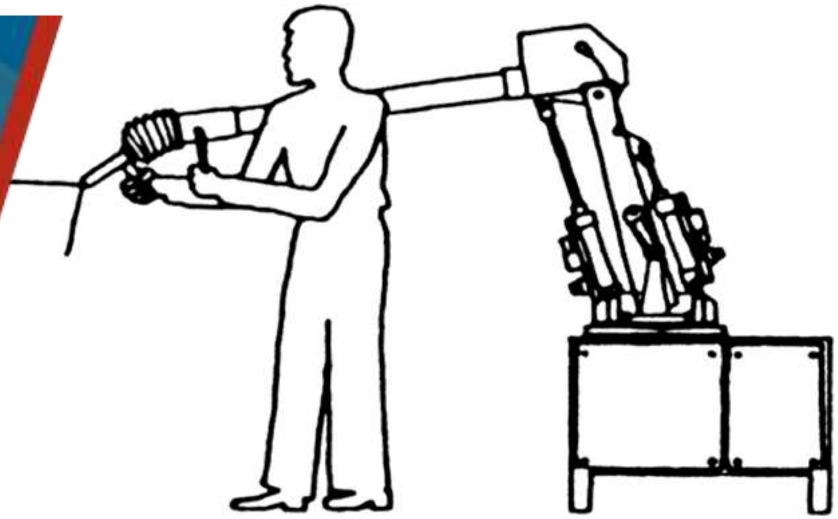
Robot Programming



Prof. Dr. Karsten Berns
Robotics Research Lab
Department of Computer Science
TU Kaiserslautern, Germany

Contents

- Programming of industrial robots
- Online/Offline-methods
- Types of programming
- Environmental modelling

- Foliensatz z.T. von
  - Dr. R. Lafrenz, Universität Stuttgart
  - Prof. Zühlke PAK, TU KL

Programming of Industrial Robots

- Must be freely programmable
- Sequence of points to approach
- Point sequence arbitrarily often approachable
- Free choice of points restricted by ...
  - Obstacles
  - Constructive limitations of the robot

Components of Programming

- Operating system
  - Real-time capability
  - Interface for robot control
- Programming language
  - Robot-specific language (VAL, …)
- Libraries for standard languages (RCCL for C, …)

Components of Programming

- Robot-oriented routines
  - Special data types (matrices)
  - Kinematic and dynamic routines
  - Movement orders (Cartesian, joint space)
  - Effector commands
- Task-oriented routines
  - Knowledge base with environmental model
  - Rule base for the task decomposition
  - Planning algorithms
  - Issuing complex tasks

Overview of Programming Techniques

Programming Method for Industrial Robots

Direct metods,
online programming

Indirect Methods,
offline programming

Teach-In

Playback

Sensor-supported

Hybrid Methods

Textual

CAD-supported
(Graphic)

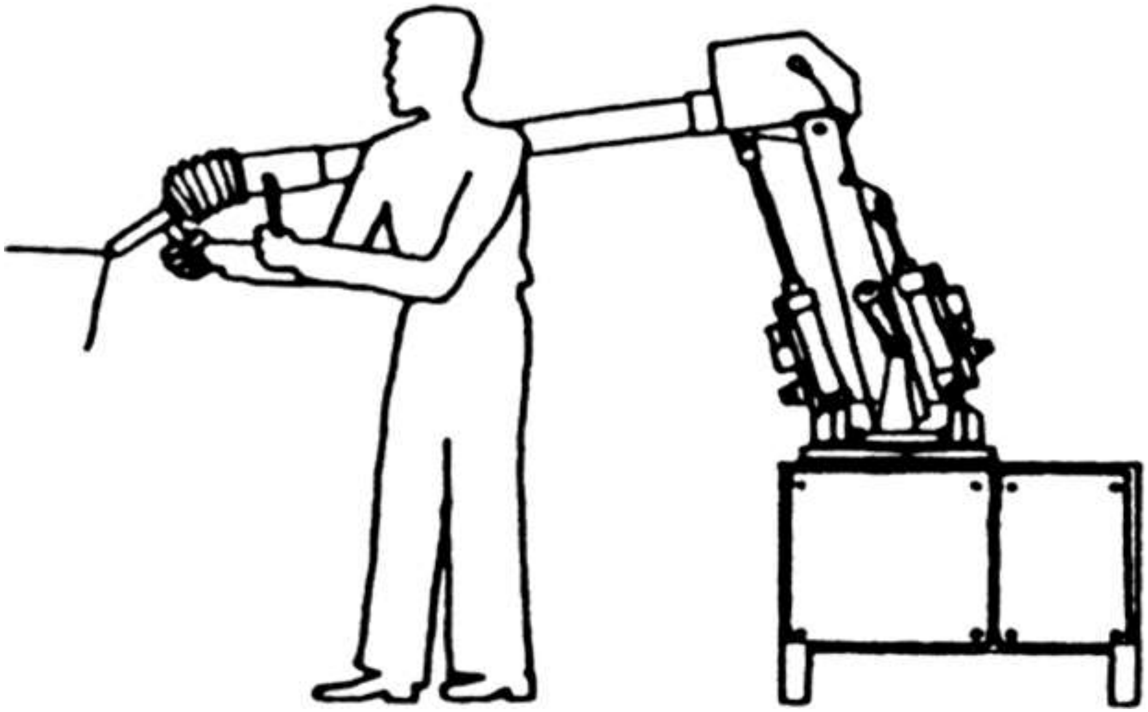
Visual

Online Methods: Teach-In

- Positioning and configuration of the robot with special control devices
- Control devices
  - Teachbox
  - Joystick
  - Mouse
  - Teach-ball
  - Applications
  - Point-to-point control
  - Multipoint control (MP)

