



Robotics 1

Introduction

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Robotics 1 – 2020-21

- First semester (11 weeks -- reduced)
 - Tuesday, October 6, 2020 – Friday, December 18, 2020
- Courses of study (with Robotics 1 mandatory or present as optional)
 - Master in Artificial Intelligence and Robotics (MARR)
 - Master in Control Engineering (MCER)
- 6 Credits
 - ~50 hours of lectures, exercises, and midterm test
 - 90 hours of individual study
- Classes (in presence: room B2, DIAG, Via Ariosto 25, online: Zoom)
 - Tuesday 8:00-10:00
 - Friday 8:00-11:00
 - Time zone CEST = UTC+2 until October 24, then CET = UTC+1



Organization and contacts

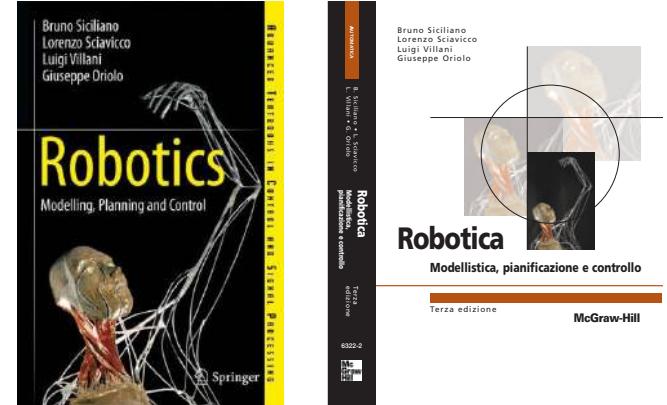
- Zoom for online
 - topic: Robotics 1 2020-21 (no waiting room if access via **uniroma1** email)
- YouTube
 - personal channel, with **some** recorded videos of the lectures
- Google group
 - join: robotics1_2020-21 (with your **uniroma1** email)
- Course website www.diag.uniroma1.it/deluca/rob1_en.php
- Email deluca@diag.uniroma1.it
- Office hours for students
 - every Tuesday 12:00-13:30 (check exceptions)
 - remote: Google Meet <https://meet.google.com/chp-fghs-fri>
 - in presence: room A-210, floor 2, left wing, DIAG, Via Ariosto 25



Course Materials

■ Textbook

- B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo:
Robotics: Modelling, Planning and Control,
3rd Edition, Springer, 2009
 - English, Italian, Chinese, or Greek editions



■ On course website

- pdf of lecture slides **ready** (but with **updates** during the course)
- all videos shown in lectures (in zipped folders by block of slides)
- written exams (most with solutions), papers, extra documents, ...
- **Video DIAG Channel** playlist [Robotics 1](#) with full course of **2014-15**
 - 30 (+1 index) videos in classroom ($\cong 41\text{h}$, > 73000 independent views)
- **DIAG Robotics Lab Channel** with more research videos
 - www.youtube.com/user/RoboticsLabSapienza



General information

- Prerequisites
 - self-contained course, without special prerequisites
 - elementary knowledge on kinematics, linear algebra, and automatic control is useful
- Aims
 - tools for kinematic analysis of chains of multiple rigid bodies, trajectory planning, and programming of motion tasks for robot manipulators in industrial and service environments
- Other strictly related courses
 - Robotics 2: II semester (of year I), 6 credits
 - Autonomous and Mobile Robotics: I semester (of year II), 6 credits
 - Elective in Robotics (12 credits, MARR) or Control Problems in Robotics (6 credits, MCER): year II, 4 or 2 modules of 3 credits



A robot manipulator

Illustrating typical features of an industrial robot

commercial video



KUKA KR 4 Agilus robot with 6 revolute joints



Programming robot motion

Teaching Cartesian poses and playing them back

video



KUKA LBR iiwa robot with 7 revolute joints



Programming robot motion

Executing nominal trajectories and “complying” with uncertainties

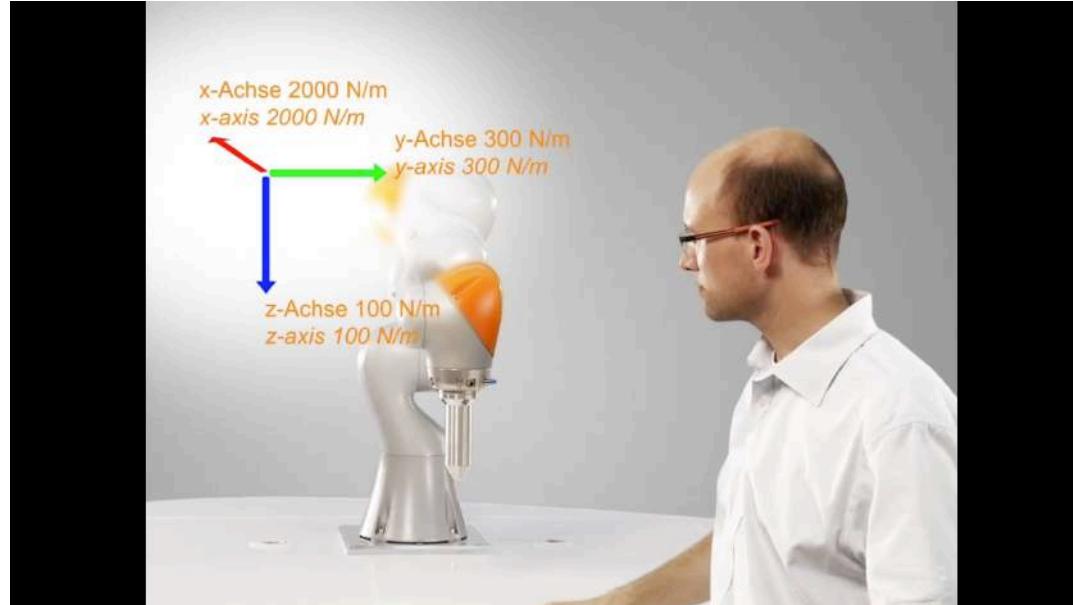
video





Programming robot compliance

Controlled reaction to applied forces/torques at robot end-effector

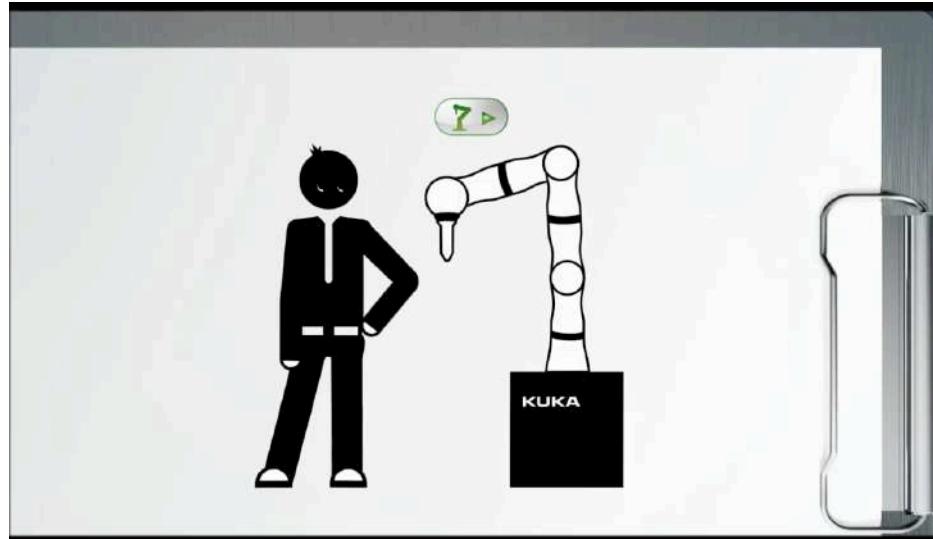


video



Programming robot motion

Teaching tasks by demonstration (kinesthetic learning)



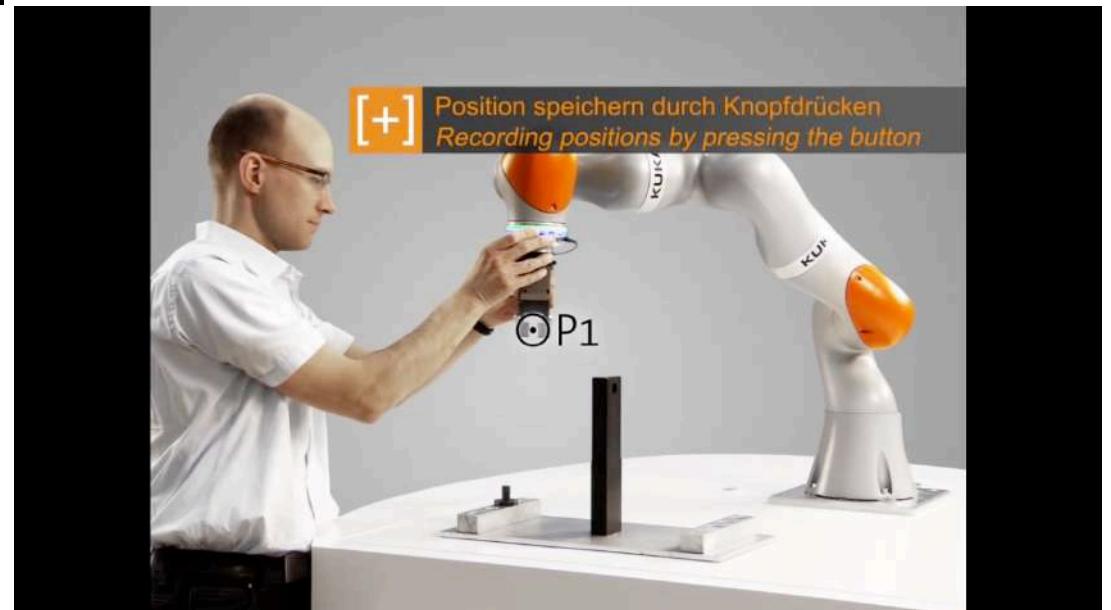
sketch of the original idea
— a first need & use of **safe**
physical Human-Robot Interaction (pHRI)

video

video

the working industrial solution

more videos on
[KUKA Robotics YouTube Channel](#)





Program

- **Introduction**
 - Manipulator arms (and few mobile base robots)
 - Industrial and service applications
- **Components**
 - Mechanical structures
 - Actuators and transmissions
 - Sensors
 - proprioceptive (encoder, tacho)
 - exteroceptive (force/torque, tactile, ultrasound, infrared, laser, vision)
- **Kinematic models**
 - Minimal representations of orientation
 - Direct and inverse kinematics of robot manipulators
 - Denavit-Hartenberg formalism for frame assignment
 - Differential kinematics: analytic and geometric Jacobians
 - Statics: Transformations of forces
 - Robot singularities



Program (continued)

- Planning of motion trajectories
 - Trajectory planning in the joint space for robot manipulators
 - Trajectory planning in the task/Cartesian space
- Motion control
 - Control system architectures
 - Kinematic control laws (in joint or in task/Cartesian space)
 - Independent joint axis control laws (P, PD, PID)
- Programming and Simulation
 - Programming language for industrial robots ([KRL](#))
 - Use of [Matlab/Simulink](#) and [CoppeliaSim](#) (V-REP)
 - If feasible, demos with [KUKA](#) robots (6-dof [KR5](#); 7-dof [LWR4+](#)) and [Universal Robots](#) (6-dof [UR-10](#), with non-spherical wrist)



Exams and beyond

- Type remote midterm test + written exam (+ oral exam, only if needed)
- Schedule of 2020-21 sessions
 - 2 sessions at the end of this semester
 - between January 7 and February 19, 2021
 - 2 sessions at the end of next semester
 - between May 31 and July 23, 2021
 - 1 session after the summer break
 - between September 1 and 17, 2021
 - 2 extra sessions **ONLY** for students of previous years, part-time, ...
 - March 15–April 21 and October 4–November 5, 2021
- Signing up to exams
 - on [Infostud](#)
- Master theses
 - samples available at DIAG Robotics Lab www.diag.uniroma1.it/labrob



Robot manipulators

available at DIAG Robotics Lab (S-218)

video



KUKA KR-5

video



KUKA LWR4+ (lightweight, about 14 kg)



Robot manipulators

available at DIAG Robotics Lab (S-218)

commercial video



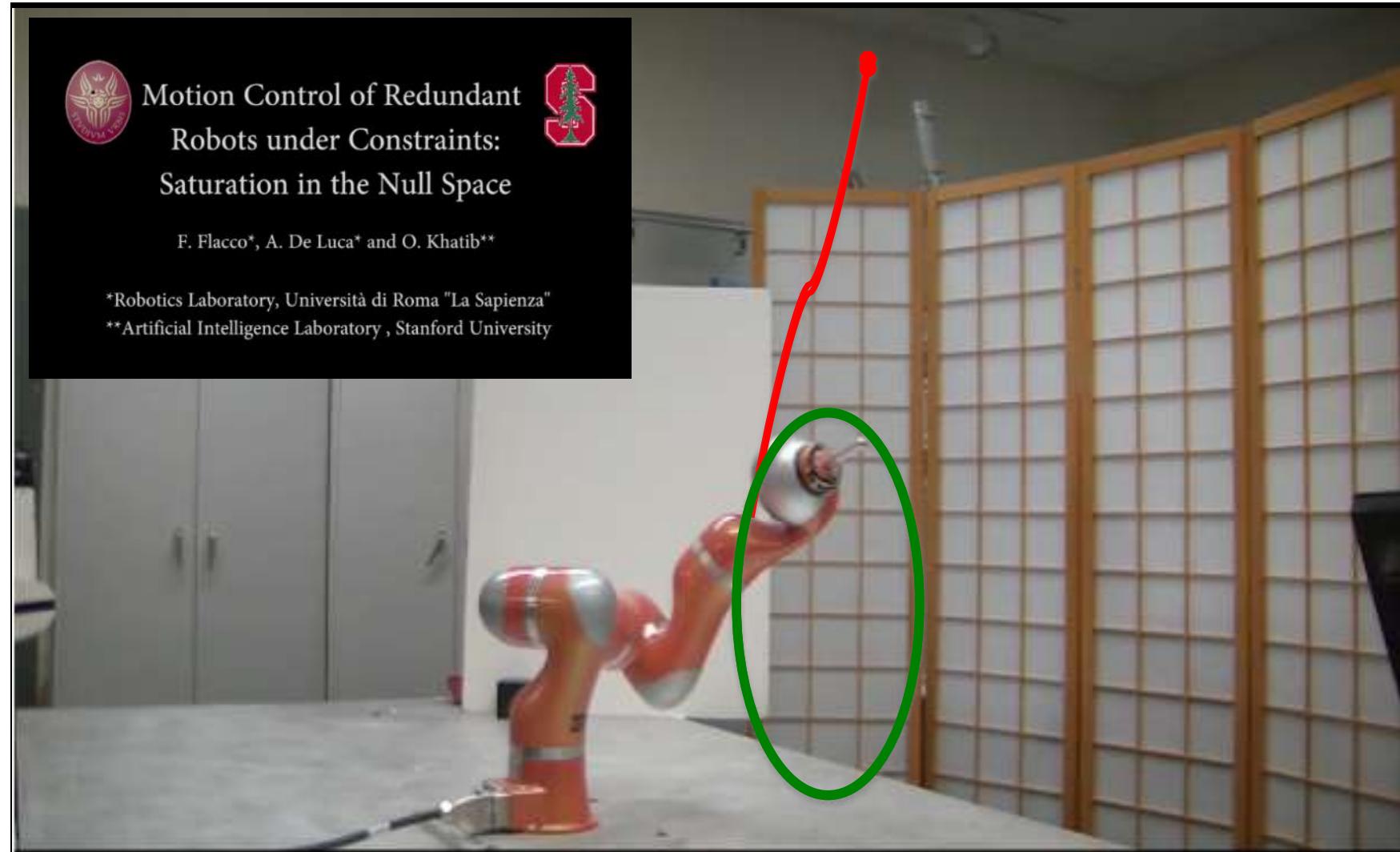
upon arrival (July 2016)



Universal Robots UR-10 (= 10 kg of payload)



Tracking a Cartesian trajectory with hard position/velocity bounds on robot motion



Motion Control of Redundant
Robots under Constraints:
Saturation in the Null Space



F. Flacco*, A. De Luca* and O. Khatib**

*Robotics Laboratory, Università di Roma "La Sapienza"

**Artificial Intelligence Laboratory, Stanford University

video DIAG Sapienza/Stanford, IEEE ICRA 2012



Robot control by visual servoing with limited joint motion range

Avoiding joint limits with a low-level fusion scheme

Olivier Kermorgant and François Chaumette

Lagadic team
INRIA Rennes-Bretagne Atlantique

video INRIA Rennes, IEEE/RSJ IROS 2011

Sensor-based robot control in dynamic environments (coexistence with human)



A Depth Space Approach to Human-Robot Collision Avoidance

F. Flacco*, T. Kröger**, A. De Luca* and O. Khatib**

*Robotics Laboratory, Università di Roma "La Sapienza"

**Artificial Intelligence Laboratory , Stanford University

video DIAG Sapienza/Stanford, IEEE ICRA 2012



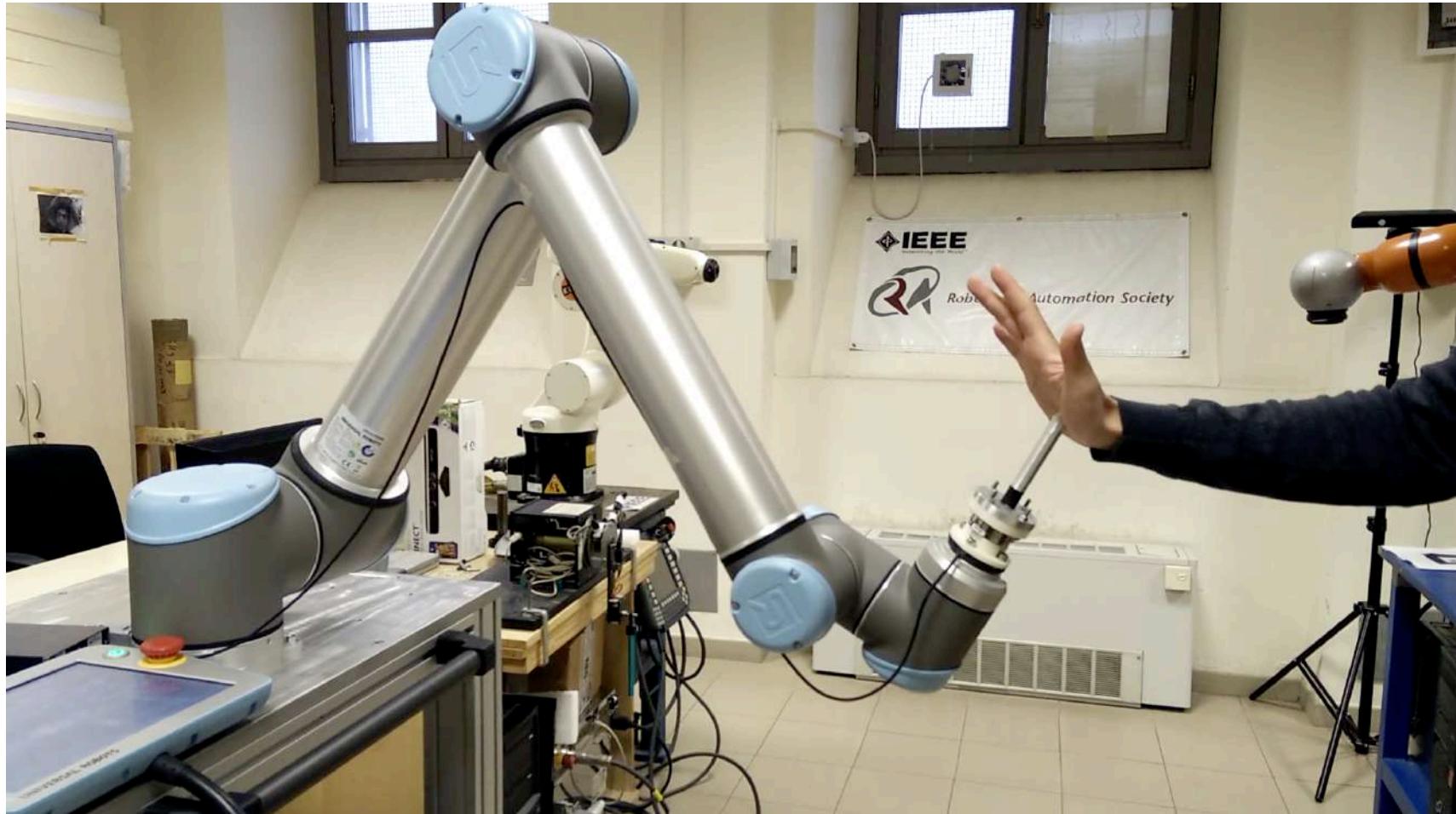
Safe physical human-robot interaction (sensor-less (!) and on a conventional industrial robot)



video DIAG Sapienza, IEEE ICRA 2013



Human-robot collaboration (with a real F/T and a “virtual” sensor to distinguish contacts)



video DIAG Sapienza, J. of Mechatronics, 2018

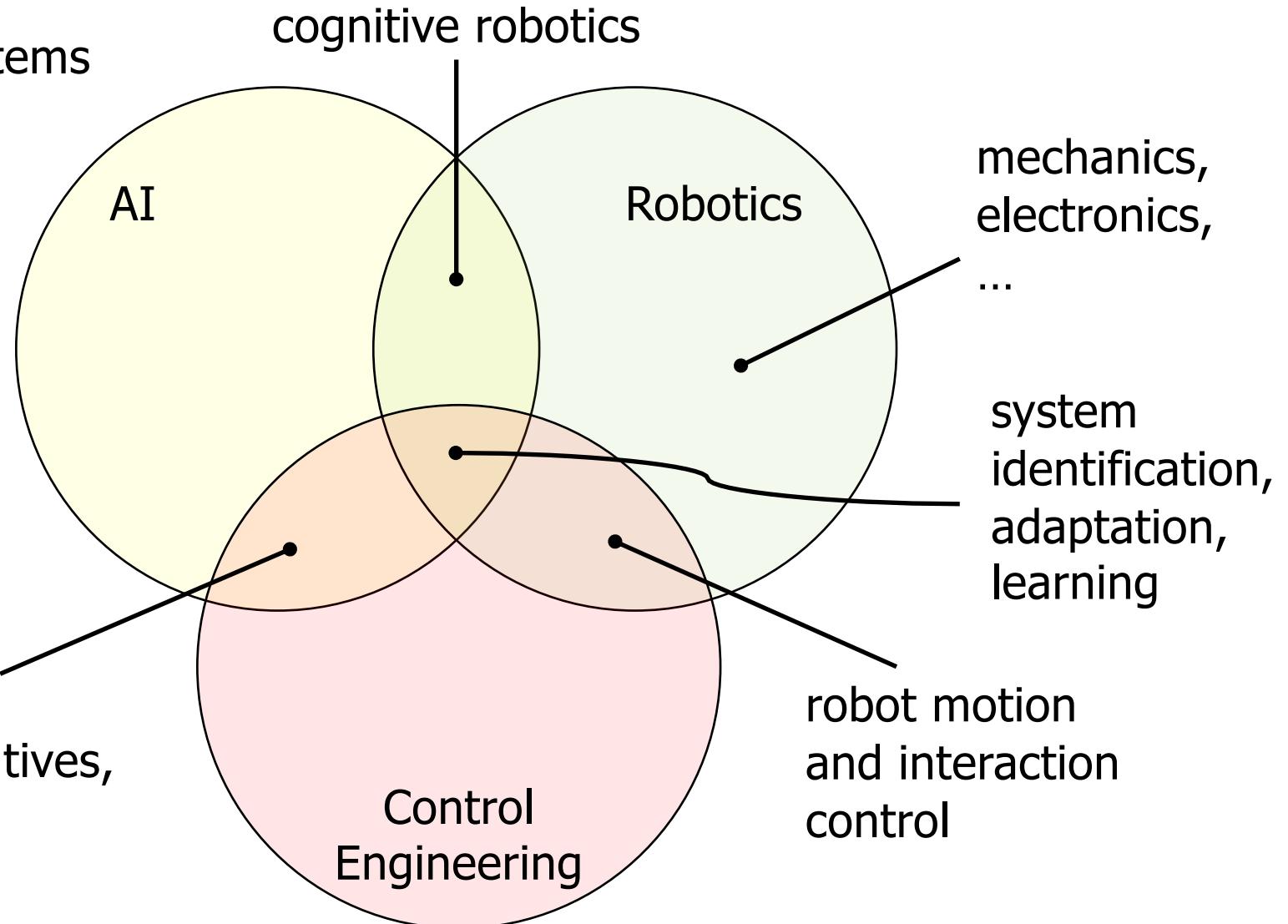


Next generation of intelligent robots?

Robots =
embodied AI systems

Robotics =
science
of artifacts
intelligently
actuated and
interacting with
the real world

model-based
techniques,
dynamic primitives,
uncertainty





Horizon Europe (2021-27)

Strategic Research, Innovation and Deployment Agenda

AI, Data and Robotics Partnership

Third release
September 2020

A joint initiative by

BDV, CIAIRE, ellis, EurAi, euRobotics

<https://ai-data-robotics-partnership.eu>

Cross-Sectorial AI, Data and Robotics Technology Enablers

Sensing and Perception
Knowledge and Learning
Reasoning and Decision Making
Action and Interaction
Systems, Methodologies, Hardware & Tools

Robotics Deep Dive

Physical Interaction
Physical and Psychological Safety
Actuated Mechanical Structures
Unpredictable and Unknown Environments
Irreversible Actions

4 Market Prioritization in Robotics (from Horizon 2020)

Healthcare
Maintenance and Inspection of Infrastructures
Agri-Food
Agile Production



Specific to Robotics

- hard physical nature of Robotics ("AI embodied")
- wide range of technologies are integrated within robotic systems
- skill mix for success is broader than that for AI or Data alone
- robots are realizations of advanced system-level concepts
 - such as autonomy, control, sensing and programming
- robots are both producers and consumers of data
 - physical model-based approaches, generation of data-driven models
- decision makers and general public need a better understanding of what Robotics is and can achieve, and how it can be deployed
 - Fukushima, Covid-19,



AI & Robotics point of views

one of the enabling technologies

... its relations with the other technologies

	Sensing and Perception	Knowledge and Learning	Reasoning and Decision Making	Systems, Hardware, Methodologies and Tools
Action and Interaction	<p>Depends on sensing of motion and mechanical properties</p> <p>Relies on perception for interaction</p> <p>Uses recognition of actions and sequences of interactions in people</p>	<p>Gets semantic knowledge around objects and human actions</p> <p>Gets data on objects and places</p>	<p>Depends on real-time context-aware decision making</p> <p>Trusted decision making</p>	<p>Depends on fast reactive architectures for control</p> <p>Relies on edge-based AI</p> <p>Requires assurance of safe operation and data privacy</p>

Real-time interpretation of multi-modal data
Safe monitoring in human environments

Active exploration strategies
Adaptive decision-making models from sparse data

Planning and re-planning under uncertainty and incomplete knowledge in dynamic environments
Real-time control (distributed/decentralized)

Safe control of physical human-interaction
Agility (speed and strength) of collaborative robots
InterAction Technology (lightweight, compliant & soft devices/materials)
Energy-efficient, robust and sustainable design



Robotics around the world...

Springer Handbook of Robotics (2nd Edition, July 2016)

robots
the journey continues



Preview of Robotics 2 (next semester)

- Advanced kinematics / Robot dynamics
 - Calibration
 - Redundant robots
 - Dynamic modeling: Lagrange and (recursive) Newton-Euler methods
 - Identification of dynamic parameters
- Control techniques
 - Free motion linear and nonlinear feedback control, iterative learning, robust control, adaptive control
 - Constrained motion impedance and hybrid force-velocity control
 - Visual servoing (kinematic approach)
- Special topics
 - Diagnosis and isolation of robot actuator faults
 - Human-robot collision avoidance & detection, with safe robot reaction



Other courses about Robotics and Control

- Autonomous and Mobile Robotics (6 credits), I semester, year 2
 - kinematics, planning, control of wheeled and legged mobile robots
 - motion planning with obstacles, navigation, and exploration
 - Prof. Oriolo www.diag.uniroma1.it/oriolo/amr
- Medical Robotics (6 credits), II semester
 - robot surgical systems, haptics, and more ...
 - Prof. Vendittelli www.diag.uniroma1.it/vendittelli/didattica/mr
- Elective in Robotics (12 credits) or Control Problems in Robotics (6 credits)
 - I-II semesters, starting this semester
 - 4 modules of 3 credits (for CPR, MCER students take 2 modules out of the 4 in EiR)
 - research-related subjects and seminars
 - multiple instructors www.diag.uniroma1.it/vendittelli/EIR
- Robot Programming (lectures, not for credits!)
 - robot programming using C++, ROS, NAO SDK as development frameworks
 - see Prof. Nardi www.diag.uniroma1.it/nardi/Didattica/CAI/robpro-free.html



Robotics 1

Industrial Robotics

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





What is a robot?

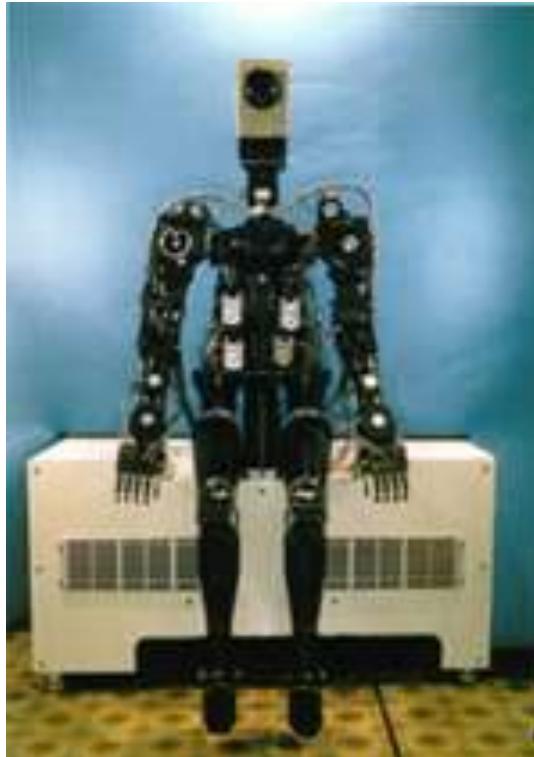
- industrial definition (RIA = Robotic Institute of America)
re-programmable multi-functional manipulator
designed to move materials, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks, which also **acquire information from the environment** and move intelligently in response
- ISO 8373:2012 definition
an automatically controlled, reprogrammable, multipurpose manipulator programmable **in three or more axes**, which may be either **fixed in place or mobile** for use in industrial automation applications
- more “**visionary**” definition
intelligent connection between **perception** and **action**



Robots !!



Comau H4
(1995)



Waseda WAM-8
(1984)



Spirit Rover
(2002)

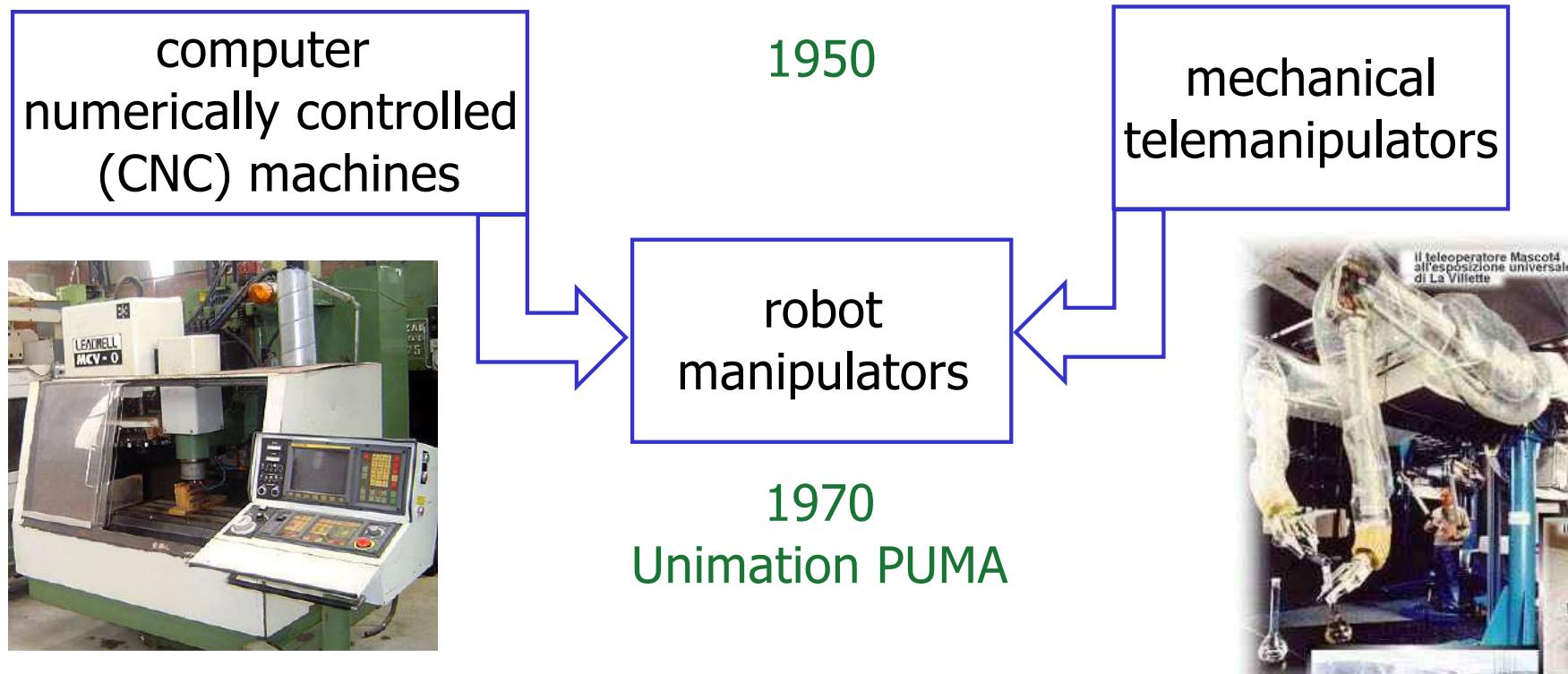


A bit of history

- **Robota** (= “work” in slavic languages) are artificial human-like creatures built for being inexpensive workers in the theater play **Rossum’s Universal Robots (R.U.R.)** written by Karel Capek in 1920
- **Laws of Robotics** by Isaac Asimov in **I, Robot** (1950)
 1. A robot may not injure a human being or, through inaction, allow a human being to come to harm
 2. A robot must obey orders given to it by human beings, except where such orders would conflict with the First Law
 3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law



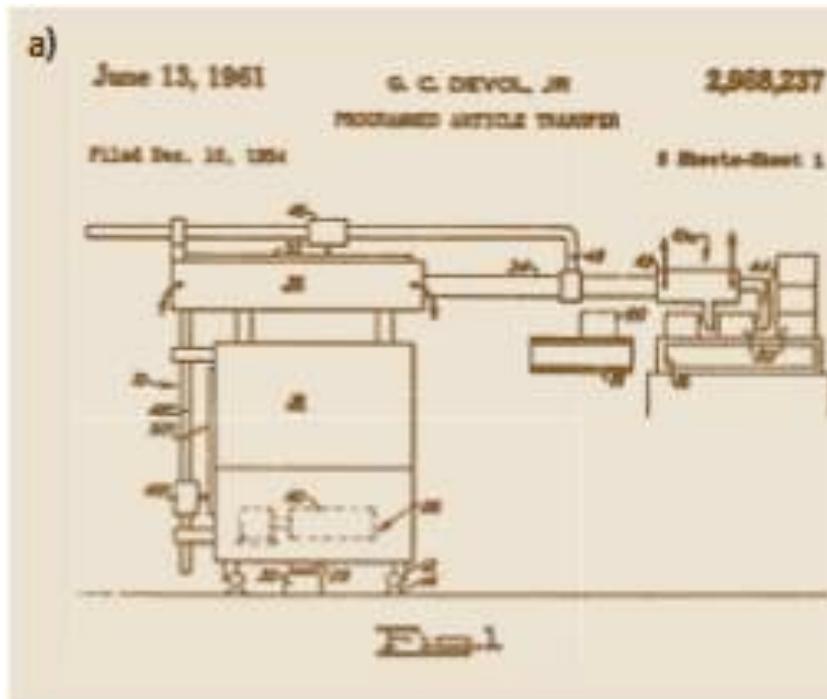
Evolution toward industrial robots



- with respect to the ancestors
 - **flexibility** of use
 - **adaptability** to a priori unknown conditions
 - **accuracy** in positioning
 - **repeatability** of operation



The first industrial robot



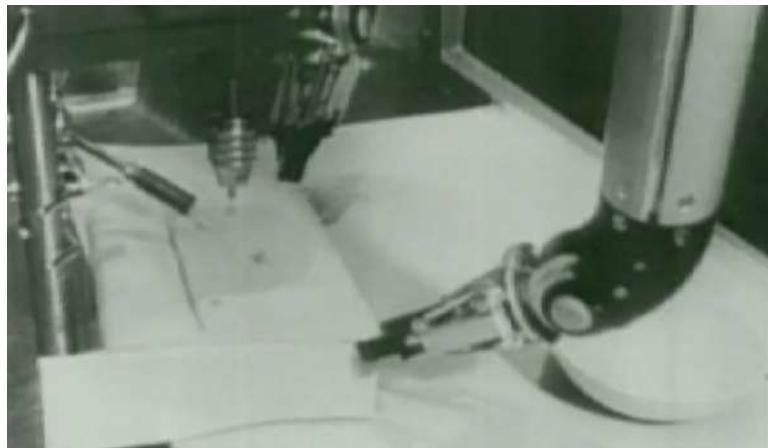
US Patent

General Motor plant, 1961

G. Devol and J. Engelberger (Unimation)



Historical pictures and clips



bimanual remote manipulation
at Oak Ridge Nat'l Labs



Unimate 6-dof robots

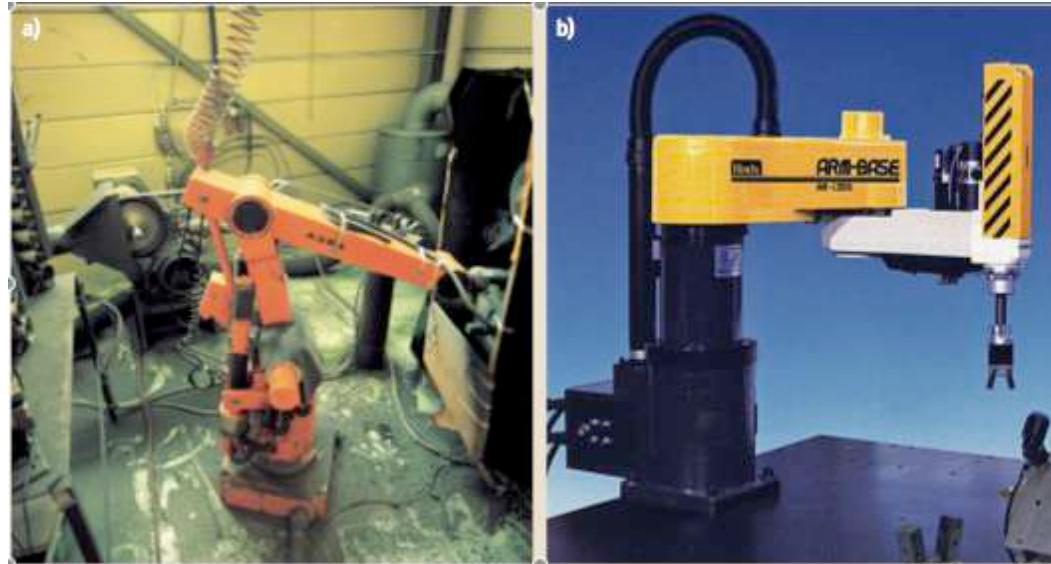
video

video



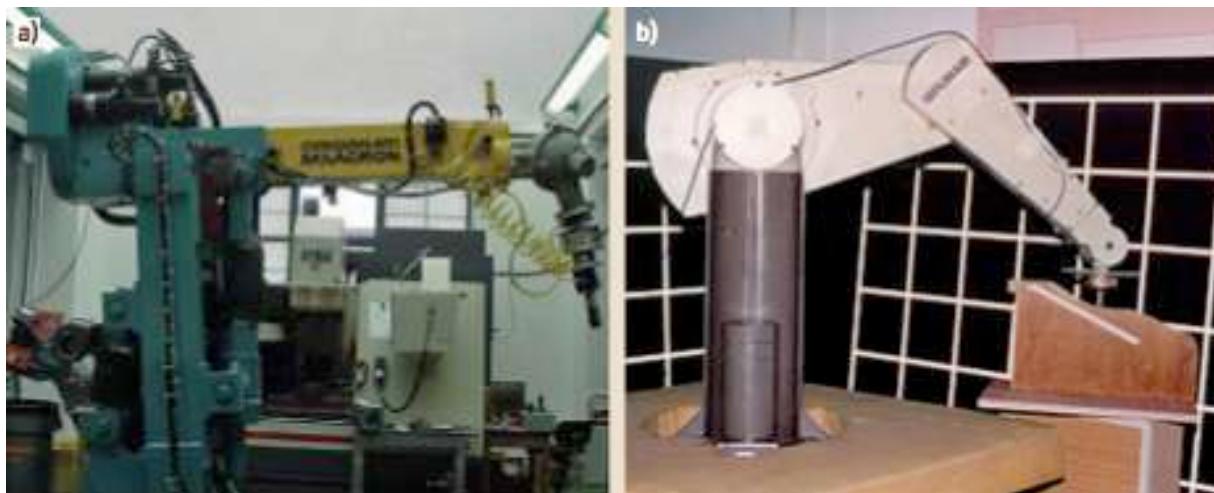
Robot manipulators

ASEA IRB-6
(1973)
first robot
all-electric-drives



Hirata AR-300
(1978)
first SCARA
robot

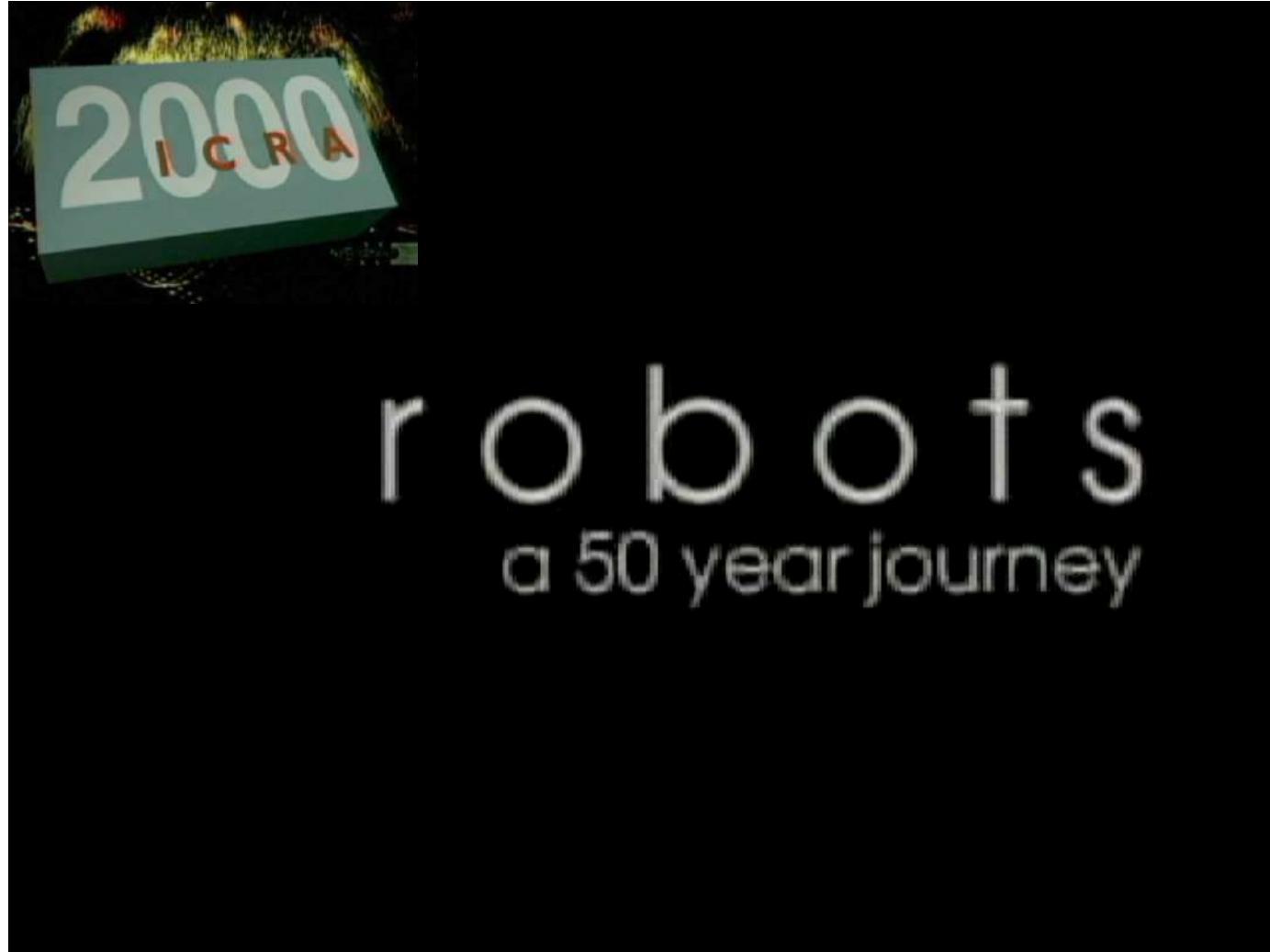
Cincinnati
Milacron T3
(1974)
first micro-
computer
controlled
robot



