Robotics 2

Introduction to Control

Prof. Alessandro De Luca

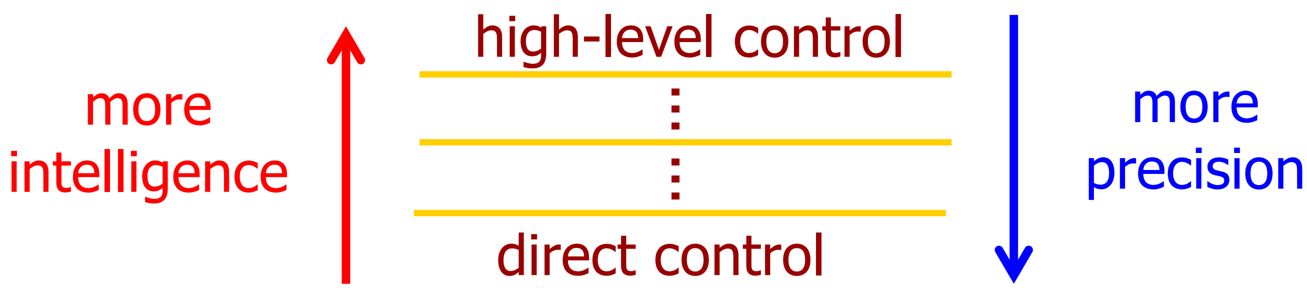
Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

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

$\Rightarrow$ control system unit has a hierarchical internal structure



- different time scales in the various control levels: lowest ≤ 1 ms, higher levels up to seconds
- different but cooperating models, objectives, methods

Evaluation of control performance

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

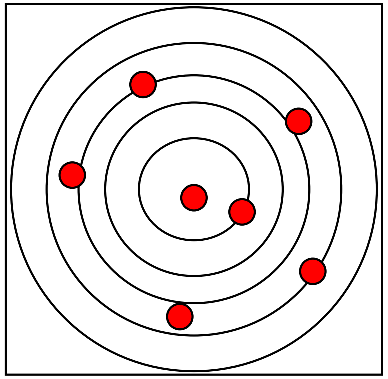
  ⇒ improvements also thanks to models (software!)
- robustness in perturbed/uncertain conditions
  - adaptation to changing environments
  - high repeatability despite disturbances, changes of parameters, uncertainties, modeling errors

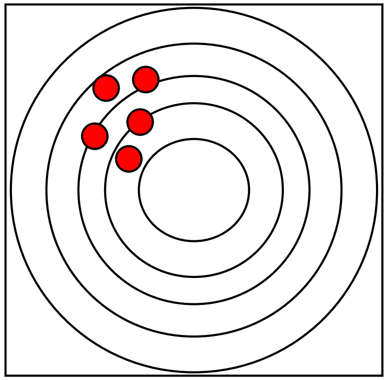
  ⇒ can be improved by a generalized use of feedback, using more sensor information