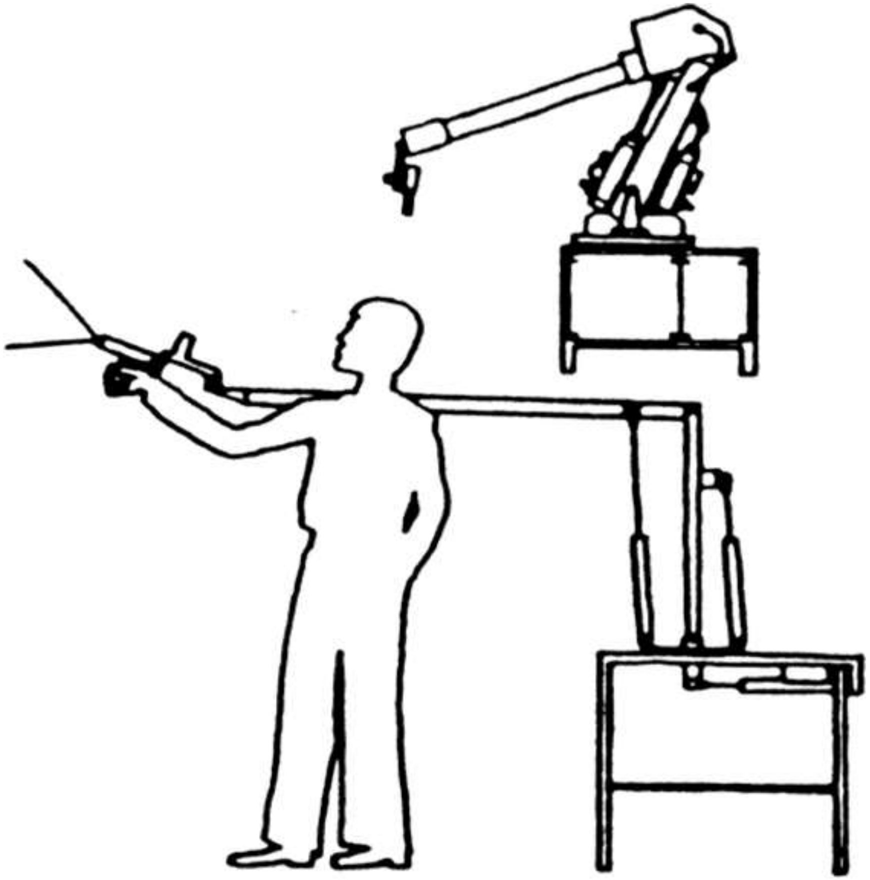
Online Methods: Playback, Manual Guidance

- Manual guidance of the gravity-free robot
- Guiding difficult even in zero gravity mode
- Today only with anthropomorphic robot arms



Online Methods: Playback, Master-Slave

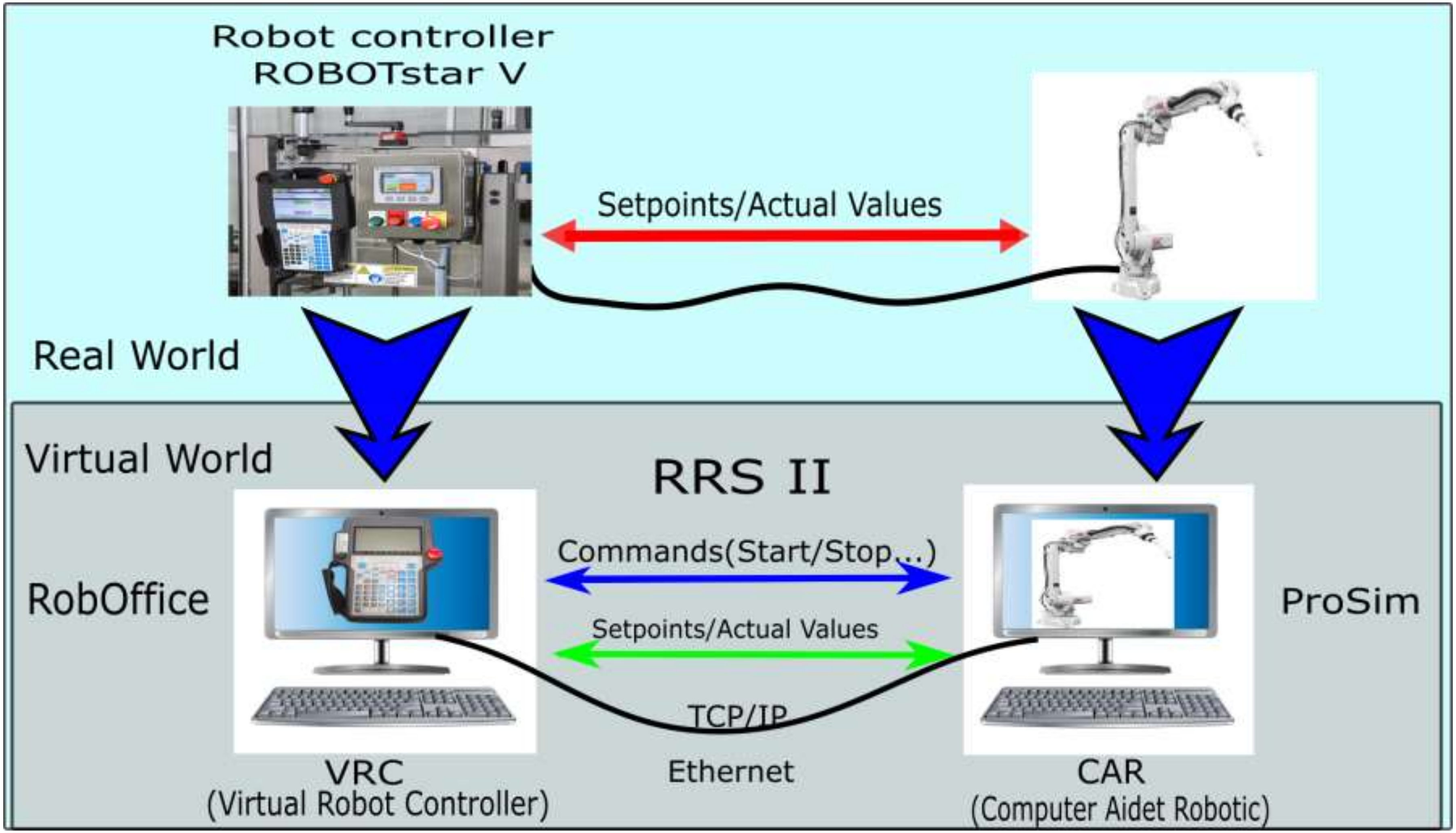
- Master-slave systems with as much identical kinematics as possible
- Manual guiding of master used only on teleoperation
- Feedback of forces (virtual reality)
- Transmission delays
- Expensive because of master system



