

**Automation and Robotics**

Time: 3 hrs.

Max. Marks: 100

*Note: Answer any FIVE full questions, choosing ONE full question from each module.*

**Module-1**

- 1 a. What is automation? Explain basic elements of an automated system. (10 Marks)  
b. Briefly explain advanced automation functions. (10 Marks)

OR

- 2 a. What is the difference between a continuous variable and discrete variable? (05 Marks)  
b. Define sensor. Explain common measuring sensors used in automation system. (08 Marks)  
c. Briefly describe the three steps of the Analog-to-digital conversion process. (07 Marks)

**Module-2**

- 3 a. What is an automated production line? Explain general configuration of an automated production line and its system configuration. (10 Marks)  
b. Explain storage buffer in automated production line. (04 Marks)  
c. A 20 station transfer line has an ideal cycle time of  $T_c = 1.2$  mins. The probability of station breakdown per cycle is equal for all stations and  $P = 0.05$  break downs/cycle. Down time  $T_d = 0.8$  min. For each of the upper bound and lower bound, determine:  
(i) Frequency of line stops per cycle  
(ii) Average actual production rate  
(iii) Line efficiency (06 Marks)

OR

- 4 a. What are the four automated assembly system configurations? Explain. (10 Marks)  
b. Define automatic identification and data capture. Explain briefly bar code and RFID. (10 Marks)

**Module-3**

- 5 a. What is an Industrial Robot? Explain common Robot configurations with a neat diagram. (12 Marks)  
b. Define the following:  
(i) Work volume  
(ii) Resolution  
(iii) Accuracy  
(iv) Repeatability (08 Marks)

OR

- 6 a. Explain different sensors used in Robot. (08 Marks)  
b. Identify the Robotics application. (08 Marks)  
c. Write a note on end effector. (04 Marks)

**Module-4**

- 7 a. Describe positions, orientation and frames as related to manipulator. (10 Marks)  
b. Illustrate the interpretation used to map points between frames as operators using Translation and Rotation. (10 Marks)

OR

- 8 a. Write notes on :  
(i) Link description (10 Marks)  
(ii) Link connection description (10 Marks)  
b. Explain Actuator space, joint space and Cartesian space using the example of PUMA-560. (10 Marks)

**Module-5**

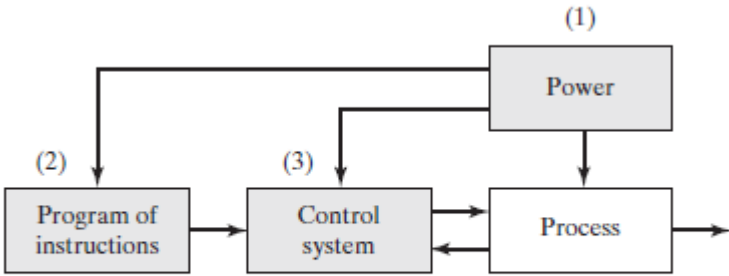
- 9 a. Define Robot Programming Language and explain the levels of Robot Programming. (10 Marks)  
b. List and explain requirements of a Robot Programming Language. (10 Marks)

OR

- 10 a. What are the problems peculiar to Robot Programming? Explain. (10 Marks)  
b. Explain central issues in OLP Systems. (10 Marks)

.....

Answer Key

Q.No	Solution
1.a)	<p><i>Automation</i> is the technology by which a process or procedure is accomplished without human assistance. It is implemented using a <i>program of instructions</i> combined with a <i>control system</i> that executes the instructions.</p> <p><b>Basic elements of an automated system</b>            An automated system consists of three basic elements:</p> <ol style="list-style-type: none"> <li>(1) <i>power</i> to accomplish the process and operate the system.</li> <li>(2) a <i>program of instructions</i> to direct the process, and</li> <li>(3) a control system to actuate the instructions.</li> </ol> <div style="text-align: center;">  </div> <p><b>Figure 4.2</b> Elements of an automated system: (1) power, (2) program of instructions, and (3) control systems.</p> <p style="text-align: center;">Fig.1 Basic elements of an automated system</p> <p><b>1. Power to accomplish the automated process</b>            An automated system is used to operate some process, and power is required to drive the process as well as the controls. The principal source of power in automated systems is electricity. Electric power has many advantages in automated as well as non-automated processes.</p> <p>In addition to driving the manufacturing process itself, power is also required for the following material handling functions:</p> <ul style="list-style-type: none"> <li>• <i>Loading and unloading the work unit.</i> All of the processes listed in Table 4.1 are accomplished on discrete parts. These parts must be moved into the proper position and orientation for the process to be performed, and power is required for this transport and placement function. At the conclusion of the process, the work unit must be removed. If the process is completely automated, then some form of mechanized power is used. If the process is manually operated or semi automated, then human power may be used to position and locate the work unit.</li> <li>• <i>Material transport between operations.</i> In addition to loading and unloading at a given operation, the work units must be moved between operations.</li> </ul>

	<p><b>Power for Automation.</b></p> <p>Above and beyond the basic power requirements for the manufacturing operation, additional power is required for automation. The additional power is used for the following functions:</p> <ul style="list-style-type: none"> <li>• <i>Controller unit.</i> Modern industrial controllers are based on digital computers, which require electrical power to read the program of instructions, perform the control calculations, and execute the instructions by transmitting the proper commands to actuating devices.</li> <li>• <i>Power to actuate the control signals.</i> The commands sent by the controller unit are carried out by means of electromechanical devices, such as switches and motors, called <i>actuators</i>. The commands are generally transmitted by means of low-voltage control signals. To accomplish the commands, the actuators require more power, and so the control signals must be amplified to provide the proper power level for the actuating device.</li> <li>• <i>Data acquisition and information processing.</i> In most control systems, data must be collected from the process and used as input to the control algorithms. In addition, for some processes, it is a legal requirement that records be kept of process performance and/or product quality. These data acquisition and record-keeping functions require power, although in modest amounts.</li> </ul> <p><b>2. Program of instructions</b></p> <p>The actions performed by an automated process are defined by a program of instructions. Whether the manufacturing operation involves low, medium, or high production, each part or product requires one or more processing steps that are unique to that part or product. These processing steps are performed during a work cycle. A new part is completed at the end of each work cycle (in some manufacturing operations, more than one part is produced during the work cycle: for example, a plastic injection molding operation may produce multiple parts each cycle using a multiple cavity mold). The particular processing steps for the work cycle are specified in a work cycle program, called <i>part programs</i> in numerical control.</p> <p><b>3. Control System</b></p> <p>The control element of the automated system executes the program of instructions. The control system causes the process to accomplish its defined function, which is to perform some manufacturing operation.</p> <p>The controls in an automated system can be either closed loop or open loop. A <b><i>closedloop control system</i></b>, also known as a <i>feedback control system</i>, is one in which the output variable is compared with an input parameter, and any difference between the two is used to drive the output into agreement with the input.</p>
1.b)	<p><b>Advanced automation functions</b></p> <p>Advanced automation functions include the following:</p> <ol style="list-style-type: none"> <li>(1) safety monitoring,</li> <li>(2) maintenance and repair diagnostics, and</li> <li>(3) error detection and recovery.</li> </ol>

### ***Safety Monitoring***

Safety monitoring in an automated system involves the use of sensors to track the system's operation and identify conditions and events that are unsafe or potentially unsafe. The safety monitoring system is programmed to respond to unsafe conditions in some appropriate way. Possible responses to various hazards include one or more of the following:

- (1) completely stopping the automated system,
- (2) sounding an alarm,
- (3) reducing the operating speed of the process, and
- (4) taking corrective actions to recover from the safety violation

The following list suggests some of the possible sensors and their applications for safety monitoring:

- ✓ Limit switches to detect proper positioning of a part in a workholding device so that the processing cycle can begin.
- ✓ Photoelectric sensors triggered by the interruption of a light beam; this could be used to indicate that a part is in the proper position or to detect the presence of a human intruder in the work cell.
- ✓ Temperature sensors to indicate that a metal work part is hot enough to proceed with a hot forging operation. If the work part is not sufficiently heated, then the metal's ductility might be too low, and the forging dies might be damaged during the operation.
- ✓ Heat or smoke detectors to sense fire hazards.
- ✓ Pressure-sensitive floor pads to detect human intruders in the work cell.
- ✓ Machine vision systems to perform surveillance of the automated system and its surroundings.

