Robotics 2

Introduction to Control

Prof. Alessandro De Luca

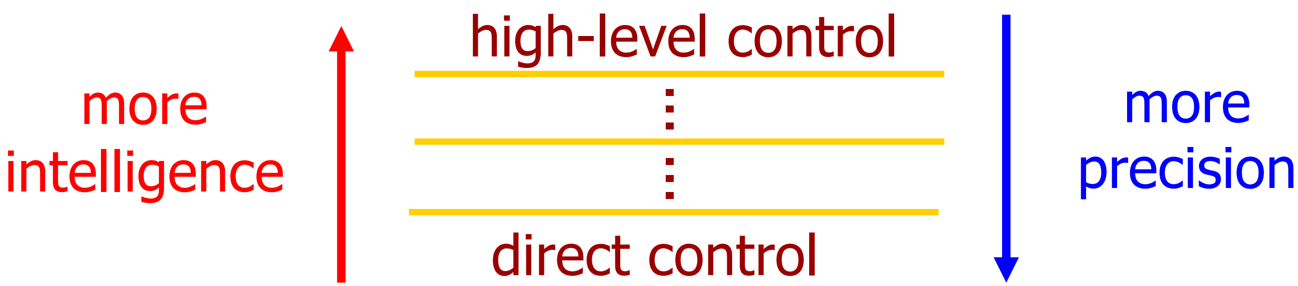
Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti

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

⇒ 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

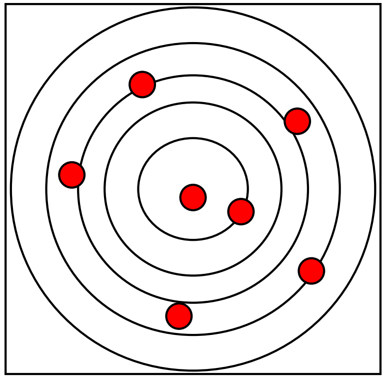
  ⇒ improvements also thanks to models (software!)
- robustness in perturbed/uncertain conditions
  - adaptation to changing environments
  - high repeatability despite disturbances, changes of parameters, uncertainties, modeling errors

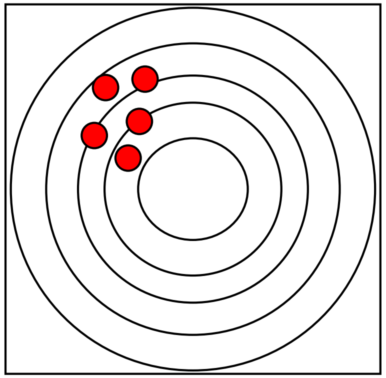
  ⇒ can be improved by a generalized use of feedback, using more sensor information