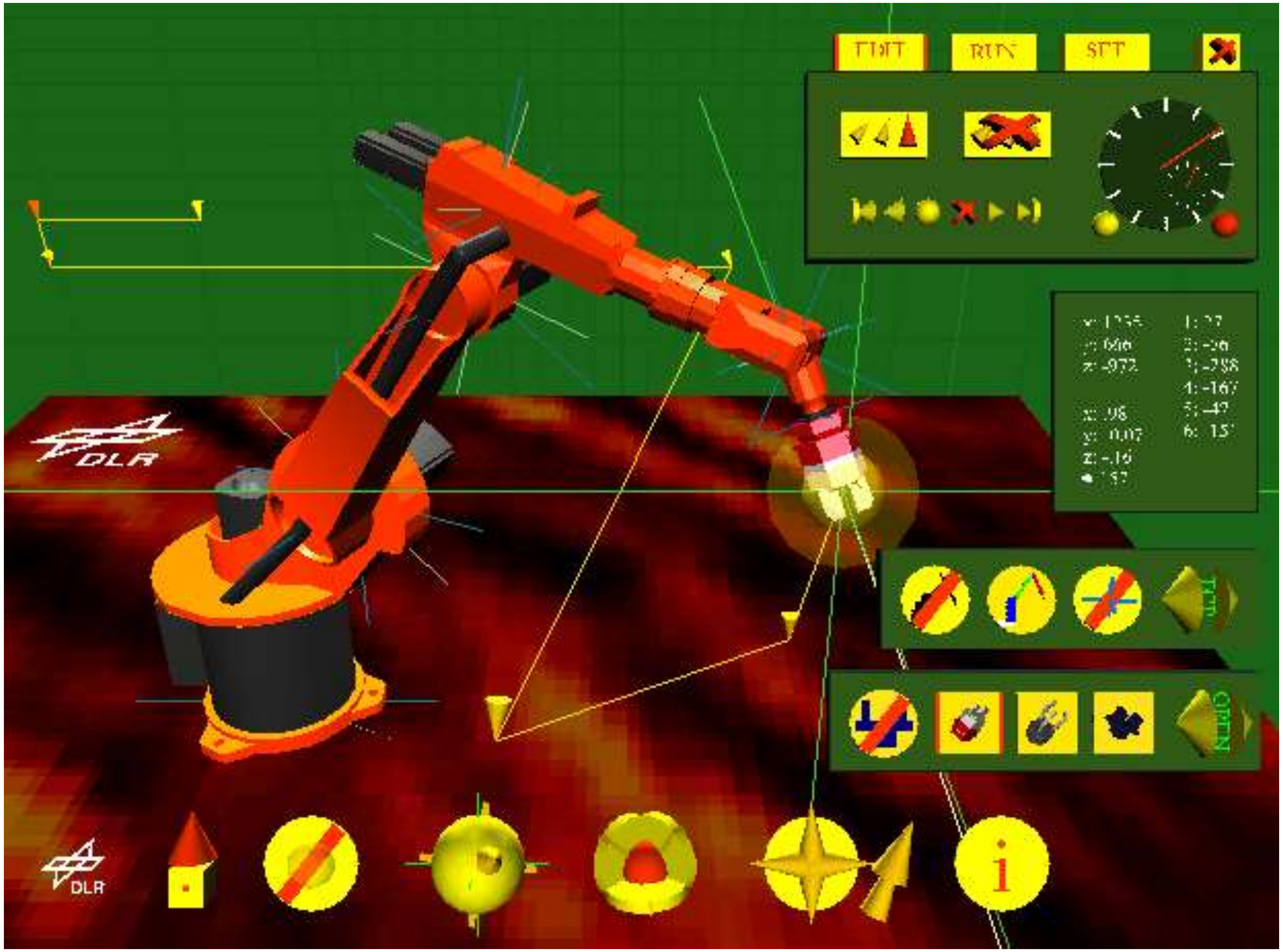
Offline Method: CAD Robot Simulation

- Software simulation (kinematics and dynamics) of robot cell and robot
- Creation of the simulation time-consuming/expensive
- Comparatively cheap optimization of motion sequences
- Risk of incorrect or incomplete simulation
  - Kinematic and dynamic parameters must be as exact as possible, otherwise damage to the robot is possible

Robot Simulation: Realistic Robot Simulation



Robot Simulation: VRML 2.0 DLR und KUKA



Comparison Between Simulation And Robot

Calibration effort

Comparison between world coordinate system

- \+ Less programming effort
- \+ Absolute coordinate input possible
- \- Calibration is expensive
- \- No compensation for inaccuracy of robot system

Coordinate input relative to reference body in work space

- \+ Position of reference body is known
- \+ Easy calibration
- \- Calibration is expensive
- \- Positioning error only less, not zero

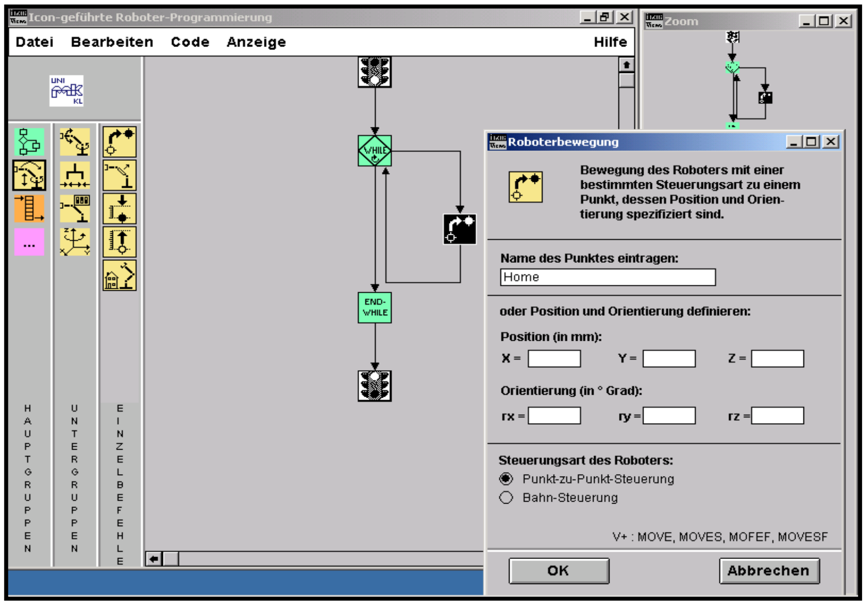
Individual detection of position via image processing

- \+ Position needs to be known only roughly. With image processing the position errors are gone
- \- High programming effort on robot and simulation side

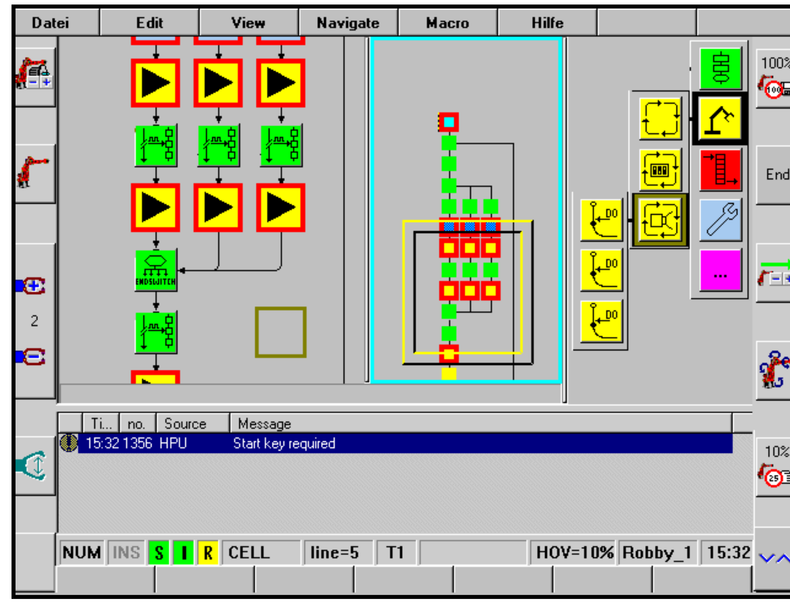
Programming effort

Visual Robot Programming

- Programing with two or higher-dimensioned structures
- Elements: Graphics, diagrams, icons, animations