Unimation
PUMA 560
(1979)
6R with
human-like
dexterity

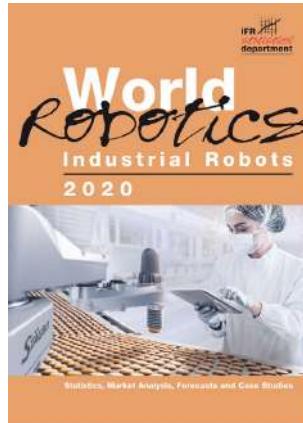
robots – a 50-year journey

robotics research up to 2000



Video compiled for the IEEE ICRA 2000 conference, S. Francisco

World Robotics 2020



executive summary for 2020
 statistics by IFR
 issued yearly in late September
 (for back issues since 2007,
 check course web site)



- total worldwide stock at end 2019: **2.7 million units** of operational industrial robots (+12% w.r.t. 2018, +13% CAGR since 2014)
- new robot sales in 2019: **~373K** (-12%, 3rd highest ever, +11% CAGR since 2014)
- robot market value in 2019: **\$13.8 billion** (without software and peripherals)
- robotic systems market value: **~4 times** as much
- **China** expanded further as the largest market since 2013, now with a **38%** share
- **73%** of new robot installations in **5 countries**: China, Japan, USA, Korea, Germany

$$\text{Compound Annual Growth Rate: CAGR} = \left(\frac{V_{end}}{V_{begin}} \right)^{1/\text{years}} - 1$$

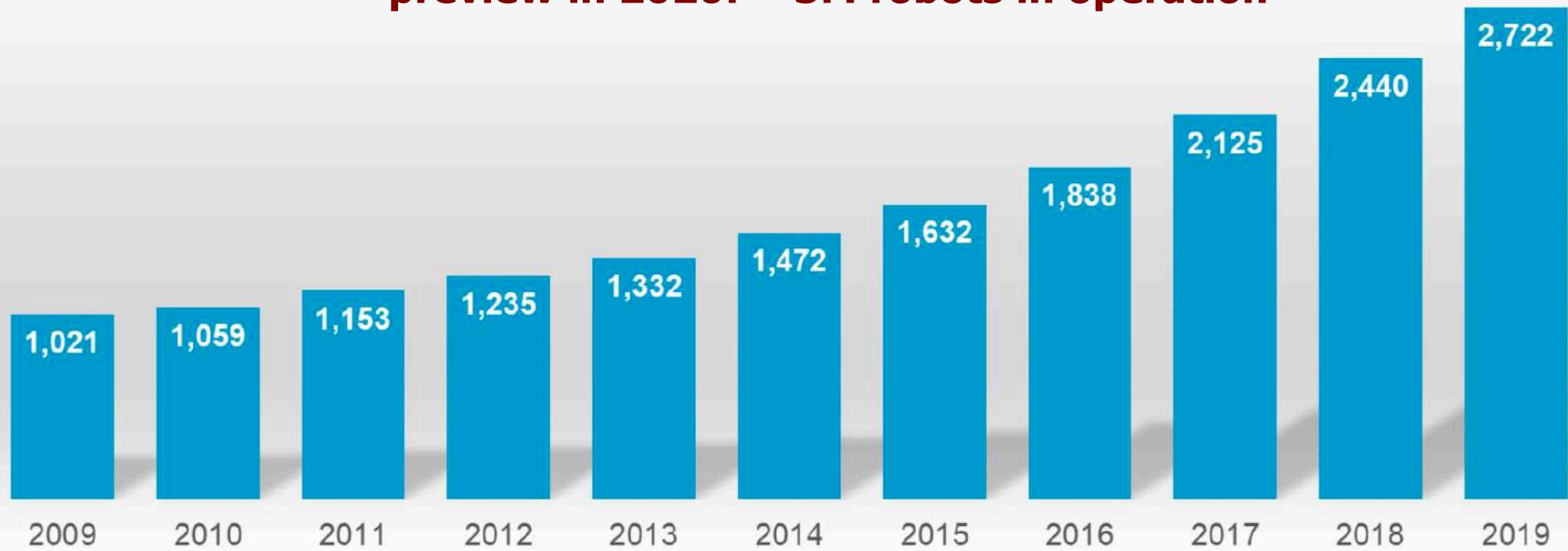
Diffusion industrial robots in operation worldwide



Operational stock of industrial robots - World

1,000 units

preview in 2020: ~3M robots in operation



Source: World Robotics 2020

(as reference, industrial robots in 1973 = 3K, 1983 = 66K, 1993 = 575K, 2003 = 800K)
length of robot service life is estimated in **12-15 years**



Diffusion

industrial robots in operation by world area

Operational stock of industrial robots
('000 of units)



Source: World Robotics 2020

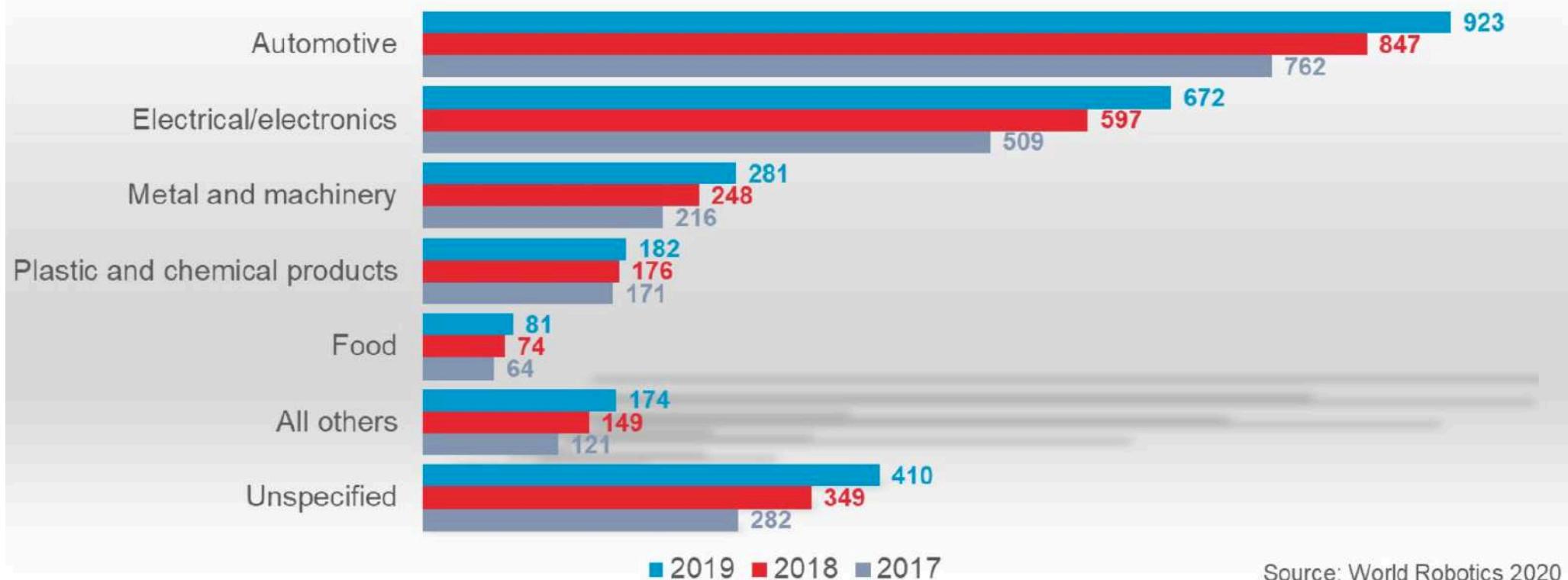
preview in 2020: almost 1M robots operating in China!



Diffusion robots in industrial sectors

Operational stock of industrial robots by customer industry - World

1,000 units



Source: World Robotics 2020

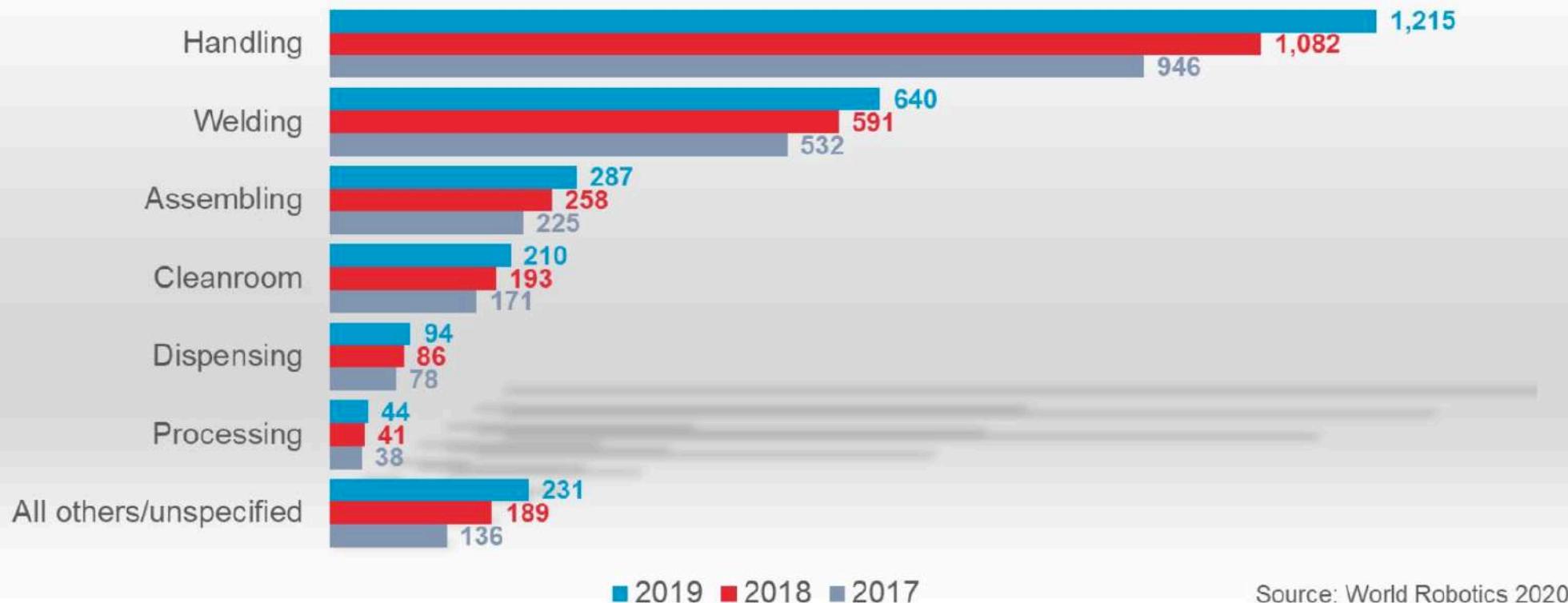
70% of robots are deployed in three main industries



Diffusion robots by main applications

Operational stock of industrial robots by application - World

1,000 units



Source: World Robotics 2020

almost 70% of robots are used for material handling or welding



Annual supply new industrial robots worldwide

Annual installations of industrial robots - World
1,000 units



Source: World Robotics 2020

stop of growth rate in 2019: automotive transition, trade & political headwinds

preview in 2020: deferred investments, plummeted consumer demand, travel restrictions, disrupted supply chains (due also to Covid-19)

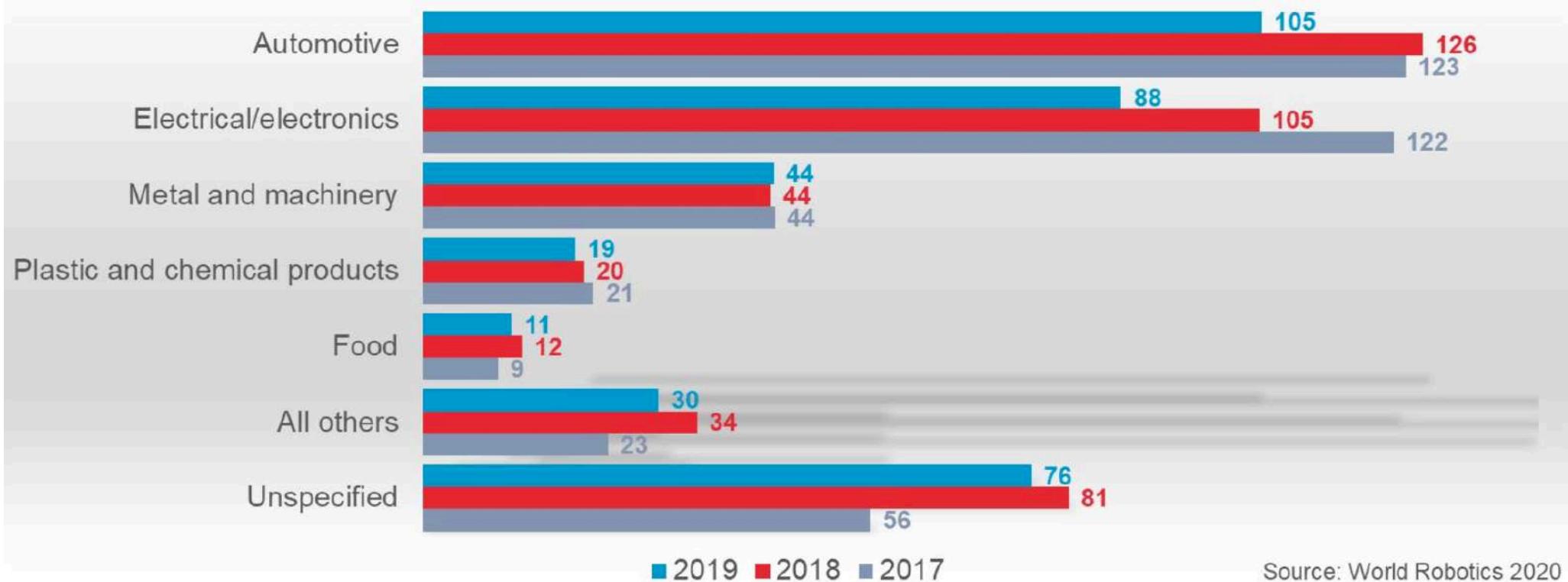
medium term preview: crisis will be a digitalization booster creating growth opportunities for robotics industry worldwide



Annual supply new robots by industrial sectors

Annual installations of industrial robots by customer industry - World

1,000 units

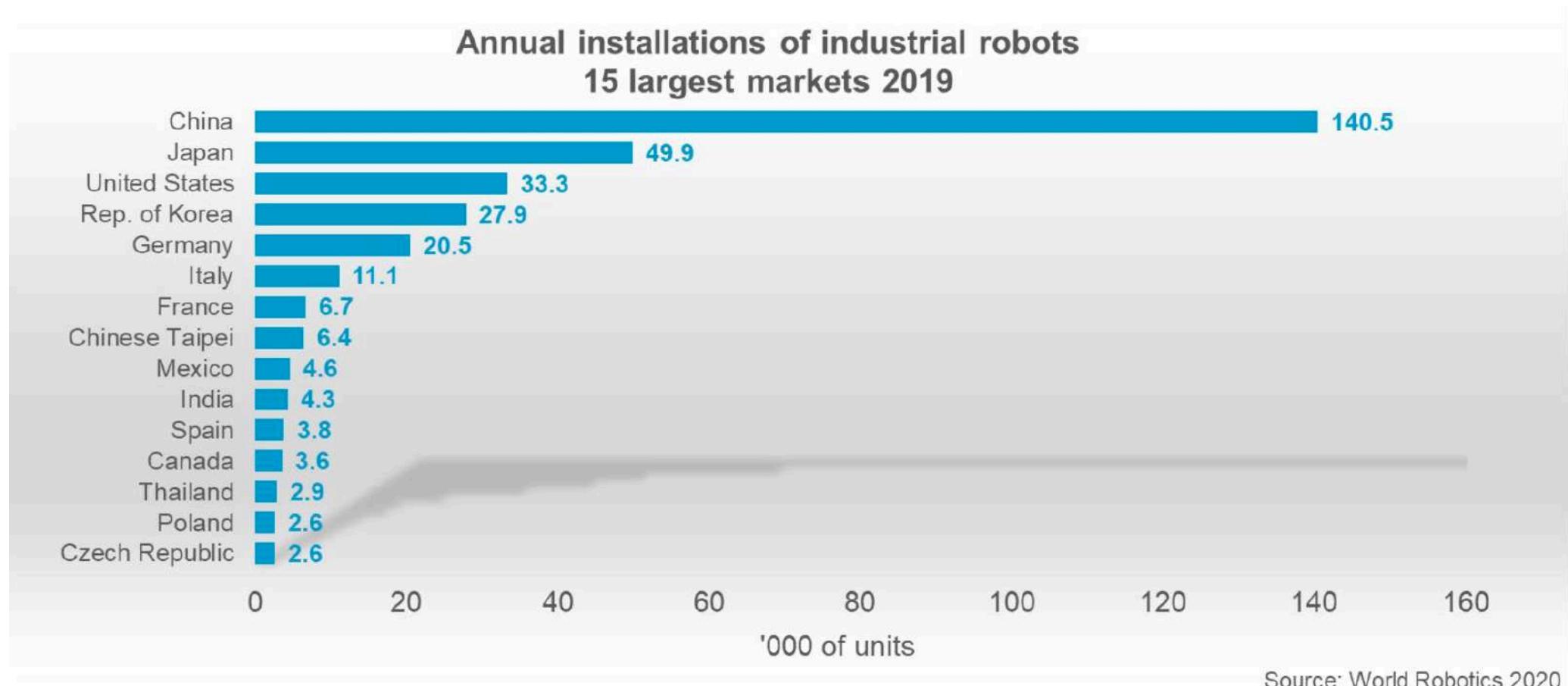


Source: World Robotics 2020

the two major industries struggled in 2019 (especially in Asia)



Annual supply new installations in top markets (countries)



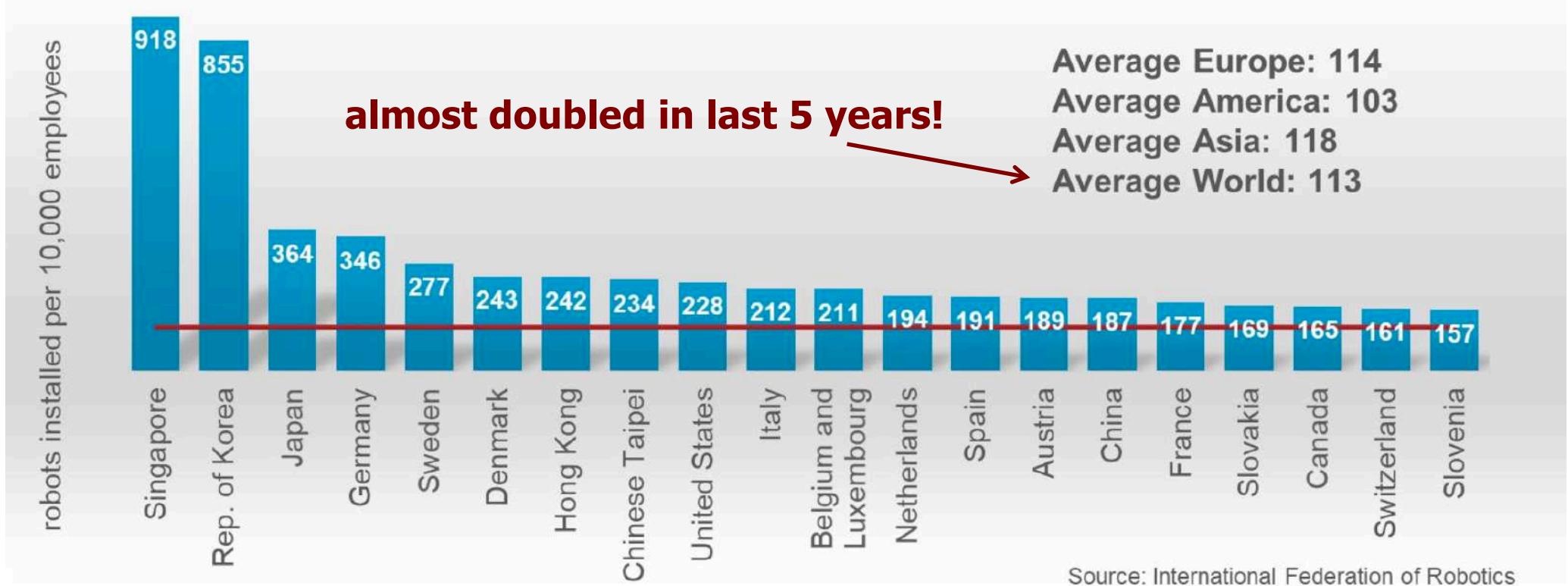
China well ahead, with 5 markets accounting for 73% of total supply

Italy (2nd EU market): double as many new robots installed as in 2015



Density of robots

Robot density in the manufacturing industry 2019

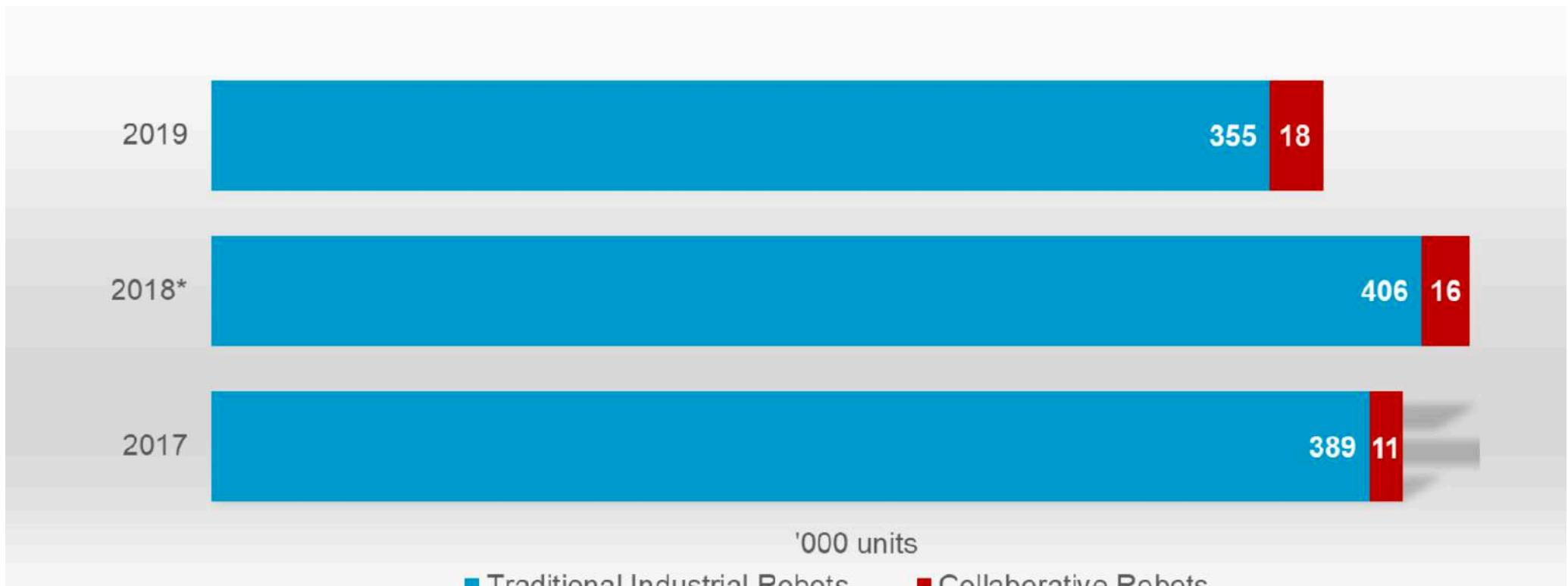


number of **robots per 10000 employees**
in the **manufacturing** industry



Collaborative robots

Collaborative and traditional industrial robots



*revised

Source: International Federation of Robotics

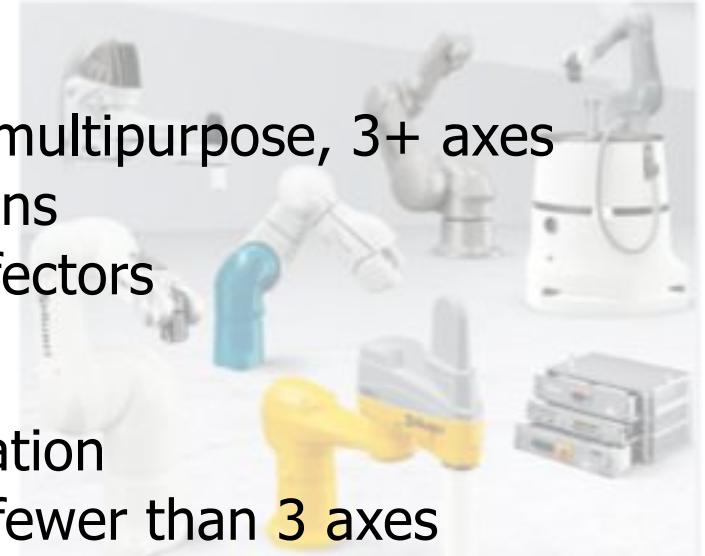
a small but steady **growing market** in the industrial setting



Industrial & service robots

Industrial robots

- automatically controlled, programmable, multipurpose, 3+ axes
- for use in industrial automation applications
- equipped with application-specific end-effectors



Service robots

- perform tasks excluding industrial automation
- usually application-specific design, often fewer than 3 axes
- sometimes not fully autonomous but remote-controlled

different customers, pricing, machinery, distribution channels, suppliers



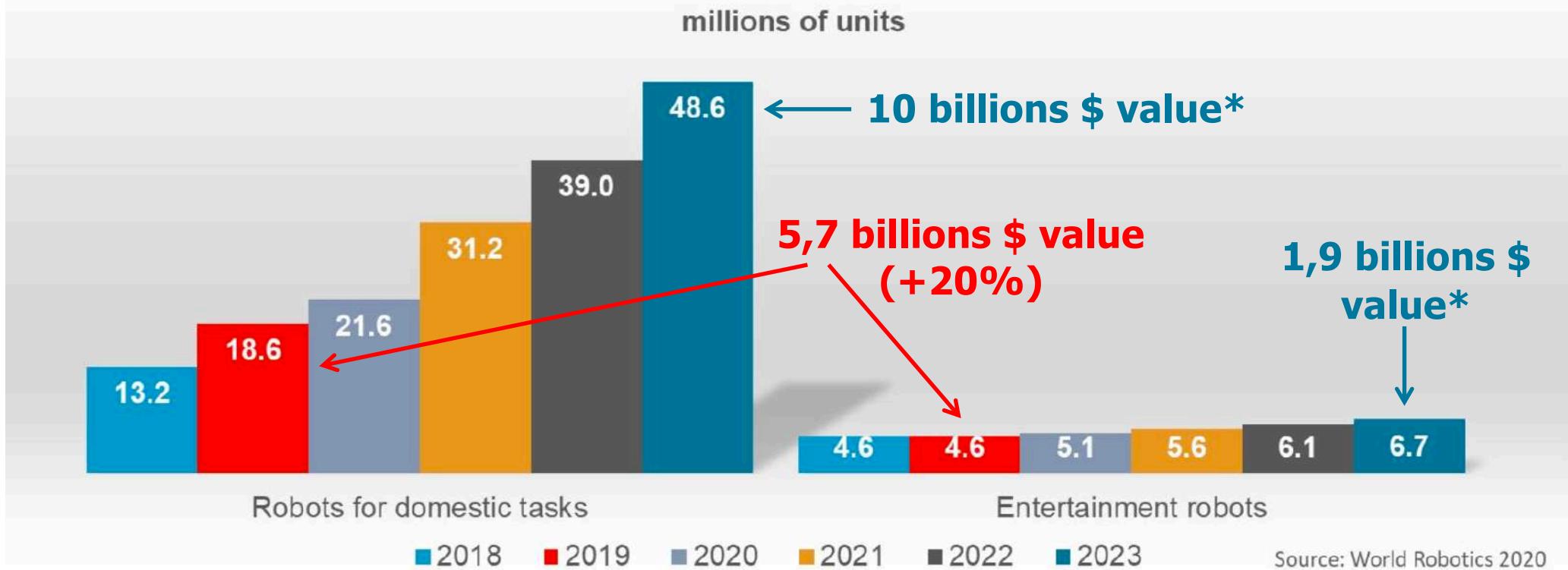
**... but separation
line is blurring:
same unit can act
as both, depending
on the application**





Personal/domestic service robots data (2018-19) and forecast*

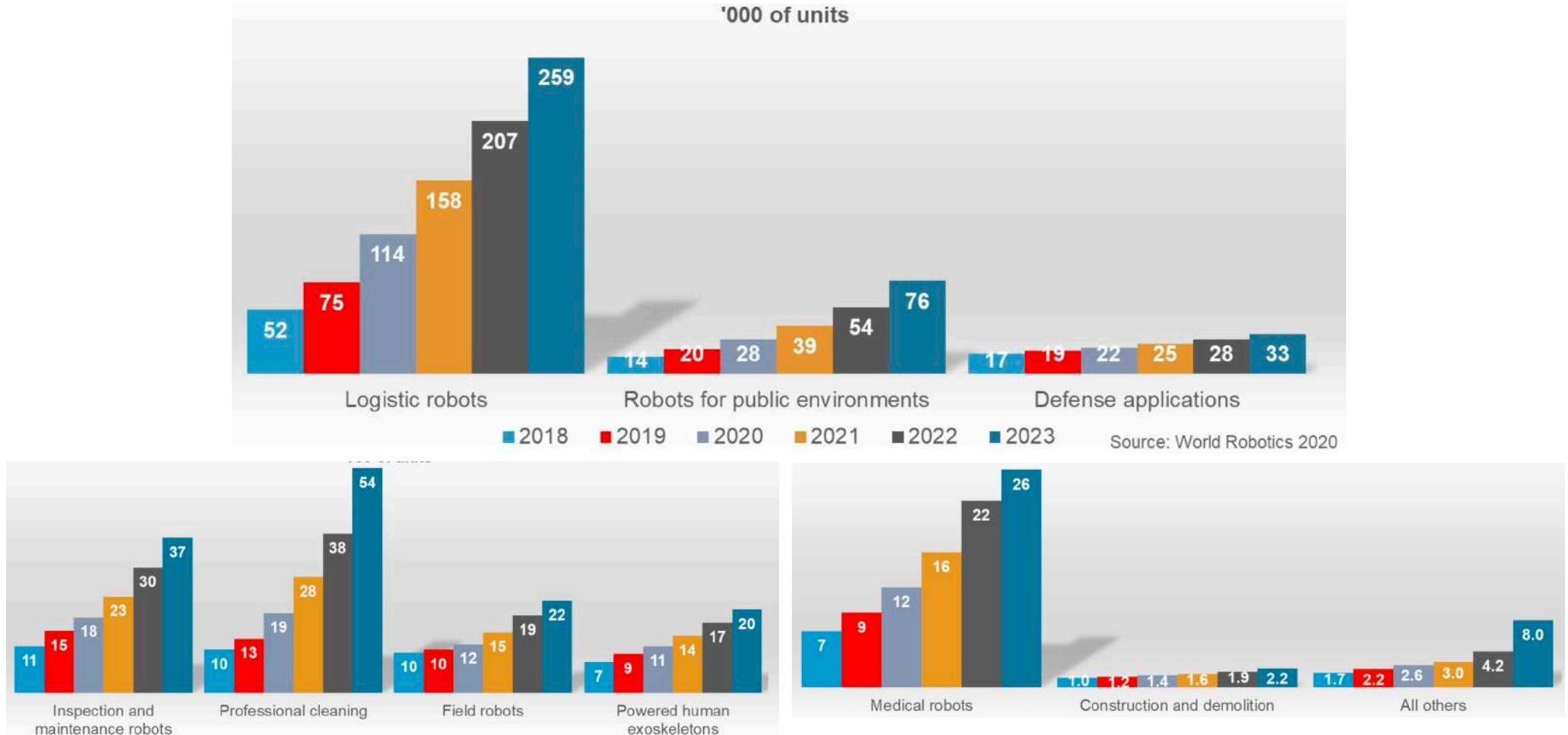
Service robots for personal/domestic use.
Unit sales 2018 and 2019, potential development 2020-2023



**vacuuming and floor cleaning: true tasks for robots
steady growth of turnover expected**



Professional service robots data (2018-19) and forecast*



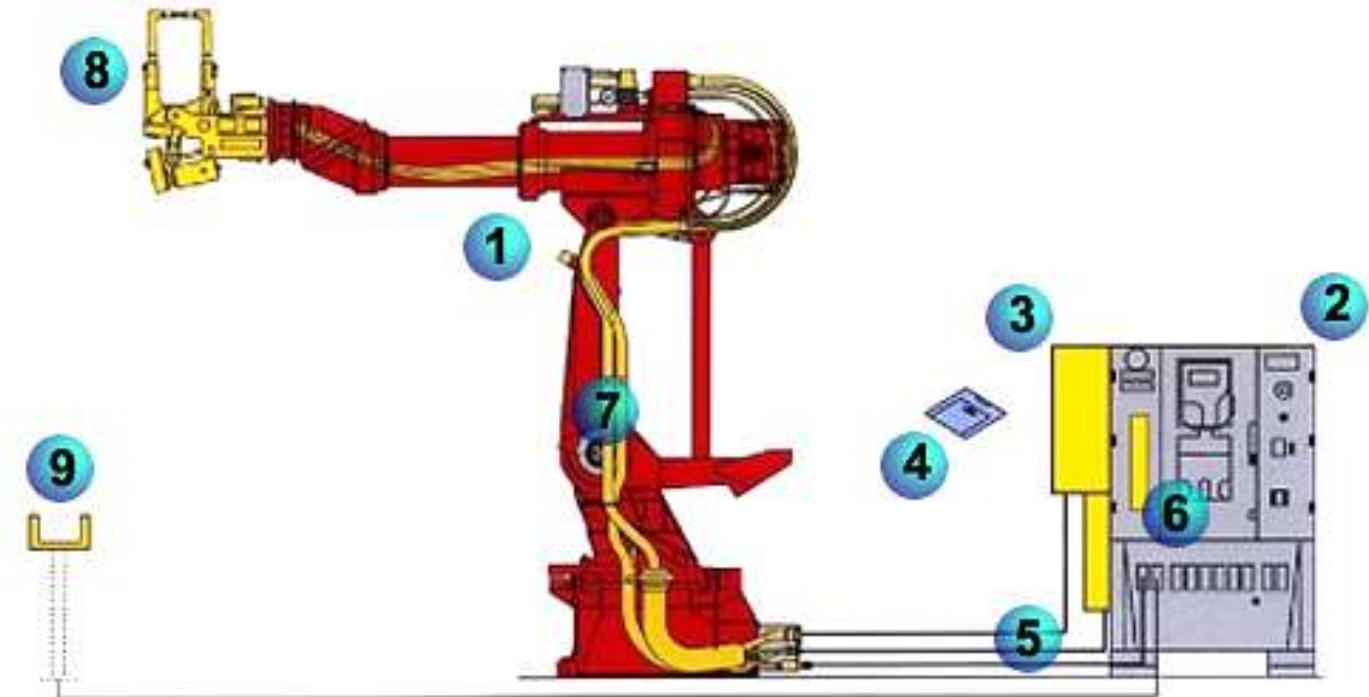
robots (AGV, AMR) in logistics are still the growth drivers

Europe is the main supplier of service robots (with many young start-ups!)



Industrial robot and its auxiliary equipments

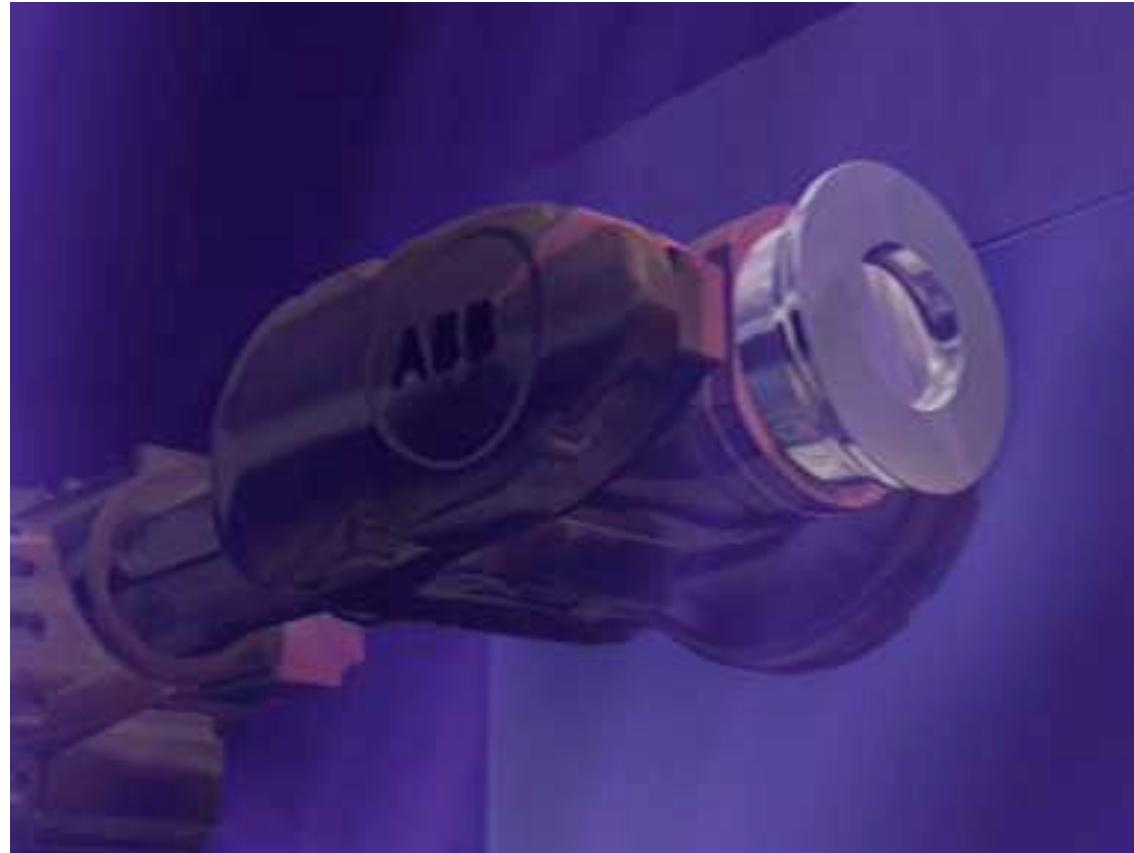
1. Comau SMART H robot
2. C3G Plus controller
3. Welding control box
4. Application software
5. Air/water supply
6. SWIM Board
7. Integrated cables
8. Welding gun
9. Auxiliary devices in the robotic cell
(servo-controlled axes)



SWIM = Spot Welding Integrated Module



ABB IRB 7600



commercial [video](#) by ABB



Industrial applications

- manipulation (pick-and-place, handling, machine feeding)
- assembly and packaging
- spray painting and coating (nozzles)
- arc welding or spot welding (with pneumatic or servo-controlled guns)
- laser cutting and welding
- gluing and sealing
- mechanical machining operations (milling, drilling, deburring, grinding, ...)



sme²
video





A day in the life of an industrial robot

- At BMW car production line with ABB robots



video

video



pick-and-place
with end-effector
to reorient part

pick-and-place
with support
to reorient part

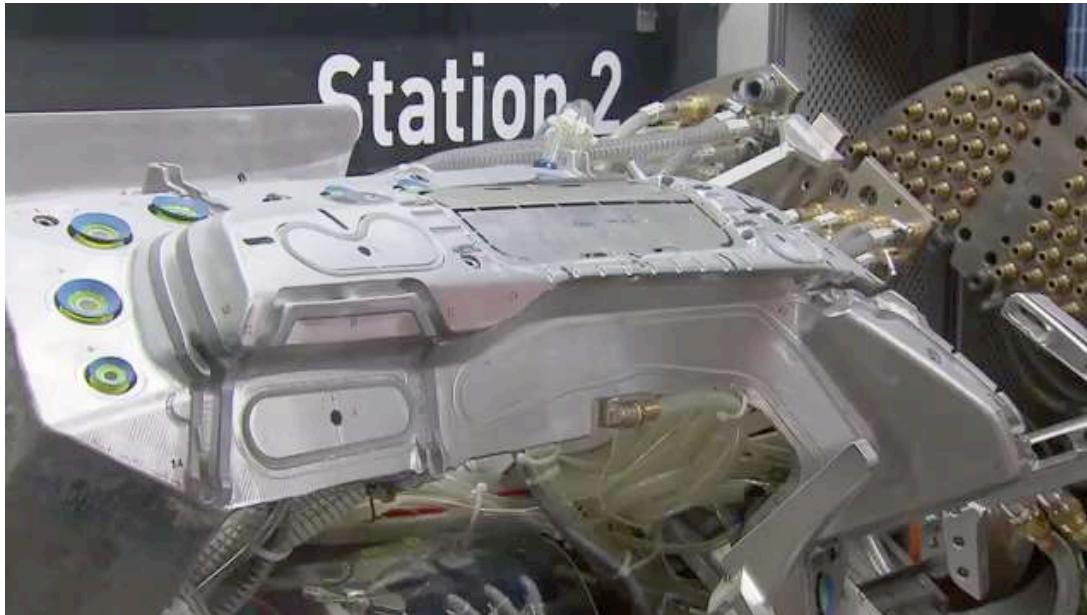


A day in the life of an industrial robot



video
video

pick-and-place
heavy parts and
human intervention



metal cutting
on a supporting
machine with dofs
(video speeded up
at some point)



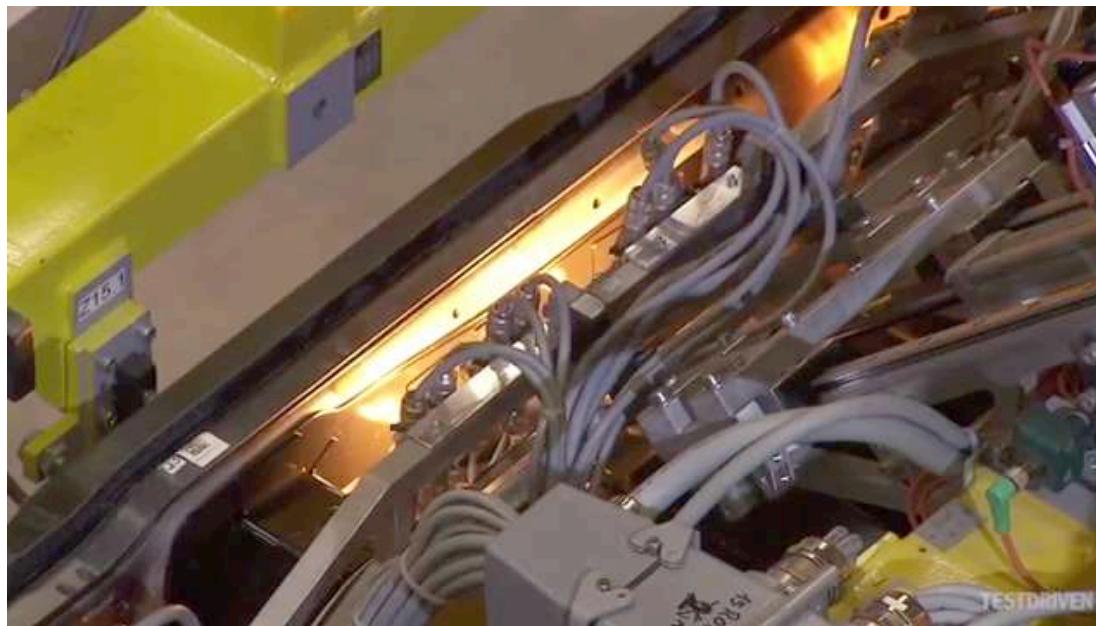
A day in the life of an industrial robot



glue deposit
(on fancy paths!)

video

video



cooperation of
multiple robots
for handling and
inspecting/sealing
a car body



A day in the life of an industrial robot



video

video

coating parts
for rust and corrosion
protection



spray painting



A day in the life of an industrial robot



hood deburring
with a suspended tool

[video](#)

[video](#)



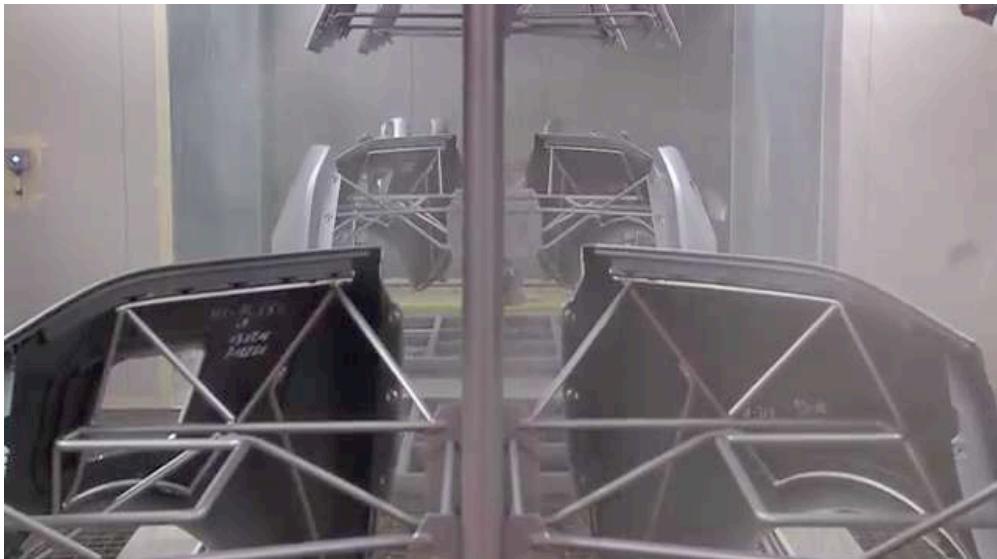
test measurements
with assembly on a AGV