  ⇒ learn through repeated robot trials/human experience

Static positioning
accuracy and repeatability

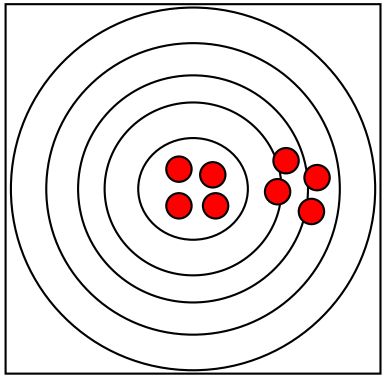
poor accuracy
poor repeatability

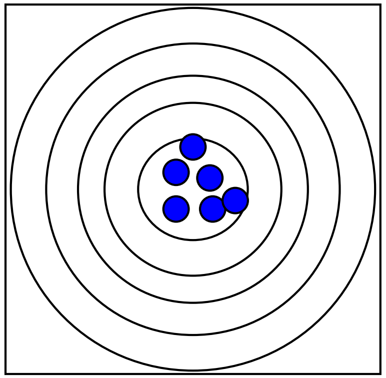




poor accuracy
good repeatability

good accuracy
poor repeatability

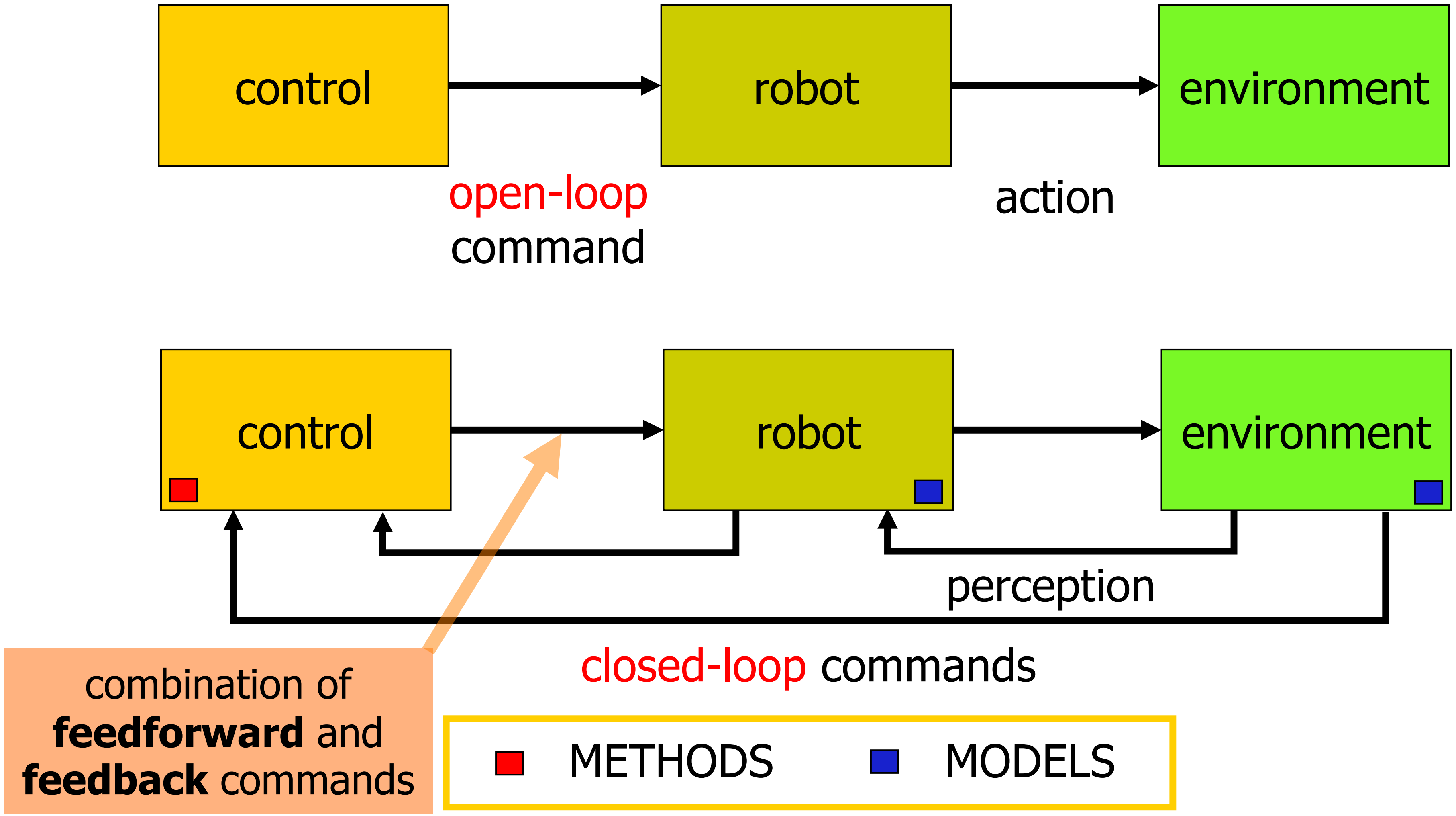




good accuracy
good repeatability

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

Basic control schemes



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 is a must
- at the higher level
  - open loop 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
  - 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

video

without modeling and controlling 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 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, ...)
  - ...

