Programming
Supervision and control architectures

Prof. Alessandro De Luca
Robot programming

- real-time operating system
- sensory data reading
- motion control execution
- world modeling
- physical/cognitive interaction with the robot
- fault detection
- error recovery to correct operative conditions
- programming language (data structure + instruction set)

Programming environments will depend also on the level at which an operator has access to the functional architecture of the robot.
Programming by teaching

- “first generation” languages
- programming by directly executing (*teaching-by-showing*)
  - the operator guides (manually or via a teach-box) the robot along the desired path (off-line mode)
  - robot joint positions are sampled, stored, and interpolated for later repetition in on-line mode (access to the primitives level)
  - automatic generation of code skeleton (later modifications of parameters is possible): no need of special programming skills
- access to the **primitive level**
- early applications: spot welding, spray painting, palletizing
- examples of languages: T3 (Milacron), FUNKY (IBM)
Robot-oriented programming

- “second generation” languages: structured programming with characteristics of an interpreted language (interactive programming environment)
- typical instructions of high-level languages are present (e.g., logical branching and while loops)
  - ad-hoc structured robot programming languages (more common)
  - development of robotic libraries in standard languages (preferred)
- access to the action level
- handle more complex applications where the robot needs to cooperate/synchronize with other machines in a work cell
- examples of languages: VAL II (Unimation), AML (IBM), PDL 2 (Comau), KRL (KUKA)
KUKA user interfaces

- Teach pendant
- KRL programming
- Ethernet RSI XML

- Fast Research Interface
KRL language

- **basic instruction set:**

<table>
<thead>
<tr>
<th>Variables and declarations</th>
<th>Inputs/outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECL</td>
<td>ANIN</td>
</tr>
<tr>
<td>ENUM</td>
<td>ANOUT</td>
</tr>
<tr>
<td>IMPORT ... IS</td>
<td>DIGIN</td>
</tr>
<tr>
<td>STRUC</td>
<td>PULSE</td>
</tr>
<tr>
<td></td>
<td>SIGNAL</td>
</tr>
</tbody>
</table>

- **Motion programming**

  | CIRC            | ANIN          |
  | CIRC_REL        | ANIN          |
  | LIN             | DIGIN         |
  | LIN_REL         | PULSE         |
  | PTP             | SIGNAL        |
  | PTP_REL         |               |

- **Program execution control**

  | CONTINUE        | RETURN        |
  | EXIT           |               |
  | FOR ... TO ... ENDFOR |               |
  | GOTO           |               |
  | HALT           |               |
  | IF ... THEN ... ENDIF |               |
  | LOOP ... ENLOOP |               |
  | REPEAT ... UNTIL |               |
  | SWITCH ... CASE ... ENDSWITCH |               |
  | WAIT ... FOR   |               |
  | WAIT ... SEC   |               |
  | WHILE ... ENDWHILE |               |

- **Inputs/outputs**

  | ANIN          | ANOUT         |
  | DIGIN         | PULSE         |
  | SIGNAL        |               |

- **Subprograms and functions**

  | RETURN        |               |

- **Interrupt programming**

  | BRAKE          |               |
  | INTERRUPT      |               |
  | INTERRUPT ... DECL ... WHEN DO |               |
  | RESUME         |               |

- **Path-related switching actions (=Trigger)**

  | TRIGGER WHEN DISTANCE |               |
  | TRIGGER WHEN PATH    |               |

- **Communication**

  | VARSTATE()       |               |

- **System functions**

  | VARSTATE()       |               |
KRL language

- typical motion primitives

PTP motion (point-to-point, linear in joint space)
LIN motion (linear in Cartesian space)
CIRC motion (circular in Cartesian space)
CONST orientation
end-effector orientation
PTP motion (linear in RPY angles)
KRL language