### ***Maintenance and Repair diagnostics***

Three modes of operation are typical of a modern maintenance and repair diagnostics subsystem:

1. *Status monitoring.* In the status monitoring mode, the diagnostic subsystem monitors and records the status of key sensors and parameters of the system during normal operation. On request, the diagnostics subsystem can display any of these values and provide an interpretation of current system status, perhaps warning of an imminent failure.

2. *Failure diagnostics.* The failure diagnostics mode is invoked when a malfunction or failure occurs. Its purpose is to interpret the current values of the monitored variables and to analyze the recorded values preceding the failure so that its cause can be identified.

3. *Recommendation of repair procedure.* In the third mode of operation, the subsystem recommends to the repair crew the steps that should be taken to effect repairs. Methods for developing the recommendations are sometimes based on the use of expert systems in which the collective judgments of many repair experts are pooled and incorporated into a computer program that uses artificial intelligence techniques.

### ***Error detection and recovery***

**Error Detection.** The error detection step uses the automated system's available

sensors to determine when a deviation or malfunction has occurred, interpret the sensor signal(s), and classify the error. Design of the error detection subsystem must begin with a systematic enumeration of all possible errors that can occur during system operation. The errors in a manufacturing process tend to be very application-specific. They must be anticipated in advance in order to select sensors that will enable their detection.

**Error Recovery.** Error recovery is concerned with applying the necessary corrective action to overcome the error and bring the system back to normal operation. The problem of designing an error recovery system focuses on devising appropriate strategies and procedures that will either correct or compensate for the errors that can occur in the process.

2a)

**TABLE 5.3 Comparison Between Continuous Control and Discrete Control**

Comparison Factor	Continuous Control in Process Industries	Discrete Control in Discrete Manufacturing Industries
Typical measures of product output	Weight measures, liquid volume measures, solid volume measures	Number of parts, number of products
Typical quality measures	Consistency, concentration of solution, absence of contaminants, conformance to specification	Dimensions, surface finish, appearance, absence of defects, product reliability
Typical variables and parameters	Temperature, volume flow rate, pressure	Position, velocity, acceleration, force
Typical sensors	Flow meters, thermocouples, pressure sensors	Limit switches, photoelectric sensors, strain gages, piezoelectric sensors
Typical actuators	Valves, heaters, pumps	Switches, motors, pistons
Typical process time constants	Seconds, minutes, hours	Less than a second

2.b) Sensor is a **transducer**, which is a device that converts a physical variable of one form into another form that is more useful for the given application. In particular, a **sensor** is a device that converts a physical stimulus or variable of interest (such as temperature, force, pressure, or displacement) into a more convenient form (usually an electrical quantity such as voltage) for the purpose of measuring the stimulus. The conversion process quantifies the variable, so that it can be interpreted as a numerical value.

The sensors used in robotics include a wide range of devices which can be divided into the following general categories:

- Tactile sensors
- Proximity and range sensors
- Miscellaneous sensors and sensor based systems
- Machine vision systems

**Tactile Sensors**

Tactile sensors are devices which indicate contact between themselves and some other solid object. Tactile sensing devices can be classified into two classes: Touch Sensors and Force sensors.

Touch sensors provide a binary output signal which indicates whether or not contact has been made with the object.

Force sensors (Stress Sensors) indicate not only the contact has been made with the object but

also the magnitude of the contact force between the two objects.

### **Touch sensors**

Touch sensors are used to indicate that contact has been made between two objects without regard to the magnitude of the contacting force. Included within this category are simple devices such as limit switches, micro-switches, and the like. The simpler devices are frequently used in the design of interlock systems in robotics. For example, they can be used to indicate the presence or absence of parts in a fixture or at the pick-up point along a conveyor.

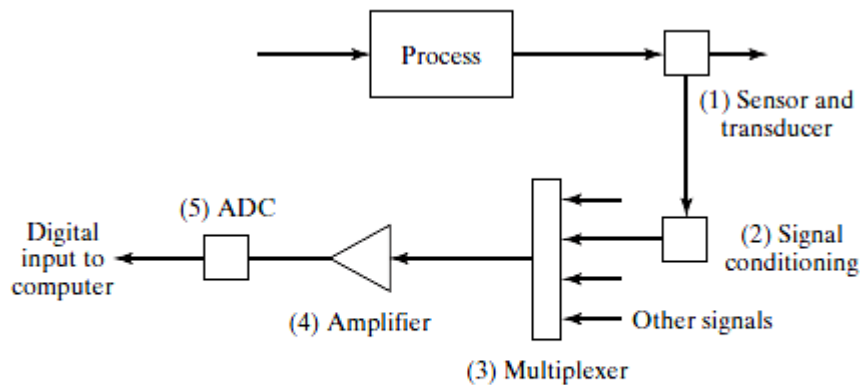
Another use for a touch sensing device would be as part of an inspection probe which is manipulated by the robot to measure dimensions on a work part. A robot with 6 degrees of freedom would be capable of accessing surfaces on the part that would be difficult for a three-axis coordinate measuring machine, the inspection system normally considered for such an inspection task. Unfortunately, the robot's accuracy would be a limiting factor in contact inspection work.

### **Force Sensors**

The capacity to measure the forces permits the robot to perform a number of tasks. These include the capability to grasp parts of different sizes in material handling, machine loading and assembly work, applying the appropriate level of force for the given part.

Force sensing in robotics can be accomplished in several ways. A commonly used technique is a "force sensing wrist". This consists of a special load-cell mounted between the gripper and the wrist. Another technique is to measure the torque being exerted by each joint. A third technique is to form an array of force-sensing elements so that the shape and other information about the contact surface can be determined.

2.c)



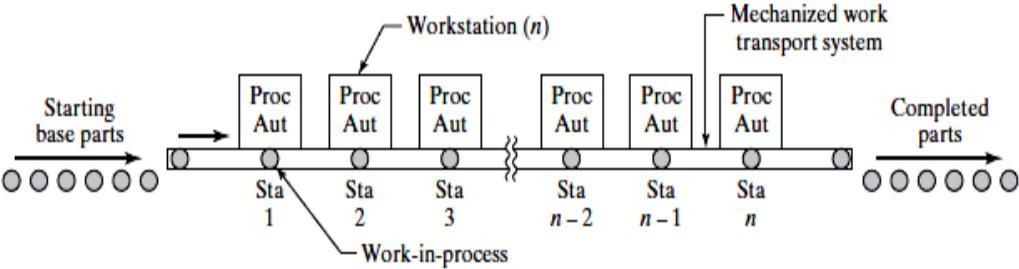
**Figure 6.9** Steps in analog-to-digital conversion of continuous analog signals from process.

Consider the operation of the ADC, which is the heart of the conversion process. Analog-to-digital conversion occurs in three steps: (1) sampling, (2) quantization, and (3) encoding. Sampling consists of converting the continuous signal into a series of discrete analog signals at periodic intervals, as shown in Figure 6.10. In quantization, each discrete analog signal is assigned to one of a finite number of previously defined amplitude levels. The amplitude levels are discrete values of voltage ranging over the full scale of the ADC

In the encoding step, the discrete amplitude levels obtained during quantization are

converted into digital code, representing the amplitude level as a sequence of binary digits.

3.a) An automated production line consists of multiple workstations that are automated and linked together by a work handling system that transfers parts from one station to the next, as depicted in Figure 16.1. A raw work part enters one end of the line, and the processing steps are performed sequentially as the part progresses forward (from left to right in the drawing). The line may include inspection stations to perform intermediate quality checks. Also, manual stations may be located along the line to perform certain operations that are difficult or uneconomical to automate. Each station performs a different operation, so all operations must be performed to complete each work unit. Multiple parts are processed simultaneously on the line, one part at each station. In the simplest form of production line, the number of parts on the line at any moment is equal to the number of workstations, as in the figure. In more complicated lines, provision is made for temporary parts storage between stations, in which case there are more parts than stations.



