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/uncertain** 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
  - learn through repeated robot trials/human experience
Static positioning
accuracy and repeatability

poor accuracy
poor repeatability

poor accuracy
good repeatability

good accuracy
poor repeatability

good accuracy
good repeatability

what about “dynamic” accuracy on (test or selected) motion trajectories?
Basic control schemes

- **Open-loop control**
  - Control \(\rightarrow\) Robot \(\rightarrow\) Environment
  - Commands
    - Open-loop command

- **Closed-loop control**
  - Control \(\rightarrow\) Robot \(\rightarrow\) Environment
  - Perception
    - Feedback
  - Commands
    - Combination of feedforward and feedback commands
    - Closed-loop commands
      - METHODS
      - MODELS
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 → **IDENTIFICATION**

... on the system structure → ...
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 **hard real-time** operation is a must

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

- at the **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 the lower (direct) level
  - reduced execution speed ("control bandwidth")
    ⇒ typically heavy mechanical structure
  - reduced dynamic accuracy on fast motion trajectories
    ⇒ standard use of kinematic control + PID only
  - 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 to include better dynamic models and model-based control laws
- handled, e.g., using direct-drive actuators or online friction compensation
Example of robot positioning

- low damped vibrations due to joint elasticity

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

video

without modeling and explicit control of joint elasticity
Advanced robot control laws

- deeper mathematical/physical analysis and modeling of robot components (model-based approach)
- schemes using various control loops at different/multiple 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, ...)
  - ...
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
Functional structure of a control unit

sensor measurements

SENORS:
optical encoders,
velocity tachos,
strain gauges,
joint or wrist
F/T sensors,
tactile sensors,
micro-switches,
range/depth
sensors, laser,
CCD cameras,
RGB-D cameras
...

task
program

fault detection, vision

trajectory planning

vision, force

direct control
algorithms

position velocity

actuators

proprioceptive sensors

robot

exteroceptive sensors
(also “virtual” ones, i.e., model-based)

environment
Functional structure of a control unit programming languages

- **Task program**
  - Java, Lisp, expert- and rule-based systems

- **Trajectory planning**
  - Matlab, C++, Python

- **Direct control algorithms**
  - Assembler (PICs), C, C++

- **Actuators**

- **Robot**

- **Environment**

- **T-O:** insert P1 into H5
  - **O-O:** move APPR frame #13
  - **R-O:** rotate joint 3 by -45°

- Often “addressed” using the manual TEACH BOX in conventional industrial robots
Functional structure of a control unit modeling issues

- 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 (and acquisition)
Robot control/research software
(last updated in April 2020)

- 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 ⇒ github.com/rtv/stage
  - Stage: in origin, a networked Linux/MacOS X robotics server serving as abstraction layer to support a variety of hardware ⇒ now a 2(.5)D mobile robot standalone simulation environment
  - Gazebo: 3D robot simulator (ODE physics engine and OpenGL rendering), now an independent project ⇒ gazebosim.org

CoppeliaSIM (ex VREP; edu version available) 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) petercorke.com
- study and simulation of kinematics, dynamics, trajectory planning, control, and vision for serial manipulators and beyond ⇒ releases 9 & 10

ROS (Robot Operating System) ros.org
- 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
- drivers, tools, state-of-the-art algorithms ... (all open source)

PyRobotics (Python API) pypi.org/project/pyRobotics (v1.8 in 2015)

OpenRDK openrdk.sourceforge.net ⇒ developed @DIAG, but dismissed
- “agents”: modular processes dynamically activated, with blackboard-type communication (repository)
OROCOS control software

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

⇒ [github](https://github.com/orocos)
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

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

- trajectory tracking task: comparison between standard industrial and new model-based controller

![Image showing robot arms and labeled planes]

- three squares in:
  - horizontal plane
  - vertical front plane
  - vertical sagittal plane

(video)
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 & methods for robot manipulators that will be considered (control command is always a joint torque, if not else specified)