Graphical Robot programming, prototype of PAK



Implementation of the PAK prototype for use with the KUKA handheld terminal

Second Classification of Programming

- Programming by examples
  - Settings of the robot
  - Manual programming
  - Teach-in-programming
  - Master-slave programming
- Programming through training
- (Textual) programming
  - Robot oriented
  - Task oriented

Programming by Examples: Settings

- Discrete poses, no continuous control
- Joint angles with mechanical switches and stops
- Task of robot control: Send signals to actuators, so that and the end point in time the stop signal is set to active
- Only a small set of points can be used for the program
- Free programming is greatly restricted

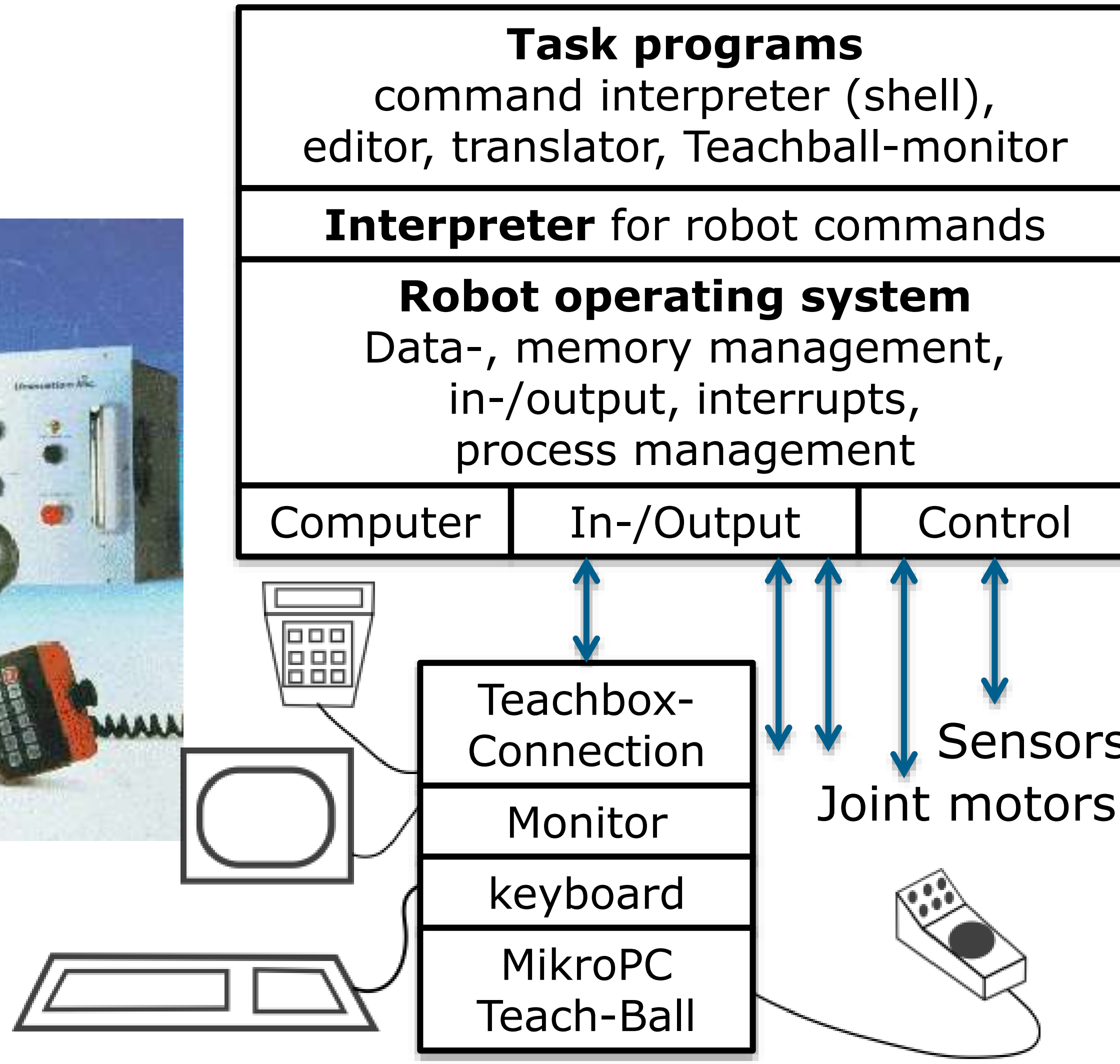
Programming by Examples: Settings: Manual

- Joint motors and breaks are set, such the robot can be moved manually
- Effector is moved manually along desired path
- The path is then defined by a set of intermediate points
- If an endpoint is reached, the joint angles can be saved by pressing a key
- Problems
  - Narrow production cells prevent access at arbitrary positions
  - Heavy robots
  - Dangerous
- Seldom used

Programming by Examples: Teach-In

- Special input hardware to position effector (Teach Box, Teach Pentant)
- Three possibilities
  - Individual movement of each Joint
  - Movement of effector in x-, y-, z-direction (position setting)
  - Rotation around angles O, A, T (orientation setting)
- On key press
  - Save/delete endpoint
  - Start/Abort program
  - Set velocities
- Alternative input methods
  - Joystick, mouse
  - Teach-balls

Programming by Examples: Teach-In

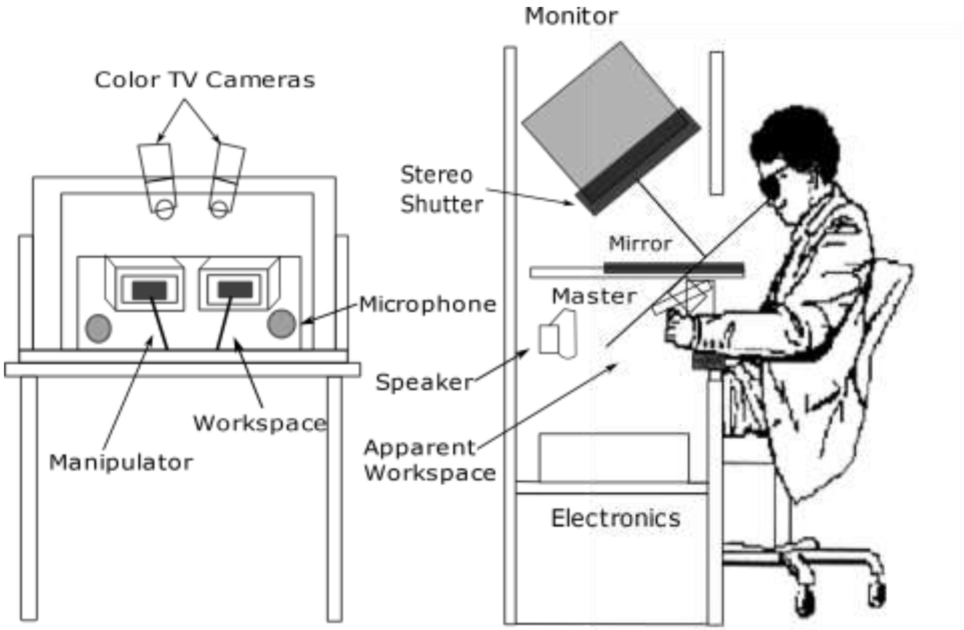


Programming by Examples: Master-Slave

- Manual programming of heavy robots
- Guidance of small and light master-robots
- Transmission of guidance to slave-robot
- Expensive, since two robots are needed
- Mostly used in „teleoperation“