**Figure 16.1** General configuration of an automated production line. Key: Proc = processing operation, Aut = automated workstation.

3.b) A storage buffer is a location in the production line where parts can be collected and temporarily stored before proceeding to downstream workstations. The storage buffers can be manually operated or automated. When it is automated, a storage buffer consists of a mechanism to accept parts from the upstream workstation, a place to store the parts, and a mechanism to supply parts to the downstream station. A key parameter of a storage buffer is its storage capacity, that is, the number of work parts it can hold. Storage buffers may be located between every pair of adjacent stations, or between line stages containing multiple stations.

4.a) Automated assembly systems can be classified according to physical configuration. The principal configurations, illustrated in Figure 17.1, are (a) in-line assembly machine, (b) dial-type assembly machine, (c) carousel assembly system, and (d) single-station assembly machine.

The in-line assembly machine, Figure 17.1(a), is a series of automatic workstations located along an in-line transfer system. It is the assembly version of the machining transfer line. Synchronous and asynchronous transfer systems are the common means of transporting base parts from station to station with the in-line configuration.

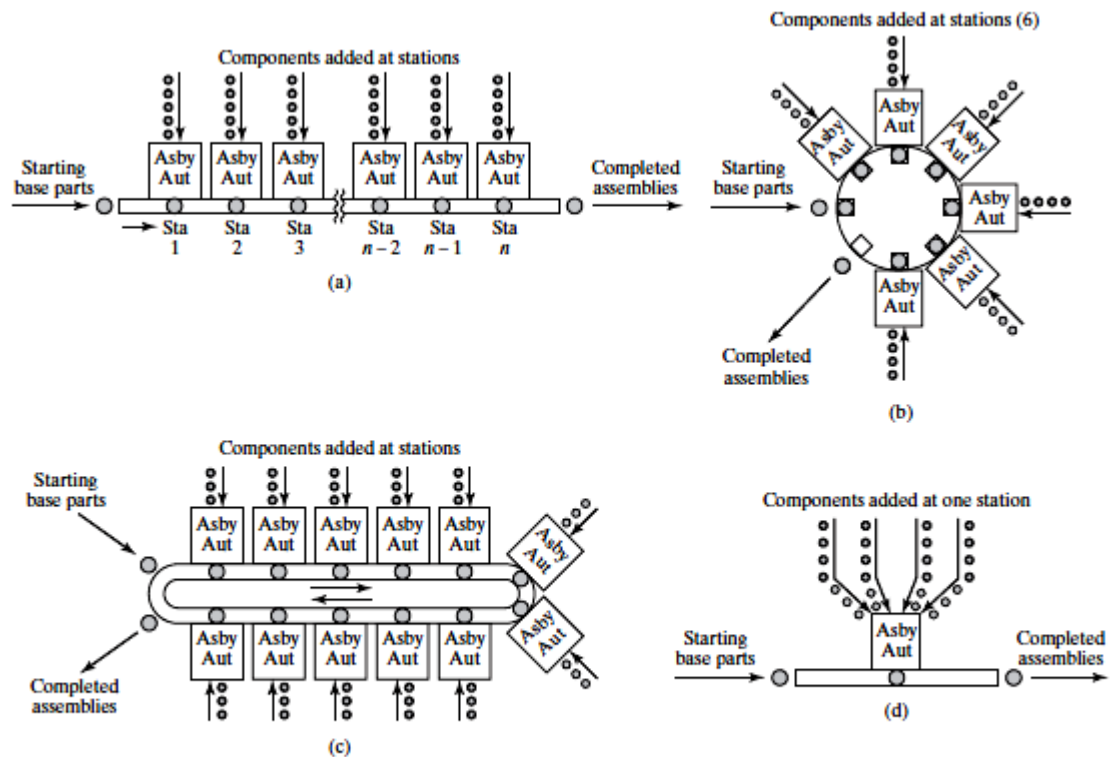
In the typical application of the dial-type machine, Figure 17.1(b), base parts are loaded onto fixtures or nests attached to the circular dial. Components are added and/or joined to the base part at the various workstations located around the periphery of the dial. The dial-indexing machine operates with a synchronous or intermittent motion, in



which the cycle consists of the service time plus indexing time. Dial-type assembly machines are sometimes designed to use a continuous rather than intermittent motion. This is common in beverage bottling and canning plants, but not in mechanical and electronics assembly.

The carousel assembly system represents a hybrid between the circular work flow of the dial-type assembly machine and the straight work flow of the in-line system. The carousel configuration can be operated with continuous, synchronous, or asynchronous transfer mechanisms to move the work around the carousel.

In the single-station assembly machine, Figure 17.1(d), assembly operations are performed on a base part at a single location. The typical operating cycle involves the placement of the base part at a stationary position in the workstation, the addition of components to the base, and finally the removal of the completed assembly from the station. An important application of single-station assembly is the component placement machine, widely used in the electronics industry to populate components onto printed circuit boards.



**Figure 17.1** Types of automated assembly systems: (a) in-line, (b) dial-type, (c) carousel, and (d) single station.

4.b) **Automatic identification and data capture (AIDC)** refers to technologies that provide direct entry of data into a computer or other microprocessor-controlled system without using a keyboard.

### **BAR CODE TECHNOLOGY**

Bar codes divide into two basic types: (1) linear, in which the encoded data are read using a linear sweep of the scanner, and (2) two-dimensional, in which the encoded data must

be read in both directions.



**Figure 12.1** Two forms of linear bar codes are (a) width-modulated, exemplified here by the Universal Product Code, and (b) height-modulated, exemplified here by Postnet, used by the U.S. Postal Service.

### **Linear (One-Dimensional) Bar Codes**

Linear bar codes are the most widely used automatic identification and data capture technique. There are actually two forms of linear bar code symbologies, illustrated in Figure 12.1: (a) width-modulated, in which the symbol consists of bars and spaces of varying width; and (b) height-modulated, in which the symbol consists of evenly spaced bars of varying height. The only significant application of the height-modulated bar code symbologies is in the U.S. Postal Service for ZIP code identification, so the discussion here focuses on the width-modulated bar codes, which are used widely in retailing and manufacturing.

In linear width-modulated bar code technology, the symbol consists of a sequence of wide and narrow colored bars separated by wide and narrow spaces (the colored bars are usually black and the spaces are white for high contrast). The pattern of bars and spaces is coded to represent numeric or alphanumeric character.

Bar code readers interpret the code by scanning and decoding the sequence of bars. The reader consists of the scanner and decoder. The scanner emits a beam of light that is swept past the bar code (either manually or automatically) and senses light reflections to distinguish between the bars and spaces. The light reflections are sensed by a photo detector, which converts the spaces into an electrical signal and the bars into absence of an electrical signal. The width of the bars and spaces is indicated by the duration of the corresponding signals.

### **RFID**

In radio frequency identification, an identification tag or label containing electronically encoded data is attached to the subject item, which can be a part, product, or container (e.g., carton, tote pan, pallet). The identification tag consists of an integrated circuit chip and a small antenna, as pictured in Figure 12.8. These components are usually enclosed in a protective plastic container or are imbedded in an adhesive-backed label that is attached to item. The tag is designed to satisfy the Electronic Product Code (EPC) standard, which is the RFID counterpart to the Universal Product Code (UPC) used in bar codes. The tag communicates the encoded data by RF to a reader or interrogator as the item is brought into the reader's proximity. The reader can be portable or stationary. It decodes and confirms the RF signal before transmitting the associated data to a collection

computer.

5.a) A robot is a reprogrammable multi functional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of variety of tasks.