<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 scheme)</td>
</tr>
<tr>
<td></td>
<td>trajectory tracking</td>
<td>feedback linearization, inverse dynamics + PD, passivity-based control, robust/adaptive control</td>
<td>feedback linearization</td>
<td></td>
</tr>
<tr>
<td>contact motion (with force exchange)</td>
<td></td>
<td></td>
<td>impedance control (with variants), admittance control (kinematic scheme)</td>
<td>hybrid force-velocity control</td>
</tr>
</tbody>
</table>
Control laws: dynamic or kinematic

- **torque-controlled robots**
  - issue *current* commands $i = i_c$ (with $\tau_c = K_i i_c$) to drive the (electrical) motors, based on information on the dynamic models
  - often, a low-level (analog) current loop is present to enforce the execution of the desired command
  - may use a torque measure $\tau_J$ (by joint torque sensors) to do the same, in case of joint/transmission elasticity (with $\tau_J = K(\theta - q)$)
  - best suited for high dynamic performance and ‘transparent’ control of interaction forces

- **position/motion-controlled robots**
  - issue *kinematic* commands: velocity $\dot{q} = \dot{q}_c$, acceleration $\ddot{q} = \ddot{q}_c$, or their integrated/micro-interpolated version $q = q_c$
  - references for a low-level direct loop at high frequency ($T_c \approx 400 \mu s$)
  - both modes can be present also on the same robotic system
HRI in industrial settings

non-collaborative robots: safety fences are required to prevent harming human operators

collaborative robots: allow human workers to stand in their proximity and work together on the same task

Main robot safety standards
ISO 10218-1/2:2011

ISO 13849-1 IEC 62061

Type A standard
Basic safety standards

ISO 12100 IEC 61508

Type B standard
Generic safety standards
B1 for specific safety aspects
B2 for safeguard

ISO 13850 ISO 13851

Type C standard
Machine safety standards
(product standard)

ISO 10218-1 ISO 10218-2

ANSI/RIA R15.06

CAN/CSA-Z434

ISO TS 15066

Robotics 2 25
Human-Robot Interaction taxonomy

- **cognitive** (cHRI) vs. **physical** (pHRI) Human-Robot Interaction
- cHRI models of humans, of robots, and of the interaction itself
  - dialog-based, intention- and activity-based, simulation-theoretic models

Human-Robot Interaction taxonomy

- pHRI planned and controlled robot behaviors: 3-layer architecture

Safety
- lightweight mechanical design
- compliance at robot joints
- collision detection and safe reaction

Coexistence
- robot and human sharing
- the same workspace
- collision avoidance
- no need of physical contact

Collaboration
- contactless, e.g., gestures or voice commands
- with intentional contact and coordinated exchange of forces

A. De Luca, F. Flacco: *IEEE BioRob Conference, 2012*
the different possible levels of pHRI are represented also within ISO safety standards (from safe coexistence to safe collaboration)

V. Villani et al.: *Mechatronics*, 2018

**Human-Robot Collaboration**

- LEVEL 1 - Safety-rated monitored stop
- LEVEL 2 - Hand guiding
- LEVEL 3 - Speed and separation monitoring
- LEVEL 4 - Power and force limiting

*Manual Positioning*

Video
## Panoramic view of control laws

reprise for HRI

<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 scheme)</td>
</tr>
<tr>
<td></td>
<td>trajectory tracking</td>
<td>feedback linearization, inverse dynamics + PD, passivity-based control, robust/adaptive control</td>
<td>feedback linearization</td>
<td>HRI control</td>
</tr>
<tr>
<td>contact motion (with force exchange)</td>
<td>-</td>
<td>-</td>
<td>impedance control (with variants), admittance control (kinematic scheme)</td>
<td>hybrid force-velocity control</td>
</tr>
</tbody>
</table>

### Notes:
- **PD**: Proportional-Derivative control
- **PID**: Proportional-Integral-Derivative control
- **PD with gravity compensation**: PD control with additional compensation for gravity
- **Visual servoing**: Control based on visual feedback
- **Gravity compensation**: Compensation for gravity effects
- **Kinematic scheme**: Control approach based on kinematic modeling
- **Impedance control**: Control that regulates the interaction forces between the robot and the environment
- **Admittance control**: Control that regulates the motion of the robot in response to external forces