- multiple coordinate frames (in Cartesian space) and jogging of robot joints
cyclical data transmission from the robot controller to an external system (e.g., position data, axis angles, operating mode, etc.) and vice versa (e.g., sensor data) in the interpolation cycle of 12 ms

- influencing the robot in the interpolation cycle by means of an external program
- direct intervention in the path planning of the robot
- recording/diagnosis of internal signals
- communication module with access to standard Ethernet via TCP/IP protocol as XML strings (real-time capable link)
- freely definable inputs and outputs of the communication object
- data exchange timeout monitoring
Example of RSI use - 1

- deburring task with robot motion **controlled by a force sensor**

1. work piece to be deburred along the edge under force control
2. tool with force sensor
3. robot
4. robot controller

$F_X$ measured force in the X direction of the BASE coordinate system (perpendicular to the programmed path)

$\vec{v}$ direction of motion

$\text{LIN\_REL} = \text{linear Cartesian path relative to an initial position (specified here by the force sensor signal)}$
Example of RSI use - 2

- **redundancy resolution** on cyclic Cartesian paths
  - task involves position only \((m=3, n=6\) for the KUKA KR5 Sixx)
  - **without joint range limits** or including **virtual limits**
Example of RSI use - 3

- human-robot interaction through vocal and gesture commands
- voice and human gestures acquired through a Kinect sensor

Kinect RGB-D sensor (with microphone)

simple vocabulary, e.g.:
- listen to me
- give me
- follow
  - right/left hand
  - the nearest hand
- thank you
- stop collaboration

video
Fast Research Interface (FRI) for KUKA Light Weight Robot (LWR-IV)

- UDP socket communication up to 1 KHz (1÷100 ms cycle time)

here, we develop our C++/ROS code for:
- trajectory planning
- kinematic control
- redundancy resolution
- torque/dynamic control
- physical HRI
- ...

available at DIAG Robotics Lab since Sep 2012
Kinematic control using the FRI
KUKA Light Weight Robot (LWR-IV)

- joint velocity commands that mimic second-order control laws (defined in terms of acceleration or torques), exploiting task redundancy of the robot
- discrete-time implementation is simpler and still very accurate
Other uses of the FRI

- Haptic feedback to the user

- Coordinated dual-arm motion
Robot research software

- a (partial) list of open source robot software
  - for simulation and/or real-time control
  - for interfacing with devices and sensors
  - research oriented

Player/Stage playerstage.sourceforge.net
  - networked robotics server (running on Linux, Mac OS X) as an abstraction layer supporting a variety of hardware + 2D robot simulation environment
  - **Gazebo**: 3D robot simulator (with ODE physics engine and OpenGL rendering), now an independent project

VREP (edu version) www.coppeliarobotics.com
  - each object/model controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution
  - controllers written in C/C++, Python, Java, Matlab, ...
Robotics Toolbox (free addition to Matlab) www.petercorke.com
- study and simulation of kinematics, dynamics, and trajectory generation for serial-link manipulators

OpenRDK openrdk.sourceforge.net
- “agents”: modular processes dynamically activated, with blackboard-type communication (repository)

ROS (Robot Operating System) www.ros.org/wiki
- middleware with: hardware abstraction, device drivers, libraries, visualizers, message-passing, package management
- “nodes”: executable code (in Python, C++) running with a publish/subscribe communication style

Pyro (Python Robotics) pyrorobotics.org
Task-oriented programming

- “third generation” languages (for research, not yet available on the market)
- similar to object-oriented programming
- task specified by high-level instructions performing actions on the parts present in the scene (artificial intelligence)
- understanding and reasoning about a dynamic environment around the robot
- access to the task level
Functional control architecture

Reference model

Sensor processing

Knowledge models

Decision strategies

Task level

Action level

Primitives level

Servo level

Operator

Global memory

Sensors

Actuators

Robotics 1
Functional architecture: Modules

horizontal decomposition

reference model

sensor processing

knowledge models

decision strategies

SENSORY MODULES
acquisition, processing and integration of sensory data

global memory

operator

sensors

actuators

primitives level

task level

Robotics 1
Functional architecture: Modules

MODELING MODULES
a priori knowledge about robot + environment system, updated using information from sensory modules
**Functional architecture: Modules**

