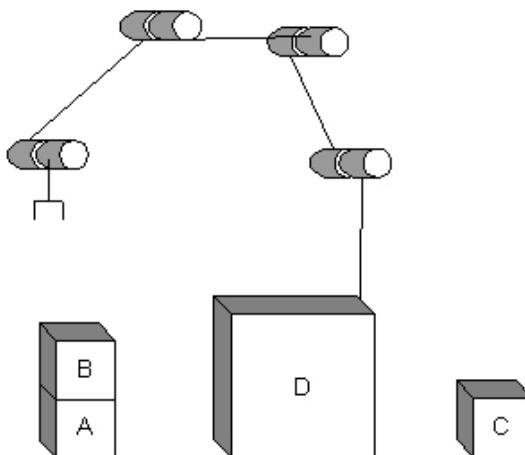


Figure i.6: Moving blocks.

## Solutions

### Solution to exercise i.1

To analyze how many degrees of freedom the robot should have to develop this task, a good strategy is to imagine that the initial position of the robot's end-effector is set one meter above the table and then try to perform a generic task:

1. **Go down** to the table surface and pick up a piece of fur.
2. **Go up** and make **horizontal movements** to go to the appropriate storing box.
3. Over the boxes, make **vertical movements** to reach the destination box.
4. **Go down** and put the piece of fur in the destination box.

As we can see, we have made movements around the 3 positioning axis of the coordinate system, so we need 3 degrees of freedom to make this work. But do not forget that we may need one additional DoF (a Roll) for the wrist to align the gripper's fingers properly to get and drop the fur pieces correctly.

Now, we have to choose which robot geometry best fits this type of work. Since the pick-and-place movements are done in a flat surface, it's convenient to have a DoF aligned with the Z-axis. This requirement is fulfilled by Cartesian, Cylindrical and SCARA robots. If the parts are very heavy, a Cartesian-Gantry robot is the best choice. Otherwise, it's more convenient to reduce the number of linear DoF (since the maintenance of linear DoF is more expensive and there are more frequent problems of misalignment and flexion). So, the best robot could be a SCARA robot, with 3 DoF or 4 DoF, depending on how critical/crucial the orientation is with respect to grabbing and dropping parts.

If the choice is a Cartesian robot, it has 3 prismatic joints (PPP), which only allows translational movements.

If the selection is a SCARA robot, then we find two rotational joints and a prismatic joint (RRP), so we will use the translational movement to pick the pieces up and the rotational movements to put the pieces in the corresponding boxes.

### Solution to exercise i.2

First of all, we have to identify how many DoFs the robot has. We can see 5 rotational joints marked on the picture. Rotational joints contained on a vertical plane are represented by continuous lines and the other ones are represented by discontinuous lines.

Now, after identifying which are the robot movements in the picture, we will have to translate them into our nomenclature in order to simplify the system, and the result is shown in figure i.8.

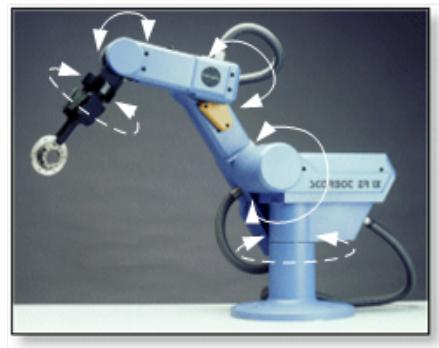


Figure i.7: Scobot's DoFs.

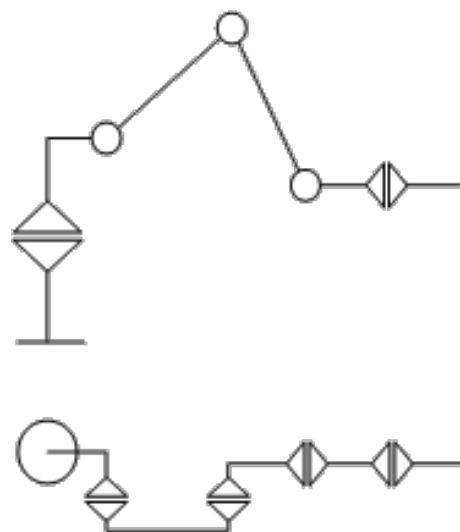


Figure i.8: Scobot's kinematic chain.

### Solution to exercise i.3

We will have to proceed the same way we did in the previous exercise. First we need to identify which movements the robot has by looking at the picture of the robot and the dimension diagram. Upon a simple inspection of the photo of the robot, we may find the search for the DoFs quite difficult. But notice that the drawings with dimensions show several dimension lines that are not constant, but rather a range of dimensions. For example, in the side view, the dimension that is on the top of the figure depicts the range '842-1147'. This is a clue that there is a linear DoF whose stroke varies from 842mm to 1147mm. We can find the last clues in the names of the axes: Z-Axis, T-Axis, R-Axis, A-Axis. These names should recall the axes of a cylindrical robot ( $\hat{z}$ ,  $\hat{R}$ ,  $\hat{\theta}$ ).

As a conclusion, we can place the arrangement of the DoF as it is depicted in the next figure.



Figure i.9: RT3300's DoFs.

Now, after identifying which are the robot movements in the picture, we will have to translate them into our nomenclature in order to simplify the system, and the result is:

### Solution to exercise i.4

- If we look at this picture we will notice that this robot is an anthropomorphic robot: it has 3 rotational joints (RRR), and its appearance is similar to a human arm. Moreover, this robot has a wrist with a PPR<sup>1</sup> movement. So, this is an anthropomorphic robot with 6 DoFs.
- On this robot we can distinguish a translational and two rotational movements, so we have a SCARA robot (PRR)<sup>2</sup>. It could also be a Spherical robot (because it also has a RRP joint arrangement), but the diagram does not concord with that kinematic

<sup>1</sup>Notice that PPR means Pitch-Pitch-Roll in wrist nomenclature (and not Prismatic-Prismatic-Rotational as when we speak about the positioner kinematic chain).

<sup>2</sup>As we are speaking about the positioner chain, once again P means prismatic DoF and R means rotational DoF

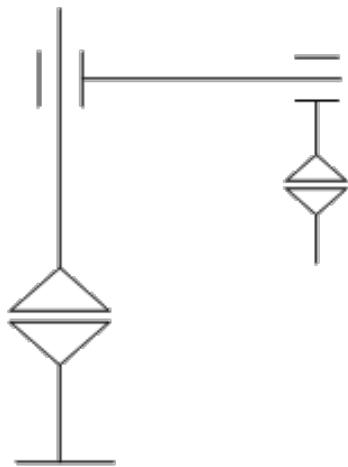


Figure i.10: RT3300's kinematic chain.

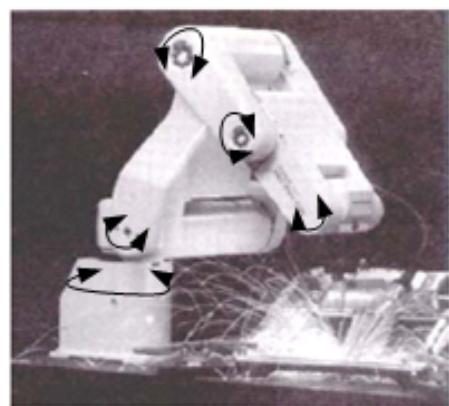


Figure i.11: Anthropomorphic Robot, sample 1.

structure. We also have another degree of freedom on the wrist: a roll movement. Therefore, this is a SCARA robot with 4 DoFs.

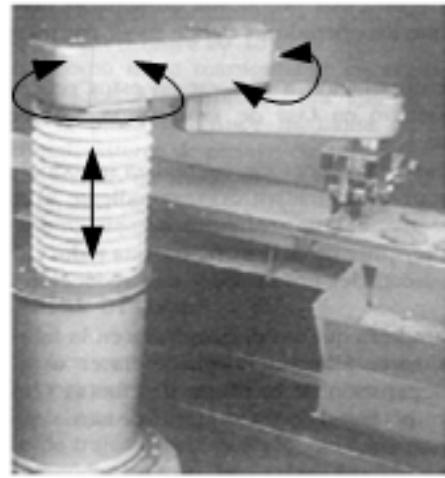


Figure i.12: SCARA Robot.

- c) This image, once again, represents an anthropomorphic robot, with 6 rotational movements. The first 3 are the positioner and the last 3 constitute the wrist (the typical roll-pitch-roll wrist), so this is an anthropomorphic robot with 6 DoFs. Furthermore, we can also say that this robot fits the structure of the well-known PUMA robot (Programmable Universal Machine for Assembly), which was developed in 1975. This robot geometry is, by far, the most popular in industrial robotics.

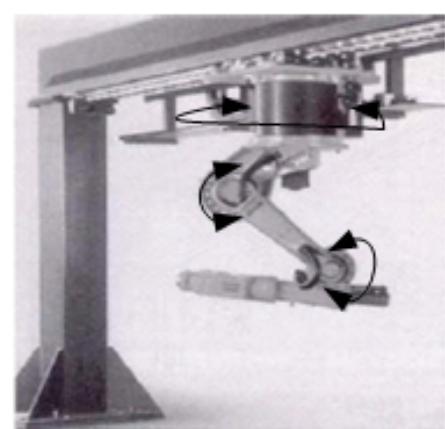


Figure i.13: Anthropomorphic Robot, sample 2.

### Solution to exercise i.5

First of all, we have to define the points we want the robot to reach. These points are depicted in the following figure.

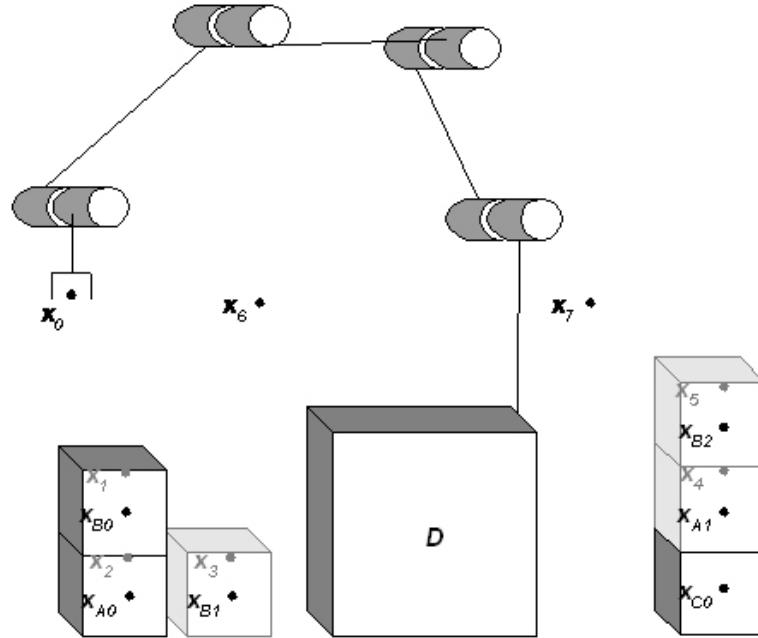


Figure i.14: Programming example points.

Then, we have to remember that there are four robot programming levels. However, we will only use three (we omit the lowest level, the joint-level): robot level, object level, and task level. Let's see an example of how it could be solved, because there may be more than one solution due to the steps and language chosen by each person.

**ROBOT LEVEL:**

Go to point X0 with high velocity  
Go to point X1 with high velocity  
Open the gripper  
Go to point XB0 with low velocity  
Close the gripper  
If the block has not been picked up by the gripper, error and end  
If the block has been picked up by the gripper, continue  
Go to point X1 with low velocity  
Go to point X3 with high velocity  
Go to point XB1 with low velocity  
Open the gripper  
Go to point X3 with low velocity  
Go to point X2 with high velocity  
Go to point XA0 with low velocity  
Close the gripper  
If the block has not been picked up by the gripper, error and end  
If the block has been picked up by the gripper, continue  
Go to point X2 with low velocity  
Go to point X6 with high velocity  
Go to point X7 with high velocity  
Go to point X4 with high velocity  
Go to point XA1 with low velocity  
Open the gripper  
Go to point X4 with low velocity  
Go to point X7 with high velocity  
Go to point X6 with high velocity  
Go to point X3 with high velocity  
Go to point XB1 with low velocity  
Close the gripper  
If the block has not been picked up by the gripper, error and end  
If the block has been picked up by the gripper, continue  
Go to point X3 with low velocity  
Go to point X6 with high velocity  
Go to point X7 with high velocity  
Go to point X5 with high velocity  
Go to point XB2 with low velocity  
Open the gripper  
Go to point X5 with low velocity  
End

**OBJECT LEVEL:**

```
Pick up block-B  
Put block-B at XB1  
Pick up block-A  
Put block-A at XA1  
Pick up block-B  
Put block-B at XB2  
Align block-A with block-C  
Align block-B with block-A  
End
```

**TASK LEVEL:**

Stack a tower in order block-C, block-A, block-B without moving block-C and checking that the blocks are correctly aligned.

## **Exercises ii**

# **Transformations in space (homogeneous transformation)**

---

### **Exercise ii.1**

Obtain the rotation matrix after doing the following operations to go from the reference system  $\{0\}$  (fixed) to the system  $\{e\}$  (mobile):

1. Rotate an angle of  $90^\circ$  around  $\hat{z}_0$
2. Rotate an angle of  $45^\circ$  around  $\hat{x}_e$
3. Rotate an angle of  $30^\circ$  around  $\hat{z}_e$

### **Exercise ii.2**

Obtain the rotation matrix after doing the following operations to go from the reference system  $\{0\}$  (fixed) to the system  $\{e\}$  (mobile):

1. Rotate an angle of  $90^\circ$  around  $\hat{z}_0$
2. Rotate an angle of  $30^\circ$  around  $\hat{z}_0$
3. Rotate an angle of  $45^\circ$  around  $\hat{x}_e$

### **Exercise ii.3**

Obtain the rotation matrix after doing the following operations to go from the reference system  $\{0\}$  (fixed) to the system  $\{e\}$  (mobile):

1. Rotate an angle of  $90^\circ$  around  $\hat{z}_0$
2. Rotate an angle of  $45^\circ$  around  $\hat{y}_0$
3. Rotate an angle of  $30^\circ$  around  $\hat{z}_e$

**Exercise ii.4**

Obtain the rotation matrix after doing the following operations to go from the reference system  $\{0\}$  (fixed) to the system  $\{e\}$  (mobile):

1. Rotate an angle of  $90^\circ$  around  $\hat{x}_0$
2. Move 5 units along  $\hat{x}_0$
3. Rotate an angle of  $45^\circ$  around  $\hat{y}_0$

We know that the homogeneous coordinates of a vector  $P$  are  $P = (1, 1, 1, 1)^T$  expressed on the  $\{e\}$  coordinate system.

Obtain the coordinates of the  $P$  vector expressed on the  $\{0\}$  system.

**Exercise ii.5**

Obtain the rotation matrix after doing the following operations to go from the reference system  $\{0\}$  (fixed) to the system  $\{e\}$  (mobile):

1. Rotate an angle of  $90^\circ$  around  $\hat{z}_0$
2. Move 3 units along  $\hat{x}_0$
3. Rotate an angle of  $90^\circ$  around  $\hat{y}_e$
4. Rotate an angle of  $90^\circ$  around  $\hat{x}_0$
5. Move 2 units along  $\hat{z}_e$

Obtain the homogeneous transformation matrix that relates both reference systems. Is there any other sequence of movements simpler than this one?

**Exercise ii.6**

We have a robot equipped with a video camera for artificial vision applications. For the camera we define a coordinate system  $\{0\}$  whose origin is set at the focus of the camera (centre of the shutter) and its  $\hat{y}$  axis is set along the vision axis (direction from the camera to the target). We also add another coordinate system  $\{e\}$  whose origin is set at the end effector. Initially both systems coincide, except that the end effector is 2 units away from the focus of the camera, measured along  $\hat{y}$  axis (figure ii.1). From this initial situation, the coordinate system attached to the end effector undergoes the following transformations:

1. It moves 1 unit along  $\hat{z}_0$

2. It rotates an angle of  $45^\circ$  around  $\hat{x}_0$

3. It moves 1 unit along  $\hat{y}_e$

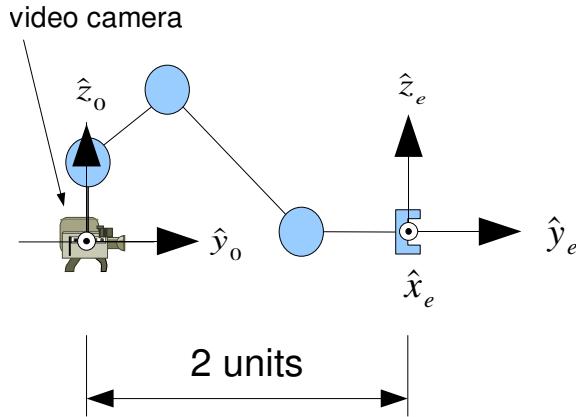


Figure ii.1: Robot with camera: initial configuration.

If we know that the range of vision of the camera is 60 degrees, is the end effector within or without this range after doing the sequence of movements?

### Exercise ii.7

The following relative descriptions between the  $\{U\}$ ,  $\{A\}$ ,  $\{B\}$  and  $\{C\}$  systems are known:

$${}^U T_A = \begin{bmatrix} 0.866 & -0.5 & 0 & 11 \\ 0.5 & 0.866 & 0 & -1 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^B T_A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.866 & -0.5 & 10 \\ 0 & 0.5 & 0.866 & -20 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^C T_U = \begin{bmatrix} 0.866 & -0.5 & 0 & -3 \\ 0.433 & 0.75 & -0.5 & -3 \\ 0.25 & 0.433 & 0.866 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Draw a diagram that shows qualitatively the relative arrangement of the four systems of reference. Obtain  ${}^B T_C$  and interpret the result over the diagrams.

**Exercise ii.8**

Obtain the matrix that represents the rotation of an angle of  $45^\circ$  around an axis set by the vector  $u = [1 \ 1 \ 2]^T$ . Obtain equivalent roll, pitch and yaw angles around the main axis of the reference system (fixed) that orient the mobile system in the same manner as the rotation around  $\hat{u}$  axis.

**Exercise ii.9**

For small rotations, the approximations  $\sin \theta \approx \theta$ ,  $\cos \theta \approx 1$  and  $\theta^2 \approx 0$  are valid. Obtain the matrix that describes a small rotation over a general axis  $\hat{u}$ . Demonstrate that from here on, that two infinitesimal rotations are commutative.

## Solutions

### Solution to exercise ii.1

The first thing we have to do to solve this exercise is to calculate the transformation matrices following the instructions according to the theory:

1. Rotate an angle of  $90^\circ$  around  $\hat{z}_0$

$$T_1 = \text{Rot}(\hat{z}_0, 90^\circ) = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Rotate an angle of  $45^\circ$  around  $\hat{x}_e$

$$T_2 = \text{Rot}(\hat{x}_e, 45^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 45^\circ & -\sin 45^\circ & 0 \\ 0 & \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Rotate an angle of  $30^\circ$  around  $\hat{z}_e$

$$T_3 = \text{Rot}(\hat{z}_e, 30^\circ) = \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After doing this, we will have to multiply the three matrices to obtain the transformation matrix. We put the matrix of the first movement in the middle of the equation, and then let's see what we have. The next instruction is related to the  $\{e\}$  system so, as we saw in the lesson, we will have to put this matrix after the matrix related to first movement.

The third movement is related to the  $\{e\}$  system again, so we will have to post-multiply the term again. The result then is:

$${}^0T_e = T_1 \cdot T_2 \cdot T_3 = \begin{bmatrix} -0.3536 & -0.6124 & 0.7071 & 0 \\ 0.866 & -0.5 & 0 & 0 \\ 0.3536 & 0.6124 & 0.7071 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Solution to exercise ii.2

The first thing we have to do to solve this exercise is to calculate the transformation matrices following the instructions according to the theory:

(a) Rotate an angle of  $90^\circ$  around  $\hat{z}_0$

$$T_1 = \text{Rot}(\hat{z}_0, 90^\circ) = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Rotate an angle of  $30^\circ$  around  $\hat{z}_0$

$$T_2 = \text{Rot}(\hat{z}_0, 30^\circ) = \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) Rotate an angle of  $45^\circ$  around  $\hat{x}_e$

$$T_3 = \text{Rot}(\hat{x}_e, 45^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 45^\circ & -\sin 45^\circ & 0 \\ 0 & \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After doing this, we will have to multiply the three matrices to obtain the transformation matrix. We put the matrix of the first movement in the middle of the equation, and then let's see what we have. The next instruction is related to the  $\{0\}$  system so, as we saw in the lesson, we will have to put this matrix before the matrix related to first movement.

The third movement is related to the  $\{e\}$  system again, so we will have to post-multiply the term. The result then is:

$${}^0T_e = T_2 \cdot T_1 \cdot T_3 = \begin{bmatrix} -0.5 & \frac{\sqrt{6}}{4} & \frac{\sqrt{6}}{4} & 0 \\ \frac{\sqrt{3}}{2} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Solution to exercise ii.3

The first thing we have to do to solve this exercise is to calculate the transformation matrices following the instructions according to the theory:

(a) Rotate an angle of  $90^\circ$  around  $\hat{z}_0$

$$T_1 = \text{Rot}(\hat{z}_0, 90^\circ) = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Rotate an angle of  $45^\circ$  around  $\hat{y}_0$

$$T_2 = \text{Rot}(\hat{y}_0, 45^\circ) = \begin{bmatrix} \cos 45^\circ & 0 & \sin 45^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 45^\circ & 0 & \cos 45^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) Rotate an angle of  $30^\circ$  around  $\hat{z}_e$

$$T_3 = \text{Rot}(\hat{z}_e, 30^\circ) = \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After doing this, we will have to multiply the three matrices to obtain the transformation matrix. We put the matrix of the first movement in the middle of the equation, and then let's see what we have. The next instruction is related to the  $\{0\}$  system so, as we saw in the lesson, we will have to put this matrix before the matrix related to first movement.

The third movement is related to the  $\{e\}$  system again, so we will have to post-multiply the term. The result then is:

$${}^0T_e = T_2 \cdot T_1 \cdot T_3 = \begin{bmatrix} 0.8836 & -0.51 & 0 & 0 \\ 0.5102 & 0.8836 & 0 & 0 \\ 0 & 0 & 1.0204 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Solution to exercise ii.4

The first thing we have to do to solve this exercise is to calculate the transformation matrices following the instructions according to the theory:

(a) Rotate an angle of  $90^\circ$  around  $\hat{x}_0$

$$T_1 = \text{Rot}(\hat{x}_0, 90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 90^\circ & -\sin 90^\circ & 0 \\ 0 & \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Move 5 units along  $\hat{x}_0$

$$T_2 = \text{Trasl}(\hat{x}_0, 5) = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) Rotate an angle of  $45^\circ$  around  $\hat{y}_0$

$$T_3 = \text{Rot}(\hat{y}_0, 45^\circ) = \begin{bmatrix} \cos 45^\circ & 0 & \sin 45^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 45^\circ & 0 & \cos 45^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After doing this, we will have to multiply the three matrices to obtain the transformation matrix. We put the matrix of the first movement in the middle of the equation, and then let's see what we have. The next instruction is related to the  $\{0\}$  system so, as we saw in the lesson, we will have to put this matrix before the matrix related to first movement.

