# Robot Programming

# Introductory review

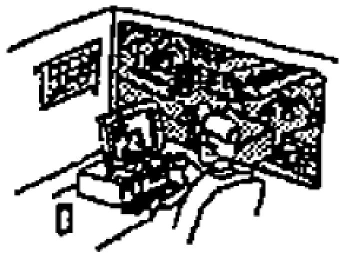
From the definition:

The industrial robot is an automatically controlled freely programmable, multifunctional manipulator with three or more programmable axes, for Industrial applications in either a fixed or mobile platform, can be used.

Freely programmable: programmable movements or auxiliary functions without physical amendment to be changed.

Robot programming

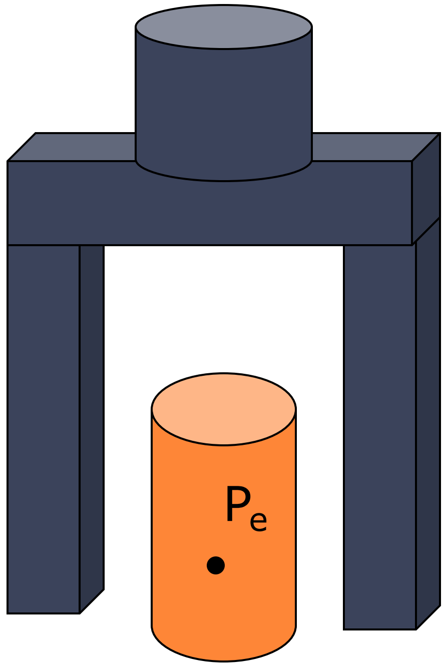
- On-Line
- Off-Line



## Robot Programming

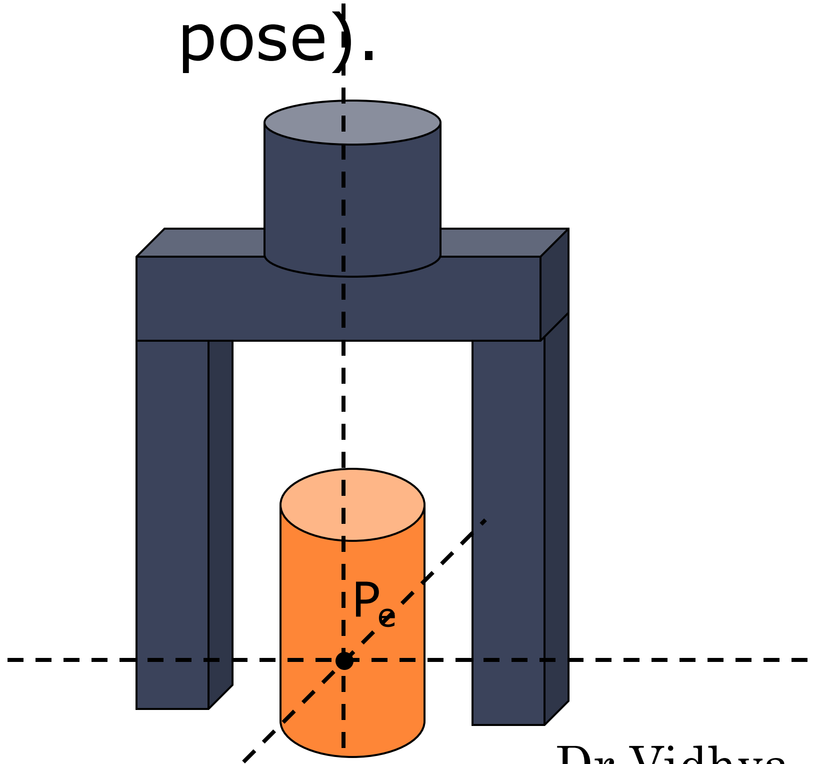
- **Programming** is the *identification* and *specification* of a series of *basic actions* which, when executed in the specified order, achieve some specific task or realize some specific process.
- **Robot Programming** is the defining of desired motions so that the robot may perform them without human intervention.

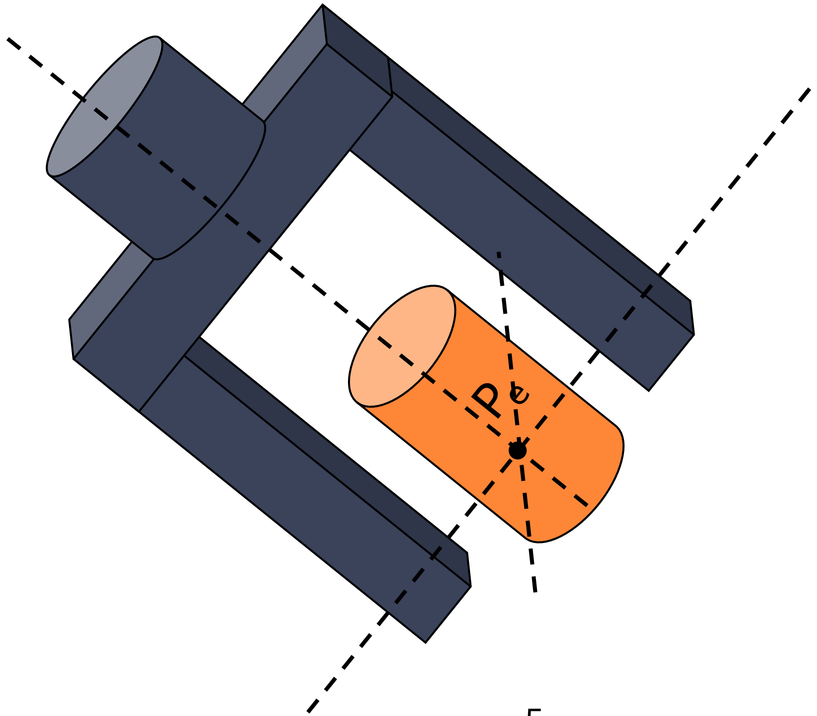
If, for example, the robot has a two finger gripper, to pick things up with, we usually define $P_e$ to be a point between the two fingers, so that when this point is geometrically inside some object to be picked up, all the robot has to do is to close the fingers of its gripper to grasp the object.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



**Pose**: both the position of $P_e$ in space, and its orientation

- 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 (together they define the object pose).