1. Polar configuration
2. Cylindrical Configuration
3. Cartesian Configuration
4. Jointed Arm configuration

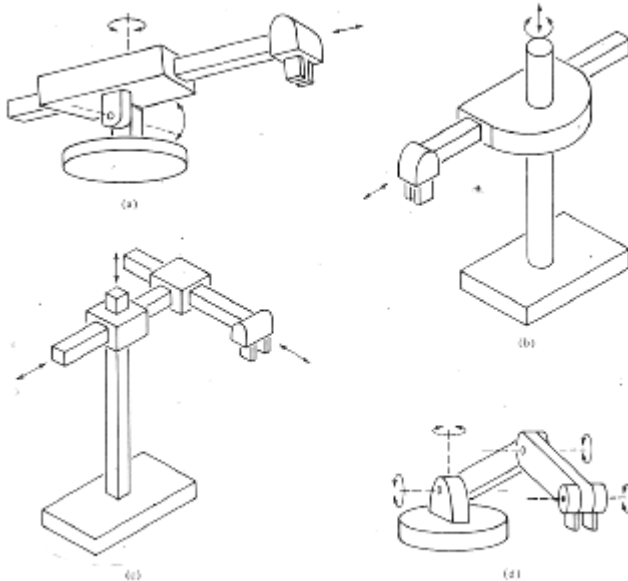


Figure 2-1 The four basic robot anatomies: (a) polar, (b) cylindrical, (c) cartesian, and (d) jointed-arm. (Reprinted from Reference [7])

### 1. Polar configuration/Spherical configuration

Notation: [LTR]: Linear, Twisting and Rotational joint

This configuration also called as Polar coordinate configuration. It goes by the name “spherical coordinate” also because the workspace within which it can move its arm is a partial sphere as shown in figure. The robot has a rotary base and a pivot that can be used to raise and lower a telescoping arm.

- i) Operate within a **spherical** work volume
- ii) Has 1 prismatic and 2 revolute axes.
- iii) First motion is a base rotation, Second motion correspond to an elbow rotation and Third

motion is radial or in-out motion

iv) Elbow rotation and arm reach limit the design of full spherical motion.

v) Rarely used in industries but common in automated cranes.

### 2. Cylindrical Configuration

Notation: [TLL]: Twisting, Linear and Linear.

This also has 3 degrees of freedom, 2 prismatic and 1 revolute joints. It moves linearly along X and Y axes and rotation about at its base i.e. Z- axis. The robot body is a vertical column that swivels about a vertical axis. The arm consists of several orthogonal slides

which allow the arm to be moved up or down and in and out with respect to the body. This is illustrated schematically in figure.

Features:

- i) Operate within a **cylindrical** work volume
- ii) 2 prismatic and 1 revolute joints.
- iii) Position is specified by Y value ( height) extension of arm X axis and angle of rotation of Z axis ( $\theta$ )
- iv) Recommended for pick and place operation such as machine loading and unloading.
- v) Lower repeatability and accuracy
- vi) Require more sophisticated control
- vii) Rigid structure & high lift-carrying capacity

### 3. Cartesian / Rectangular configuration

Notation: [LOO]: Linear, Orthogonal, Orthogonal

Cartesian configuration is also called as **Rectilinear or Rectangular** configuration as the joints allow only translational or linear relative motion between the adjacent links of the joint. A robot using such a configuration is called as X-Y-Z robot. Other names are xyz robot or Rectilinear robot or **Gantry robot**. Any point in X, Y and Z coordinate system can be reached using this configuration. By appropriate movements of these slides, the robot is capable of moving its arm at any point within its three dimensional rectangular spaced work space.

Features:

- i) Operate within a **rectangular** work volume
- ii) Three prismatic joints are used.
- iii) The position is specified by X, Y and Z locations.
- iv) Easy to visualize motion
- v) Easy to program the motions
- vi) Adapted in gantry crane and CNC milling machines.
- vii) Gantry type can handle heavy loads.
- viii) Addition axes can be incorporated to the wrist action.
- ix) Difficult to protect the sliding axes from contaminants such as dust and moisture as it is open.

### 4. Revolute / Articulate / Jointed-arm configuration:

Notation: [TRR]: Twisting, Rotational and Rotational joint

It is combination of cylindrical and articulated configurations. This is similar in appearance to the human arm, as shown in fig. the arm consists of several straight members connected by joints which are analogous to the human shoulder, elbow, and

	<p>wrist. The robot arm is mounted to a base which can be rotated to provide the robot with the capacity to work within a quasi-spherical space.</p> <p>Features:</p> <ul style="list-style-type: none"> <li>i) Operate within a <b>quasi-spherical</b> work volume.</li> <li>ii) All 3 are revolute joints.</li> <li>iii) Can reach above, below and around obstacles.</li> <li>iv) Joints can be sealed easily.</li> <li>v) Difficult to calculate angular motion of the axis for a given top or end motion.</li> </ul>
5.b)	<p><b>Robot Work Volume</b></p> <p>It is the term that refers to the space within which the robot can manipulate its wrist end. The convention of using the wrist end to define the robot's work volume is adopted to avoid the complication of different sizes of end effectors that might be attached to the robot's wrist.</p> <p><b>Spatial resolution</b></p> <p>The spatial resolution of a robot is the smallest increment of movement into which the robot can divide its work volume. Spatial resolution depends on two factors: the system's control resolution and the robot's mechanical inaccuracies. It is easiest to conceptualize these factors in terms of a robot with 1 degree of freedom.</p> <p><b>Accuracy</b></p> <p>Accuracy refers to a robot's ability to position its wrist end at a desired target point within the work volume. The accuracy of a robot can be defined in terms of spatial resolution because the ability to achieve a given target-point depends on how closely the robot can define the control increments for each of its joint motions.</p> <p><b>Repeatability</b></p> <p>Repeatability is concerned with the robot's ability to position its wrist or an end effector attached to its wrist at a point in space that had previously been taught to the robot. Repeatability and accuracy has to two different aspects of the robot's precision.</p>
6.a)	<p>The sensors used in robotics include a wide range of devices which can be divided into the following general categories:</p> <ul style="list-style-type: none"> <li>a. Tactile sensors</li> <li>b. Proximity and range sensors</li> <li>c. Miscellaneous sensors and sensor based systems</li> <li>d. Machine vision systems</li> </ul> <p><b>Tactile Sensors</b></p> <p>Tactile sensors are devices which indicate contact between themselves and some other solid object. Tactile sensing devices can be classified into two classes: Touch Sensors and Force sensors.</p> <p>Touch sensors provide a binary output signal which indicates whether or not contact has been made with the object.</p> <p>Force sensors (Stress Sensors) indicate not only the contact has been made with the object but also the magnitude of the contact force between the two objects.</p> <p><b>Touch sensors</b></p> <p>Touch sensors are used to indicate that contact has been made between two objects without regard to the magnitude of the contacting force. Included within this category are simple</p>

	<p>devices such as limit switches, micro-switches, and the like. The simpler devices are frequently used in the design of interlock systems in robotics. For example, they can be used to indicate the presence or absence of parts in a fixture or at the pick-up point along a conveyor.</p> <p>Another use for a touch sensing device would be as part of an inspection probe which is manipulated by the robot to measure dimensions on a work part. A robot with 6 degrees of freedom would be capable of accessing surfaces on the part that would be difficult for a three-axis coordinate measuring machine, the inspection system normally considered for such an inspection task. Unfortunately, the robot's accuracy would be a limiting factor in contact inspection work.</p> <p><b>Force Sensors</b></p> <p>The capacity to measure the forces permits the robot to perform a number of tasks. These include the capability to grasp parts of different sizes in material handling, machine loading and assembly work, applying the appropriate level of force for the given part.</p> <p>Force sensing in robotics can be accomplished in several ways. A commonly used technique is a "force sensing wrist". This consists of a special load-cell mounted between the gripper and the wrist. Another technique is to measure the torque being exerted by each joint. A third technique is to form an array of force-sensing elements so that the shape and other information about the contact surface can be determined.</p>
6.b)	<p>Robots are used in a wide field of applications in industry. Most of the current applications are in manufacturing. The applications can usually be classified into one of the following categories: (1) material handling, (2) processing operations, and (3) assembly and inspection.</p> <p><b>Material Handling Applications</b></p> <p>In material handling applications, the robot moves materials or parts from one place to another. To accomplish the transfer, the robot is equipped with a gripper that must be designed to handle the specific part or parts to be moved. Included within this application category are (1) material transfer and (2) machine loading and/or unloading. In many material handling applications, the parts must be presented to the robot in a known position and orientation. This requires some form of material handling device to deliver the parts into the work cell in this position and orientation.</p> <p><b>Material Transfer.</b></p> <p>These applications are ones in which the primary purpose of the robot is to move parts from one location to another. In many cases, reorientation of the part is accomplished during the move. The basic application in this category is called a pick-and-place operation, in which the robot picks up a part and deposits it at a new location.</p> <p>Industrial robot applications of machine loading and/or unloading include the following processes:</p> <ul style="list-style-type: none"> <li>• <i>Die casting.</i> The robot unloads parts from the die casting machine. Peripheral operations sometimes performed by the robot include dipping the parts into a water bath for cooling.</li> <li>• <i>Plastic molding.</i> Plastic molding is similar to die casting. The robot unloads molded parts from the injection molding machine.</li> <li>• <i>Metal machining operations.</i> The robot loads raw blanks into the machine tool and unloads finished parts from the machine.</li> <li>• <i>Forging.</i> The robot typically loads the raw hot billet into the die, holds it during the forging strikes, and removes it from the forge hammer. The hammering action and</li> </ul>

the risk of damage to the die or end effector are significant technical problems.