The third movement is related to the  $\{0\}$  system again, so we will have to pre-multiply the term again. The result then is:

$$\begin{aligned} {}^0T_e &= T_3 \cdot T_2 \cdot T_1 = \begin{bmatrix} 0.7071 & 0.7071 & 0 & 3.5355 \\ 0 & 0 & -1 & 0 \\ -0.7071 & 0.7071 & 0 & -3.5355 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^0p &= {}^0T_e {}^ep = {}^0T_e \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6.414 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{aligned}$$

### Solution to exercise ii.5

The first thing we have to do to solve this exercise is to calculate the transformation matrices following the instructions according to the theory:

- (a) Rotate an angle of  $90^\circ$  around  $\hat{z}_0$

$$T_1 = Rot(\hat{z}_0, 90^\circ) = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Move 3 units along  $\hat{x}_0$

$$T_2 = Trasl(\hat{x}_0, 3) = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. Rotate an angle of  $90^\circ$  around  $\hat{y}_e$

$$T_3 = Rot(\hat{y}_e, 90^\circ) = \begin{bmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6. Rotate an angle of  $90^\circ$  around  $\hat{x}_0$

$$T_4 = \text{Rot}(\hat{x}_0, 90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 90^\circ & -\sin 90^\circ & 0 \\ 0 & \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7. Move 2 units along  $\hat{z}_e$

$$T_5 = \text{Trasl}(\hat{z}_e, 2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix is:

$${}^0T_e = T_4 \cdot T_2 \cdot T_1 \cdot T_3 \cdot T_5 = \begin{bmatrix} 0 & -1 & 0 & 3 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Solution to exercise ii.6

In this exercise, we have to note that we have a first transformation matrix before doing any other movement because the text says that the end effector is 2 units away from the focus of the camera, measured along the  $\hat{y}$  axis, so the first matrix in this case will be:

$$T_1 = \text{Transl}(\hat{y}_0, 2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1. Move 1 unit along  $\hat{z}_0$

$$T_2 = \text{Transl}(\hat{z}_0, 1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Rotate an angle of  $45^\circ$  around  $\hat{x}_0$

$$T_3 = \text{Rot}(\hat{x}_0, 45^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 45^\circ & -\sin 45^\circ & 0 \\ 0 & \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Move 1 unit along  $\hat{y}_e$

$$T_4 = \text{Trasl}(\hat{y}_e, 1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix is:

$${}^0T_e = T_3 \cdot T_2 \cdot T_1 \cdot T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7071 & -0.7071 & 1.4142 \\ 0 & 0.7071 & 0.7071 & 2.8284 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To calculate the angle we will use the dot product:

$$\|{}^0P_e\| \cdot \|\hat{y}_0\| \cdot \cos \alpha = {}^0P_{ex} \cdot \hat{y}_{0x} + {}^0P_{ey} \cdot \hat{y}_{0y} + {}^0P_{ez} \cdot \hat{y}_{0z}$$

by substituting the numerical values obtained above, it yields:

$$\sqrt{1.4142^2 + 2.8284^2} \cdot \sqrt{1+0+0} \cdot \cos \alpha = 0 \cdot 0 + 1.4142 \cdot 1 + 2.8284 \cdot 0$$

and solving for  $\alpha$  we have:  $\cos \alpha = 63.435^\circ > 60^\circ$

The camera cannot see the robot's end-point.

However, only in this case does the method fail. As we have only done two translations before rotating around the  $\hat{x}$  axis, the  $\hat{x}_0$  axis is parallel to the  $\hat{x}_e$  axis. So, the rotations we have after this transformation in the equation are done around the mobile axis. What we have to do is to consider directly the equation of the homogeneous transformation matrix that involves the first three basic transformations:

$${}^0T_3 = \begin{bmatrix} {}^0R_1 & {}^0P_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 45^\circ & -\sin 45^\circ & 2 \\ 0 & \sin 45^\circ & \cos 45^\circ & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And now we can obtain the total transformation matrix by adding just the last basic transformation:

$${}^0T_e = {}^0T_3 \cdot T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7071 & -0.7071 & 2.7071 \\ 0 & 0.7071 & 0.7071 & 1.7071 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and redo the dot product.

### Solution to exercise ii.7

To obtain  ${}^B T_C$  we will proceed applying the properties of homogeneous transformation matrices:

$${}^B T_C = {}^B T_A \cdot {}^A T_U \cdot {}^U T_C$$

Notice that homogeneous transformation matrices are not orthonormal, they cannot be inverted by transposing it. Then we invert this matrix by using the homogeneous transformation matrix inversion rule.

$$\begin{aligned} {}^A T_U &= \begin{pmatrix} T \\ A \end{pmatrix}^{-1} = \begin{bmatrix} 0.866 & 0.5 & 0 & -9.026 \\ -0.5 & 0.866 & 0 & 6.3662 \\ 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^U T_C &= ({}^C T_U)^{-1} = \begin{bmatrix} 0.866 & 0.433 & 0.25 & 3.1471 \\ -0.5 & 0.75 & 0.433 & -0.55 \\ 0 & -0.5 & 0.866 & -4.098 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

The solution is:

$${}^B T_C = {}^B T_A \cdot {}^A T_U \cdot {}^U T_C = \begin{bmatrix} 0.5 & 0.75 & 0.433 & -6.5752 \\ -0.75 & 0.625 & -0.2165 & 19.7876 \\ -0.433 & -0.2165 & 0.8749 & -28.3183 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Solution to exercise ii.8

To solve this exercise we will have to use the formulation of the rotation matrix according to the convention “rotation around a single axis”. Let’s remember its expression:

$${}^a R_b = \begin{bmatrix} -(u_z^2 + u_y^2) A + 1 & u_x u_y A - u_z B & u_x u_z A + u_y B \\ u_x u_y A + u_z B & -(u_x^2 + u_z^2) A + 1 & u_y u_z A - u_x B \\ u_x u_z A - u_y B & u_y u_z A + u_x B & -(u_x^2 + u_y^2) A + 1 \end{bmatrix}$$

where:  $A = 1 - \cos \psi$  and  $B = \sin \psi$

Now we have to identify the terms given in the text of the exercise with these parameters. The angle is of  $45^\circ \rightarrow \psi = 45^\circ$  and the vector is  $u = [1 \ 1 \ 2]^T$ . However, if we take a look we realize that this is not a normal vector. So, the first thing to do is to normalize it:

$$\hat{u} = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} \end{bmatrix}^T$$

Now, we can apply the equations:

$$A = 1 - \cos \psi = 1 - \cos 45^\circ = 0.2929 \text{ and } B = \sin \psi = \sin 45^\circ = 0.7071$$

$${}^a R_b = \begin{bmatrix} 0.7559 & -0.5285 & 0.3863 & 0 \\ 0.6262 & 0.7559 & -0.191 & 0 \\ -0.191 & -0.3863 & 0.9024 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To obtain the roll, pitch and yaw angles we will equalize terms between this matrix and the terms of the version of the rotation matrix based on the “Euler angles convention”:

$$-\sin \phi_2 = -0.191 \rightarrow \phi_2 = 11.014^\circ \rightarrow \text{Pitch},$$

$$\cos \phi_y \cdot \sin \phi_x = 0.3863 \rightarrow \phi_x = 23.176^\circ \rightarrow \text{Roll and}$$

$$\cos \phi_z \cdot \cos \phi_y = 0.7559 \rightarrow \phi_z = 39.6365^\circ \rightarrow \text{Yaw}$$

### Solution to exercise ii.9

To solve this exercise we will proceed the same way as in the previous exercise:

$${}^a R_b = \begin{bmatrix} -(u_z^2 + u_y^2) A + 1 & u_x u_y A - u_z B & u_x u_z A + u_y B \\ u_x u_y A + u_z B & -(u_x^2 + u_z^2) A + 1 & u_y u_z A - u_x B \\ u_x u_z A - u_y B & u_y u_z A + u_x B & -(u_x^2 + u_y^2) A + 1 \end{bmatrix}$$

But now, we have to consider small rotation angles. In that case we can substitute  $\sin \theta \approx \theta$ ,  $\cos \theta \approx 1$  and  $\theta^2 \approx 0$  (small motion hypothesis):

$$A = 1 - \cos \psi = 1 - 1 = 0 \text{ and } B = \sin \psi = \psi$$

substituting in the rotation matrix, we have:

$${}^a R_b = \begin{bmatrix} 1 & -u_z \psi & u_y \psi \\ u_z \psi & 1 & -u_x \psi \\ -u_y \psi & u_x \psi & 1 \end{bmatrix}$$

The commutative property of this matrix can be easily checked by multiplying two matrices for different angles.

## Exercises iii

# Forward Kinematics

---

### Exercise iii.1

In the robotized environment shown in figure iii.1, the location of the gripper  ${}^W T_T$  is not known precisely. By using pressure sensors and using force control, the robot tries to insert the piece <sup>1</sup> into the hole <sup>2</sup>. Finally the robot introduces the tool into the hole, so  $\{T\}$  and  $\{G\}$  are coincident, and at that moment the robot position  ${}^B T_W$  is determined by reading the sensors of the joint angles and calculating the forward kinematics.

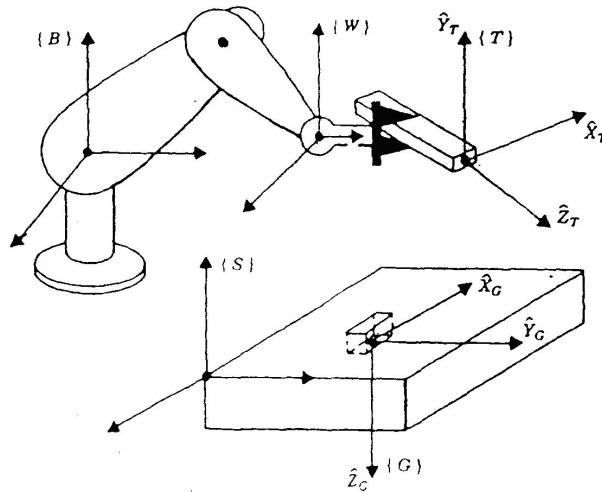


Figure iii.1: Peg-in-hole task.

If we suppose that  ${}^B T_S$  and  ${}^S T_G$  are known, obtain the expression to calculate the location of the tool  ${}^W T_T$ .

<sup>1</sup>located by the reference system  $\{T\}$

<sup>2</sup>which is located by the reference system  $\{G\}$  that is to say  ${}^S T_G$

**Exercise iii.2**

For the 2-element manipulator shown in figure iii.2 the transformation matrices  ${}^0T_1$  and  ${}^1T_2$  have been calculated.

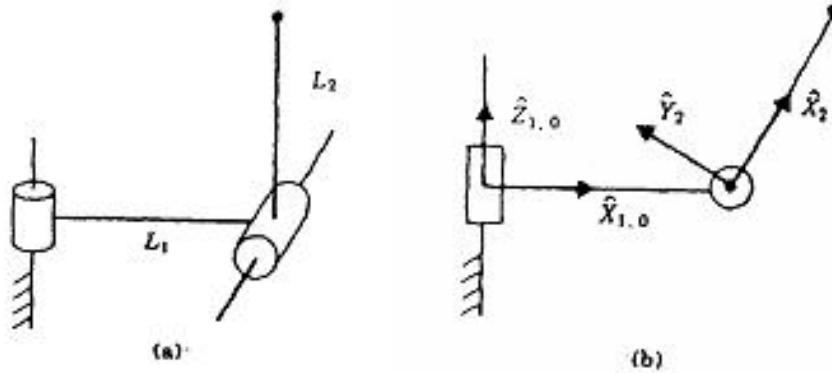


Figure iii.2: 2-DoF manipulator.

Assuming the reference systems shown in figure iii.2<sup>3</sup>, the product of these transformations is:

$$\begin{bmatrix} \cos \theta_1 \cdot \cos \theta_2 & -\cos \theta_1 \cdot \sin \theta_2 & \sin \theta_1 & L_1 \cdot \cos \theta_1 \\ \sin \theta_1 \cdot \cos \theta_2 & -\sin \theta_1 \cdot \sin \theta_2 & -\cos \theta_1 & L_1 \cdot \sin \theta_1 \\ \sin \theta_2 & \cos \theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Find an expression for the vector  ${}^0T_e$  that locates the end effector with respect to the reference system  $\{0\}$ . (The end effector will be situated at the end of the second element of the manipulator).

**Exercise iii.3**

The robot shown in figure iii.3 is a 3 DoF manipulator with three rotational joints and two axes that intersect and two axes which are parallel.

Obtain four different and equivalent representations of the Denavit and Hartenberg parameters.

**Exercise iii.4**

If we know the Denavit and Hartenberg table of a certain robot (iii.1). Draw the robot.

<sup>3</sup>Note that reference system  $\{0\}$  is coincident with reference system  $\{1\}$  when the first variable  $\theta_1$  is 0

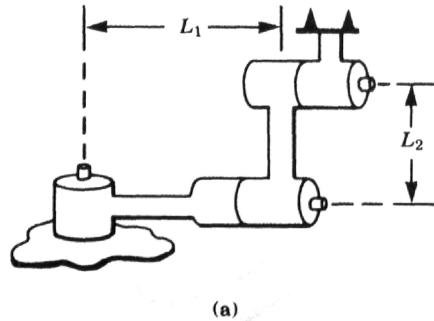


Figure iii.3: 3-DoF manipulator with multiple D-H solutions.

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
<i>link</i> $i-1 \rightarrow$	<i>link</i> $i$				
0 → 1		0	0	$L_1$	$\theta_1$
1 → 2		0	$\frac{\pi}{2}$	$d_2$	0
2 → 3		0	0	$L_2$	$\theta_3$

Table iii.1: D-H table of an unknown robot.

### Exercise iii.5

Figure iii.4 represents a PUMA robot. If this robot has all its joint angles equal to zero ( $\theta_i = 0, \forall i = 1, 2 \dots 6$ ), obtain an expression related to the characteristic parameters for the distance between the reference systems  $\{0\}$  and  $\{6\}$ .

### Exercise iii.6

Solve the forward kinematics problem for the 3R non-planar robot shown in figure iii.5. Obtain the characteristic parameters and the transformation matrix  ${}^B T_W$ .

### Exercise iii.7

Draw the reference systems for the joints of the planar robot RPR (the second joint is prismatic) shown in figure iii.6. Obtain the geometric characteristic parameters for this robot. Calculate the transformation matrices  ${}^0 T_1$ ,  ${}^1 T_2$ , and  ${}^2 T_3$  by using  $\theta_1$ ,  $d_2$  and  $\theta_3$  and other necessary parameters.

### Exercise iii.8

Draw the reference systems for the 3 joints of the RRP manipulator shown in figure iii.7. Obtain the D-H characteristic parameters for this robot and the three transformation

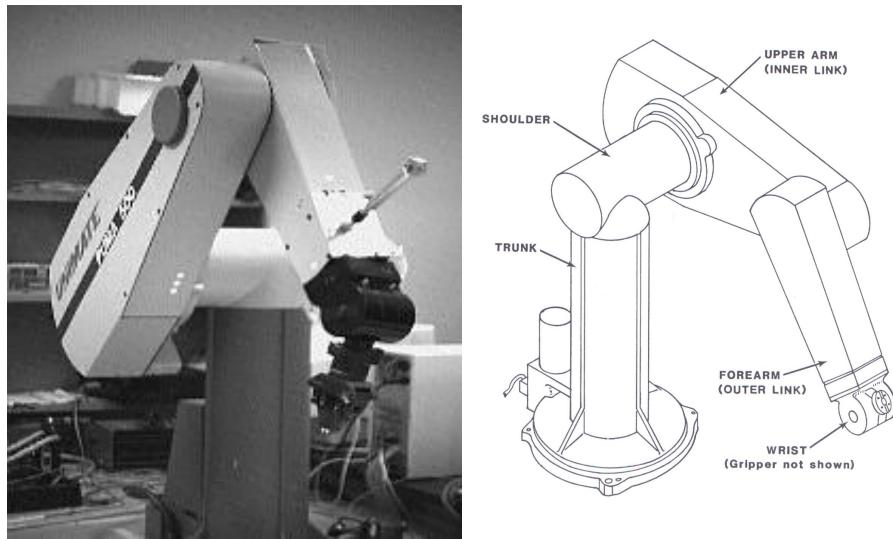


Figure iii.4: PUMA manipulator.

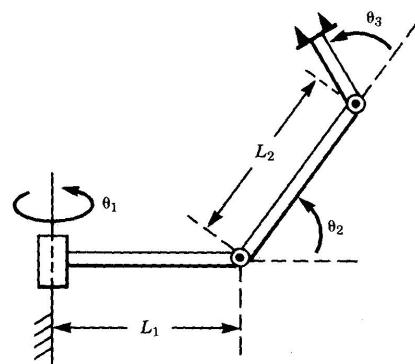


Figure iii.5: 3R non-planar robot.

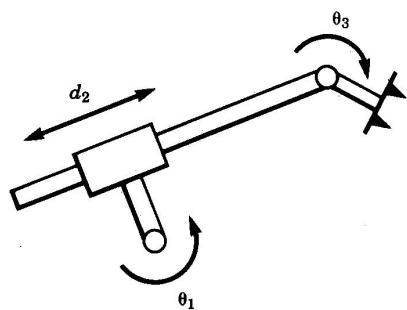


Figure iii.6: Planar robot RPR.

matrices  ${}^0T_1$ ,  ${}^1T_2$ , and  ${}^2T_3$  that define the forward kinematics of the manipulator.

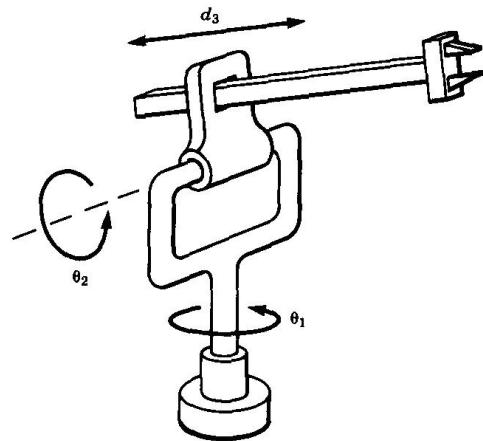


Figure iii.7: RRP manipulator.

### Exercise iii.9

Draw the reference systems for the 3 joints of the RRR manipulator shown in figure iii.8. Obtain the geometric characteristic parameters for this robot and the three transformation matrices  ${}^0T_1$ ,  ${}^1T_2$ , and  ${}^2T_3$  that define the forward kinematics of the manipulator.

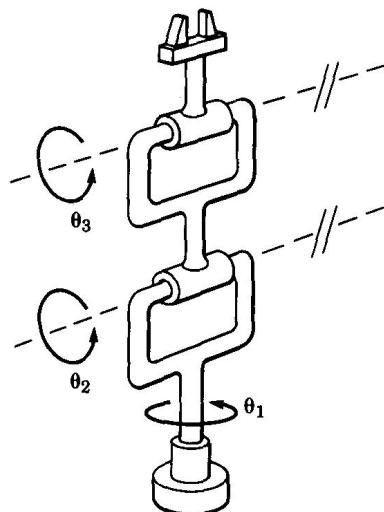


Figure iii.8: RRR manipulator.

**Exercise iii.10**

Solve the forward kinematic problem of the SCARA manipulator shown in figure iii.9:

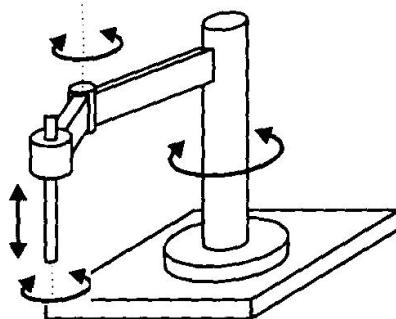


Figure iii.9: SCARA manipulator.

**Exercise iii.11**

Solve the forward kinematic problem of a cylindrical robot with 4DoF in figure iii.10.

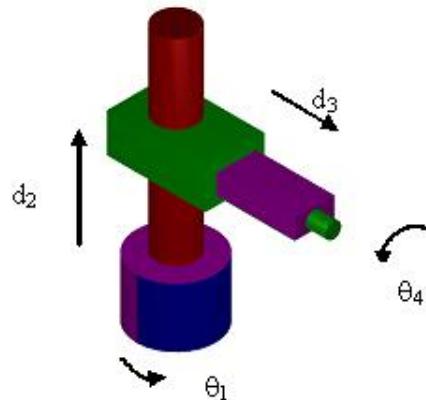


Figure iii.10: Cylindrical robot.

**Exercise iii.12**

Draw the reference systems for the 6 joints of the 6R industrial robot shown (MALIBA manipulator) in figure iii.11. Obtain the D-H characteristic parameters for this robot and the six transformation matrices  $i^{-1}T_i$  that define the forward kinematics of the manipulator.

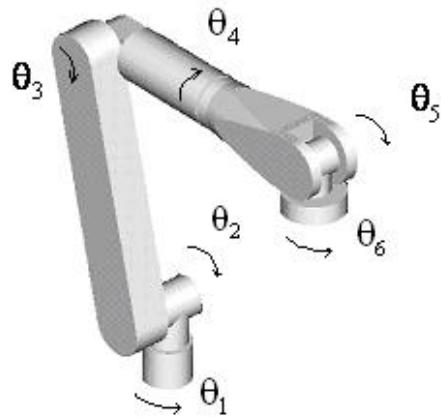


Figure iii.11: MALIBA manipulator.

### **Exercise iii.13**

Solve the forward kinematics of the SCORBOT robot with 5 DoF shown in figure iii.12 using the Denavit and Hartenberg method. Obtain its characteristic parameters and the transformation matrices  $i^{-1}T_i$ , including the one related to the end effector of the robot.

### **Exercise iii.14**

Solve the forward kinematics of the RT3300 robot of 4 DoF shown in figure iii.13 using the Denavit and Hartenberg method. Obtain its characteristic parameters and the transformation matrices ( ${}^0T_1$ ,  ${}^1T_2$ ,  ${}^2T_3$  and  ${}^3T_4$ ).

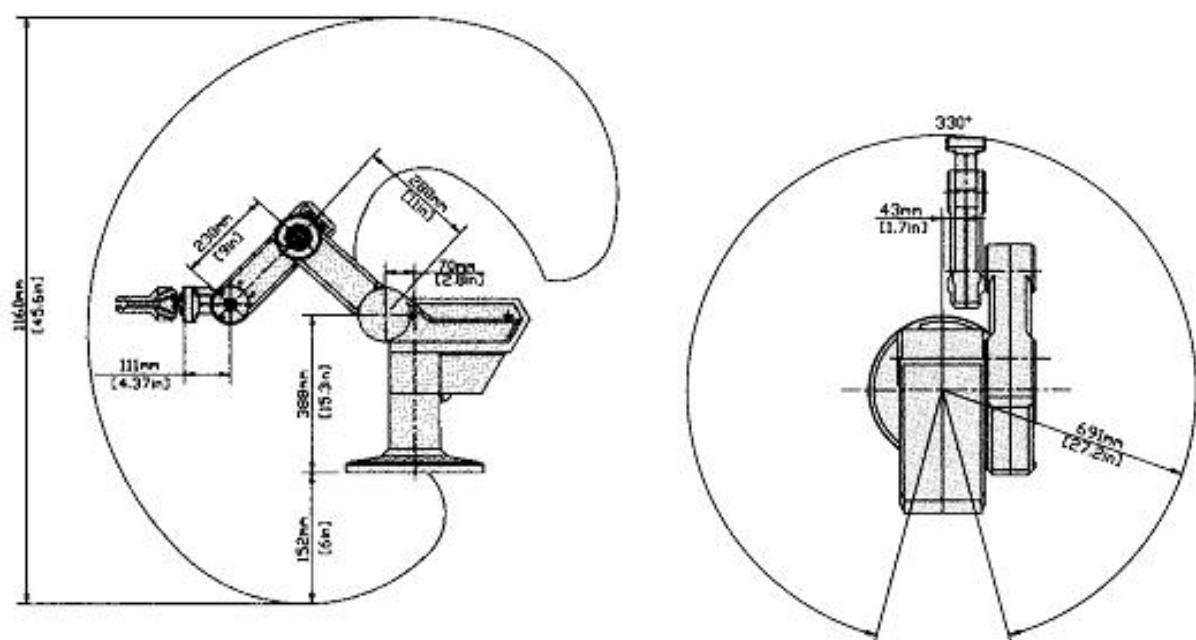


Figure iii.12: SCORBOT robot.