## Key Concepts in Robotics Programming

- Sensors and Perception: Robots use various sensors (such as cameras, lidar, and touch sensors) to perceive their surroundings and gather data.
- Actuators and Control: Actuators (such as motors and servos) enable robots to interact with their environment. Control systems determine how robots move and respond to inputs.
- Kinematics and Dynamics: Kinematics deals with the motion of robots without considering forces, while dynamics includes the study of forces and torques affecting robot movement.

## Key Concepts in Robotics Programming

- Motion Planning: This involves generating paths or trajectories for robots to follow while avoiding obstacles and considering constraints.
- Localization and Mapping: Robots need to know their own position (localization) and create maps of their environment (mapping) for effective navigation.
- Artificial Intelligence and Machine Learning: AI techniques like reinforcement learning and computer vision are used to make robots smarter and more adaptable.

## Robot Programming Methods

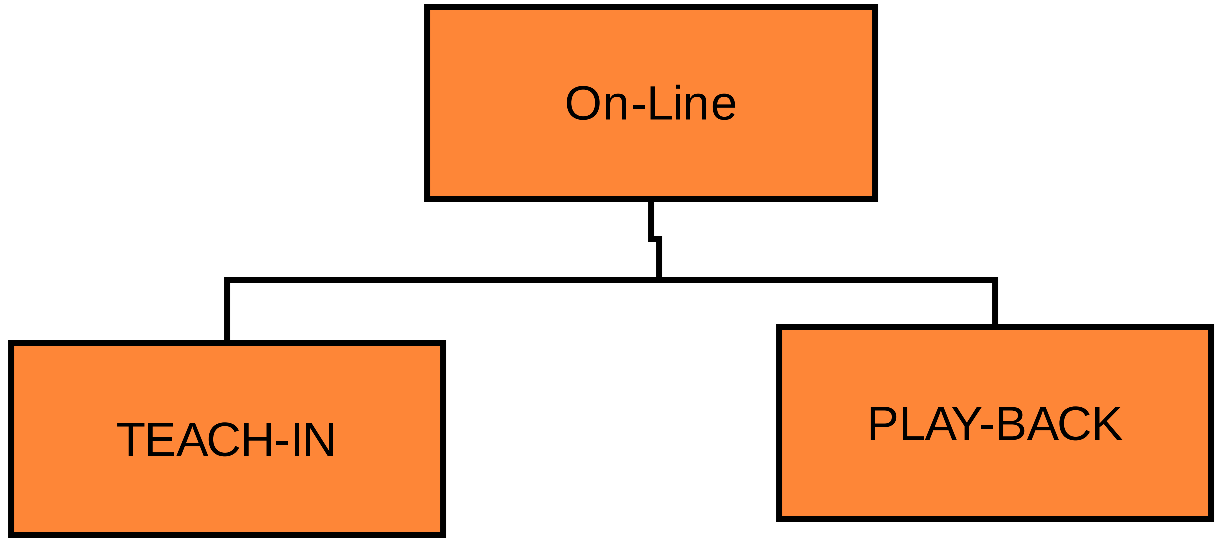
- Offline:
  - write a program using a text-based robot programming language
  - does not need access to the robot until its final testing and implementation
- On-line:
  - Use the robot to generate the program
    - Teaching/guiding the robot through a sequence of motions that can them be executed repeatedly
- 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"

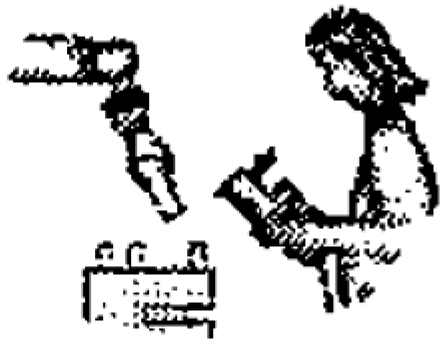
## ON-LINE/LEAD THROUGH

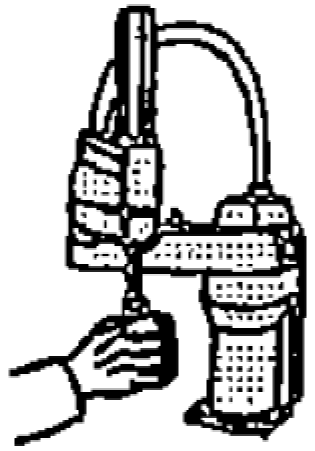
- Advantage:
  - Easy
  - No special programming skills or training
- Disadvantages:
  - not practical for large or heavy robots
  - High accuracy and straight-line movements are difficult to achieve, as are any other kind of geometrically defined trajectory, such as circular arcs, etc.
  - difficult 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

https://www.youtube.com/watch?app=desktop&v=hgkO1g8aAoo

## ON-LINE PROGRAMMING



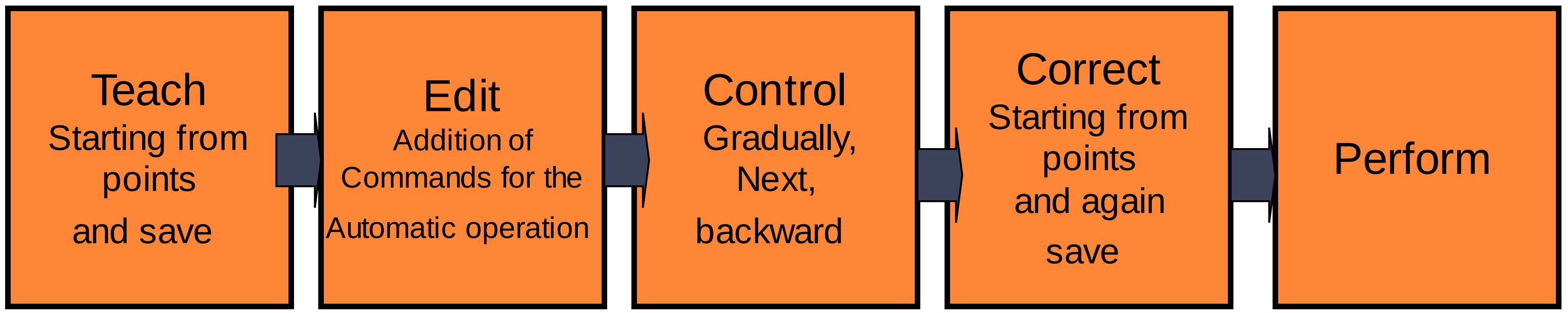




# TEACH-IN

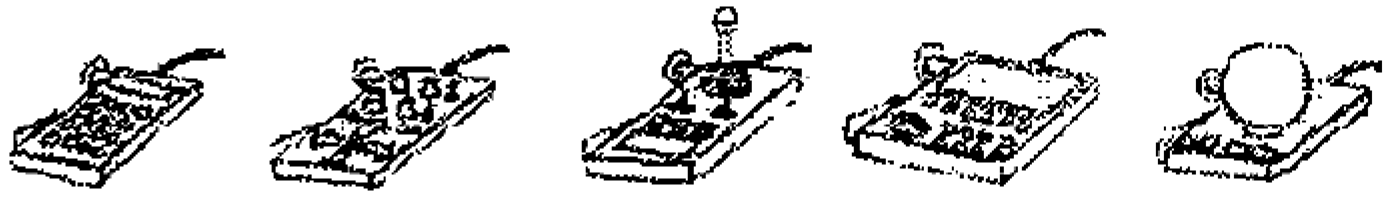
Task: transfer of appropriate programs in the robot controller