  ⇒ learn through repeated robot trials/human experience

Static positioning
accuracy and repeatability

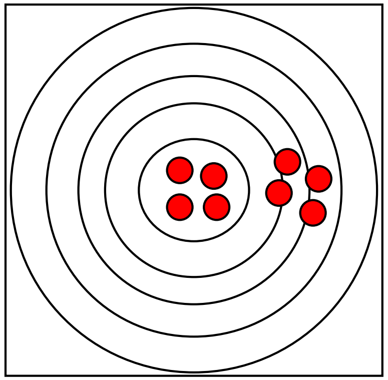
poor accuracy
poor repeatability

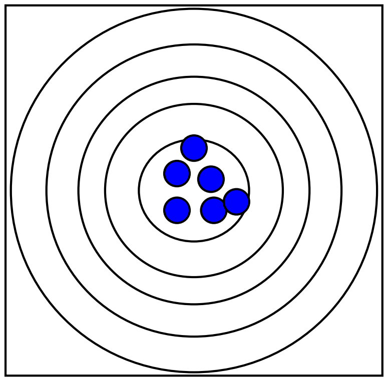




poor accuracy
good repeatability

good accuracy
poor repeatability

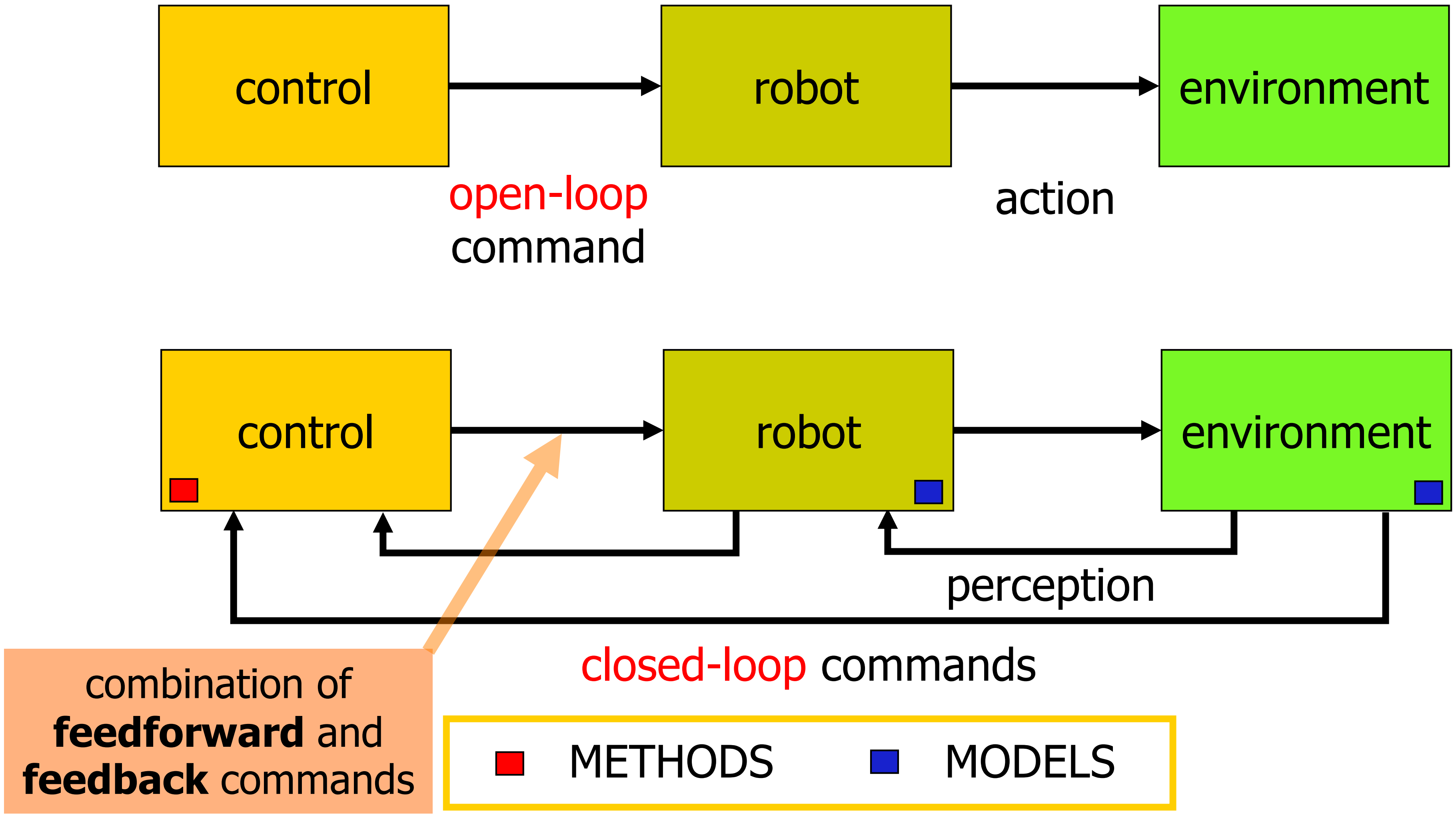




good accuracy
good repeatability

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

Basic control schemes



Control schemes and uncertainty

- feedback control
  - insensitivity to mild disturbances, small variations of parameters, and different initial conditions
- robust control
  - tolerates relatively large uncertainties of known range
- adaptive control
  - improves performance online, adapting the control law to unknown range of uncertainties and/or large (but slow) parameter variations
- AI-based (intelligent) control
  - performance improved based on trials/experience: LEARNING
  - autonomous/supervised search of internal structure for capturing system behavior: SELF-ORGANIZING
  - Data-driven models: (Deep) Neural Networks, Gaussian Processes

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 programs and sensing devices for hard real-time operation
  - need of some expertise for programming and handling of exceptions
  - ⇒ introducing easy/more intuitive user (multi-modal) interfaces
- at the higher level
  - open-loop task command generation
  - ⇒ exteroceptive sensory feedback absent or very loose, with low capability of autonomous reasoning
- 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 use of the extra degrees of freedom

Limits in control of industrial robots - 2

- at the lower (direct) level
  - reduced execution speed ("control bandwidth")
    - ⇒ typically, heavy mechanical structures ■
  - 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

video

without modeling and explicit control of 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 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, ...)
  - …

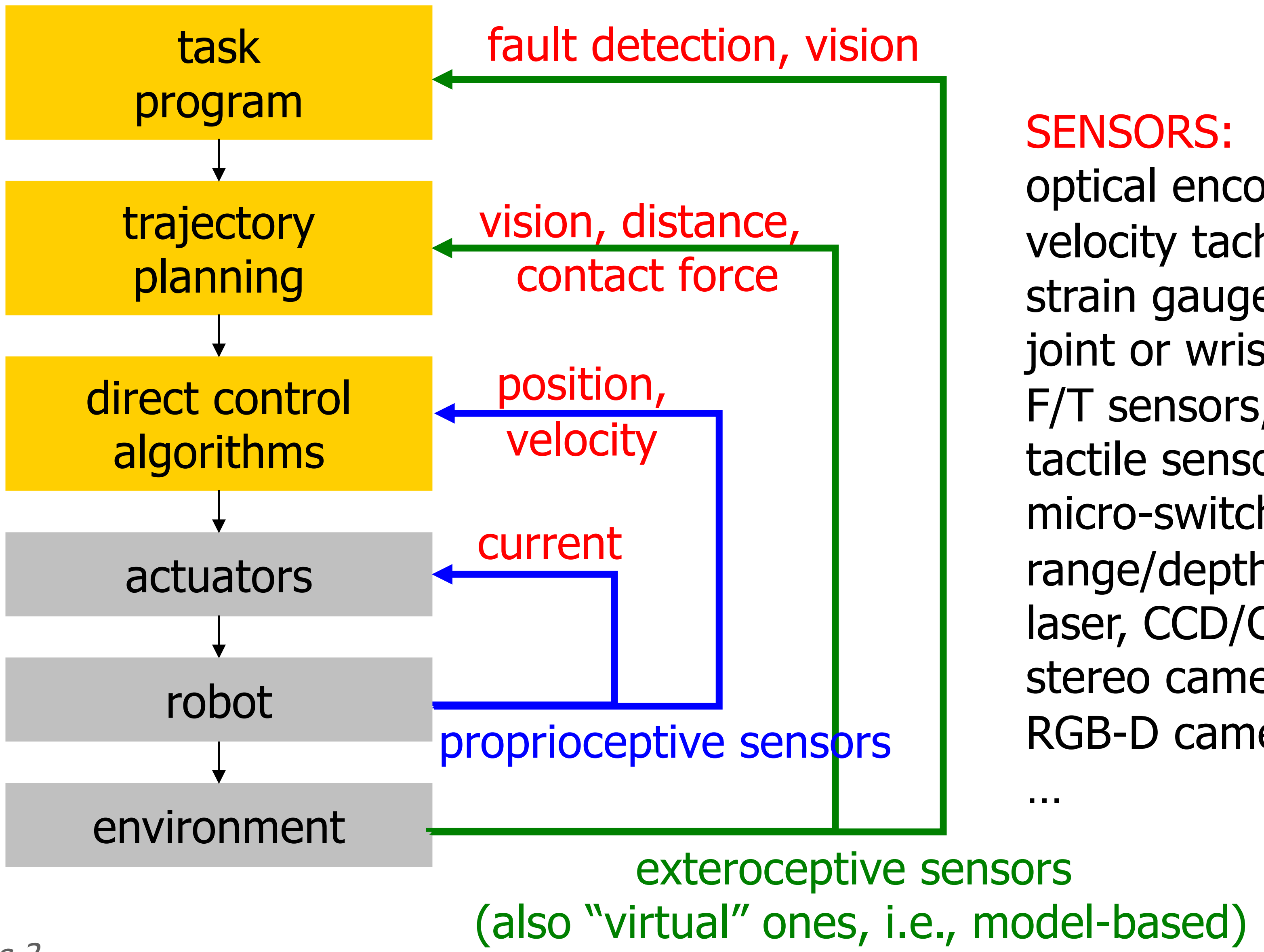
Example of visual-based control

- human-obstacle collision avoidance

video

- 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



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

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

dedicated programming languages
Task-
Object-
Robot-
Oriented

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

- task program: modeling of tasks (with AI reasoning)
- trajectory planning: geometric and kinematic models; coordinate transformations
- direct control algorithms: dynamic-based methods; (nonlinear) torque control
- actuators, robot: (electrical and mechanical) dynamic models
- environment: structured and unstructured world modeling (and acquisition)

Industrial robot programming languages

- ABB Rapid
- COMAU PDL2
- FANUC Karel
- KUKA KRL
- MITSUBISHI MELFA
- UNIVERSAL ROBOTS RoboDK
- ...