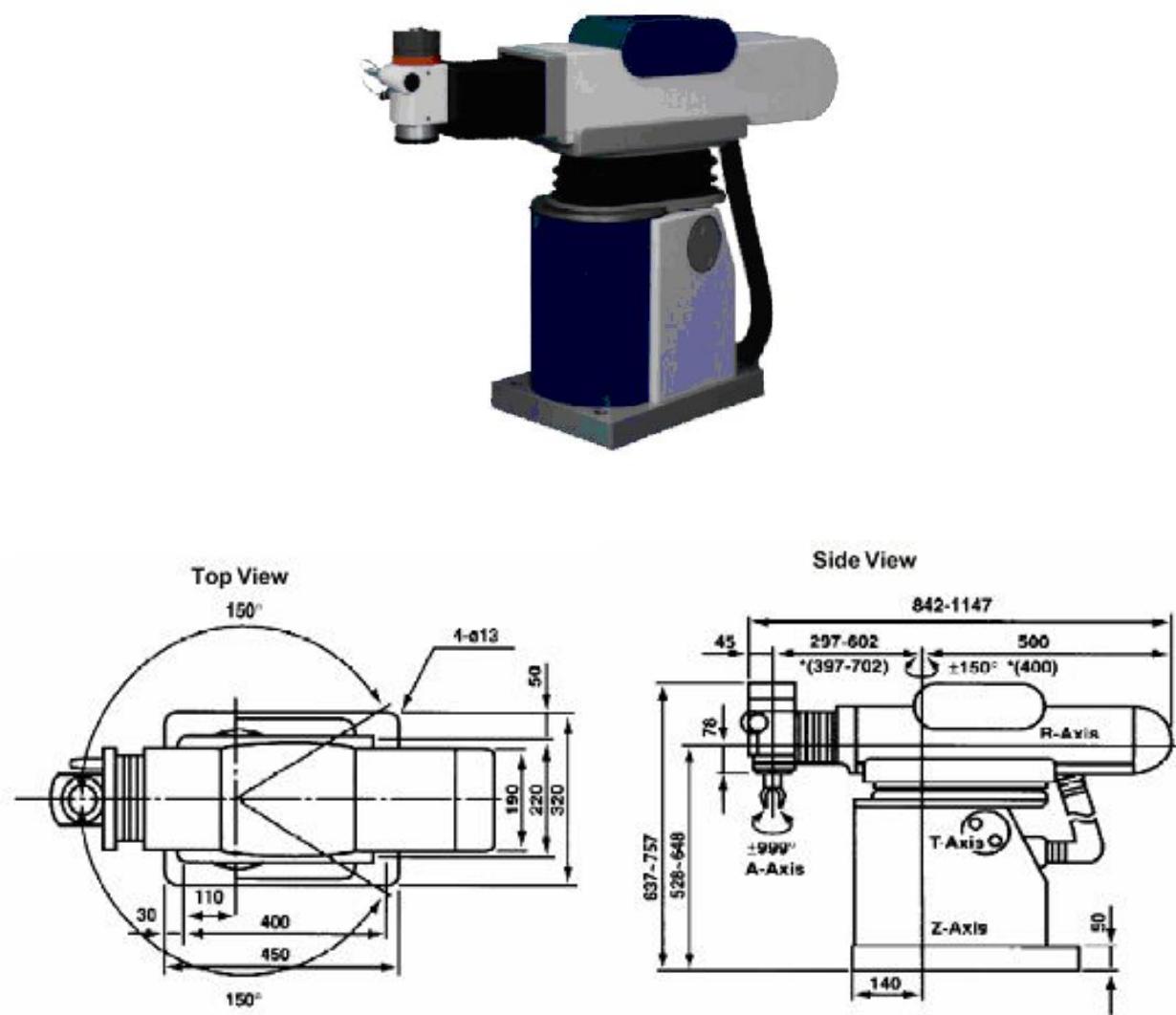


Figure iii.13: RT3300 robot.

## Solutions

### Solution to exercise iii.1

There are two ways to go from  $\{B\}$  to  $\{G\}$ :

- 1<sup>st</sup> way:  ${}^B T_W \cdot {}^W T_T$
- 2<sup>nd</sup> way:  ${}^B T_S \cdot {}^S T_G \cdot {}^G T_T$

equating between both expressions, it yields:

$${}^B T_W \cdot {}^W T_T = {}^B T_S \cdot {}^S T_G \cdot {}^G T_T$$

When  $\{G\}$  and  $\{T\}$  are coincident, we can say that  ${}^G T_T$  is the identity matrix. This situation happens when the tool is in the hole. Thus, we have:

$${}^W T_T = {}^B T_W^{-1} \cdot {}^B T_S \cdot {}^S T_G$$

where:  ${}^B T_W$  has been measured by the sensors, and  ${}^B T_S$  and  ${}^S T_G$  are known, so we can calculate  ${}^W T_T$ .

### Solution to exercise iii.2

The first issue we have to deal with is to set the reference system for the end effector (figure iii.14) and make the Denavit and Hartenberg table for this joint:

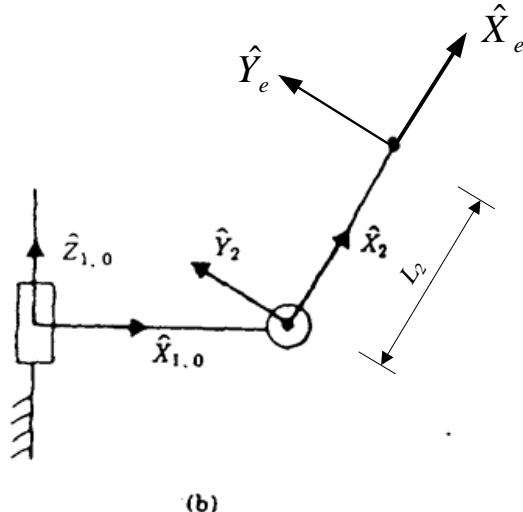


Figure iii.14: D-H coordinate systems for a 2-DoF manipulator.

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$link i-1 \rightarrow$					
$link i$		0	0	$L_2$	0
$2 \rightarrow e$					

You can get the complete solution with Matlab® using the following coding:

```

syms c1 s1 c2 s2 l1 l2

T02=[c1*c2 -c1*s2 s1 l1*c1;...
      s1*c2 -s1*s2 -c1 l1*s1;...
      s2 c2 0 0;...
      0 0 0 1]
T2e= [l2; 0; 0; 1]

T=T02*T2e

```

### Solution to exercise iii.3

After setting the different configurations of the reference systems we obtain from each figure (figures iii.15 and iii.16) the corresponding four Denavit-Hartenberg tables.

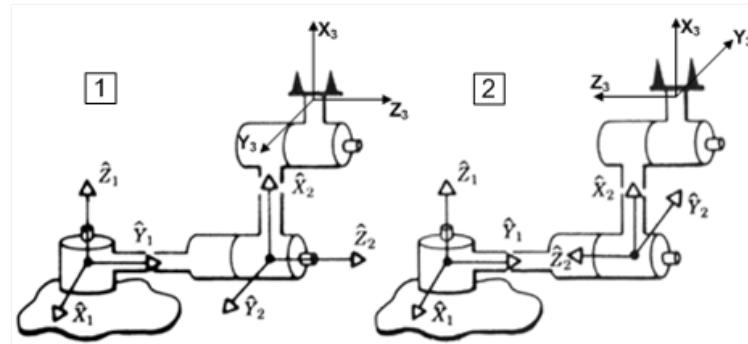


Figure iii.15: 3-DoF manipulator 1<sup>st</sup> and 2<sup>nd</sup> solutions.

First configuration:

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$link i-1 \rightarrow$					
$link i$		0	-90°	$L_1$	-90°
$1 \rightarrow 2$		$L_2$	0	0	0
$2 \rightarrow 3$					

Second configuration:

DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$link\ i-1 \rightarrow link\ i$				
1 → 2	0	90°	- $L_1$	90°
2 → 3	$L_2$	0	0	0

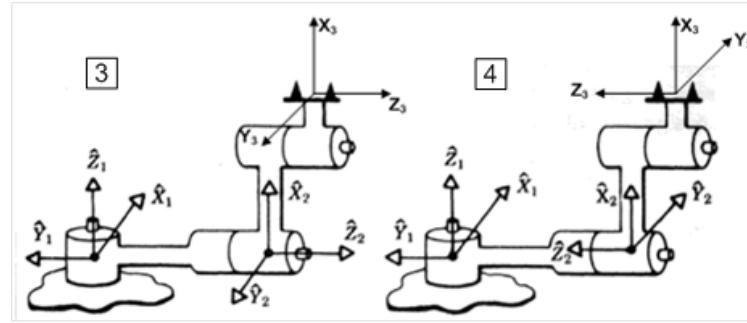


Figure iii.16: 3-DoF manipulator 3<sup>rd</sup> and 4<sup>th</sup> solutions.

Third configuration:

DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$link\ i-1 \rightarrow link\ i$				
1 → 2	0	90°	$L_1$	90°
2 → 3	$L_2$	0	0	0

Fourth configuration:

DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$link\ i-1 \rightarrow link\ i$				
1 → 2	0	-90°	- $L_1$	-90°
2 → 3	$L_2$	0	0	0

### Solution to exercise iii.4

The robot that gives that Denavit and Hartenberg table should be like it is shown in figure iii.17.

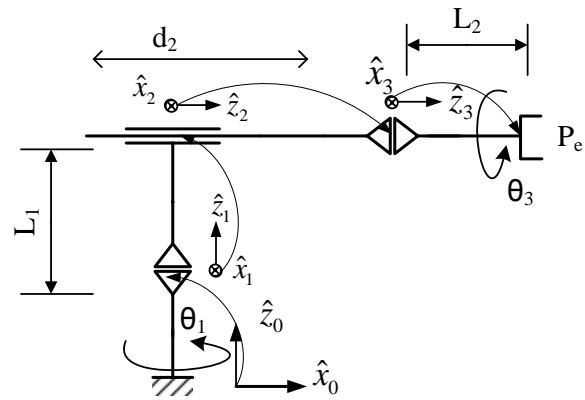


Figure iii.17: Solution to an unknown robot.

### Solution to exercise iii.5

If we develop the PUMA robot of the figure and do the Denavit and Hartenberg table we obtain figure iii.18.

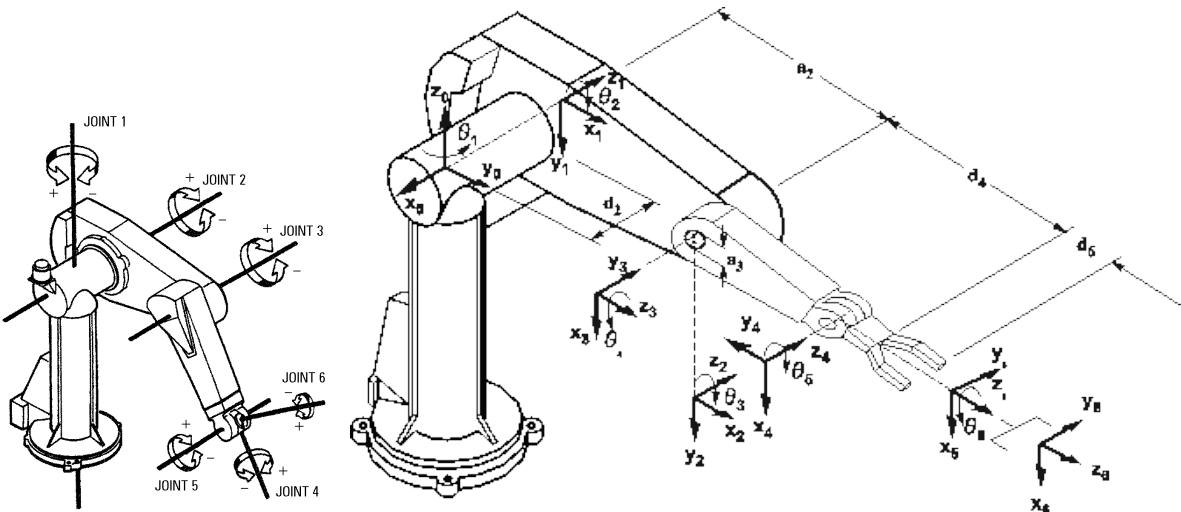


Figure iii.18: PUMA DoFs and D-H coordinate systems.

Denavit-Hartenberg table:

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
<i>link <math>i-1 \rightarrow</math></i>	<i>link <math>i</math></i>				
$0 \rightarrow 1$		0	$-90^\circ$	0	$90^\circ$
$1 \rightarrow 2$		$a_2$	0	$d_2$	0
$2 \rightarrow 3$		$a_3$	$90^\circ$	0	$90^\circ$
$3 \rightarrow 4$		0	$-90^\circ$	$d_4$	0
$4 \rightarrow 5$		0	$90^\circ$	0	0
$5 \rightarrow e$		0	0	$d_6$	0

So, the distance between the reference systems {0} and {6} is:

$$p_x = a_2 + a_3$$

$$p_y = d_3$$

$$p_z = a_1 - d_4$$

### Solution to exercise iii.6

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.19), do the Denavit and Hartenberg table and then obtain the transformation matrices.

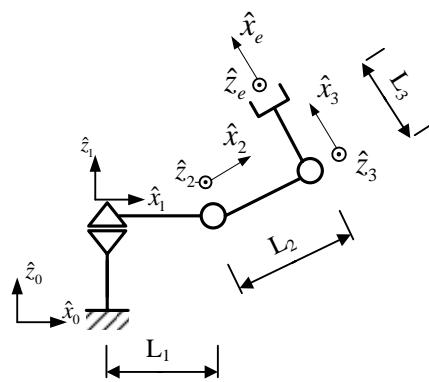


Figure iii.19: D-H coordinate systems for a 3R non-planar robot.

Denavit-Hartenberg table:

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
<i>link i-1</i>	$\rightarrow$				
<i>link i</i>					
$0 \rightarrow 1$		0	0	0	$\theta_1$
$1 \rightarrow 2$		$L_1$	$90^\circ$	0	$\theta_2$
$2 \rightarrow 3$		$L_2$	0	0	$\theta_3$
$3 \rightarrow e$		$L_3$	0	0	0

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 theta_2 theta_3 L_1 L_2
syms thetadot_1 thetadot_2 thetadot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T12=[1 0 0 L_1;...
      0 0 -1 0;...
      0 1 0 0;...
      0 0 0 1]*....
[cos(theta_2) -sin(theta_2) 0 0;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T23=[cos(theta_3) -sin(theta_3) 0 L_2;...
      sin(theta_3) cos(theta_3) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T03=simple(T01*T12*T23)

```

### Solution to exercise iii.7

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.20), do the Denavit and Hartenberg table and then obtain the transformation matrices.

Denavit-Hartenberg table:

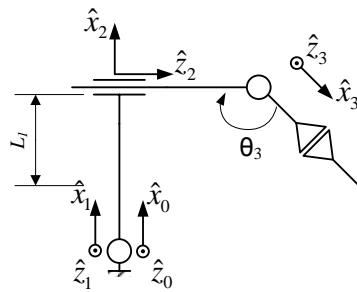


Figure iii.20: D-H coordinate systems for a planar robot RPR.

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
<i>link <math>i-1 \rightarrow</math></i>	<i>link <math>i</math></i>				
$0 \rightarrow 1$		0	0	0	$\theta_1$
$1 \rightarrow 2$		$L_1$	$90^\circ$	$d_2$	0
$2 \rightarrow 3$		0	$-90^\circ$	0	$\theta_3$

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 d_2 theta_3 L_1 L_3
syms thetadot_1 ddot_2 thetadot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T12=[1 0 0 L_1;...
      0 0 -1 0;...
      0 1 0 0;...
      0 0 0 1]*...
      [-1 0 0 0;...
      0 -1 0 0;...
      0 0 1 d_2;...
      0 0 0 1]

T23=[1 0 0 0;...
      0 0 -1 0;...
      0 1 0 0;...
      0 0 0 1]...
      *[cos(theta_3) -sin(theta_3) 0 0;...
      sin(theta_3) cos(theta_3) 0 0;...
      0 0 1 0;...
      0 0 0 1]

```

```

T3e=[1 0 0 L_3; ...
0 1 0 0; ...
0 0 1 0; ...
0 0 0 1]

T0e=simple(T01*T12*T23*T3e)

```

### Solution to exercise iii.8

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.21), do the Denavit and Hartenberg table and then obtain the transformation matrices.

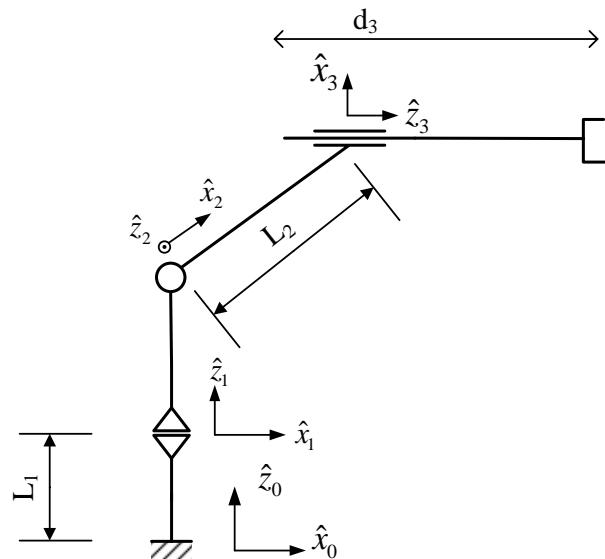


Figure iii.21: D-H coordinate systems for a RRP manipulator.

Denavit-Hartenberg table:

link $i-1 \rightarrow$ link $i$	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$0 \rightarrow 1$	0	0	$L_1$	$\theta_1$	
$1 \rightarrow 2$	0	$90^\circ$	0	$\theta_2$	
$2 \rightarrow 3$	$L_2$	$90^\circ$	$d_3$	0	

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 theta_2 d_3 L_1 L_2
syms thetadot_1 thetadot_2 ddot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 L_1;...
      0 0 0 1]

T12=[cos(theta_2) -sin(theta_2) 0 0;...
      0 0 -1 0;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 0 1]

T23=[1 0 0 L_2;...
      0 0 -1 -d3;...
      0 1 0 0;...
      0 0 0 1]

T03=simple(T01*T12*T23)

```

### Solution to exercise iii.9

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.22), do the Denavit and Hartenberg table and then obtain the transformation matrices.

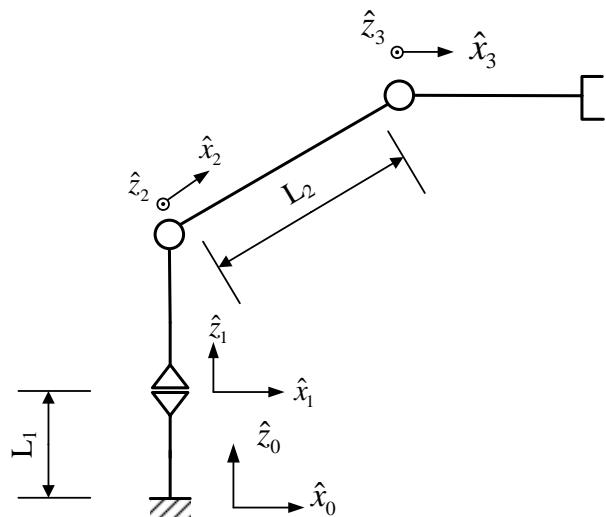


Figure iii.22: D-H coordinate systems for a RRR manipulator.

Denavit-Hartenberg table:

		DH par.			
<i>link i-1 → link i</i>		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
0 → 1		0	0	$L_1$	$\theta_1$
1 → 2		0	$90^\circ$	0	$\theta_2$
2 → 3		$L_2$	0	0	$\theta_3$

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 theta_2 theta_3 L_1 L_2
syms thetadot_1 thetadot_2 thetadot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 L_1;...
      0 0 0 1]

T12=[cos(theta_2) -sin(theta_2) 0 0;...
      0 0 -1 0;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 0 1]

T23=[cos(theta_3) -sin(theta_3) 0 L_2;...
      sin(theta_3) cos(theta_3) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T03=simple(T01*T12*T23)

```

### Solution to exercise iii.10

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.23), do the Denavit and Hartenberg table and then obtain the transformation matrices.

Denavit-Hartenberg table:

		DH par.			
<i>link i-1 → link i</i>		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
0 → 1		0	0	$L_0$	$\theta_1$
1 → 2		$L_1$	0	0	$\theta_2$
2 → 3		$L_2$	0	$-d_3$	0
3 → 4		0	0	0	$\theta_4$
4 → e		0	0	$L_4$	0

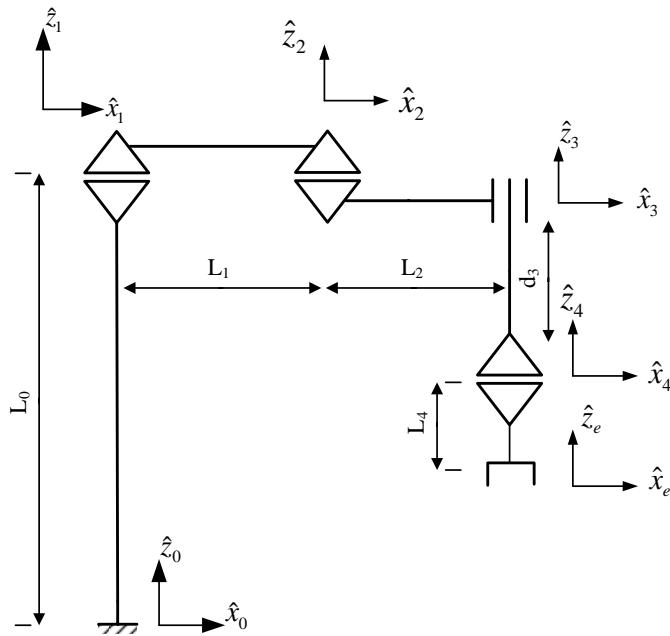


Figure iii.23: D-H coordinate systems for a SCARA manipulator.

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 theta_2 d_3 theta_4
syms L_0 L_1 L_2 L_3 L_4
syms thetadot_1 thetadot_2
syms ddot_3 thetadot_4

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 1 L_0;0 0 0 1]

T12=[cos(theta_2) -sin(theta_2) 0 L_1;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T23=[1 0 0 L_2;...
      0 1 0 0;...
      0 0 1 -d_3;...
      0 0 0 1]

T34=[cos(theta_4) -sin(theta_4) 0 0;...
      sin(theta_4) cos(theta_4) 0 0;...
      0 0 1 0;...
      0 0 0 1]

```

```
0 0 0 1]
```

```
T4e=[1 0 0 0; ...
0 1 0 0; ...
0 0 1 L_4; ...
0 0 0 1]

T04=simple(T01*T12*T23*T34*T4e)
```

### Solution to exercise iii.11

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.24), do the Denavit and Hartenberg table and then obtain the transformation matrices.

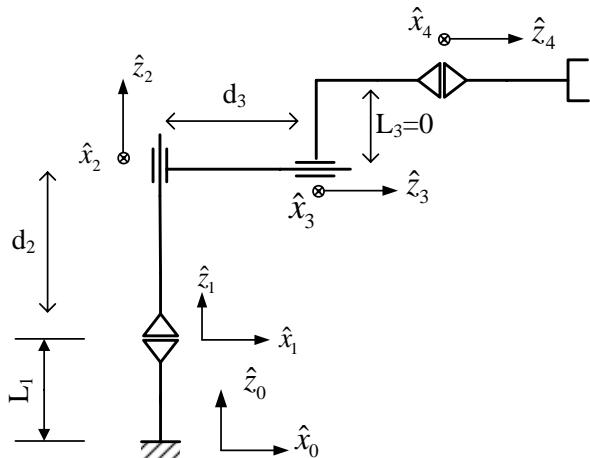


Figure iii.24: D-H coordinate systems for a cylindrical manipulator.

Denavit-Hartenberg table:

<i>link</i> $i-1 \rightarrow$ <i>link</i> $i$	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$0 \rightarrow 1$		0	0	0	$\theta_1$
$1 \rightarrow 2$		0	0	$d_2$	$90^\circ$
$2 \rightarrow 3$		0	$90^\circ$	$d_3$	0
$3 \rightarrow 4$		0	0	0	$\theta_4$

You can get the complete solution with Matlab® using the following coding:

```
syms theta_1 theta_4 d_2 d_3
```

```

syms thetadot_1 thetadot_4
syms ddot_2 ddot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T12=[1 0 0 0;...
      0 0 -1 0;...
      0 1 0 d_2;...
      0 0 0 1]

T23=[1 0 0 0;...
      0 0 -1 -d_3;...
      0 1 0 0;...
      0 0 0 1]

T34=[cos(theta_4) -sin(theta_4) 0 0;...
      sin(theta_4) cos(theta_4) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T04=simple(T01*T12*T23*T34)

```

### Solution to exercise iii.12

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.25), do the Denavit and Hartenberg table and then obtain the transformation matrices.

Denavit-Hartenberg table:

<i>link i-1 → link i</i>	DH par.			
	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$0 \rightarrow 1$	0	0	$L_0 + L_1$	$\theta_1$
$1 \rightarrow 2$	0	$90^\circ$	0	$\theta_2$
$2 \rightarrow 3$	$L_2$	0	0	$\theta_3$
$3 \rightarrow 4$	0	$90^\circ$	$L_3 + L_4$	$\theta_4$
$4 \rightarrow 5$	0	$90^\circ$	0	$\theta_5$
$5 \rightarrow 6$	0	$90^\circ$	$L_5$	$\theta_6$

You can get the complete solution with Matlab® using the following coding:

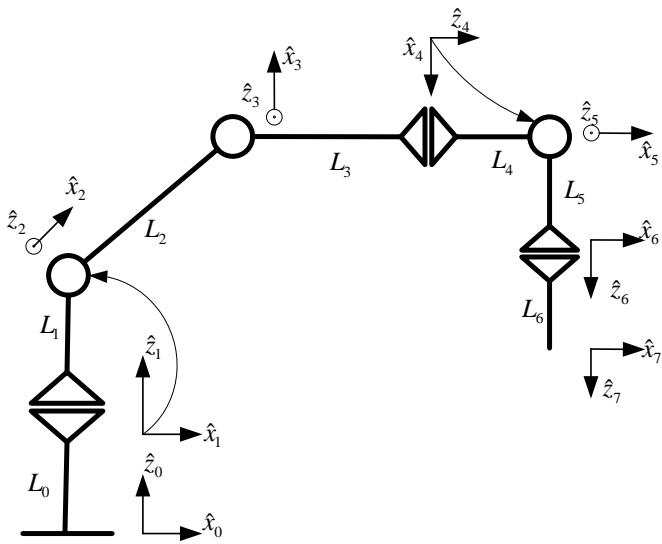


Figure iii.25: D-H coordinate systems for MALIBA manipulator.