What a robot should do and what cannot do

Yet

video



spray painting
very unhealthy
for human operators

video

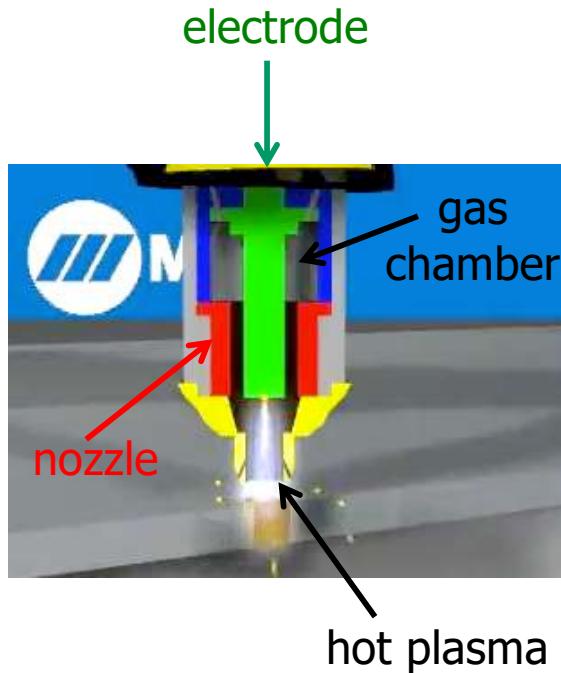


assembly of flexible
or complex parts
(here a car dashboard)

⇒ human-robot **collaboration**
(co-bots or co-workers)



Plasma cutting

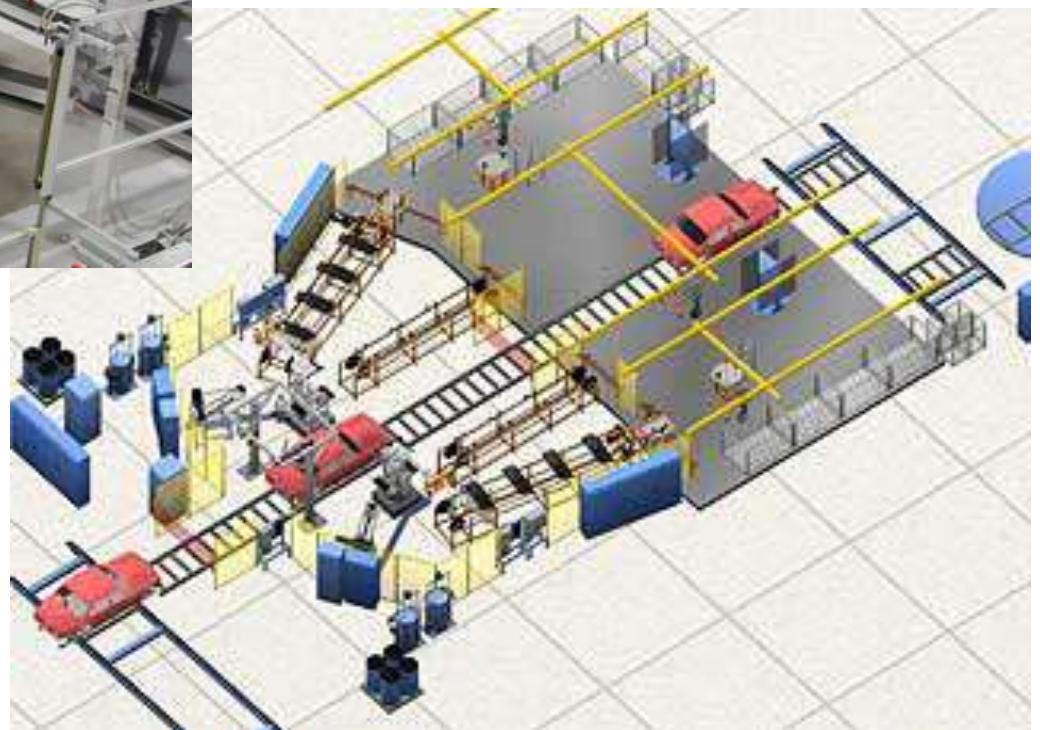


video

small KUKA robot used for plasma cutting of a stainless steel toilet
(courtesy of Engenious Solutions Pty)

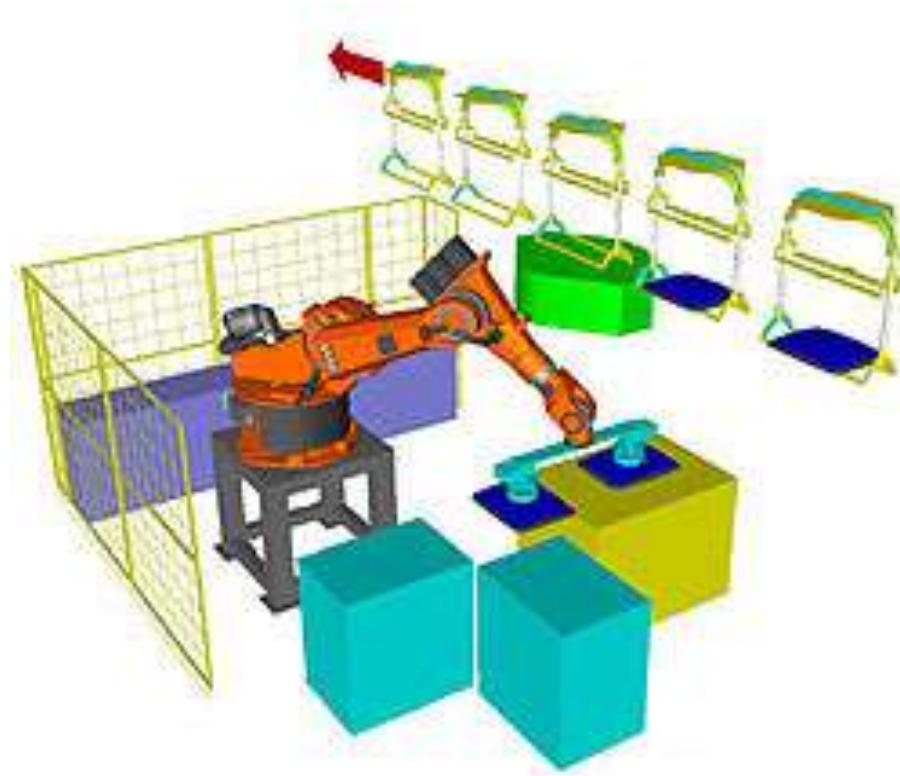


Robotized workcells





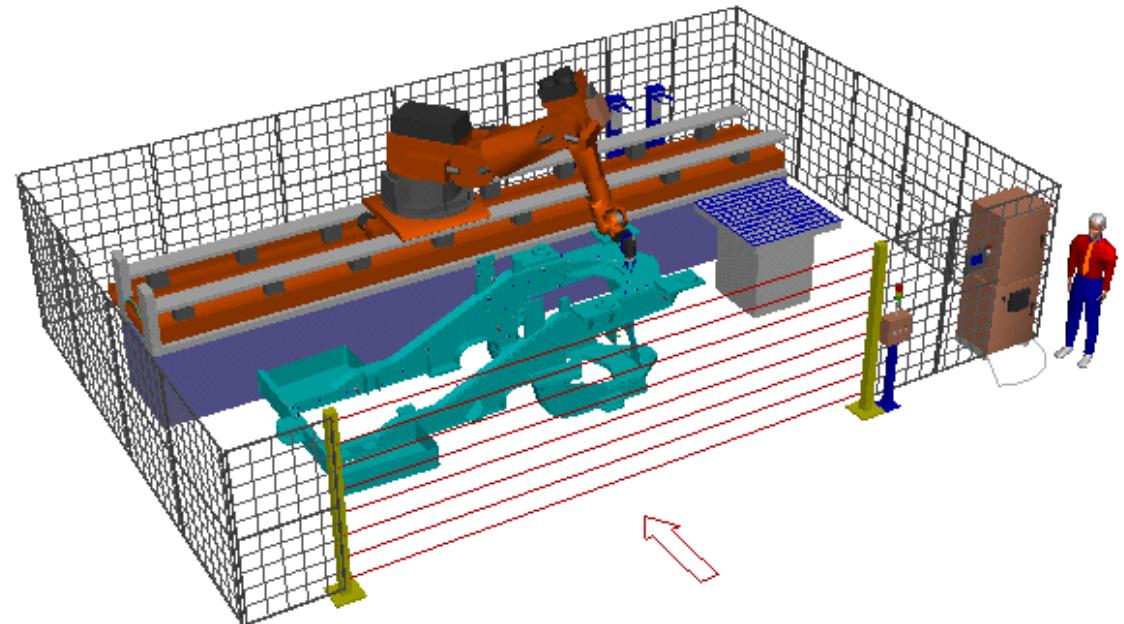
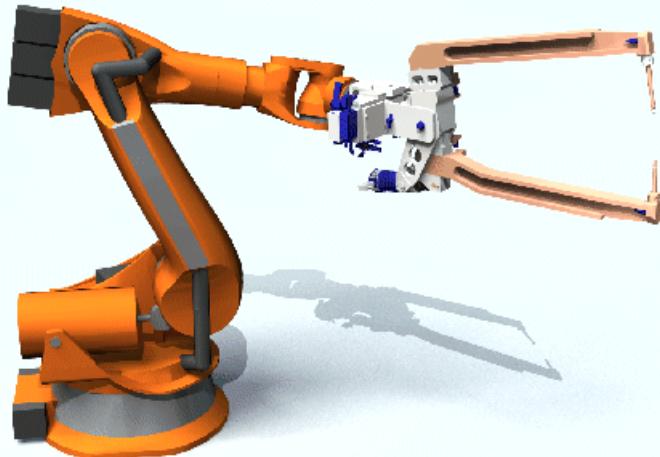
3D simulation of robotic tasks



- analysis of operative cycle times
- off-line programming and optimization
- layout design and collision checking
- 3D graphic simulation



Welding - 1



- spot with servo-controlled gun

- stud welding



Welding - 2



- spot (discrete) or arc (continuous)



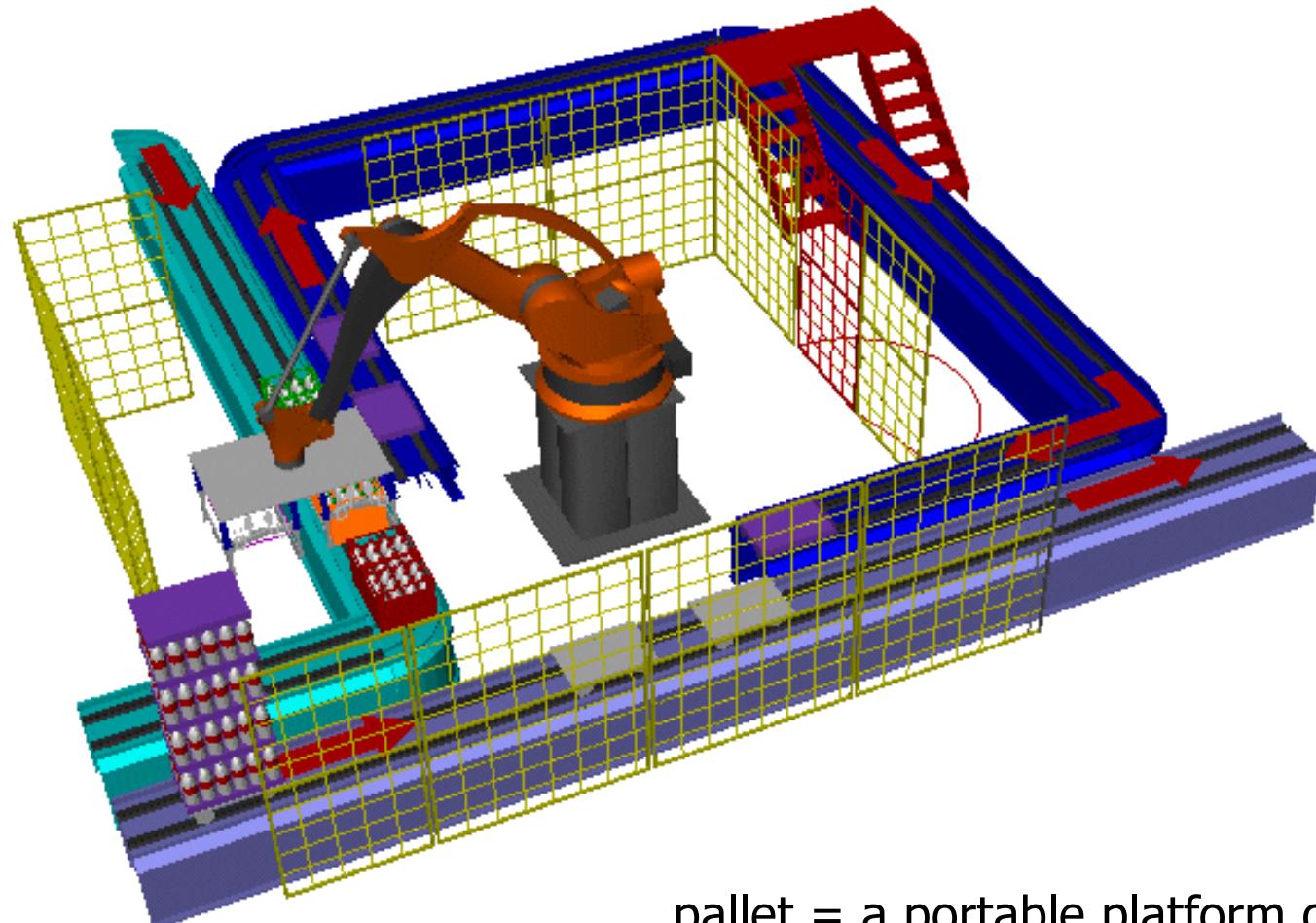
Two cooperating robots in arc welding



ABB video at Laxa, Sweden



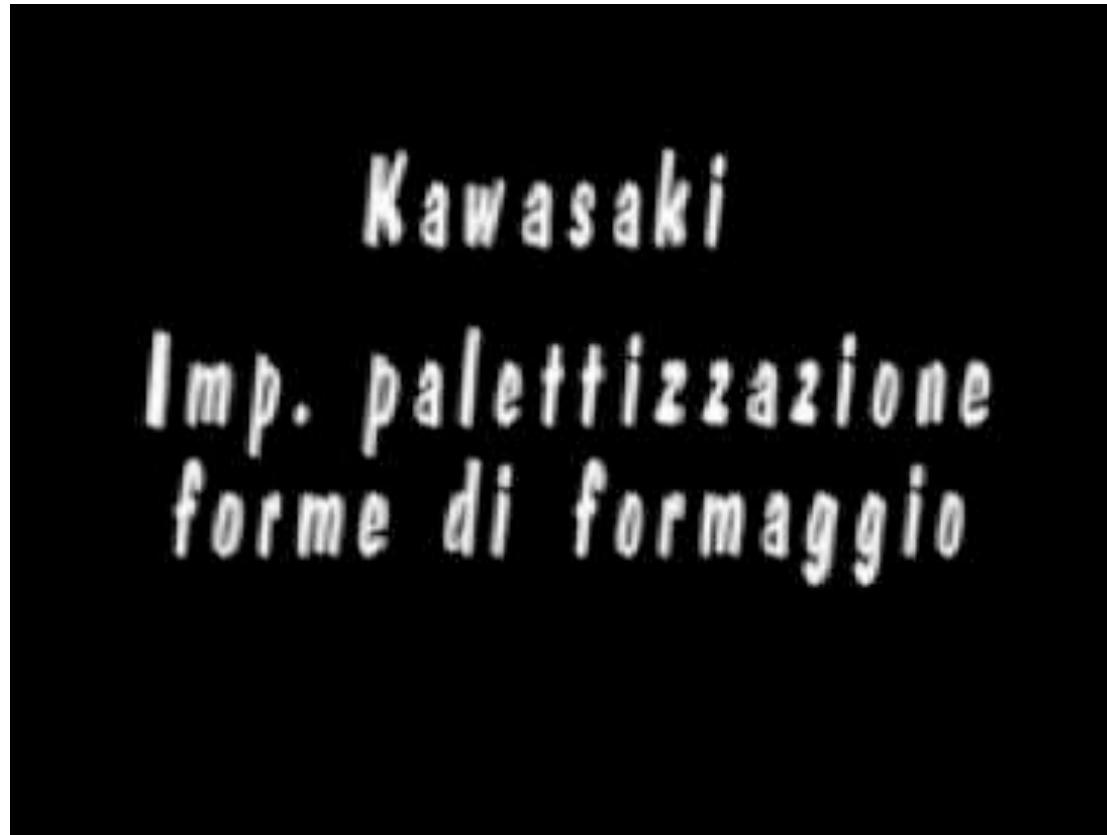
Palletizing



pallet = a portable platform on which goods can be moved, stacked, and stored



Palletizing of cheese forms

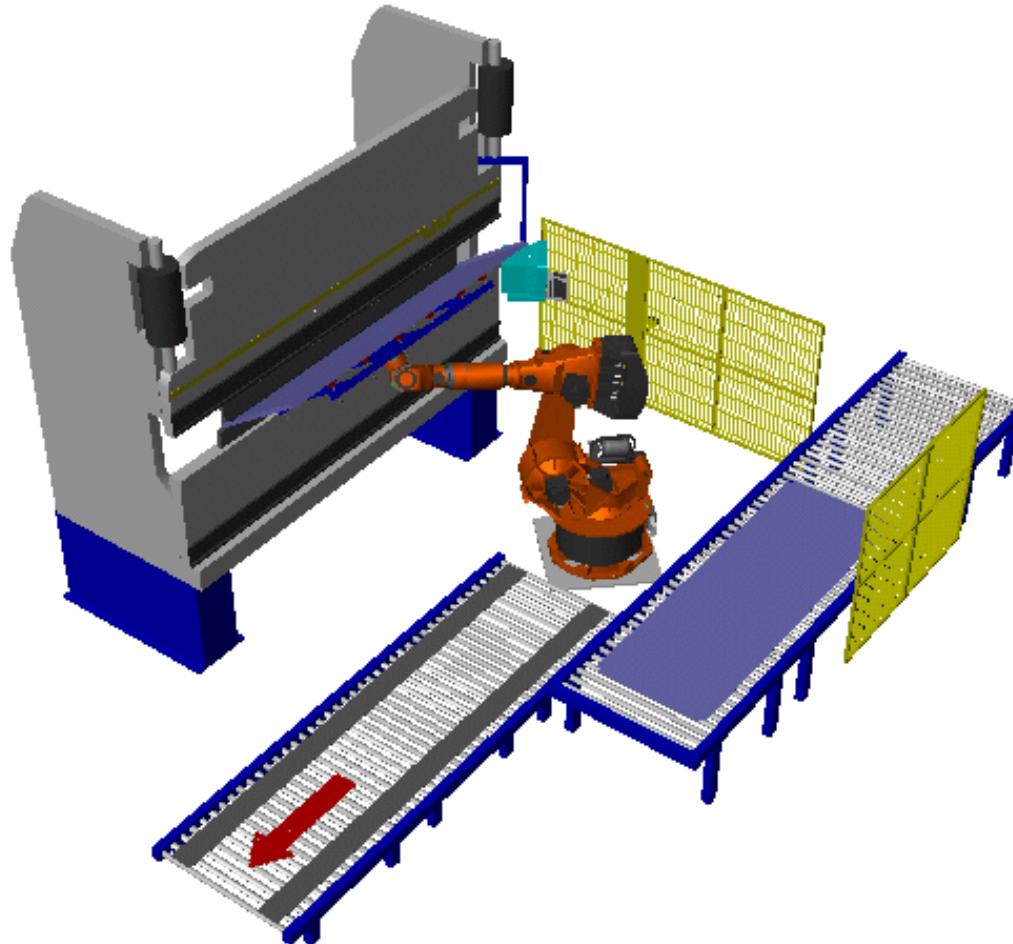


video

using Kawasaki robots (courtesy of Effedue Engineering)



Folding

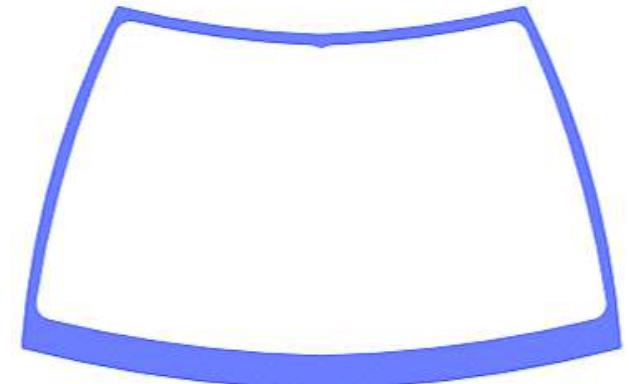
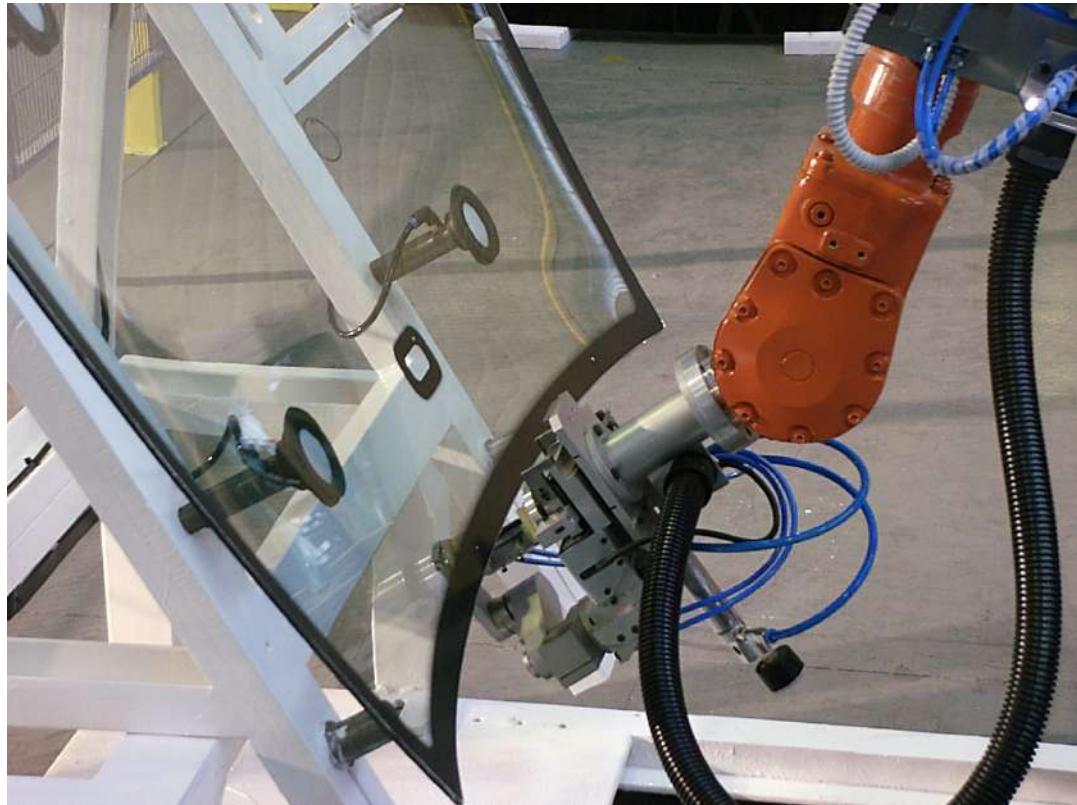


with loading of sheets under the press



Deburring

- car windshields may have large manufacturing tolerances and a sharp contour profile



- the robot follows a given predefined Cartesian path
- the contact force between cutting blade and glass must be feedback controlled
- deburring robot head mounts a force load cell and is pneumatically actuated



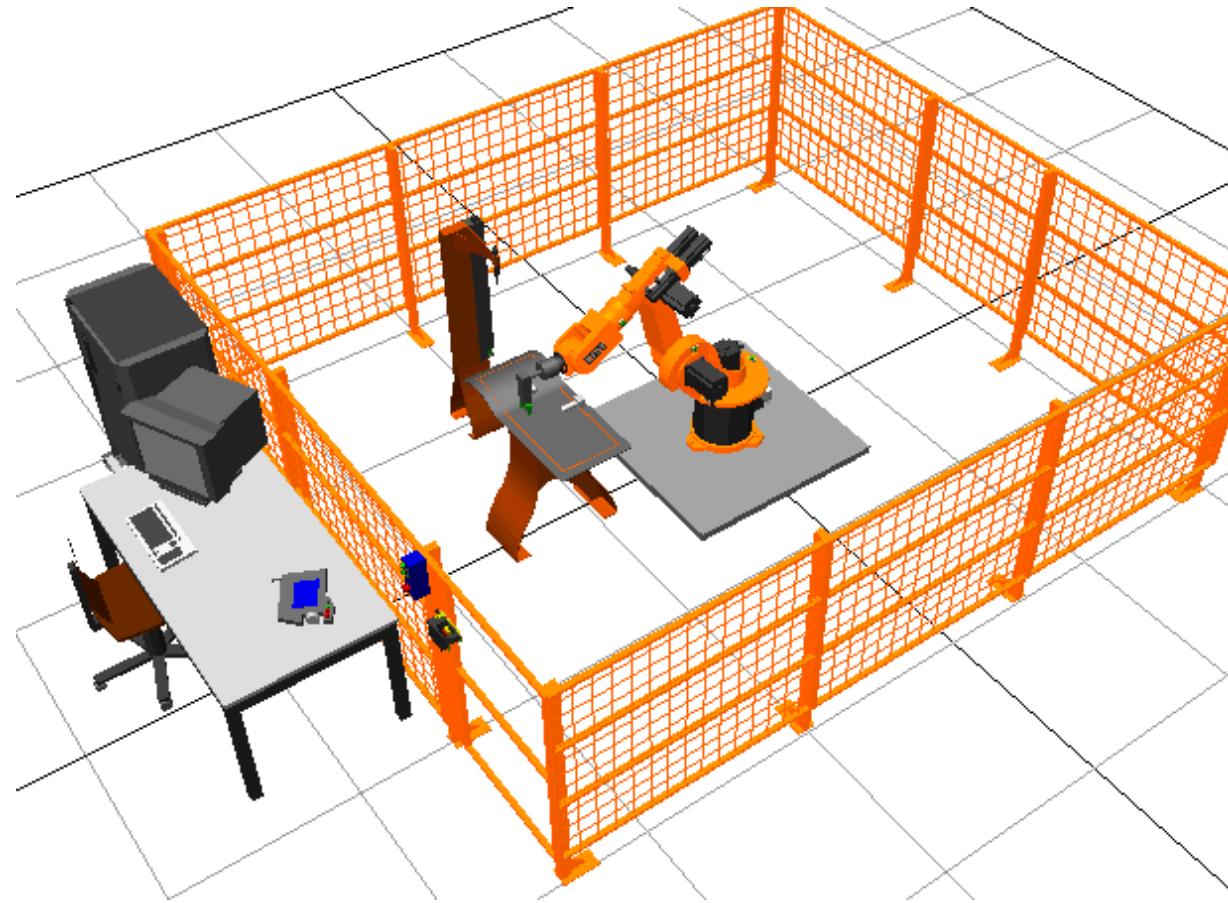
Deburring center



deburring center for steel parts
using Comau SMART NJ 110-3.0/foundry robot (courtesy of Adami srl)



Off-line robot workstation



articulated robot in metal surface finishing operation



Safety in robotic cells



commercial [video](#) from ABB
SafeMove (2008) cell monitoring system: no fences!



Robot manipulator kinematics



Kuka 150_2 S2000
open kinematic chain
(series of rigid bodies
connected by joints)



Comau
Smart H4
closed kinematic chain



Fanuc
F-200iB
parallel kinematics



SCARA-type robots



Mitsubishi RP
(repeatability 5 micron,
payload 5 kg)



Mitsubishi RH
(workspace 850 mm,
velocity 5 m/s)



Bosch Turbo

SCARA (Selective Compliant Arm for Robotic Assembly)

- 4 degrees of freedom (= joints): 3 revolute + 1 prismatic (vertical) axes
- compliant in horizontal plane for micro-assembly and pick-and-place



Adept Cobra i600



video

fastest SCARA robot for pick-and-place tasks!



Other types of robots



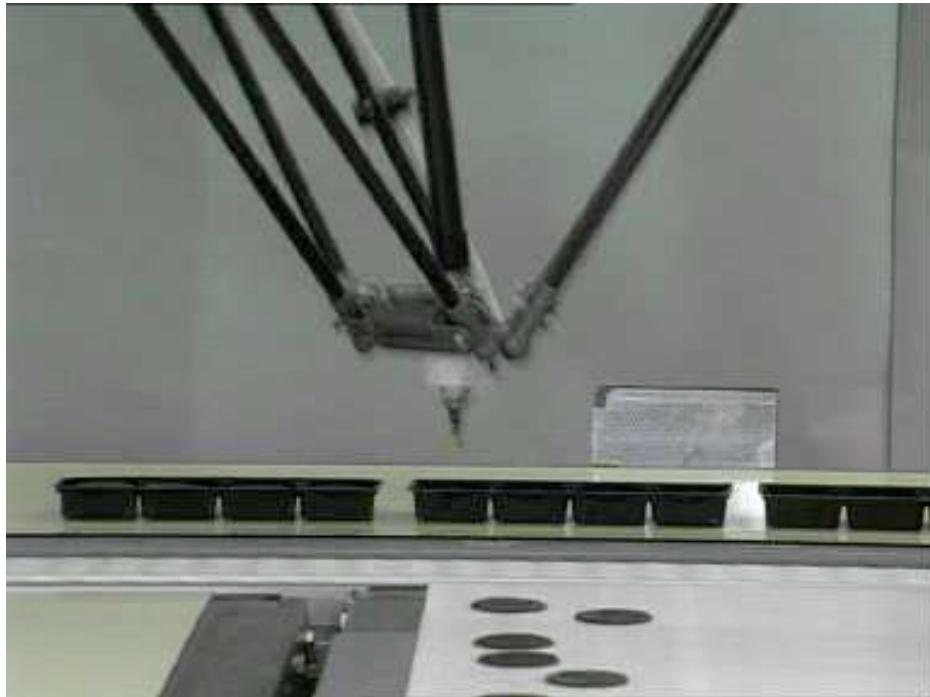
Comau Mast
gantry robot
(payload up to 560 kg)



ABB Flexpicker
(150 pick-and-place
operations/minute)



Chocolate packaging with lightweight parallel robots



test [video](#) with
ABB Flexpicker

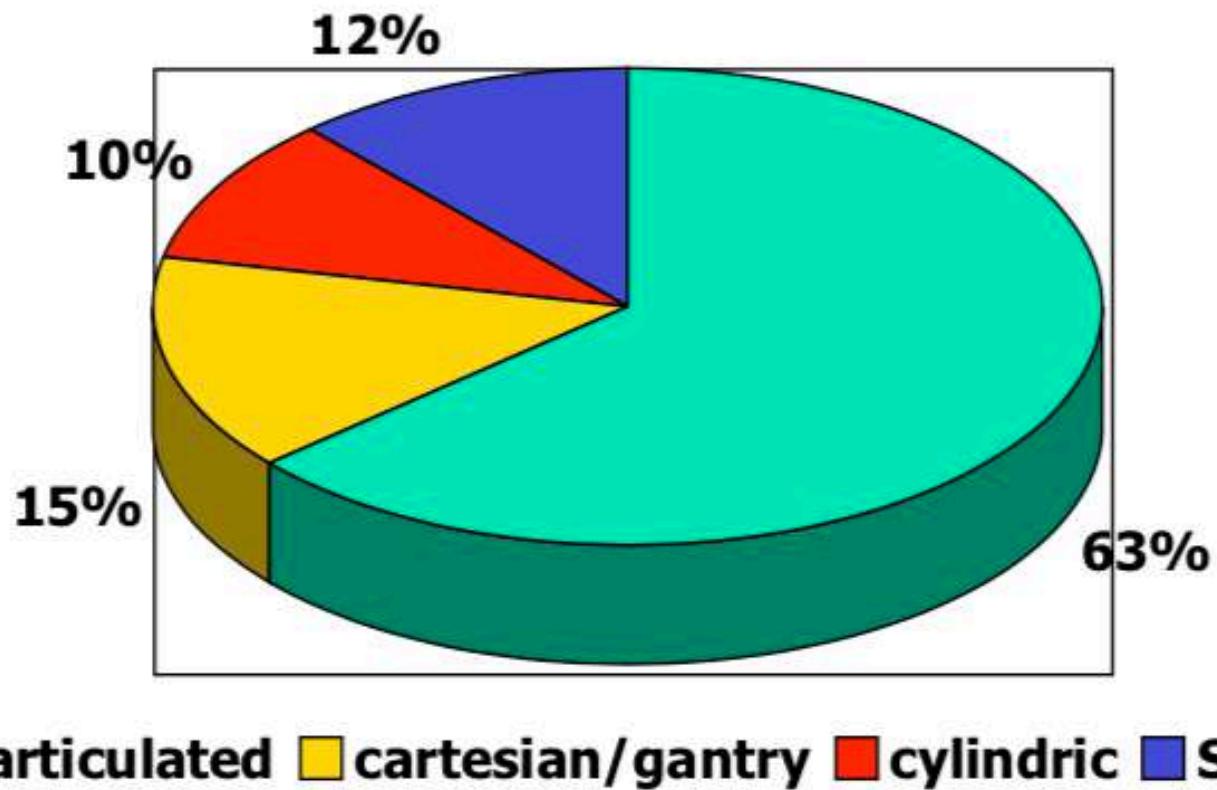


[video](#) with
Adept Quattro s650



Distribution by robot type

of kinematic configuration



for 59600 articulated robots installed back in 2004
(90% of all robots installed in America, 74% in Europe, only 49% in Asia)



Robot data sheet



Fanuc
R-2000i/165F

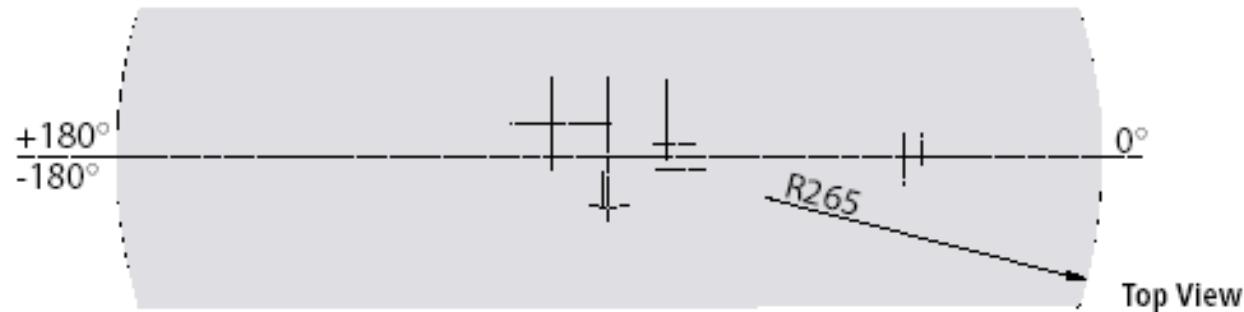
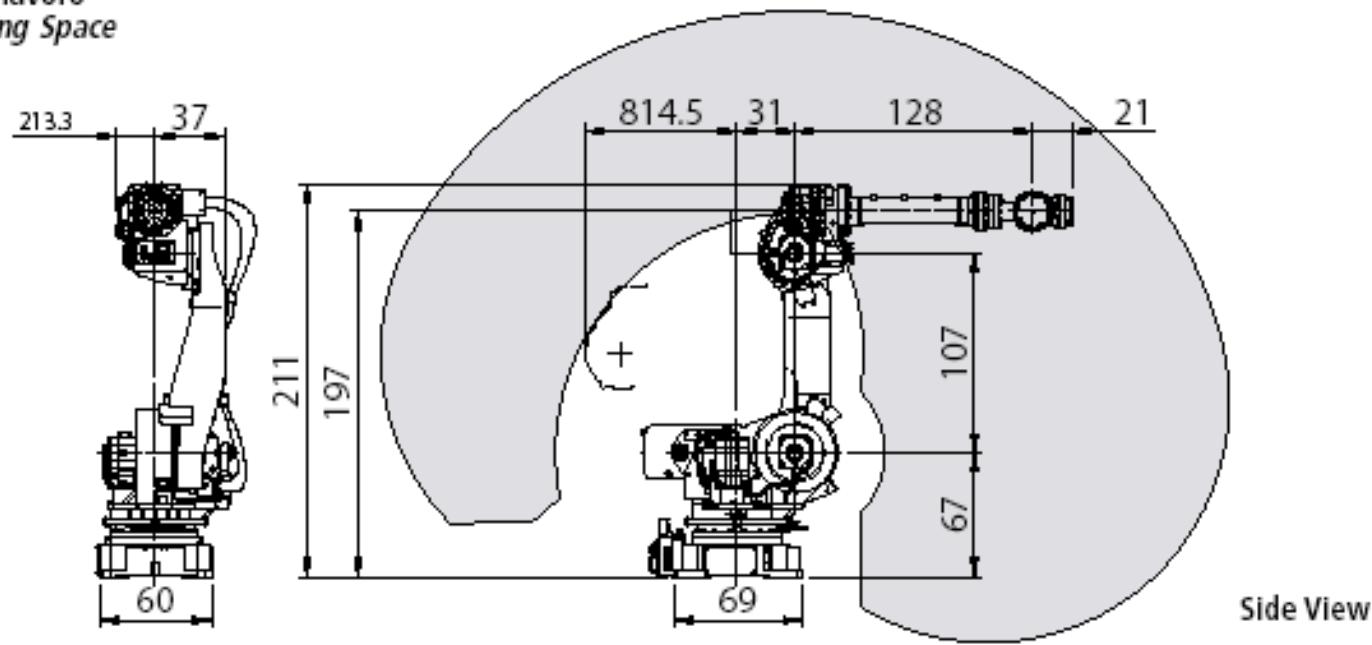
Specifiche tecniche

Voce	R-2000i/165F		
Tipo	Articolato		
Assi controllati	6 assi (J1, J2, J3, J4, J5, J6)		
Installazione	A pavimento		
Area di lavoro (Velocità massima)	Rotazione asse J1	360° (105°/s)	
	Rotazione asse J2	135° (105°/s)	
	Rotazione asse J3	361,8° (105°/s)	
	Rotazione asse J4	720° (130°/s)	
	Rotazione asse J5	250° (130°/s)	
	Rotazione asse J6	720° (210°/s)	
Carico massimo al polso	165 kg		
Momento di carico max. al polso (Nota 1)	Asse J4	94 kgf.m	921 Nm
	Asse J5	94 kgf.m	921 Nm
	Asse J6	47 kgf.m	461 Nm
Momento di inerzia max. al polso	Asse J4	800 kgf.cms ²	78,4 kgm ²
	Asse J5	800 kgf.cms ²	78,4 kgm ²
	Asse J6	410 kgf.cms ²	40,12 kgm ²
Tipo di azionamento	Motori elettrici AC		
Ripetibilità	± 0,3 mm		
Peso	1.210 kg		
Ambiente installazione	Temperatura ambiente: Normale: Breve (in un mese) Vibrazioni	0-45° C ≤ 75% ≤ 95% 0,5 G max.	



Workspace

Area di lavoro
Operating Space



Visualization of workspace and mobility



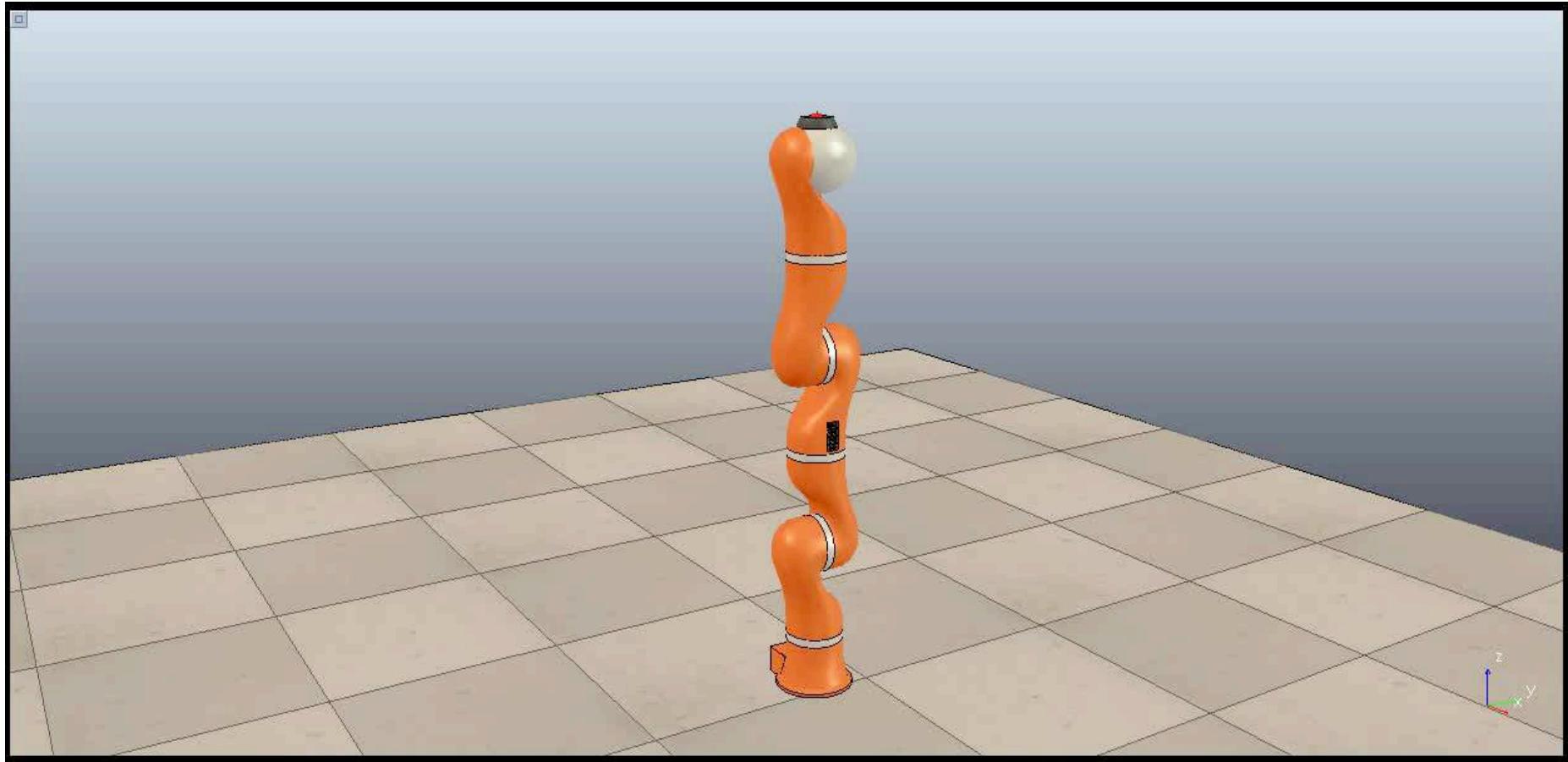
[video](#)

kinematic simulation of a 6-dof Comau robot (all revolute joints)

Visualization of workspace and mobility



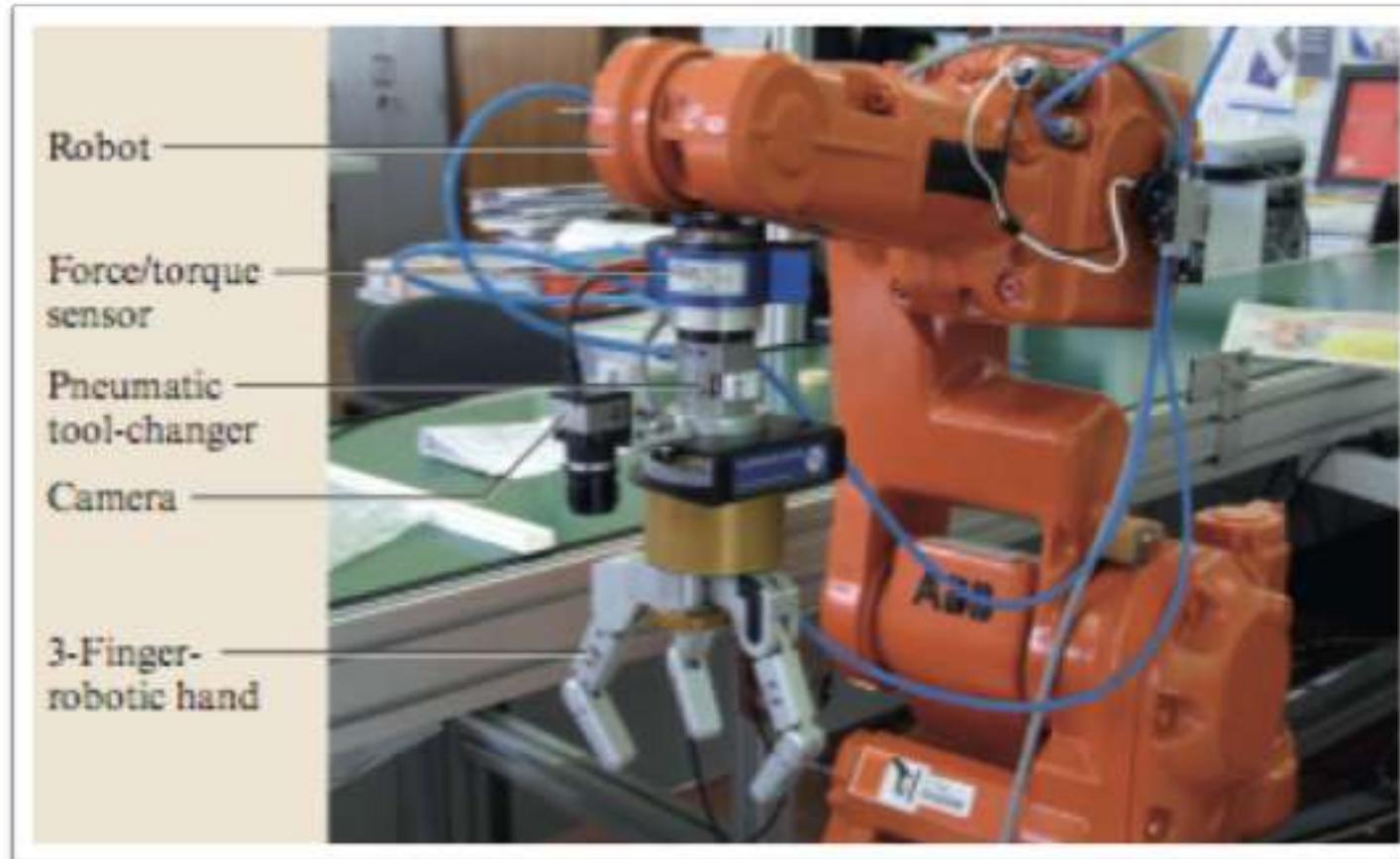
[video](#)