Robot control/research software

(last updated in April 2025)

- 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 acting 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 (was V-REP; 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 System Toolbox (license for Sapienza)

- tools/algorithms for simulation of kinematics, dynamics, trajectory planning, control of serial manipulators, mobile robots and humanoids
- library of robots, scene and map creation, Gazebo interface …

QUT Robot Academy petercorke.com

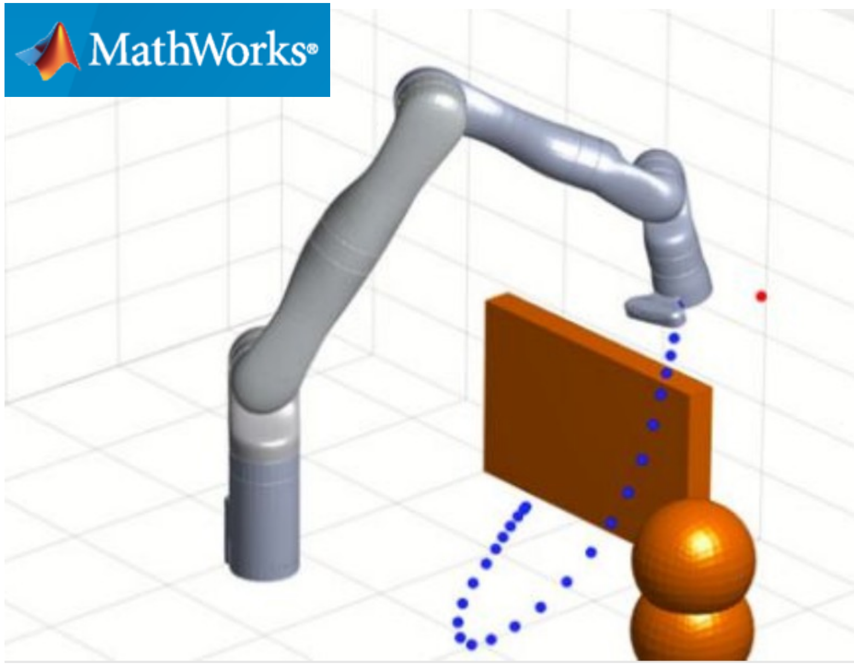
- free software, including Robotics Toolbox (release 10) and Machine Vision Toolbox (release 4) for MATLAB (open source on github) → now in Python

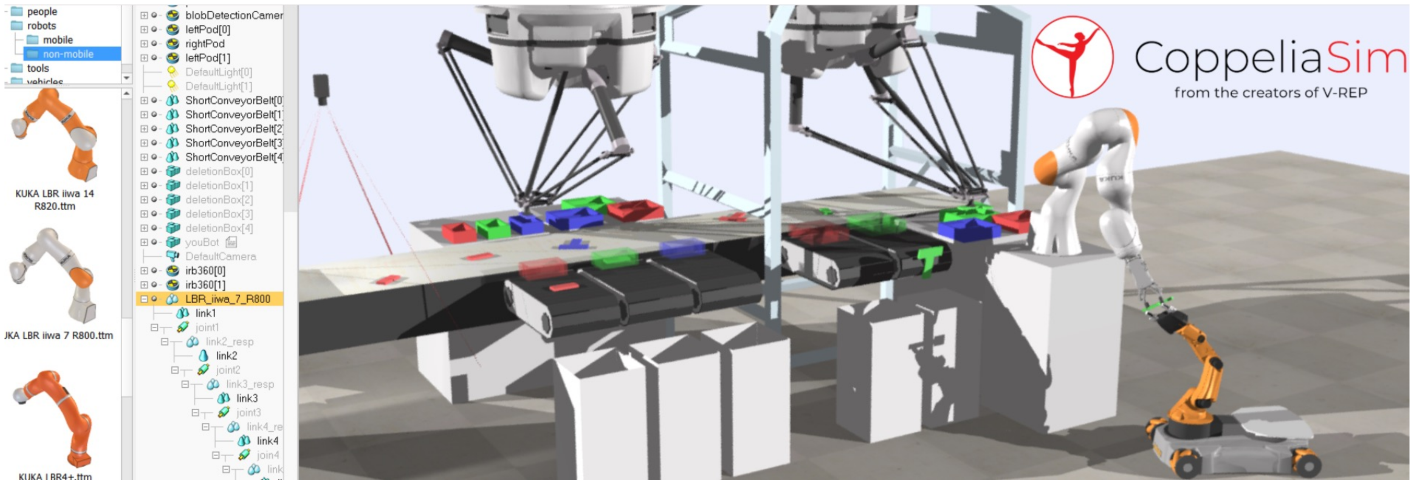
ROS (Robot Operating System) ros.org

- middleware with hardware abstraction, device drivers, libraries, visualizers, message-passing, package management (ROS 2: Linux, Windows, MacOS)
- "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)