```

syms theta_1 theta_2 theta_3 theta_4 theta_5 theta_6
syms L_0 L_1 L_2 L_3 L_4 L_5
syms thetadot_1 thetadot_2 thetadot_3
syms thetadot_4 thetadot_5 thetadot_6

T01=[cos(theta_1) -sin(theta_1) 0 0;...
sin(theta_1) cos(theta_1) 0 0;...
0 0 1 L_0+L_1;...
0 0 0 1]

T12=[cos(theta_2) -sin(theta_2) 0 0;...
0 0 -1 0;...
sin(theta_2) cos(theta_2) 0 0;...
0 0 0 1]

T23=[cos(theta_3) -sin(theta_3) 0 L_2;...
sin(theta_3) cos(theta_3) 0 0;...
0 0 1 0;...
0 0 0 1]

T34=[cos(theta_4) -sin(theta_4) 0 0;...
0 0 -1 -L_3;...
sin(theta_4) cos(theta_4) 0 0;...
0 0 0 1]

T45=[cos(theta_5) -sin(theta_5) 0 0;...
0 0 -1 0;...
sin(theta_5) cos(theta_5) 0 0;...
0 0 0 1]

```

0 0 0 1]

```
T56=[cos(theta_6) -sin(theta_6) 0 0;...
0 0 -1 -L_5;...
sin(theta_6) cos(theta_6) 0 0;...
0 0 0 1]
```

### Solution to exercise iii.13

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.26), do the Denavit and Hartenberg table (looking the dimension lines included in the dimension plots of the robot) and then obtain the transformation matrices.

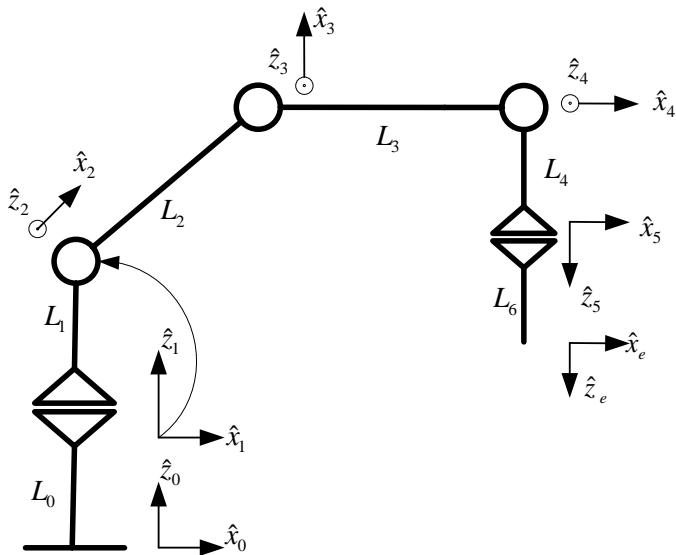


Figure iii.26: D-H coordinate systems for SCORBOT robot.

Denavit-Hartenberg table:

DH par. link $i-1 \rightarrow$ link $i$	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$0 \rightarrow 1$	70	0	$L_0 + L_1$	$\theta_1$
$1 \rightarrow 2$	0	$90^\circ$	0	$\theta_2$
$2 \rightarrow 3$	$L_2$	0	0	$\theta_3$
$3 \rightarrow 4$	$L_3$	0	43	$\theta_4$
$4 \rightarrow 5$	0	$90^\circ$	$L_4$	$\theta_5$
$5 \rightarrow e$	0	0	$L_5$	0

Where:

$$L_0 + L_1 = 388\text{mm}.$$

$$L_2 = 280\text{mm}.$$

$$L_3 = 43\text{mm}.$$

$$L_4 = 230\text{mm}.$$

$$L_5 = 111\text{mm}.$$

### Solution to exercise iii.14

To solve this exercise we first have to draw the robot, after that we will have to set the reference systems (figure iii.27), do the Denavit and Hartenberg table (taking into account the dimension lines included in robot's dimension plot) and then obtain the transformation matrices.

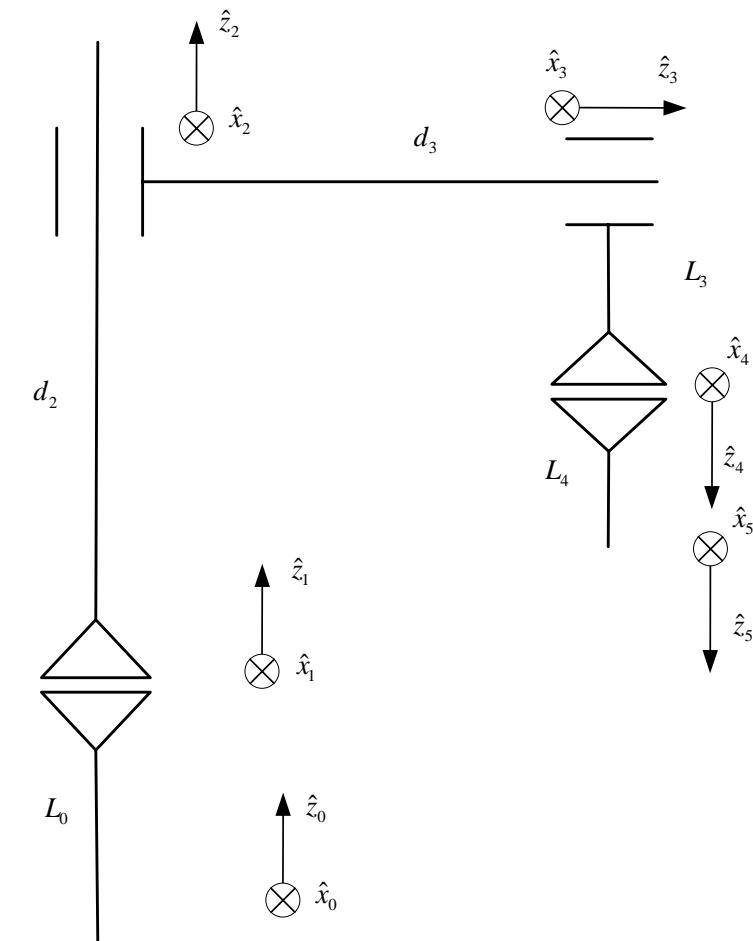


Figure iii.27: D-H coordinate systems for RT3300 robot.

Denavit-Hartenberg table:

$link\ i-1 \rightarrow$ $link\ i$	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$0 \rightarrow 1$		0	0	$L_0$	$\theta_1$
$1 \rightarrow 2$		0	0	$d_2$	0
$2 \rightarrow 3$		0	$90^\circ$	$d_3$	0
$3 \rightarrow 4$		0	$90^\circ$	$L_0$	$\theta_4$
$4 \rightarrow e$		0	0	$L_5$	0

## Exercises iv

# Inverse Kinematics

---

### Exercise iv.1

In a 3R planar robot, given the desired position and orientation for the hand, we know that there are two possible groups of values that the joint angles can take. Deduce how many solutions there will be if we add another joint to the manipulator, in such way that the arm continues to be planar.

### Exercise iv.2

Calculate the reachable workspace by the RP manipulator shown in figure iv.1. Do not think only of the points that can be reached, but also of the orientation from which those points are reachable. In other words, give the general expression for the transformation matrix that defines the position and the orientation desired for system {2} with respect to the base in function of the Cartesian coordinates that the origing points of system {2} can cover. What is the workspace of the manipulator? (Note: the reference system {0} of the base is coincident with {1} when the robot is at its initial position, i.e.  $\theta = 0$ )

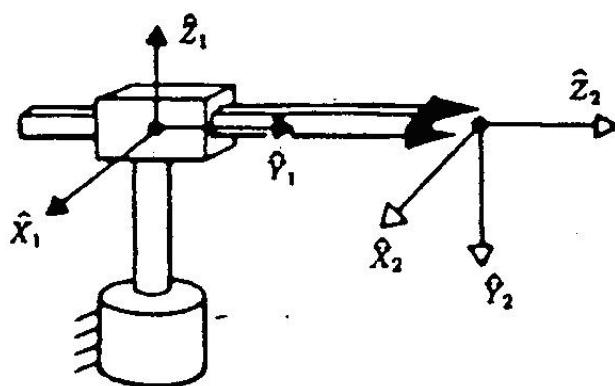


Figure iv.1: RP manipulator.

**Exercise iv.3**

Given the description of the reference system  $\{i\}$  of the  $i^{th}$  joint of a manipulator in relation to the  $\{i-1\}$  system (matrix  ${}^{i-1}T_i$ ), obtain the expressions for the four Denavit and Hartenberg parameters:  $a_{i-1}$ ,  $\alpha_{i-1}$ ,  $d_i$  and  $\theta_i$  as a function of the elements of  ${}^{i-1}T_i$ .

**Exercise iv.4**

Solve the inverse kinematics problem for the RPR manipulator in figure iv.2.

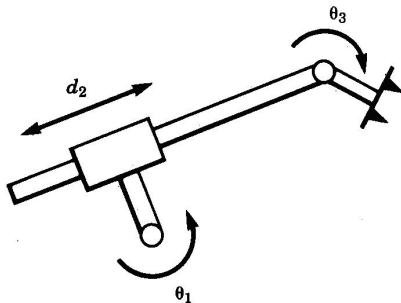


Figure iv.2: RPR manipulator.

**Exercise iv.5**

Figure iv.3 shows a planar arm with two joints. The joint angles can vary between the limits  $0 \leq \theta_1 \leq 180^\circ$  and  $-90^\circ \leq \theta_2 \leq 180^\circ$ . Draw approximately the workspace reachable by this manipulator.

**Exercise iv.6**

Given the SCARA robot in figure, we want to move the end-effector to the position given by  $[0.7 \ 0.5 \ -0.5]^T$ , with respect to the coordinate system of the base. Which values will the variables  $\theta_1$ ,  $\theta_2$  and  $d_3$  need to achieve this? ( $L_0 = -1m$ ,  $L_0 = 0.5m$  and  $L_0 = 0.4m$ ).

**Exercise iv.7**

Given the location of the end-effector of an anthropomorphic robot  $P_e = [P_{e_x} \ P_{e_y} \ P_{e_z}]^T$  and its orientation, calculate the inverse kinematics for its three first degrees of freedom.

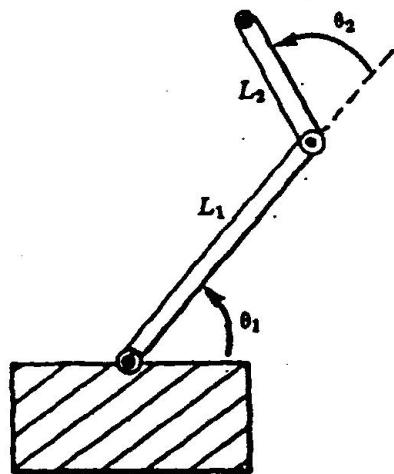


Figure iv.3: Planar arm with two joints.

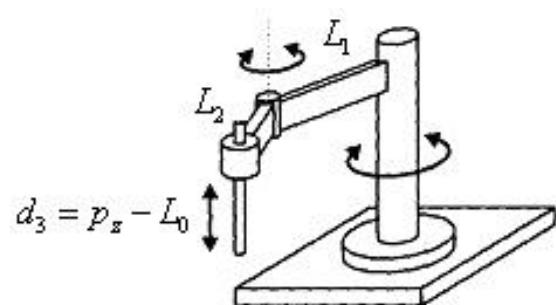


Figure iv.4: SCARA robot.

$${}^0 R_e = \begin{bmatrix} \hat{x}_{e_x} & \hat{y}_{e_x} & \hat{z}_{e_x} \\ \hat{x}_{e_y} & \hat{y}_{e_y} & \hat{z}_{e_y} \\ \hat{x}_{e_z} & \hat{y}_{e_z} & \hat{z}_{e_z} \end{bmatrix}$$

**Exercise iv.8**

Solve the inverse kinematics for a spherical robot of three degrees of freedom.

## Solutions

### Solution to exercise iv.1

If we add another joint, we will get a planar robot with redundant degrees of freedom. That is way there will be infinite solutions to this exercise.

### Solution to exercise iv.2

In this exercise we are asked to do the inverse kinematics. The first step is to solve the forward kinematics problem to obtain the  ${}^0T_3 = {}^0T_e$  matrix and, after that, obtain the parameters needed to do the inverse kinematics.

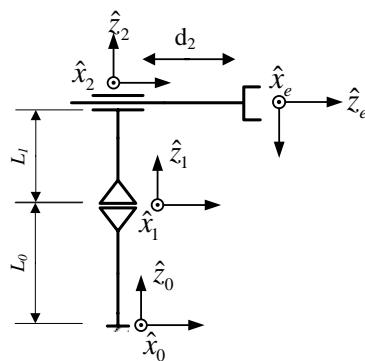


Figure iv.5: D-H coordinate systems of a RP manipulator.

Denavit-Hartenberg table:

link $i-1 \rightarrow$ link $i$	DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$0 \rightarrow 1$		0	0	$L_0$	$\theta_1$
$1 \rightarrow 2$		0	0	$L_1$	0
$2 \rightarrow e$		0	$-90^\circ$	$d_2$	0

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 L_0 L_1 d_2
syms thetadot_1 ddot_2

T01=[cos(theta_1) -sin(theta_1) 0 0; ...
sin(theta_1) cos(theta_1) 0 0; ...
0 0 1 L_0; ...

```

```
0 0 0 1]
```

```
T12=[1 0 0 0;...
0 1 0 0;...
0 0 1 L_1;...
0 0 0 1]
```

```
T2e=[1 0 0 0;...
0 0 1 d_2;...
0 -1 0 0;...
0 0 0 1]
```

```
T04=simple(T01*T12*T2e)
```

And now we can obtain the inverse kinematics parameters:

$$L_0 + L_1 = P_{ez}$$

$$d_2 = \sqrt{P_{ex}^2 + P_{ey}^2}$$

$$\theta_1 = \arctan \frac{-P_{ex}}{P_{ey}} \rightarrow \cos \theta_1 = \frac{P_{ey}}{\sqrt{P_{ex}^2 + P_{ey}^2}} \rightarrow \sin \theta_1 = \frac{-P_{ex}}{\sqrt{P_{ex}^2 + P_{ey}^2}}$$

$${}^0T_e = \begin{bmatrix} \frac{P_{ey}}{\sqrt{P_{ex}^2 + P_{ey}^2}} & 0 & \frac{-P_{ex}}{\sqrt{P_{ex}^2 + P_{ey}^2}} & P_{ex} \\ \frac{P_{ex}}{\sqrt{P_{ex}^2 + P_{ey}^2}} & 0 & \frac{-P_{ey}}{\sqrt{P_{ex}^2 + P_{ey}^2}} & P_{ey} \\ 0 & 1 & 0 & P_{ez} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_1 & 0 & -\sin \theta_1 & -d_2 \sin \theta_1 \\ \sin \theta_1 & 0 & \cos \theta_1 & d_2 \cos \theta_1 \\ 0 & 1 & 0 & L_1 + L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Solution to exercise iv.3

Let's make a general Denavit-Hartenberg table and obtain the transformation matrix. We will rename the terms of this matrix in order to make the expressions simpler:

DH par.	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$link i-1 \rightarrow$				
$link i$				
$\{i-1\} \rightarrow \{i\}$	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$

$${}^{i-1}T_i = \text{trans}(a_{i-1}, \hat{x}_{i-1}) \cdot \text{rot}(\alpha_{i-1}, \hat{x}_{i-1}) \cdot \text{trans}(d_i, \hat{z}_i) \cdot \text{rot}(\theta_i, \hat{z}_i) =$$

$$\begin{aligned}
&= \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \cos \alpha_{i-1} \cdot \sin \theta_i & \cos \alpha_{i-1} \cdot \cos \theta_i & -\sin \alpha_{i-1} & \sin \alpha_{i-1} \cdot d_i \\ \sin \alpha_{i-1} \cdot \sin \theta_i & \sin \alpha_{i-1} \cdot \cos \theta_i & \cos \alpha_{i-1} & \cos \alpha_{i-1} \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\
&= \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\theta_i &= \arctan \left( \frac{\sin \theta_i}{\cos \theta_i} \right) = \arctan \left( \frac{-r_{12}}{r_{11}} \right) \\
\alpha_{i-1} &= \arctan \left( \frac{\sin \alpha_{i-1}}{\cos \alpha_{i-1}} \right) = \arctan \left( \frac{-r_{23}}{r_{33}} \right) \\
a_{i-1} &= r_{14} \\
d_i &= \sqrt{r_{24}^2 + r_{34}^2}
\end{aligned}$$

### Solution to exercise iv.4

As we did in previous exercises, to calculate the inverse kinematics of this robot, we must first calculate the forward kinematics problem to obtain the  ${}^0T_e$  matrix and, after that, obtain the parameters needed to do the inverse kinematics.

You can get the complete forward kinematics solution from the forward kinematics exercise 7.

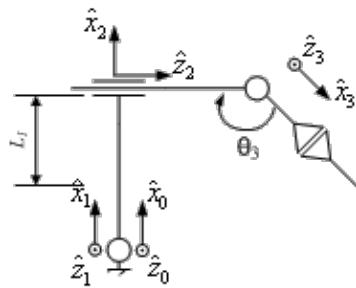


Figure iv.6: D-H coordinate systems of a RPR manipulator.

Denavit-Hartenberg table:

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
<i>link <math>i-1 \rightarrow</math></i>	<i>link <math>i</math></i>				
$0 \rightarrow 1$		0	0	0	$\theta_1$
$1 \rightarrow 2$		$L_1$	$90^\circ$	$d_2$	0
$2 \rightarrow 3$		0	$-90^\circ$	0	$\theta_3$

And now we can obtain the inverse kinematics parameters (figure iv.7):

$$\theta_1 = \alpha + \beta$$

$$\theta_1 = \arctan \frac{P_{ey}}{P_{ex}} \pm \arccos \left( \frac{L_1}{\sqrt{P_{ex}^2 + P_{ey}^2}} \right)$$

$$d_2 = \sqrt{\left( \sqrt{P_{ex}^2 + P_{ey}^2} \right)^2 - L_1^2}$$

$$\theta_3 = \Phi - \theta_1$$

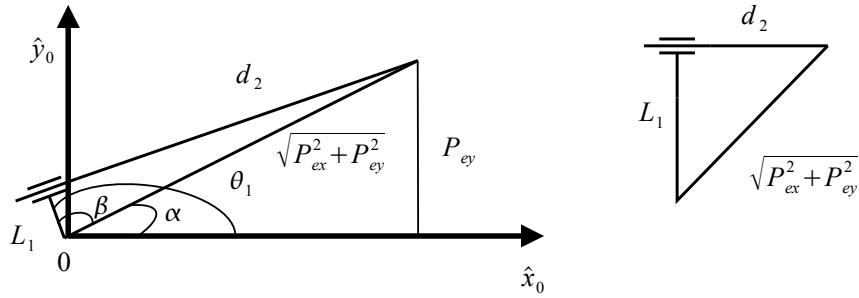


Figure iv.7: Inverse kinematic equations of a RPR manipulator.

On the other hand, we know the expression for the homogeneous transformation between  $\{0\}$  and  $\{e\}$ :

$${}^0T_3 = \begin{bmatrix} \cos\Phi & -\sin\Phi & 0 & p_{ex} \\ \sin\Phi & \cos\Phi & 0 & p_{ey} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, this last matrix should be equated with the forward kinematics expression.

### Solution to exercise iv.5

The workspace reachable by the robot will look like this:

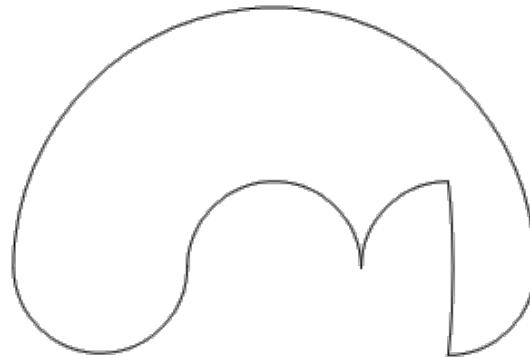


Figure iv.8: Workspace of a planar arm with two joints.

### Solution to exercise iv.6

To calculate the inverse kinematics of this robot, we must first calculate the forward kinematics problem to obtain the  ${}^0T_e$  matrix and, after that, obtain the parameters needed to do the inverse kinematics.

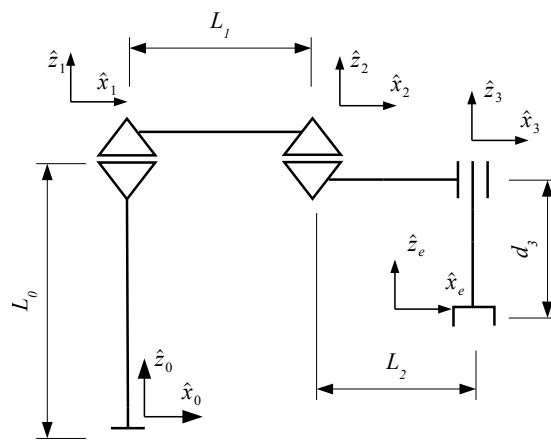


Figure iv.9: DH coordinate systems of a SCARA robot.

Denavit-Hartenberg table:

DH par.		$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
<i>link i-1</i> →	<i>link i</i>				
0 → 1		0	0	$L_0$	$\theta_1$
1 → 2		$L_1$	0	0	$\theta_2$
2 → 3		$L_2$	0	$-d_3$	0

You can get the complete forward kinematics solution with Matlab® using the following coding:

```

syms theta_1 theta_2 d_3 L_0 L_1 L_2
syms thetadot_1 thetadot_2 ddot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 L_0;...
      0 0 0 1]

T12=[cos(theta_2) -sin(theta_2) 0 L_1;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T23=[1 0 0 L_2;...
      0 1 0 0;...
      0 0 1 -D_3;...
      0 0 0 1]

T03= simple(T01*T12*T23)

```

$${}^0P_e = [\cos_{12} L_2 + \cos \theta_1 L_1 \quad \sin_{12} L_2 + \sin \theta_1 L_1 \quad -d_3 + L_0]^T$$

And now we can obtain the inverse kinematics parameters. Only left-elbow solution has been obtained (figure iv.10):

$$\theta_1 = \alpha + \beta$$

$$\theta_1 = \arctan \frac{P_{4y}}{P_{4x}} \pm \arccos \left( \frac{L_1^2 + R^2 - L_2^2}{2L_1R} \right)$$

$$\alpha = \arctan \frac{P_{4y}}{P_{4x}}$$

To calculate the values for  $\beta$  and  $\gamma$  we use the cosine theorem:

$$L_2^2 = L_1^2 + R^2 - 2L_1R \cos \beta \rightarrow \cos \beta = \left( \frac{L_1^2 + R^2 - L_2^2}{2L_1R} \right) \rightarrow \beta = \arccos \left( \frac{L_1^2 + R^2 - L_2^2}{2L_1R} \right)$$

$$R^2 = L_1^2 + L_2^2 - 2L_1L_2 \cos \gamma \rightarrow \cos \gamma = \frac{L_1^2 + L_2^2 - R^2}{2L_1L_2}$$

$$\begin{aligned} \theta_2 &= 2\pi - \gamma' \\ \gamma' &= \pi - \gamma \end{aligned} \left. \right\} \theta_2 = 2\pi - \pi + \gamma = \pi \pm \arccos \left( \frac{L_1^2 + L_2^2 - R^2}{2L_1L_2} \right)$$

And now we can obtain the values of the unknown parametres:

$${}^0P_e = \begin{bmatrix} L_2 \cos \theta_{12} + L_1 \cos \theta_1 \\ L_2 \sin \theta_{12} + L_1 \sin \theta_1 \\ -d_3 + L_0 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.5 \\ -0.5 \end{bmatrix}$$

then:

$$R = 0.86$$

$$d_1 = -0.5$$

$$\theta_1 = 50.82^\circ \| 20.26^\circ$$

$$\theta_2 = 325.49^\circ \| 34.51^\circ$$

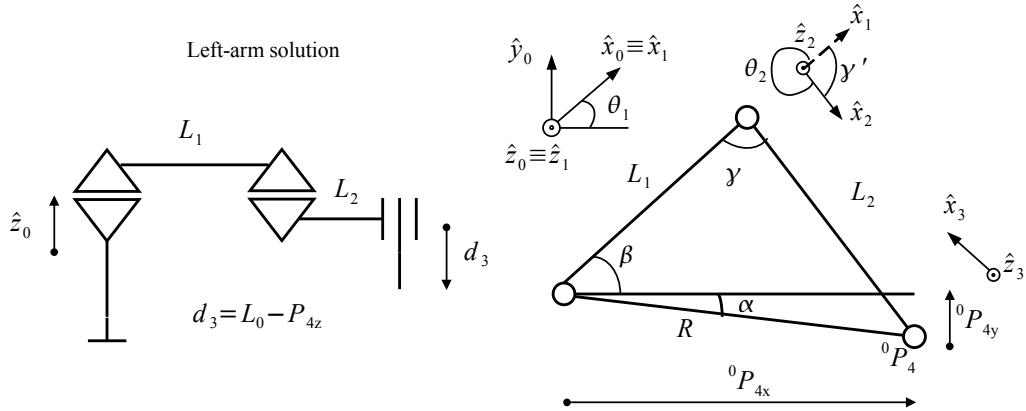


Figure iv.10: Inverse kinematic equations of a SCARA robot.