Programming by Examples: Teleoperation

- Use-case: Dangerous environment for humans
- Used in: radiated rooms, under the sea, space
- Like Master-slave-programming (without intermediate points)
- Situation at slave mostly transmitted by a camera
- Many problems
  - Image transmission
  - Forces
  - Transmission times
  - …



Programming by Examples : Advantage/Disadvantages

- Advantages
  - No programming knowledge required
  - No further computers necessary for programming
  - No workspace measurement (WCS position not needed)
  - Programming is done directly with real robots
  - Consideration of all constructive inaccuracies
  - Consideration of all noise
- Disadvantages
  - Inclusion of sensors not possible
  - Path correction with the help of sensor information not possible
- No longer used in (today's) intelligent robots

Programming through Training

- Actions which should be executed are shown to the robot
- Recording of the action via sensors
- Robot repeats the action (training), until the rating (speed, precision, …) is good enough
- External sensors record deviation to goal state
- Program improvements with correction (self-analysis)

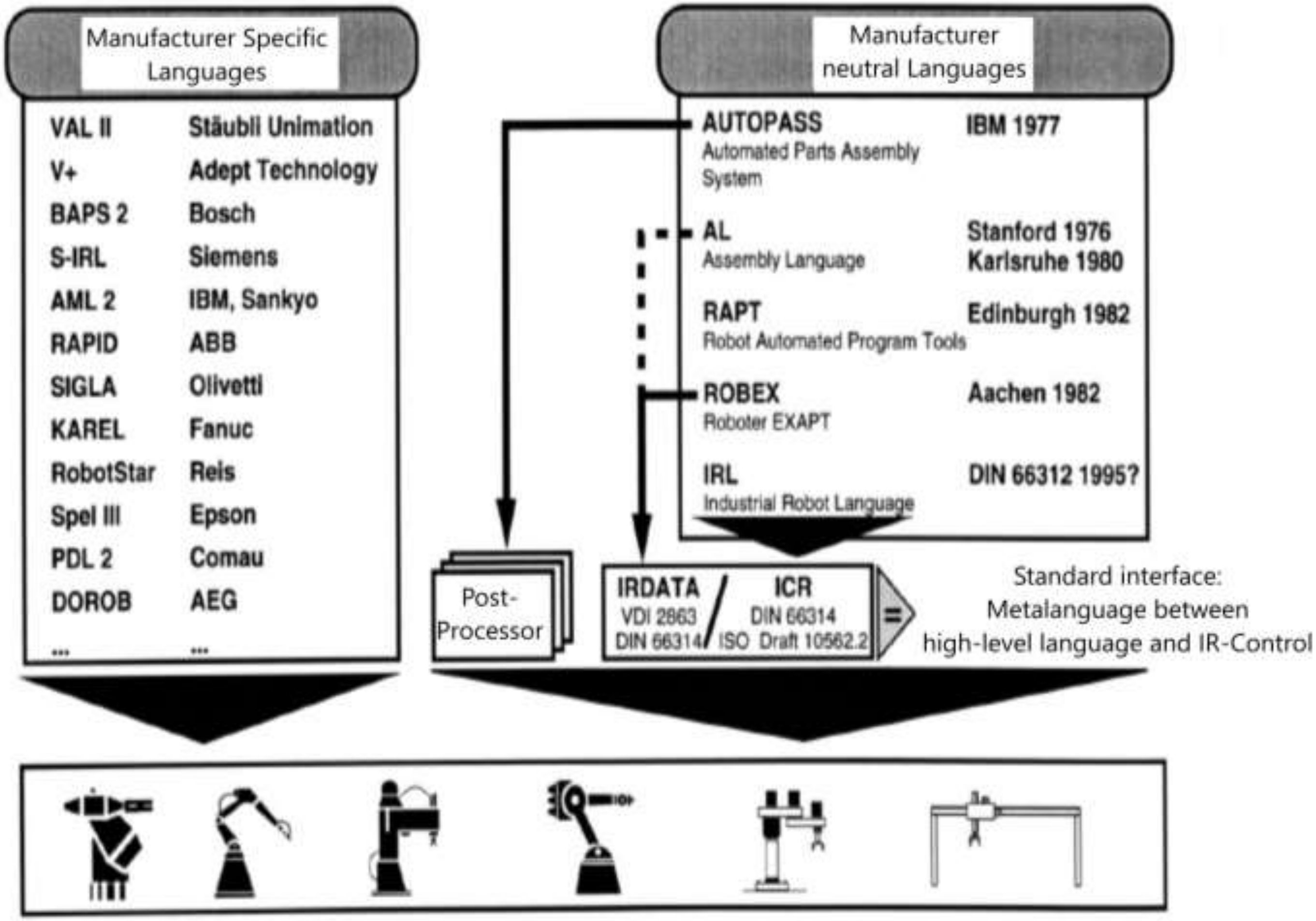
Programming through Training

- Research topic
  - Not yet in real use
  - Today simple programs are possible (shift around bocks, manual insertion
- Image processing and interpretations
  - Object detection
  - Pose- and orientation estimation of objects
  - Tracking of moving objects
- Sensor integration: Evaluation of multiple sensors
- Process analysis: From observed action a sequence of basic operations needs to be extracted

Robot-Oriented Programming

- Robot programs with explicit movement commands (e.g.: „Drive straight to point B“)
- Textual programming in robot programming languages
- Mostly extensions of universal languages (e.g. C)
  - Robot specific datatypes (transformation matrices, operators)
  - Movement commands
  - Effector commands

Robot-Oriented Programming Language



Robot-Oriented Programming: V+ (Adept)

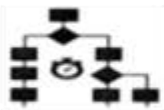
Robotic Movement

MOVE
DRIVE
APPRO
SPEED
FRAME
HERE...

End Effector Control

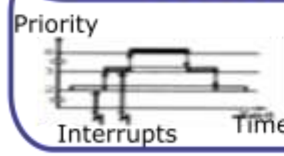
OPEN
CLOSE
RELAX HAND
...

Program flow control



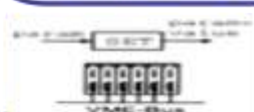
IF...THEN
CASE...OF
DO...UNTIL
FOR...TO
CALL
DELAY...

Event Control



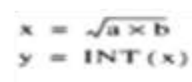
REACT
IGNORE
SET.REVENT
GET.EVENT
ERROR...