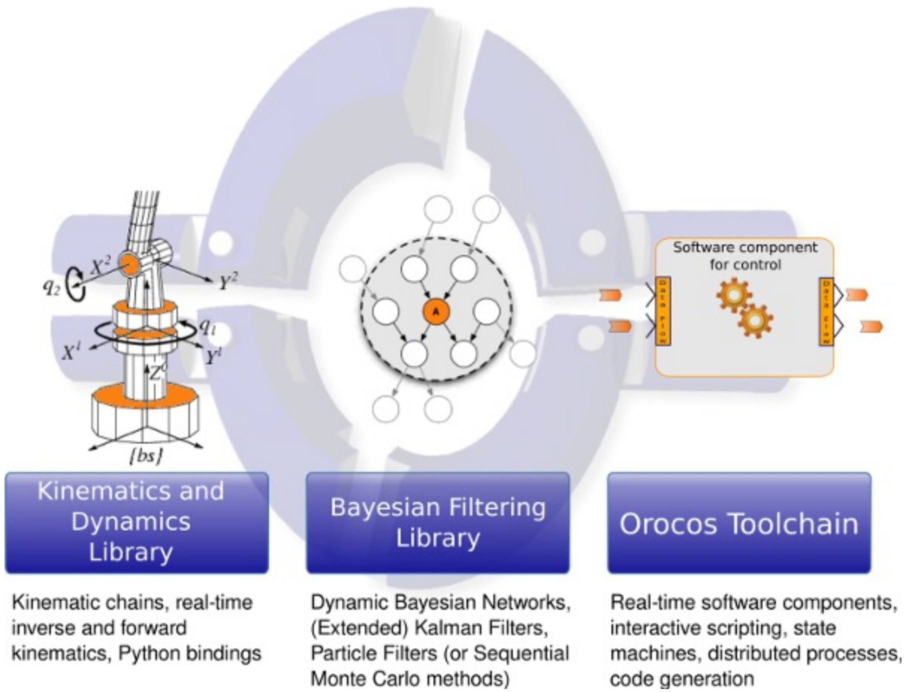
Robot control/research software



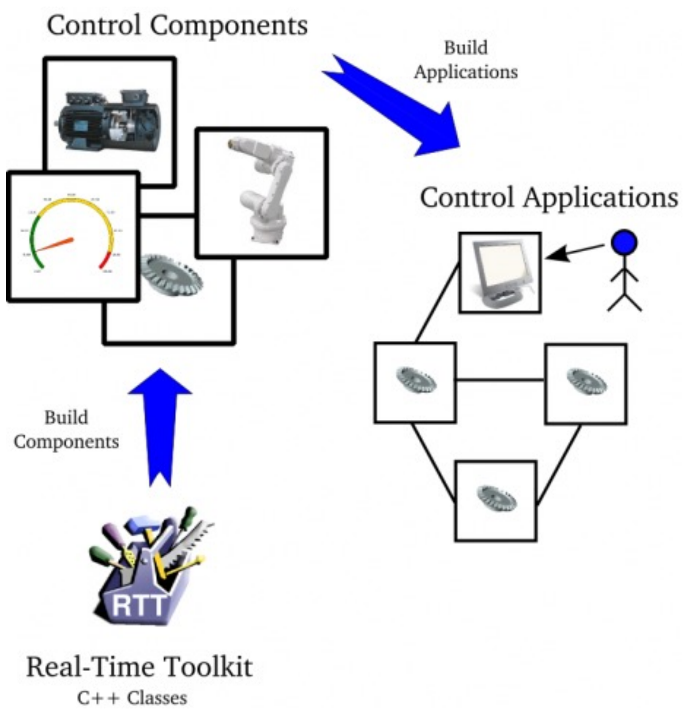


OROCOS control software

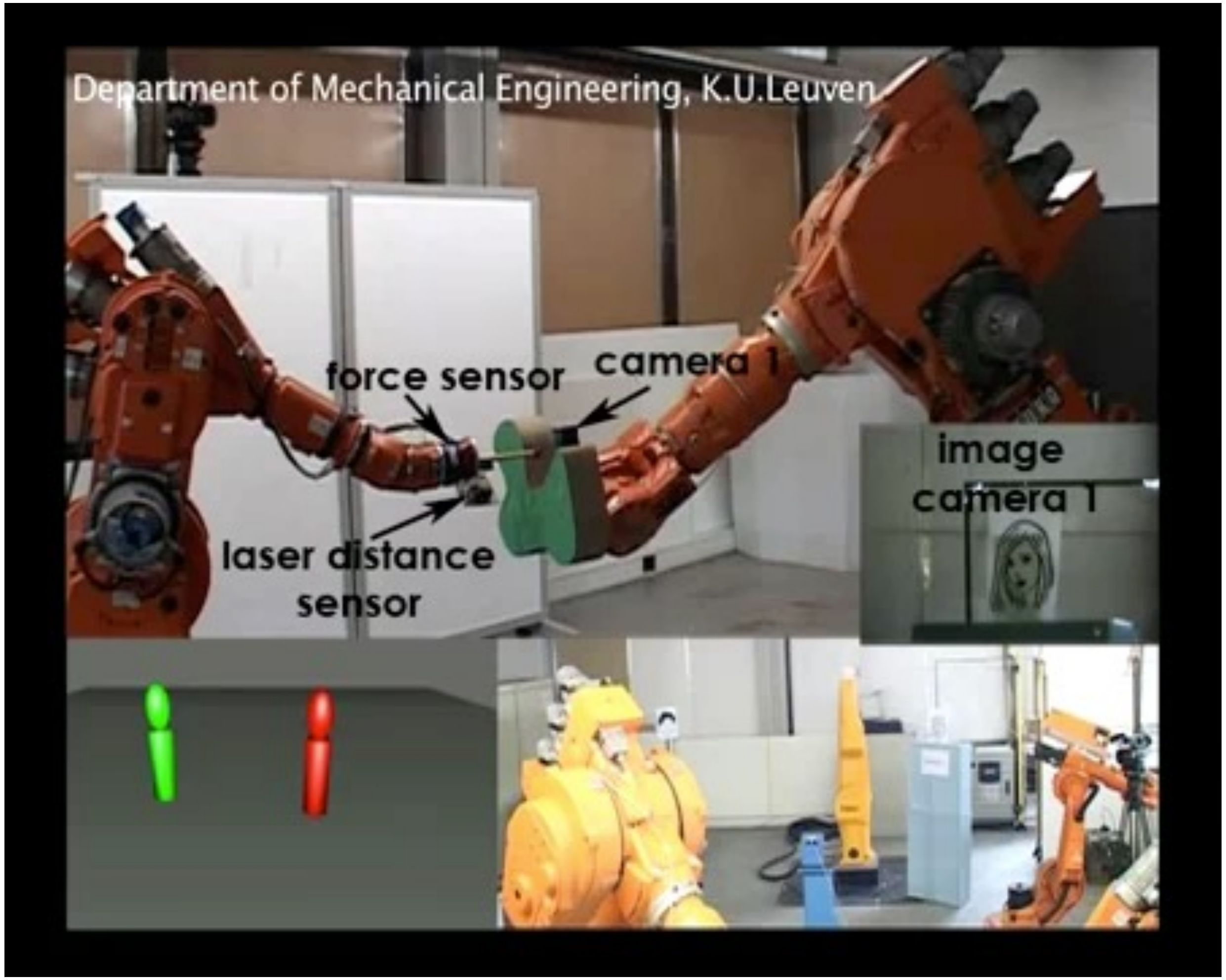
- OROCOS (Open RObot COntrol Software) orocos.org
  - open-source, portable C++ libraries for robot control
  - Real-Time Toolkit (for Linux, MacOS X, Windows Visual Studio)
  - supports CORBA for distributed network computing and ROS interface
  - (user-defined) application libraries