### Solution to exercise iv.7

To calculate the inverse kinematics of this robot, we must first calculate the forward kinematics problem to obtain the  ${}^0T_e$  matrix and, after that, obtain the parameters needed to do the inverse kinematics.

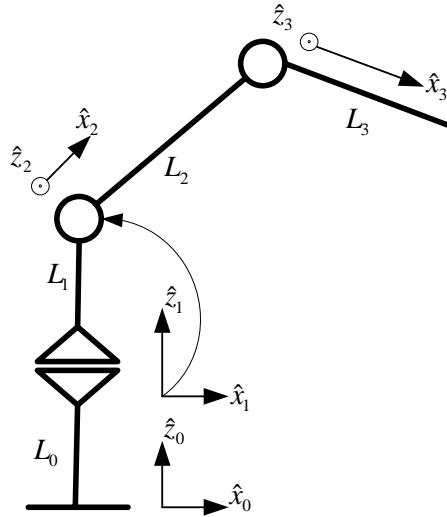


Figure iv.11: D-H coordinate systems of a anthropomorphic robot.

Denavit-Hartenberg table:

		DH par.			
<i>link i-1 →</i>	<i>link i</i>	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
$0 \rightarrow 1$		70	0	$L_0 + L_1$	$\theta_1$
$1 \rightarrow 2$		0	$90^\circ$	0	$\theta_2$
$2 \rightarrow 3$		$L_2$	0	0	$\theta_3$
$3 \rightarrow e$		$L_3$	0	0	0

You can get the complete forward kinematics solution with Matlab® using the following coding:

```

syms theta_1 theta_2 theta_3
syms L_0 L_1 L_2 L_3
syms thetadot_1 thetadot_2 thetadot_3

T01=[cos(theta_1) -sin(theta_1) 0 0; ...
sin(theta_1) cos(theta_1) 0 0; ...
0 0 1 L_0+L_1; ...

```

```
0 0 0 1]
```

```
T12=[cos(theta_2) -sin(theta_2) 0 0;...
```

```
0 0 -1 0;...
```

```
sin(theta_2) cos(theta_2) 0 0;...
```

```
0 0 0 1]
```

```
T23=[cos(theta_3) -sin(theta_3) 0 L_2;...
```

```
sin(theta_3) cos(theta_3) 0 0;...
```

```
0 0 1 0;...
```

```
0 0 0 1]
```

```
T3e=[1 0 0 L_3;...
```

```
0 1 0 0;...
```

```
0 0 0 1]
```

```
T0e=simple(T01*T12*T23*T3e)
```

And now we can obtain the inverse kinematics parameters:

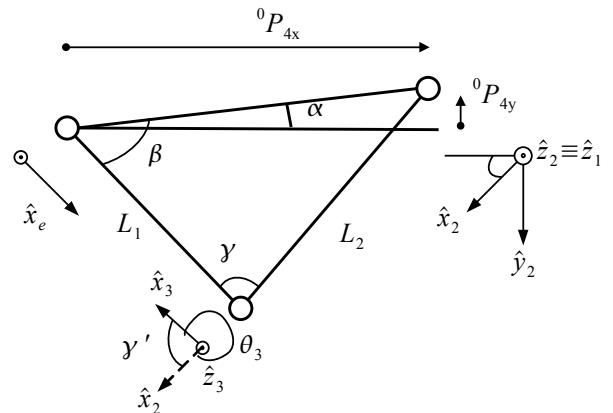
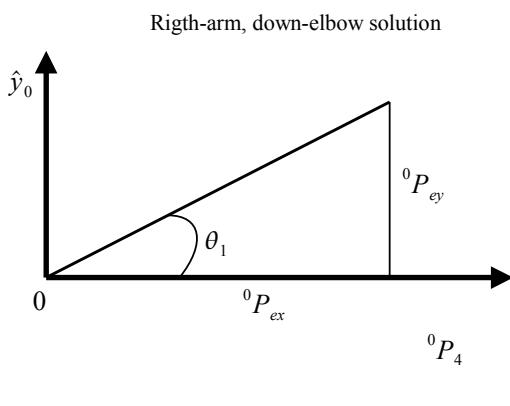


Figure iv.12: Inverse kinematic equations of a anthropomorphic robot.

### Solution to exercise iv.8

To solve this exercise we are going to suppose that we know  ${}^0P_e$  and  ${}^0R_e$ . Thus, we can calculate the inverse kinematics of the robot:

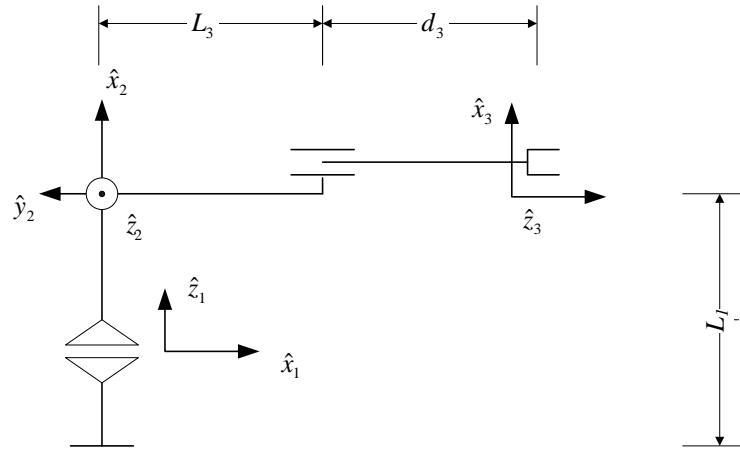


Figure iv.13: D-H coordinate systems of a spherical robot.

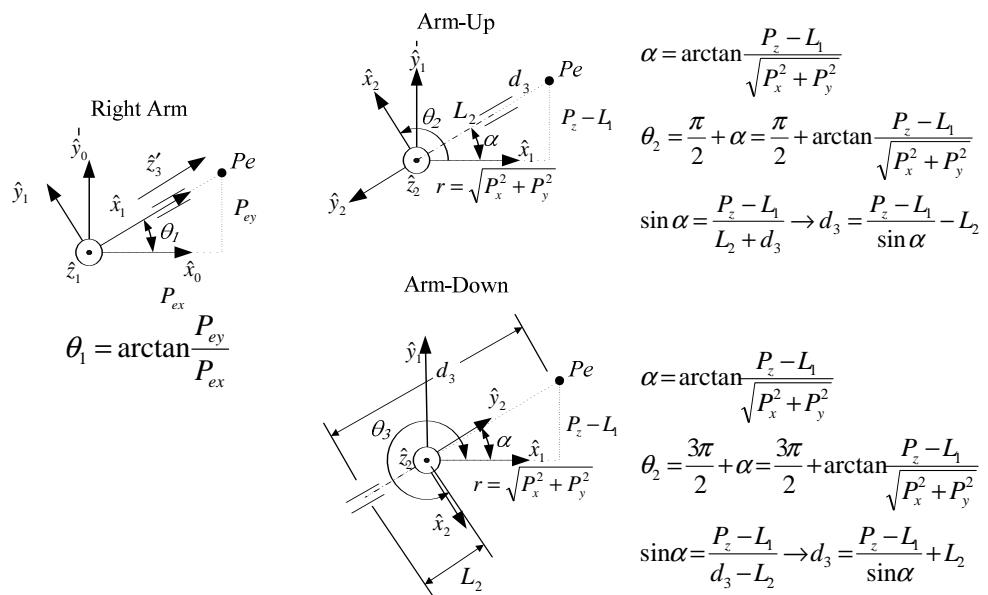


Figure iv.14: Inverse kinematic equations of a spherical robot: right-arm solution.

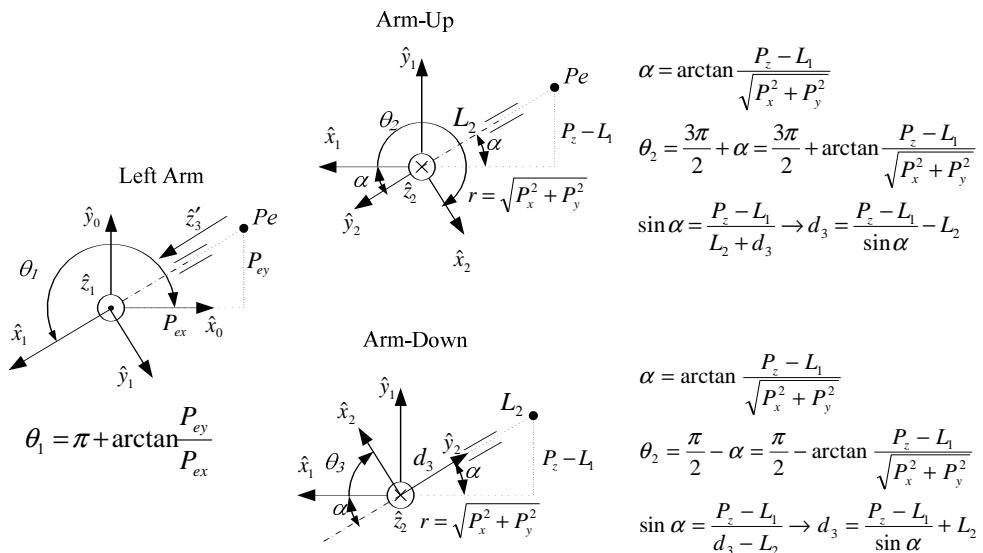


Figure iv.15: Inverse kinematic equations of a spherical robot: left-arm solution.



## **Exercises v**

# **Jacobian**

---

### **Exercise v.1**

Knowing the following expressions that solve the forward kinematics of a robot, calculate the corresponding jacobian matrix.

$$\begin{aligned} p_x &= -l_3 \cos \theta_1 \sin \theta_3 + (l_2 + d_2) \sin \theta_1 \\ p_y &= -l_3 \sin \theta_1 \sin \theta_3 - (l_2 + d_2) \cos \theta_1 \\ p_z &= l_3 \cos \theta_3 + l_1 \end{aligned}$$

**NOTE:** Only positions will be taken into account.

### **Exercise v.2**

Calculate the maximum force that the robot can exert from the previous exercise if the maximum forces/torques that the joints can develop are:

$$\tau_1 = 10N - m, \tau_2 = 5N, \tau_3 = 3N - m$$

and the robot is set in the direction given by:

$$\theta_1 = 0, d_2 = d_2, \theta_3 = \pi/2.$$

### **Exercise v.3**

Using the generalized velocity propagation analysis from joint to joint, obtain the jacobian matrix (related to the reference system  $\{0\}$ ) of the 2R planar manipulator shown in figure v.1. Obtain the singularities of the mechanism. Do we obtain the same singularities if we use the jacobian matrix expressed in system  $\{3\}$  (the one attached to the end-effector)?

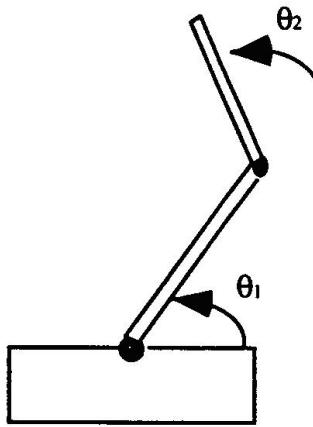


Figure v.1: RR planar manipulator.

**Exercise v.4**

Let's consider again the 2R planar robot used in the previous exercise (figure v.2). We want to move the end of the second arm over the  $\hat{x}$  axis at a constant velocity  $v$ , as is shown in the figure. Obtain the velocities of joint angles,  $\theta_1$  and  $\theta_2$ , that are required to keep that constant velocity. Would it be impossible to maintain the velocity for over time?

**Exercise v.5**

In the design process of a RR manipulator, one of the requirements is that manipulator would be able to exert, by its end-effector, a constant static force  ${}^0F = [f \ 0 \ 0]^T$ . Obtain the force/torque that must be applied to the joints. The jacobian matrix of the manipulator is:

$${}^0J = \begin{bmatrix} -\sin \theta_1 L_1 - \sin_{12} L_2 & -\sin_{12} L_2 \\ \cos \theta_1 L_1 + \cos_{12} L_2 & \cos_{12} L_2 \end{bmatrix}$$

**Exercise v.6**

Calculate the jacobian matrix for the 3R planar manipulator shown in the figure. Obtain the singularities of the mechanism.

**Exercise v.7**

Calculate the jacobian matrix of a 3 DoF anthropomorphic robot using the velocity propagation method. If the maximum torques that the robot can develop are  $\tau_1 = 1N-m$ ,

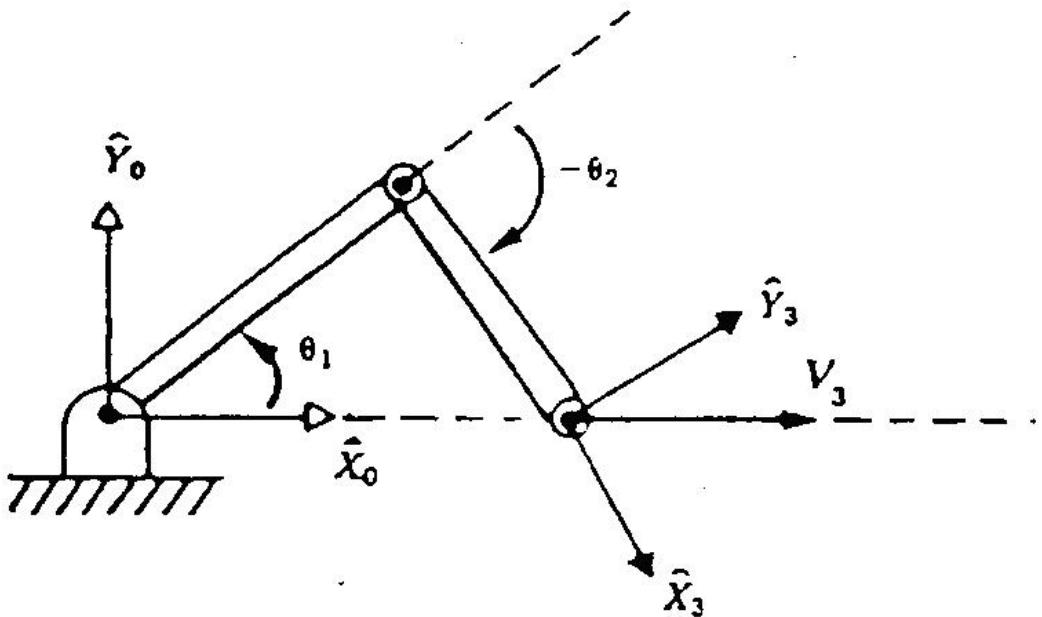


Figure v.2: RR planar manipulator with an end-point frame.

$\tau_2 = 0.5N - m$  and  $\tau_3 = 0.1N - m$ , what are the maximum forces and torques that the end effector can exert?

### Exercise v.8

Calculate the jacobian matrix of a cylindrical robot of 3 DoF using the force propagation method. Calculate the velocity on the end-effector of the robot, related to the velocities of each degree of freedom when  $q_1 = \pi/2$ ,  $q_2 = 0$ ,  $q_3 = 0$ .

### Exercise v.9

Given the following jacobian matrix:

$$\begin{bmatrix} l_3 \sin \theta_1 \sin \theta_3 + (d_2 + l_2) \cos \theta_1 & l_2 \sin \theta_1 & -l_3 \cos \theta_3 \cos \theta_1 \\ -l_3 \cos \theta_1 \sin \theta_3 + (d_2 + l_2) \sin \theta_1 & -l_2 \cos \theta_1 & -l_3 \cos \theta_3 \sin \theta_1 \\ 0 & 0 & -l_3 \sin \theta_3 \end{bmatrix}$$

calculate the singular configurations.

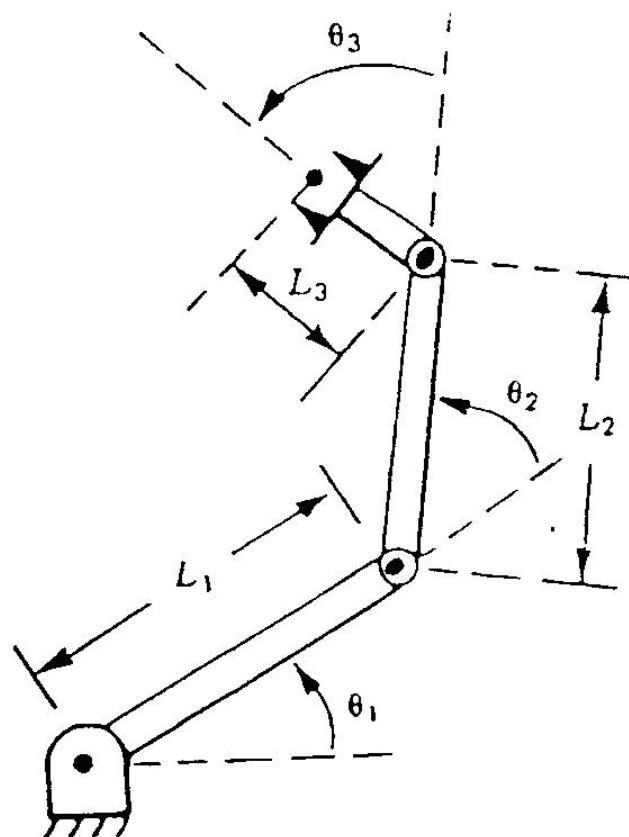


Figure v.3: RRR planar manipulator.

## Solutions

### Solution to exercise v.1

The Jacobian matrix is composed of velocity expressions. To obtain a velocity expression we have to derive the position equation,

$$\dot{p}_x = [L_3 \sin \theta_1 \sin \theta_3 + (L_2 + d_2) \cos \theta_1] \cdot \dot{\theta}_1 + \sin \theta_1 \cdot \dot{d}_2 + [-L_3 \cdot \cos \theta_1 \cos \theta_3] \cdot \dot{\theta}_3$$

$$\dot{p}_y = [-l_3 \sin \theta_3 \cos \theta_1 + (L_2 + d_2) \sin \theta_1] \cdot \dot{\theta}_1 - \cos \theta_1 \cdot \dot{d}_2 + [-L_3 \cdot \sin \theta_1 \cos \theta_3] \cdot \dot{\theta}_3$$

$$\dot{p}_z = [-L_3 \sin \theta_3] \cdot \dot{\theta}_3$$

$$J = \begin{bmatrix} L_3 \sin \theta_1 \sin \theta_3 + (L_2 + d_2) \cos \theta_1 & \sin \theta_1 & -L_3 \cos \theta_1 \cos \theta_3 \\ -L_3 \cos \theta_1 \sin \theta_3 + (L_2 + d_2) \sin \theta_1 & -\cos \theta_1 & -L_3 \sin \theta_1 \cos \theta_3 \\ 0 & 0 & -L_3 \sin \theta_3 \end{bmatrix}$$

### Solution to exercise v.2

$$J = \begin{bmatrix} L_2 + d_2 & 0 & 0 \\ -L_3 & -1 & 0 \\ 0 & 0 & -L_3 \end{bmatrix} \rightarrow J^T = \begin{bmatrix} L_2 + d_2 & -L_3 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -L_3 \end{bmatrix}$$

$$\tau = J^T \cdot f \rightarrow \begin{bmatrix} 10 \\ 5 \\ 3 \end{bmatrix} = \begin{bmatrix} L_2 + d_2 & -L_3 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -L_3 \end{bmatrix} \cdot \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$$

$$\left. \begin{array}{l} 10 = L_2 \cdot f_x + d_2 \cdot f_x - L_3 \cdot f_y \\ 5 = -f_y \\ 3 = -L_3 \cdot f_z \end{array} \right\} f_{max} \begin{bmatrix} \frac{10-5L_3}{L_2+d_2} \\ -5 \\ \frac{-3}{L_3} \end{bmatrix}$$

### Solution to exercise v.3

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 theta_2 L_1 L_2
syms thetadot_1 thetadot_2

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 0;...
      0 0 0 1]

```

```

T12=[cos(theta_2) -sin(theta_2) 0 L_1;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T23=[1 0 0 L_2; 0 1 0 0; 0 0 1 0;0 0 0 1]

T0e=simple(T01*T12*T23)

v_0=[0; 0; 0]
omega_0=[0; 0; 0]

R10=transpose(T01(1:3,1:3))
P_1=T01(1:3,4)

omega11=R10*omega_0+[0; 0; thetadot_1]
v11=R10*(v_0+cross(omega_0,P_1))

R21=transpose(T12(1:3,1:3))
P_2=T12(1:3,4)

omega22=R21*omega11+[0; 0; thetadot_2]
v22=R21*(v11+cross(omega11,P_2))

R32=transpose(T23(1:3,1:3))
P_3=T23(1:3,4)

omega33=simple(R32*omega22+[0; 0; thetadot_3])
v33=R32*(v22+cross(omega22,P_3))

omegaaee=omega33
vee=v33

```

### Solution to exercise v.4

To do this exercise, we first have to calculate the  ${}^0J$  matrix of the preceeding exercise.

$${}^0V_3 = {}^0J \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \sin \theta_1 L_1 \cos \theta_{12} - (\cos \theta_2 L_1 + L_2) \sin \theta_{12} & -\sin \theta_{12} L_2 \\ \sin \theta_1 L_1 \sin \theta_{12} + (\cos \theta_2 L_1 + L_2) \cos \theta_{12} & \cos \theta_{12} L_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \frac{v_x \cos \theta_{12}}{\sin \theta_1 L_1 + \sin \theta_{12} L_2 + \cos \theta_{12} L_2} \\ \left( \frac{\sin \theta_1 L_1 \sin \theta_{12} + \cos \theta_{12} \cos \theta_2 L_1 + \cos \theta_{12} L_2}{\sin \theta_1 L_1 + \sin \theta_{12} L_2 + \cos \theta_{12} L_2} \right) \cdot v_x \end{bmatrix}$$

### Solution to exercise v.5

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -\sin \theta_1 L_1 - \sin \theta_{12} L_2 & \cos \theta_1 L_1 + \cos \theta_{12} L_2 & 0 \\ -\sin \theta_{12} L_2 & \cos \theta_{12} L_2 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_x \\ 0 \\ 0 \end{bmatrix}$$

$$\tau_1 = f_x \cdot (-\sin \theta_1 L_1 - \sin \theta_{12} L_2)$$

$$\tau_2 = f_x \cdot (-\sin \theta_{12} L_2)$$

### Solution to exercise v.6

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 theta_2 theta_3 L_1 L_2 L_3
syms thetadot_1 thetadot_2 thetadot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T12=[cos(theta_2) -sin(theta_2) 0 L_1;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T23=[cos(theta_3) -sin(theta_3) 0 L_2;...
      sin(theta_3) cos(theta_3) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T3e=[1 0 0 L_3; 0 1 0 0; 0 0 1 0; 0 0 0 1]

T0e=simple(T01*T12*T23*T3e)

v_0=[0; 0; 0]
omega_0=[0; 0; 0]

R10=transpose(T01(1:3,1:3))

```

```

P_1=T01(1:3,4)

omega11=R10*omega_0+[0; 0; thetadot_1]
v11=R10*(v_0+cross(omega_0,P_1))

R21=transpose(T12(1:3,1:3))
P_2=T12(1:3,4)

omega22=R21*omega11+[0; 0; thetadot_2]
v22=R21*(v11+cross(omega11,P_2))

R32=transpose(T23(1:3,1:3))
P_3=T23(1:3,4)

omega33=simple(R32*omega22+[0; 0; thetadot_3])
v33=R32*(v22+cross(omega22,P_3))

Re3=transpose(T3e(1:3,1:3))
P_e=T3e(1:3,4)

omegaaee=simple(Re3*omega33)
vee=simple(Re3*(v33+cross(omega33,P_e)))

