University School of Automation and Robotics
GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
East Delhi Campus, Surajmal Vihar
Delhi - 110092

Introduction to Robotics
ARA-203

By: Dr. Divya Agarwal

- **UNIT- I** **10HOURS**
- **Fundamentals of Robot Technology:** Robot definition, automation and robotics, Robot anatomy, Brief History, Types of robots, Overview of robot subsystems, resolution, repeatability and accuracy, Degrees of freedom of robots, Robot configurations and concept of workspace, Mechanisms and transmission
- **End effectors:** Mechanical and other types of grippers, Tools as end effectors, Robot and effector interface, Gripper selection and design.
- **Sensors and actuators used in robotics**: Pneumatic, hydraulic and electrical actuators, applications of robots, specifications of different industrial robots

- **UNIT- II** **10HOURS**
- **Kinematics of Robots:** Transformation Matrices, Inverse transformation matrices, Forward and Inverse kinematic equation for position and orientation, Denavit-Hartenberg representation of robot, inverse kinematic solution for articulated robot, Numericals.
- **Differential Motions and velocities:** Jacobian, Differential motions of a frame, Differential motion between frames, Calculation of the Jacobian, Inverse Jacobian, Numericals.

- **UNIT-III** **10 HOURS**
- **Dynamic analysis of Force:** Lagrangian and Newtonian mechanics, Dynamic equations form multiple –DOF Robots, Static force analysis of Robots, Transformation of forces and moments between coordinate frames, Numericals.
- **Trajectory Planning:** Basics of Trajectory planning, Joint space trajectory planning, Cartesian Space trajectories, Numericals.

- **UNIT-IV** **10 HOURS**
- **Robot Programming languages & systems:** Introduction, the three levels of robot programming, requirements of a robot programming language, problems peculiar to robot programming languages.
- **Off-line programming systems:** Introduction, central issues in on-line and offline programming, Programming examples.
- **Application of robots:** Typical applications of robots in material transfer, machine loading/unloading; processing operations; assembly and inspection.

Text Books:

1. Saha, S. K. (2014). *Introduction to robotics*. Tata McGraw-Hill Education.

2. Mittal, R. K., & Nagrath, I. J. (2003). *Robotics and control*. Tata McGraw-Hill.

3. Fu, K. S., Gonzalez, R., & Lee, C. G. (1987). *Robotics: Control Sensing. Vis*. Tata McGraw-Hill Education.

4. Niku, S. B. (2001). *Introduction to robotics: analysis, systems, applications* (Vol. 7). New Jersey: Prentice hall.

Reference Books:

1. Spong, M. W., & Vidyasagar, M. (2008). *Robot dynamics and control*. John Wiley & Sons.

2. Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., & Burgard, W. (2005). *Principles of robot motion: theory, algorithms, and implementations*. MIT press.

3. Bhaumik, A. (2018). *From AI to robotics: mobile, social, and sentient robots*. CRC Press.

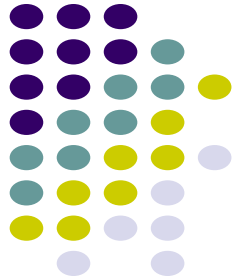
Programming Modes

- **Is the defining of desired motions so that the robot may perform them without human intervention.**
- **Identifying and specifying the robot configurations (i.e. the pose of the end-effector, P_e, with respect to the base-frame)**

Types of Robot Programming are:

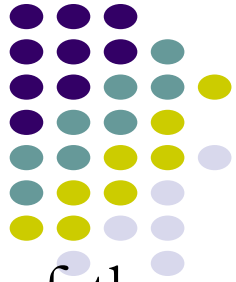
1.MANUAL METHOD
2.WALKTHROUGH METHOD
3.LEADTHROUGH METHOD
4.OFF-LINE PROGRAMMING

Robot Programming Methods



- Offline:
  - write a program using a text-based robot programming language
  - does not need access to the robot until its final testing and implementation
- On-line:
  - Use the robot to generate the program
    - Teaching/guiding the robot through a sequence of motions that can them be executed repeatedly
- Combination Programming:
  - Often programming is a combination of on-line and off-line
    - on-line to teach locations in space
    - off-line to define the task or “sequence of operations"

Type of Robot Programming

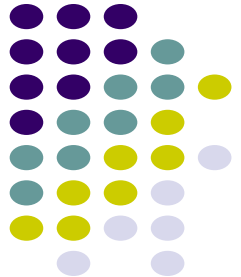


- **Joint level programming** - basic actions are positions (and possibly movements) of the individual joints of the robot arm: joint angles in the case of rotational joints and linear positions in the case of linear or prismatic joints.

- **Robot-level programming** - the basic actions are positions and orientations (and perhaps trajectories) of Pe and the frame of reference attached to it.

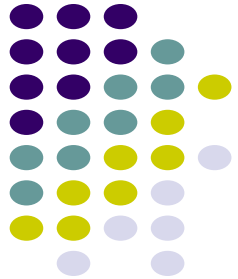
- **High-level programming**
  - Object-level programming
  - Task-level programming

Teach by showing



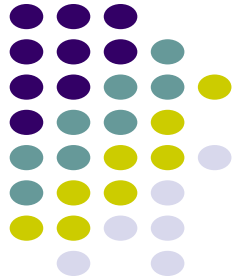
- Involved moving the robot to a desired goal point and recording its position in a memory that the sequencer would read during playback.
- During the teach phase, the user would guide the robot either by hand or through interaction with a teach pendant.
- Teach pendants are handheld button boxes that allow control of each manipulator joint or of each Cartesian degree of freedom.
- Some such controllers allow testing and branching, so that simple programs involving logic can be entered.
- Some teach pendants have alphanumeric displays and are approaching hand-held terminals in complexity.