V-REP simulation of the 7-dof KUKA LWR4+ robot (all revolute joints)

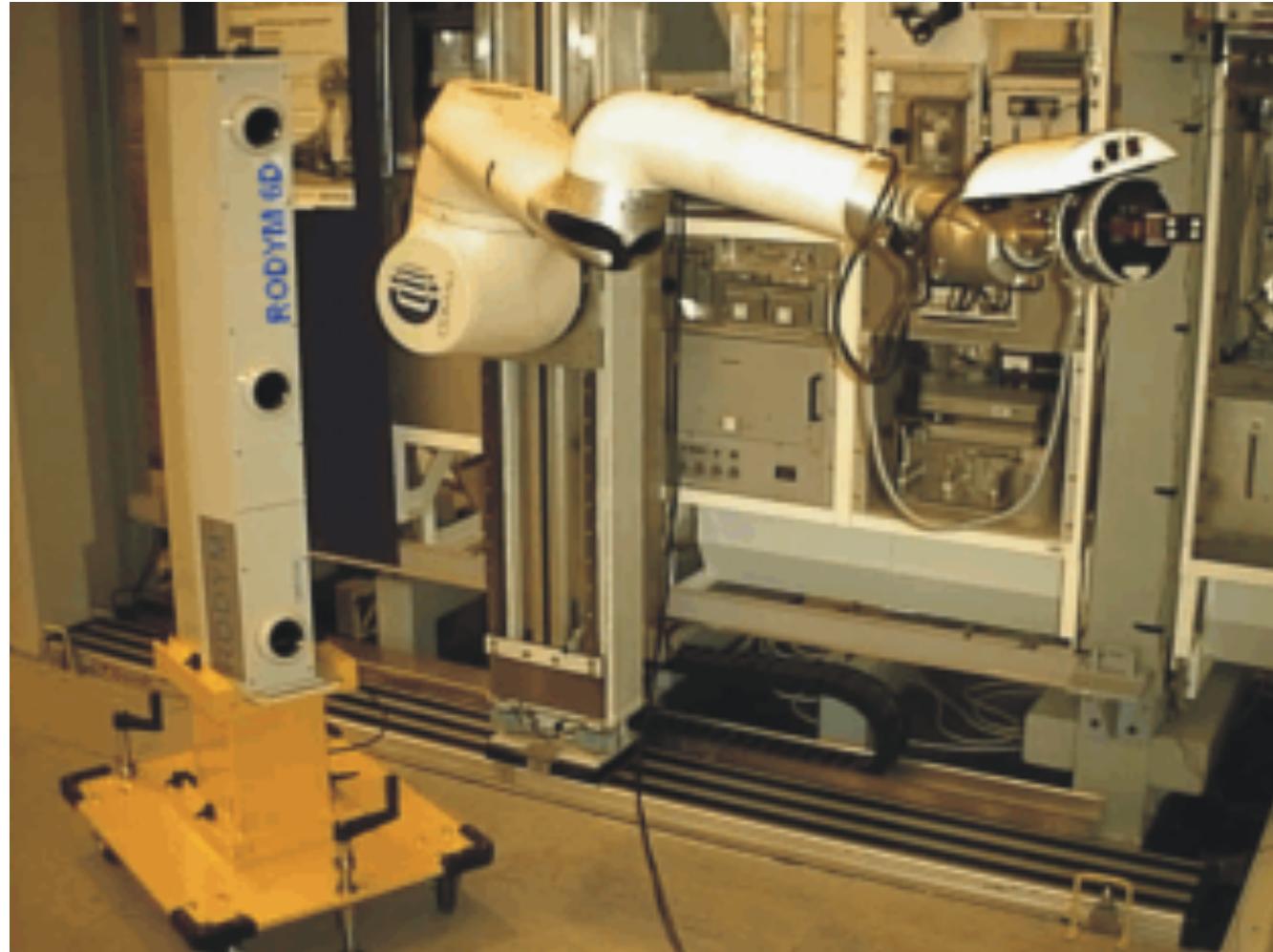


Robot end-effector sensors and tools





Calibration of robot kinematics





Man-machine interface



- teach-box pendant used as robot programming interface

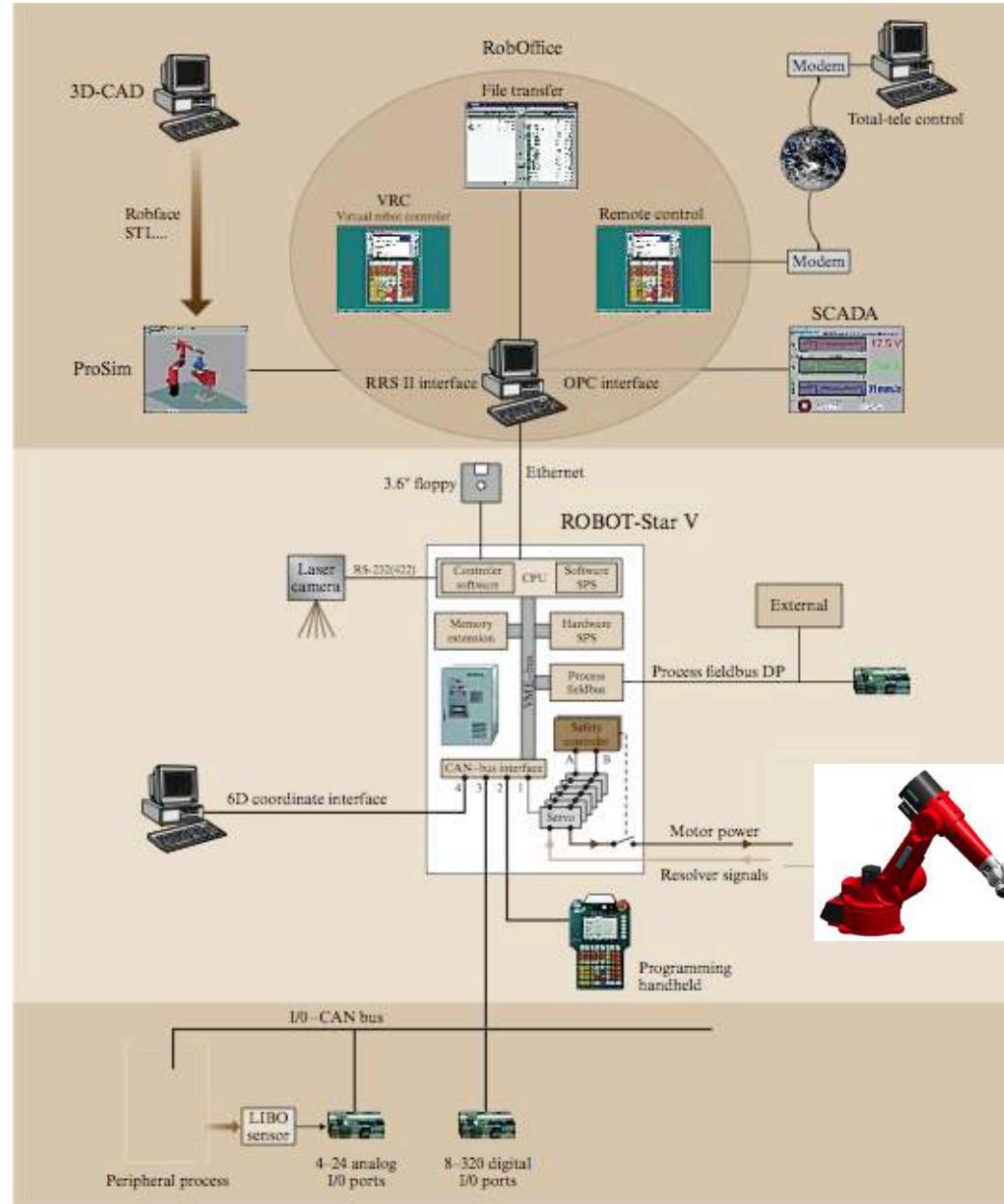


- cabinet with power electronics for robot supervision and control



Programming and control environment

control modules
and interfaces
(Reis Robotics)





Motion programming and scaling



commercial [video](#) from ABB
TrueMove & QuickMove fast motion control performance



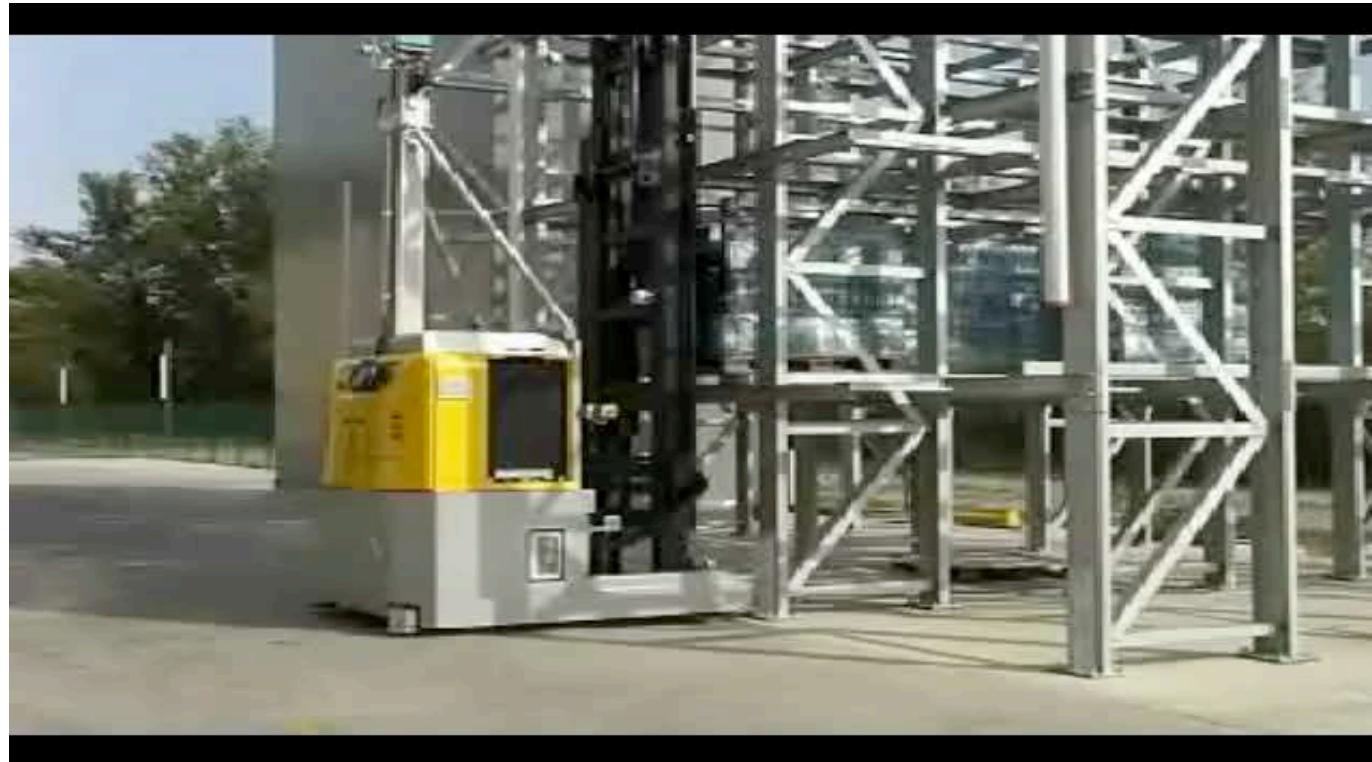
Mobile base robots in industry



- **AGV** (Automated Guidance Vehicles) for material and parts transfer on the factory floor: wire- or laser-driven along predefined paths



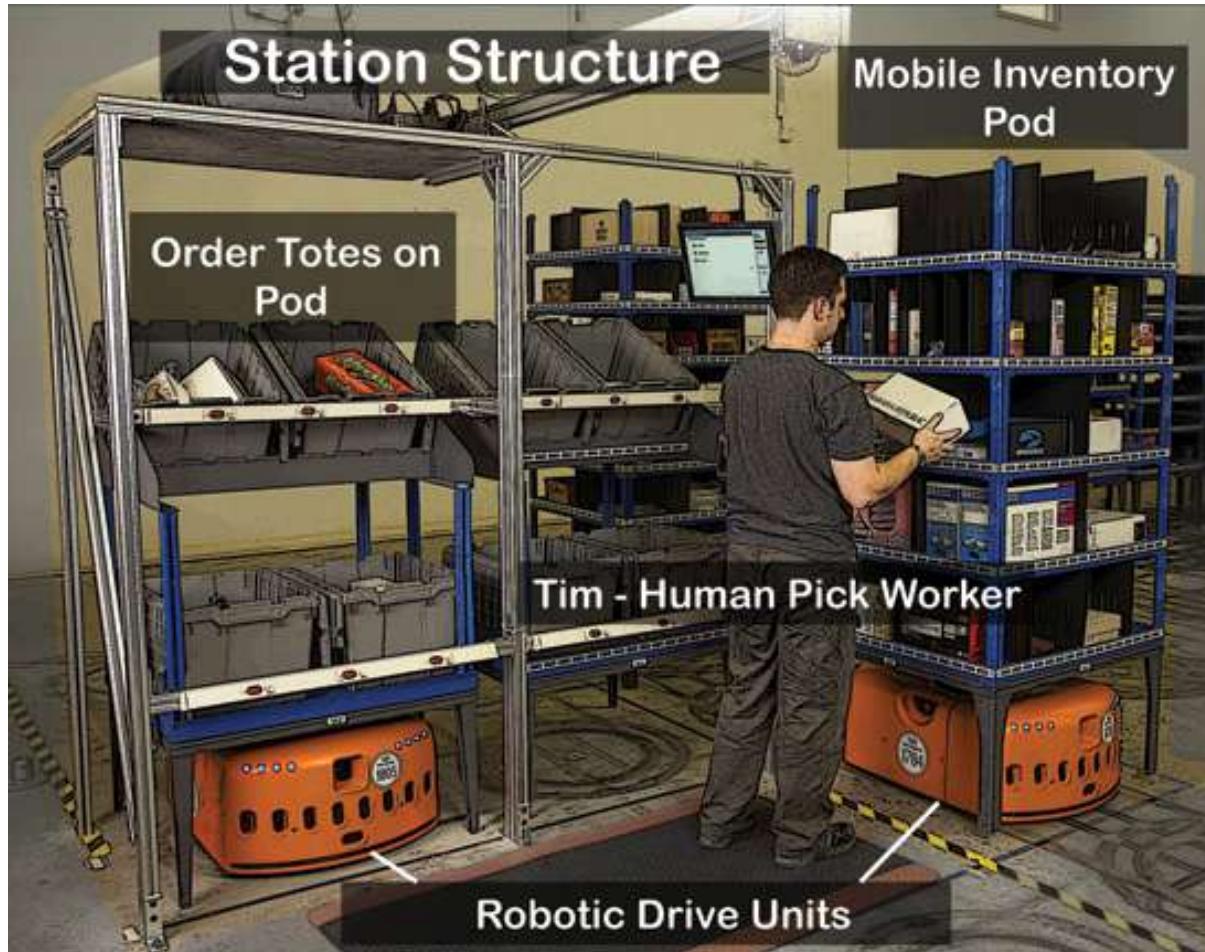
Lifting AGV for warehouses



video by Elettric80



Kiva Systems



company acquired in 2012 for \$775 million by Amazon ([store automation](#))



Intelligent AGV in factories



commercial [video](#) of ADAM mobile robot (RMT Robotics)



What's next in industrial robotics?

changing nature of manufacturing and work

- shift from high volume/low mix to low volume/high mix is having a profound impact on manufacturing
- many industries are facing acute shortages of skilled labor
- quicker return-of-investment (ROI) of automation and rising wages are eventually discouraging labour arbitrage
- increased focus is being placed on workplace safety



Source: Steven Wyatt (IFR). "Today's trends, tomorrow's robots!" Frankfurt, 27 September 2017



What's next in industrial robotics?

addressing some real facts opens huge opportunities

	The Trends	The Challenges	The Enablers
	Low volume high mix	Automation complexity and unpredictability	Collaborative automation for greater flexibility
	Shorter cycles, faster launches	Shop floor disruptions and high engineering costs	Better software for engineering efficiency
	Increased need for automation and scalability in SMEs	Lack of robot integration and programming expertise	Easier to use robots with more intuitive programming
	Rising cost of downtime	Higher lifetime TCO due to increase in planned downtime	Advanced analytics and services for greater reliability
	Increased and sporadic human intervention	Lost productivity to maintain safety	Collaborative automation to maintain safety and productivity

**answers to these challenges lie in
Simplification, Digitalisation, and Collaboration**



What's next in industrial robotics?

Simplification (critical for SME, but also for large global manufacturers)

- robots **easier** to install, program (with open source) and operate will unlock entry barriers to the large market of small and medium enterprises (SMEs)
- trend towards having production closer to the end consumer is driving the importance of **standardisation** & consistency across global brands

Digitalisation (Big Data allows taking better decisions on factory operations)

- "Industry 4.0", linking the real-life factory with a **virtual/digital** one, will play an increasingly important role in global manufacturing
- **vision and sensing** devices, coupled with analytics platforms, will pave the way for new industry business models
- IoT/AI/Machine Learning will drive many robotics developments in coming years

Collaboration

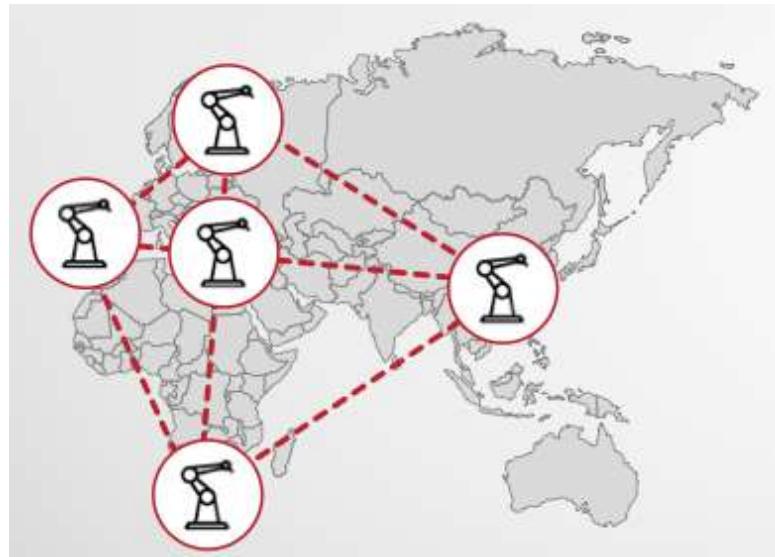
- collaborative robots are shifting the traditional limits of "what can be automated?"
- **collaborative** robots increase manufacturing flexibility as 'low-volume, high-mix' becomes the main standard
- collaboration is also about productivity with increased physical and cognitive **human/robot interaction**



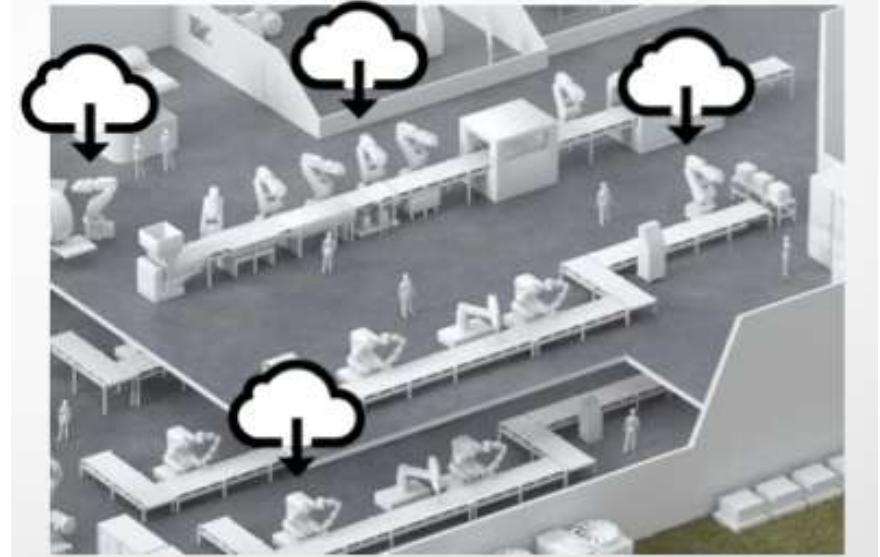
What's next in industrial robotics?

"connected" future of robotics

self-optimizing production



self-programming robots



- robots doing the same task connect across all global locations so performance can be easily compared and improved
- robots automatically download what they need to get started from a cloud library and then optimize through "self-learning"

**connected and collaborative robots will enable
SMART Manufacturing for both SMEs & Global Enterprises**



Franka Emika robot

... one possible example (dated 2016)

[video](#)





Robotics 1

Service robotics

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Some application domains

- extreme environments
 - space
 - underwater
- medical robotics
 - assistive
 - rehabilitative
 - surgical
- home cleaning
- agriculture
- lawn mowing
- food industry
- mine exploration
- de-mining
- civil and naval construction
- automatic refueling
- museum guide
- fire fighting
- inspection and surveillance
- emergency rescue
- entertainment
- humanoids

professional & personal service robots



Service robots on the market!



Bluebotics Esatroll - Paquito 2.0
logistics in factory floor



Yujin GoCart2
elderly and health care



Cyberdyne HAL
exoskeleton for walking



Lely Vector automated feeding

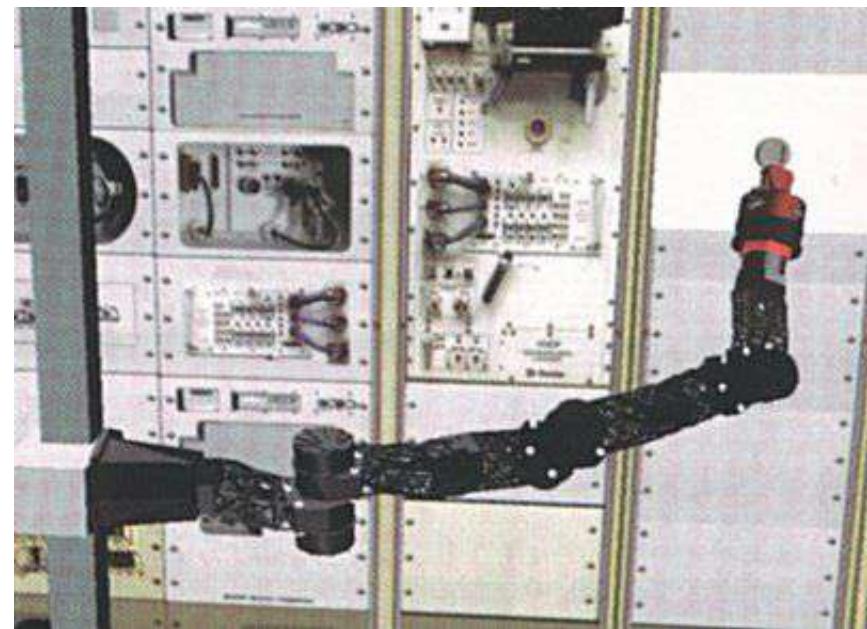
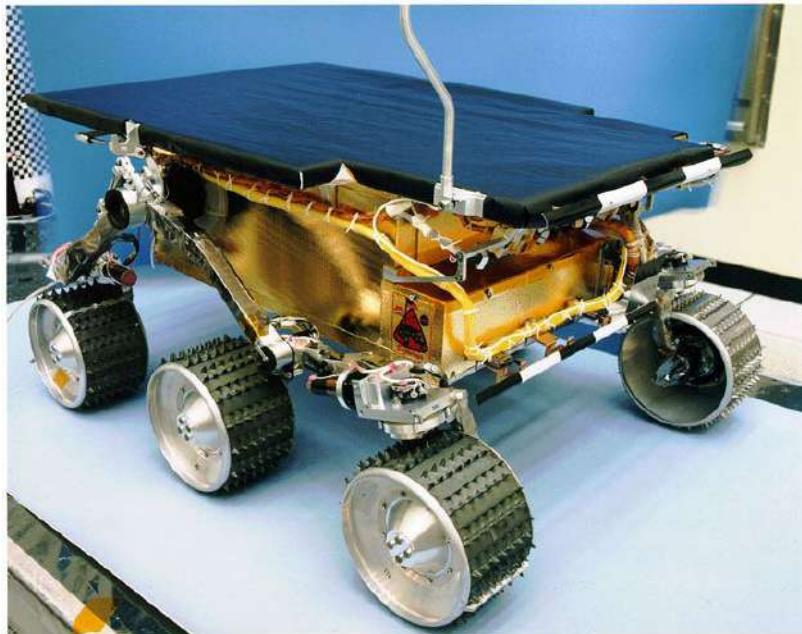


Vorwerk
vacuum cleaner



Thymio educational
mobile robot

Space robotics



- NASA *Sojourner*, first robot to explore Mars in 1997
- DLR *Rotex* robot arm in a set of experiments of the Spacelab-D2 mission on the Columbia shuttle in 1993



Space robotics

video



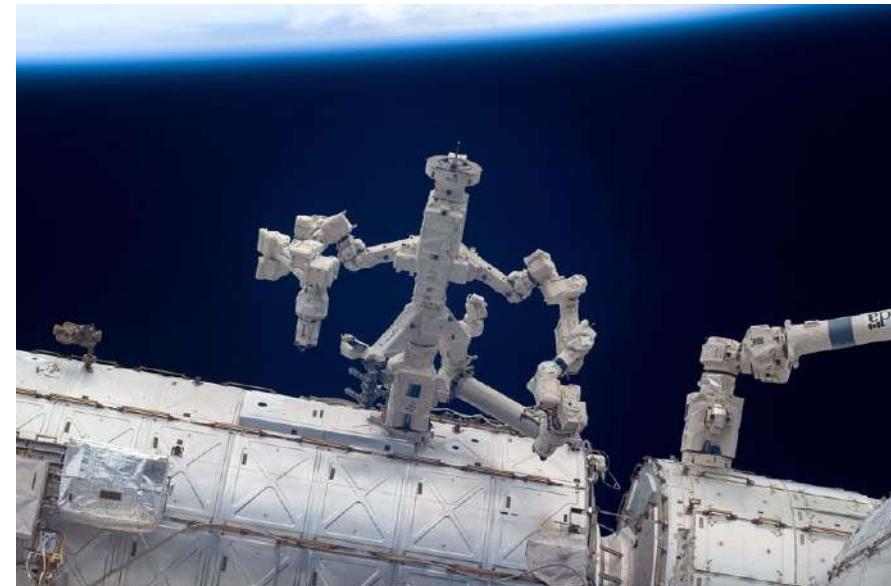
wheels untrapping
on sandy soil

assembly tasks and
catching floating objects
with *Rotex*



Space robotics

- robotic arm *SSRMS (Canadarm)* in operation on the Space Shuttle, with outstretch of about 17 meters



- service manipulator *SPDM (Dextre)* on the International Space Station (ISS), mounted on a supporting rail and used to refuel satellites



Robots on ISS

video



Canadarm2 delivering Destiny Lab
from Space Shuttle Endeavour to ISS
(about 2007)

video



service manipulator and Robonaut
on the ISS (artistic views)



Underwater robotics



- Odyssey-IV (MIT)



- Odin-III, omni-directional
(University of Hawaii)

- typically actuated by thrusters
(directional forces on the tail)
- cannot translate sideways
("maneuvers" are necessary)



- ROMEO in Antarctica (CNR,
Automazione Navale, Genova)



Underwater robotics



Ansaldo underwater arm
performing a cable hooking task
(SAUVIM project)

video

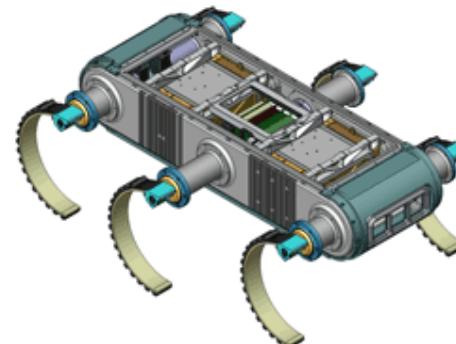
video

UBC Gavia underwater robot
(University of British Columbia)





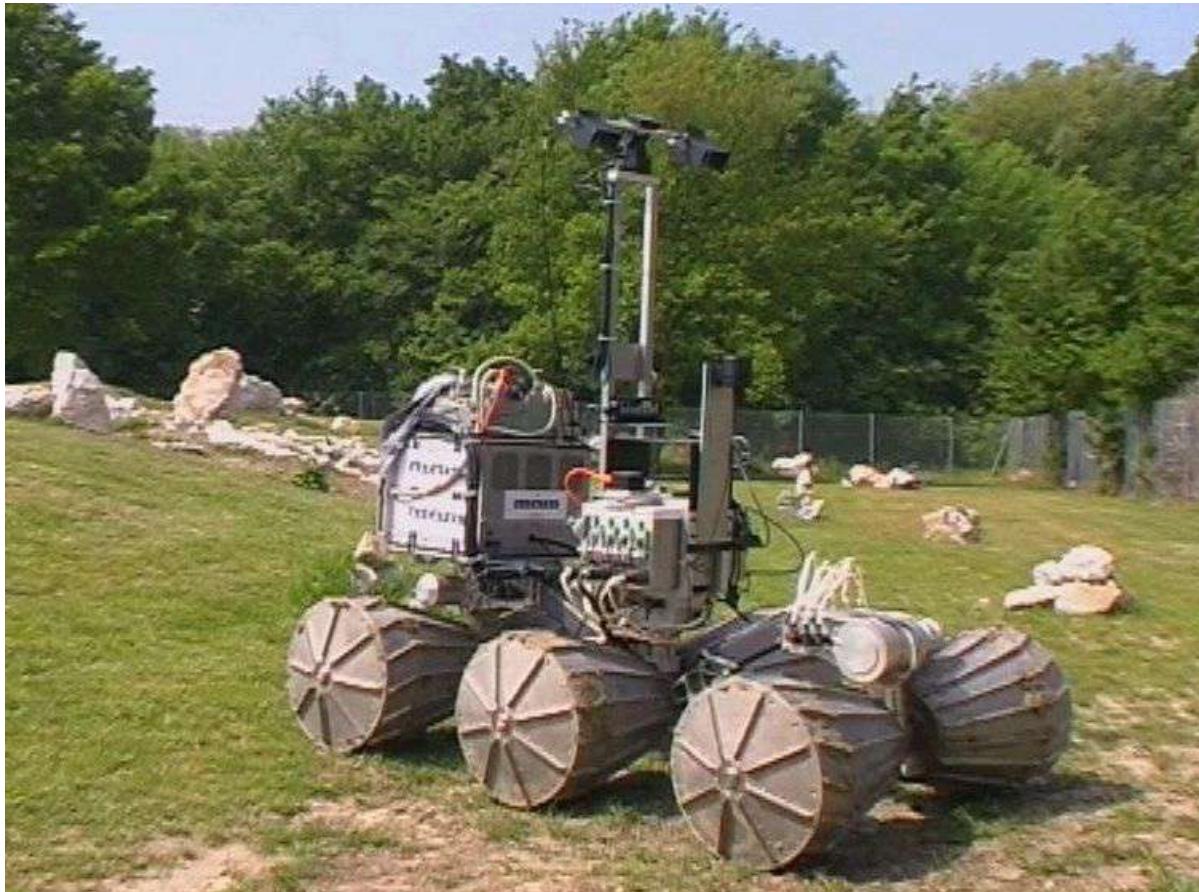
Underwater robotics



- Aqua robot, amphibious robotic vehicle (McGill University)
- size and weight: 50x65x13 cm, 18 kg
- locomotion: through six independently actuated flippers
- maximum depth: 37 m
- sensors: two cameras (front/back), acoustic sensor for localization (sonar), tri-ocular sensor (structured light)
- power source: 48V lithium battery



Outdoor exploration



- the *LAMA* robot at CNRS-LAAS (Toulouse) is a french-russian cooperation



Volcanology

video



video



RoboVolc vehicles on the surface
of the Etna volcano:
wheeled and tracked robots
(University of Catania, 2003)



De-mining



- teleoperated mobile robot on tracks used by the police for bomb disposal

- PEMEX lightweight anti-personnel mine detector (EPFL, Lausanne)
- weight: 16 kg, max 6 kg for wheel
- two 70 W DC motors (vel 2 m/s)
- oscillating sensorized head

Medical robotics patient aid



- *MOVAID* project for the aid of disabled people in home activities (Scuola Sup Sant'Anna, Pisa)
- deambulation support system
PAM-AID (Trinity College, Ireland)



MOVAID project

video



video



domestic activities using the 8R *Dexter* arm

Medical Robotics

rehabilitative



- robotic arm with shoulder and elbow having full mobility and with a gripper hand (Pittsburgh University)
- in tests on monkeys (with immobilized upper limbs), motion commands sent to the arm by the central nervous system (brain) are measured by a set of electrodes and used to command the robotic arm

Medical robotics

rehabilitative



- commercialized by Ossur (Iceland)
- a prosthesis sensorized at the knee (angle and force), capable of processing sensor data and of extracting a gait model of the user, so as to adapt its dynamical behavior (knee motion and stiffness)

Medical robotics rehabilitative

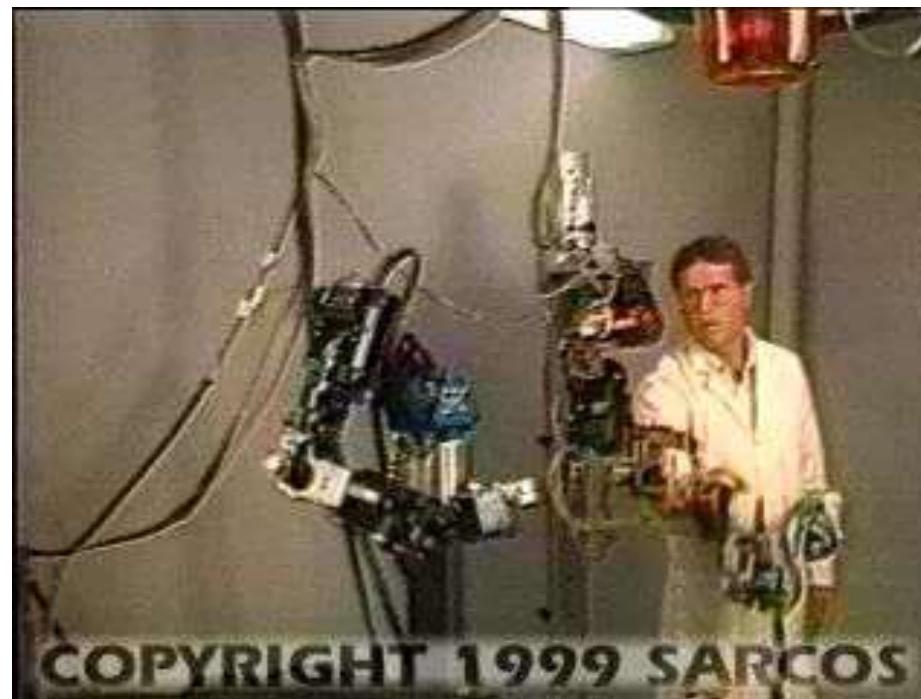


- "RUPERT" Robotic Upper Extremity Repetitive Therapy (Arizona State University + Kinetic Muscles, Inc.)
- sustains the human arm with pneumatic muscles (McKibben actuators)
- it can be programmed for the execution of cyclic exercises of rehabilitation



Exoskeletons

video



SARCOS master-slave for teleoperation

Medical robotics hospital and nursing



video



- *HELPmate* mobile robot (USA) works in hospitals as auxiliary personnel
- user interface of the *Care-O-Bot* robot nurse (IPA Fraunhofer, Germany)



Surgical robotics



overview of the operating room



command station



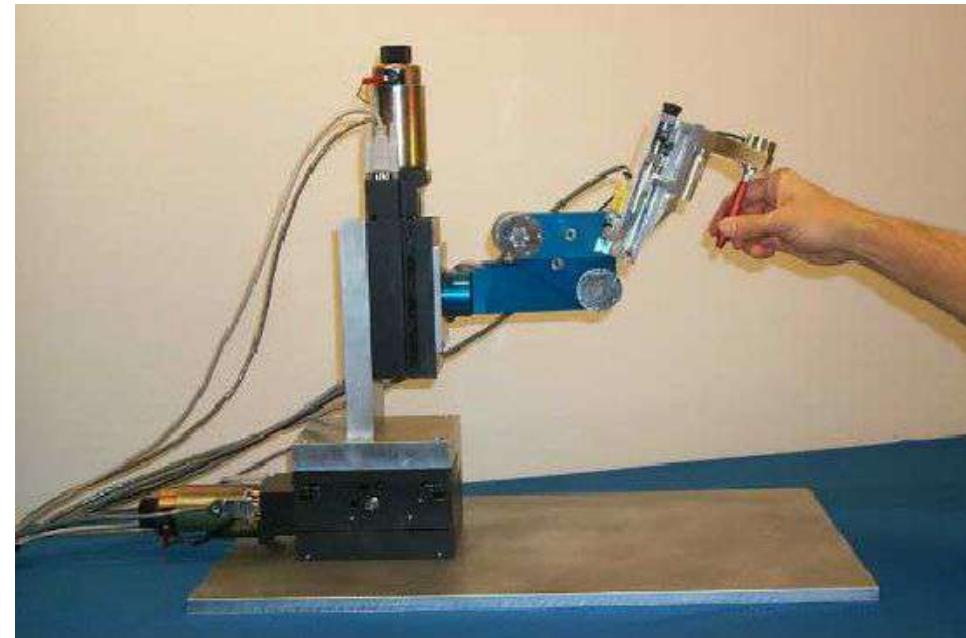
(haptic) interface

- da Vinci[©] system (Intuitive Surgical Inc.)
[see the course “Medical Robotics”]



Surgical robotics

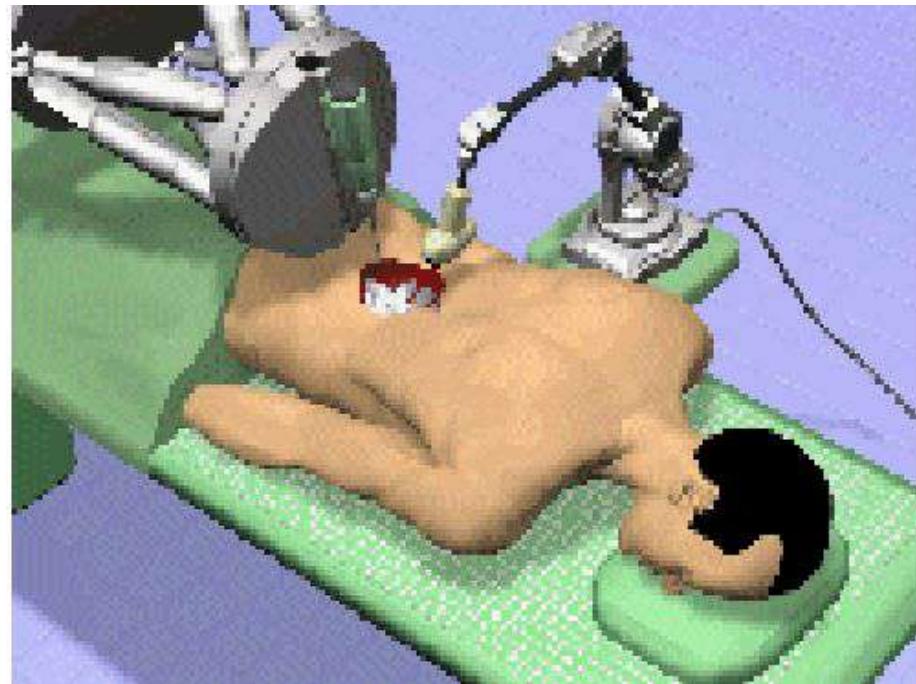
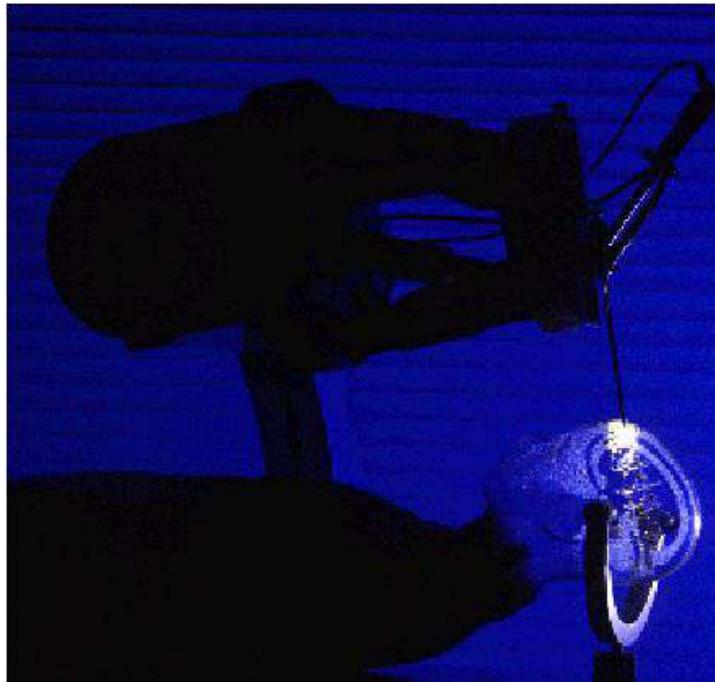
- *Robodoc* by Integrated Surgical Systems (USA) was used first for orthopedic surgery (ankle replacement)



- *Steady-Hand* force-assisted system (Johns Hopkins Univ) improves accuracy and repeatability of surgeons allowing task-driven compliance



Surgical robotics



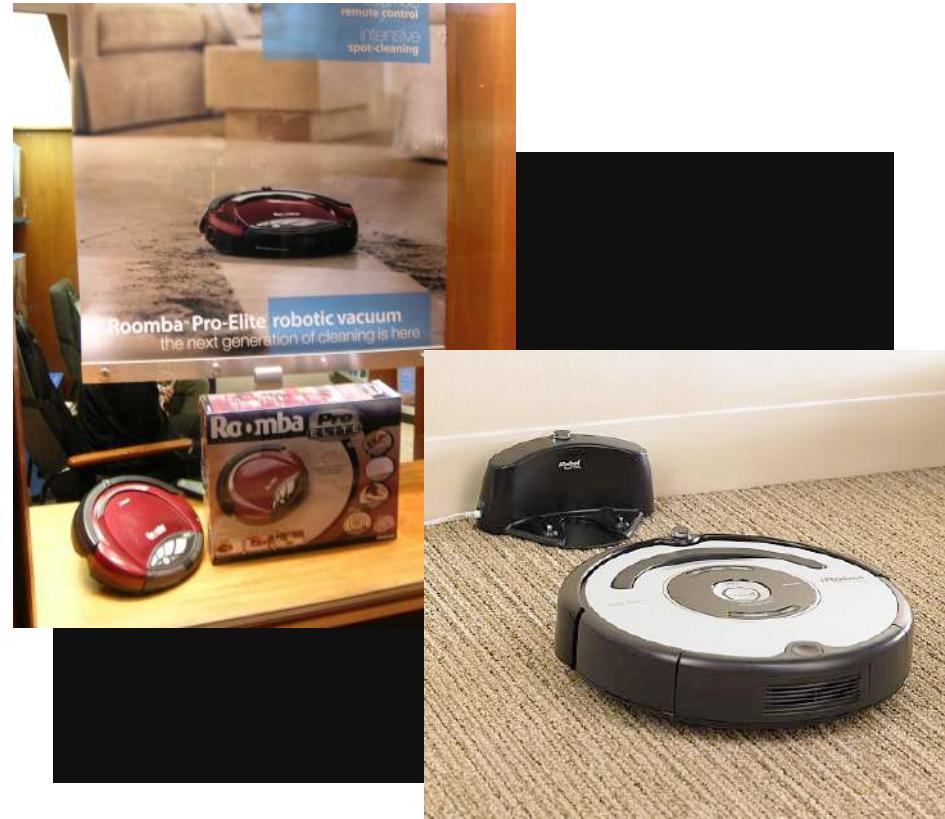
- emulation of a laser brain surgery operation and graphic rendering of a surgery intervention on the spinal cord
patient is first “mapped” off-line by a series of CAT scans;
data are then “localized” in the actual operation field
(IPA Fraunhofer)



Home cleaning



- vacuum cleaner robot
Trilobite by Electrolux (Sweden)



- commercial [video](#)
iRobot *Roomba 560* (USA)
– now available everywhere!