```

### Solution to exercise v.7

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 theta_2 theta_3 L_0 L_1 L_2 L_3 L_4
syms thetadot_1 thetadot_2 thetadot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 L_0+L_1;...
      0 0 0 1]

T12=[1 0 0 0;0 0 -1 0;0 1 0 0;0 0 0 1]...
      *[cos(theta_2) -sin(theta_2) 0 0;...
      sin(theta_2) cos(theta_2) 0 0;...
      0 0 1 0;0 0 0 1]

T23=[cos(theta_3) -sin(theta_3) 0 L_2;...
      sin(theta_3) cos(theta_3) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T3e=[1 0 0 L_3+L_4;0 1 0 0;0 0 1 0; 0 0 0 1]

```

```

v_0=[0; 0; 0]
omega_0=[0; 0; 0]

R10=transpose(T01(1:3,1:3))
P_1=T01(1:3,4)

omega11=R10*omega_0+[0; 0; thetadot_1]
v11=R10*(v_0+cross(omega_0,P_1))

R21=transpose(T12(1:3,1:3))
P_2=T12(1:3,4)

omega22=R21*omega11+[0; 0; thetadot_2]
v22=R21*(v11+cross(omega11,P_2))

R32=transpose(T23(1:3,1:3))
P_3=T23(1:3,4)

omega33=simple(R32*omega22+[0; 0; thetadot_3])
v33=R32*(v22+cross(omega22,P_3))

Re3=transpose(T3e(1:3,1:3))
P_e=T3e(1:3,4)

omegaaee=simple(Re3*omega33)
vee=simple(Re3*(v33+cross(omega33,P_e)))

```

### Solution to exercise v.8

You can get the complete solution with Matlab® using the following coding:

```

syms theta_1 d_2 d_3
syms thetadot_1 ddot_2 ddot_3

T01=[cos(theta_1) -sin(theta_1) 0 0;...
      sin(theta_1) cos(theta_1) 0 0;...
      0 0 1 0;...
      0 0 0 1]

T12=[1 0 0 0; 0 1 0 0; 0 0 1 d_2;0 0 0 1]

T23=[1 0 0 0; 0 0 -1 -d_3; 0 0 1 d_2;0 0 0 1]

f_3=[f_x;f_y;f_z]
n_3=[n_x;n_y;n_z]

```

```

R23=(T23(1:3,1:3))
P_3=T23(1:3,4)

f_2=R23*f_3
n_2=simple(R23*(n_3)+cross(P_3,f_2))

R12=(T12(1:3,1:3))
P_2=T12(1:3,4)

f_1=R12*f_2
n_1=simple(R12*(n_2)+cross(P_2,f_1))