Data Processing and Communication



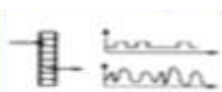
GLOBAL SET
FDELETE
SEND
...

Calculation and operations



+,-,*,/
SIN,COS
INT,ABS
MIN,MAX
TRUE,FALSE...

Signal Processing



SIGNAL()
RESET
RUNSIG
AIO.IN
AIO.OUT...

Sensor control

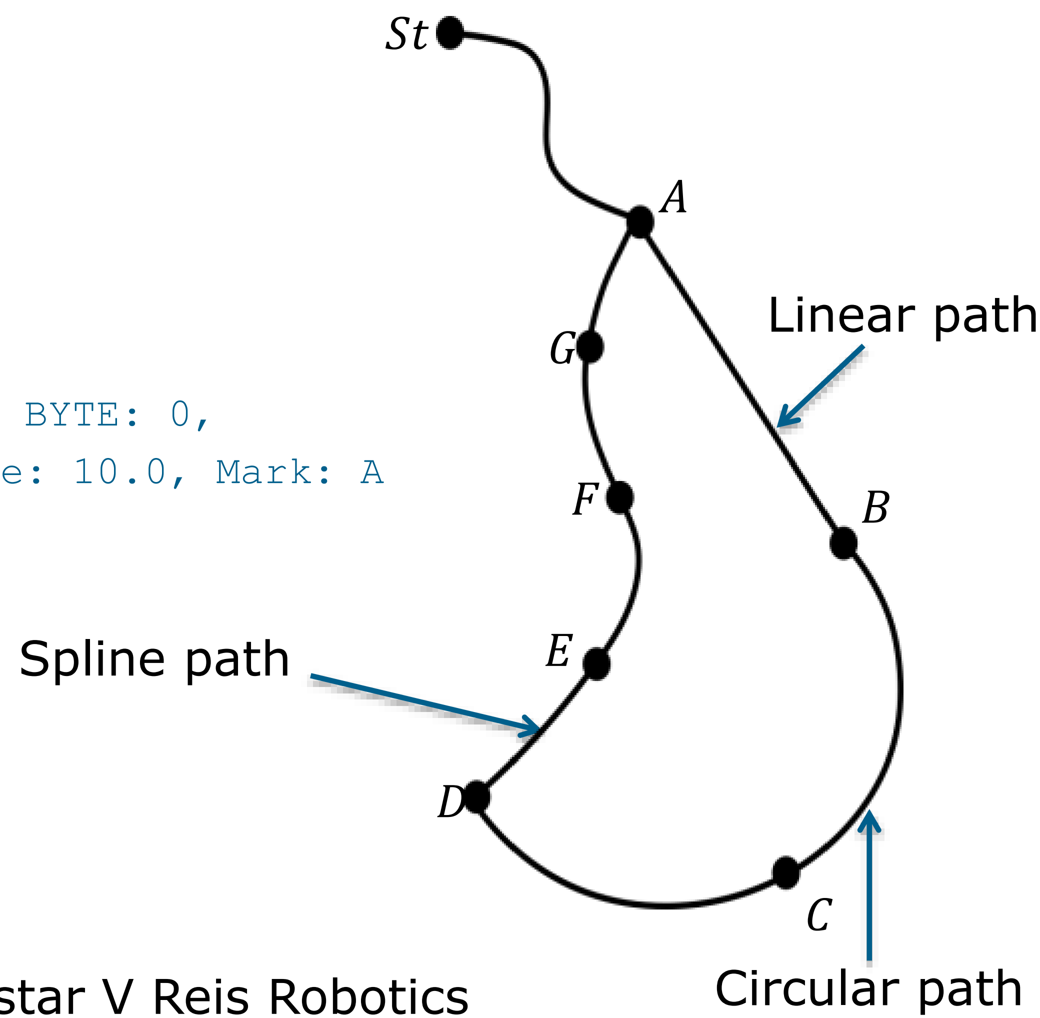
SIGNAL
RESET
FORCE
VLOCATE
VPICTURE...

Other

Technican settings
Graphical Operations
System Configurations
Data Management
...

Robot-Oriented Programming: ROBOTstar

```
    Tool variable T
    Position     A
    MOV_TYPE    #LINEAR
    PATH_VEL 150
    PATH_ACC 60
    POSITION     B
    MOV_TYPE    PTP
    WAIT_BIT    #EINGANG, Level: 1, BYTE: 0,
                 Bit_Nr: 5, Max_Time: 10.0, Mark: A
    MOV_TYPE    CIRK
    POSITION     C
    POSITION     D
    MOV_TYPE    #SPLINE
    POSITION     E
    POSITION     F
    POSITION     G
A:  POSITION     A
```



Programming example of ROBOTstar V Reis Robotics

Robot-Oriented Programming: SRCL

Textual programming with Siemens Robot Control Language

DEF HP 5	DEF-Command sets up main program
GES BAN 20	Sets Path velocity, e.g. 20 m/min
GES ALL 50	With 50% of maximal axial velocity
PTP X1 Y1 Z1	With manual movement keys move robot
A1 B1 C1	in wait position and teach position
GRF 1 AUF	Open gripper
WRT E1 H	Wait for part
PTP X2 Y2 Z2	Move to part
A2 B2 C2	
GRF 1 ZU	Grip part
PTP X3 Y3 Z3	Move to tools

Robot-Oriented Programming: SRCL

LIN X4 Y4 Z4	Inert part
A4 84 C4	
GRF 1 AUF	Open gripper
LIN X5 Y5 Z5	Move arm out of machine
A5 B5 C5	
S IA 1	Start machine
WRT E2 H	Wait for ready signal
LIN X4 Y4 24	Move to part
A4 B4 C4	
GRF 1 ZU	Grip part
LIN X5 Y5 Z5	Move arm out of machine
A5 B5 C5	
PTP X6 Y6 Z6	Move to set position
A6 B6 C6	
GRF 1 AUF	Set part
END HP 1	Program end

Robot-Oriented Programming: Realization/Implementation Variants

- Complete new design of a language
  - Free of practical constraints and implementation details
  - Avoidance of known weaknesses
  - Rich, robot oriented data types
  - AL, VAL, VAL II (Unimation, Puma-Robot)

Code	Definition
X.TO.Y	Name of program
Open	Open robot gripper as it approaches point X
APPRO X,25	Approach point X within $25mm$
MOVE X	Move to point X
CLOSEI	Close jaws immediately
DEPART 25	Back away from point X $25mm$
APPRO Y,25	Approach point Y within $25mm$
MOVE Y	Move to point Y
OPENI	Open robot gripper immediately
DEPART 25	Back away from point $25mm$

Robot-Oriented Programming: Realization/Implementation Variants

- Improvement of automation/control language
  - Improvement of known NC-language is easy
  - Easy transfer of existing programs
  - RAPT of APT for NC-Machines, ROBEX from EXAPT
- Extension of general programming languages with robot oriented language elements
  - Extension easier then new development
  - Usage of existing libraries
  - AUTOPASS embedded in PL/1, PASRO in Pascal