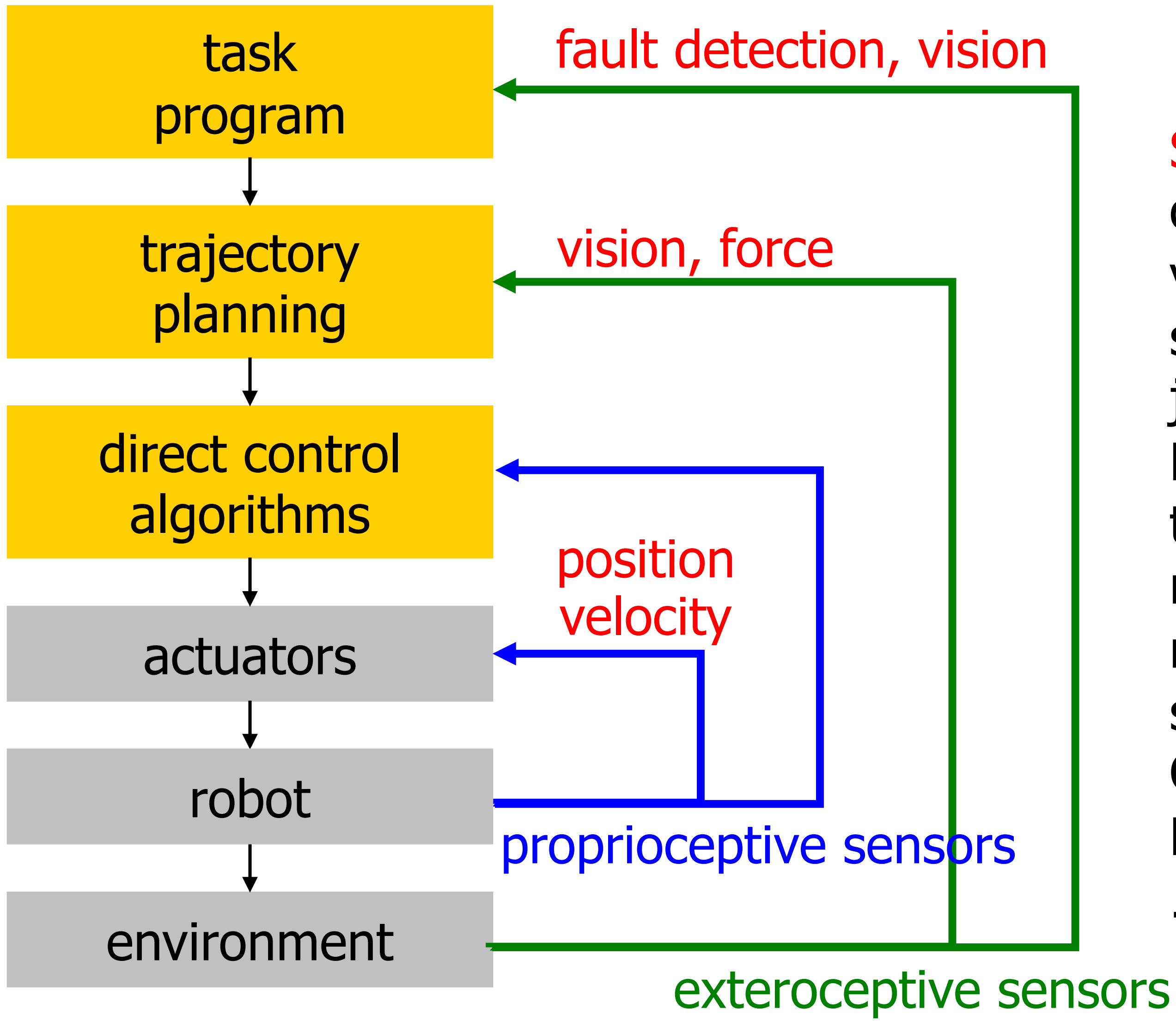
Example of visual-based control

- 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



SENSORS:
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
...

Functional structure of a control unit

- 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°

■ use TEACH BOX in industrial robots

Functional structure of a control unit

task program	modeling of tasks
↓	
trajectory planning	geometric and kinematic models coordinate transformations
↓	
direct control algorithms	nonlinear methods dynamic control
↓	
actuators	(electrical and mechanical) dynamic models
↓	
robot	
↓	
environment	structured and unstructured world modeling

Robot control/research software
(last updated in April 2019)

- 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

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, and trajectory generation for serial-link manipulators ⇒ 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)