⇒ github



Example application using OROCOS



video

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

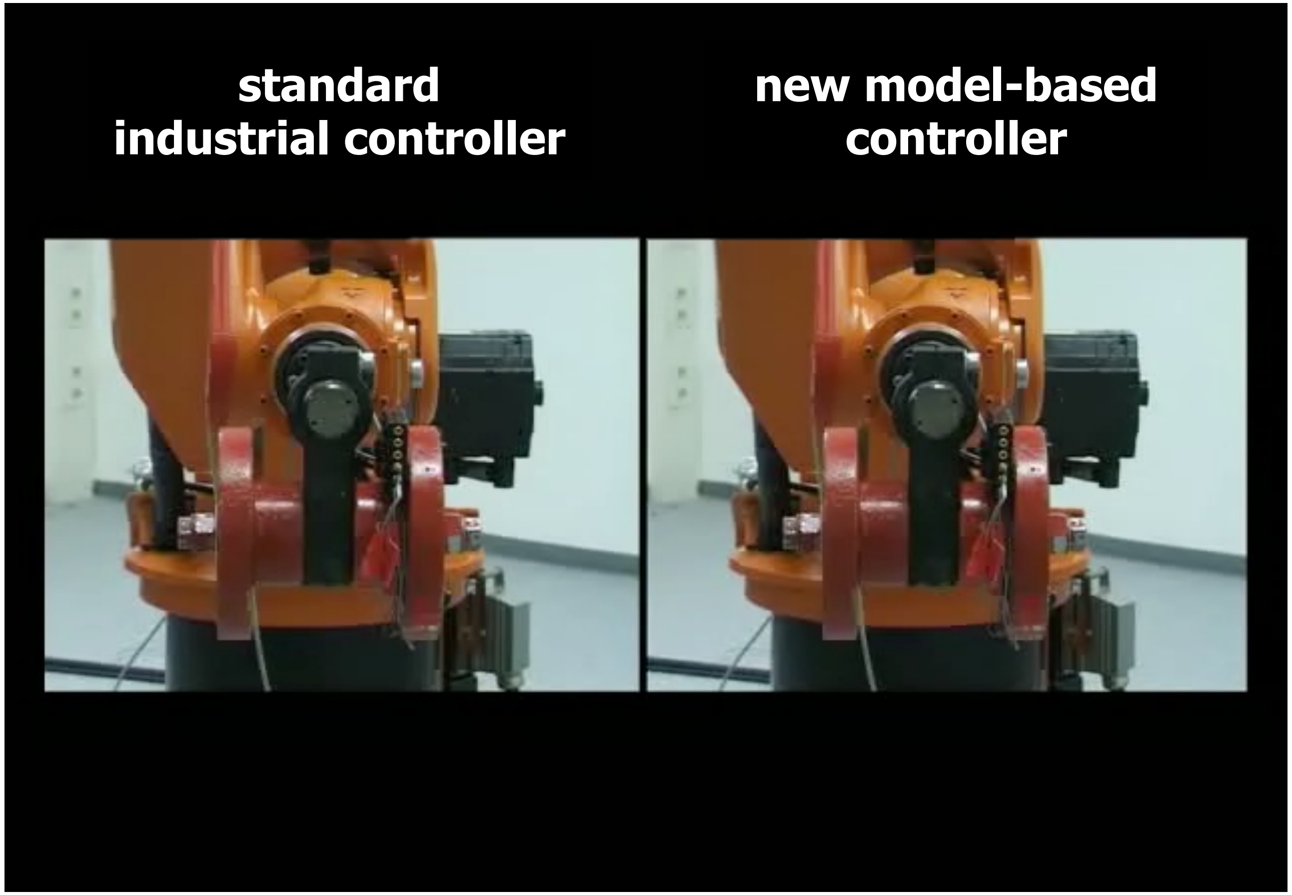
Summarizing …

- 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)

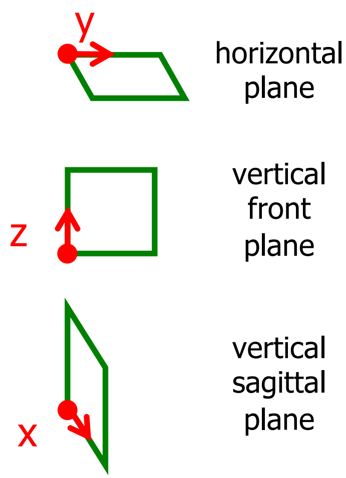
Benefits of model-based control

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



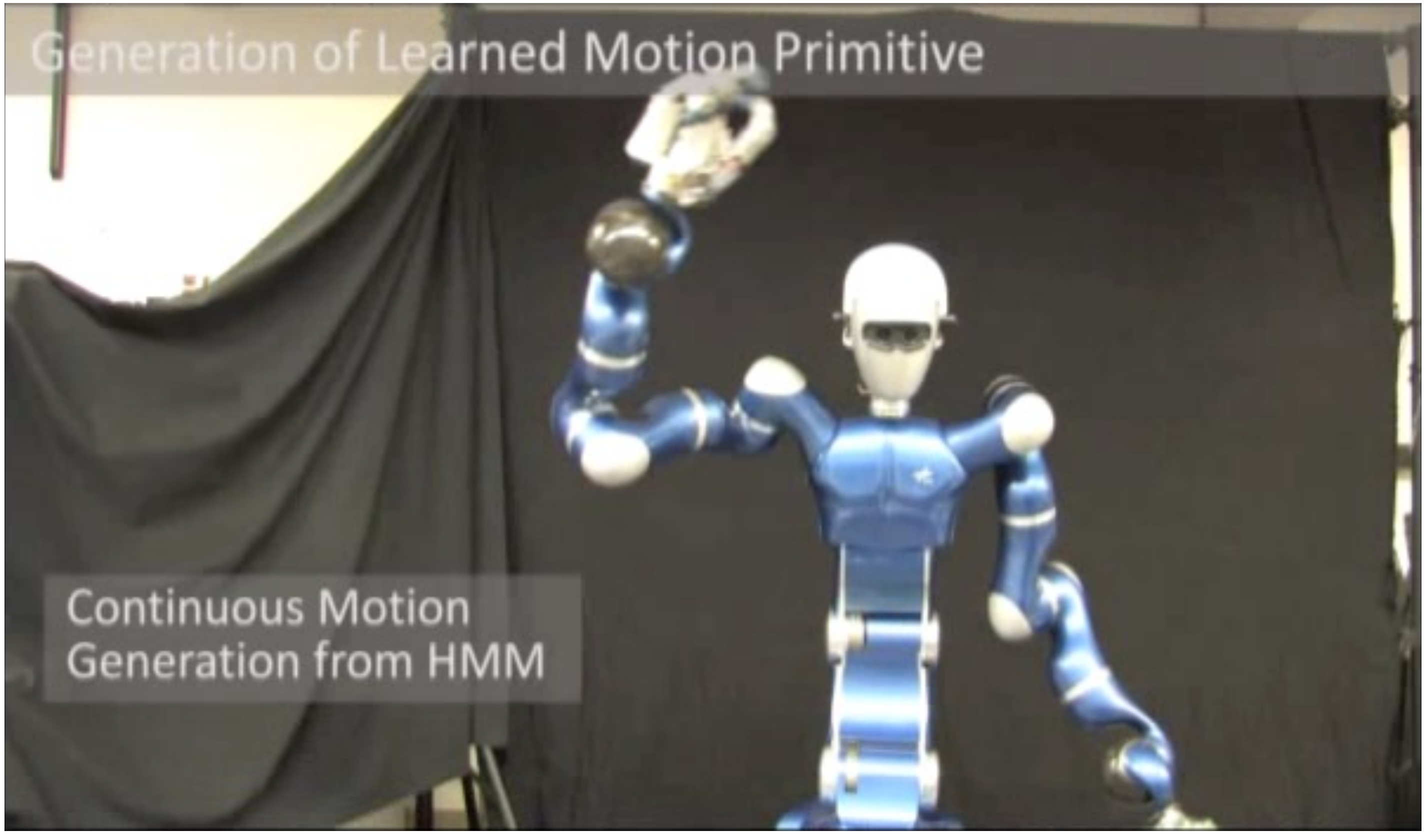
video

three squares in:



Robot learning by imitation

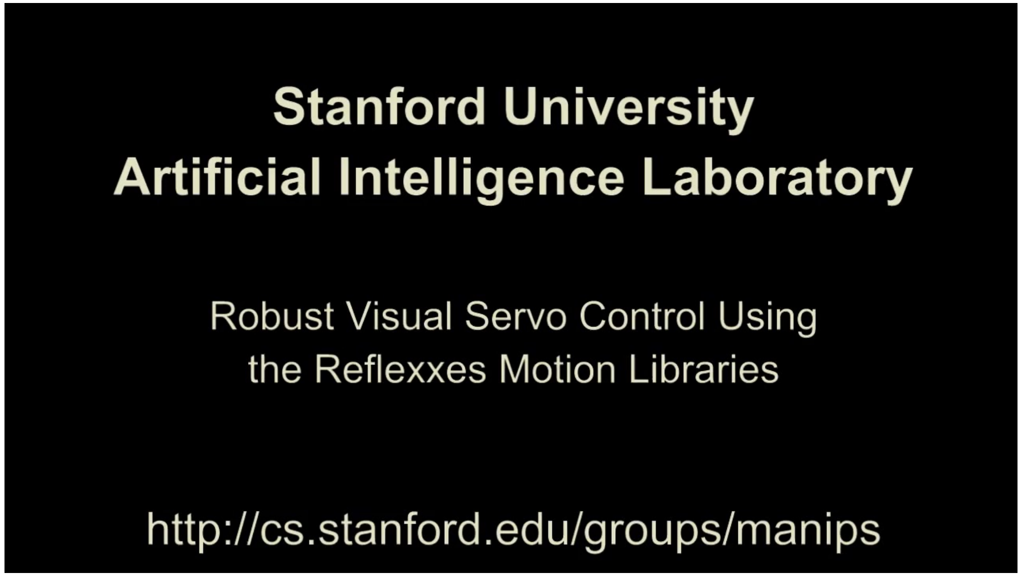
- learning from human motion primitives (imitation)
- motion refinement by kinesthetic teaching (with impedance control)



video

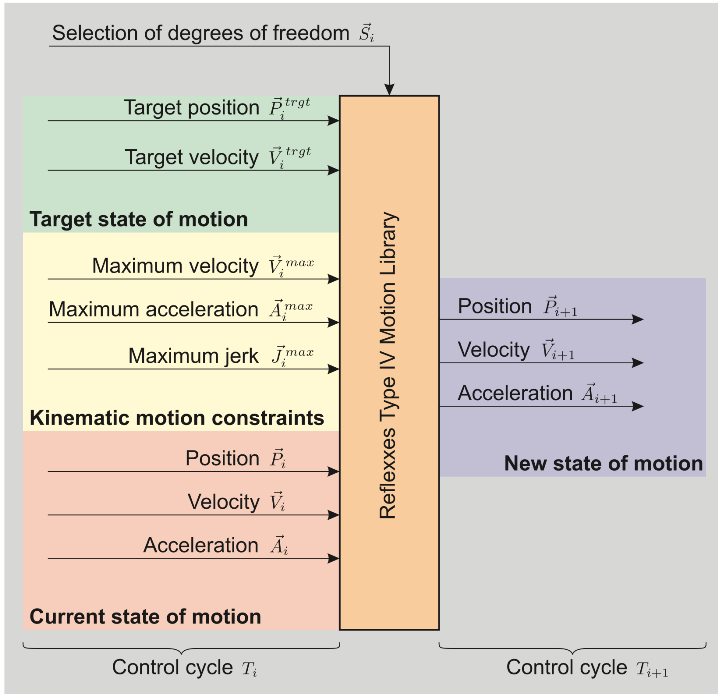
@TUM, Munich (D. Lee, C. Ott), for the EU SAPHARI project

Using visual or depth sensor feedback



video

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



- collision avoidance schemes (here, redundancy w.r.t. an E-E task)

video

Panoramic view of control laws

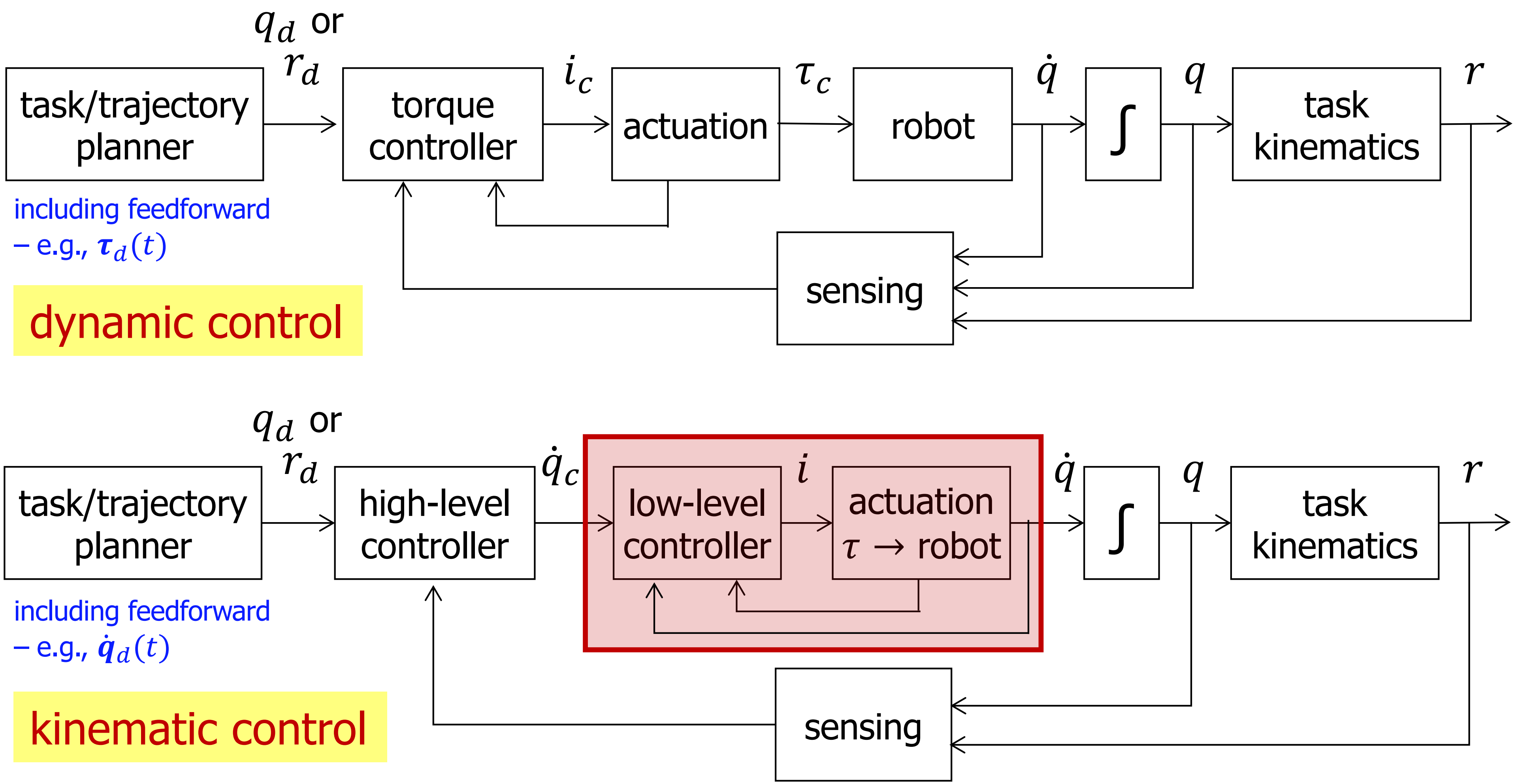
- problems & methods for robot manipulators that will be considered (control command is always a joint torque, if not else specified)