Cleaning robot contest



- competition among robot vacuum cleaners
in home environments (IROS'02, Lausanne)



Cleaning of external surfaces



- *Skywash* cleans civil airplane bodies and is “the largest robot worldwide” (AEG/Dornier/FhG-IPA/Putzmeister)
- a robot prototype for cleaning large glass windows of civil buildings



Lawn mowers



video

- *Automower* autonomous robot by Husqvarna (Sweden) has low power consumption (biocut) and solar recharge



Food industry



- *Ulices* robot by IMT (Germany) aligns 10000 sausages per hour



video

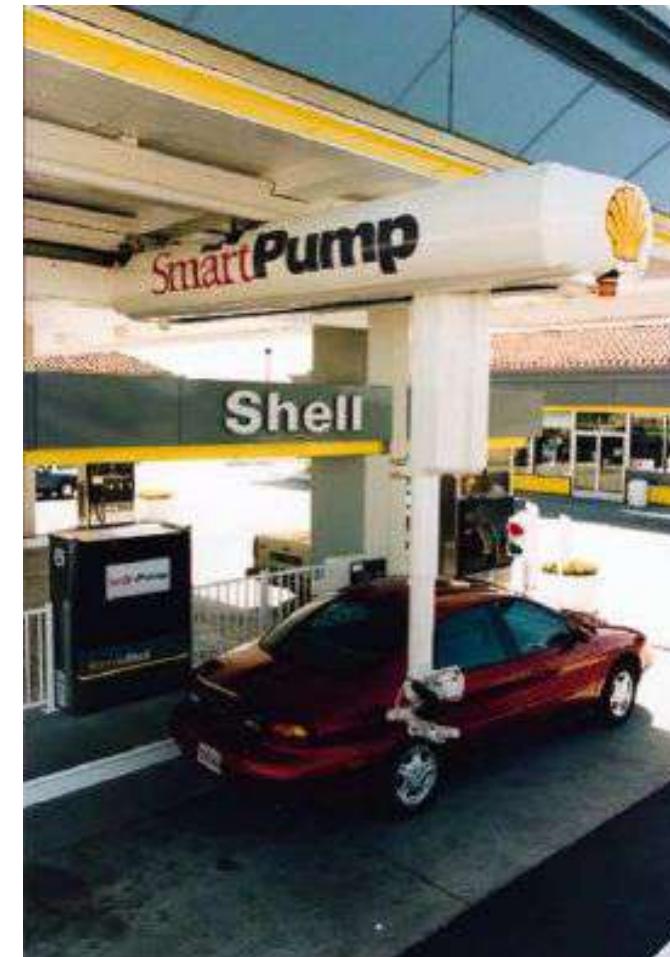
- *AdeptOne* SCARA robot with 4-sausage gripper



Automatic refueling



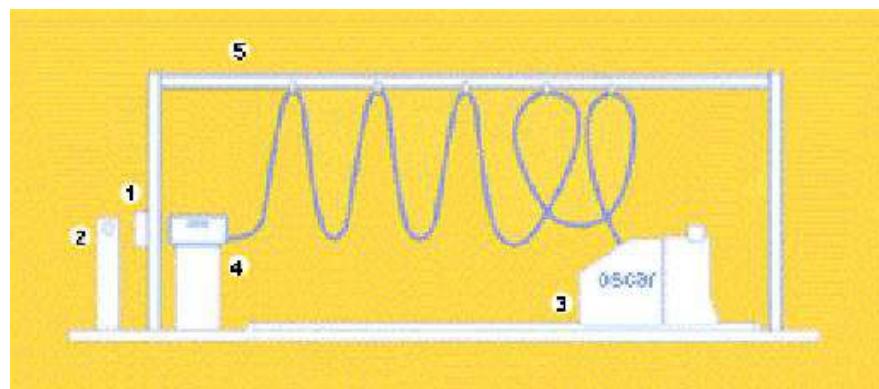
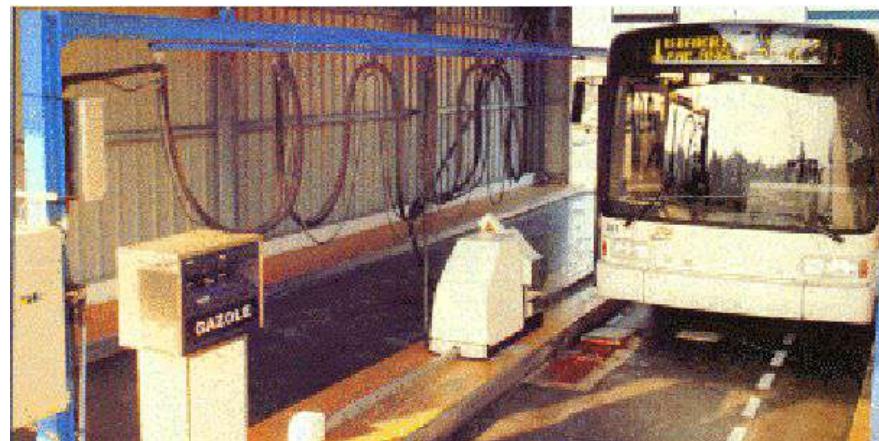
- cooperation of Reis Robotics, Mercedes, BMW, and IPA Fraunhofer



- *Smart Pump* system (USA)



Automatic refueling



- OSCAR robot (France) for gasoline refuel of flees of transportation busses



Automatic refueling



a “kit” is available for all car models:
tank cap, transponder, pipe union

Autofill system in two tank stations of
OK (Mörgby, Sweden) and BP (USA)



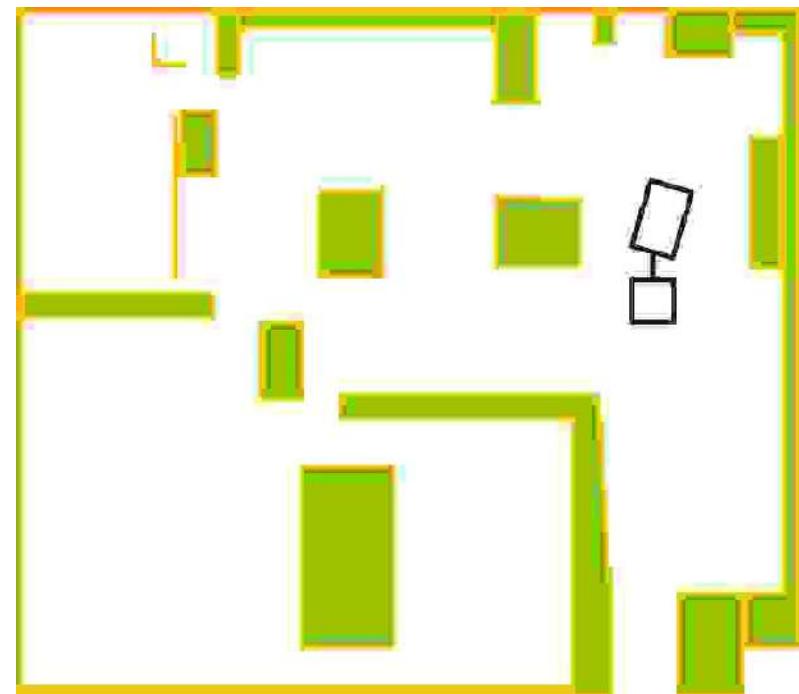
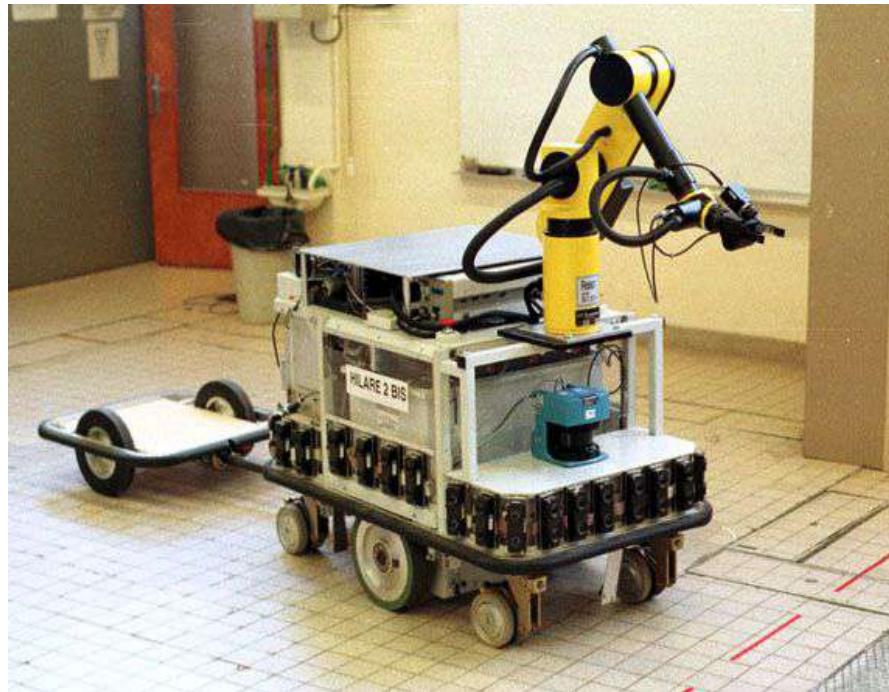
Inspection and surveillance



- 6-dof *Puma* arm mounted on the *Nomad XR400* (multiple steering wheels, synchro-driven)
 - 5-dof *Scrbot* arm mounted on a *ATRV-JR* (fixed wheels, skid-steering vehicle)
- two examples of
mobile manipulators



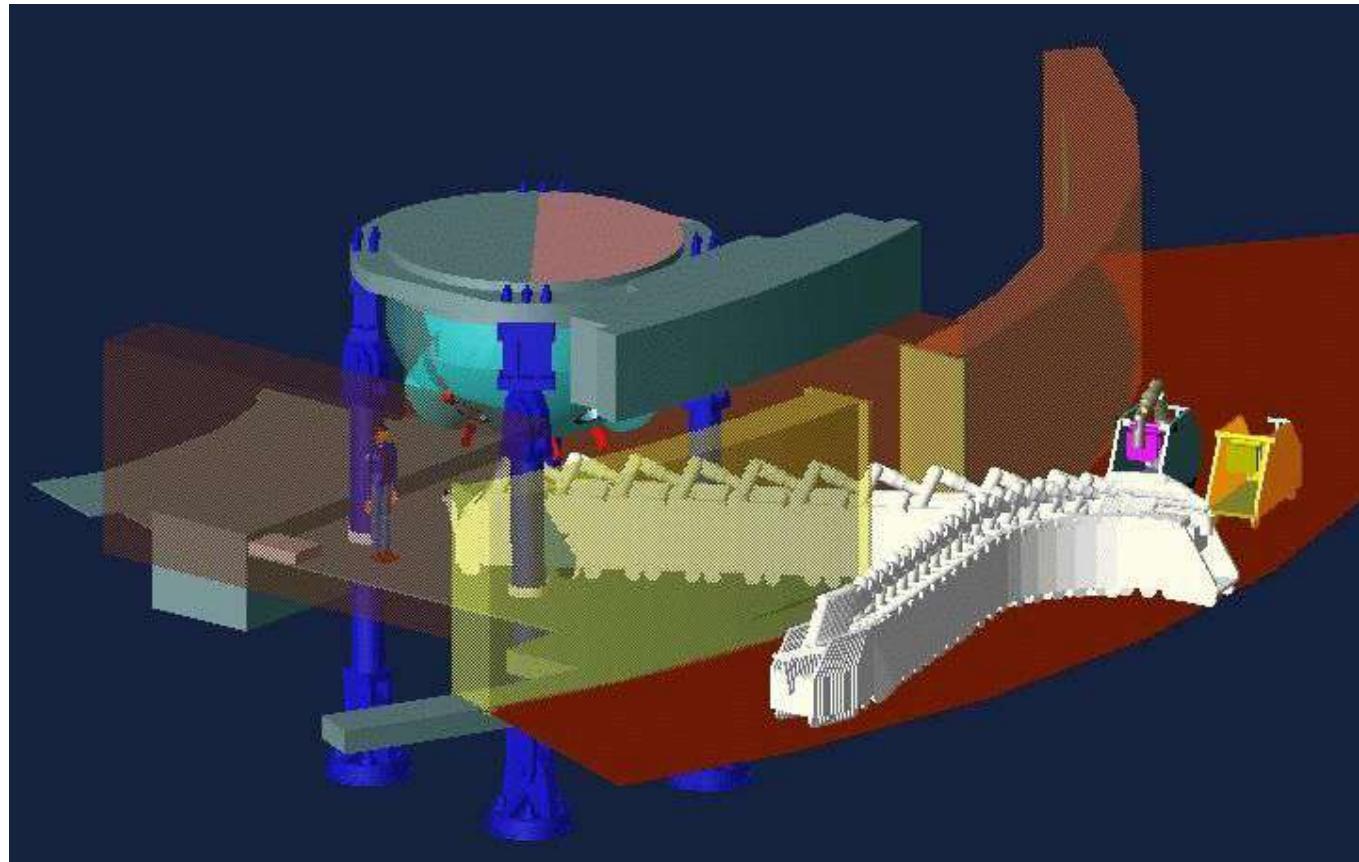
Inspection and surveillance



- *Hilare 2bis* mobile robot (LAAS), with trailer and manipulator arm, and its localization on a indoor map
 - sensors: encoders (on wheels and arm joints), ultrasound, SICK laser, and camera on end-effector gripper



Inspection and surveillance



- motion planning of a robotized inspection task inside an electricity power plant (*Move3D* simulation)



Mine exploration



- *Groundhog* (Carnegie Mellon)
- 750 kg, double axes, articulated
- movable SICK laser (rangefinder)
- gas and immersion sensors
- SLAM algorithm (Simultaneous Localization And Mapping)



RoboCup and RoboRescue



- RoboCup middle-size league (wheeled mobile robots, here with omni-directional vision)
- *Orpheus* robot won the RoboRescue (exploration and search of victims in a disaster environment)

2003 edition, Padova Fair



DARPA Grand Challenge



5 SICK lasers for mapping and localization
on the 2005 winning VW Touareg "Stanley"



the "Ghostrider" motorcycle
testing in Nevada

- competition for fully **autonomous** vehicles on a long mixed-type track

DARPA Grand Challenge



video

video interview



S. Thrun of [Stanford Racing](#)
(Stanford Univ+VW America+many more)

[Stanley](#) navigation:
GPS, laser scanners,
vision, radar

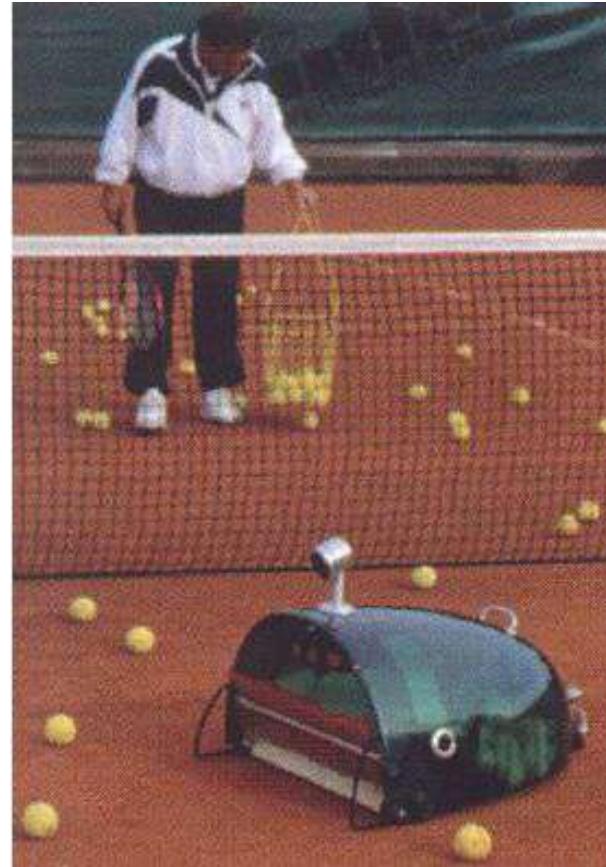


A. Levandowski of [Blue Team](#)
(LaRaison Inc+Univ Berkeley+Texas A&M)

[Ghostrider](#) navigation:
GPS, inertial unit, motorcycle
dynamics, stereovision



Free time



- bartender robot by Erhardt+Abt (Germany)
- the robotic ball boy (RWI and Carnegie Mellon Univ, USA) that won in 1996 the "Clean up the tennis court!" competition of the AAAI



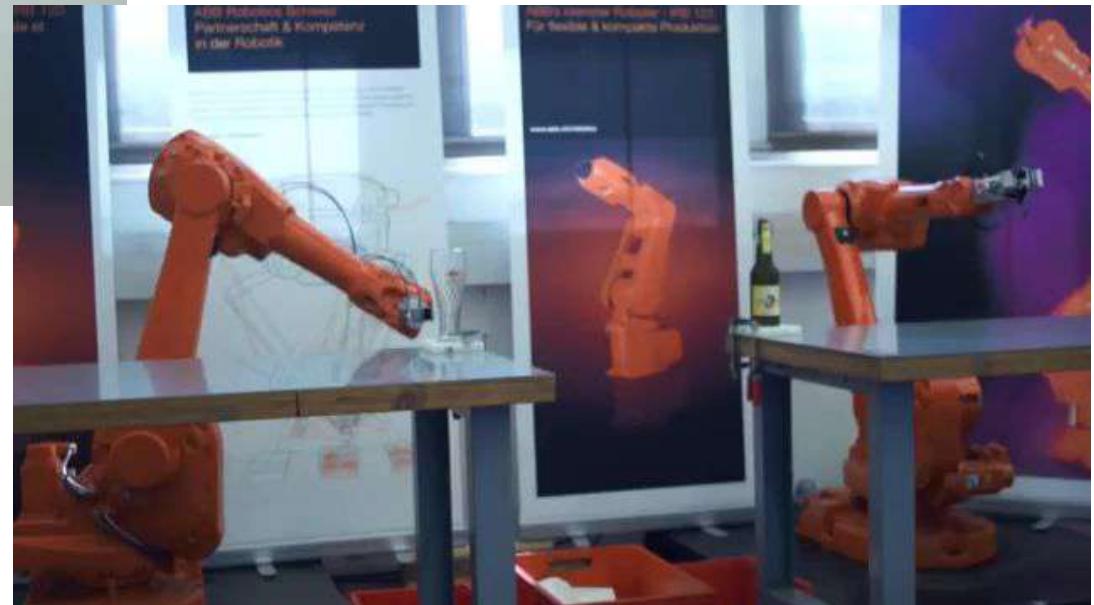
Robots filling a glass of beer



a single KUKA robot

video

two cooperating ABB robots



video



Museum guidance



- three mobile robots for museum guidance
(Museum für Kommunikation, Berlin)



Entertainment



- the *Anaconda* robot (Edge Innovations, USA) weights various tons, has 60 artificial spinal vertebrae, is 12 meters long, and is actuated by hydraulic motors so as to reach a speed of up to 60 km/h



Human motion replication



- the anthropomorphic robot by Sarcos Entertainment Systems (USA) replicates the movements of a human wearing a sensorized exoskeleton



Human-Robot Interaction



- physical and cognitive interaction between a Sarcos robot and a human
 - intrinsic mechanical compliance in the robot structure is here more important than accuracy in motion execution



Human-Robot Interaction (HRI)

video



video



cognitive interaction (cHRI)
in Robot@CWE EU Project

physical interaction (pHRI)
in PHRIENDS EU Project



Human-robot collaboration



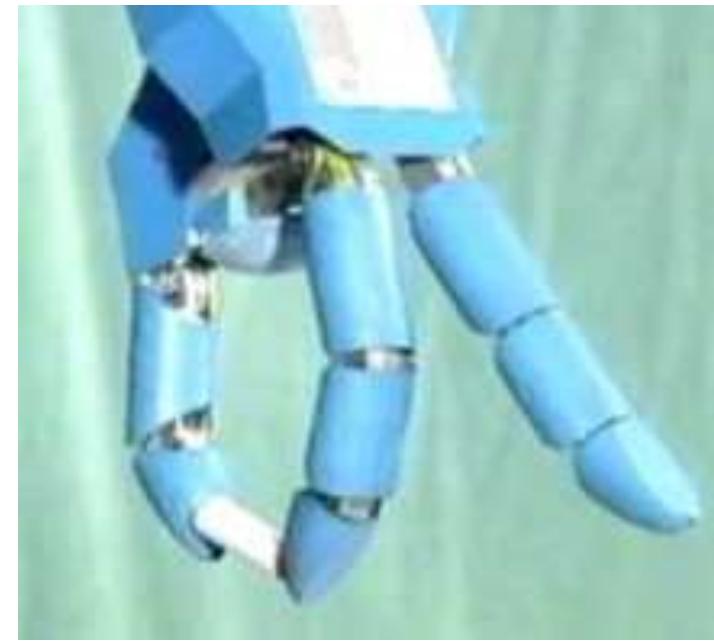
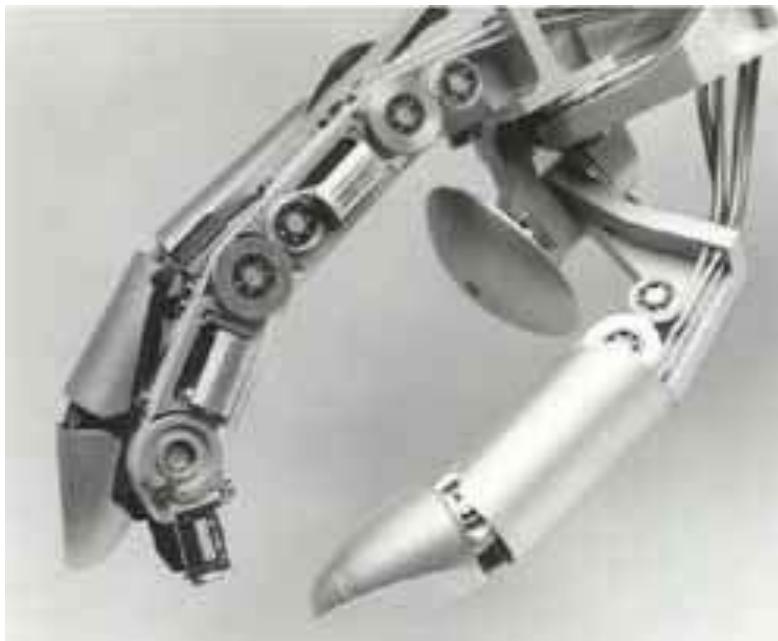
- *Mr. Helper* (Tohoku Univ) collaborates in carrying heavy and/or large loads



- *CoBot* scooter-like robot for mounting car doors (General Motors)



Robot hands



- the *UBHand* series of robot dexterous hands (Univ Bologna)
3 fingers with 9 degrees of freedom, tendon actuation,
supporting palm, and tactile sensors on all phalanges



Anthropomorphic UBHand IV



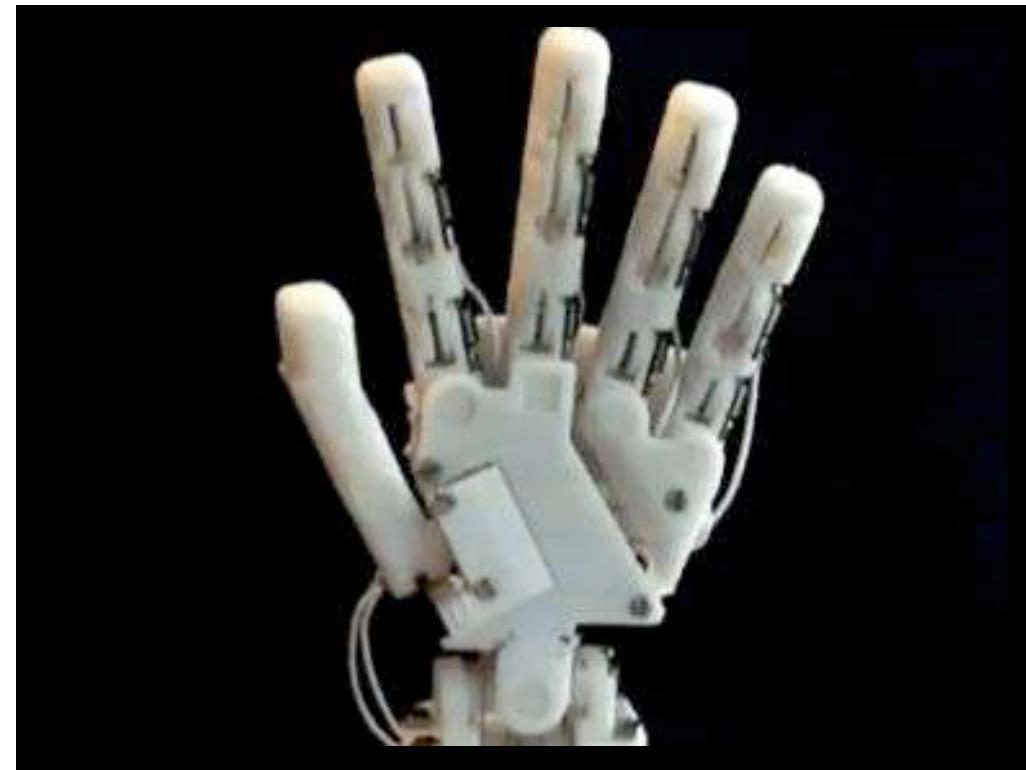
video

← data glove
for motion
capturing

- the *UBHand IV* has **deformable** elements as joint hinges (compliant mechanisms);
the endo-skeletal structure with **5 fingers** may host distributed sensors
and continuous compliant cover (G. Vassura, Univ Bologna)



New robot hand for the iCub



[video](#)

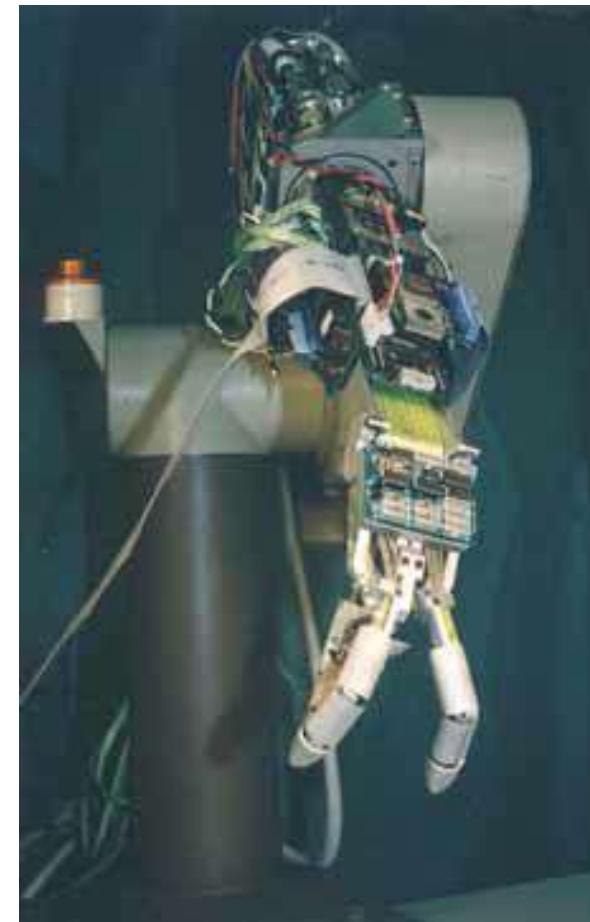
- *iCub* robot is like a 3.5y old child, developed by IIT Genova in 2005 in the [RobotCub](#) EU Project (platform distributed openly, with open-source SW)



Integration of robot hand + arm



- the complete *UBHand II*, with electrical motors and electronics presented at EXPO92 in Seville
 - integration in the forearm of the *Unimation PUMA 560*





... “minimalistic” solutions



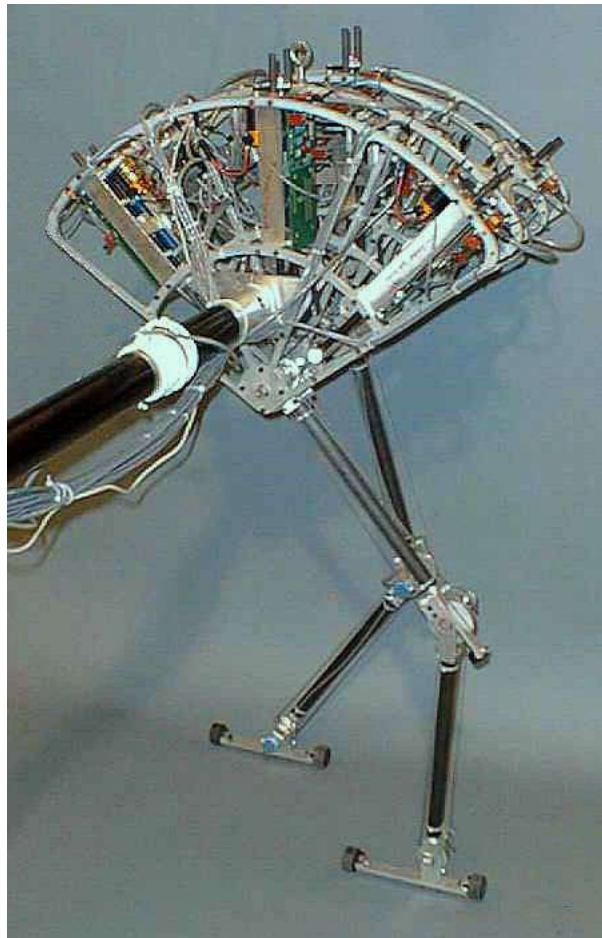
- 5D manipulation of objects having arbitrary form, using only two linear actuators and sensorized contact surfaces
(Univ Pisa, about 15 years ago)



- 1 motor only for the *Soft Hand*, an articulated and compliant robotic hand of human size and synergy
(Centro “E. Piaggio”, Univ Pisa – IIT, 2014)



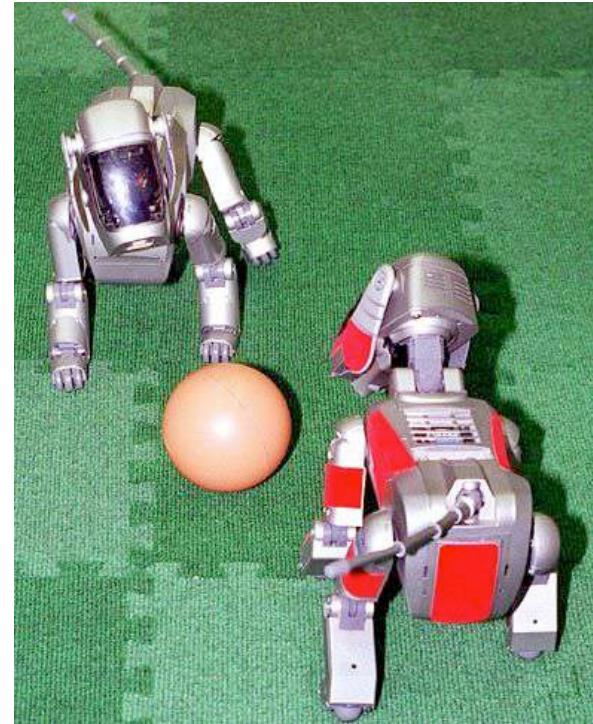
Biomorphic robots



- biomorphic robots by MIT Leg Lab, USA:
Troody dinosaur and *Flamingo* bird



Four-legged locomotion

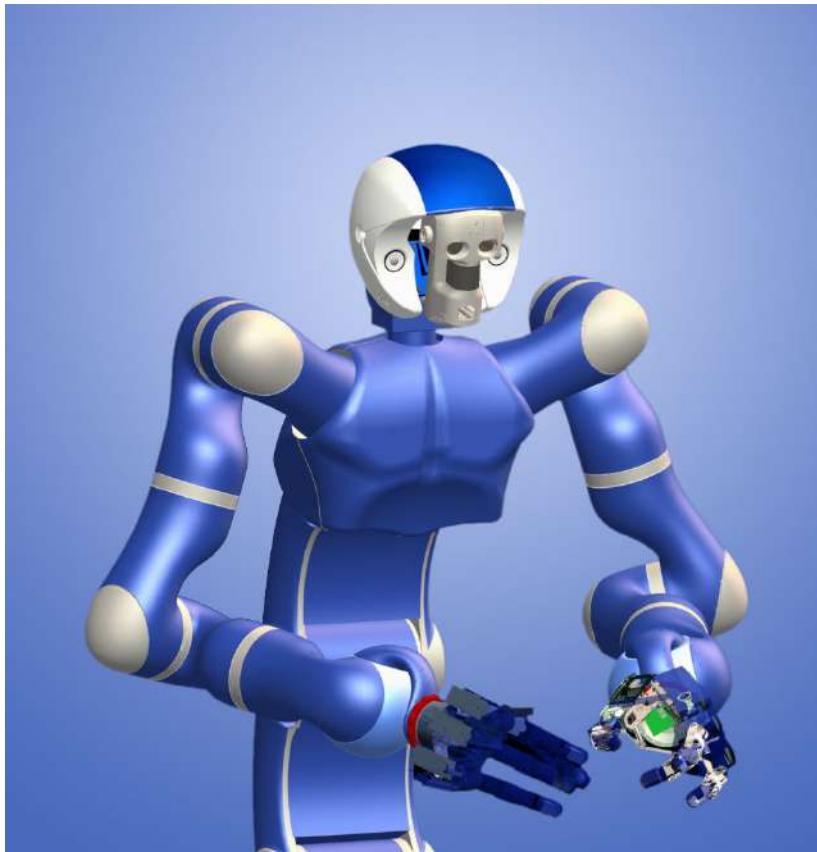


- *AIBO ERS-210* by Sony, playing on the soccer field of RoboCup

16 actuated dofs with encoders, color camera, 3 accelerometers, ultrasound sensors, tactile and micro-switch (feet), battery: everything in 1.6 kg!



Anthropomorphic upper limbs



- *Justin* robot has 7+7+3 degrees of freedom + many dofs in the two hands (DLR, Germany)



- the robot developed in the German national project on humanoids

Justin robotic system @ DLR

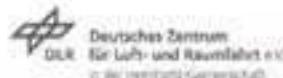
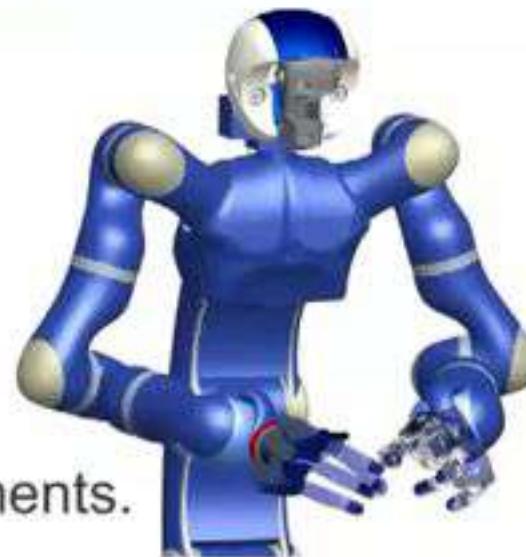


video



Justin

A humanoid upper
body system for
two-handed
manipulation experiments.

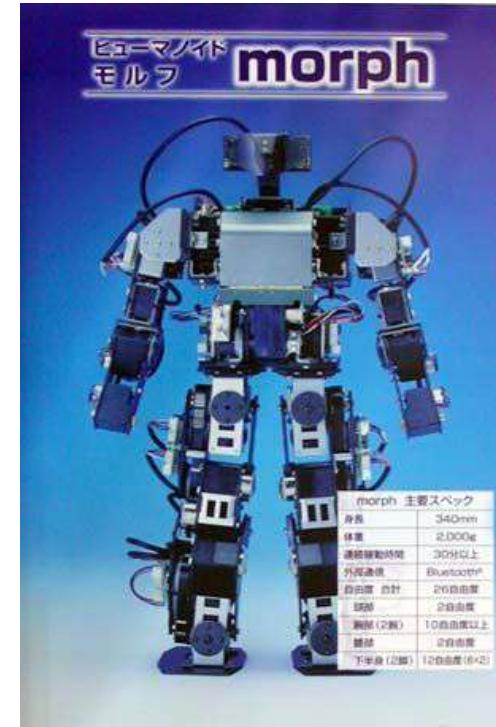




Humanoid robots



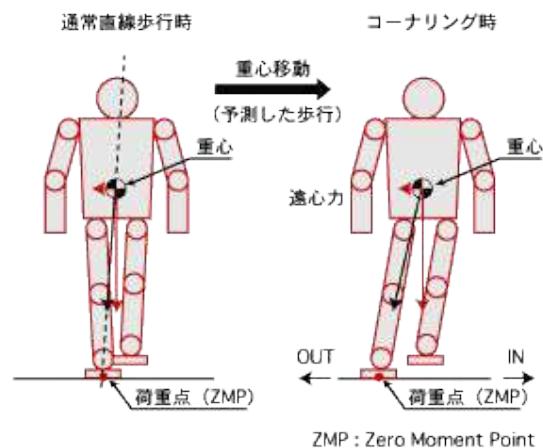
- Metropolis (Fritz Lang, 1927)



- *Pino* by ZMP (2003)



Humanoid robots



- the *ASIMO* project by Honda started in 1986



ASIMO in action

ASIMO
climbing stairs
(Robodex 2003)



first and
second series
(smaller size)

video



video

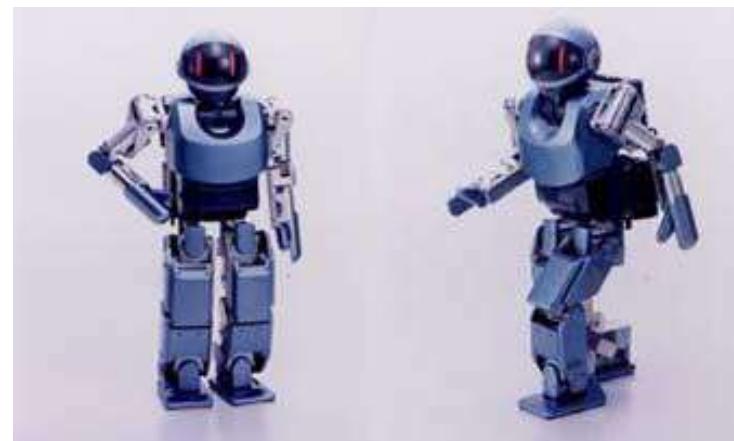




Humanoid robots



- *HRP-2*
(58 kg, 150 cm, 30 dofs)
2002 Tokyo Univ



- Sony *SDR-3X*
(about 60 cm)



- humanoid robot
(Q. Huang, PR China)



Sony Q-RIO



group dancing [video](#) (2003)

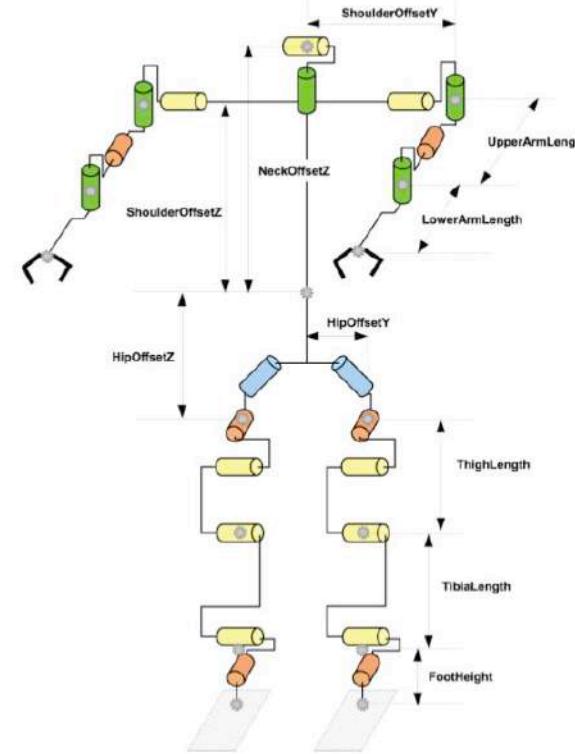
- *Sony Q-RIO*

the first robot able to balance on a surf and stand up from the floor
(dead in 2006...)



Humanoid robots

height = 57 cm
weight = 4.5 kg



kinematic
description

- NAO, Aldebaran Robotics
since 2008, replaces AIBO quadrupeds in RoboCup standard league



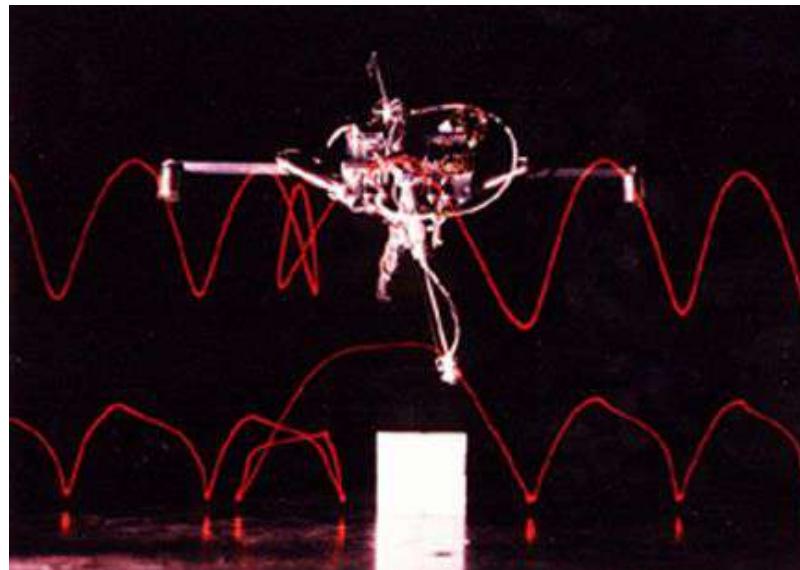
NAO playtime



Aldebaran Robotics
commercial [video](#)



... what about dynamic stability?



video

- the *One-Leg Hopper* robot (MIT, USA) demonstrated back in 1982 the feasibility of maintaining a purely **dynamic** equilibrium during motion



we could go on and on, forever...