Lead Through Programming



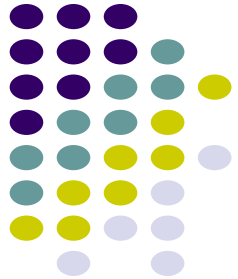
- lead the robot physically through the required sequence of motions
- trajectory and endpoints are recorded, using a sampling routine which records points at 60-80 times a second
- when played back results in a smooth continuous motion
- large memory requirements

Lead Through Programming



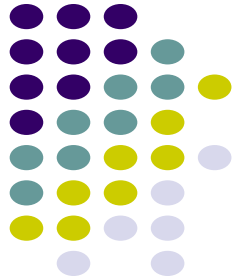
- Advantage:
  - Easy
  - No special programming skills or training
- Disadvantages:
  - not practical for large or heavy robots
  - High accuracy and straight-line movements are difficult to achieve, as are any other kind of geometrically defined trajectory, such as circular arcs, etc.
  - difficult to *edit out* unwanted operator moves
  - difficult to incorporate external sensor data
  - Synchronization with other machines or equipment in the work cell is difficult
  - A large amount of memory is required

On-Line Programming



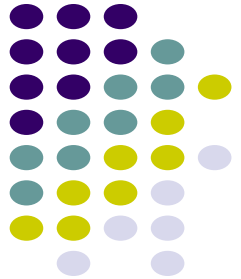
- Requires access to the robot
- Programs exist only in the memory of robot control system – often difficult to transfer, document, maintain, modify
- Easy to use, no special programming skills required
- Useful when programming robots for wide range of repetitive tasks for long production runs
- RAPID

On-Line/Teach Box



- Advantage:
  - Easy
  - No special programming skills or training
  - Can specify other conditions on robot movements (type of trajectory to use – line, arc)
- Disadvantages:
  - Potential dangerous (motors are on)

Off-line Programming



- Programs can be developed without needing to use the robot
- Sequence of operations and robot movements can be optimized or easily improved
- Previously developed and tested procedures and subroutines can be used
- External sensor data can be incorporated, though this typically makes the programs more complicated, and so more difficult to modify and maintain
- Existing CAD data can be incorporated-the dimensions of parts and the geometric relationships between them, for example.
- Programs can be tested and evaluated using simulation techniques, though this can never remove the need to do final testing of the program using the real robot
- Programs can more easily be maintained and modified
- Programs can more be easily properly documented and commented.

Object Level Programming

- basic actions are operations to be performed on the parts, or relationships that must be established between parts

pick-up part-A **by** side-A1 **and** side-A3

move part-A **to** location-2

pick-up part-B **by** side-B1 **and** side-B3

put part-B **on-top-off** part-A

with side-A5 **in-plane-with** side-B6 **and**

with side-A1 **in-plane-with** side-B1 **and**

with side-A2 **in-plane-with** side-B2

Task Level Programming

- basic actions specified by the program are complete tasks or subtasks

paint-the car-body *red*

assemble the gear-box

ROBOT PROGRAMMING

- Typically performed using one of the following
  - On line
    - teach pendant
    - lead through programming
  - Off line
    - robot programming languages
    - task level programming

Motion Commands

MOVE P1

HERE P1 - used during lead through of manipulator

MOVES P1

DMOVE(4, 125)

APPROACH P1, 40 MM

DEPART 40 MM

DEFINE PATH123 = PATH(P1, P2, P3)

MOVE PATH123

SPEED 75

Programming Languages

- Motivation
  - need to interface robot control system to external sensors, to provide “real time” changes based on sensory equipment
  - computing based on geometry of environment
  - ability to interface with CAD/CAM systems
  - meaningful task descriptions
  - off-line programming capability

- Large number of robot languages available
  - AML, VAL, AL, RAIL, RobotStudio, etc. (200+)
- Each robot manufacturer has their own robot programming language
- No standards exist
- Portability of programs virtually non-existent

Robot Programming Languages

- The VALTM Language
- The VAL language was developed for PUMA robot
- Monitor command are set of administrative instructions that direct the operation of the
- robot system. Some of the functions of Monitor commands are
- Preparing the system for the user to write programs for PUMA
- Defining points in space
- Commanding the PUMA to execute a program
- Listing program on the CRT
- Examples for monitor commands are: EDIT, EXECUTE, SPEED, HERE etc.

THE MCL LANGUAGE

- MCL stands for Machine Control Language developed by Douglas.
- Language is based on APT and NC language.
- Designed control complete manufacturing cell.
- MCL is enhancement of APT which possesses additional options and features needed to do off-line programming of robotic work cell.
- Additional vocabulary words were developed to provide supplementary capabilities intended to be covered by the MCL.
- These capability include Vision, Inspection and Control of signals
- MCL also permits the user to define MACROS like statement that would be convenient to use for specialized applications.
- MCL program is needed to compile to produce CLFILE.
- Some commands of MCL programming languages are DEVICE, SEND, RECEIV, WORKPT, ABORT, TASK, REGION, LOCATE etc.