Teach-in Process:


Teach
Starting from points and save
Edit
Addition of Commands for the Automatic operation
Control
Gradually, Next, backward
Correct
Starting from points and again save
Perform

## TEACH PENDANT

PHG-types to a Teach-In program:

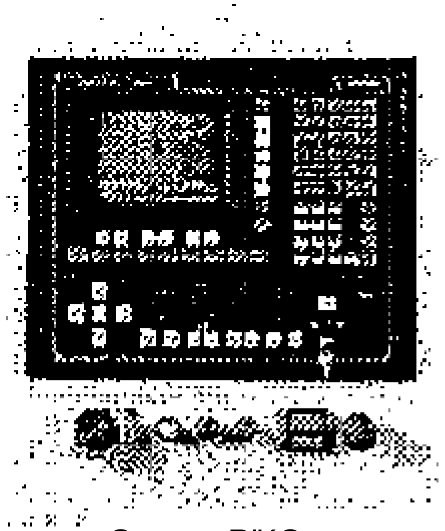


Keyboard Function keys Joy-Stick Display F-M Kugel

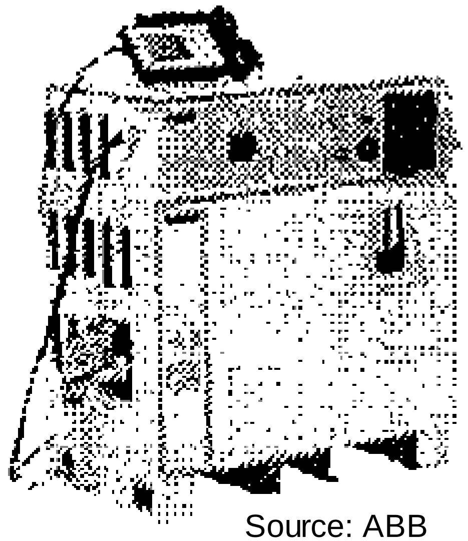
**Features:**
- To enable long cable to a good view
- Emergency stop switch
- Enabling switch
- Selection of TCP (Tool Center Point) function
- Selection of the coordinate
- Range of motion mode
- Selection of the speed of movement
- Status Display

## Control Equipment

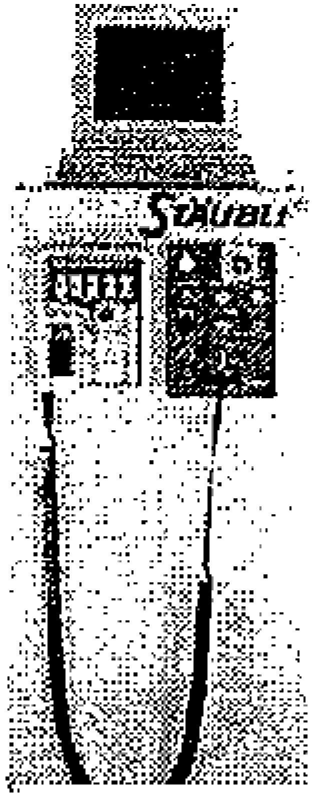
Types of robot-Program:


Source: RIKO

Editing on a CNC robot controlled by buttons and a built-in display



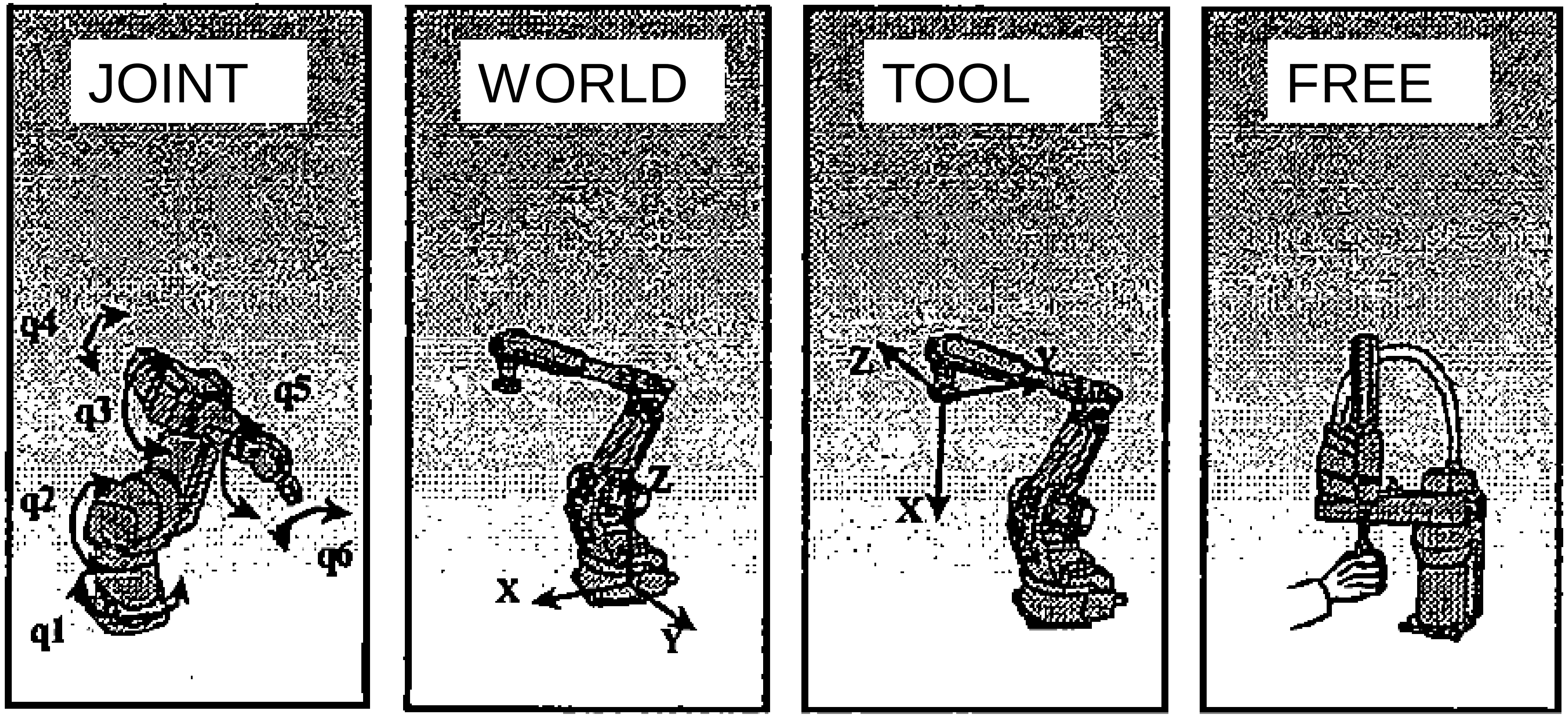
Editing by a keyboard and an LCD display on PHG