[video](#)



MIT planar two-legged
robot doing a flip (1984)

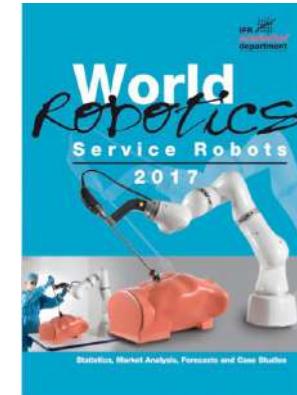
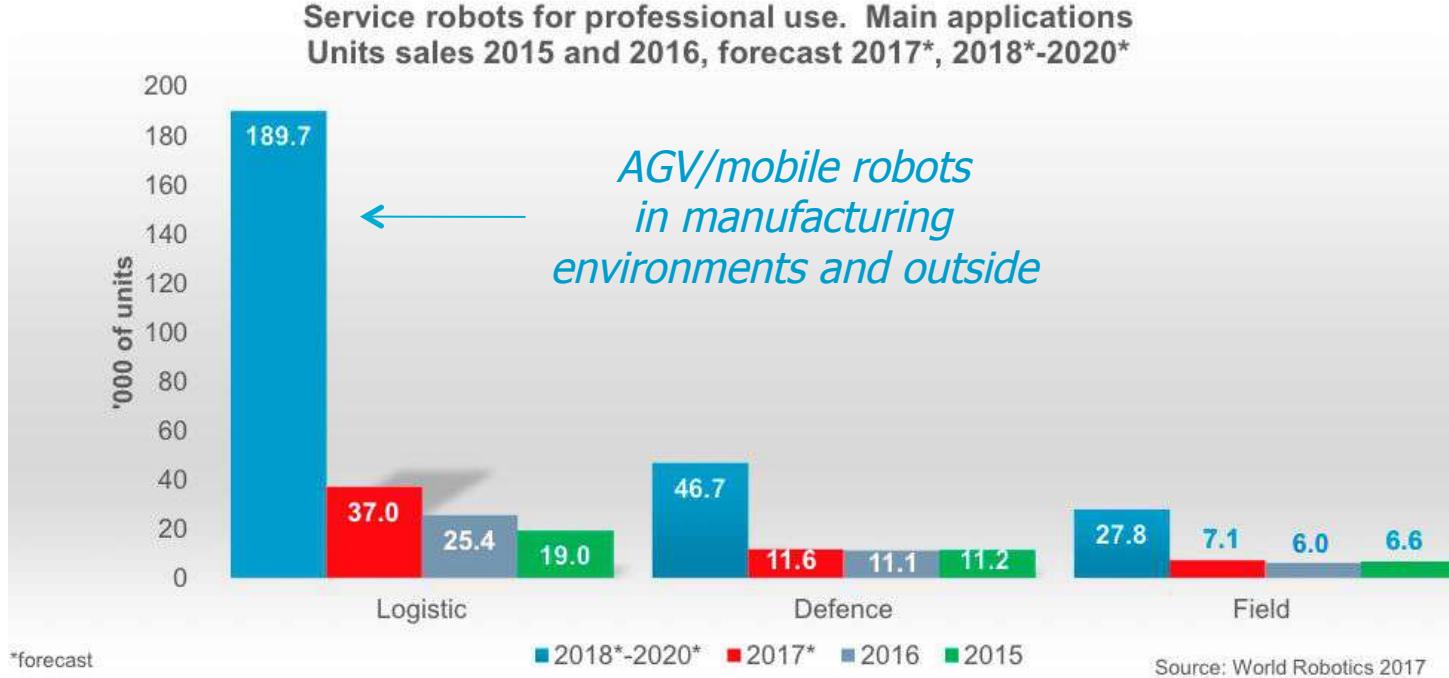
[video](#)



ping-pong with KUKA KR5 robot

the beauty of dynamics and juggling

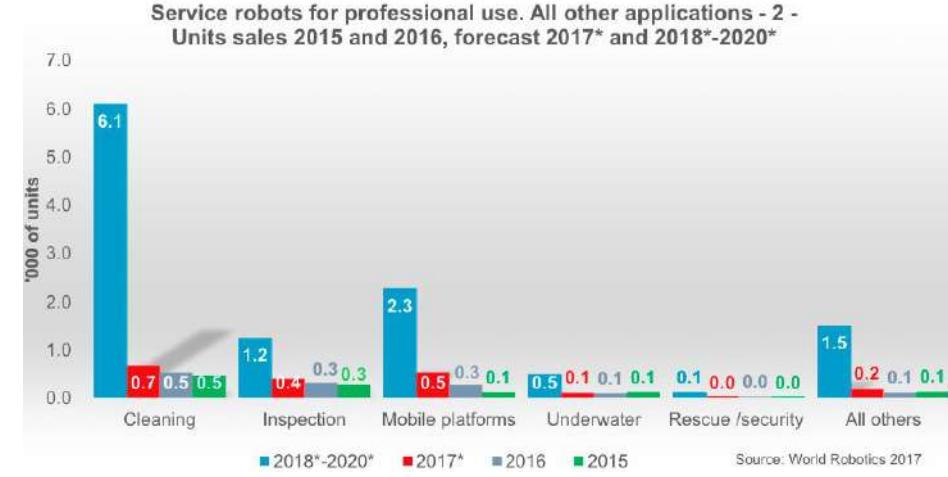
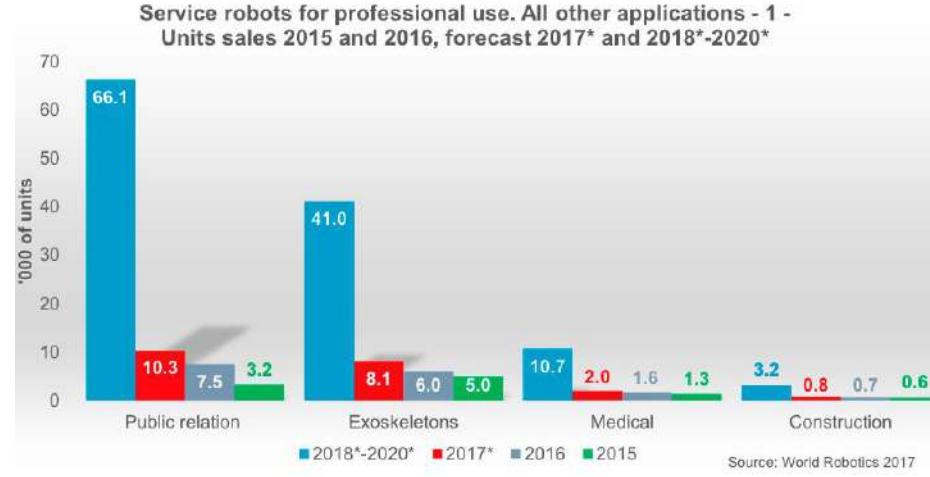
Diffusion of service robots for professional use



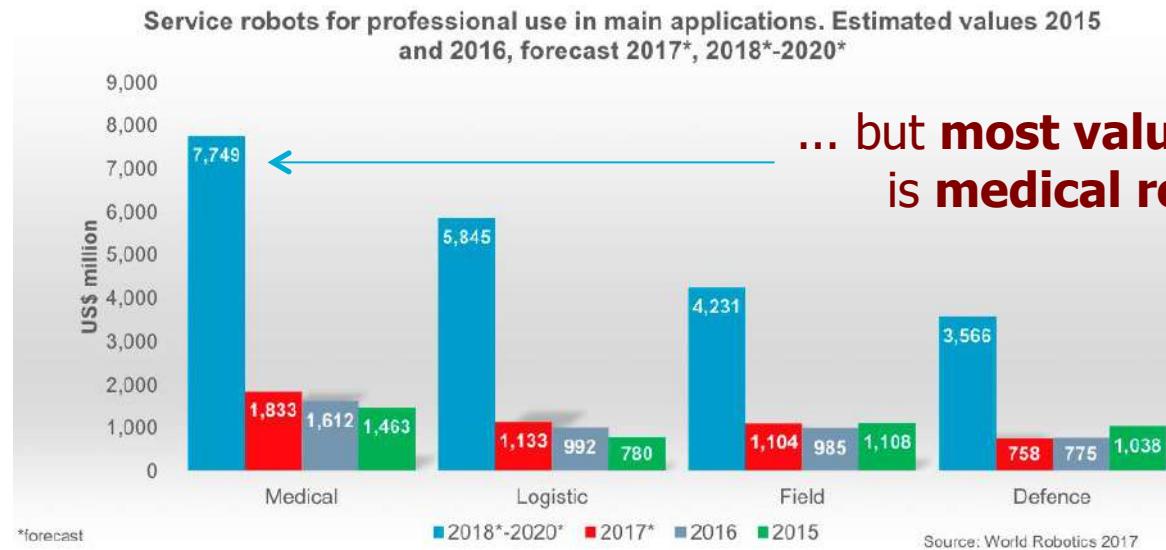
World Robotics
Report 2017

- main applications: logistic, defense (mostly, UAV), field robots (e.g., milking)
- almost **60,000** (+24%) service robots for **professional** use sold in 2016
- ~ **400,000** new service robots for professional use to be sold in 2018-20*
(+45% estimate over last period)
- > 50% of professional service robots are manufactured in the Americas

Diffusion of service robots for professional use



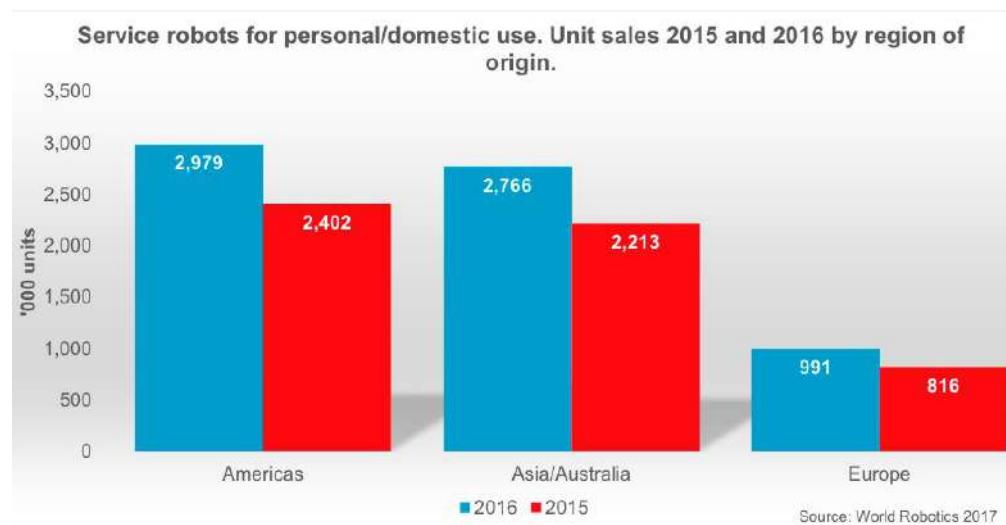
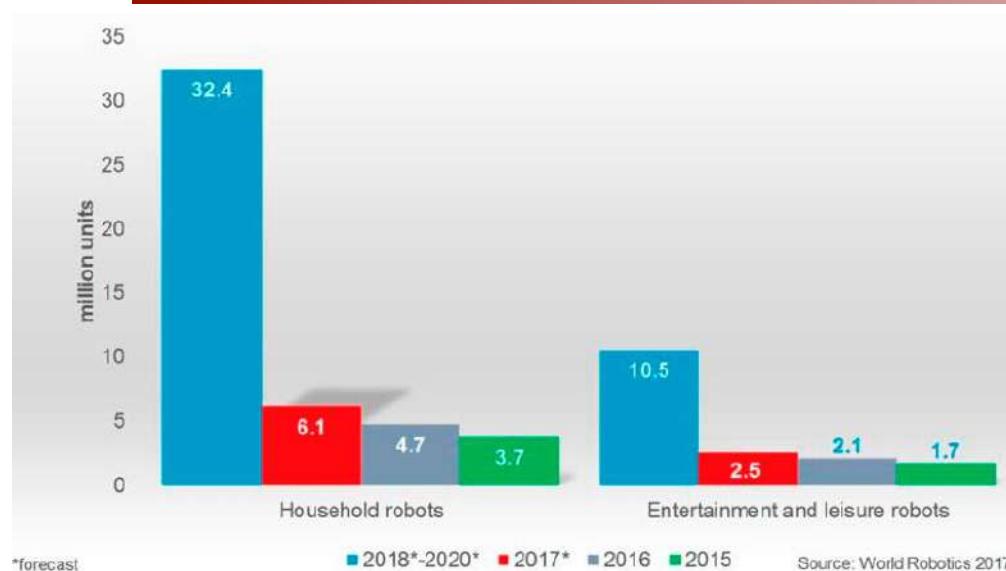
other applications: **social** (↗), exoskeletons (↗), medical systems (↗), construction, cleaning, inspection, agriculture, underwater, rescue...



public relation
robots also expect significant increase in turnover (4.5 M\$ in 2018-20*)

... but **most valuable** market is **medical robotics!**

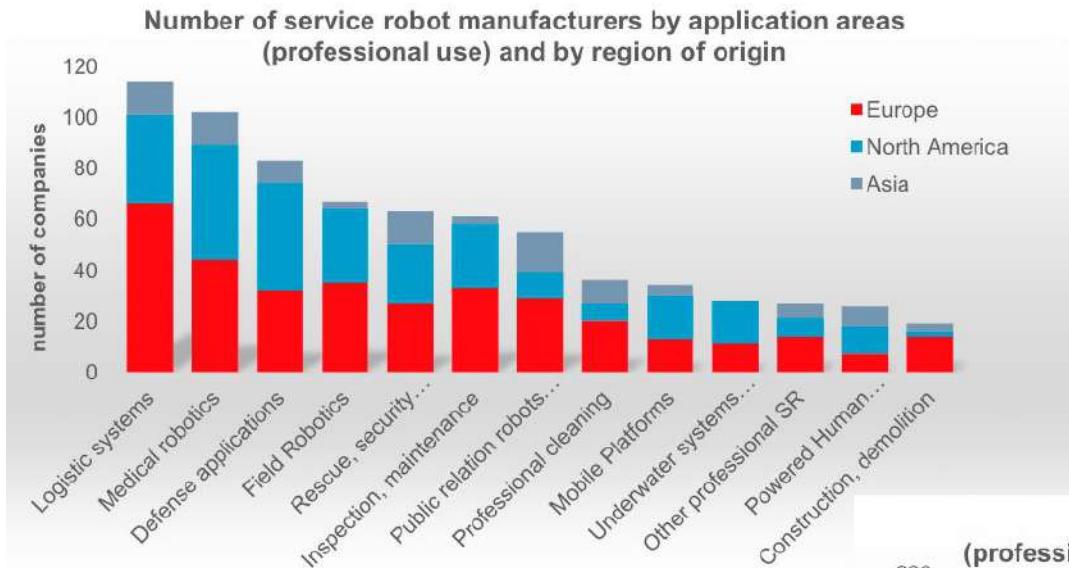
Diffusion of service robots for personal/domestic use



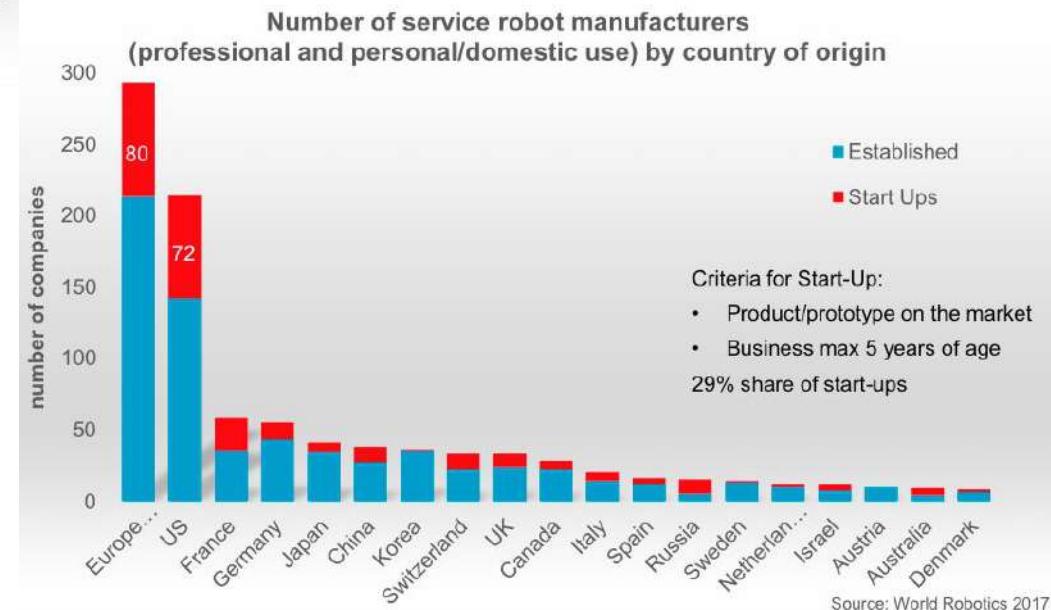
- personal/domestic robots market is **huge**: hard to assess figures due large variety in products and lifetime
- **6.8M units** sold in 2016 (+25%), for a value of **2.6 G\$** (+15%)
- forecast 2018-20*: **43M units**
- vacuum and floor cleaning robots
- lawn-mowing robots
- entertainment (e.g., Mindstorm) and leisure
- robots for elderly/handicap assistance

Diffusion of service robots

manufacturers by application areas and regions



- 75% of European service robot suppliers are SMEs
- many start ups ...





Web sites

- <http://ieee-ras.org>

The web site of IEEE-RAS (*Institute of Electrical and Electronics Engineers - Robotics & Automation Society*) with access to conferences, publications, technical committees, etc. – *become a student member of the IEEE-RAS !!*

- <http://handbookofrobotics.org>

Multimedia extension of the *Springer Handbook of Robotics* (2nd Ed, 2016), with >700 *free* robotics videos of historical and state-of the-art value

- <http://www.eu-robotics.net>

The new *European Robotics AIBSL*, with euRobotics Forum & Week, etc.

- <http://www.euron.org>

EUropean RObotics research Network, with a gallery of robots, videos, European robotics projects (no longer updated since 2012)

- <http://www.youtube.com/user/RoboticsLabSapienza>

YouTube channel of *DIAG Robotics Lab*, with videos of our latest research



Robotics 1

Robot components: Actuators

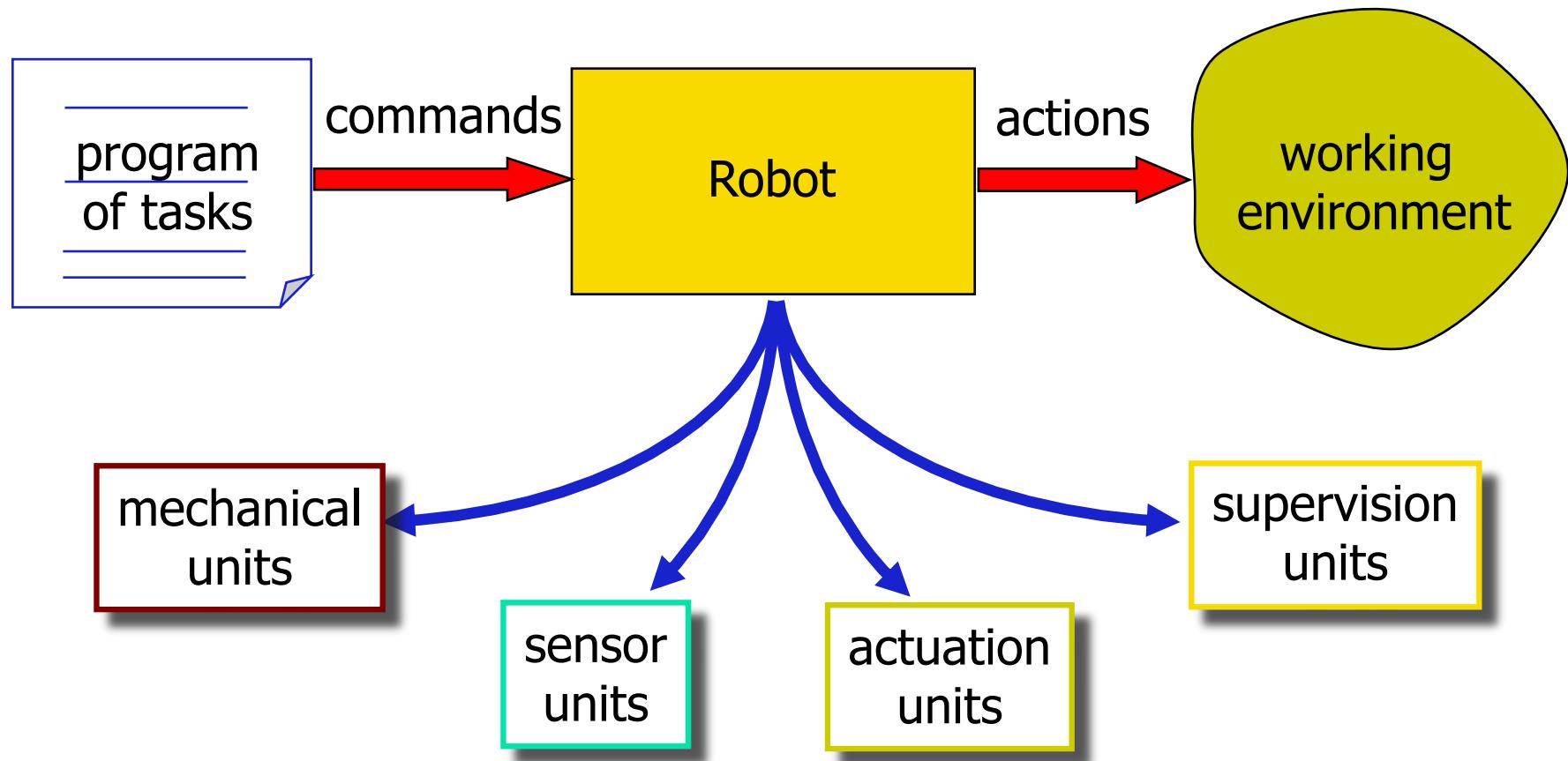
Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Robot as a system





Functional units of a robot

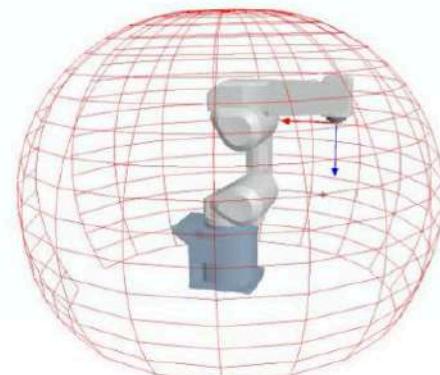
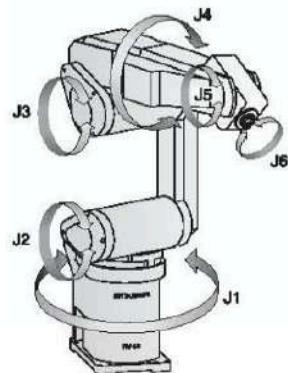
- mechanical units (robot arms)
 - serial manipulators: rigid links connected via **rotational** or **prismatic** joints (each giving 1 degree of freedom = DOF)
 - **supporting structure** (mobility), **wrist** (dexterity), **end-effector** (for task execution, e.g., manipulation)
- actuation units
 - motors (**electrical**, **hydraulic**, **pneumatic**) and transmissions
 - motion control algorithms
- sensor units
 - **proprioceptive** (internal robot state: position and velocity of the joints)
 - **exteroceptive** (external world: force and proximity, vision, ...)
- supervision units
 - task **planning** and **control**
 - artificial intelligence and reasoning



Arrangement of mechanical links

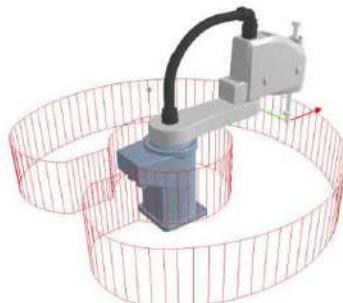
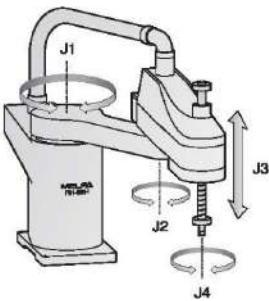
4, 5, or 6 joints
(DOFs)

Articulated Robot

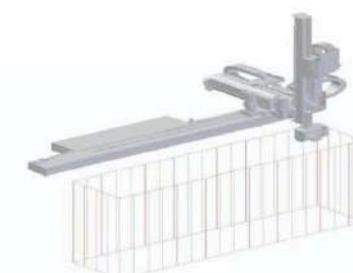
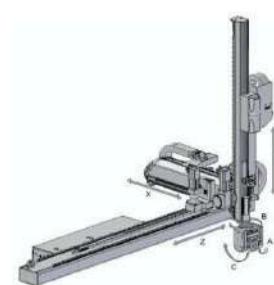


different kinematic types of robot arms

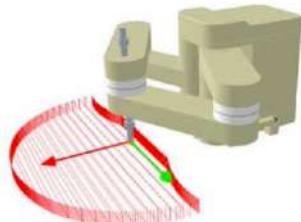
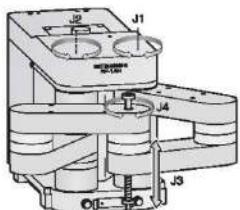
SCARA Robot



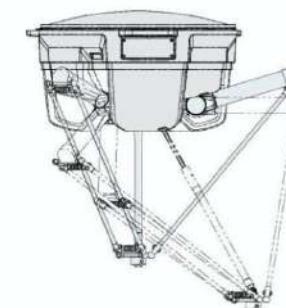
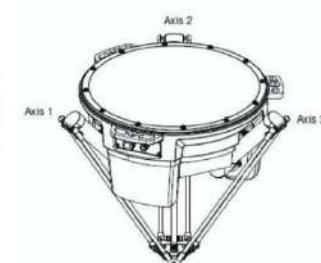
Cartesian Robot



SCARA Robot



Parallel/Delta Robot



Examples of industrial robots with brands



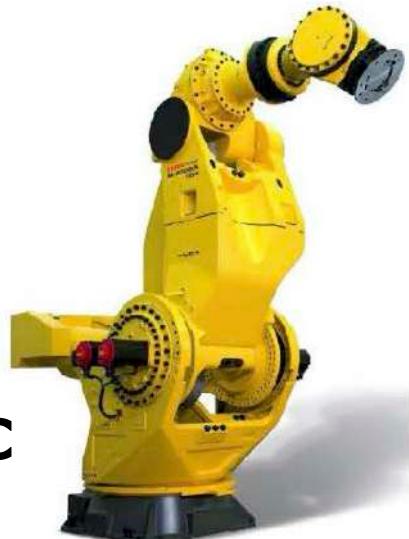
ABB



DAIHEN



EPSON



FANUC



KUKA



NAICHI



Bi-manual industrial robots with brands



ABB



COMAU

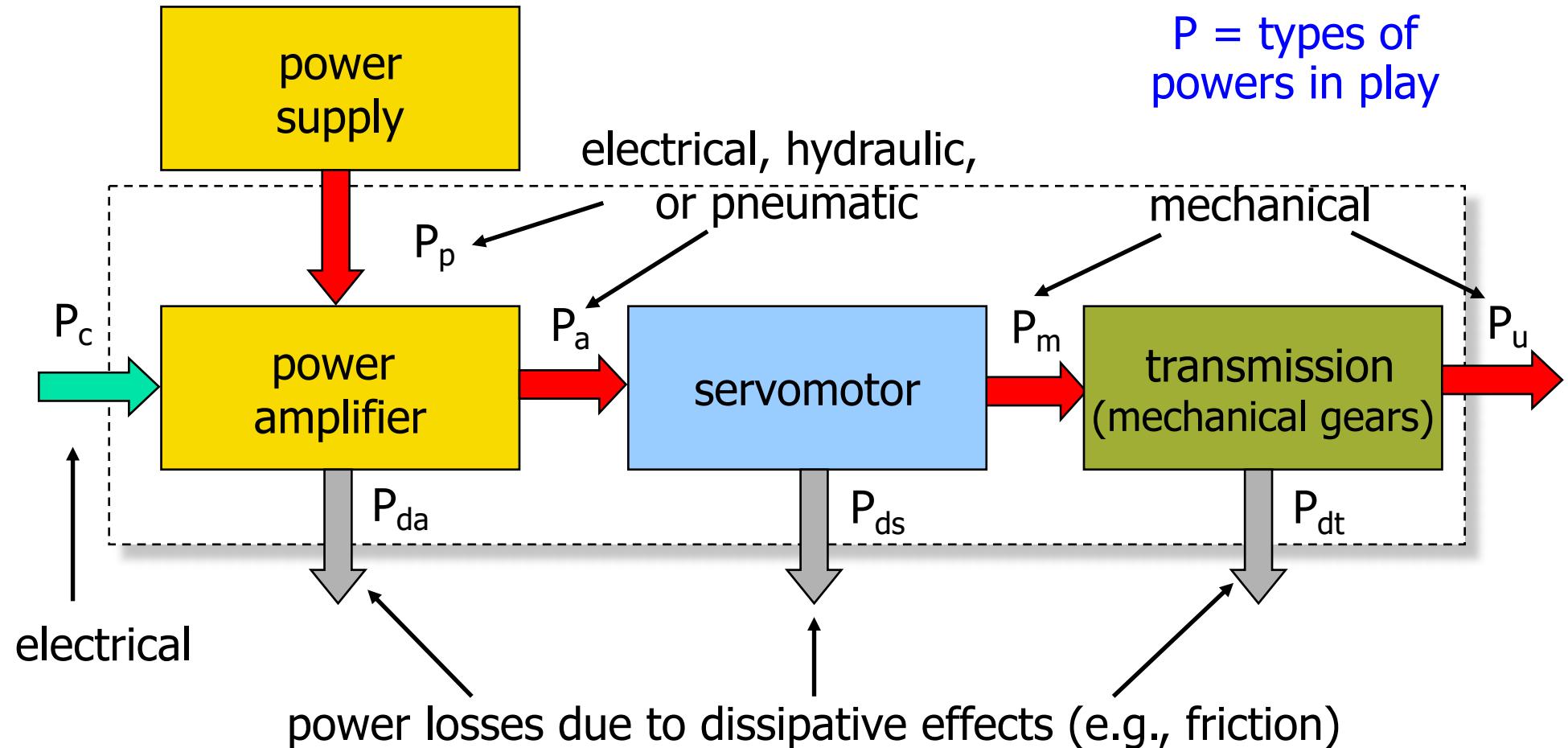
**UNIVERSAL
ROBOTS**



YASHKAWA



Actuation systems



power = voltage · current = pressure · flow rate = force · speed = torque · angular speed [W, Nm/s]

efficiency = power out/power in [%] **energy** ~ **work** = power · time [kWh, Nm, J]



Desired characteristics for robot servomotors

- low inertia
- high power-to-weight ratio
- high acceleration capabilities
 - variable motion regime, with several stops and inversions
- large range of operational velocities
 - 1 to 2000 rpm (round per min)
- high accuracy in positioning
 - at least 1/1000 of a turn
- low torque ripple
 - continuous rotation at low speed
- power: 10 W to 10 kW



Servomotors

- **pneumatic:** pneumatic energy (compressor) → pistons or chambers → mechanical energy
 - difficult to control accurately (change of fluid compressibility) → no trajectory control
 - used for opening/closing grippers
 - ... or as artificial muscles (McKibben actuators)
- **hydraulic:** hydraulic energy (accumulation tank) → pumps/valves → mechanical energy
 - **advantages:** no static overheating, self-lubricated, inherently safe (no sparks), excellent power-to-weight ratio, large torques at low velocity (w/o reduction)
 - **disadvantages:** needs hydraulic supply, large size, linear motion only, low power conversion efficiency, high cost, increased maintenance (oil leaking)



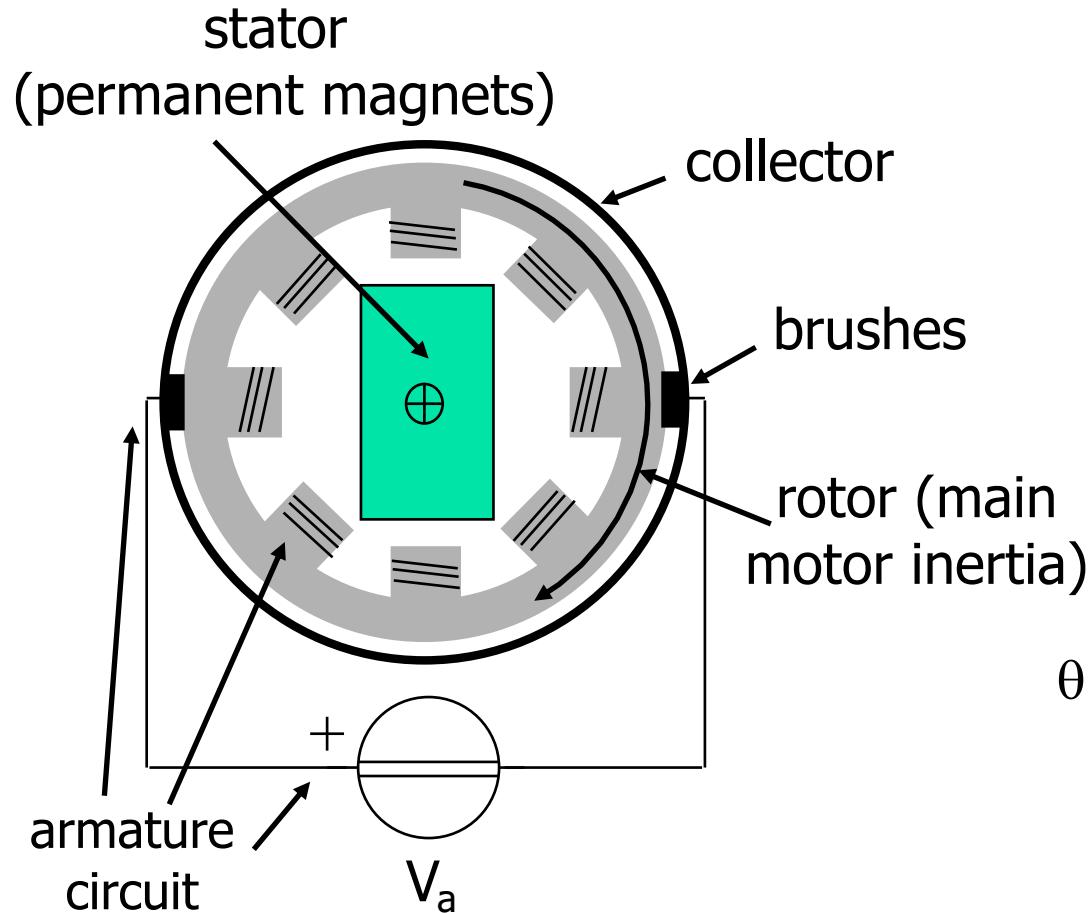


Electrical servomotors

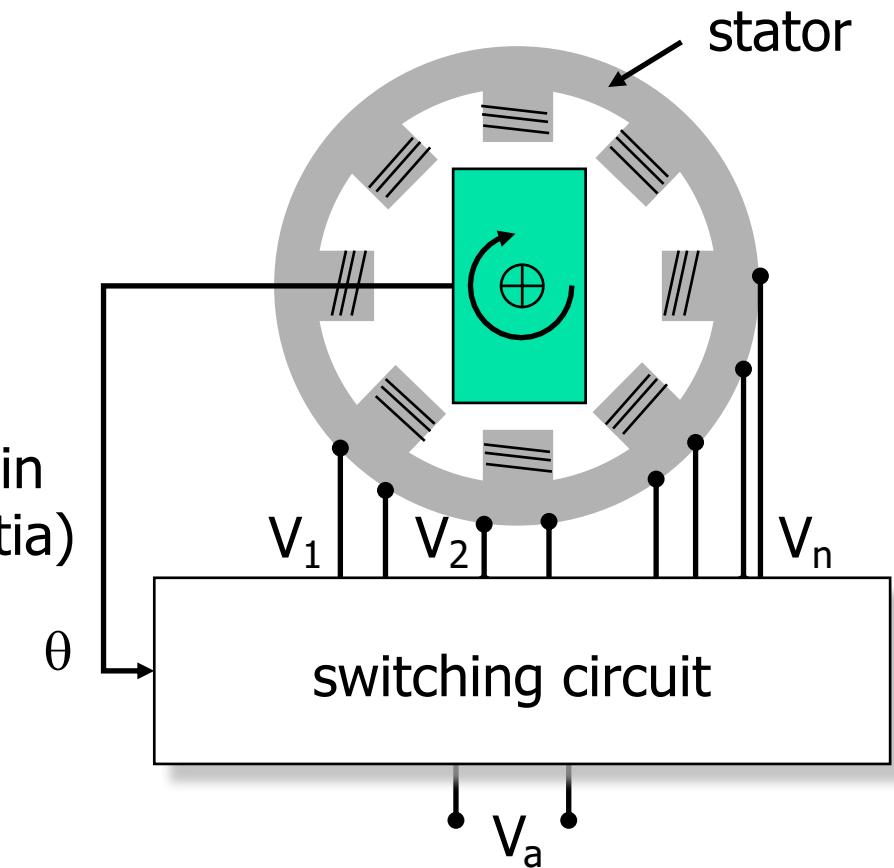
- **advantages**
 - power supply available everywhere
 - low cost
 - large variety of products
 - high power conversion efficiency
 - easy maintenance
 - no pollution in working environment
- **disadvantages**
 - overheating in static conditions (in the presence of gravity)
 - use of emergency brakes
 - need special protection in flammable environments
 - some advanced models require more complex control laws



Electrical servomotors for robots



direct current (DC) motor



with electronic switches (brushless)



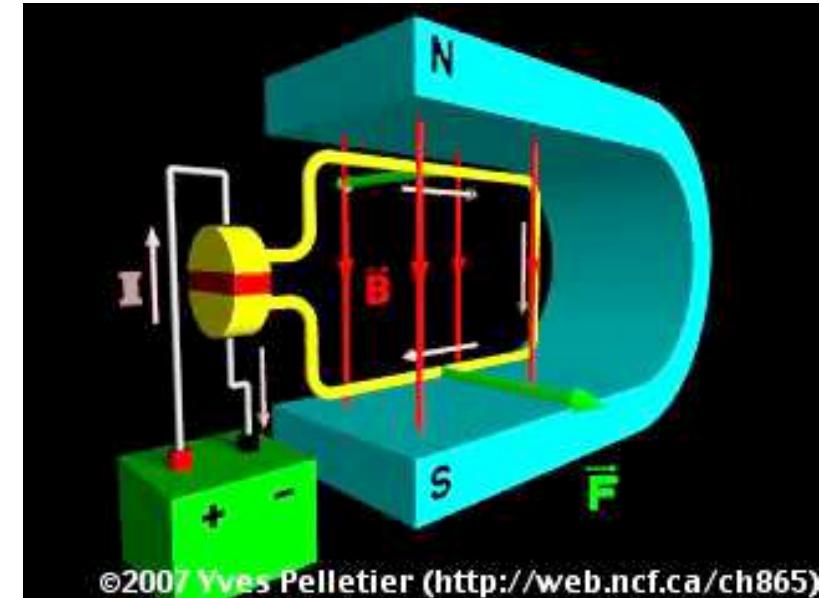
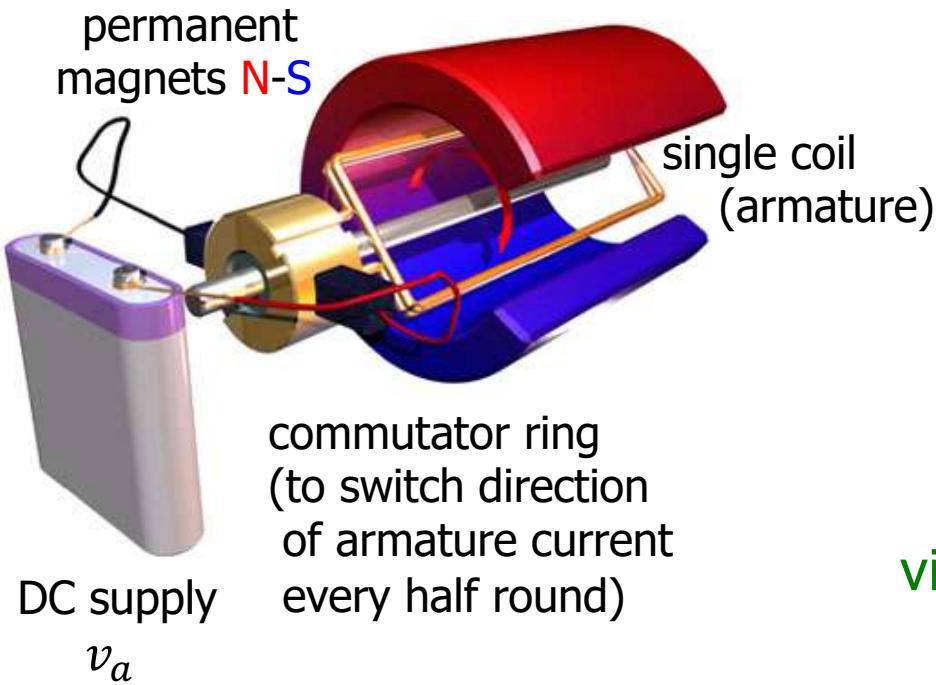
Advantages of brushless motors

- reduced losses, both electrical (due to tension drops at the collector-brushes contacts) and mechanical (friction)
- reduced maintenance (no substitution of brushes)
- easier heat dissipation
- more compact rotor (less inertia and smaller dimensions)

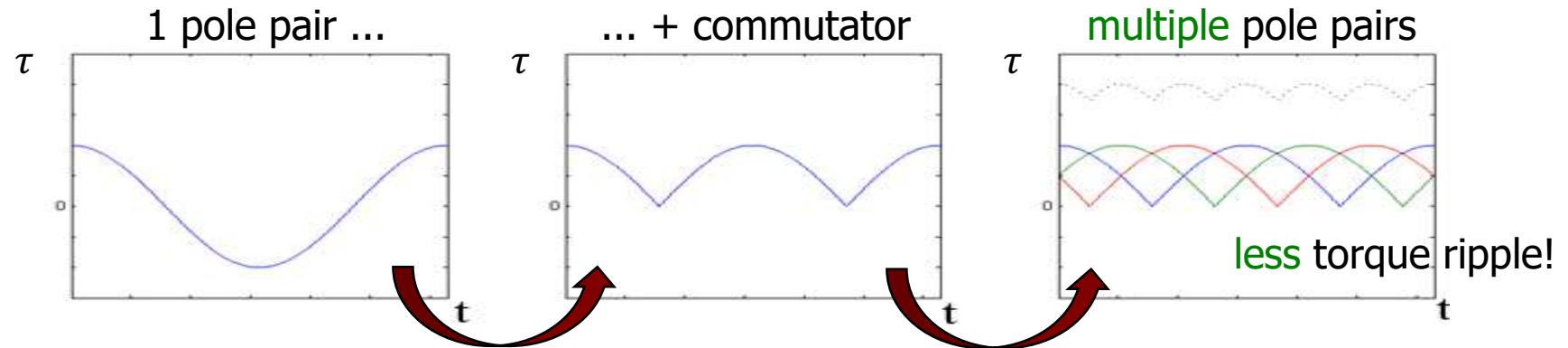
... but indeed a higher cost!



Principle of operation of a DC motor



$$\vec{F} = L(\vec{i} \times \vec{B}) \quad \tau = d|\vec{F}|$$





DC electrical motor

mathematical model (in the time domain)

electrical balance
(on the equivalent armature circuit)

$$v_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + v_{emf}(t)$$

$$v_{emf}(t) = k_v \omega(t)$$

(back emf)

mechanical balance
(Newton law on torques)

$$\tau_m(t) = I_m(t) \frac{d\omega(t)}{dt} + F_m \omega(t) + \tau_{load}(t)$$

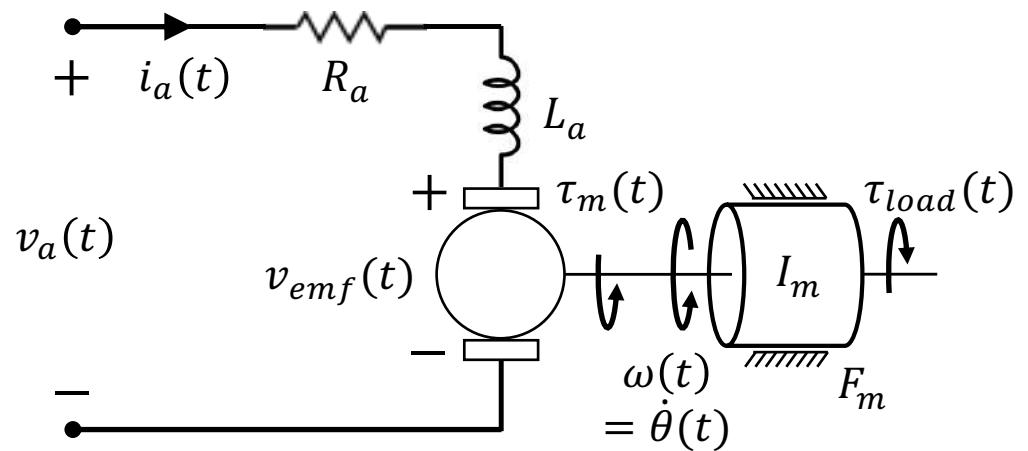
$$\tau_m(t) = k_t i_a(t)$$

(motor torque)

in absence of losses, conservation of power holds in energy transformation

$$P_{elec} = v_{emf} i_a = \tau_m \omega = P_{mecc}$$

$$\Rightarrow k_v = k_t \quad (\text{in SI units})$$



using Laplace transform, differential equations become algebraic relations!

$$X(s) = \mathcal{L}[x(t)] = \int_0^{\infty} x(t) e^{-st} dt$$

DC electrical motor

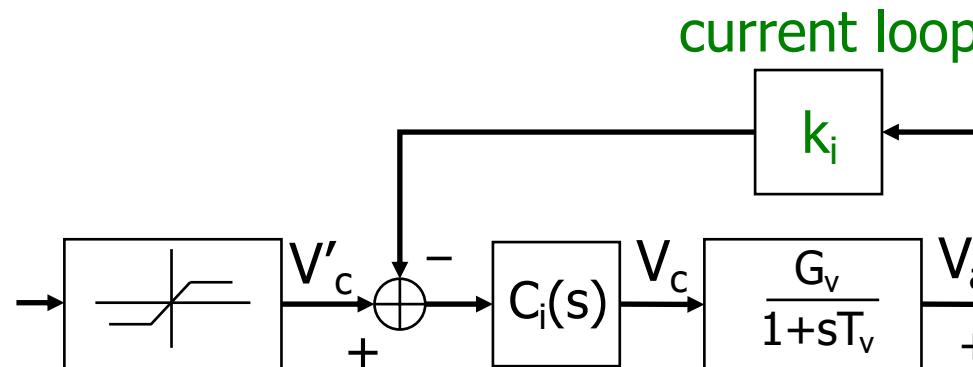
mathematical model for command and control



electrical balance

$$V_a = (R_a + sL_a) I_a + V_{\text{emf}}$$

$$V_{\text{emf}} = k_v \Omega$$



$k_i = 0 \rightarrow$ velocity generator*
 $k_i C_i(0) G_v \gg R_a \rightarrow$ torque generator*

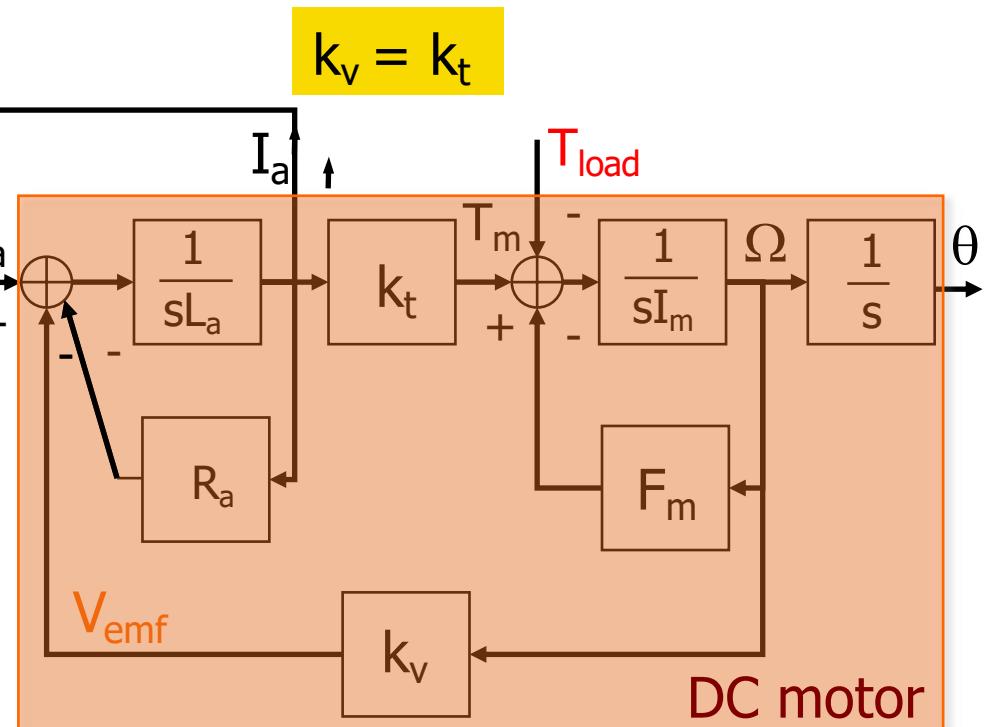
* = the motor is seen here as a steady state "generator"
 in order to actually **regulate** velocity or torque in an
 efficient way against T_{load} , further control loops are needed!