type of task \ definition of error		joint space	Cartesian space	task space
free motion	regulation	PD, PID, gravity compensation, iterative learning	PD with gravity compensation	visual servoing (kinematic scheme)
	trajectory tracking	feedback linearization, inverse dynamics + PD, passivity-based control, robust/adaptive control	feedback linearization	
contact motion (with force exchange)		-	impedance control (with variants), admittance control (kinematic scheme)	hybrid force-velocity control

Dynamic or kinematic control laws

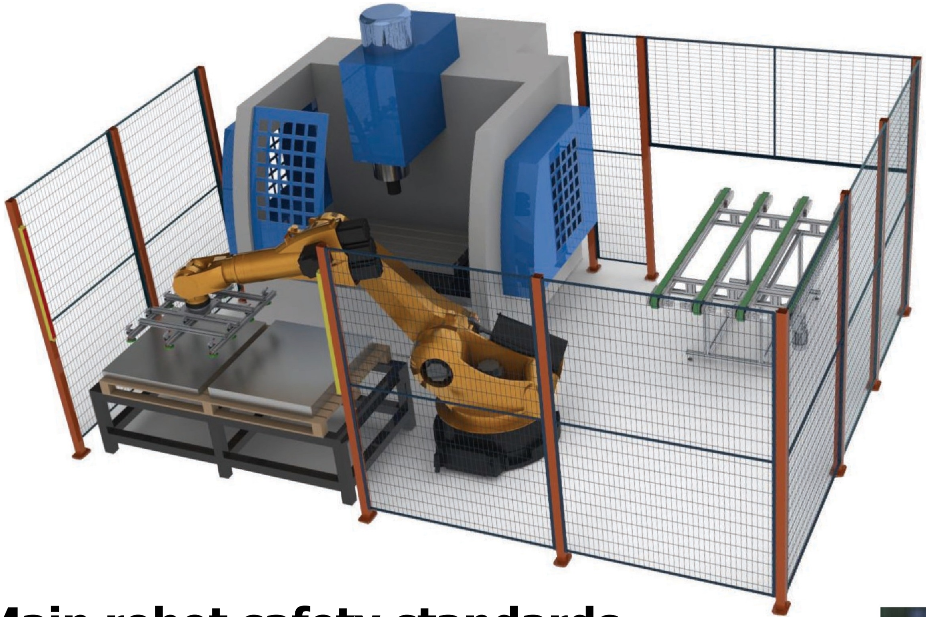
- torque-controlled robots
  - issue current commands $i = i_c$ (with $\tau_c = K_i\, i_c$) to drive the (electrical) motors, based on information on the dynamic model
  - often, a low-level (analog) current loop is present to enforce the execution of the desired command
  - may use a torque measure τ_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 ($T_c \cong 400\ \mu s$!)

Torque- vs. position-controlled robots



- both modes may be present even in the same robotic system

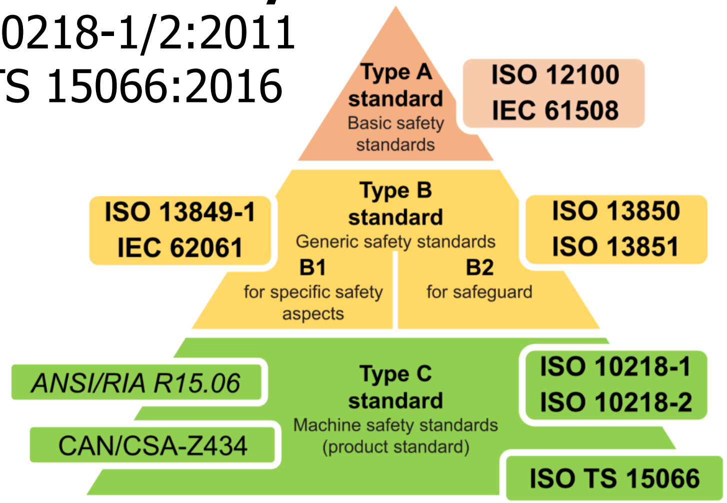
HRI in industrial settings



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

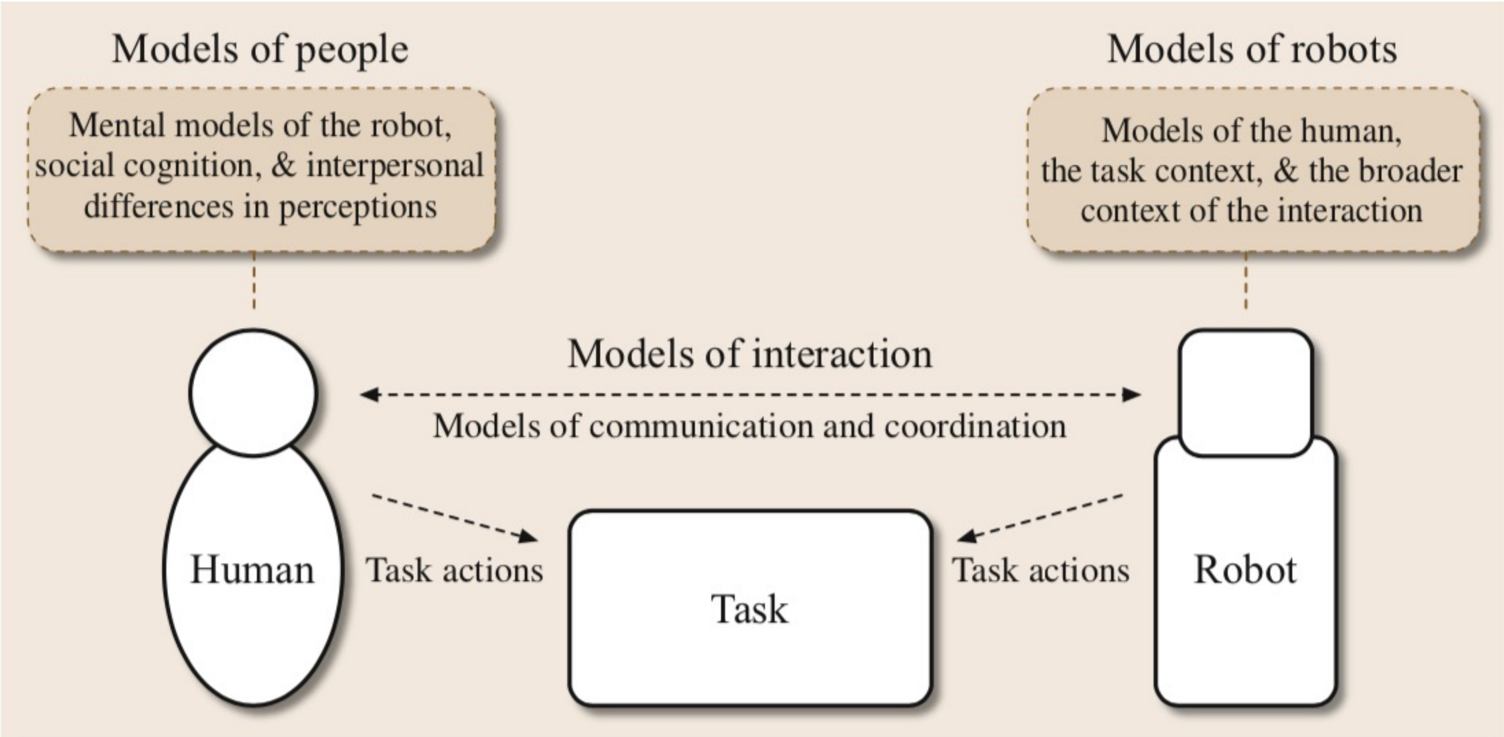
collaborative robots:
allow human workers to
stand in their proximity and
work together on the same task