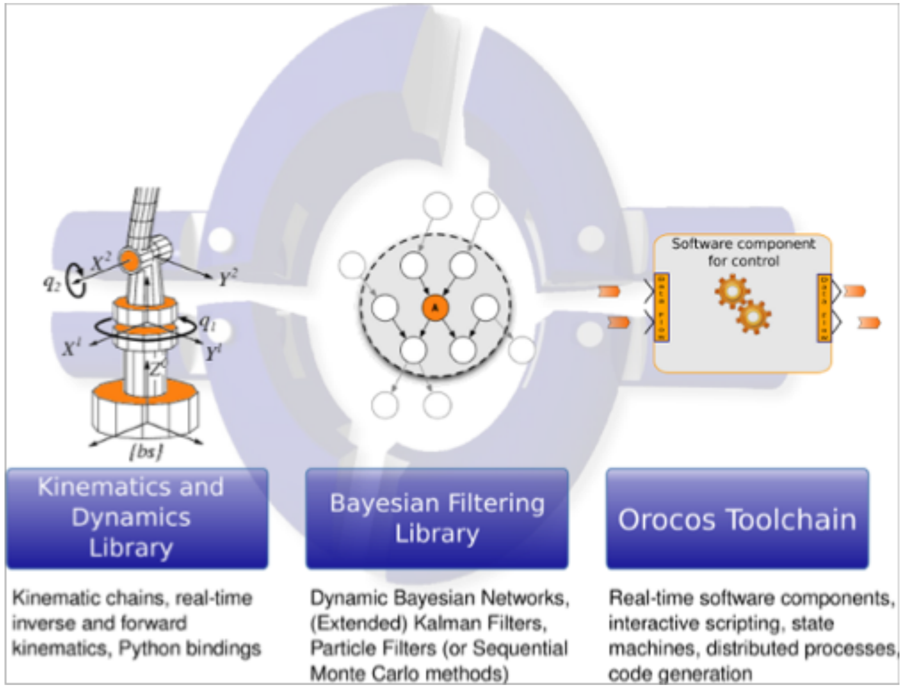
PyRobotics (Python API) pypi.org/project/pyRobotics (v1.8 in 2015)

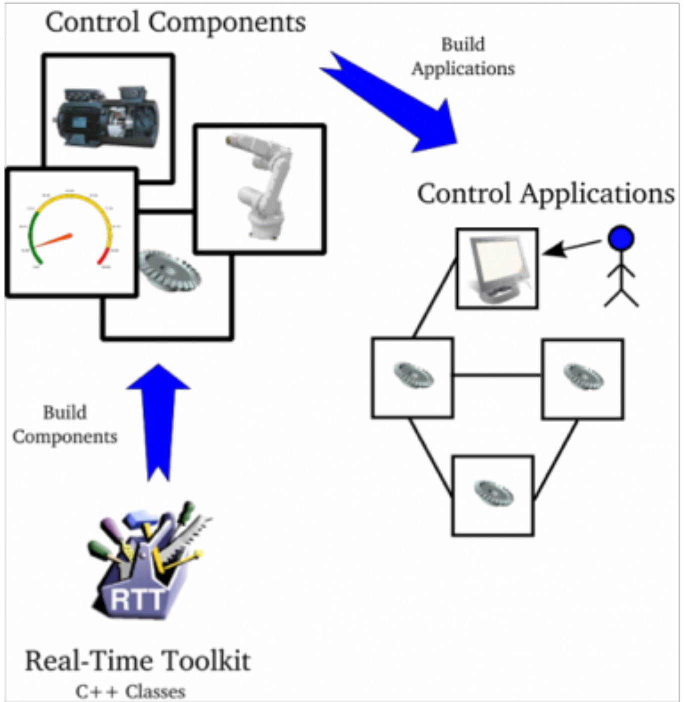
OpenRDK openrdk.sourceforge.net ⇒ developed @DIAG, dismissed

- "agents": modular processes dynamically activated, with blackboard-type communication (repository)