Laplace domain (transfer functions)

mechanical balance

$$T_m = (sI_m + F_m) \Omega + T_{\text{load}}$$

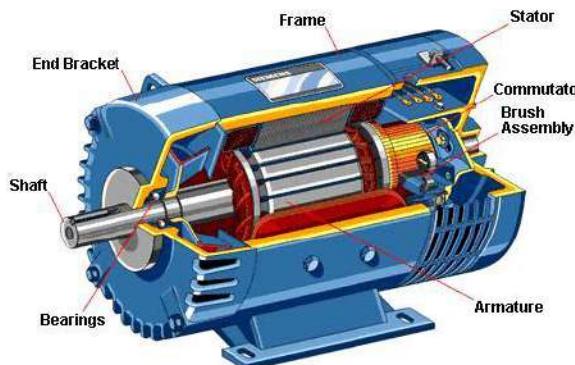
$$T_m = k_t I_a$$



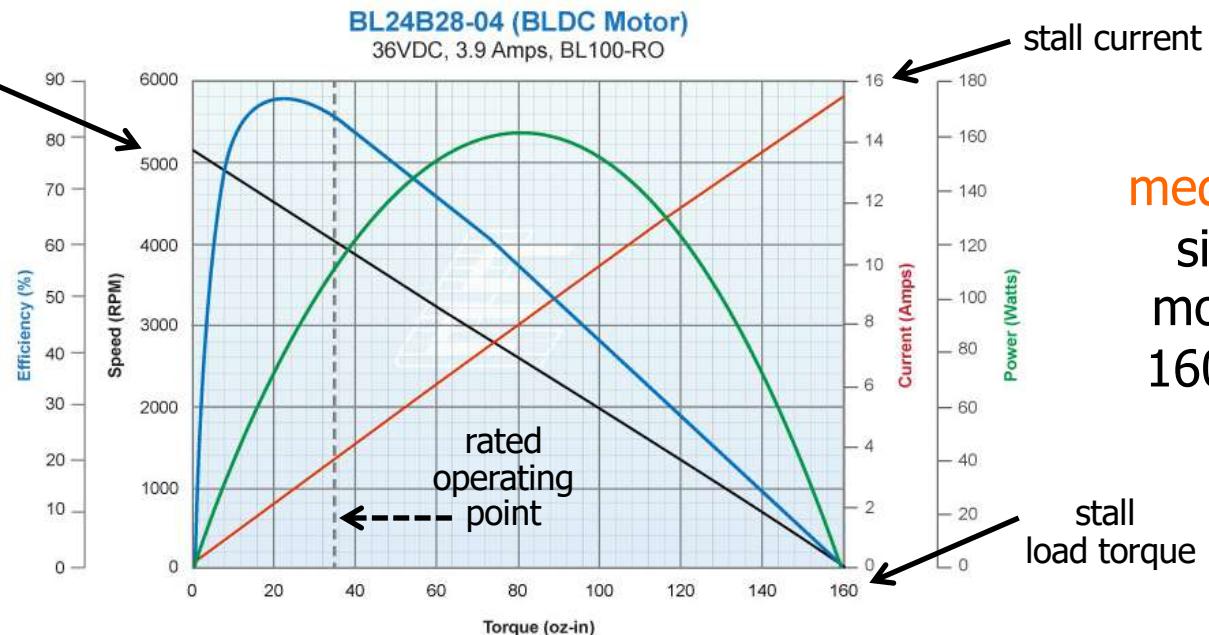


Characteristic curves of a DC motor

at steady-state,
for **constant**
applied tension v_a



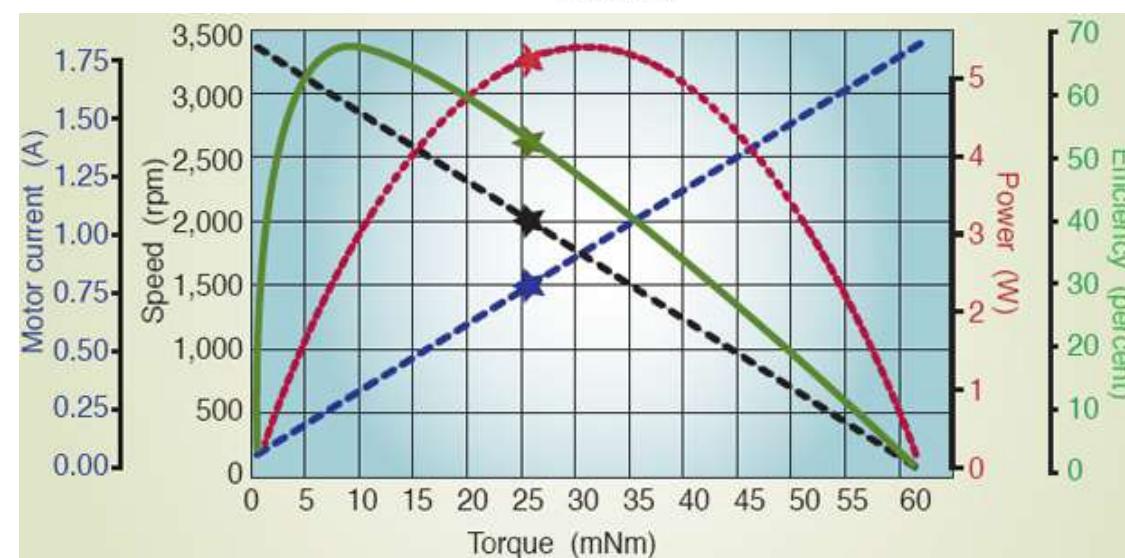
no-load
max speed



medium
size
motor
160 W

stall
load torque

conversion SI \leftrightarrow US
unit systems (!!)
 $1 \text{ Nm} = 141.61 \text{ oz-in}$
 $100 \text{ oz-in} = 0.70 \text{ Nm}$



small
size
motor
5.5 W



Data sheet electrical motors

- DC drives



Model of actuator		RHS-14		RHS-17		RHS-20/RFS-20				RHS-25/RFS-25				RHS-32/RFS-32			
		6003	3003	6006	3006	6007	3007	6012	3012	6012	3012	6018	3018	6018	3018	6030	3030
Rated Torque	Inlb	48	69	87	177	106	212	177	266	177	354	266	531	266	531	443	885
	Nm	5.4	7.8	9.8	20	12	24	20	30	20	40	30	60	30	60	50	100
Rated Speed of Rotation	rpm	60	30	60	30	60	30	60	30	60	30	60	30	60	30	60	30
Max. Instant. Torque	Inlb	159	248	301	478	504	743	504	743	885	1416	885	1416	1947	3009	1947	3009
	Nm	18	28	34	54	57	84	57	84	100	160	100	160	220	340	220	340
Max. Speed of Rotation	rpm	100	50	80	40	80	40	80	40	80	40	80	40	80	40	80	40

nominal/peak torques and speeds



Data sheet electrical motors

■ AC drives



	unit	HKM-20-60	HKM-20-30	HKM-25-60	HKM-25-30
Rated Power	Watts	100		200	
Rated Torque	in-lb	115	223	233	440
	N-m	13	26	26	50
Maximum Torque	in-lb	345	700	830	1330
	N-m	39	79	94	150
Rated Speed	r/min	60	30	60	30
Maximum Speed	r/min	80	40	80	40
Current Rated	A	1.8	1.4	4.8	3
Current Max	A	5	4	14	9
Thermal Time Constant	min.				
Gear Reduction Ratio	R:1	50	100	50	100
Output Resolution	P/rev	50,000	100,000	75,000	150,000
	arc sec	26	13	17	9
Absolute Accuracy	+/- arc sec	75	40	60	40

- for applications requiring a rapid and accurate response (in robotics!)
- induction motors driven by alternate current (AC)
- small diameter rotors, with low inertia for fast starts, stops, and reversals



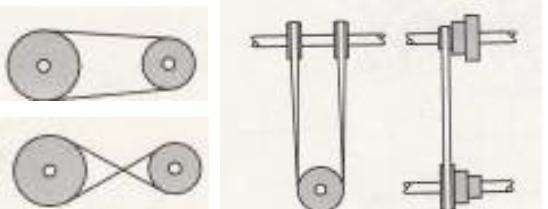
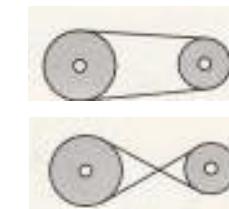
Motion transmission gears

- optimize the transfer of mechanical torque from actuating motors to driven links
- quantitative transformation (from **low torque/high velocity** to **high torque/low velocity**)
- qualitative transformation (e.g., from **rotational** motion of an electrical motor to a **linear** motion of a link along the axis of a prismatic joint)
- allow improvement of static and dynamic performance by reducing the weight of the actual robot structure in motion (locating the motors remotely, closer to the robot base)



Transmissions in robotics

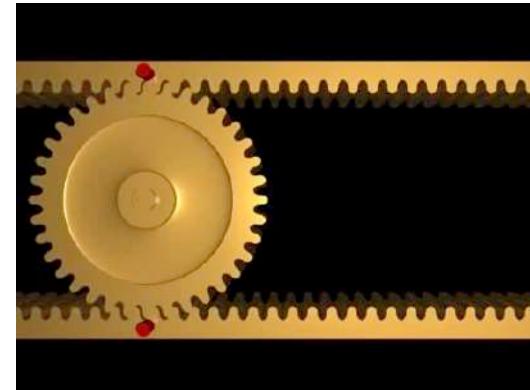
- **spur gears:** modify direction and/or translate axis of (rotational or translational) motor displacement
 - problems: **deformations, backlash**
- **lead screws, worm gearing:** convert rotational into translational motion (prismatic joints)
 - problems: **friction, elasticity, backlash**
- **toothed belts and chains:** dislocate the motor w.r.t. the joint axis
 - problems: **compliance** (belts) or **vibrations** induced by larger mass at high speed (chains)
- **harmonic drives:** compact, in-line, power efficient, with high reduction ratio (up to 150-200:1)
 - problems: **elasticity**
- **transmission shafts:** long, inside the links, with flexible couplings for alignment



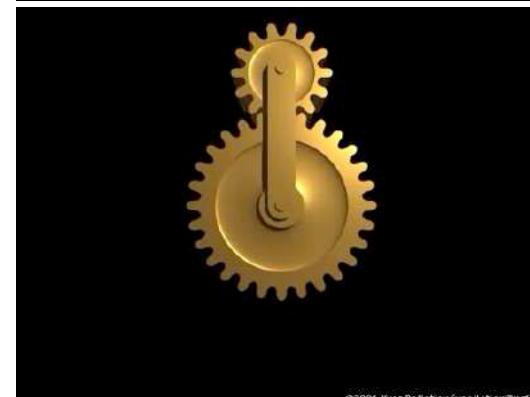


Transmission gears in motion

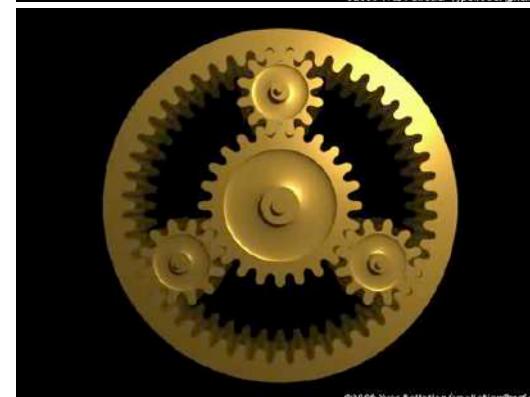
- racks and pinion
 - one rack moving (or both)
- epi-cycloidal gear train
 - or hypo-cycloidal (small gear inside)
- planetary gear set
 - one of three components is locked:
sun gear, planet carrier, ring gear



video



video

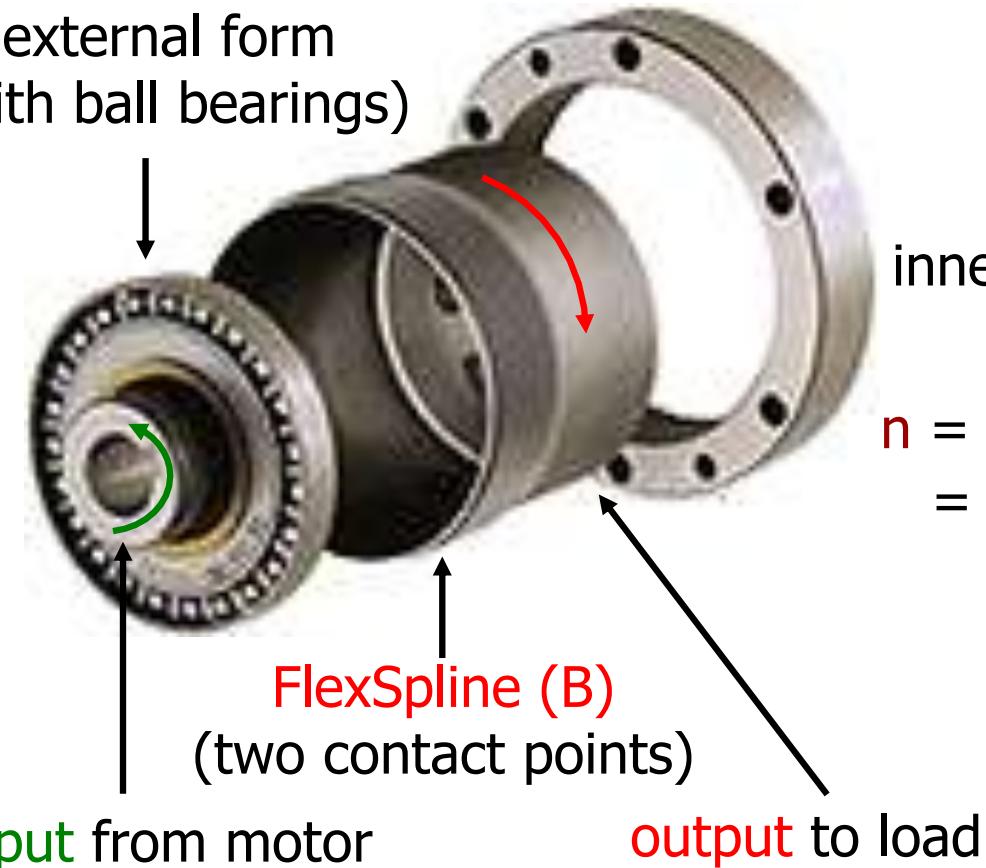


video



Harmonic drives

Wave Generator (C)
of slightly elliptic
external form
(with ball bearings)



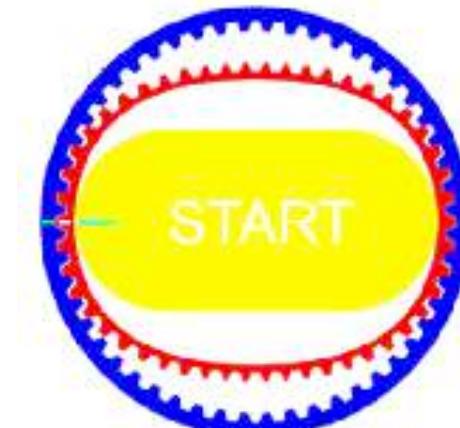
Circular Spline (A)



inner #teeth CS = outer #teeth FS + 2

reduction ratio

$$n = \frac{\text{#teeth FS}}{(\text{#teeth CS} - \text{#teeth FS})} = \frac{\text{#teeth FS}}{2}$$





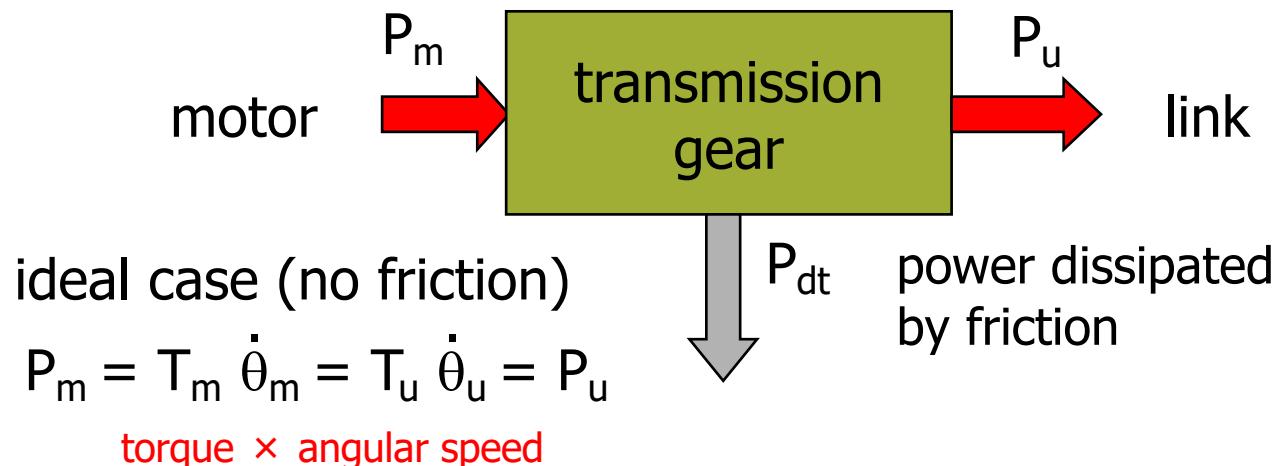
Operation of an harmonic drive

Harmonic Drive Gearing
PRINCIPLE *of* OPERATION

commercial video by Harmonic Drives AG



Optimal choice of reduction ratio



$$n = \text{reduction ratio } (\gg 1) \quad \dot{\theta}_m = n \dot{\theta}_u \quad \rightarrow \quad T_u = n T_m$$

to have $\ddot{\theta}_u = a$ (thus $\ddot{\theta}_m = n a$), the motor should provide a torque

$$T_m = J_m \ddot{\theta}_m + 1/n (J_u \ddot{\theta}_u) = (J_m n + J_u/n) a$$

inertia × angular acceleration

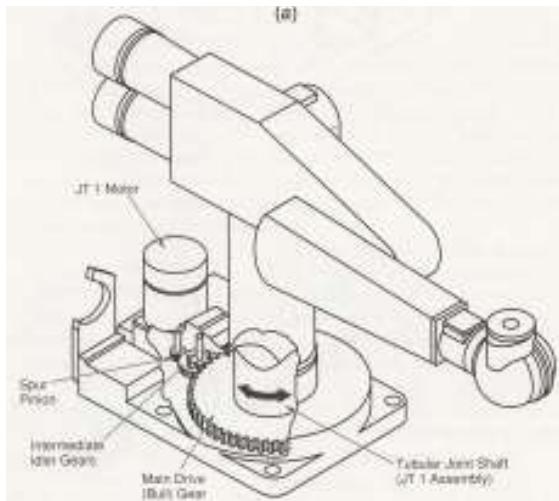
for minimizing T_m , we set: $\frac{\partial T_m}{\partial n} = (J_m - J_u/n^2) a = 0$

→ $n = (J_u / J_m)^{1/2}$ “matching” condition between inertias

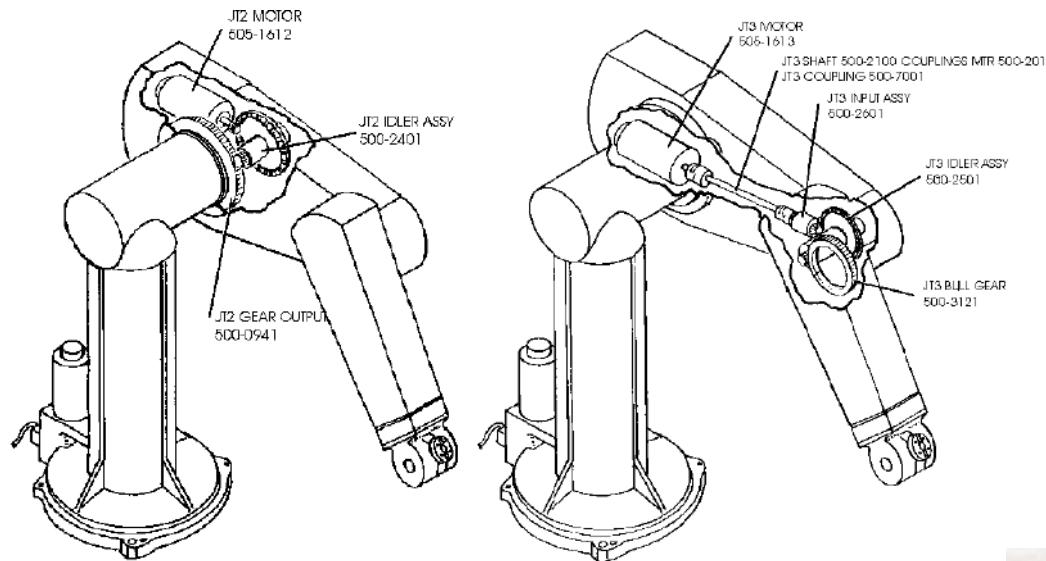


Transmissions in industrial robots

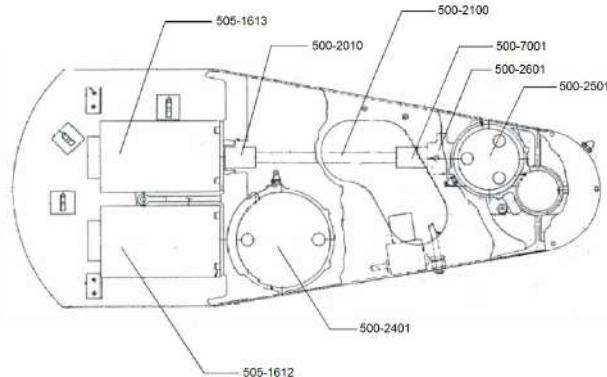
- transmissions used (inside) 6-dof Unimation industrial robots with serial kinematics



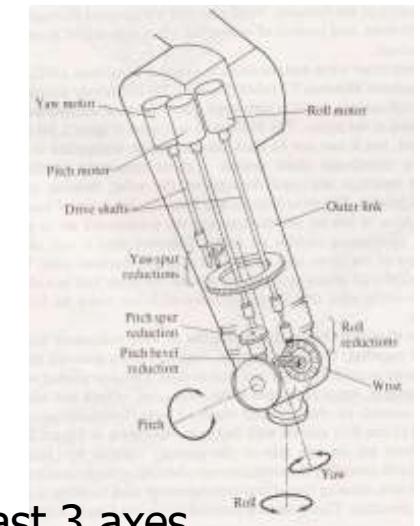
PUMA 260: 1st axis



PUMA 560: 2nd and 3rd axes



PUMA 560: inner and outer links



PUMA 560: last 3 axes

Inside views on joint axes 4, 5 & 6 of an industrial KUKA robot



- looking inside the forearm to see the transmissions of the spherical wrist
- motor rotation seen from the encoder side (small couplings exist)



Robotics 1

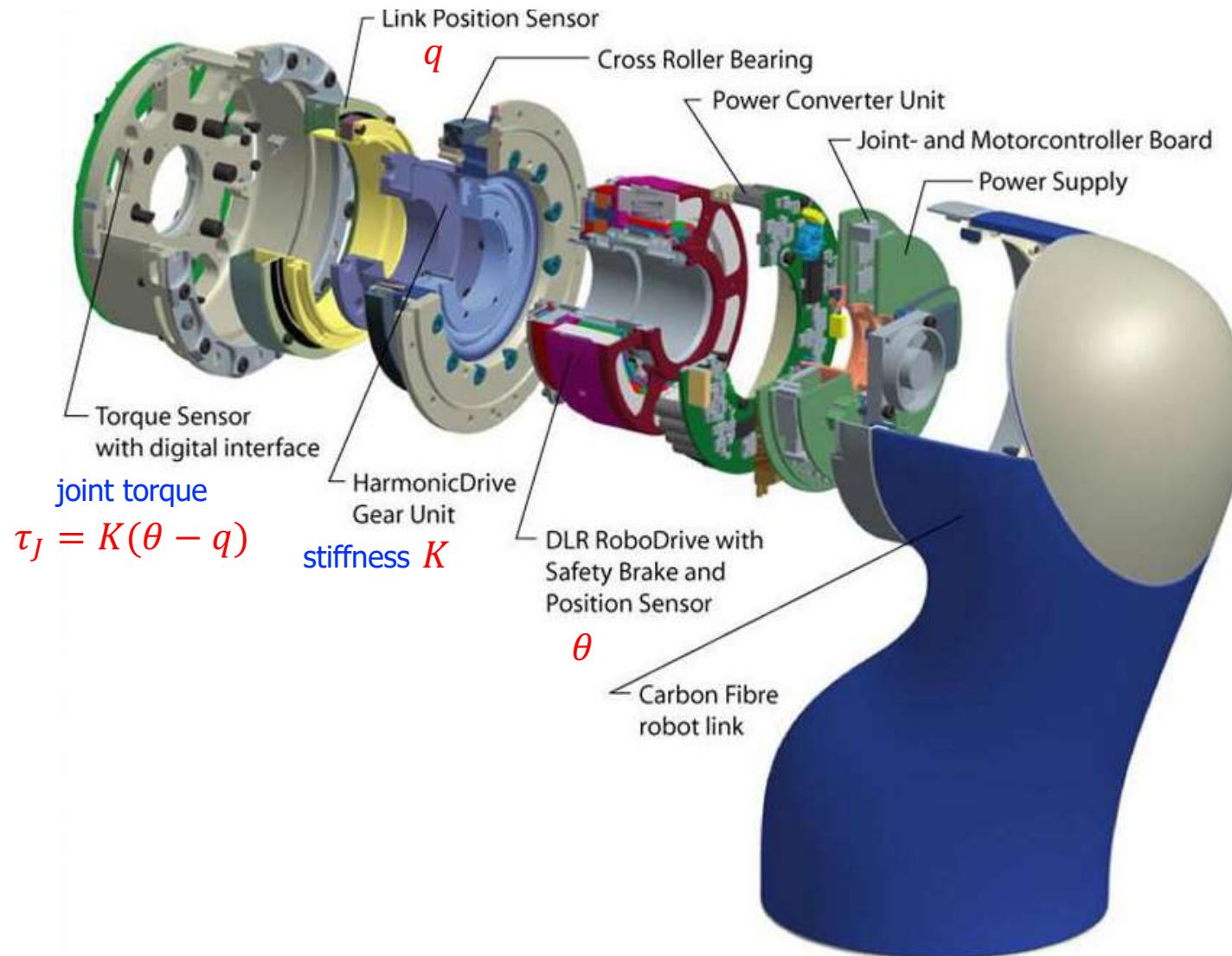
video



video



Exploded view of a joint in the DLR-III robot





Robotics 1

Robot components: Proprioceptive sensors

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Properties of measurement systems - 1

- **accuracy**

agreement of measured values with a given reference standard (e.g., ideal characteristics)

- **repeatability**

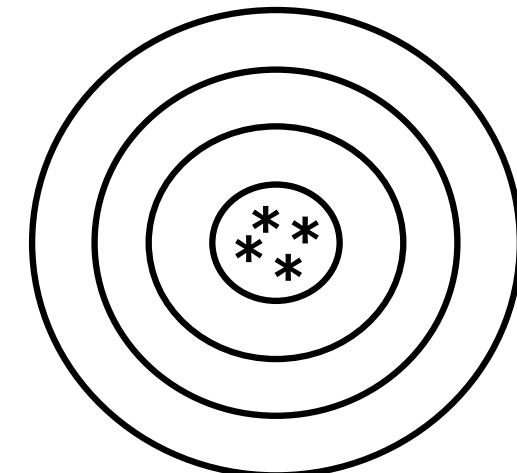
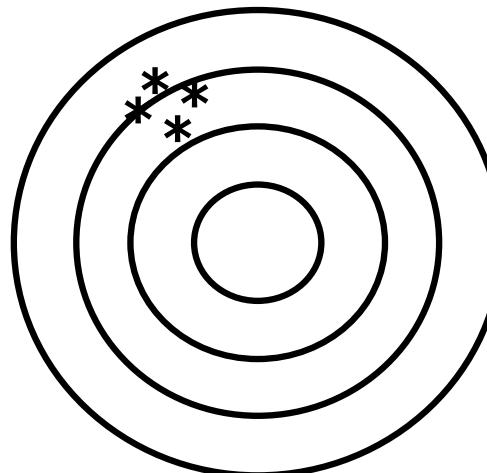
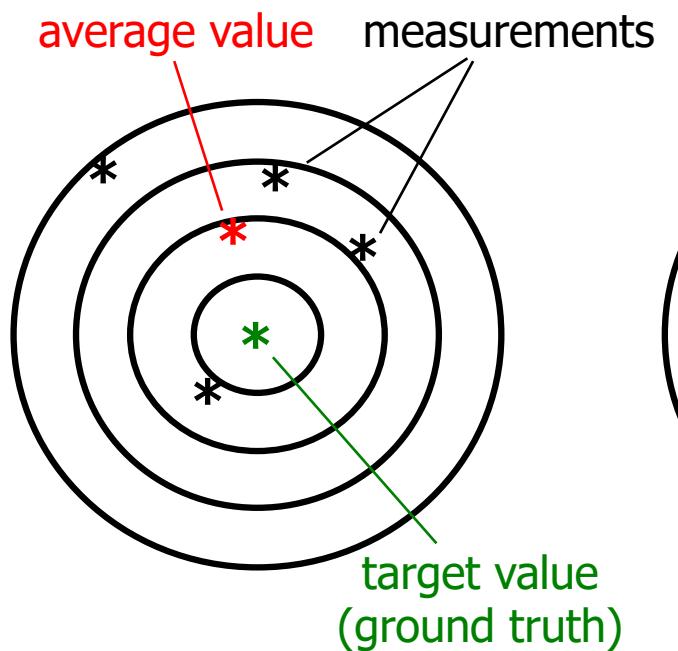
capability of reproducing as output similar measured values over consecutive measurements of the same constant input quantity

- **stability**

capability of keeping the same measuring characteristics over time/temperature (similar to accuracy, but in the long run)



Accuracy and Repeatability



low accuracy
low repeatability

low accuracy
high repeatability

high accuracy
high repeatability

better components!

calibration!

Accuracy and Repeatability in robotics



- **accuracy** is how close a robot can come to a given point in its workspace
 - depends on machining accuracy in construction/assembly of the robot, flexibility effects of the links, gear backlash, payload changes, round-off errors in control computations, ...
 - can be improved by (**kinematic**) **calibration**
- **repeatability** is how close a robot can return to a previously taught point
 - depends only the robot controller/measurement resolution
- both may vary in different areas of the robot workspace
 - standard ISO 9283 defines conditions for assessing robot performance
 - limited to static situations (recently, interest also in dynamic motion)
 - robot manufacturers usually provide only data on “repeatability”

[video](#)



simple test on repeatability of a
Fanuc ArcMate100i robot (1.3 m reach)



Properties of measurement systems - 2

- **linearity** error

maximum deviation of the measured output from the straight line that best fits the real characteristics

- as % of the output (measurement) range

- **offset** error

value of the measured output for zero input

- sometimes not zero after an operation cycle, due to **hysteresis**

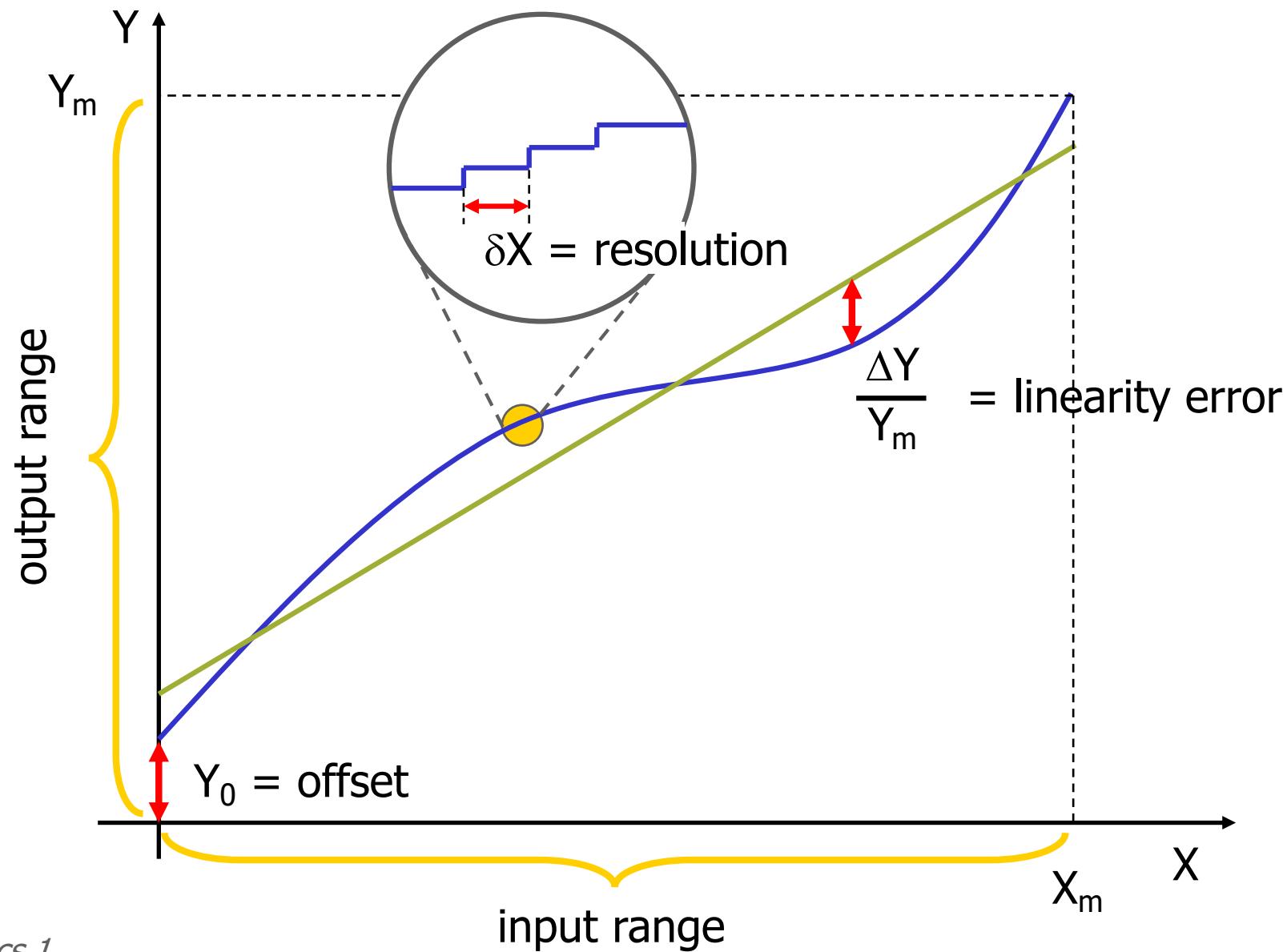
- **resolution** error

maximum variation of the input quantity producing no variation of the measured output

- in absolute value or in % of the input range



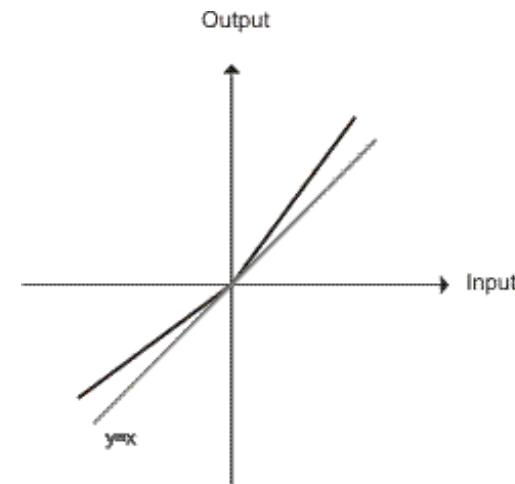
Linearity, Offset, Resolution



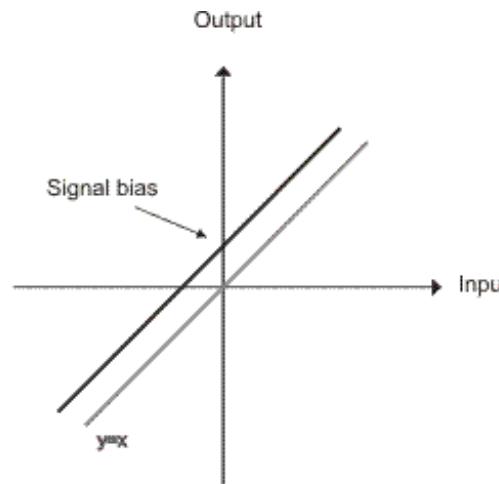


Sensor measurements

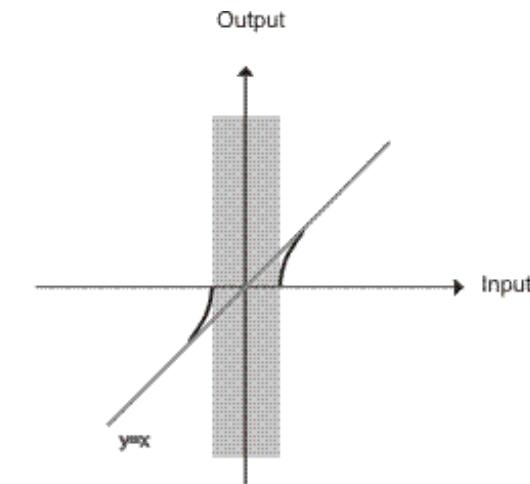
some non-idealities



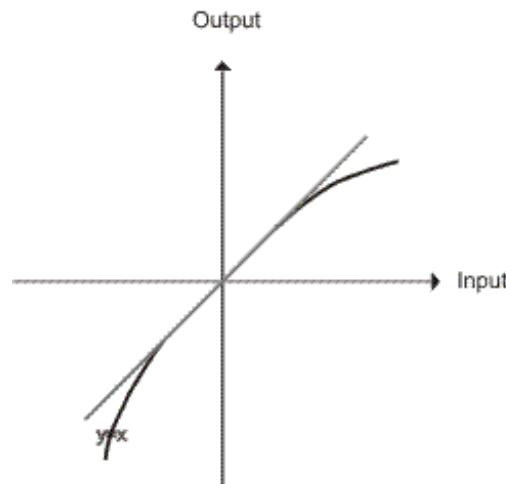
Asymmetry



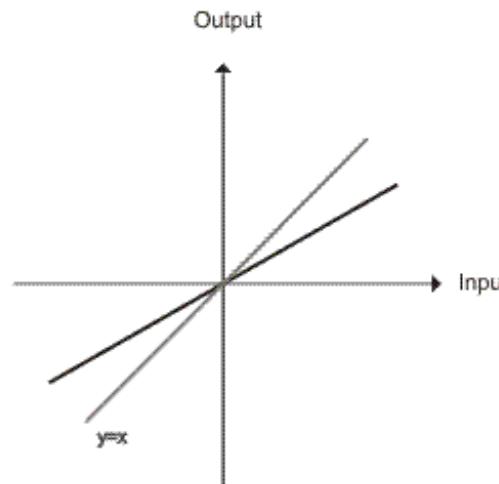
Bias



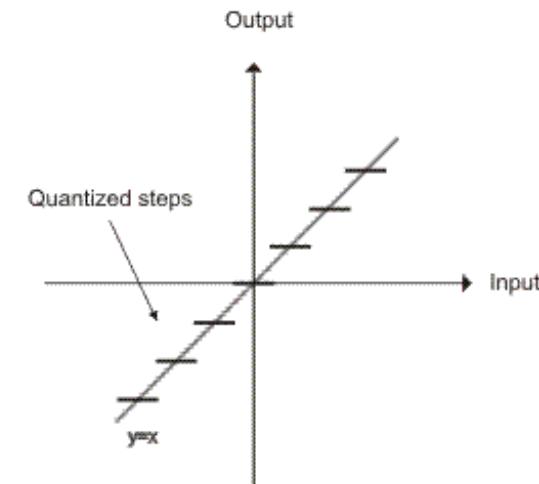
Dead zone



Nonlinearity



Scaling factor



Quantization



Classes of sensors for robots

- **proprioceptive sensors** measure the internal state of the robot (position and velocity of joints, but also torque at joints or acceleration of links)
 - kinematic calibration, identification of dynamic parameters, control
- **exteroceptive sensors** measure/characterize robot interaction with the environment, enhancing its autonomy (forces/torques, proximity, vision, but also sensors for sound, smoke, humidity, ...)
 - control of interaction with the environment, obstacle avoidance in the workspace, presence of objects to be grasped, ...
 - mobile-base robots: localization in a map, navigation in unknown environments, ...



Position sensors

- provide an **electrical signal proportional to the displacement** (linear or angular) of a mechanical part with respect to a reference position
- **linear** displacements: potentiometers, linear variable-differential transformers (LVDT), inductosyns
- **angular** displacements: potentiometers, resolvers, syncros (all analog devices with A/D conversion), optical **encoders (digital)**, Hall sensors, ...

the most used in robotics, since also linear displacements are obtained through rotating motors and suitable transmissions





Absolute encoders

Photo-emitter

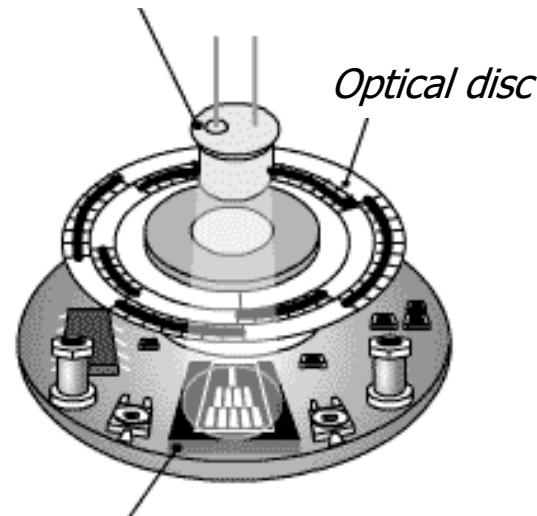
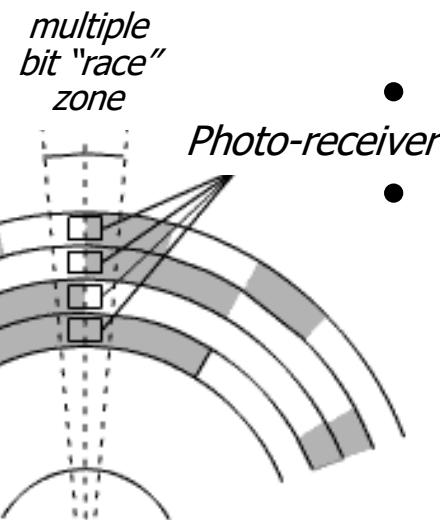


Photo-receiver

$$N_t = \# \text{ tracks} = \# \text{ bits} \\ (\text{min } 12 \text{ in robotics})$$

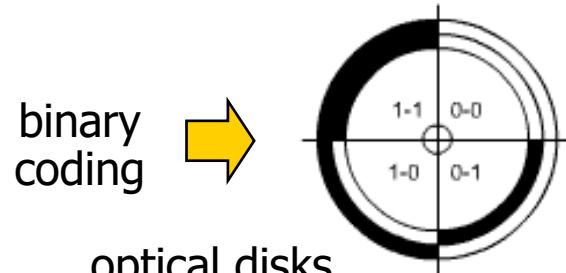


- rotating optical disk, with alternate transparent and opaque sectors on multiple concentric tracks
- (infrared) light beams are emitted by leds and sensed by photo-receivers
- light pulses are converted into electrical pulses, electronically processed and transmitted in output
- **resolution** = $360^\circ / 2^{N_t}$
- digital encoding of **absolute** position

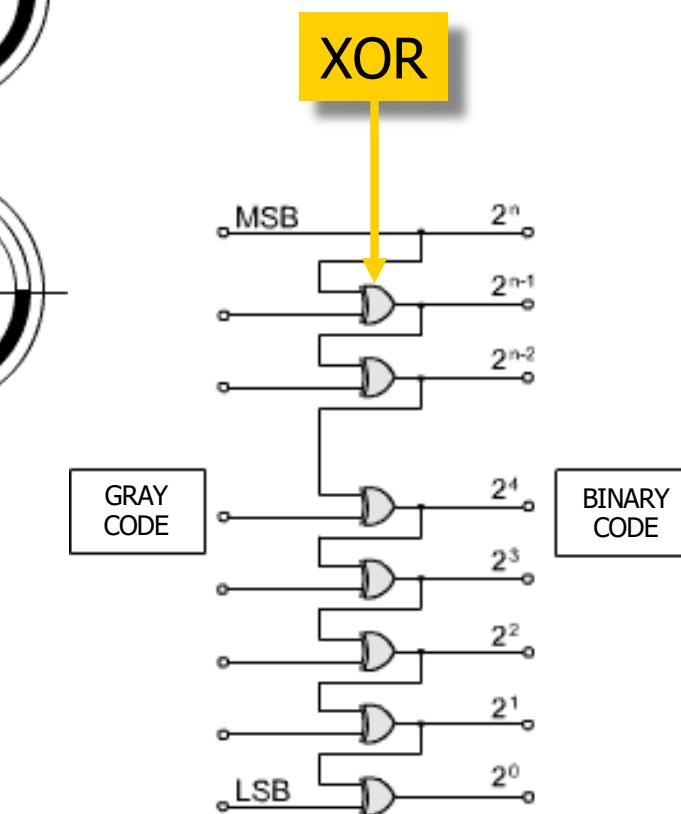
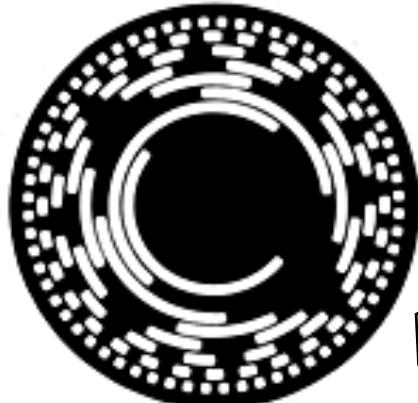
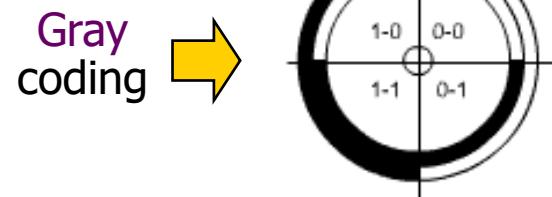
when the optical disk is rotating fast, the use of **binary coding** may lead to (large) reading errors, in correspondence to multiple transitions of bits



Absolute encoding



optical disks
with **2 bits**



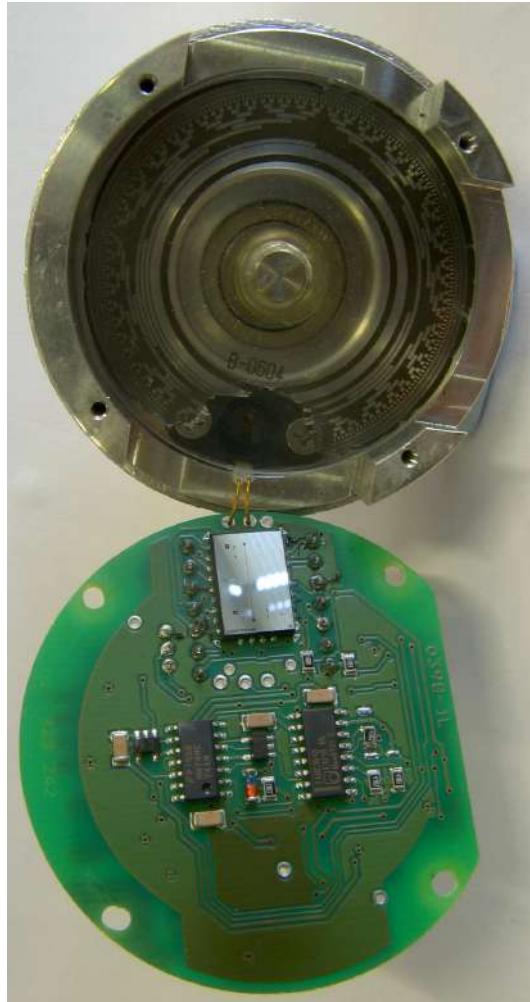
8-bit Gray-coded
absolute encoder

DECIMAL	BINARY	GRAY
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

adjacent codes differ
by just one bit



Use of absolute encoders



13-bit absolute encoder opened:
Gray-coded disk and electronics

- ready to measure at start (no “homing”)
- two modes for permanent operation
 - when switching off the drive, position parameters are saved on a flash memory (and brakes activated)
 - battery for the absolute encoder is always active, and measures position even when the drive is off
 - data memory > 20 years
- single-turn or multi-turn versions, e.g.
 - 13-bit single-turn has $2^{13} = 8192$ steps per revolution (resolution = 0.044°)
 - 29-bit multi-turn has 8192 steps/revolution + counts up to $2^{16} = 65536$ revolutions
- aluminum case with possible interface to field bus systems (e.g., CANopen or PROFIBUS)
- typical supply 5/28V DC @1.2 W