taul=n_1(3)
tau2=n_2(3)
tau3=n_3(3)

```

Once we have the Jacobian Matrix, we can finish the exercise:

$$\begin{aligned} v_x &= d_3 \cos \theta_1 \cdot \frac{\pi}{2} & \omega_x &= 0 \\ v_y &= d_3 \sin \theta_1 \cdot \frac{\pi}{2} & \omega_y &= 0 \\ v_z &= 0 & \omega_z &= \frac{\pi}{2} \end{aligned}$$

### Solution to exercise v.9

To calculate the singularities we have to equate the determinant of the jacobian matrix to zero.

$$\begin{vmatrix} L_3 \sin \theta_1 \sin \theta_3 + (d_2 + L_2) \cos \theta_1 & L_2 \sin \theta_1 & -L_3 \cos \theta_3 \cos \theta_1 \\ -L_3 \cos \theta_1 \sin \theta_3 + (d_2 + L_2) \sin \theta_1 & -L_2 \cos \theta_1 & -L_3 \cos \theta_3 \sin \theta_1 \\ 0 & 0 & -L_3 \sin \theta_3 \end{vmatrix} = 0$$

Simplifying this equation it yields:

$$\left\{ \begin{array}{l} \sin \theta_3 = 0 \\ (d_2 + L_2) \cdot \underbrace{(\cos^2 \theta_1 + \sin^2 \theta_1)}_1 = 0 \rightarrow (d_2 + L_2) = 0 \end{array} \right.$$

# **Part V**

# **Appendix**



## Appendix A

# Robot History

---

### A.1 Introduction

The Robot concept is very ancient. It may surprise us. But mankind has always been fascinated by machines and devices capable of moving themselves, constructed to imitate the motions of human beings and animals. The ancient Greek culture named them as **automatos**. Those **automatos -automaton-** can be considered as the prehistory of present robots.

### A.2 Greek Mythology

The oldest reference about a robot -or a similar device- may be in the "Iliada" of Homer where we can read that **Hephaestus** (Greek: **Hephaistos**; Spanish: **Hefestos**, figure A.1<sup>1</sup>), god of fire, (Vulcan for Romans) had two golden servants:

\*Achilles, when he receives the news about the death of his friend Patroclus, wishes to take revenge. His mother, Tethys, asks Hephaestus if he would provide her son with a shield, which replaces the one that has been taken by Hector.

... Then he (Hephaestus) took a sponge and washed his face and hands, his shaggy chest and brawny neck; he donned his shirt, grasped his strong staff, and limped towards the door. **There were golden handmaids also who worked for him**, and were like real young women, with sense and reason, voice also and strength, and all the learning of the immortals; these busied themselves as the king bade them, while he drew near to Thetis, seated her upon a goodly seat, and took her hand in his own, saying, "Why have you come to our house, Thetis honoured and ever welcome -for you do not visit us often? Say what you want, and I will do it for you at once if I can, and if it can be done at all..."

The Iliad, Homer

---

<sup>1</sup>Collection: Toledo Museum of Art; Ware: Attic Red Figure; Shape: Skyphos; Painter: Attributed to the Kleophon Painter; Date: 430- 420 BC; Period: High Classical.



Figure A.1: The return of Hephaestus to Olympus.

#### Book XVIII, Fabrication of weapons

Another reference, in Greek mythology, is the Giant of Talus (figure A.2<sup>2</sup>). This giant automaton was given to Europa, Queen of Crete, by Zeus. In this case, we can read its description in ***The Argonautica***: His (Talus) task was patrolling the island of Crete in order to defend the island from pirates and enemies. Talus was killed by the magic of the witch Medea when the giant tried to prevent the Argonauts from arriving to the island.

(ll. 1638-1653) And **Talus, the man of bronze**, as he broke off rocks from the hard cliff, stayed them from fastening hawsers to the shore, when they came to the roadstead of Dicte's haven. He was of the stock of bronze, of the men sprung from ash-trees, the last left among the sons of the gods; and **the son of Cronos gave him to Europa to be the warden of Crete** and to stride round the island thrice a day with his feet of bronze. Now in all the rest of his body and limbs was he fashioned of bronze and invulnerable; but beneath the sinew by his ankle was a blood-red vein; and this, with its issues of life and death, was covered by a thin skin. So the heroes, though outworn with toil, quickly backed their ship from the land in sore dismay. And now far from Crete would they have been borne in wretched plight, distressed both by thirst and pain, had not Medea addressed them as they turned away...

The Argonautica, Book IV, Apollonius Rhodius  
(3<sup>rd</sup> Century B.C.)

It is obvious these machines are fictitious, and they have never existed. But we can notice that, in Greek culture, the fabrication and use of these machines were restricted to the gods.

---

<sup>2</sup>Collection: Ruvo, Museo Jatta; Summary: death of Talos  
Ware: Attic Red Figure  
Shape: Volute krater  
Painter: Name vase of the Talos Painter  
Date: 400-390 BC  
Period: Late Classical

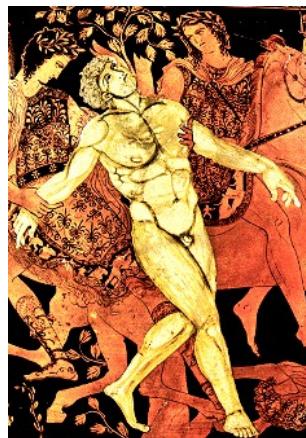


Figure A.2: Death of Talus.

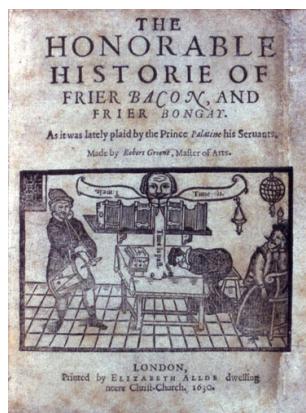


Figure A.3: Fryer Bacon's speaking head.

### A.3 Middle ages

There are few documented references in the XIII century, such as the iron man of St. Albert the Great (1204-1282) or the fryer Roger Bacon's (1214-1294) brozen head that was able to speak. The popular legend was written in The Famous History of Fryer Bacon (figure A.3). The oldest automaton that is conserved nowadays is in the original clock in Strasbourg cathedral (figure A.4). This automaton is a cock (1352) that crows three times and flaps its wings at noon. At the beginning of the 16<sup>th</sup> century, Juanelo Turriano (Gianello Torriano) designed and constructed one of the finest astronomical clocks of the Renaissance. He also constructed astonishing waterworks system which provided water from the River Tagus<sup>3</sup> to Toledo. There is also a legend about this engineer that says he constructed an automaton monk. This monk was called ***the wooden man***<sup>4</sup>. According to tradition, this automaton used to walk daily to the archbisop's palace, and return with bread and meat.

In 1515, Leonardo da Vinci created a mechanical lion that moved around like a real

<sup>3</sup>River Tagus=Río Tajo

<sup>4</sup>The wooden man: el hombre de palo

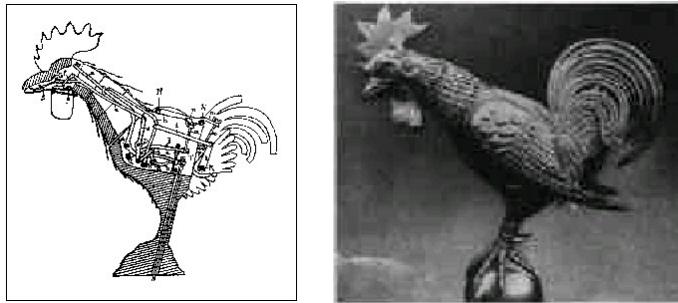


Figure A.4: Cock of the Strasbourg Cathedral.

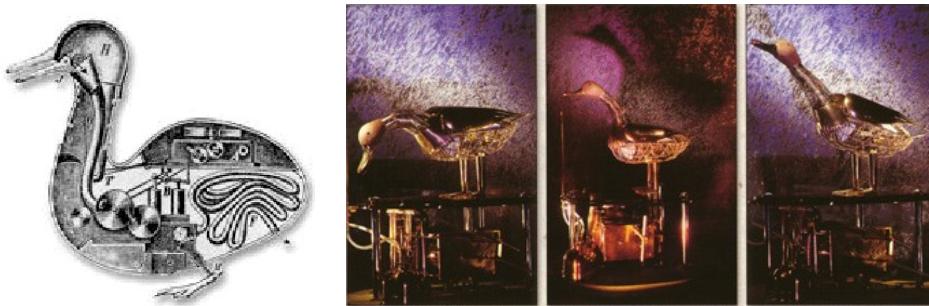


Figure A.5: Vauncanson's duck: a diagram of its interior and a replica of it at ***Le Musée de Automates de Grenoble***.

one and presented it to Francois I of France during a banquet: it crossed the hall, halted before the king, roared, and opened its chest to reveal a bunch of lilies.

## A.4 XVIII-XIX centuries

Later versions of automatons were based on self-contained clock mechanisms. In 1738, Vaucanson presented his first complete automaton, ***The Flute Player***. A year later, he produced ***The Tambourine Player*** and ***The Duck*** (figure A.5). This duck was able to quack, flap its wings, drink water, eat food, and discard waste. In 1759, Von Kempelen developed an amazing automaton. It was a very good chess player called ***The Turk*** (A.6). But, finally, the trick was discovered: "the wisdom of the automaton" came from a man hidden inside of a secret compartment. Between 1768 and 1770, Jacques Droz invented one of the most complicated automatons in history: ***The Writer*** that could write any message up to 40 characters long. During the same period of time, he also built ***The Musician*** and ***The Draughtsman***. The figure A.7 shows these three dolls. In 1805, Maillardet built a spring-activated automaton that could draw pictures and write in both French and English (figure A.8).

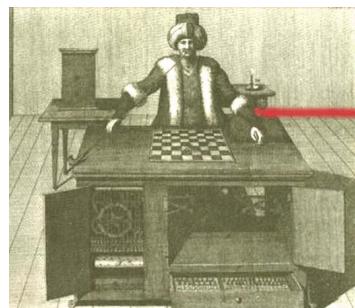


Figure A.6: Von Kempelen's chess player.



Figure A.7: Droz's dolls at the Art and History Museum in Neuchâtel, Switzerland.



Figure A.8: Maillert's automaton and two of its drawings

## A.5 XX Century: Robot Timeline

This section reviews the most important events in XX<sup>th</sup> century related to robotics <sup>5</sup>.

**1921** The term “robot” is first used in *R.U.R.* (Rossum’s Universal Robots), a play by Czech writer Karel Capek.

**1926** Fritz Lang’s movie *Metropolis* features Maria, a robot seductress.

**1930s** Hollywood’s serial films, such as *Flash Gordon* and *Buck Rogers*, often portray robots as malevolent machines.

**1938** Willard Pollard and Harold Roselund invent a mechanical arm with joints for an automated spray-painting machine from DeVilbiss Co.

**1939** For the New York World’s Fair, Westinghouse Electric Corp. builds a mechanical man and dog: Electro danced, counted to 10, smoked, and described Westingouse’s products and his dog walked, stood on its hind legs, and barked.

**1942** Isaac Asimov writes *Runaround*, in which he promulgates the Three Laws of Robotics: A robot may not injure a human being or, through inaction, allow a human being to come to harm. A robot must obey the orders given it by human beings, except where such orders would conflict with the First Law. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

**1946** George C. Devol patents a general-purpose device for controlling factory machines, using magnetically stored instructions.

**1947** Alan M. Turing’s article on intelligent machinery launches the modern field of artificial intelligence (AI).

**1950** *I, Robot*, a landmark collection of Asimov’s stories, is published.

**1950s** At Carnegie Mellon University (CMU), Herbert A. Simon, Allen Newell, and J. Clifford Shaw lay many cornerstones of AI.

**1951** Raymond Goertz designs a tele-operated arm to handle radioactive materials for the Atomic Energy Commission.

**1951** In the movie “The Day the Earth Stood Still,” the robot Gort possesses superior intelligence.

**1951** Japanese artist Osamu Tezuka creates Tetsuwan Atomu, or Mighty Atom, a cartoon series that influences several generations of Japan’s roboticists.

**1954** Devol designs a programmable factory robot (patent granted in 1961) aimed at “Universal Automation,” later trimmed to Unimation.

**1956** Devol’s design prompts Joseph F. Engelberger to champion industrial robots

---

<sup>5</sup>Source: BusinessWeek, electronic edition. March 19, 2001 [http://www.businessweek.com/magazine/content/01\\_2/b3724008.htm](http://www.businessweek.com/magazine/content/01_2/b3724008.htm) :

and make UnimationInc . the world's robot pioneer.

**1956** Robby the robot is featured in *Forbidden Planet*—and later appears in more than a dozen movies and TV shows.

**1959** A prototype Unimate arm from Unimation is installed in a General Motors Corp. die-casting factory in New Jersey—where the first commercial industry robot goes online in 1961.

**1959** Marvin L. Minsky and John McCarthy establish the AI Laboratory at Massachusetts Institute of Technology.

**1960** AMF Corp. introduces its Versatran industrial robot, developed by Harry Johnson and Veljko Milenkovic.

**1963** Stanford University forms an AI Lab headed by McCarthy.

**1965** CMU creates the Robotics Institute.

**1966** B-9 is the robotic hero in the TV series *Lost in Space*.

**1967** At General Electric Co., Ralph Moser designs the Walking Truck, a big four-leg robot, which the Pentagon wanted for hauling loads.

**1967** Japan imports its first industrial robot, a Versatran from AMF.

**1968** Unimation licenses its technology to Kawasaki Heavy Industries Ltd. The deal helps precipitate an explosion of robot development in Japan, and by 1990, Japan's 40-odd robotmakers dominate world markets.

**1968** Shakey, the first mobile robot with vision and AI, emerges from Stanford Research Institute (SRI). The aptly named robot is an unstable box on wheels that figures out how to get around obstacles.

**1968** Arthur C. Clarke's best-selling *2001: A Space Odyssey* inspires many students to take up robotics and AI, including several of today's robotics gurus.

**1970** *Doraemon*, a cartoon series featuring a robotic cat from the 22<sup>nd</sup> century, takes Japan by storm. It becomes a so-called Manga series in 1974 and later a TV series.

**1970** SRI unveils the Stanford arm, an improvement on the Unimate.

**1971** Cincinnati Milacron Inc. markets T3 (The Tomorrow Tool), a computer-controlled robot designed by Richard Hohn.

**1972** Shigeo Hirose, a graduate student at Tokyo Institute of Technology, builds a snakelike robot.

**1973** Wabot-1, a life-size humanoid robot, is born at Tokyo's Waseda University under Ichiro Kato.

**1974** Victor Scheinman leaves Stanford and founds Vicarm Inc. to commercialize the Stanford arm.

- 1976** NASA provides Mars landers with robot arms for its Viking I and II missions.
- 1977** Asea Brown Boveri Ltd. introduces microcomputer-controlled robots.
- 1977** Unimation purchases Vicarm. Scheinman later starts Automatix Inc.
- 1977** *Star Wars* stars an android, C3PO, and a mobile robot, R2D2. By the early 1980s, R2D2 lookalikes are vacuuming floors and singing songs in Japan.
- 1978** Unimation and GM develop Puma (programmable universal machine for assembly), based on Vicarm's technology.
- 1979** Yamanashi University designs the Scara arm for assembly jobs in factories. IBM teams with Sankyo Robotics to market the robots.
- 1980** Marc Raipert establishes the Leg Lab at MIT to develop robots that mimic human walking.
- 1982** Fanuc Ltd. and GM form a joint venture, and Fanuc Robotics North America Inc. quickly becomes a leading supplier in the U.S.
- 1983** A six-leg walking robot is unwrapped by Odetics Inc.
- 1984** Waseda University's Wabot-2 reads music and plays an electronic organ at Tsukuba Science Expo.
- 1984** Engelberger starts Transition Research Corp. (later renamed HelpMate Robotics Inc.) to develop service robots for hospitals.
- 1986** Atsuo Takanishi of Waseda University develops advanced controls for a walking robot.
- 1986** Honda Motor Co. launches a secret project to build a humanoid robot.
- 1988** The first HelpMate robot goes to work at Danbury (Conn.) Hospital.
- 1990** Robodoc, developed by Dr. William Bargar and Howard Paul of Integrated Surgical Systems Inc. and the University of California at Davis, performs a hip-replacement operation on a dog and in 1992, on a human patient.
- 1993** MIT's Rodney A. Brooks starts building Cog, a robot that is being raised and educated like a human.
- 1994** Dante II, a walking robot built by CMU's Robotics Institute, explores an active volcano in Alaska, collecting samples of volcanic gases.
- 1996** Honda unveils P-2 (prototype 2), a humanoid robot that walks.
- 1997** The first annual RoboCup soccer tournament is held in Nagoya, Japan, as a test bed for the latest technology in robotics and AI. Subsequent events have been staged in Paris, Stockholm, and Melbourne.
- 1997** NASA's Pathfinder lands on Mars, and the Sojourner rover robot explores the Martian terrain.

**2000** At RoboCup 2000, three humanoid robots meet for the first time: Johnny Walker from the University of Western Australia, the Mk-II from Japan's Aoyama Gakuin University, and Pino from Kitano Symbiotic Systems Project.



## Appendix B

# Mobile Robot Samples

---

### B.1 Microbots

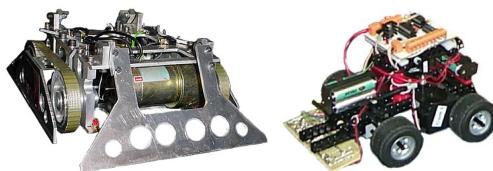


Figure B.1: Samples of Micro-robots: Sumo Fighter (MAZO),line-tracking (PIONERO), Microrobotics Club of TECNUN.

### B.2 Legged Robots

#### B.2.1 AIBO



Figure B.2: AIBO (SONY): <http://www.sony.com.au/aibo>

#### B.2.2 SDR

The AIBO's specifications are summarized in the next table:

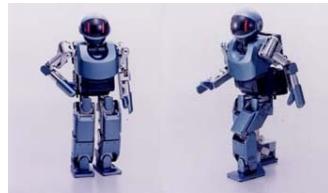


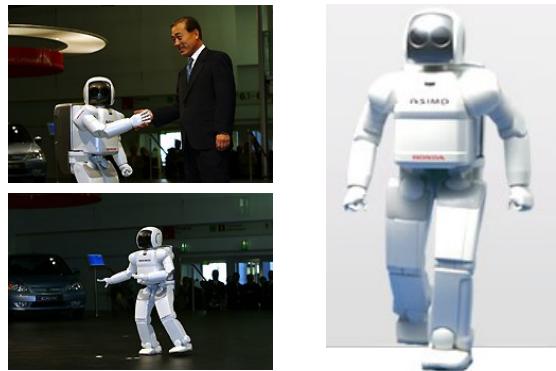
Figure B.3: SDR 3x (SONY): <http://au.playstation.com/technology/sonyrobot.jhtml>



Figure B.4: SDR 4x (SONY): <http://au.playstation.com/technology/sonyrobot.jhtml>

CPU	64 bit RISC processor (x2)
Main Recording Device	64MB DRAM (x2)
Operating System	Aperios (Sony's original real time OS)
Robot Control Architecture	OPEN-R
Control Program Supplying media	16MB Memory Stick
Joint Degrees of Freedom	Neck: 4 degrees of freedom, Body: 2 degrees of freedom, Arms: 5 degrees of freedom (x2), Legs: 6 degrees of freedom (x2); total 28 degrees of freedom + 5 fingers on each hand
Walking Speed	Approximately 6m/minute max (irregular surface) Pace: 10cm, Walking Cycle: 1.0 second/step  Approximately 20m/minute max (flat, smooth surface) Pace: 6.5cm, Walking Cycle: 0.20 second/step
Irregular Surface Walking Ability	Irregularity degree: 10mm irregular surface on non-slip condition  Tilt degree: Up to approx. 10 degrees tilted surface on non-slip condition
Weight	Approximately 6.5Kg with battery and memory
Dimensions (height x width x depth)	Approximately 580 x 260 x 190mm

### B.2.3 ASIMO

Figure B.5: ASIMO (HONDA): <http://world.honda.com/ASIMO>

Specifications		
Weight	52kg	
Walking Speed	0-1.6km/h	
Walking Cycle	Cycle Adjustable Stride Adjustable	
Grasping Force	0.5kg/hand(5-finger hand)	
Actuator	Servomotor+Harmonic Speed Reducer +Drive Unit	
Control Unit	Walk/Operating Control Unit, Wireless Transmission Unit	
Sensors	Foot 6-Axis Foot Area Sensor Torso Gyroscope & Acceleration Sensor	
Power Section	36.4V/10AH(Ni-MH)	
Operating Section	Workstation and portable Controller	

Degrees of Freedom (For Human Joints)		
Head	Neck Joint(U/D,RT)*1	2DOF
Arm	Shoulder Joint(F/B,U/D,RT)	3DOF
	Elbow joint(F/B)	1DOF
	Wrist joint(R/T)	1DOF
		5DOF X 2arms=10DOF
Hand	5fingers(Grasping)	1DOF
		1DOF X 2hands=2DOF
Leg	Hip joint(F/B,L/R,RT)	3DOF
	Knee joint(F/B)	1DOF
	Ankle joint(F/B,L/R)	2DOF
		6DOF X 2legs=12DOF

\*1  
F/B : Forward/Backward U/D : Up/Down  
L/R : Left/Right RT \* Rotation DOF : Degrees of Freedom

Figure B.6: Specifications of ASIMO.

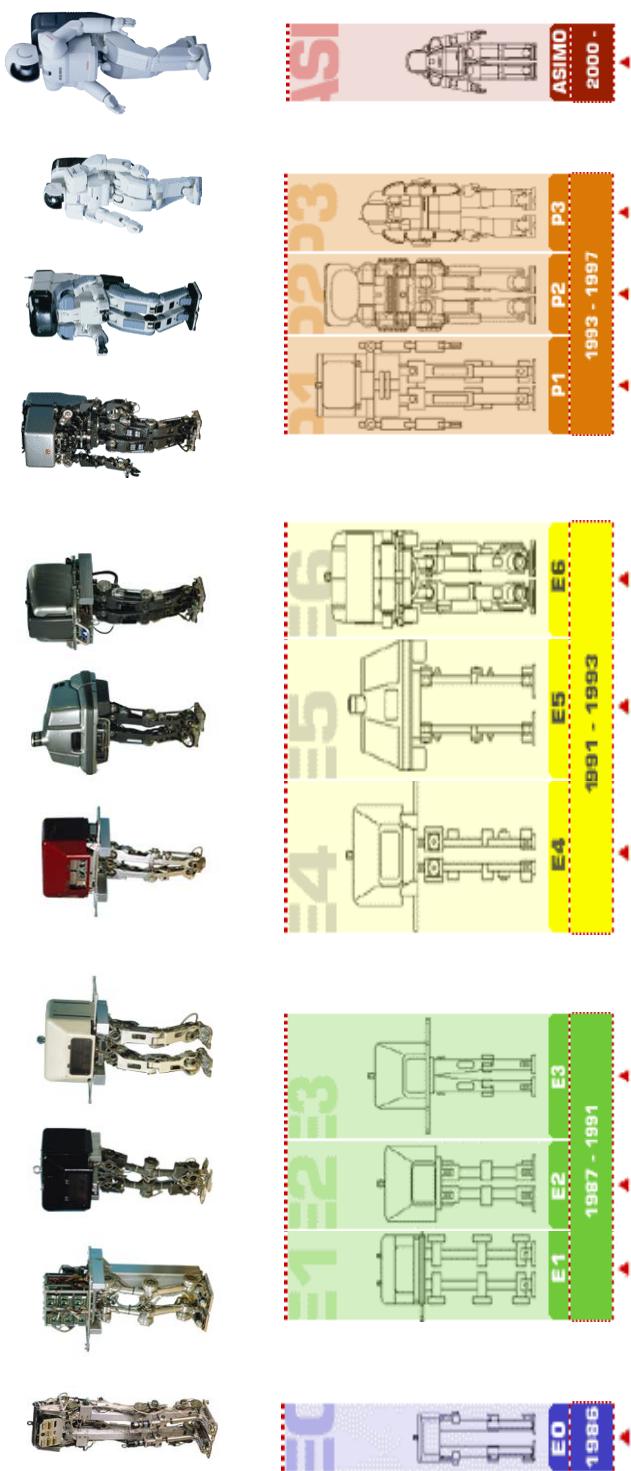


Figure B.7: Evolution of ASIMO

## **Appendix C**

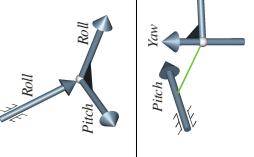
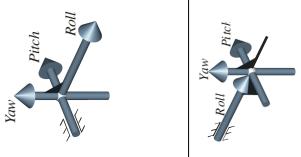
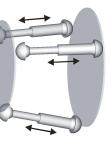
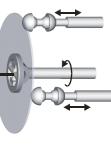
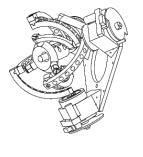
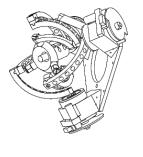
# **Robotic Wrists**

---

Extracted from a work of Javier Martín<sup>1</sup>.

---

<sup>1</sup>J. Martin, J. Savall, "Mechanisms for Haptic Torque Feedback", World Haptics Conference. March 21, 2005, Pisa, Italy.

DoFs Workspace dexterity	Singular configurations	Complexity and actuator arrangement	Pictures
3 Wide and homogen.	When the rolls DoFs are aligned	Small complexity. The first actuator has to move the others. It's the most common.	 Roll Pitch Yaw
3 Wide and homogen., but the rotations and translations are coupled	None	Very small complexity. The first actuator has to move the others. a Yaw-Pitch-Roll, rotated 90°	 Pitch Yaw Roll
3 Wide and homogen.	Two. When the Pitch and the Roll are aligned. There are 180° between the two singularities. They are perpendicular to the RPR one.	Very small complexity. The first actuator has to move the others. They are perpendicular to the RPR one.	 Pitch Yaw Roll
3 Wide and homogen.	Two. When the Pitch and the Roll are aligned. There are 180° between the two singularities. They are perpendicular to the RPR one.	Very small complexity. The first actuator has to move the others. They are perpendicular to the RPR one.	 Pitch Yaw Roll
3 Poor Hemispheric WS. Strike short.	dexterity. None	Difficult to make. Three linear actuators, none has to move others.	
3 Poor Hemispheric WS. Strike short. But longer roll.	WS. Roll	Very difficult to make. Three fixed actuators. Two of them are linear	
3 Poor dexterity. Less than hemispheric WS.	None	Difficult to make. Three fixed and rotating actuators.	

## **Appendix D**

# **Glossary**

---

Based on information extracted from University of North Texas <sup>1</sup>.

### **-A-**

**Accuracy (precisión)**-The precision with which a computed point can be attained, or measured. The more precise the measurements of joint angles will enable a finer degree of accuracy with a robot.

**Active Compliant Robot (robot con acomodación activa)**- A compliant robot in which motion modification during the performance of a task is initiated by the control system. The induced motion modification is slight, but sufficient to facilitate the completion of a desired task. This assumes the robot is mechanically designed to allow the successful intervention of the control system

**Actuator (actuador)**- A mechanical, pneumatic, hydraulic or electric device in a control system that is able to change and/or maintain the position of an element (such as an end-effector) the performs a task. The actuator responds to a signal received from the control system.

**Adaptive Control System (sistema de control activo)**- A control system that senses one or more parameters and utilizes the sensor data to adjust control signals to meet the performance criteria.

**Android (android)**- A robot resembling a human in physical appearance. This robot also possesses many human-like movements.

**Anthropomorphic (antropomórfico)** - Human like. An articulated manipulator is anthropomorphic due to its shoulder, elbow, and wrists joint movements. See articulation.

**Arm (brazo)**- An interconnected set of links and powered joints comprising a robot manipulator that supports and/or moves a wrist and hand or end-effector through space. The arm itself does not include the end-effector. See Manipulator, End-effector and Wrist.

---

<sup>1</sup><http://www.unt.edu/>

**Articulated Manipulator (manipulador articulado)**- A manipulator with an arm that is broken into sections (links) by one or more joints. Each of the joints represents a degree of freedom in the manipulator system and allows translation and rotary motion.

**Articulation (articulación)**- Describes a jointed device, such as a jointed manipulator. The joints provide rotation about a vertical axis, and elevation out of the horizontal plane. This allows a robot to be capable of reaching into confined spaces.

**Artificial Intelligence (inteligencia artificial)**- The programming and ability of a robot to perform functions that are normally associated with human intelligence, such as reasoning, planning problem solving pattern recognition, perception, cognition, understanding, and learning.

**Assembly Robot (robot de ensamblado)**- A robot designed specifically for mating, fitting, or otherwise assembling various parts or components into completed products. Primarily used for grasping parts and mating or fitting them together, such as in assembly line production.

**Auto-adaptive (auto-adaptativo)**- The capability of adjusting to and performing tasks in unanticipated environmental conditions.

**Automated Guided-Vehicle System (AGVs) – (vehículos con guiado automático)** - Vehicles that posse automatic guidance equipment and follow a prescribed guide path.

**Autonomous Robot (robot autónomo)** - A self-sufficient robot. This robot is programmed to learn from its environment and provide self-neural technology to operate without further human intervention.**Azimuth** - Direction of a straight line to a point in a horizontal plane, expressed as the angular distance from a reference line, such as the observer's, or robot's lines of views.

## **-B-**

**Base (Base)** - The stable platform to which a robot arm is attached.

**Bilateral manipulator (manipulador bilateral)**- A master-slave manipulator with symmetric force reflection in which both the master and slave arms have sensors and actuators such that for any degree of freedom a positional error between the master and slave results in the application of equal and opposite forces to the master and slave arms.

## **-C-**

**Cable Drive (transmisión por cable)**- Transmission of power from an actuator to an object by means of a cable and pulleys, i.e.; a crane type robot may use such a device.

---

**Cartesian Manipulator (manipulador cartesiano)**- An arm with prismatic joints that allows movement along one or more of the three- axes in the x, y, z coordinate system.

**Cartesian-Coordinate Robot (robot cartesiano)** - A robot whose manipulator-arm degrees of freedom are defined by Cartesian coordinates. This describes motions that are east-west, north-south and up-down, as well as rotary motions to change orientation.

**Centrifugal Force (fuerza centrífuga)**- When a body rotates about an axis other than one at its center of mass, it exerts an outward radial force called centrifugal force upon the axis which restrains it from moving in a straight tangential line. To offset this force, the robot must exert an opposing torque at the joint of rotation.

**Chain Drive (transmisión de cadena)**- Transmission of power from an actuator to a remote mechanism by means of a flexible chain and a toothed sprocket wheel. i.e.: such as used in a bicycle, but can be used in robots to transfer power as well.

**Clamp (pinza)**- An end-effector that serves as a hand that controls the grasping and releasing of an object. Tactile, and feedback force sensors are used to manage the applied force to the object by the clamp. *See End-Effector.*

**Closed-Loop Control (control en lazo cerrado)**- control achieved by a robot manipulator by means of feedback information. As a manipulator is in action, its sensors continually feed back information to the robot's controller. This information is used to further guide the manipulator within the given task. Many sensors are used to feedback information about the manipulator's placement, speed, torque, applied forces, as well as the placement of a targeted moving object, etc. *See feedback.*

**Compliance (acomodado)**- Displacement of a manipulator in response to a force or torque. A high compliance means the manipulator moves a good bit when it is stressed. This is called spongy or springy. Low compliance would be a stiff system when stressed.

**Compliant Robot (robot con movimiento acomodable)**- A robot that performs tasks, with respect to external forces, by modifying its motions in a manner that minimizes those forces. The indicated or allowed motion is accomplished through lateral (horizontal), axial (vertical) or rotational compliance.

**Contact Sensor (sensor de contacto)**- A device that detects the presence of an object or measures the amount of applied force or torque applied on the object through physical contact with it. Contact sensing can be used to determine location, identity, and orientation of workpieces

**Continuous Path (trayectoria/camino continuo)**- Describes the process where by a robot is controlled over the entire path traversed, as opposed to a point-to-point method of traversal. This is used when the trajectory of the end-effector is most important to provide a smooth movement, such as in spray painting etc. *See Point-to-Point.*

**Control Algorithm (algoritmo de control)**- A controller used to detect trajectory deviations, in which sensors detect such deviations, and torque/force applications are computed for the actuators.

**Controllability (controlabilidad)**- The property of a system by which an input signal can take the system from an initial state to a desired state along a predictable path within a predetermined period of time.

**Controller System (controlador)**- The robot control mechanism. Usually a computer of some type that is used to store data (both robot and work environment), and store and execute programs which operate the robot. The controller system contains the programs, data, algorithms; logic analysis, and various other processing circuits that enable it to perform. See Robot.

**Cylindrical Coordinate System (sistema de coordenadas cilíndricas)**- A coordinate system that defines the position of any point in terms of an angular dimension, a radial dimension, and a height from a reference plane. These three dimensions specify a point on a cylinder.

## **-D-**

**Degrees of Freedom (DoF) (grados de libertad)** - The number of independent position variables a manipulator posses, which would allow the robot to move its end effector through the required sequence of motions. For example, a robot with six degrees of freedom (DoF) consists of three arm and body motions and three wrist motions. The six DOFs are vertical traverse (up and down along the horizontal axis or moving the arm along a vertical slide) of the arm, radial traverse (extension and retraction, in-out movement) of the arm, rotational traverse (rotation about the vertical axis), wrist swivel (roll), wrist bend (pitch), and wrist yaw.

**Downtime (tiempo de apagado)**- A period of time in which a robot, or production line is shut down due to malfunction or failure. See Uptime.

**Drop Delivery (depositar por gravedad)**- A method of introducing an object to the workplace by gravity. Usually, a chute or container is so placed that, when work on the part is finished, it will fall or drop into a chute or onto a conveyor with little or no transport by the robot.

**Dynamics (dinámica)**- The study of motion and of the forces that cause the motion.

## **-E-**

**Elastic Limit (límite elástico)**- The maximum stress to which a material may be subjected without any permanent strain remaining upon complete release of stress. This knowledge is used to determine the amount of force a robot may apply to a material without harming it. See Stress.

**End Effector (herramienta/pinza colocada en el extremo final del robot)**- A tool or gripping mechanism attached to the wrist of a robot to accomplish some task. It usually encompasses a motor, or driven mechanical device. It is used as a sensor,

gripping device, paint gun, drill, arc welding device, etc. *See Wrist, and Sensor.*

**Error** - The difference between the actual response of a robot to a command issued.

**Exoskeleton (exoesqueleto)**- An articulated mechanism whose joints correspond to those of a human arm. When attached to the arm of a human operator, it will move in correspondence to his or her arm. Sometimes used as prosthesis devices for handicapped humans. *See Prosthetic Robot.*

## -F-

**Feedback (retroalimentación, realimentación)**- The return of information from a manipulator, or sensor to the processor of the robot to provide self-correcting control of the manipulator. *See Feedback Control, and Feedback Sensor.*

**Feedback Control (control retroalimentado)**- A type of system control obtained when information from a manipulator, or sensor is returned to the robot controller in order to obtain a desired robot effect. *See Feedback, Closed-Loop Control and Feedback Sensor.*

**Feedback Sensor (sensor de realimentación)**- A mechanism through which information from sensing devices is fed back to the robot's control unit. The information is utilized in the subsequent direction of the robot's motion. *See Closed-Loop Control and Feedback Control.*

**Feedforward (prealimentación)**- The portion of a control algorithm that does not receive any feedback. *See Feedback Control, and Feedback Sensor.*

**Flexibility (flexibilidad)**- The ability of a robot to perform a variety of different tasks.

**Force Control (control en fuerza)**- the regulation of the amount of force applied to a surface by an actuator.

**Force Feedback (realimantación de fuerza)**- a sensing technique using electrical or hydraulic signals to control a robot end-effector during the task of the end-effector. Information is fed from the force sensors of the end-effector to the robot control unit during the particular task to enable enhanced operation of the end-effector. *See Feedback, Feedback Sensor and Force Sensor.*

**Force Sensor (sensor de fuerza)**- A sensor capable of measuring the forces and torque exerted by a robot and its wrist. Such sensors usually contain strain gauges. The sensor provides information needed for force feedback. *See Force feedback, Strain, Stress, and Strain Gauge.*

**Forearm (antebrazo)**- The portion of the robot's jointed arm that is connected at the wrist. *See Wrist, and Arm.*

**Forward Kinematics (cinemática directa)**- Procedures that determine where the end-effector of a robot is located in space. The procedures use mathematical algorithms along with joint sensors to determine its location. *See Inverse Kinematics.*

**Frame (sistema coordenado)**- A coordinate system used to determine a position and orientation of an object in space, as well as the robot's position within its model.

## -G-

**Gantry (pórtico/puente grúa)**- An adjustable hoisting machine that slides along a fixed platform or track, either raised or at ground level along the x, y, z axes.

**Gantry Robot (robot cartesiano tipo pórtico/puente grúa)**- A robot that has three degrees of freedom along the X, Y, and Z coordinate system. Usually consists of a spooling system (used as a crane) which when reeled or unreeled provides the up and down motion along the Z axis. The spool can slide from left to right along a shaft that provides movement along the Z axis. The spool and shaft can move forward and back along tracks that provide movement along the Y axis. Usually used to position it's end-effector over a desired object and pick it up. See *Cartesian robot*.

**Gravity Loading (carga gravitatoria)**- The force exerted downward, due to the weight of the robot arm and/or the load at the end of the arm. The force creates an error with respect to position accuracy of the end-effector. A compensating force can be computed and applied bringing the arm back to the desired position.

**Gripper (pinza)**- The device used by the end-effector that "grips" the object. It is attached to the last link of the arm. It may hold an object using several different methods, such as: applying pressure between its "fingers", or may even use magnetization to hold the object, etc. See *End-Effector*.

## -H-

**Haft-bridge Robot (robot cartesiano de 2gdl)**- A Cartesian robot in which there is a north-south axis and an up-down axis but no east-west axis. See *Cartesian Robot*.

**Hand (pinza, mano, garra)**- A clamp or gripper used as an end-effector to grasp objects. See *End-Effector*.

**Hazardous Motion (movimiento peligroso/no esperado)**- Unintended/unexpected robot motion that may cause injury. This can be eliminated with the use of computer simulation. See *Simulator*.

**Hold (positionamiento)**- A stopping of all movements of a robot during its sequence, in which some power is maintained on the robot. For example, on hydraulically driven robots, power is shut off to the servovalves but is present -on the main electrical and hydraulic systems.

**Home Position (posición casa)**- A known and fixed location on the basic coordinate axis of the manipulator where it comes to rest, or to zero-out its position. Usually, this is the point where the manipulator is fully retracted.

**Hydraulically Driven Robot (robot hidráulico)**- Robots that use hydraulic servo valves that operate on petroleum-based hydraulic fluid. The hydraulic system includes a hydraulic power supply as well. These robots can provide very high repeatability and accuracy. This type of robot is mechanically simpler and has more physical strength than electrically driven robots. *See Hydraulic motor.*

**Hydraulic Motor (motor hidráulico)**- An actuator consisting of interconnected valves, pistons, or vanes that converts high-pressure hydraulic fluid into mechanical shaft translation or rotation. *See Hydraulic Driven Robot.*

## -I-

**Inductive Sensors (sensores inductivos)**- the class of proximity sensors that has half of a ferrite core, whose coil is part of an oscillator circuit. When a metallic object enters this field, at some point the object will absorb enough energy from the field to cause the oscillator to stop oscillating. This signifies that an object is present in a given proximity. *See Proximity Sensor.*

**Industrial Robot (robot industrial)**- a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or other devices through variable programmed motions for the performance of a variety of tasks. The principle components are: one or more arms that can move in several directions; a manipulator; a computer controller that gives detailed movement instructions.