Main robot safety standards
ISO 10218-1/2:2011
ISO/TS 15066:2016



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



B. Mutlu, N. Roy, S. Sabanovic: *Ch. 71, Springer Handbook of Robotics,* 2016

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

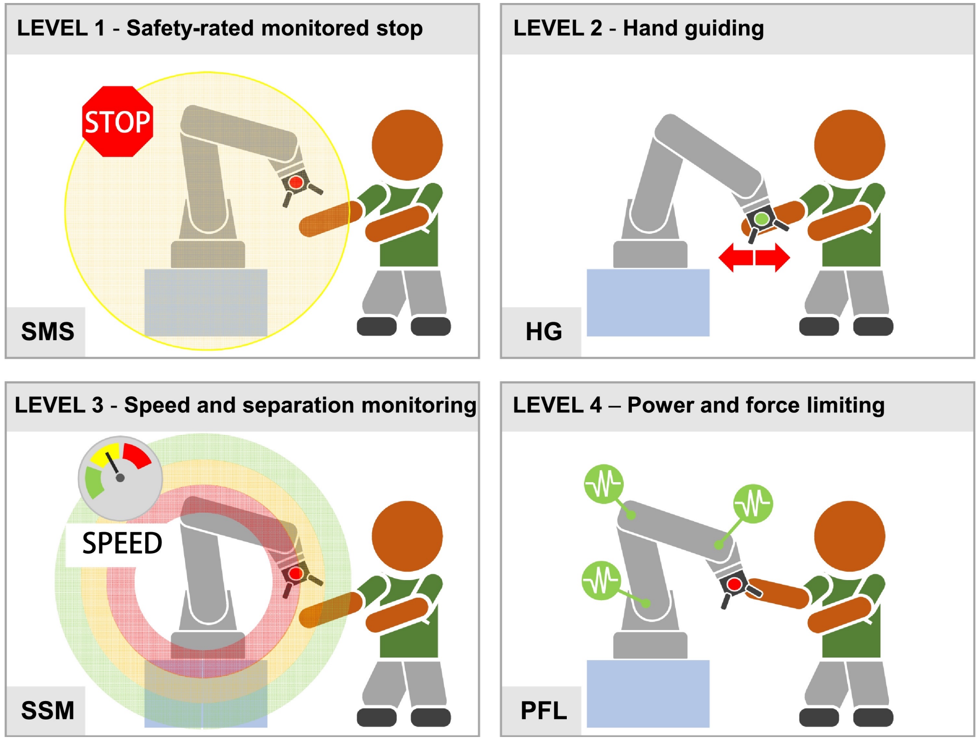
contactless, e.g., gestures
or voice commands

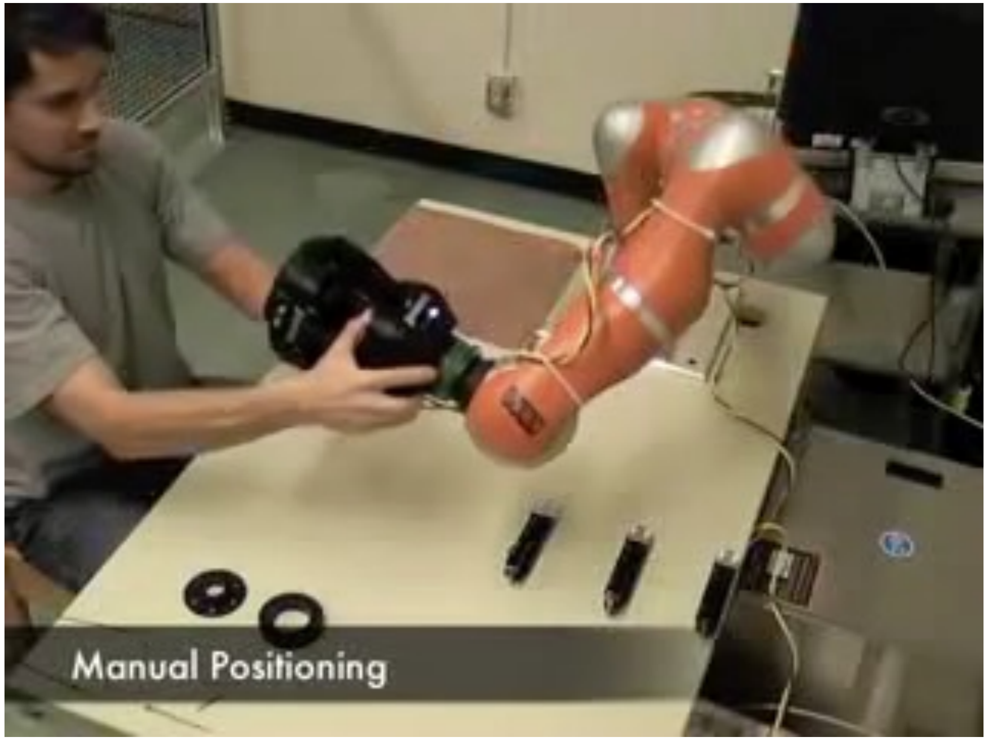
with intentional contact and
coordinated exchange of forces

A. De Luca, F. Flacco: *IEEE BioRob Conference*, 2012

Human-Robot Collaboration

- 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

video

for more info & material on safe pHRI: http://www.diag.uniroma1.it/deluca/pHRI.php