Robotics 2

Introduction to Control

Prof. Alessandro De Luca
What do we mean by robot control?

- different level of definitions may be given to robot control
  - successfully complete a task or work program
  - accurate execution of a motion trajectory
  - zeroing a positioning error

⇒ control system unit has a hierarchical internal structure

- different but cooperating models, objectives, methods are used at the various control layers
Evaluation of control performance

- **quality** of execution in **nominal** conditions
  - velocity/speed of task completion
  - accuracy/repeatability (in **static** and **dynamic** terms)
  - energy requirements

  ⇒ improvements also thanks to **models** (software!)

- **robustness** in **perturbed** conditions
  - adaptation to changing environments
  - high repeatability despite disturbances, changes of parameters, uncertainties, modeling errors

  ⇒ can be improved by a generalized use of **feedback**, using more **sensor information**
Static positioning
accuracy and repeatability

- Poor accuracy
  - Poor repeatability

- Good accuracy
  - Poor repeatability

- Poor accuracy
  - Good repeatability

- Good accuracy
  - Good repeatability
Control schemes and uncertainty

- **feedback control**
  - insensitivity to mild disturbances and small variations of parameters

- **robust control**
  - tolerates relatively large uncertainties of known range

- **adaptive control**
  - improves performance on line, adapting the control law to a priori unknown range of uncertainties and/or large (but not too fast) parameter variations

- **intelligent control**
  - performance improved based on experience: **LEARNING**
  - autonomous change of internal structure for optimizing system behavior: **SELF-ORGANIZING**

Uncertainty on parametric values $\rightarrow$ **IDENTIFICATION**

... on the system structure $\rightarrow$ ...
Basic schemes

METHODS

MODELS

combination of feedforward and feedback commands

control → robot → environment

open-loop command

action

closed-loop commands

control → robot → environment

- METHODS
- MODELS
Limits in control of industrial robots - 1

- from a **functional** viewpoint
  - “closed” control architectures, relatively difficult to interface with external computing systems and sensing devices
  ⇒ especially in applications where **real time** is a must

- at **higher** level
  - open loop command generation
  ⇒ exteroceptive sensory feedback absent or very loose

- at **intermediate** level
  - limited consideration of advanced kinematic and dynamic issues
  ⇒ e.g., singularity robustness: solved on a case-by-case basis
  ⇒ task redundancy: no automatic handling of the extra degrees of freedom of the robot
Limits in control of industrial robots - 2

- at **low (direct)** level
  - reduced execution speed ("control bandwidth")
    - typically heavy mechanical structure
  - reduced dynamic accuracy on fast motion trajectories
    - standard use of kinematic control + PID
  - problems with dry friction and backlash at the joints
  - compliance in the robot structure
    - flexible transmissions
      - (belts, harmonic drives, long shafts)
    - large structures or relatively lightweight links

Now **desired** for safe physical Human-Robot Interaction

- need the use of **dynamic models** and model-based **control laws**
- addressed, e.g., by using **direct drive** actuators
Example of robot positioning

- low damped vibrations due to joint elasticity

- 6R KUKA KR-15/2 robot (235 kg), with 15 kg payload
Advanced robot control laws

- deeper mathematical/physical analysis and modeling of robot components (model-based approach)
- schemes using various/different control loops at different hierarchical levels (feedback) and with additional sensors
  - visual servoing
  - force/torque sensors for interaction control
  - ...
- “new” methods
  - integration of (open-loop/feedforward) motion planning and feedback control aspects (e.g., sensor-based planning)
    - fast (sensor-based) re-planning
    - model predictive control (with preview)
  - learning (iterative, by imitation, skill transfer, ...)
  - ...

Robotics 2
Example of visual-based control

- human-obstacle collision avoidance

- 3R SoftArm prototype with McKibben actuators (Univ. of Pisa) using repulsive force field built from stereo camera information

video
Functional structure of a control unit

SENSORS:
- optical encoders
- velocity tachos
- strain gauges
- joint or wrist F/T sensors
- tactile sensors
- micro-switches
- range/depth sensors
- laser
- CCD cameras
- ...
Task-Object: insert P1 into H5
Object-Object: move APPR frame #13
Robot-Object: rotate joint 3 by -45°

use TEACH BOX in industrial robots

Functional structure of a control unit:

1. **Task program**
   - Java, Lisp, expert- and rule-based systems
2. **Trajectory planning**
   - Matlab, C++, Python
3. **Direct control algorithms**
   - Assembler (PICs), C, C++

- **Actuators**
- **Robot**
- **Environment**
Functional structure of a control unit

- task program
  - trajectory planning
    - direct control algorithms
      - actuators
        - robot
          - environment
  - modeling of tasks
    - geometric and kinematic models
      - coordinate transformations
    - nonlinear methods
      - dynamic control
        - (electrical and mechanical)
          - dynamic models
    - structured and unstructured world modeling
Robot control/research software
(last updated in December 2015)

▪ 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, ...
Robot control/research software (cont’d)

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
OROCOS control software

- **OROCOS** (Open RObot COntrol Software) [http://www.orocos.org](http://www.orocos.org)
  - open-source, portable C++ libraries for robot control
  - Real-Time Toolkit (for Linux, Mac OS X, Windows Visual Studio)
  - supports CORBA for distributed network computing and ROS interface
  - (user-defined) application libraries
Example application using OROCOS

multi-sensor fusion for multi-robot manipulation in a human populated environment (KU Leuven)
Summarizing ...

- to **improve performance** of robot controllers
  1. more complete **modeling** (kinematics and dynamics)
  2. introduction of **feedback** throughout all hierarchical levels
- **dynamic control** at low level allows in principle
  1. much **higher accuracy** on generic motion trajectories
  2. larger **velocity** in task execution with same accuracy
- **interplay between** control, mechanics, electronics
  1. able to control accurately also lightweight/compliant robots
  2. full utilization of task-related **redundancy**
  3. smart **mechanical design** can reduce control efforts (e.g., closed kinematic chains simplifying robot inertia matrix)
  4. **actuators** with higher dynamic performance (e.g., direct drives) and/or including controlled variable stiffness

needless to say, applications should **justify additional costs**
(e.g., laser cutting with 10g accelerations, human-robot safe interaction)
Benefits of model-based control

- trajectory tracking task: comparison between standard industrial and new model-based controller
Robot learning by imitation

- learning from human motion primitives (imitation)
- motion refinement by kinesthetic teaching (with impedance control)

@TUM, Munich (D. Lee, C. Ott), for the EU SAPHARI project
Using visual or depth sensor feedback

- robust visual or depth (Kinect) feedback for motion tracking

- collision avoidance schemes (here, redundancy w.r.t. an E-E task)
Panoramic view of control laws

- problems and methods for robot manipulators that will be considered

<table>
<thead>
<tr>
<th>type of task</th>
<th>definition of error</th>
<th>joint space</th>
<th>Cartesian space</th>
<th>task space</th>
</tr>
</thead>
<tbody>
<tr>
<td>free motion</td>
<td>regulation</td>
<td>PD, PID, gravity compensation, iterative learning</td>
<td>PD with gravity compensation</td>
<td>visual servoing (kinematic approach)</td>
</tr>
<tr>
<td>trajectory tracking</td>
<td>feedback linearization, inverse dynamics + PD, passivity-based control, robust/adaptive control</td>
<td>feedback linearization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>contact motion</td>
<td></td>
<td>-</td>
<td>impedance control (with variants)</td>
<td>hybrid force-velocity control</td>
</tr>
</tbody>
</table>