Source: Stäubli

Editing by a keyboard and LCD display

## Teach-in coordinate systems



## ON-LINE/TEACH BOX

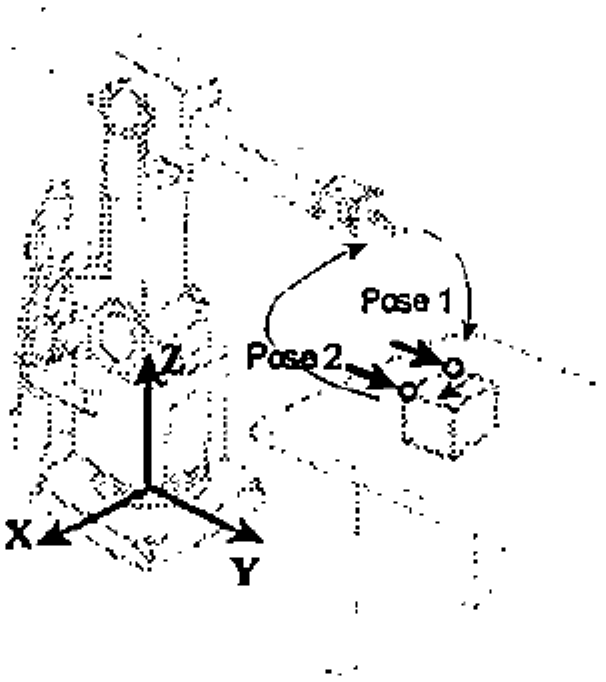
- Advantage:
  - Easy
  - No special programming skills or training
  - Can specify other conditions on robot movements (type of trajectory to use – line, arc)
- Disadvantages:
  - Potential dangerous (motors are on)

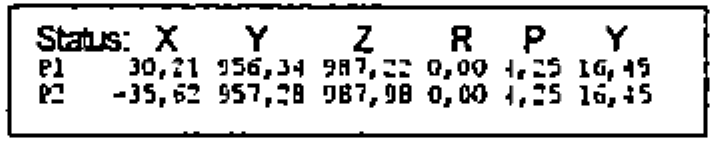
## On-Line Programming

- Requires access to the robot
- Programs exist only in the memory of robot control system – often difficult to transfer, document, maintain, modify

# Pose Programming

Abstract task:





| Status: | X | Y | Z | R | P | Y |
|---|---|---|---|---|---|---|
| P1 | 30,21 | 956,34 | 987,22 | 0,00 | 4,25 | 16,45 |
| P2 | -35,62 | 957,28 | 987,98 | 0,00 | 4,25 | 16,45 |

- Enter the "TCP" data
- Selection of the "Joint" coordinates
- Rotation of axes 1, 2 and 3 to pose a
- Choosing the "World or tool" coordinates
- Align the tool with "Roll, Pitch, Yaw Movement
- Switch to "Step or Slow Motion
Pose, exact position and orient
- Pose 1 save
    - Poses are always stored in set-point (position and orientation)
- Motion in Cartesian coordinates to pose 2
- Pose, position them and orient
- Pose 2 save
- Movement in the "wait-and-home" position

# PROGRAMMING FUNCTIONS AND SIGNAL

Example of a program on a very simple but defined robot!

Quelle: RJKO und US

| Robot program | Explanation |
|---|---|
| ▪ MAXSP = 1000 60 | -Determination of the total velocity in mm / s and degree / s |
| ▪ ACC = 2 | - Determination of the acceleration factor |
| ▪ TCP1 = 0 155 0 0 0 0 | - Determination of the TCPs in mm (X, Y, Z, R, P, Y) |
| ▪ SPEED = 100 | - Determination of the rate in % overall speed |
| ▪ APPRO TO 1 FOR 0 -20 0 | - Bringing the Pose 1 for X, Y, Z |
| ▪ SPEED = 20 | - |
| ▪ SMOVE TO 1 | Motion in Cartesian coordinates to Pose 1 |
| ▪ SPEED = 7 | - |
| ▪ WEAVE 2 5 8 0 | Determination of the pendulum motion during welding |
| ▪ WELD START | - Welding start (signal to welding device) |
| ▪ LINE TO 2 CON | - A straight weld line to pull Pose 2 |
| ▪ WELD STOP | - Welding End |
| ▪ SPEED = 20 | - |
| ▪ DEPART FOR 0 30 0 | Move away from Pose 2 for X, Y, Z |
| ▪ SPEED = 100 | - Movement in joint coordinates to the home position |
| ▪ HOME | |

## Programming Languages

Almost every robot has its own programming language!

| | |
|---|---|
| ARLA | ABB |
| AML | IBM |
| BAPS | Bosch |
| DOROB | AEG |
| HELP | DEA |
| KAREL | GMF |
| ROBOTsar | Reis |
| ROLF | Cloos |
| SIGLA | Oliveti |
| SRCL | Siemens |
| VAL II | Unimation |
| V+ | Stäubli |
| V+ | ADEPT |

Control independent language:

PasRo
SRL
C++

https://www.youtube.com/watch?v=XxBWydEtgWk&list=PLdG9DWheQeVIuLbnW-xQPJGr8pew8jWOQ&index=2

https://www.youtube.com/watch?v=ohePUt-NylU

# Play-Back Programming

- The operator uses the robot (or a device driver) directly.
- He leads the mechanism manually the task accordingly.
- During the movement the respective joint angle values can be read in a fixed time frame and stored.