- *Pressworking*. Human operators work at considerable risk in sheetmetal pressworking operations because of the action of the press. Robots are used to substitute for the workers to reduce the danger. In these applications, the robot loads the blank into the press, then the stamping operation is performed, and the part falls out of the machine into a container.
- *Heat-treating*. These are often relatively simple operations in which the robot loads and/or unloads parts from a furnace.

6.c) **ROBOT END EFFECTORS**

An end effector is a device that attaches to the wrist of the robot arm and enables the general-purpose robot to perform a specific task. It is sometimes referred to as the robot's "hand."

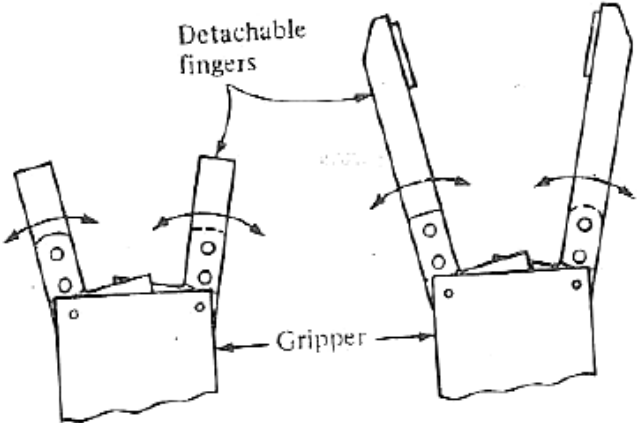
Types of end effectors

The End effectors can be divided into two major categories:

1. Grippers
2. Tools

Grippers are and effectors used to grasp and hold objects. The objects are generally work parts that are to be moved by the robot. These part handling applications include machine loading and unloading, picking parts from a conveyor and arranging parts onto a pallet.

Grippers can be classified as single, double or multiple. The single gripper is distinguished with only one grasping device mounted to the robot's wrist.



7.a) **Positions, Orientations and Frames**

**DESCRIPTIONS: POSITIONS, ORIENTATIONS, AND FRAMES**

A description is used to specify attributes of various objects with which a manipulation system deals. These objects are parts, tools, and the manipulator itself. In this section, we discuss the description of positions, of orientations, and of an entity that contains both of these descriptions: the frame.

**Description of a position**

Once a coordinate system is established, we can locate any point in the universe with a 3 x 1 position vector. Because we will often define many coordinate systems in addition to the universe coordinate system, vectors must be tagged with information identifying which coordinate system they are defined within. In this book, vectors are written with a leading superscript indicating the coordinate system to which they are referenced (unless it is clear from context)—for example,  ${}^A P$ . This means that the components of  ${}^A P$  have numerical values that indicate distances along the axes of  $\{A\}$ . Each of these distances along an axis can be thought of as the result of projecting the vector onto the corresponding axis. Figure 2.1 pictorially represents a coordinate system,  $\{A\}$ , with three mutually orthogonal unit vectors with solid heads. A point  $A P$  is represented as a vector and can equivalently be thought of as a position in space, or simply as an ordered set of three numbers. Individual elements of a vector are given the subscripts  $x$ ,  $y$ , and  $z$ :

$${}^A P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \quad (2.1)$$

### Description of an orientation

In order to describe the orientation of a body, we will attach a coordinate system to the body and then give a description of this coordinate system relative to the reference system.

In Fig. 2.2, coordinate system (B) has been attached to the body in a known way. A description of  $\{B\}$  relative to  $\{A\}$  now suffices to give the orientation of the body. Thus, positions of points are described with vectors and orientations of bodies are described with an attached coordinate system. One way to describe the body-attached coordinate system, (B), is to write the unit vectors of its three principal axes<sup>2</sup> in terms of the coordinate system  $\{A\}$ .

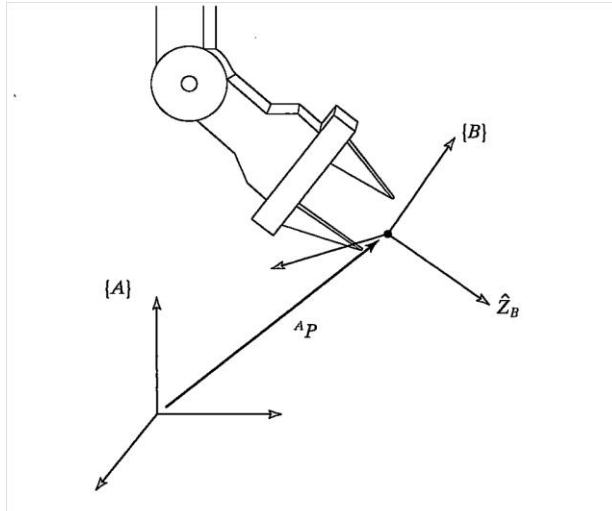


FIGURE 2.2: Locating an object in position and orientation.



We denote the unit vectors giving the principal directions of coordinate system  $\{B\}$  as  $\hat{X}_B$ ,  $\hat{Y}_B$ , and  $\hat{Z}_B$ . When written in terms of coordinate system  $\{A\}$ , they are called  ${}^A\hat{X}_B$ ,  ${}^A\hat{Y}_B$ , and  ${}^A\hat{Z}_B$ . It will be convenient if we stack these three unit vectors together as the columns of a  $3 \times 3$  matrix, in the order  ${}^A\hat{X}_B$ ,  ${}^A\hat{Y}_B$ ,  ${}^A\hat{Z}_B$ . We will call this matrix a **rotation matrix**, and, because this particular rotation matrix describes  $\{B\}$  relative to  $\{A\}$ , we name it with the notation  ${}^A R_B$  (the choice of leading sub- and superscripts in the definition of rotation matrices will become clear in following sections):

$${}^A R_B = [ {}^A\hat{X}_B \quad {}^A\hat{Y}_B \quad {}^A\hat{Z}_B ] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \quad (2.2)$$

In summary, a set of three vectors may be used to specify an orientation. For convenience, we will construct a  $3 \times 3$  matrix that has these three vectors as its columns. Hence, whereas the position of a point is represented with a vector, the orientation of a body is represented with a matrix.

### Description of a frame

The information needed to completely specify the whereabouts of the manipulator hand in Fig. 2.2 is a position and an orientation. The point on the body whose position we describe could be chosen arbitrarily, however. For convenience, the point whose position we will describe is chosen as the origin of the body-attached frame. The situation of a position and an orientation pair arises so often in robotics that we define an entity called a frame, which is a set of four vectors giving position and orientation information. For example, in Fig. 2.2, one vector locates the fingertip position and three more describe its orientation. Equivalently, the description of a frame can be thought of as a position vector and a rotation matrix. Note that a frame is a coordinate system where, in addition to the orientation, we give a position vector which locates its origin relative to some other embedding frame.