Robot-Oriented Programming: Language Elements

- Commands
  - For movement of one/multiple robots
  - For operation of grippers/tools
  - For external sensors
  - For in-/output of data/signals via interfaces
  - For synchronization/communication with processes
  - For parallel handling
  - For interrupt treatment
  - For logical chains of CS
- Instructions
  - For the calculation of expressions
  - For the control of sequence/flow

Robot-Oriented Programming: Language Elements

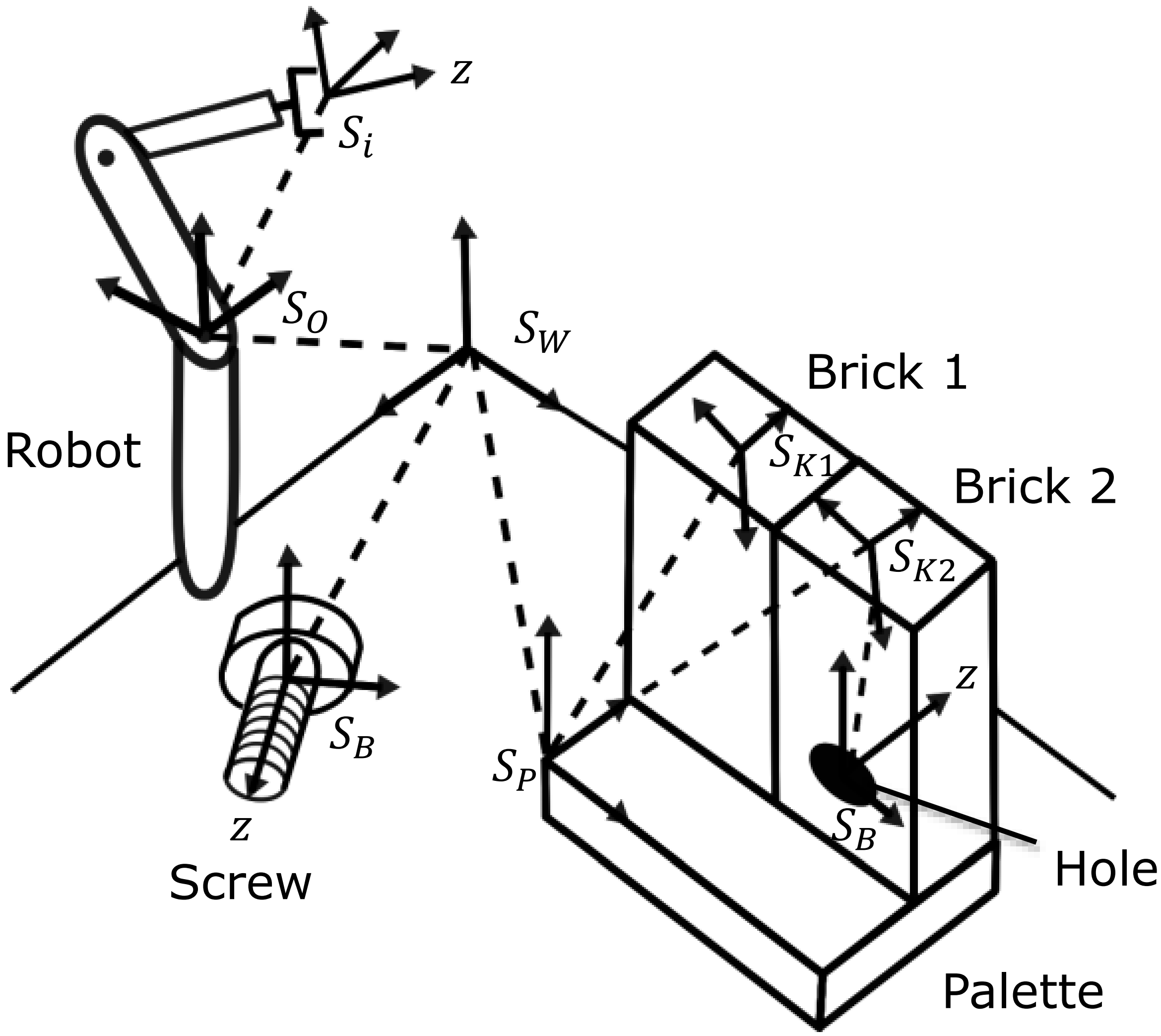
- Real-time processing with period, duration and deadline
- Procedure concept
- Constructor and selector commands for complex, structured data types
- Definition of generic operations by the user

Ideal case	Reality
All language supported	Not all language elements in robot programming languages

Robot-Oriented Programming: Example for Robot`s World

- Cartesian coordinates for path points
- Advantages
  - Knowledge about robot-specific kinematics at the joint level not needed
  - Readable programs
  - Cartesian positions easier than joint angle information
  - Direct conversion of construction data into programs
  - Easy transfer of programs to other robots
  - Type and number joints are hidden for the programmer
- Disadvantages
  - Cartesian position is needed to manipulate objects
  - Working space must be exactly measured

Robot-Oriented Programming: Example for Robot's World



Robot-Oriented Programming: Data Types

- VECTOR: Points in space with homogeneous coordinates
- RotMatrix: Rotational matrices
- TransMatrix: Homogeneous transformation matrices
- FRAME: Frames
- JointPosition: Joint angles for rotational joints and translations for translational joints

Robot-Oriented Programming: Data Objects

- STARTPOS or PARKPOS: Parking position of the robot
- WORLD: Frame of the WCS
- BASE: BCS-frame of the robot relative to WCS
- HAND: Frame, representing the adapter-CS
- TOOL: Effector-frame (working point) relative to HAND
- Xvector: Homogeneous coordinates (1,0,0,1)
- Yvector: Homogeneous coordinates (0,1,0,1)
- Zvector: Homogeneous coordinates (0,0,1,1)
- NULLVEKTOR: (0,0,0,1)
- IdRotMatrix, IdTransMatrix: Identity matrix
- RobError: Variable, containing last error of the program

Robot-Oriented Programming: Expressions

$$\begin{array}{ccccc}
\text{SCALAR} & * & \text{VECTOR} & \rightarrow & \text{VECTOR} \\
\text{VECTOR} & \pm & \text{VECTOR} & \rightarrow & \text{VECTOR} \\
\text{VECTOR} & * & \text{VECTOR} & \rightarrow & \text{SCALAR} \\
\text{VECTOR} & \times & \text{VECTOR} & \rightarrow & \text{VECTOR} \\
\text{RotMatrix} & * & \text{VECTOR} & \rightarrow & \text{VECTOR} \\
\text{RotMatrix} & * & \text{RotMatrix} & \rightarrow & \text{RotMatrix} \\
\text{TransMatrix} & \pm & \text{VECTOR} & \rightarrow & \text{TransMatrix} \\
\text{TransMatrix} & * & \text{VECTOR} & \rightarrow & \text{VECTOR} \\
\text{TransMatrix} & * & \text{TransMatrix} & \rightarrow & \text{TransMatrix} \\
\text{FRAME} & \pm & \text{VECTOR} & \rightarrow & \text{FRAME} \\
\text{FRAME} & * & \text{VECTOR} & \rightarrow & \text{VECTOR} \\
\text{FRAME} & * & \text{TransMatrix} & \rightarrow & \text{FRAME}
\end{array}$$