**DECISION MODULES**
- decomposition (in time and space) of tasks into actions of lower level
- choice and processing of strategies
Functional architecture: Modules

horizontal decomposition

reference model

GLOBAL MEMORY

data and information relevant to all levels (updated estimate of robot + environment state)

sensors

global memory

knowledge models

decision strategies

operator

GLOBAL MEMORY

data and information relevant to all levels (updated estimate of robot + environment state)

task level

information flow

primitives level

servo level

interface

Robotics 1
Functional architecture: Modules

Horizontal decomposition

Reference model

Sensor processing
Knowledge models
Decision strategies

OPERATOR INTERFACE
allows intervention by an operator at any level of the functional hierarchy

Sensors
Global memory
Actuators
Operator interface

Robotics 1
Reference model: Levels

- **task level**: objective of the task (as specified by the user) analyzed and decomposed into actions (based on knowledge models about the robot and the environment systems)
- **action level**: symbolic commands converted into sequences of intermediate configurations
- **primitives level**: reference trajectories generation for the servo level, choice of a control strategy
- **servo level**: implementation of control algorithms, real-time computation of driving commands for the actuating servomotors
A functional architecture for industrial robots

- camera
- force
- velocity
- position

- request
- data

- state

- action
- reference frames
- path points
- interpolation modes
- primitives

- control algorithm
- \( q_{\text{des}} \)
- \( q_{\text{des2}} \)
- \( q_{\text{des3}} \)

- servo

- actuator commands

Robotics 1
A functional architecture for industrial robots

**ACTION LEVEL**
- interpreter of high-level commands
- task decomposition made by human operator
- no sensory and modeling modules (unless a multi-modal cognitive human-robot interaction is possible)

vertical decomposition

**ACTION LEVEL**
- interpreter of high-level commands
- task decomposition made by human operator
- no sensory and modeling modules (unless a multi-modal cognitive human-robot interaction is possible)

vertical decomposition

- camera
- force velocity position
- data
- reference frames path points interpolation modes
- primitives
- q_{des} q_{des} q_{des}
- control algorithm
- servo
- actuator commands

Robotics 1
A functional architecture for industrial robots

PRIMITIVES LEVEL

• **S:** (only for an active interaction with the environment) world geometry, interaction state
• **M:** direct and inverse kinematics, dynamic models
• **D:** command encoding, path generation, trajectory interpolation, kinematic inversion, analysis of servo state, emergency handling
A functional architecture for industrial robots

SERVO LEVEL
- **S**: signal conditioning, internal state of manipulator, state of interaction with environment
- **M**: direct kinematics, Jacobian, inverse dynamics
- **D**: command encoding, micro-interpolation, error handling, digital control laws, servo interface
Interaction among modules

- horizontal activation
  - sequential

- vertical activation
  - on demand

Sensors ➔ perception ➔ modeling ➔ planning ➔ task execution ➔ motor control ➔ actuators

Plan changes to world

- identify object
- monitor changes
- build map
- explore
- wander
- avoid obstacles

Sensors ➔ actuators
LAAS architecture

- alternative example by LAAS/CNRS in Toulouse
- five levels
  - decision
  - execution (synchronization)
  - functional (modules)
  - logical for interface
  - physical devices

R. Alami et al.
“An Architecture for Autonomy,”
Development of architectures - 1

example: a navigation task for a wheeled mobile robot

- hierarchical system
  - initial localization
  - off-line planning
  - on-line motion control
  - possible acquisition/update of a model of the environment = map (at a slow time scale)
Development of architectures - 2