OROCOS control software

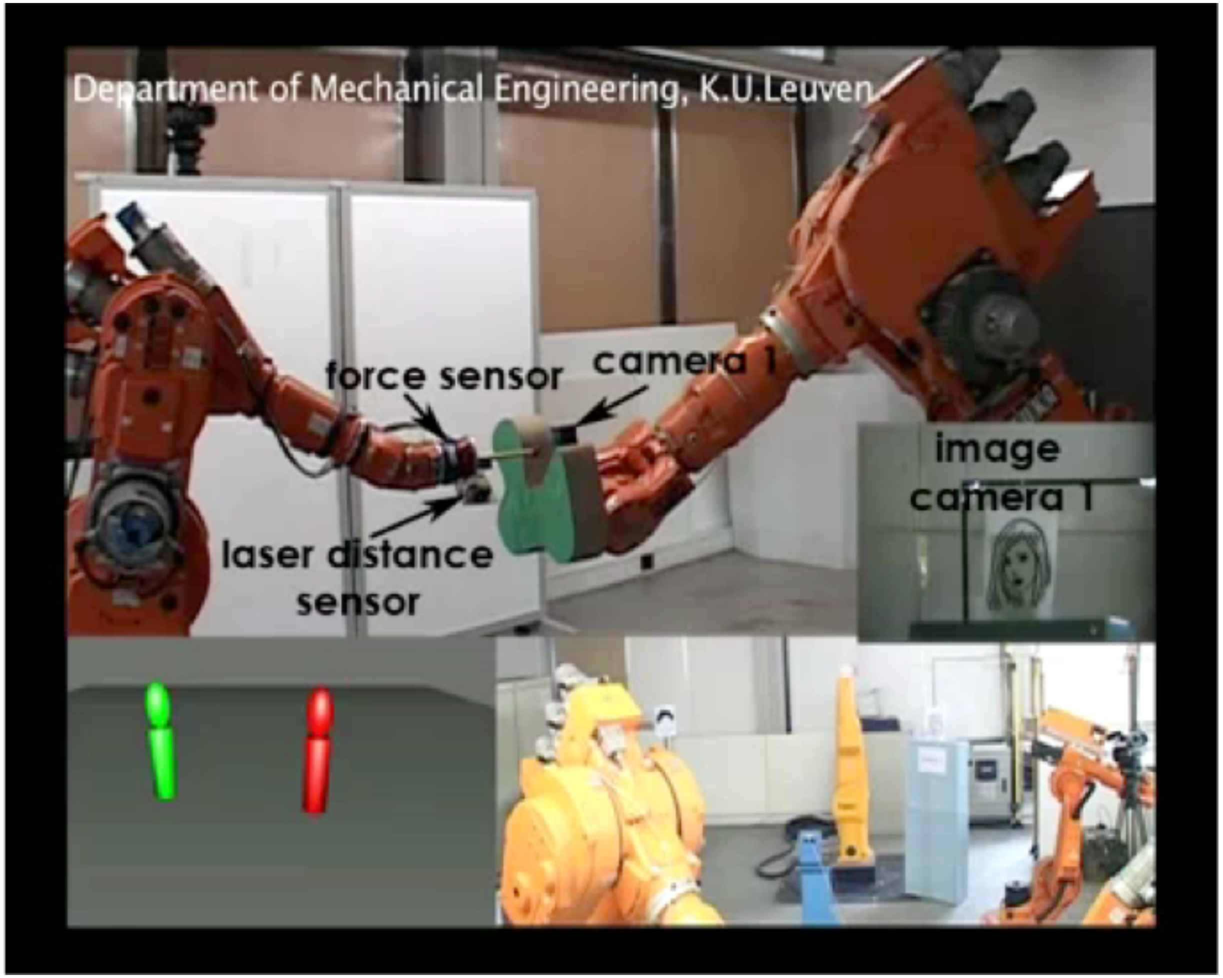
- OROCOS (Open RObot COntrol Software) orocos.org
  - open-source, portable C++ libraries for robot control
  - Real-Time Toolkit (for Linux, MacOS X, Windows Visual Studio)
  - supports CORBA for distributed network computing and ROS interface
  - (user-defined) application libraries