For example, frame

$\{B\}$  is described by  ${}^A R_B$  and  ${}^A P_{BORG}$ , where  ${}^A P_{BORG}$  is the vector that locates the origin of the frame  $\{B\}$ :

$$\{B\} = \{ {}^A R_B, {}^A P_{BORG} \}. \quad (2.8)$$

In Fig. 2.3, we introduce a graphical representation of frames, which is convenient in visualizing frames. A frame is depicted by three arrows representing unit vectors defining the principal axes of the frame. An arrow representing a vector is drawn from one origin to another. This vector represents the position of the origin at the head of the arrow in terms of the frame at the tail of the arrow. The direction of this locating arrow tells us, for example, in Fig. 2.3, that  $\{C\}$  is known relative to  $\{A\}$  and not vice versa.

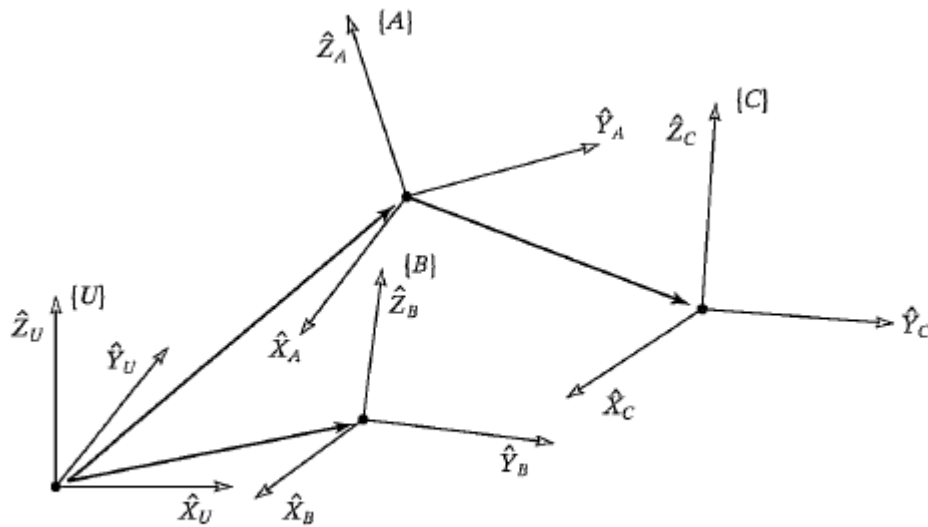


FIGURE 2.3: Example of several frames.

7.b)

### Mappings involving rotated frames

Section 2.2 introduced the notion of describing an orientation by three unit vectors denoting the principal axes of a body-attached coordinate system. For convenience we stack these three unit vectors together as the columns of a  $3 \times 3$  matrix. We will call this matrix a rotation matrix, and if this particular rotation matrix describes  $\{B\}$  relative to  $\{A\}$ , we name it with the notation  ${}^A_B R$ .

Note that by our definition, the columns of a rotation matrix all have unit magnitude, and further, these unit vectors are orthogonal. As we saw earlier, a consequence of this is that

$${}^A_B R = {}^B_A R^{-1} = {}^B_A R^T. \quad (2.10)$$

Therefore, since the columns of  ${}^A_B R$  are the unit vectors of  $\{B\}$  written in  $\{A\}$ , then the rows of  ${}^A_B R$  are the unit vectors of  $\{A\}$  written in  $\{B\}$ .

So a rotation matrix can be interpreted as a set of three column vectors or as a set of three row vectors as follows:

$${}^A_B R = \begin{bmatrix} {}^A \hat{X}_B & {}^A \hat{Y}_B & {}^A \hat{Z}_B \end{bmatrix} = \begin{bmatrix} {}^B \hat{X}_A^T \\ {}^B \hat{Y}_A^T \\ {}^B \hat{Z}_A^T \end{bmatrix}. \quad (2.11)$$

8.a)

A manipulator may be thought of as a set of bodies connected in a chain by joints. These bodies are called links. Joints form a connection between a neighboring pair of links. The term **lower pair** is used to describe the connection between a pair of bodies when the relative motion is characterized by two surfaces sliding over one another. Figure 3.1 shows the six possible lower pair joints.

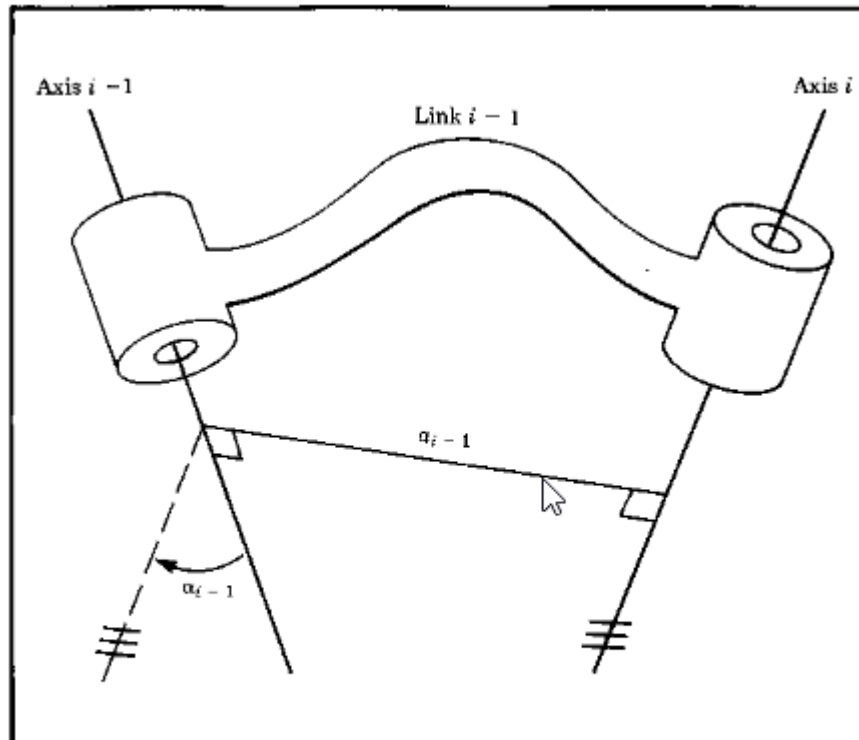
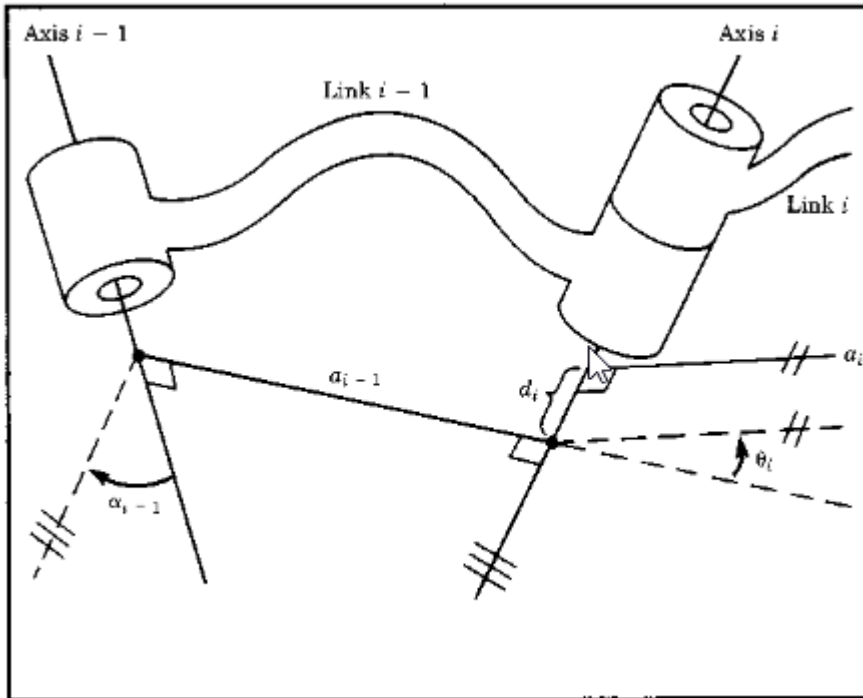


FIGURE 3.2 The kinematic function of a link is to maintain a fixed relationship between the two joint axes it supports. This relationship can be described with two parameters, the link length,  $a$ , and the link twist,  $\alpha$ .