- pure reactive system
  - global positioning task (goal)
  - on-line estimate of the local environment (unknown)
  - local reaction strategy for obstacle avoidance and guidance toward the goal
Development of architectures - 3

- hybrid system
  - SLAM = simultaneous localization and mapping
  - navigation/exploration on the current model (map)
  - sensory data fusion
  - on-line motion control
IPA robotic cell for garbage collection and separation for recycling

Semi-automatic robotized garbage collection
Raffaella Mattone, Linda Adduci
c/o Fraunhofer IPA, Stuttgart, 1997

Objective: replace operator

Semi-automatic version at Fraunhofer IPA Stuttgart, 1997
Sensory module in fully automatic version

operator + touch-screen
replaced by
structured light vision +
neuro-fuzzy system for object localization and classification

operation principle of the structured light sensor
Sensory data interpretation

possible sources of lack of information on a single line scan
Sensory data interpretation

Integration of data collected in successive sampling instants.
Decision module

$h_1 | h_2 | h_3 | h_4 | ...$ → vector of height samples

Rule-Level I

$s_1 | e_1 | a_1$
$s_2 | e_2 | a_2$

Rule-Level II

$x \ y \ z$

Classification results

structure of the object localization and classification module
Modeling module

example of models for objects on the conveyor belt
Functional architecture of the IPA cell

- camera + laser
- compute h on each line scan
- current model of objects on conveyor
- object present in gripper
- force (gripper) velocity
- position
- data
- request
- state
- localization classification
- classification
- object present in gripper
- state
- servo
- actuator commands
- request
- state
- primitives
- reference frames
- path points
- interpolation modes
- $p_{des}$
- actions
- request
- state
- data
- data
- request
- state
- data
Test results

includes optimal scheduling of pick & place operations to maximize throughput (minimize loss of pieces)

work by Dr. Raffaella Mattone (PhD @ DIS)

Automatic robotized garbage collection

Raffaella Mattone, Linda Adduci

c/o Fraunhofer IPA. Stuttgart, 1997
Flow diagrams of operation

PETRI NETS
oriented graphs with two types of nodes

- **places** \((p_1, \ldots, p_4)\)
  states or functional blocks: active if a “token” is present (e.g., \(p_1\) and \(p_3\))

- **transitions** \((t_1, \ldots, t_3)\)
  changes from a state to another state, **fired** by events: if enough (at least one) tokens are present in all input places of a transition, tokens are moved to the output places; transitions may be timed (e.g., \(t_1\) and \(t_3\))
Petri net model of the IPA cell

- $p_1$: robot picking & placing
  - $T_1$: pick & place time
- $p_2$: robot ready
- $p_3$: new part on conveyor
- $p_4$: waiting for a part
  - $T_3$ (random variable): time interval between two successive parts

Initial marking/state: robot ready, waiting for a part
Hardware architecture

**SYSTEM** includes: one/multiple microprocessor(s), local/shared RAM, EPROM, interrupt handler, ...

- **EXTERNAL MEMORY**
- **TEACH PENDANT**
- **KEYBOARD**
- **I/O**
- **SYSTEM**
- **KINEMATICS**
- **DYNAMICS**
- **SERVO**
- **FORCE**
- **VISION**
- **BUS**
- **POWER AMP**
- **CAMERA**
- **MONITOR**

- position/velocity transducers
- force sensor
- servomotors

Robotics 1
Hardware architecture
Example of the IPA cell

- **EXTERNAL MEMORY**
- **I/O**
- **TEACH PENDANT**
- **KEYBOARD**
- **SYSTEM**
- **STRUCTURED VISION**
- **MONITOR**
- **CAMERA + LASER**
- **BUS**

**SCARA robot**
- **SERVO**
- **FORCE**
- **KINEMATICS**

- position/velocity transducers
- **POWER AMP**
- force sensor
- servomotors
Hardware architecture
Example including vision in an open controller

Robot COMAU
SMART-3S

Robotics 1