Source: Gorenje

**Requirements:**

**- Kinematic balance**

**- Low friction in the joints**

**- Very high flexibility of movement**

**Advantages:**

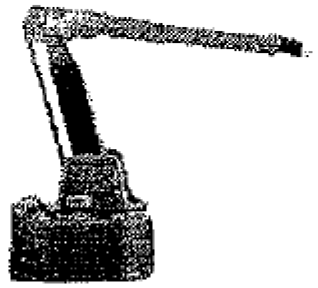
**- Very easy programming**

**- No prior knowledge of the operator necessary**

**Disadvantages:**

**- Correction in difficult sections of path**

**- Difficult change of velocity**


Source: ABB

Primarily for the coating or glue-robot

## TASK OF THE SENSORS

Approach to the programming ease, flexibility and versatility

**Important role** ➡ **SENSORS**

- Searching for objects or poses

  Parts on an assembly line, seam beginning, seam edge, plate boundary, installation help ...
- Tracking contours or litigation

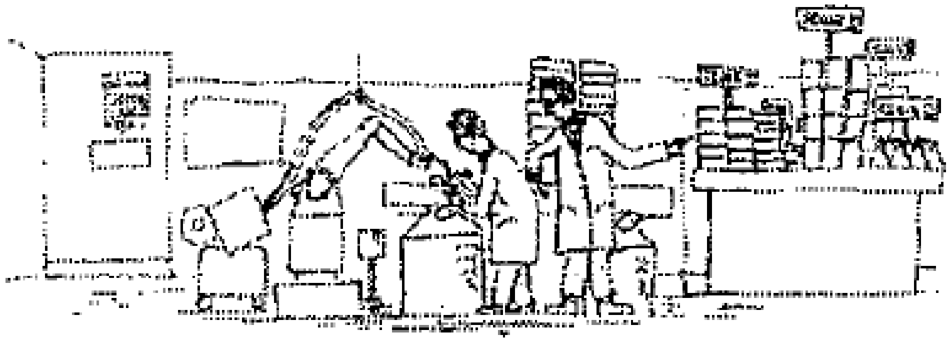
  Seam tracking, editing, with constant force (brushes, polishing, grinding, ...)

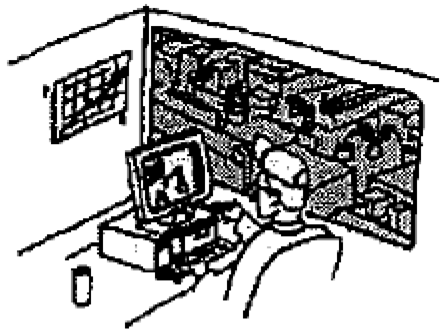
  Bending process pursue prosecution of Machine removal of parts ...
- Velocity fitting

  Teaching at a standstill, performing the move (coating on the assembly line, ...)

## Off-Line Programming

What is Off-Line Programming?



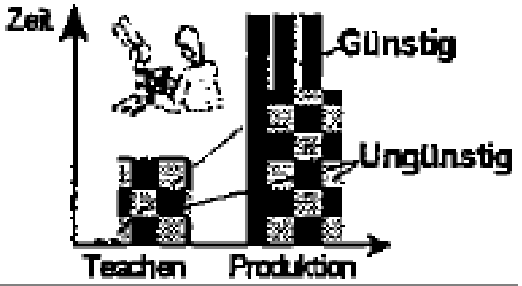


When?

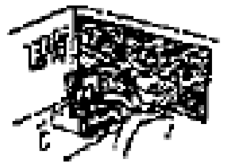
Why Off-Line Programming?

## WHAT IS OFF-LINE PROGRAMMING?

Teach-in methods:
Satisfactory only when the time for teaching in comparison to the production time is low.
The robot must be used instead for the programming for the production.



Off-Line Programming:
The robot can work, while a new program is created.
Robot programs are independent of the singly or in total produced in the production of industrial robots working.



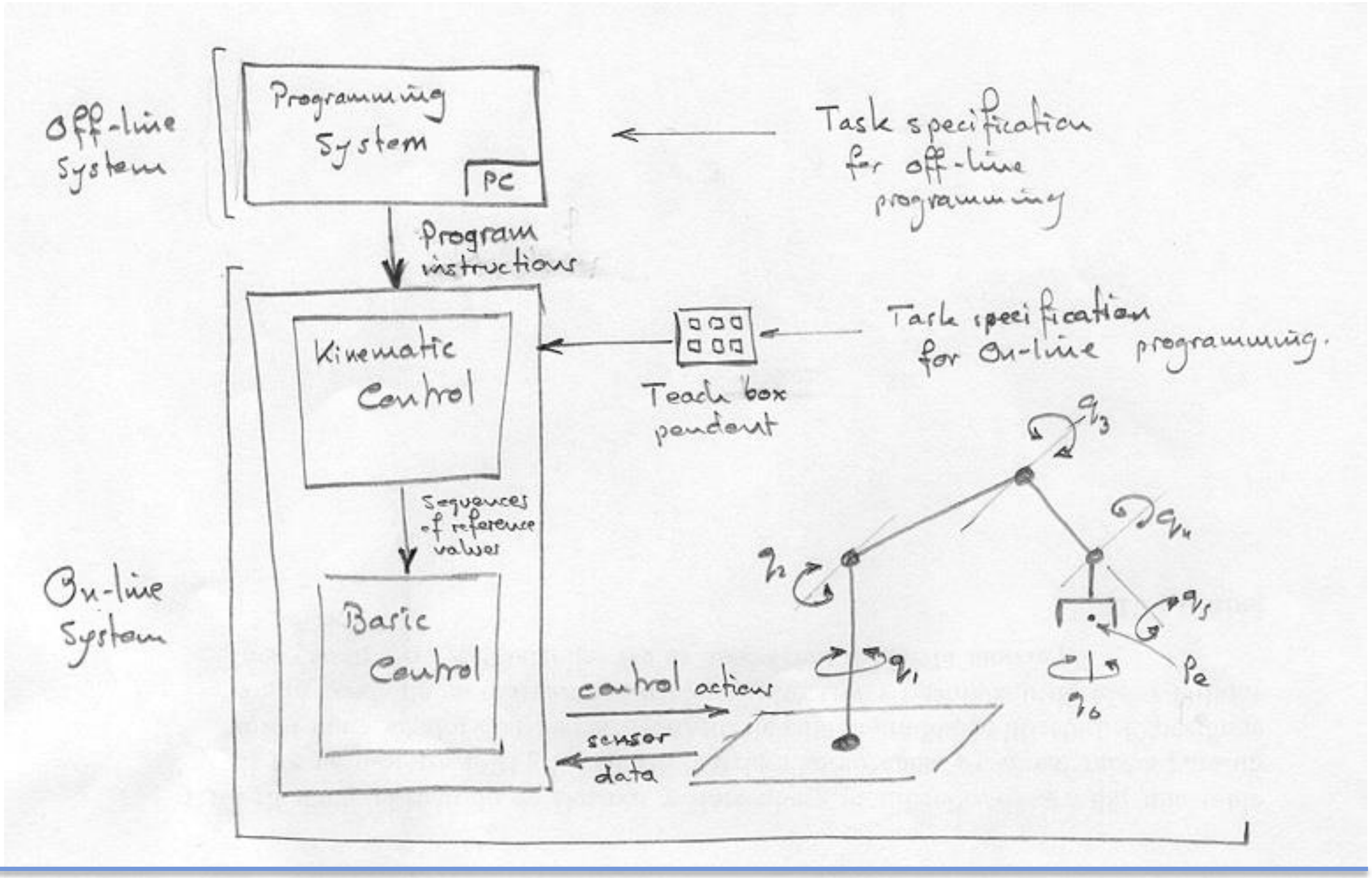
Modern robot:

- -Advanced Robot Controllers
- -Increase of the absolute pose accuracy
- -Adaptation of sensor technologies

These techniques are increasingly used in the industry.

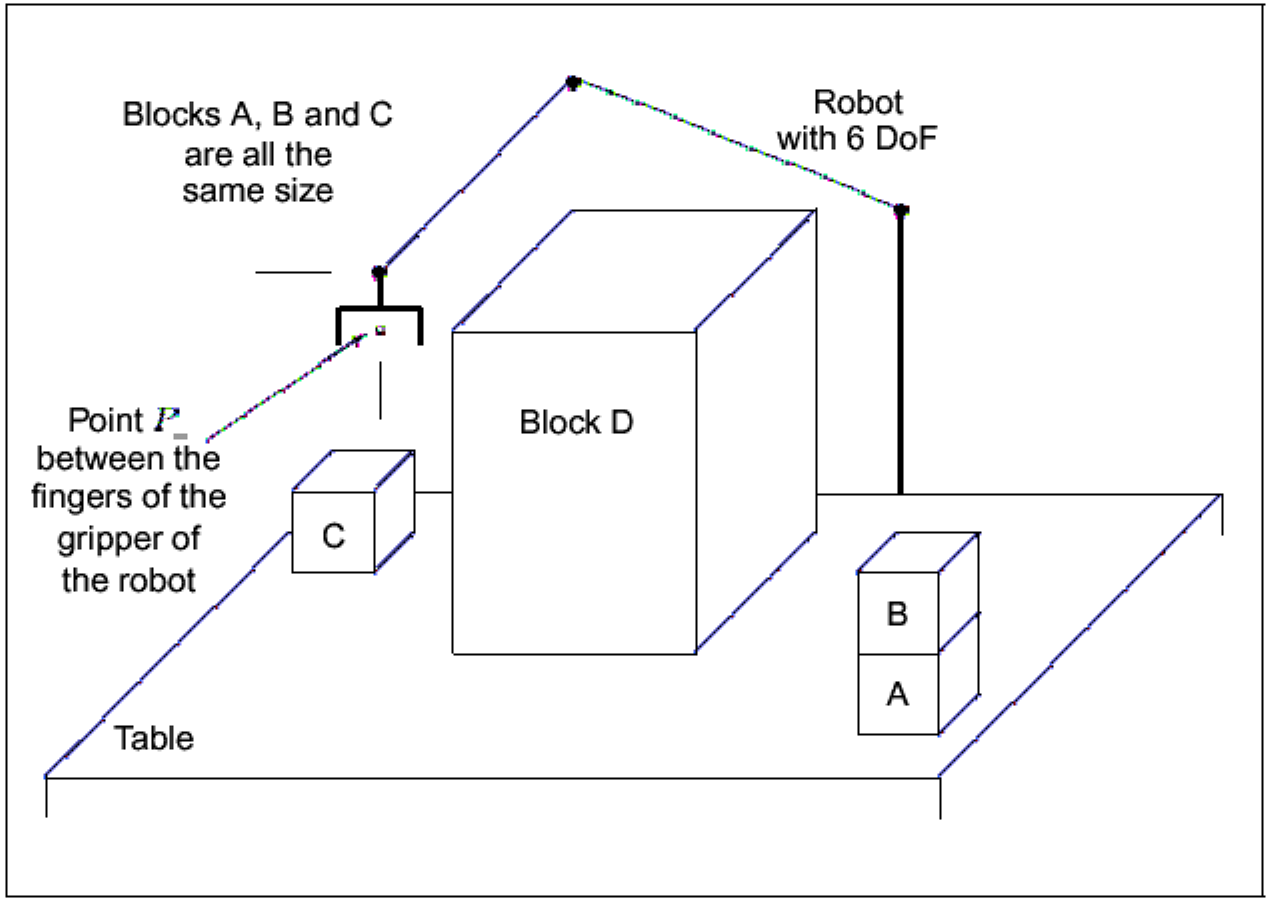
## Off-line Programming

- Programs can be developed without needing to use the robot
- The 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.