For any two axes in 3-space there exists a well-defined measure of distance between them. This distance is measured along a line which is mutually perpendicular to both axes. This mutual perpendicular always exists and is unique except when both axes are parallel, in which case there are many mutual perpendiculars of equal length. Figure 3.2 shows link  $i - 1$  and the mutually perpendicular line along which the **link length**,  $a_{i-1}$ , is measured. Another way to visualize the link parameter  $a_{i-1}$  is to imagine an expanding cylinder whose axis is the joint  $i - 1$  axis — when it just touches joint axis  $i$  the radius of the cylinder is equal to  $a_{i-1}$ .

The second parameter needed to define the relative location of the two axes is called the **link twist**. If we imagine a plane whose normal is the mutually perpendicular line just constructed, we can project both axes  $i - 1$  and  $i$  onto this plane and measure the angle between them. This angle is measured from axis  $i - 1$  to axis  $i$  in the right-hand sense about  $a_{i-1}$ .<sup>†</sup> We will use this definition of the twist of link  $i - 1$ ,  $\alpha_{i-1}$ . In Fig. 3.2,  $\alpha_{i-1}$  is indicated as the angle between axis  $i - 1$  and axis  $i$  (the lines with the triple hash marks are parallel). In the case of intersecting axes, twist is measured in the plane containing both axes, but the sense of  $\alpha_{i-1}$  is lost. In this special case, one is free to assign the sign of  $\alpha_{i-1}$  arbitrarily.



### Intermediate links in the chain

Neighboring links have a common joint axis between them. One parameter of interconnection has to do with the distance along this common axis from one link to the next. This parameter is called the **link offset**. The offset at joint axis  $i$  is called  $d_i$ . The second parameter describes the amount of rotation about this common axis between one link and its neighbor. This is called the **joint angle**,  $\theta_i$ .

Figure 3.4 shows the interconnection of link  $i-1$  and link  $i$ . Recall that  $a_{i-1}$  is the mutual perpendicular between the two axes of link  $i-1$ . Likewise  $a_i$  is the mutual perpendicular defined for link  $i$ . The first parameter of interconnection is the link offset,  $d_i$ , which is the signed distance measured along the axis of joint  $i$  from the point where  $a_{i-1}$  intersects the axis to the point where  $a_i$  intersects the axis. The offset  $d_i$  is indicated in Fig. 3.4. The link offset  $d_i$  is variable if joint  $i$  is prismatic.

### First and last links in the chain

Link length,  $a_i$ , and link twist,  $\alpha_i$ , depend on joint axes  $i$  and  $i+1$ . Hence  $a_1$  through  $a_{n-1}$  and  $\alpha_1$  through  $\alpha_{n-1}$  are defined as discussed above in this section. At the ends of the chain, it will be our convention to assign zero to these quantities. That is,  $a_0 = a_n = 0.0$  and  $\alpha_0 = \alpha_n = 0.0$ . Link offset,  $d_i$ , and joint angle,  $\theta_i$ , are well defined for joints 2 through  $n-1$  according to the conventions discussed above in this section. If joint 1 is revolute, the zero position for  $\theta_1$  may be chosen arbitrarily and  $d_1 = 0.0$  will be our convention. Similarly, if joint 1 is prismatic, the

zero position of  $d_1$  may be chosen arbitrarily, and  $\theta_1 = 0.0$  will be our convention. Exactly the same statements apply to joint  $n$ .

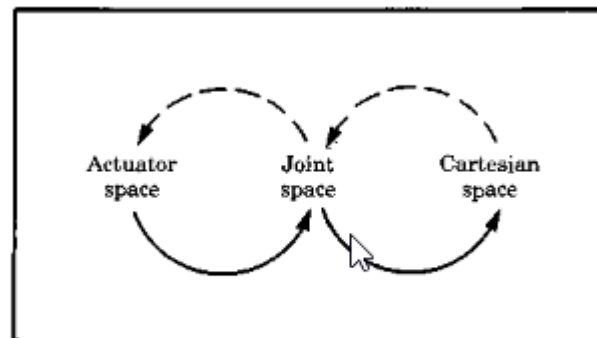
These conventions have been chosen so that in a case where a quantity could be assigned arbitrarily, a zero value is assigned so that later calculations will be as simple as possible.

8.b)

### 3.6 Actuator space, joint space, and Cartesian space

The position of all the links of a manipulator of  $n$  degrees of freedom can be specified with a set of  $n$  joint variables. This set of variables is often referred to as the  $n \times 1$  **joint vector**. The space of all such joint vectors is referred to as **joint space**. Thus far in this chapter we have been concerned with computing the **Cartesian space** description from knowledge of the joint space description. We use the term *Cartesian space* when position is measured along orthogonal axes, and orientation is measured according to any of the conventions outlined in Chapter 2. Sometimes the terms **task-oriented space** or **operational space** are used for what we will call Cartesian space.

So far we have implicitly assumed that each kinematic joint is actuated directly with some sort of actuator. However, in the case of many industrial robots, this is not so. For example, sometimes two actuators work together in a differential pair to move a single joint, or sometimes a linear actuator is used to rotate a revolute joint through the use of a four-bar linkage. In these cases it is helpful to consider the notion of *actuator positions*. Since the sensors which measure the position of the manipulator are often located at the actuators, some computations must be performed to compute the joint vector as a function of a set of actuator values, or **actuator vector**.



3.16 Mappings between kinematic descriptions.

10.a

## Internal world model versus external reality

A central feature of a robot programming system is the world model that is maintained internally in the computer. Even when this model is quite simple, there are ample difficulties in assuring that it matches the physical reality that it attempts to model. Discrepancies between internal model and external reality result in poor or failed grasping of objects, collisions, and a host of more subtle problems.

This correspondence between internal model and the external world must be established for the program's initial state and must be maintained throughout its execution. During initial programming or debugging it is generally up to the user to suffer the burden of ensuring that the state represented in the program corresponds to the physical state of the workcell. Unlike more conventional programming, where only internal variables need to be saved and restored to reestablish a former situation, in robot programming, physical objects must usually be repositioned.

### Context sensitivity

**Bottom-up programming** is a standard approach to writing a large computer program in which one develops small, low level pieces of a program and then puts them together into larger pieces, eventually resulting in a completed program. For this method to work it is essential that the small pieces be relatively insensitive to the language statements that precede them and that there are no assumptions concerning the context with which these program pieces execute. For manipulator programming this is often not the case; code that worked reliably when tested in isolation frequently fails when placed in the context of the larger program. These problems generally arise from dependencies on manipulator configuration and speed of motions.

Manipulator programs may be highly sensitive to initial conditions, for example, the initial manipulator position. In motion trajectories, the starting position will influence the trajectory that will be used for the motion. The initial manipulator position may also influence the velocity with which the arm will be moving during some critical part of the motion. For example, these statements are true for manipulators that follow cubic spline joint space paths studied in Chapter 7. While these effects might be dealt with by proper programming care, such problems may not arise until after the initial language statements have been debugged in isolation and are then joined with statements preceding them.

## Error recovery

Another direct consequence of working with the physical world is that objects may not be exactly where they should be and hence motions that deal with them may fail. Part of manipulator programming involves attempting to take this into account and making assembly operations as robust as possible, but, even so, errors are likely; and an important aspect of manipulator programming is how to recover from these errors.

Almost any motion statement in the user's program can fail, sometimes for a variety of reasons. Some of the more common causes are objects shifting or dropping out of the hand, an object missing from where it should be, jamming during an insertion, not being able to locate a hole, and so on.

10.b

## User interface

Since a major motivation for developing an OLP system is to create an environment that makes programming manipulators easier, the user interface is of crucial importance. However, the other major motivation is to remove reliance on use of the physical equipment during programming. Upon initial consideration, these two goals seem to conflict—robots are hard enough to program when you can see them, how can it be easier

## 3-D modeling

A central element in OLP systems is the use of graphic depictions of the simulated robot and its workcell. This requires the robot and all fixtures, parts, and tools in the workcell to be modeled as three-dimensional objects. To speed up program development, it is desirable to use any CAD models of parts or tooling that are directly available from the CAD system on which the original design was done. As CAD systems become more and more prevalent in industry, it becomes more and more likely that this kind of geometric data will be readily available. Because of the strong desire for this kind of CAD integration from design to production, it makes sense for an OLP system to contain a CAD modeling subsystem, or to be itself a part of a CAD design system. If an OLP system is to be a stand-alone system, it must have appropriate interfaces to transfer models to and from external CAD systems. However, even a stand-alone OLP system should have at least a simple local CAD facility for quickly creating models of noncritical workcell items, or for adding robot-specific data to imported CAD models.