⇒ github

Example application using OROCOS



video

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

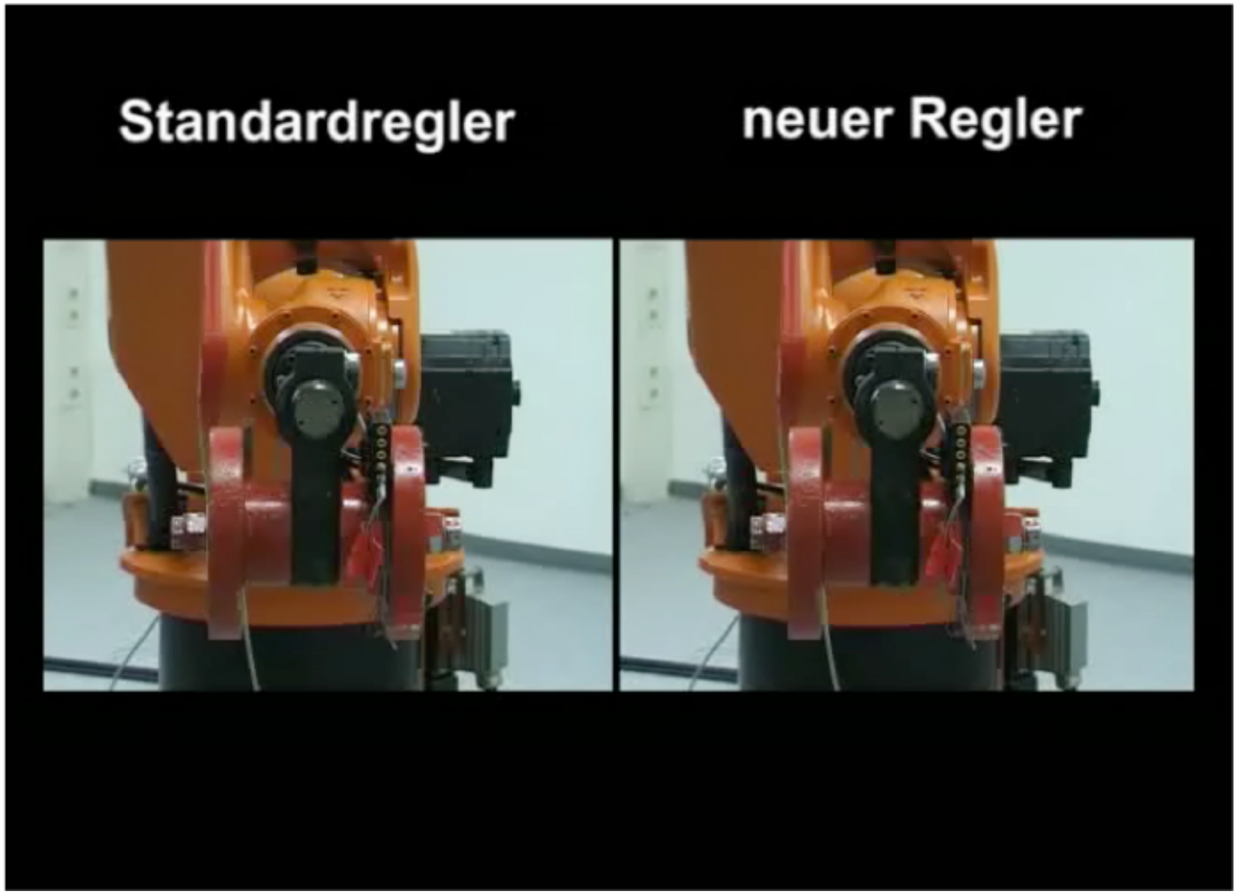
Summarizing …

- 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

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

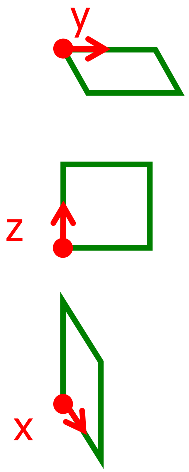
Benefits of model-based control

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



video

three squares in:



horizontal plane

vertical front plane

vertical sagittal plane

Robot learning by imitation

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

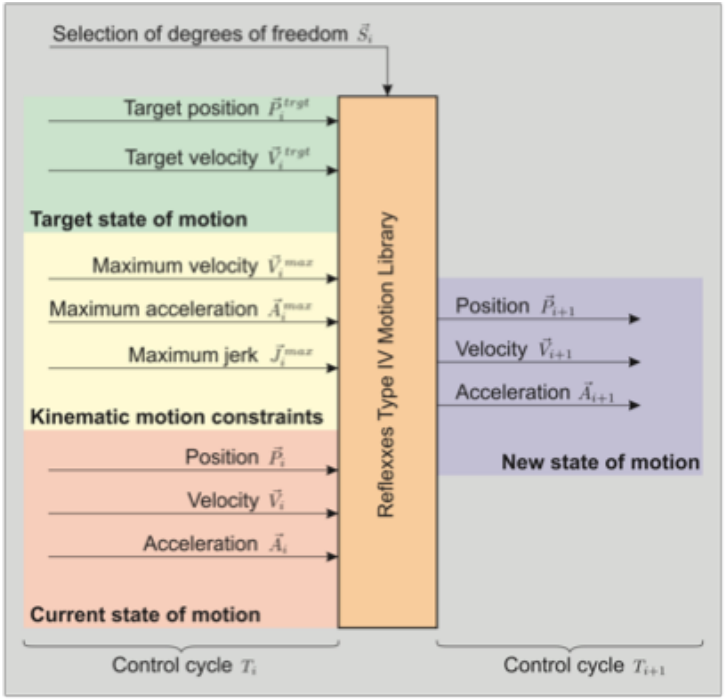
video

@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 and methods for robot manipulators that will be considered

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 approach)
	trajectory tracking	feedback linearization, inverse dynamics + PD, passivity-based control, robust/adaptive control	feedback linearization	
contact motion		-	impedance control (with variants)	hybrid force-velocity control