Robot-Oriented Programming: Communication

- Synchronization with robot and tools
  - Signals
  - Messages
  - Explicit wait commands: `WAITROBOTER`, `WAITTIME`
- Enables all types of synchronization without explicit language message (semaphore, monitor)

Robot-Oriented Programming: Interruption

- For real-time languages operations need to be interruptible
- Asynchronous execution of a user-defined interruption handling
- Interrupt events
  - Messages
  - Signal `ON` and since last `OFF`-state no interruption was triggered
  - Alarm in the program
  - Alarm of robot control
  - Alarm of operating system

Robot-Oriented Programming: Interruption

- User-defined interruption handling procedure defined (IHP) for certain events
- Mapping of IHP and priorities of the event
- Execute interruption if no IHP is active, or the active procedure has smaller priority
- Blocked interruptions are reset
- Robot operating system organizes IHP depending on priority
- Jump to interruption point if IHP is stopped with `RETURN`

Task-Oriented Programming

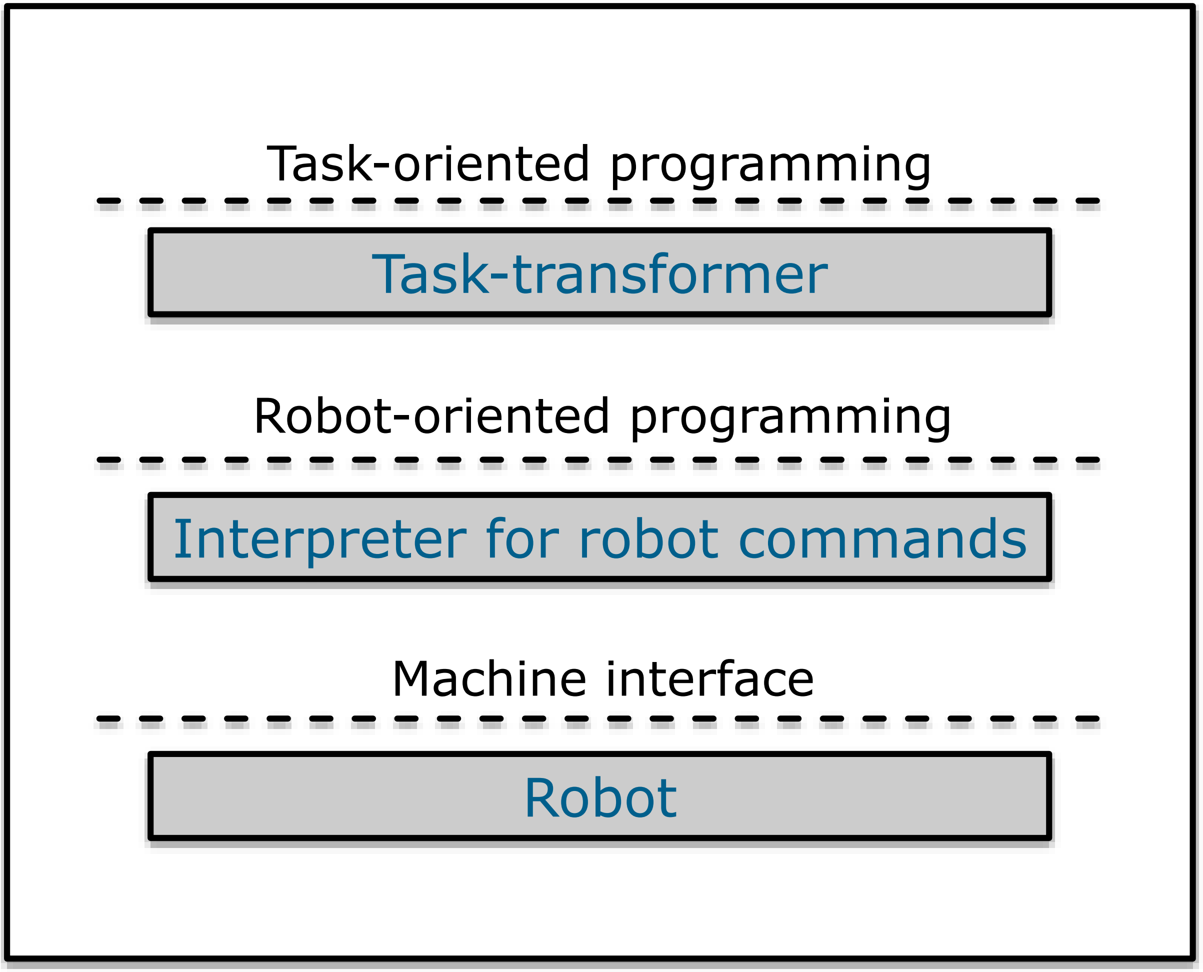
- Execution of abstraction level (abstracter then normal robot programming language
- Textual programming
- Programming level mostly for intelligent robots

Robot-oriented	Task-oriented
How a robot solves a task	**What** the robot should do

Task-Oriented Programming: Task Scheduler

- Creates robot program in robot oriented language
- Needed:
  - Fact knowledge: Knowledge basis with environmental models (factory, working cell, robots, machines, …)
  - Operational knowledge: Knowledge base with rules for division of tasks into subtasks
  - Different algorithms for mounting, gripping and path planning as well as sensor integration
  - Synchronization pattern for coordination of robot tasks with its environment

Task-Oriented Programming : Task Transformer Layer Model



Learning process	Textual	Graphic/Interactive
Movement-oriented	Flow-oriented	Movement-oriented
Easy (learnable)	Complex (Pre-knowledge)	Easy/complex
Logical program development must be supplemented	Real position values must be supplemented	Position values must be corrected
Online processing (production equipment blocked)	Offline processing	
No documentation	Good documentation	
Bad correction options (possibilities)	Easy correction options	
-	-	Collision handling is possible through simulation
-	-	Determination of cycle times etc. by simulation
Low HW/SW-effort	Average HW/SW effort	High HW/SW effort

Procedure: Application criteria

	Learning Methods	Textual	Graphical/ interactive
Kinematics	Easy	Arbitrary	
Periphery	Less	Bulky	Less
Sensor	Rarely	Arbitrary	Rarely
Range of task	Narrow (movement-oriented)	Broad spectrum	
Programming effort	Low	High	High
Qualification	Low	Middle	Middle
Others			For planning tasks

Next Lecture

- Application
  - „Human-centered automation“
  - Robot system and control concepts
  - Test results
- Summary and overview