**Input Devices (dispositivos de entrada)**- a variety of devices which allow a human to machine interface. This allows the human to program, control, and simulate the robot. Such devices include computer keyboards, a mouse, joy-sticks, push buttons, etc.

**Instruction (instrucción)**- A basic command/order fed to the robot by means of the human-to-machine input device. This command is received by the robot's controller system and is interpreted. Then, the proper instruction is fed to the robot's actuators that enable it to react to the initial command. Many times the command must be interpreted with the use of logic units and specific algorithms. *See Input Device and Instruction Cycle.*

**Instruction Cycle (ciclo de instrucción)**- the time it takes for a robot controller system's cycle to decode a command or instruction before it is executed. This has to be analyzed very closely by robotics programmers to enable speedy and proper reaction to varying commands.

**Intelligent Robot (robot inteligente)**- a robot that can be programmed to make performance choices contingent on sensory inputs with little or no help from human intervention. *See Robot, and Autonomous.*

**Internal Sensor (sensor interno)**- a feedback device in the robot manipulator arm that provides data to the controller on position and orientation of the arm. *See Feedback, and Feedback Sensor.*

**Inverse Kinematics (cinemática inversa)**- procedures that determine where the end-effector needs to be placed to reach a particular point in space. The procedures use mathematical algorithms along with sensors to determine the desired location of the point in space, and the necessary manipulator joint movements to reach this point. *See Forward Kinematics.*

## -J-

**Joint (articulación)**- A part of the manipulator system that allows a rotation and/or translational degree of freedom of a link of end-effector.

**Joint Angles (ángulos articulares /de las articulaciones)**- A measure of the displacement incurred by revolute joints.

**Jointed-Arm Robot (robot antropomórfico)**- A robot that looks like a human arm. It consists of several links connected by joints. Every joint receives metaphoric names: the waist, elbow and shoulder joints, the first three (rotational) joints, provide the translation movement. *See Arm.*

**Joint-Interpolated Motion (movimiento interpolado en el espacio articular)**- A method of coordinating the movement of the joints, such that all joints arrive at the desired location simultaneously. This method of servo control produces a predictable path regardless of speed and results in the fastest pick and place cycle time for a particular move. *See Pick-and-Place Cycle, Servo-system.*

**Joint Torque (pares en articulares/en el espacio articular)**- The set of joint operations needed to enable the end-effector to apply the prescribed amount of force to the workspace.

## -K-

**Kinematics (cinemática)**- The study of motion without regard to the forces that cause it. *See Forward kinematics and Inverse kinematics.*

## -L-

**Ladle Gripper (cuchara de colada)**- An end-effector that acts as a spoon. It is commonly used to scoop up liquids, transfer it to a mold and pour the liquid into the mold. Commonly used for handling molten metal under hazardous conditions. *See End-Effector.*

**Laser** - Acronym for light amplification by stimulated emission of radiation. It is a device that produces a coherent monochromatic beam of light which is extremely narrow and focused but still within the visible light spectrum. This is commonly used as a non-contact sensor for robots. Robotic applications include: distance finding, identifying accurate locations, surface mapping, bar-code scanning, etc.

**Limited-Degree-of-Freedom Robot (robot limitado en número de grados de libertad)**- A robot able to position and orient its end-effector in fewer than six degrees of freedom.

**Link (eslabón)**- A rigid part of a manipulator that connects adjacent joints.

**Load Cycle Time (ciclo de trabajo)**- A manufacturing or assembly line process term that describes the complete time to unload the last workpiece and load the next one.

**Location (localización=posición+orientación)**- Describes the linear and angular position of an object. The linear position includes the azimuth, elevation, and range of the object. The angular position includes the roll, pitch, and yaw of the object. See Roll, Pitch, and Yaw. *See position.*

## -M-

**Machine Learning (máquina con capacidad de aprendizaje)**- The ability of a robot to take a new situation and apply knowledge from a previous situation to help solve a new problem. That is to say. it is the ability to acquire new data in memory and world model that will influence actions subsequent to addition or modification.

**Magnetic Detectors (detectores magnéticos)**- Robot sensors that can sense the presence of ferromagnetic material. Solid-state detectors with appropriate amplification and processing can locate a metal object to a high degree of precision. *See Sensor.*

**Manipulator (manipulador)**- A robotic mechanism consisting of an arm and an end-effector. It contains a series of segments, jointed or sliding relative to one another, for the purpose and moving objects. The manipulator usually has several degrees of freedom. It includes the arm, wrist, and end-effector. *See Arm, Wrist, and End-Effector, Master-Slave manipulator.*

**Master-Slave Manipulator (manipulador maestro-esclavo)**- A class of manipulators that operates in such a method: the master is held in a position by a human, and the slave duplicates the motion demonstrated by the master. Sometimes the duplicated motion is done with a change of scale in displacement or force. *See Teleoperation.*

**Material Processing Robot (robot de procesado (de materiales))**- A robot designed and programmed so that it can machine, cut, form, or change the shape, function or properties of materials it handles between the time the materials are first grasped and the time they are released in a manufacturing process

**Mobile Robot (robot móvil)**- A robot mounted on a movable platform. The robot is then able to move about its work environment by several different means such as wheels, legs, flying, etc. The motions of the robot are controlled by the robot's control system. *See Industrial Robot, and Robot.*

**Mobility (móvilidad)**- The ability to move from one physical location to another with reference to a fixed frame.

**Model Based Path Planning (planificación de trayectorias basada en un modelo)**- A method of path planning for a robot that uses a known world model. A computer simulation is executed to test and plan the robots actions. Several simulations may be needed to successfully plan the robots actions before the robot is allowed to execute its procedure. *See Path, Path Planning, and Sensor Based Path Planning.*

**Modularity (modularidad)**- The property of flexibility built into a computer system by assembling discrete units that can be easily joined to or arranged with other-parts or units.

**Module (módulo)**- Self contained component of a package. This component may contain sub-components known as sub-modules.

**Monitoring Controller (control de monitorización)**- A controller within the robot's control system that continually checks the processing of the robot and alerts the operator to possible malfunctions. This controller may also be used as a feedback device to enable a more autonomous robot to react to certain conditions it encounters during its tasks. *See Autonomous, and Feedback.*

**Motion Axis (eje/dirección de movimiento)**- The line defining the axis of motion either linear or rotary, of a segment of a manipulator.

**Motion Sensor (sensor de movimiento)**- A robotic sensing device which has the ability to detect an object that is in motion and determine the direction and speed of that motion.

## -N-

**Navigator/Executor (navegador)**- Informs the path planner where to go.

**Negative Image (negativo fotográfico)**- A picture signal having the polarity that is opposite to normal polarity and that results in an image in which white and black areas are reversed. This is sometimes used by image-sensors to determine edge detection or shape detection.

**Neural Network (red neural)**- An information-processing device that consists of a large number of simple non-linear processing modules, connected by elements that have information storage and programming functions. In general, the modules involve four functions: input/output, processing memory, and connections between different modules providing for information flow and control.

**-0-**

**Off-Line Programming System (sistema de programación off-line)**- A programming environment without the need to use the robot.

**On-Line Programming (programación on-line)**- A means of programming a robot using the robot in order to record the trajectories that must be performed latter by the robot.

**Open-Loop Control (control en lazo abierto)**- A robotic control system in which data flows only from the controller to the mechanism and does not flow from the mechanism back to the controller. There is no feedback. This does not allow self-correcting action that can be provided with feedback. *See Feedback, and Closed-Loop Control.*

**Optical Encoder (encoder/codificador óptico)**- A detection sensor that measures linear or rotary motion by detecting the movement of markings past a fixed beam of light. This can be used to count revolutions, identify parts, etc.

**Optical Proximity Sensors (sensores de proximidad ópticos)**- Robot sensors that measure visible or invisible light reflected from an object to determine distance. Lasers are used for greater accuracy.

**Orientation (orientación)**- The angle formed by the major axis of an object relative to a reference axis. It must be defined relative to a three-dimensional coordinate system. It's the angular position of an object with respect to the robot's reference system. *See Roll, Pitch, and Yaw.*

**-P-**

**Passive Compliant Robot (robot con acomodación pasiva)**- A compliant robot is allowed to modify its motion during the performance of a task due to the mechanical design of the robot. The compliance is given by a mechanical spring/damper attached to the robot's end effector, nothing matters with the controller.

**Path (camino/trayectoria)**- The locations (or points in three dimensional space, trajectory) through which a manipulator must pass in order to complete its tasks. *See Via Points, and Path Planning.*

**Path Planner (planificador de trayectorias)**- A module or set of modules that plans how to get from a starting point to an ending point that is by the navigator.

**Path Planning (planificación de trayectorias)**- the act of a robot, using programmed intelligence, to determine its path with regard to its current position and current world model. *See Path, Via Points, and World Model.*

**Payload (carga (sujetada por el robot))**- The total amount of weight that a robot can handle without suffering any harm, or malfunction to the robot. Factors include:

the sizing of the structural members, power transmission system, and actuators. The load placed on the actuators depends on the configuration of the robot, amount of time supporting the load, and inertial and velocity related forces. This is also called the **load capacity**. See *Actuator*.

**Pendant Control (consola de programación/teach pendant)**- A control panel, mounted on a pendant cable, that enables the human operator to stand in the most favorable position to observe, control, and record the desired movements in the robot's memory. See *Teach, and Teach pendant*.

**Pendant Teaching (almacenaje de posiciones por consola de programación)**- The mapping and recording of the position and orientation of a robot and/or manipulator system as the robot is manually moved in increments from an initial state along a path to a final goal state. The position and orientation of each critical point (joints, robot base) is recorded and stored in a database for each way point the robot passes through on its track toward its final goal. The robot may now repeat the recorded track on its own by following the path stored in the database.

**Perception (precepción)**- A robot's ability to sense its environment by sight, touch, or some other means and to understand it in terms of a task. For example, it is the ability to recognize an obstruction, or find a designated object in an arbitrary location.

**Pick and Place Cycle (ciclo de recogida y posicionamiento)**- The amount of time it takes for a manipulator to pick up an object and place it in a desired location, then return to its rest position. This includes time during the acceleration and deceleration phases of a particular task. The robots movement is controlled from one point location in space to another in a point-to-point (PTP) motion system. Each point is programmed into the robot's control memory and then played back during the work cycle.

**Pitch (cabeceo/cabezada/pitch)**- Rotation of the end-effector in a vertical plane around the end of the robot manipulator arm. See *Roll, and Yaw*.

**Pneumatically Driven Robot (robot neumático)**- Robot in which compressed air drives the mechanical arm. Usually used for pick-and-place activities where speed and precision are not critical.

**Point Set (conjunto de puntos)**- A set of way points established during the pendant teaching process that describe a unique path from an initial state to a final goal state.

**Point-to-Point (punto-a-punto)**- Manipulator motion in which a limited number of points along a projected path of motion is specified. The manipulator moves from point to point rather than a continuos smooth path

**Position (posición)**- the definition of an object's location in 3-D space, its orientation, and its velocity. Usually stipulated by a 3-D coordinate system using its X, Y, and Z coordinates.

**Position Control System (sistema de control de posición)**- A system that suppresses disturbances that perturb the system from the desired trajectory by calculating the velocity; and positioning necessary to counteract such disturbances.

**Potentiometer (potenciómetro)**- An encoding position sensing device for manipulators that produces a voltage proportional to the shaft position to measure joint displacement. Its uses are limited due to poor resolution, linearity, and noise susceptibility.

**Power Cylinder (cilindro (hidráulico) de potencia)**- A linear mechanical actuator consisting of a piston in a cylindrical volume and driven by high-pressure hydraulic fluid. *See Actuator.*

**Presence-sensing Safeguarding Device (dispositivo sensor de presencia)**- A device designed, constructed, and installed to create a sensing field to detect an intrusion into such field by people, robots, or objects. *See Sensor.*

**Prismatic joint (articulación/unión prismática)**- A joint of two nested links that slide onto or alongside of each other.**Programmable Robot** - A feature that allows a robot to be instructed to perform a sequence of steps and then to perform this sequence in a repetitive manner. It can then be reprogrammed to perform a different sequence of steps if desired.

**Prosthetic Device/Robot ( prótesis (robótica))**- A mechanical robot device that substitutes for lost manipulative or mobility functions of the human limbs, providing a substitute for human arms or legs when their function is lost. *See Exoskeleton.*

**Proximity Sensor (sensor de proximidad)**- A non-contact sensing device used to sense when objects are a short distance away, and determine the distance of the, object. Several types include: radio frequency, magnetic bridge, ultrasonic, and photoelectric. Commonly used for: high speed counting, sensing metal objects, level control, reading coding marks, and limit switches. *See Acoustic Proximity Sensor, and Inductive Sensor.*

## -R-

**Reachable volume (volumen alcanzable)**- The volume of space that a robot can reach in at least one orientation.

**Reactive (reactivo)**- Tending to react.

**Real-Time System (sistema en tiempo real)**- A computer system than can perform operations in deterministic way. That is to say, it can be determined how a operation will last a priori. No jitter, no lags.

**Reconfigurable (reconfigurable)**- Ability to modify the program to do something at least slightly different. Ability of a robot to modify itself to accommodate new or added (or subtracted) parts: joints, links, manipulators, etc.

**Rectangular-Coordinate Robot (robot rectangular/cartesiano)**- A robot whose manipulator's arm moves in linear motions along a set of Cartesian or rectangular axis. The work envelope forms the outline of a three dimensional rectangular figure. *See Work Envelope.*

**Reliability (fiabilidad)**- The percentage of time during which a robot can be expected to be in normal operation (not out of service for repair or maintenance). Also called the robot's uptime.

**Repeatability (repetibilidad)**- The specification of how accurately a manipulator can return to a "taught point" repeatedly. *See Teach and accuracy.*

**Resolution (resolución)**- It is the minimum 'distance' between two valid values due to the digital component.

**Reusable (recicitable/reutilizable)**- The quality to use software over again instead of being forced to rewrite it.

**Revolute Joint (articulación rotativa)**- The joints of a robot that are capable of rotary motion.

**Robot** - A re-programmable, multifunctional manipulator designed to move material, parts, tools, or specified devices through variable programmed motions for the performance of a variety of tasks. Common elements that make up a robot are: controller, manipulator, and end-effector. *See Manipulator, Controller, and End-Effector.*

**Robot Programming Language (lenguaje de programación de robot/robótico)**- An interface between a human user and a robot that relates humans commands to the robot.

**Roll (alabeo/roll)**- Rotation of the robot end-effector in a plane perpendicular to the end of the manipulator arm. *See Pitch, and Yaw.*

## -S-

**SCARA (Selective Compliance Assembly Robot Arm) Manipulator (manipulador SCARA (Brazo Robot de Ensamblaje de acomodación selectiva))**- An arm with parallel revolute joints that allow movement and orientation with respect to a family of planes parallel to the horizontal plane (as limited vertical motion is allowed).

**Sensor** - Instruments used as input devices for robots that enable it to determine aspects regarding the robot's environment, as well as the robot's own positioning. Sensors are used for robot information, and robot control.

**Sensor Based Path Planning (planificación de trayectoria basada en sensores)**- A method of path planning for a robot that uses a dynamic world model. The robot uses sensors to evaluate its environment. *See Model Based Path Planning.*

**Sensory Feedback (realimentación sensorial)**- Variable data measured by sensors and relayed to the controller in a closed-loop system. If the controller receives feedback that lies outside an acceptable range, then an error has occurred. The controller sends an error signal to the robot. The robot makes the necessary adjustments in accordance with the error signal.

---

**Servo-controlled Robot (robot servo-controlado)**- The control of a robot through the use of a closed-loop Servo-system, in which the position of the robot axis is measured by feedback devices and is stored in the controller's memory. *See Closed-Loop System, and Servo-system.*

**Servo-system (servo-sistema)**- A system in which the controller issues commands, the pressure motor drives the arm, and a sensor measures the motions and signals the amount of motions back to the controller. This process is continued until the arm is repositioned to the point requested. *See Servo-controlled Robot.*

**Simulation (simulación)**- A graphical computer program that represents the robot and its environment, which emulates the robot's behavior during a simulated run of the robot. This is used to determine a robot's behavior in certain situations, before actually commanding the robot to perform such tasks. Simulation items to consider are: the 3-D modeling of the environment, kinematics emulation, path-planning emulation, and simulation of sensors. *See Sensor, Forward Kinematics, and Robot.*

**Spatial Resolution (resolución espacial)**- The minimum or smallest dimension to which the robot system can define the workspace. This resolution determines the smallest error that can be sensed by the robot, as limited by the minimum resolution of the controller or the minimum resolving increment of the servo-system. *See Resolution, and Resolvers.*

**Spherical-Coordinate Robot (robot esférico)**- A robot whose construction consists of a horizontally rotating base, a vertically rotating shoulder, and a linear traversing arm connected in such a way that the work envelope traced by the end of the robot arm at full extension defines a sphere in space.

**Spline (Spline)**- A smooth, continuous function used to approximate a set of functions that are uniquely defined on a set of sub-intervals. The approximating function and the set of functions being approximated intersect at a sufficient number of points to insure a high degree of accuracy in the approximation. The purpose for the smooth function is to allow a robot manipulator to complete a task without jerky motion.

**State Space (espacio de estado)**- The set of all possible states available for use in solving a given problem.

**Strain (tensión/esfuerzo)**- A measure of the change in the size or shape of an object when subjected to different physical forces. This change is in reference to the objects original size and shape. *See Strain Gauges and Force Sensor.*

**Strain Gauges (galgas extensiometrías)**- Force sensors that usually consists of fine wires which can measure very small amounts of motion caused by the flexing of an object, or manipulator. They are used to measure strains and stresses in many types of components. *See Strain and Force Sensor.*

**-T-**

**Tachometer (tacómetro)**- A sensing device capable of sensing the speed at which a shaft is rotating. Generally used to determine revolutions per minute. May be used in conjunction with contouring systems as a supplemental control for governing feed-rates. See *Sensor*.

**Tactile Sensor (sensor táctil)**- A sensing device, normally used with the robot's hand or gripper, which senses physical contact with an object, thus giving the robot an artificial sense of touch. The sensors respond to contact forces that arise between themselves and solid objects. The object must actually be touched, unlike proximity sensors. See *Touch sensor, End-Effector, and Gripper*.

**Task Planner (planificador de tareas)**- A module or set of modules that plans how to perform a certain job.

**Teach** - To program a manipulator arm by manually guiding it through a series of motions.

**Teach Pendant (consola de programación/teach pendant)**- A handheld control box that is used by an operator to remotely guide a robot through the motions of its tasks. The motions are recorded by the robot control system for future playback. See *Accuracy, Pendant Control, Playback Accuracy, Repeatability, and Teach*.

**Teleoperational (teleoperacional)**- A method of controlling a robot through some means of remote control. This usually consists of a master-slave device that produces movements identical to or in direct proportion to actions or motions of the remotely operated human operator. The robot is entirely controlled by a human with this means of remote control. See *Master-Slave Manipulator*.

**Telerobot** - A robot that can be controlled remotely.

**Telerobotic (telerobótica)**- A kind of teleoperation in which the human operator acts as supervisor. The human sends information, to the robot's controller, about targets , constraints, requirements,... And he receives, from the robot's controller, data about the robot's status, impairments, errors, measurements (from sensors),...

**Thermistors (termistores)**- Used as temperature sensors within a robot's sensing system. Thermally sensitive resistors change in electrical resistance with variations in temperature.

**Thermocouple (termopares)**- Used as temperature sensors within a robot's sensing system. Consists of two dissimilar metals that produce an electromotive force roughly proportional to the temperature difference between their hot and cold junction ends.

**Time-of-Flight (tiempo de vuelo)**- The calculation of time it takes for a signal to reach and return from an object. The time it takes to reach an object is equal to the time to return from the object so the range is one-half the product of the velocity of the signal and the round-trip time.

**Tool (herramienta)**- A term used loosely to define a working apparatus mounted to the end of the robot arm, such' as a hand, gripper, welding torch, screw driver, etc. See Arm, Gripper, and End-Effector.

**Tool Frame (sistema coordinado solidario a la herramienta)**- A coordinate system attached to the end-effector of a robot (relative to the base frame).

**Touch Sensor (sensor de contacto)**- Sensing device, sometimes used with the robot's hand or gripper, which senses physical contact with an object, thus giving the robot an artificial sense of touch. The sensors respond to contact forces that arise between themselves and solid objects. See Tactile Sensor

**Trajectory Generation (generación de trayectoria)**- The computation of motion functions that allow the movement of joints- in a smooth controlled manner.

**Transducer (transductor)**- A device that converts energy from one form to another. Generally, it is a device that converts an input signal into an output signal of different form. It can also be thought of as a device that converts static signals detected in the environment (such as pressure) into an electrical signal that is sent to a robot's control system.

## -U-

**Ultrasound (ultrasonido)**- Acoustical radiation, with a frequency higher than the frequency range for audible sound.

**Uptime (tiempo de puesta en marcha)**- A period of time in which a robot, or production line is operating or available to operate, as opposed to downtime. See *Downtime*.

## -V-

**Vacuum Cup Hand (ventosa de vacío)**- An end-effector for a robot arm that is used to grasp light to moderate weight objects using suction, for manipulation. Such objects may include glass, plastic; etc. See *End-Effector*.

**Via Point (punto de paso)**- Intermediate locations (or points in space) through which a manipulator must pass en route to a particular destination. See *Path, and Path Planning*.

**Vision Sensor (sensor de visión)**- A sensor that identifies the shape, location, orientation, or dimensions of an object through visual feedback, such as a television camera. See *Feedback*.

**Voice Recognition (reconocimiento de voz)**- A system of sound sensors that translate the tones of the human voice into computer commands. This is sometimes used as a human-to-machine interface to the robot. The human operator simply speaks

commands to the robot's controller, the human voice is broken down into speech patterns and interpreted by the robot as commands.

## -W-

**Work Envelope (envolvente del espacio de trabajo)**- The edge of workspace. *See Workspace.*

**Work-piece (pieza de trabajo)**- Any part which is being worked, refined, or manufactured prior to its becoming a finished product.

**Workspace (espacio de trabajo)**- The volume of space within which the robot can perform given tasks.

**World Model (modelo de trabajo)**- A three dimensional representation of the robot's work environment, including objects and their position and orientation in this environment, which is stored in robot memory. As objects are sensed within the environment the robot's controller system continually updates the world model. Robots use this world model to aid in determining its actions in order to complete given tasks.

**Wrist (muñeca)**- A set of rotary joints between the arm and the robot end-effector that allow the end-effector to be oriented to the work-piece. In most cases the wrist can have degrees of freedom that enable it to grasp an object with roll, pitch, and yaw orientation. *See Arm, End-effector, Roll, Pitch, Yaw, and work piece.*

## -Y-

**Yaw (guiñada/yaw)**- Rotation of the end-effector in a horizontal plane around the end of the manipulator arm. *See Roll, and Pitch.*

# **Index**

---

- (MRAC), 216  
Jacobian Matrix, 169  
  
Absolute encoder, 98  
absolute error, 19  
Accuracy, 19, 88  
active on-line programming, 38  
actuator, 315–317, 319, 321, 327  
admittance control, 196  
AGVs, 8  
android, 306  
Approximation-point, 47  
arc welding, 319  
Arm, 16  
automatic guided vehicles, 8  
Automaton, 299  
automaton, 301, 303  
Automatos, 299  
avatar, 13  
  
Bandwidth, 90  
Basic Control, 193  
Basic robot actions, 34  
basic rotation homogeneous transformation matrix, 125  
basic translation matrix, 122  
bilateral control, 5  
bipolar steppermotor, 80  
  
Cartesian Coordinates, 106  
Cartesian robot, 320  
Cartesian space, 18  
Closed (Kinematic Chain, 16  
CNC Machine, 5  
Computer Numerically Controlled Machine Tool, 5  
Configuration of the robot arm, 18  
Connection Parameters, 138  
Continuous Trajectories, 186  
  
Coordinated or Synchronous Trajectories, 184  
Cylindrical Coordinates, 106  
Cylindrical joint, 56  
  
D-H method, 142  
Dead zone, 90  
Decay time, 90  
Degree of Freedom, 17, 53  
Denavit and Hartenberg method, 142  
Denavit and Hartenberg method using Craig's convention, 142  
Dexterous Workspace, 18  
direct cosine matrix, 110  
Direct jacobian matrix, xxxii  
DoF, 17  
duty cycle, 77  
Dynamic range, 88  
Dynamics, 137  
  
Euler Angles, 113  
Euler Parameters, 116  
Extended Teach-box programming, 39  
External sensors, 86  
  
feedback compensation, 210  
feedforward compensation, 210  
Flowchart, 43  
force feedback, 319  
formulation singularity, 115  
forward kinematics problem, 137  
  
Gain Scheduling, 215  
Guarded Motions, 101  
guiding by dummy/mannequin, 38  
  
Handling, Robot Applications, 25  
hexapods, 65  
homogeneous coordinates, 105, 118  
homogeneous transformation, 118

- homogeneous transformation matrix, 120  
 homogeneous translation matrix, 122  
 Hybrid Manipulators, 16  
 Hybrid Stepper Motor, 80  
 Hysteresis, 88
- I, Robot, 4, 304  
 Impedance control, 196  
 Incremental Encoder, 96  
 Index channel, 97  
 Industrial mobile robots, 8  
 Industrial Robot, 23, 24  
 Industrial robot, 53  
 industrial robots, 304  
 integral term, 205  
 Internal sensors, 86  
 Inverse jacobian matrix, xxxii  
 inverse kinematics problem, 137  
 Isaac Asimov, 4
- Join space, 18  
 Joint, 15  
 joint, 54, 315, 317, 319, 322, 327  
 Joint Angle, 140  
 Joint Parameters, 138  
 Joint-by-joint motion, 184  
 Joint-level programming, 35
- Karel Capek, 4  
 kinematics, 105, 137
- lag compensation, 205  
 laws of robotics, 5  
 laws of robotics, first, 5  
 laws of robotics, third, 5  
 laws of robotics, Zeroth, 5  
 laws of robotics, second, 5  
 lead compensation, 206  
 Linearity, 88  
 Link, 15  
 link length, 139  
 link offset, 140  
 Link Parameters, 138  
 link twist, 139  
 links, 138  
 location, 105, 118  
 Low-level programming, 35  
 manipulability index, 176
- Manipulator, 16  
 manipulator, 23, 315–332  
 Manipulator Analogy, 19  
 Master robot, 5, 11  
 master-slave guiding, 38  
 mechanical singularity, 115  
 Mechanism, 16  
 Model Reference Adaptive Control, 216  
 modified Denavit and Hartenberg method, 142  
 normal matrix, 110  
 object reference point, 106  
 Object-level programming, 35  
 Off-line robot programming, 40  
 Offset, 88  
 On-line programming, 37  
 Open (Kinematic) Chain, 15  
 original Denavit and Hartenberg method, 142  
 orthogonal matrix, 110  
 orthonormal matrices, 110  
 overshoot, 205, 206
- Parallel Manipulator, 16  
 Passive on-line programming, 37  
 Permanent Magnet Stepper Motor, 80  
 PI controller, 205  
 PID controller, 206  
 pitch, 67  
 Planar joint, 56  
 Point-to-point programming, 37  
 Point-to-point trajectories, 184  
 position and orientation, 105  
 Precision, 19  
 Prismatic joint, 56  
 Processing, Robot Applications, 25  
 Programmable Object Transfer Device, 5  
 Programming by guiding, 37  
 Programming by teaching, 37  
 programming pendant, 38  
 proportional-integral controller, 205  
 Pseudocode, 42  
 PTP, 37  
 PWM, 76
- Quadrature encoder, 97  
 Quaternions, 116

- R.U.R, 4  
R.U.R., 304  
Range, 88  
Reachable Workspace, 18  
redundant robots, 54  
Repeatability, 19  
Repeatability error, 19  
Reproducibility, 19  
Resolution, 19, 88  
Resolution error, 19  
Response time, 90  
Robot, 3, 6  
robot, 304, 315–332  
Robot subsystems, 6  
Robota, 4  
robotic, 323–325  
roll, 67  
Roll-Pitch-Roll mobile Euler Angles, 114  
Roll-Pitch-Yaw mobile Euler Angles, 114  
Rosum's Universal Robots, 4  
rotation matrix, 109  
Rotational joint, 56  
  
Saturation, 90  
Screw joint, 56  
Sensitivity, 88  
Sensitivity error, 88  
sensors, 316, 317, 319, 321–325, 328–331  
Serial Manipulator, 16  
Simultaneous joint motion, 184  
singular configuration, 176  
Situations, 47  
Slave robot, 5, 11  
speed control loop, 207  
Spherical Coordinates, 106  
Spherical joint, 56  
static error, 205  
steady state error, 205  
Subroutines, 43  
subroutines, 41, 48  
  
table of D-H parameters, 145  
Target-point, 47  
Task-level programming, 36  
teach pendant, 38  
teach-box, 38  
Teach-box Programming, 38  
telemanipulator, 5  
Teleoperation, 323  
Telepresence, 10  
Terminal devices, 101  
theoretical workspace, 156  
tool, 100, 318  
Trajectory Generation, 179  
trajectory programming, 37  
  
Unimation Inc, 305  
unipolar stepper motor, 81  
unit vectors, 108  
Universal joint, 56  
  
Variable Reluctance Stepper Motor, 80  
Via-point, 47  
  
welding, 331  
Work cell, 101  
Workspace, 18  
Workspace envelope, 18  
Workspace limitation, 18  
wrist, 315, 318, 319, 323, 332  
  
yaw, 67  
YAW-PITCH-ROLL, 113



# Bibliography

---

- [1] B. Appelhof. Design of haptic interface technology. Master's thesis, Dept of Mechanical Engineering, University of Twente, 2001.
- [2] A. Barrientos, L. F. Peñín, C. Balaguer, and R. Aracil. *FUNDAMENTOS DE ROBÓTICA*. Mc Graw Hil, 1997.
- [3] J.J. Craig. *INTRODUCTION TO ROBOTICS, MECHANICS AND CONTROL*. Addison Wesley, 1986.
- [4] A. J. Critchlow. *INTRODUCTION TO ROBOTICS*. Macmillan Publishing Co, 1985.
- [5] V. Etxebarria. *SISTEMAS DE CONTROL NO LINEAL Y ROBÓTICA*. Universidad del País Vasco, 1999.
- [6] K. S. Fu, R.C. González, and C. S G. Lee. *ROBÓTICA, CONTROL, DETECCIÓN, VISIÓN E INTELIGENCIA*. Mc Graw Hill, 1993.
- [7] Ashitava Ghosal. *ROBOTICS Fundamental Concepts and Analysis*. Oxford University Press, 2006.
- [8] P. Mckerrow. *INTRODUCTION TO ROBOTICS*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991.
- [9] A. Ollero. *ROBÓTICA, MANIPULADORES ROBÓTICOS Y ROBOTS MÓVILES*. Marcombo, 2001.
- [10] R. P. Paul. *ROBOT MANIPULATORS: MATHEMATICS, PROGRAMMING AND CONTROL*. Addison Wesley, 1981.
- [11] L. Sciavicco and B. Siciliano. *MODELLING AND CONTROL OF ROBOT MANIPULATORS*. Springer, 2000.
- [12] L. Sciavicco and B. Siciliano. *ROBOTICA INDUSTRIALE: MODELLISTICA E CONTROLLO DI MANIPOLATORI*. Mc Graw-Hill, 2000.
- [13] F. Torres, F. Pomares, P. Gil, S.T. Puente, and R. Aracil. *ROBOTS Y SISTEMAS SENSORIALES*. Prentice-Hall Inc, 2002.