Off-line system
Programming System
PC
Task specification for off-line programming
Program instructions
Kinematic Control
Teach box pendant
Task specification for On-line programming.
Sequences of reference values
On-line System
Basic Control
control actions
sensor data
$q_1$
$q_2$
$q_3$
$q_4$
$q_5$
$q_6$
$p_e$

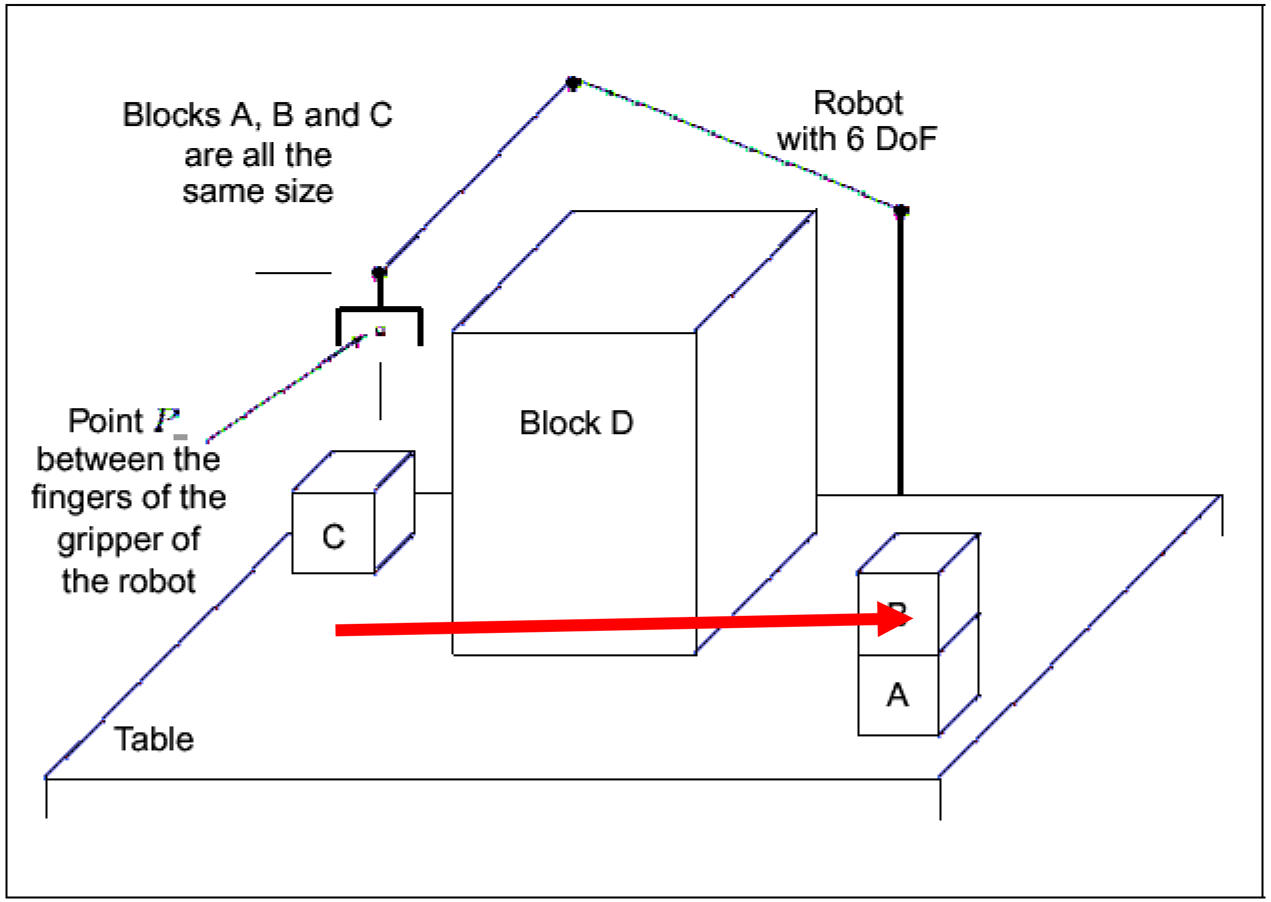
## Robot Program Development



## Robot Program Development Process

- Analyze and decompose the task into a series of operations on the objects involved, and specify their order.
- Identify and specify all the situations needed to program all the movements and actions of the robot.
- Identify any types of repeated actions or operations and specify them as subroutines with parameters.
- Design and develop the complete robot program and its documentation.
- Test and debug the program using a simulator of the robot and its work space.
- Test the program on the real robot.

## VIA POINTS!


Blocks A, B and C
are all the
same size
Robot
with 6 DoF
Point P
between the
fingers of the
gripper of
the robot
Block D
C
B
A
Table

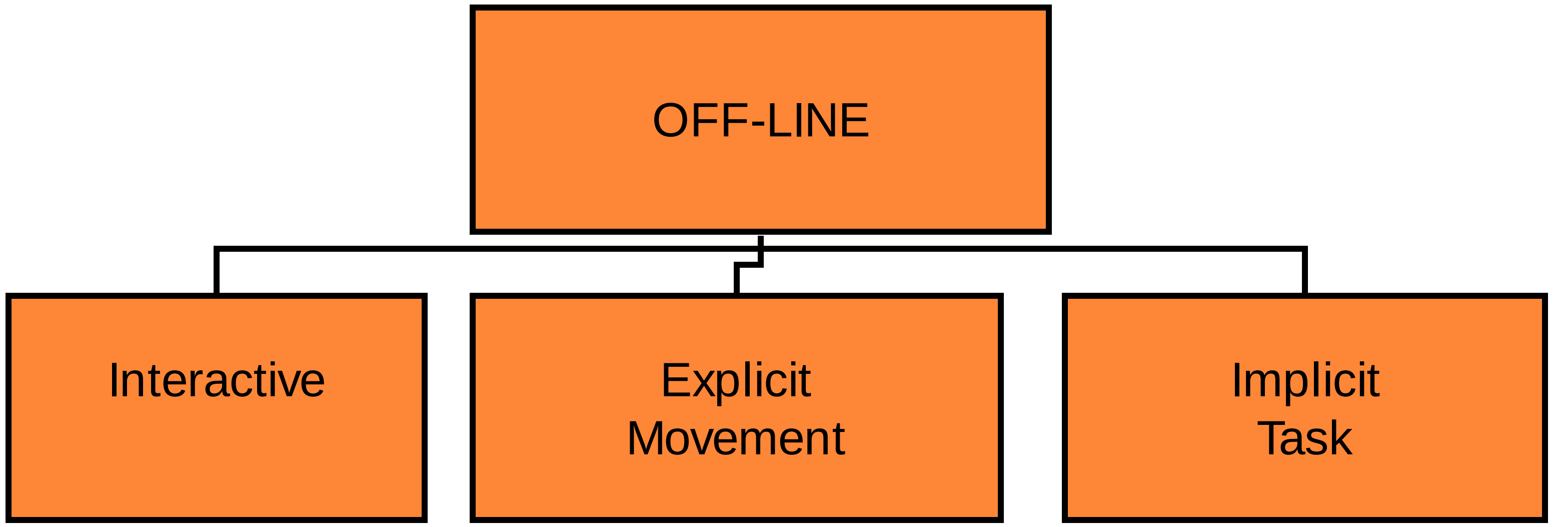
## Why Off-Line Programming?