## Kinematic emulation

A central component in maintaining the validity of the simulated world is the faithful emulation of the geometrical aspects of each simulated

manipulator. Concerning inverse kinematics, the OLP system can interface to the robot controller in two distinct ways. First, the OLP system can replace the inverse kinematics of the robot controller, and always communicate robot positions in mechanism joint space. The second choice is to communicate Cartesian locations to the robot controller and let the controller use the inverse kinematics supplied by the manufacturer to solve for robot configurations. The second choice is almost always preferable especially as manufacturers begin to build *arm signature* style calibration into their robots. These calibration techniques customize the inverse kinematics for each individual robot. In this case, it becomes desirable to communicate information at the Cartesian level to robot controllers.

## Path planning emulation

In addition to kinematic emulation for static positioning of the manipulator, an OLP system should accurately emulate the path taken by the manipulator in moving through space. Again, the central problem is that the OLP system needs to simulate the algorithms in the robot controllers, and these path planning and execution algorithms vary considerably from one robot manufacturer to another. Simulation of the spatial shape of the path taken is important for detection of collisions between the robot and its environment. Simulation of the temporal aspects of the trajectory are important in predicting the cycle times of applications. When a robot is operating in a moving environment (for example, near another robot) accurate simulation of the temporal attributes of motion is necessary to accurately predict collisions, and in some cases to predict communication or synchronization problems such as deadlock.

## Dynamic emulation

Simulated motion of manipulators can neglect dynamic attributes if the OLP system does a good job of emulating the trajectory planning algorithm of the controller and if the actual robot follows desired trajectories with negligible errors. However, at high speed or under heavy loading conditions, trajectory tracking errors can become important. Simulation of these tracking errors necessitates modeling the dynamics of the manipulator and objects which it moves, as well as the control algorithm used in the manipulator controller. Presently practical problems exist in obtaining sufficient information from the robot vendors to make this kind of dynamic simulation of practical value, but in some cases dynamic simulation can be fruitfully pursued.



	<p><b>Multiprocess simulation</b></p> <p>Some industrial applications involve two or more robots cooperating in the same environment. Even single robot workcells often contain a conveyor belt, transfer line, vision system, or some other active device with which the robot must interact. For this reason, it is important that an OLP system be able to simulate multiple moving devices and other activities that involve <b>parallelism</b>. As a basis for this capability, the underlying language in which the system is implemented should be a multiprocessing language. Such an environment makes it possible to write independent robot control programs for each of two or more robots in a single cell, and then simulate the action of the cell with the programs running concurrently. Adding signal and wait primitives to the language enables the robots to interact with each other just as they might in the application being simulated.</p>
9.b	<p><b>World modeling</b></p> <p>Since manipulation programs must by definition involve moving objects in three-dimensional space, it is clear that any robot programming language needs a means of describing such actions. The most common element of robot programming languages is the existence of special <b>geometric types</b>. For example, <i>types</i> are introduced which are used to represent joint angle sets, as well as Cartesian positions, orientations, and frames. Predefined operators which can manipulate these types often are available. The “standard frames” introduced in Chapter 3 might serve as a possible model of the world: All motions are described as tool frame relative to station frame, with goal frames being constructed from arbitrary expressions involving geometric types.</p> <p>Given a robot programming environment which supports geometric types, the robot and other machines, parts, and fixtures can be modeled by defining named variables associated with each object of interest. Figure 12.3 shows part of our example workcell with frames attached in task-relevant locations. Each of these frames would be represented with a variable of type “frame” in the robot program.</p> <p>of the objects are not part of such a world model, and neither are surfaces, volumes, masses, or other properties. The extent to which objects in the world are modeled is one of the basic design decisions made when designing a robot programming system. Most present-day systems support only the style just described.</p>

## Motion specification

A very basic function of a robot programming language is to allow the description of desired motions of the robot. Through the use of motion statements in the language, the user interfaces to path planners and generators of the style described in Chapter 7. Motion statements allow the user to specify via points and the goal point, and whether to use joint-interpolated motion or Cartesian straight-line motion. Additionally, the user may have control over the speed or duration of a motion.

To illustrate various syntaxes for motion primitives, we will consider the following example manipulator motions: 1) move to position "goal1," then 2) move in a straight line to position "goal2," then 3) move without stopping through "vial" and come to rest at "goal3." Assuming all of these path points had already been taught or described textually, this program segment would be written as follows.

In VAL II:

```
move goal1
moves goal2
move vial
move goal3
```

## Flow of execution

As in more conventional computer programming languages, a robot programming system allows the user to specify the flow of execution. That is, concepts such as testing and branching, looping, calls to subroutines, and even interrupts are generally found in robot programming languages.

More so than in many computer applications, parallel processing is generally important in automated workcell applications. First of all, very often two or more robots are used in a single workcell and work simultaneously to reduce the cycle time of the process. But even in single-robot applications such as the one shown in Fig. 12.2, there is other workcell equipment which must be controlled by the robot controller in a parallel fashion. Hence signal and wait primitives are often found in robot programming languages, and occasionally more sophisticated parallel execution constructs are provided [3].

Another frequent occurrence is the need to monitor various processes with some kind of sensor. Then, either by interrupt or through polling, the robot system must be able to respond to certain events which are detected by the sensors. The ability easily to specify such **event monitors** is afforded by some robot programming languages [2], [3].

## Programming environment

As with any computer languages, a good programming environment helps to increase programmers' productivity. Manipulator programming is difficult and tends to be very interactive, with a lot of trial and error. If the user were forced to continually repeat the "edit-compile-run" cycle of compiled languages, productivity would be low. Therefore, most robot programming languages are now *interpreted* so that individual language statements can be run one at a time during program development and debugging. Typical programming support such as text editors, debuggers, and a file system are also required.

## Sensor integration

An extremely important part of robot programming has to do with interaction with sensors. The system should have the minimum capability to query touch and force sensors and use the response in if-then-else constructs. The ability to specify event monitors to watch for transitions on such sensors in a *background* mode is also very useful.

Integration with a vision system allows the vision system to send the manipulator system the coordinates of an object of interest. For example, in our sample application, a vision system locates the brackets on the conveyor belt and returns to the manipulator controller their position and orientation relative to the camera. Since the camera's frame is known relative to the station frame, a desired goal frame for the manipulator can be computed from this information.

Some sensors may be part of other equipment in the workcell. For example, some robot controllers can use input from a sensor attached to a conveyor belt so that the manipulator can track the belt's motion and acquire objects from the belt as it moves [2].

9.a)

## Explicit robot programming languages

With the arrival of inexpensive and powerful computers, the trend has been increasingly toward programming robots via programs written in computer programming languages. Usually these computer programming languages have special features which apply to the problems of programming manipulators and so are called **robot programming languages** (RPLs). Most of the systems which come equipped with a robot programming language have also retained a teach-pendant style interface as well.

Robot programming languages have taken on many forms as well. We will split them into three categories as follows:

### Task-level programming languages

The third level of robot programming methodology is embodied in **task-level programming languages**. These are languages which allow the user to command desired subgoals of the task directly, rather than to specify the details of every action the robot is to take. In such a system, the user is able to include instructions in the application program at a significantly higher level than in an explicit robot programming language. A task-level robot programming system must have the ability to perform many planning tasks automatically. For example, if an instruction to "grasp the bolt" is issued, the system must plan a path of the manipulator which avoids collision with any surrounding obstacles, automatically choose a good grasp location on the bolt, and grasp it. In contrast, in an explicit robot programming language, all these choices must be made by the programmer.

The border between explicit robot programming languages and task-level programming languages is quite distinct. Incremental advances are being made to explicit robot programming languages which help to ease programming, but these enhancements cannot be counted as components of a task-level programming system. True task-level programming of manipulators does not exist yet but is an active topic of research [9], [10].