- Teach-In - Time-consuming activity
  - This increases with the complexity of the task
- Teaching reduces the cost
  - Robots can not produce during the Teaching
- Robot in mass production
  - Spot welding in automotive industry
  - Does teach in new constitute a minimal effort
- Robots in small and medium production
  - Programming time can be considerably-
  - Useful introduction of off-line programming
- Increasing complexity of robot tasks
  - Off-line programming can be very attractive

## Benefits of Off-Line Programming

1. Reduction of the robot operating time
2. Dislocation of both ends of the danger area of the robot
3. Single robot programming system
4. Integration of CAD / CAM systems
5. Simplification of complex tasks
6. Optimization of robot programs
7. Access control
8. Cycle time analysis
9. Verification of robot programs

## OFF-LINE PROGRAMMING



## Now we can say:

- Off-line programming is a significant increase in the productivity of the robot in the industry.
- The main arguments for Off-Line system is the reduction of the overall robot use time by the overlap of the robot-programming with the actual robot work.
- There are universal off-line programming systems, which allow the generation of robot program codes accessible for almost any industrial robot. In order to reduce their dependence start from a single robot.

## 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 $P_e$ and the frame of reference attached to it.
- High-level programming
  - Object-level programming
  - Task-level programming

## 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

*Literature:*

HTTP://WWW.SOCIETYOFROBOTS.COM/ROBOTTHEORY/SURVEY_OF_ROBOT_PROGRAMMING_SYSTEMS.PDF

https://www.youtube.com/watch?v=ZD6wf6IJv6w

https://www.youtube.com/watch?v=8lfZkDIQ6NY

https://www.halvorsen.blog/documents/programming/python/python.php

https://homes.cs.washington.edu/~ztatlock/599z-17sp/papers/robot-programming-lozano-perez-83.pdf

# ROBOT PROGRAMMING LANGUAGES

- There are ***several programming languages*** used for ***programming robots***, each with its ***own strengths and weaknesses***. Here are some of the ***most common programming languages*** used for robotics:
- ***C/C++:*** C and C++ are ***low-level programming languages*** that ***provide a high degree of control*** over the hardware and are commonly used for ***programming embedded systems***, including robots. They are ***fast and efficient***, making them well-suited for ***real-time control*** and ***processing tasks***.

- *PYTHON:* Python is a ***high-level programming language*** that is ***easy to learn*** and has a ***large ecosystem of libraries and tools*** for ***data analysis and machine learning***. It is commonly used for ***writing scripts*** and ***algorithms*** for robotics applications.
- *MATLAB:* MATLAB is a ***high-level programming language*** that is commonly used for ***scientific computing and data analysis***. It is well-suited for ***simulating and analyzing robotic systems***, and has ***built-in libraries*** for ***control and signal processing***.
- *ROBOT OPERATING SYSTEM (ROS):* ROS is a ***framework for developing robot software*** that provides a ***set of libraries and tools*** for ***building and testing robotic systems***. It is based on the C++ programming language, but also supports other languages such as Python.

- ***BLOCKLY:*** Blockly is a ***visual programming language*** that allows ***users to create programs*** by ***dragging and dropping blocks*** that represent ***different commands and functions***. It is often used for ***teaching programming to beginners*** and for ***quickly prototyping robotic systems***.
- ***LABVIEW***: LabVIEW is a that is commonly used for ***controlling and monitoring complex systems***, including robotics. It provides a ***graphical user interface*** for ***designing and debugging programs***, and supports a ***variety of hardware interfaces***.

## INTRODUCTION TO ROS

- Robot Operating System (ROS) is a ***flexible*** and ***powerful open-source framework*** for building robotic applications.
- It provides a ***set of software libraries and tools*** that enable developers to create complex and distributed robotic systems.
- ROS was first developed by ***Willow Garage***, a robotics research lab, and is now maintained by the ***Open Robotics organization***.
- ROS is designed to be ***modular and scalable***, which means that it can be used for a wide range of applications, from small mobile robots to large industrial robots.
- It is also designed to be ***language-agnostic***, which means that it can be used with a variety of programming languages, including C++, Python, and Java.

- One of the key features of ROS is its ***message-passing system***, which allows different components of a robotic system to communicate with each other seamlessly. This makes it easy to build complex robotic systems by combining different ROS packages and nodes.
- ROS also includes a wide range of ***built-in tools and libraries*** that make it easy to perform tasks such as 3D perception, mapping, navigation, and manipulation. These tools can be used to create robotic applications for a wide range of fields, including industrial automation, logistics, healthcare, and more.

Overall, ROS is a ***powerful and flexible framework*** that makes it easy to build complex robotic applications. Its open-source nature, large community of developers, and extensive documentation make it an ideal choice for anyone looking to get started with robotics.

Here are some of the ***fundamentals of Robot Operating System*** (ROS):

1. ***NODES***: Nodes are ***individual components*** of a ROS system ***that can communicate with each other***. Nodes can be written in different programming languages, and they can be run on different computers in a distributed system.

2. ***TOPICS:*** Topics are ***channels of communication between nodes***. They allow nodes to ***send and receive messages*** to each other. Topics are named, and messages are transmitted using a publisher-subscriber model.
3. ***MESSAGES***: Messages are the ***data transmitted over topics***. They are defined in message files, which specify the type of data being transmitted.
4. ***SERVICES:*** Services ***allow nodes to request and receive information from each other.*** Services use a request-response model, where a node sends a request to another node, and the other node sends a response back.

5. ***ACTIONS***: Actions allow ***nodes to execute long-running goals that require feedback***. Actions use a three-part message exchange between a client node, a server node, and a feedback node.
6. ***PACKAGES:*** Packages are the ***basic organizational unit in ROS***. They contain nodes, messages, services, and other files related to a specific functionality.
7. ***LAUNCH FILES***: Launch files are used to ***start multiple nodes and configure their communication with each other.*** Launch files can be used to start a complex system with a single command.
8. ***PARAMETER SERVER***: The parameter server is a ***shared storage system*** for configuration parameters that can be accessed by nodes. Parameters can be set and retrieved dynamically during runtime.

9. ***RVIZ:*** RViz is a ***3D visualization tool*** in ROS that can be used to visualize robot models, sensor data, and other information.

These are just some of the fundamental concepts in ROS. Understanding these concepts is essential for building complex robotic systems using ROS.