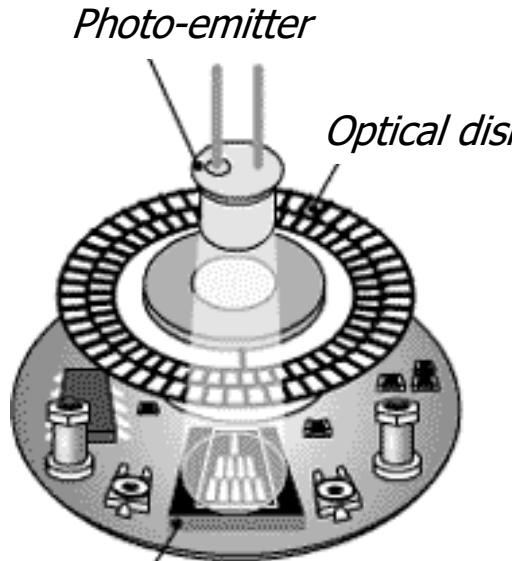
hollow shaft



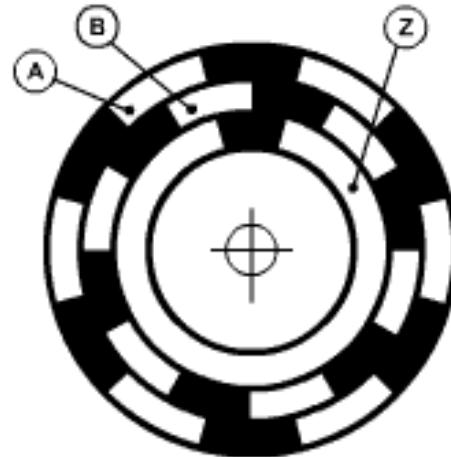
round flange



Incremental encoders



The three tracks
on an optical disk
(here $N_e = 6$)

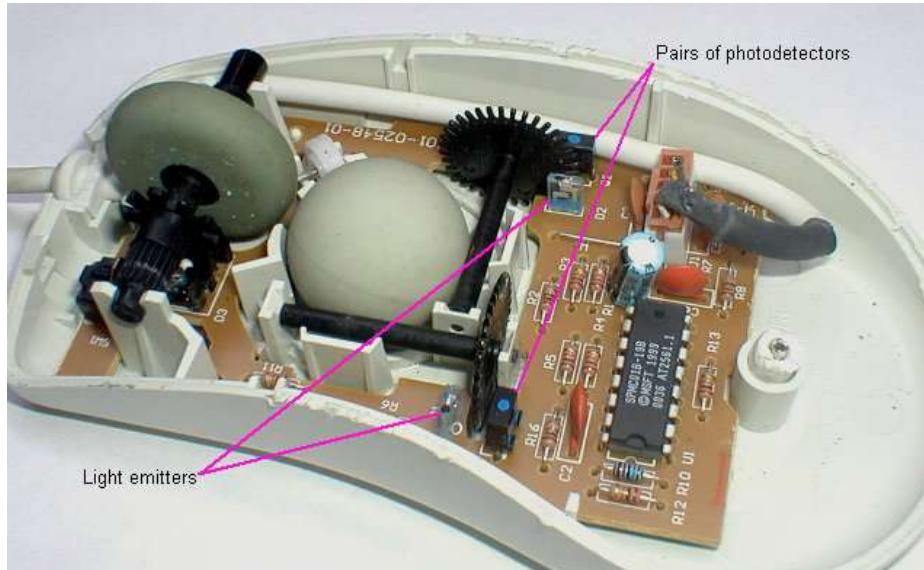


- optical rotating disk with three tracks, alternating transparent and opaque areas: measures **incremental** angular displacements by counting trains of N_e pulses ("counts") per turn ($N_e = 100 \div 5000$)
 - the two A and B tracks (**channels**) are in quadrature (phase shift of 90° electrical), allowing to detect the direction of rotation
 - a third track Z is used to define the "0" reference position, with a reset of the counter (**needs "homing"** at start)
 - some encoders provide as output also the three phases needed for the switching circuit of brushless motors



Incremental encoders

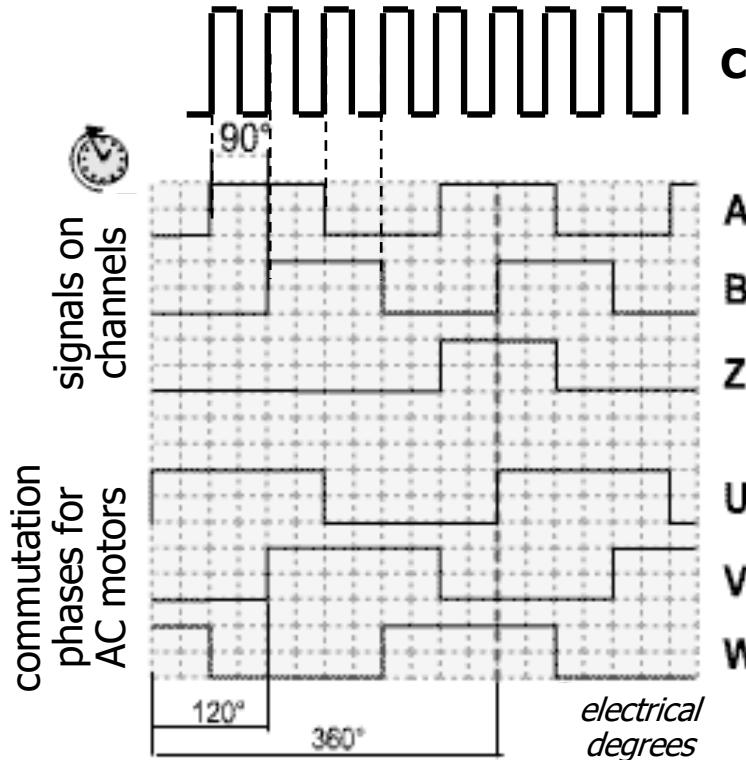
- two (cheap) incremental encoders inside a mouse
- a OMRON incremental encoder with 2000 pulses/turn



diameter $\varnothing 40$ mm
mass $m \approx 100$ g
inertia $J = 1 \cdot 10^{-6}$ kg m²

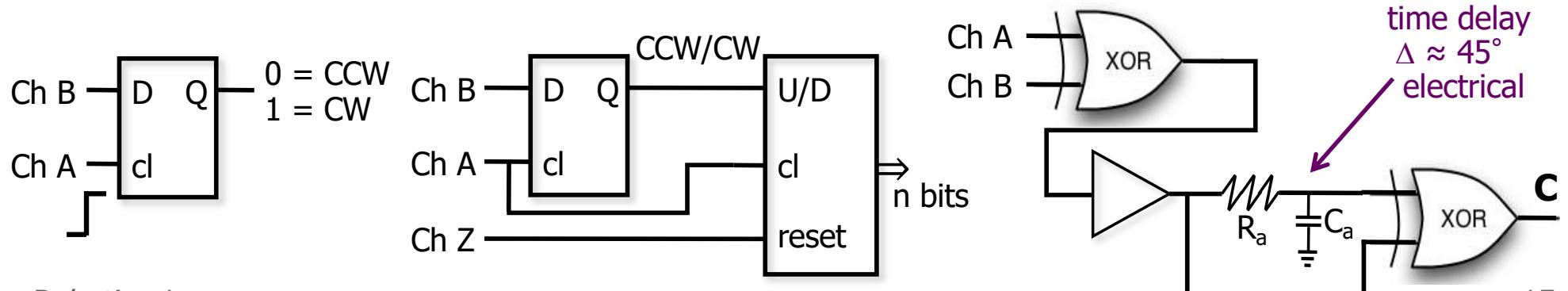


Signal processing



c

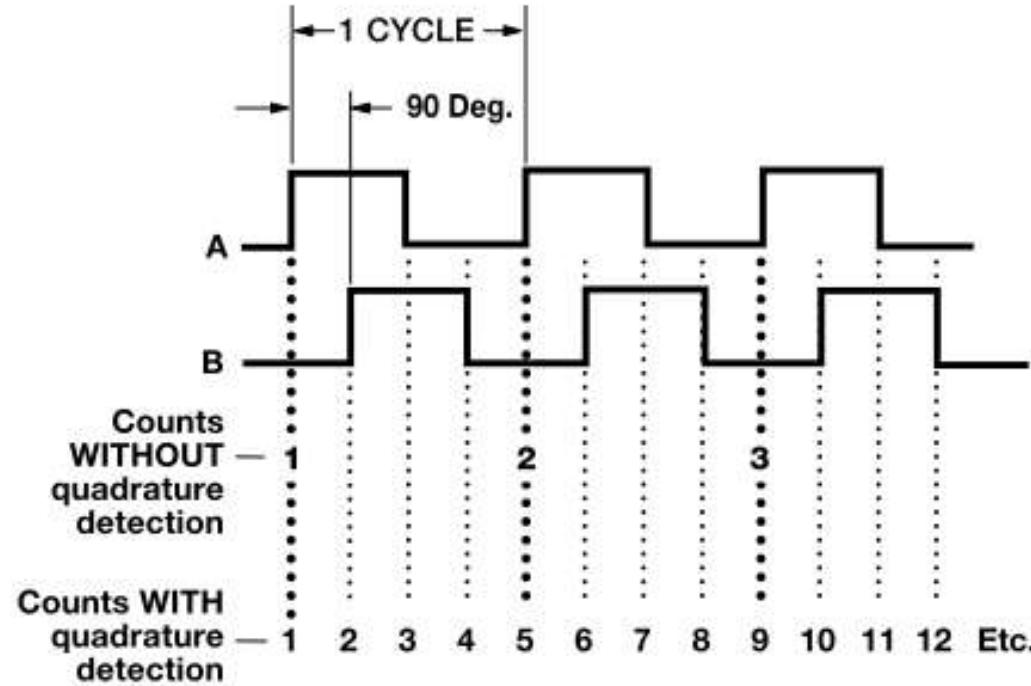
- “fractions of a cycle” of each pulse train are measured in “electrical degrees”
- $1^\circ \text{ electrical} = 1^\circ \text{ mechanical}/N_e$
 $360^\circ \text{ mechanical} = 1 \text{ turn}$
- signals are fed in a digital **counter**, with a **D-type** flip-flop to sense direction + **reset**
- to **improve resolution** ($4 \times$), the leading and trailing edges of signals A and B are used
- the sequence of pulses C will clock now the counter (**increments or decrements**)





Count multiplication

example of quadrature detection

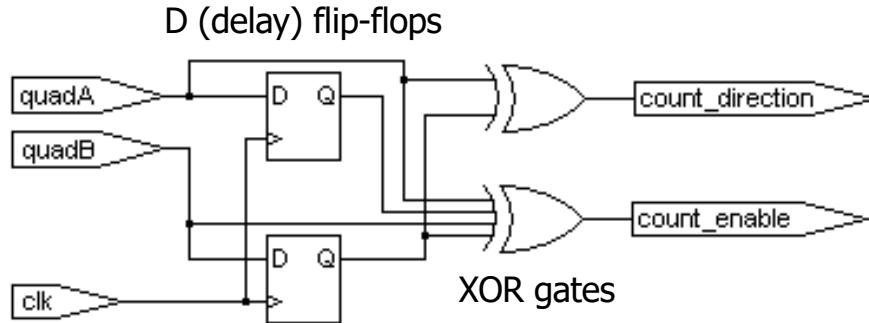


- an incremental encoder with $N_e = 2000$ (electrical) cycles provides a count of $N = 8000$ pulses/turn after electronic multiplication
- its final **resolution** is (mechanical) $360^\circ/8000 = .045^\circ$ ($= 0^\circ 2' 42''$)
- needs a 13-bit counter to cover a full turn without reset ($2^{13} = 8192$)

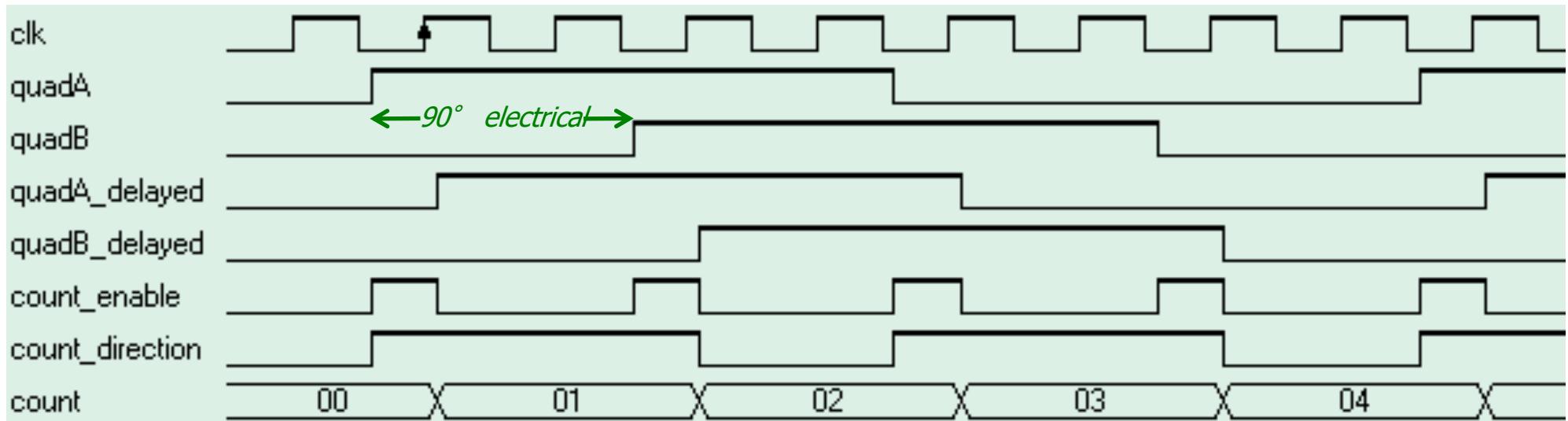


Quadrature detection in incremental encoders

a more complete implementation



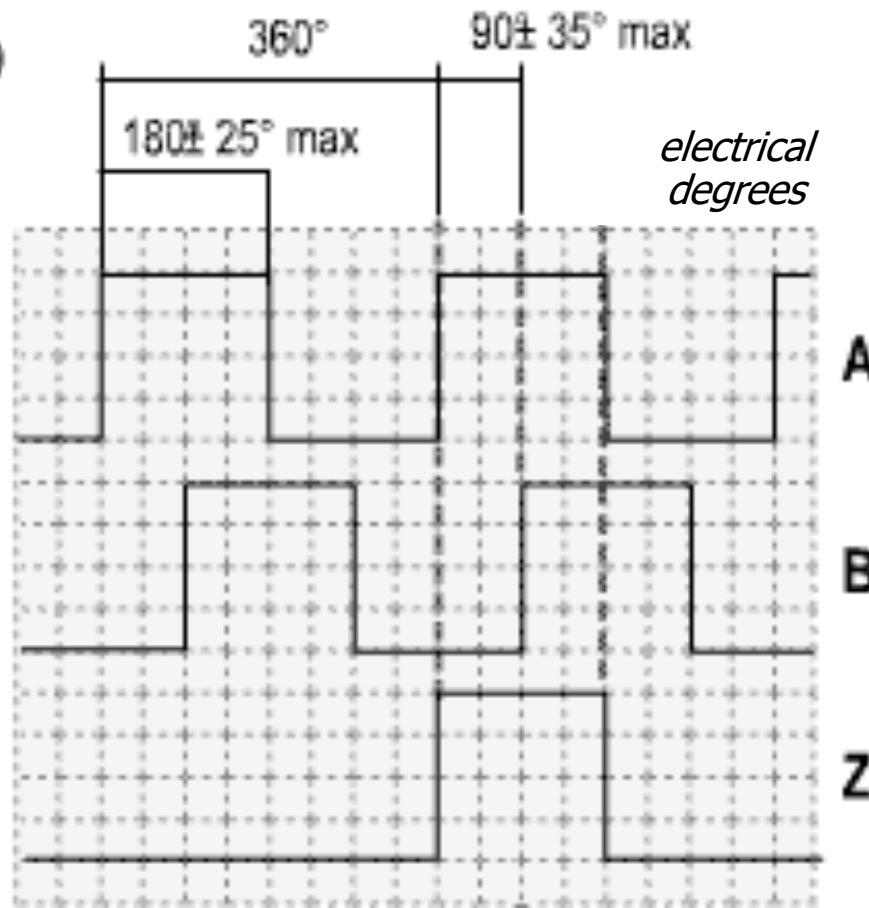
NOTE: since in practice A and B signals may **not** be synchronous to the clock signal, two extra D flip-flops per input should be used to avoid meta-stable states in the counters



- it is assumed that an oversampling clock "clk" (e.g., as provided by a FPGA) is available, which is faster than the two quadrature signals A and B
- the digital count output will have a **resolution** multiplied by 4



Accuracy in incremental encoders



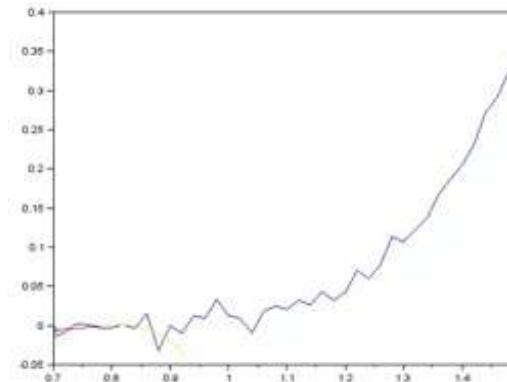
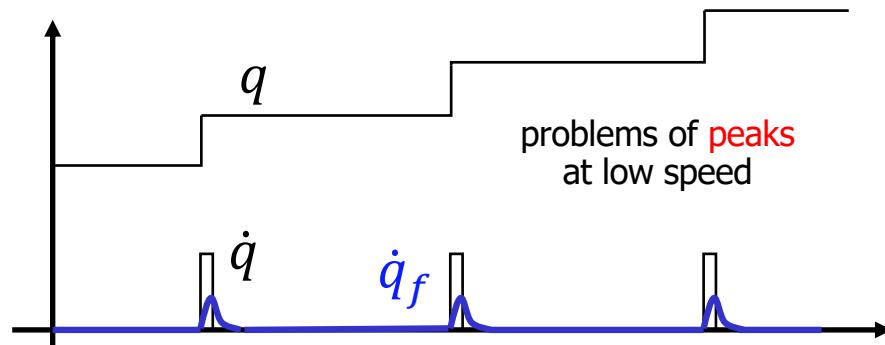
...apart from
quantization errors

- division error: maximum displacement between two consecutive leading/trailing edges, typically within $\max \pm 25^\circ$ electrical
- the phase shift of the two channels, nominally equal to 90° electrical, is typically within $\max \pm 35^\circ$ electrical (quadrature error)



Indirect measure of velocity

- numerical differentiation of digital measures of position
 - to be realized **on line** with Backward Differentiation Formulas (BDFs)
 - 1-step BDF (Euler): $\dot{q}_k = \dot{q}(kT) = \frac{1}{T}(q_k - q_{k-1}) \Leftrightarrow \dot{q}_k = \frac{\Delta q_k}{T} \Leftarrow$ directly from incremental encoder
 - 4-step BDF: $\dot{q}_k = \frac{1}{T} \left(\frac{25}{12} q_k - 4q_{k-1} + 3q_{k-2} - \frac{4}{3} q_{k-3} + \frac{1}{4} q_{k-4} \right)$
- convolution **filtering** is needed because of noise and position quantization
 - use of **non-causal** filters (e.g., Savitzky-Golay) helps, **but** introduces delays
- **Kalman filter** for on line state estimation (**optimal**, assuming Gaussian noise)



animation of Savitzky-Golay filter
with cubic polynomials



Kinematic Kalman Filter for velocity estimation

motion and
sensing
discrete-time
model for
estimation

noisy position measure
(encoder output)

$$\begin{aligned}\xi(k) &= \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \xi(k-1) + \mu \\ z(k) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \xi(k) + \nu\end{aligned}$$

zero mean
Gaussian noises
with (co)variances
 Q (a matrix) and R

T = sampling time

$$\xi(k) = (x(k) \dot{x}(k))^T$$

actual state

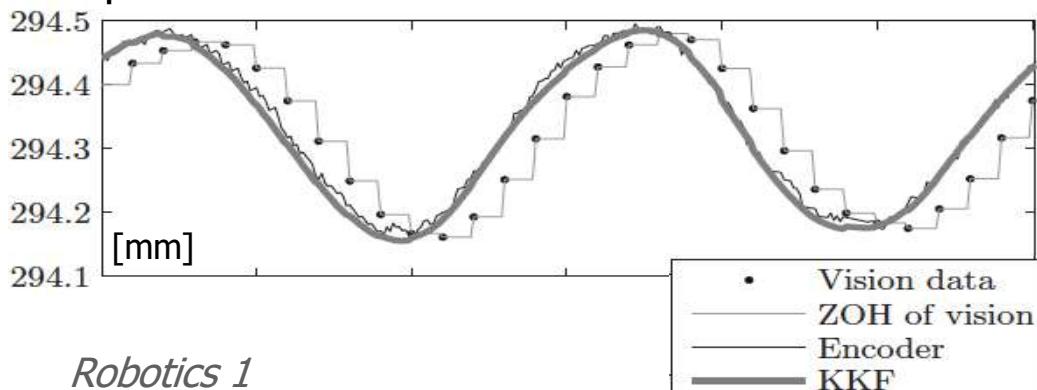
unmeasured
velocity

design a (linear) **Kalman filter** providing an estimate $\hat{\xi}(k)$ of the model state

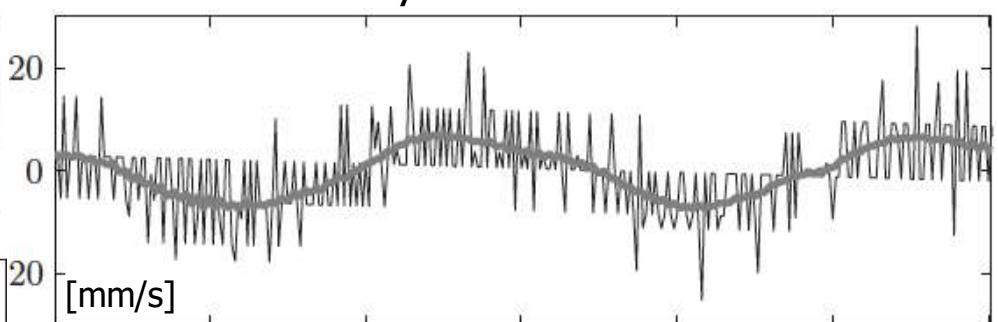
$$\hat{\xi}(k) = \underbrace{\begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \hat{\xi}(k-1)}_{\text{(a priori) prediction}} + K_k \underbrace{\left(z(k) - \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \hat{\xi}(k-1) \right)}_{\text{correction (based on the measured output)}}$$

using the **optimal** Kalman gain K_k

position measure and its filtered version



numerical velocity and its filtered estimate

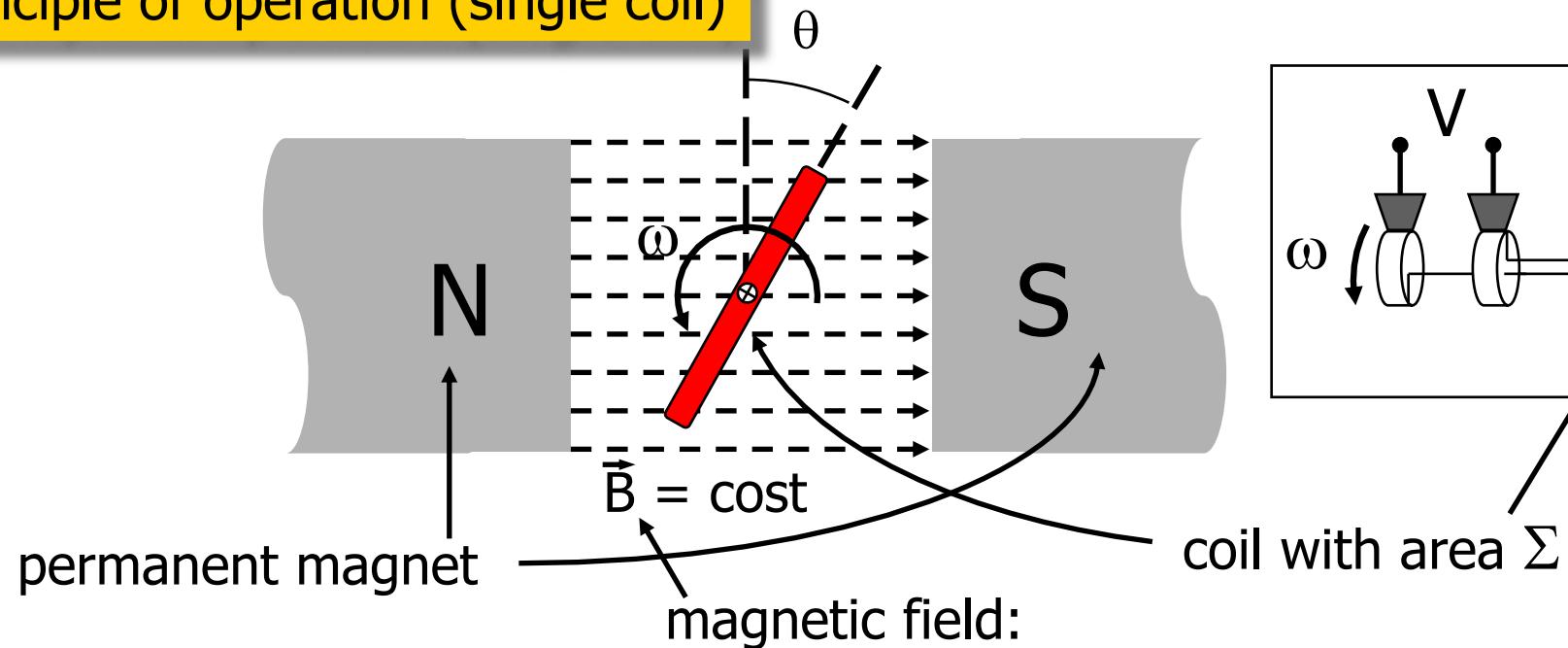




Velocity sensor: Tachometer

always mounted on the (electrical) motor axis

principle of operation (single coil)



$$V = -d\Phi/dt = |\vec{B}|\Sigma \omega \sin \omega t$$

amplitude $V \propto \omega$

⇒ to reduce ripples, use m coils rotated regularly by $180^\circ/m$



DC tachometer

an example



- Servo-Tek Tach Generator (B series)
- bi-directional
- output voltage 11 ÷ 24 V @1000 RPM
- low ripple: < 3% peak-to-peak of DC value (with 72 KHz filter)
- weight = 113 g, diameter = 2.9 cm
- linearity error < 0.1% (at any speed)
- stability 0.1% (w.r.t. temperature)

B-Series Specifications

Model Number	Mounting	Weight (approx)	Inertia (approx) oz-in.-sec ²	V/1,000 RPM	RPM (max)	Driving Torque (max)	Arm R (ohms dynamic)	Arm Ind (h)
SA-740B-1*	Face	4.0 oz	2.27×10^{-6}	20.8 V	8,000	0.25 oz-in.	1000	0.56
SB-740B-1*	Flange	4.0 oz	2.27×10^{-6}	20.8 V	8,000	0.25 oz-in.	1000	0.56
SA-757B-1*	Face	4.0 oz	2.27×10^{-6}	20.8 V	8,000	0.25 oz-in.	1000	0.56
SB-757B-1*	Flange	4.0 oz	2.27×10^{-6}	20.8 V	8,000	0.25 oz-in.	1000	0.56

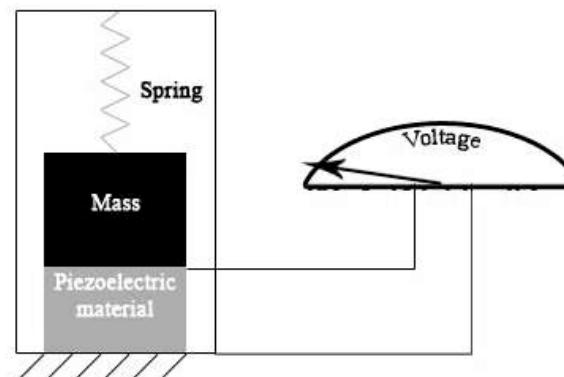
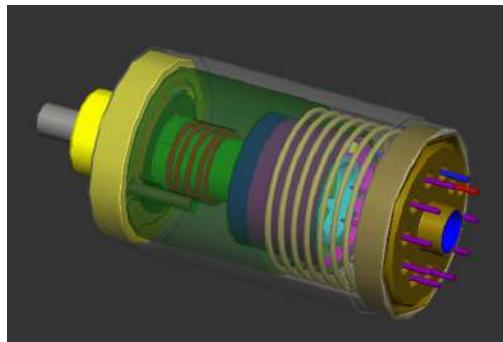


1.75 mNm (as a load)



Accelerometers

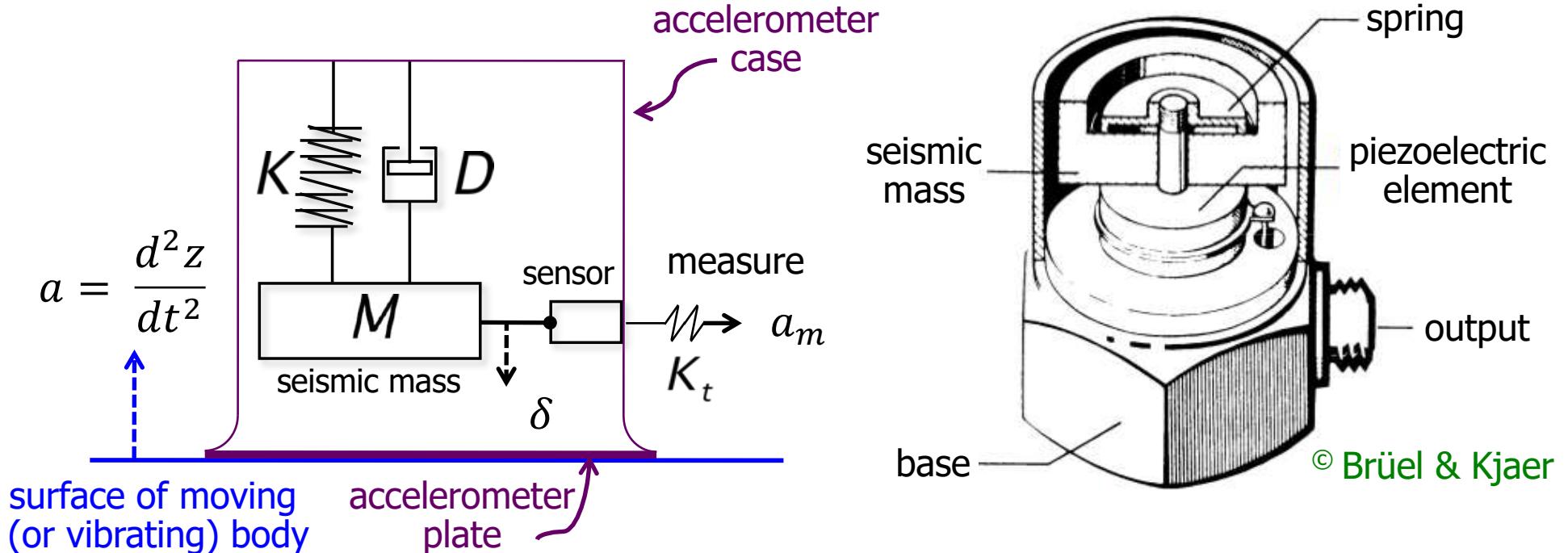
- measure of linear acceleration based on **inertial forces** (no “touch”)
 - units: [m/s²] or gravitational acceleration [g] (non-SI unit: 1g ≈ 9.81 m/s²)
- different principles for converting mechanical motion in an electrical signal
 - **piezoelectric**: piezoceramics (PZT) or crystals (quartz), better linearity & stability, wide dynamic range up to high frequencies, no moving parts, no power needed
 - **piezoresistive**: for high-shocks, measures also static acceleration (g_0), needs supply
 - **capacitive**: silicon micro-machined sensing element, superior in static to low frequency range, can be operated in servo mode, cheap but limited resolution
 - modern solution: small **MEMS** (Micro Electro-Mechanical Systems)
- multiple applications: from vibration analysis to long range navigation



animation of
measurement principle
in a piezoelectric
accelerometer



Operation principle seismic accelerometer



$$Ma = M\ddot{\delta} + D\dot{\delta} + K\delta$$

$$a_m = K_t\delta$$

by Laplace transform

$$\frac{A_m(s)}{A(s)} = K_t \frac{M}{Ms^2 + Ds + K}$$

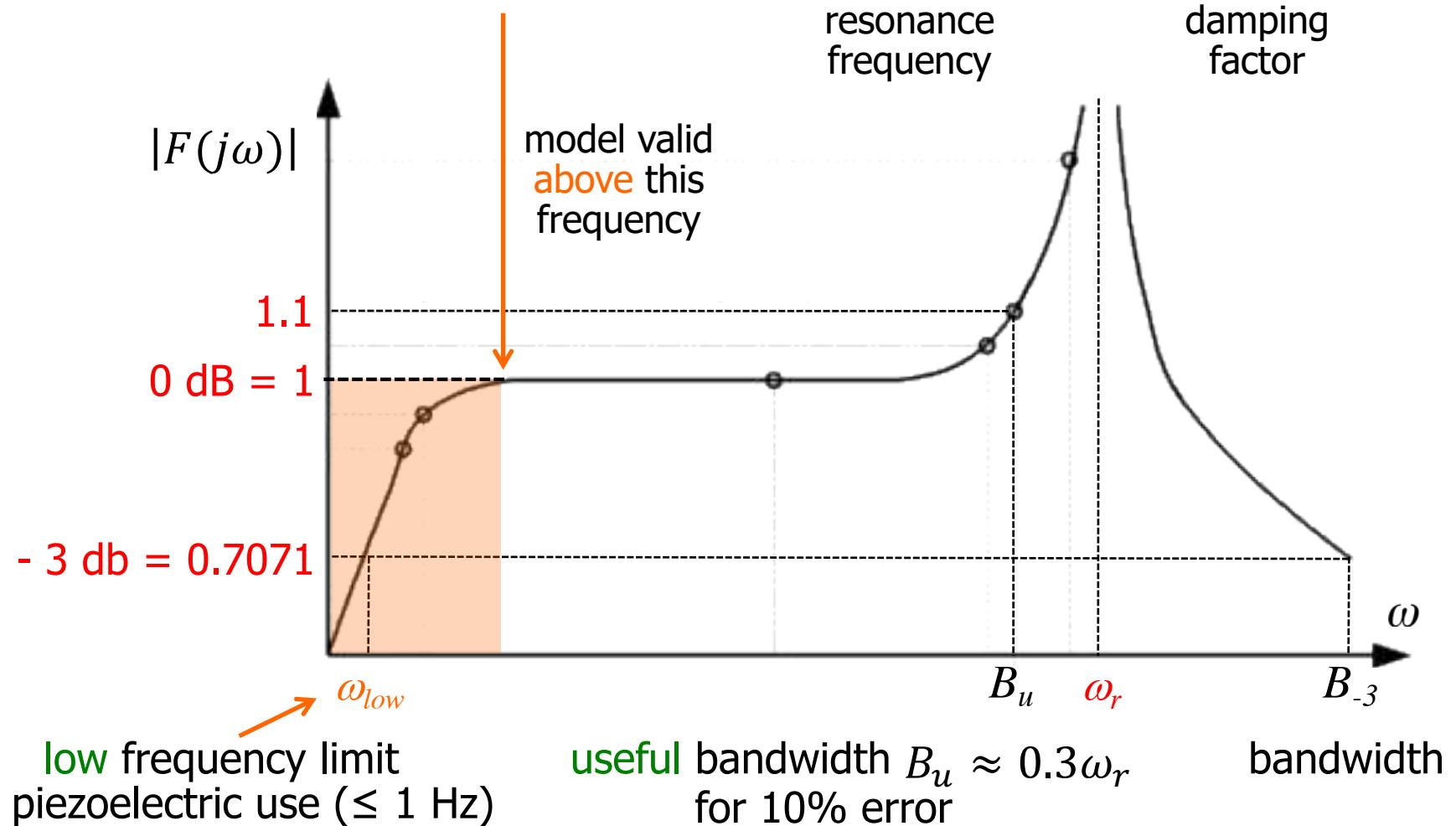
$$= \frac{K_t}{s^2 + (D/M)s + (K/M)}$$



Frequency characteristics of a piezoelectric accelerometer

$$F(s) = \frac{A_m(s)}{A(s)} = \frac{K_t}{s^2 + (D/M)s + (K/M)}$$

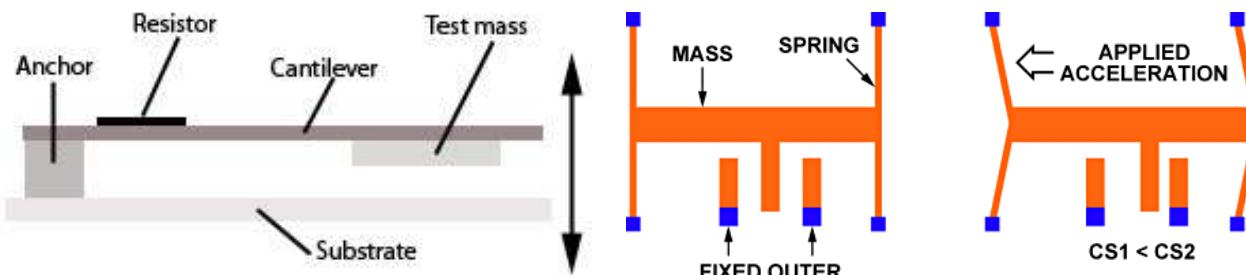
$$\omega_r = \sqrt{K/M} \quad \zeta = \frac{D}{2} \sqrt{1/KM}$$



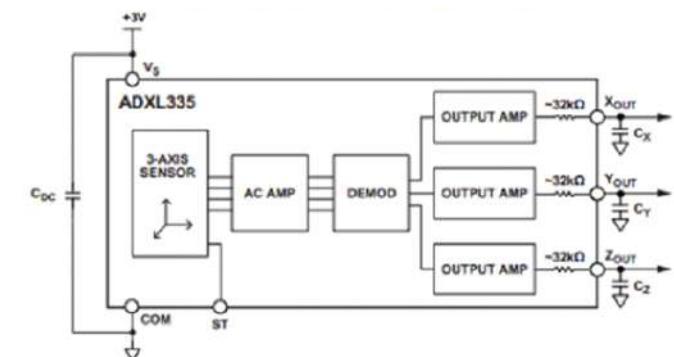
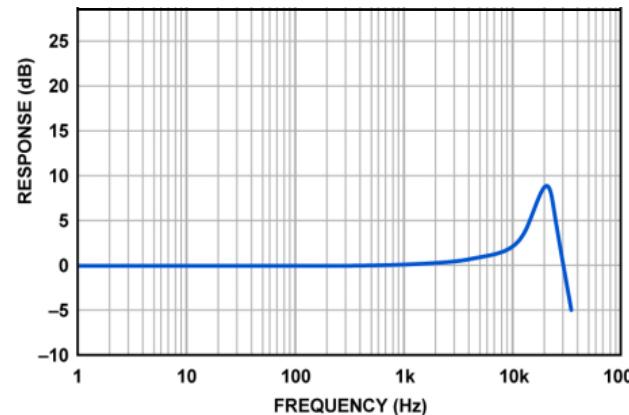
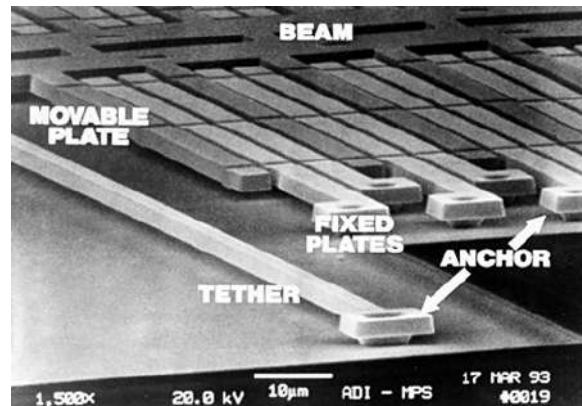


MEMS accelerometers

- very simple MEMS (a **cantilever** beam with a **test mass**, with damping from the residual gas sealed in the device), single- or **tri-axial**, very small and light
- cross-couplings** among acceleration sensing directions should be limited $\leq 3\%$



ADXL335 3-axis, small, low power, $\pm 3g$, with signal conditioned voltage outputs

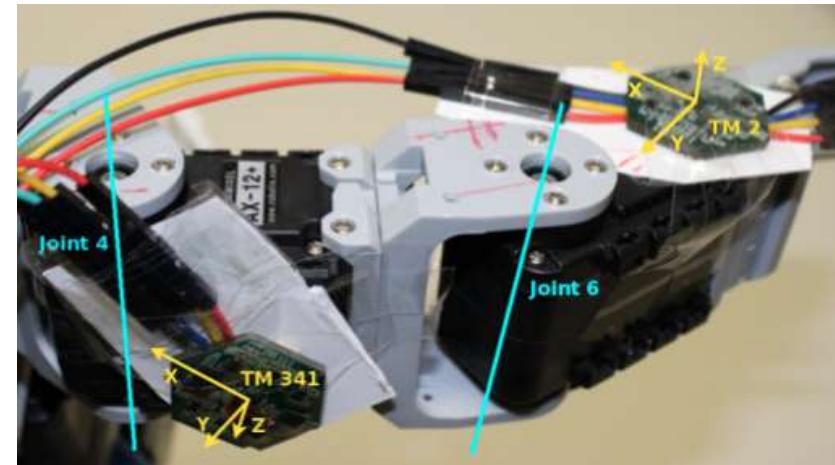
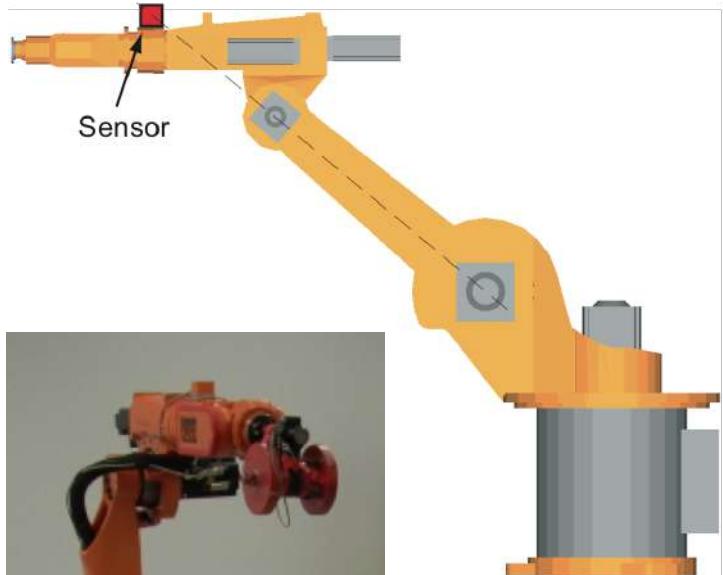




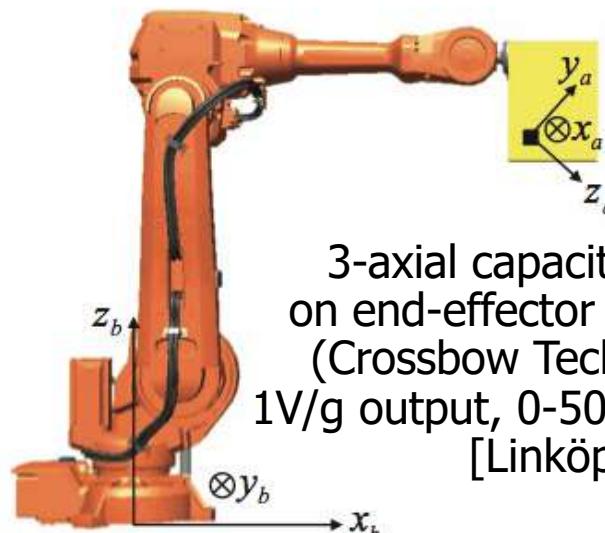
Mounting accelerometers on robots



3-axial MEMS
accelerometer
on the forearm
of a **KUKA KR15/2**
[DLR/Sapienza, 2007]



Bosch BMA 150 3-axial accelerometers
integrated in two larger Tactile Modules on the
links of a **Bioloid humanoid left arm** [TUM, 2011]



3-axial capacitive accelerometer
on end-effector tool of an **ABB robot**
(Crossbow Technology: 2g range,
1V/g output, 0-50 Hz, $\pm 2^\circ$ align error)
[Linköping, 2012]



Robotics 1

Robot components: Exteroceptive sensors

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Summary

- force sensors
 - strain gauges and joint torque sensor
 - 6D force/torque (F/T) sensor at robot wrist
 - RCC = Remote Center of Compliance (*not a sensor, but similar...*)
- proximity/distance sensors (⇒ moved to AMR course!)
 - infrared (IF)
 - ultrasound (US)
 - laser
 - with structured light
- vision
- examples of robot sensor equipment
- some **videos** intertwined, with applications

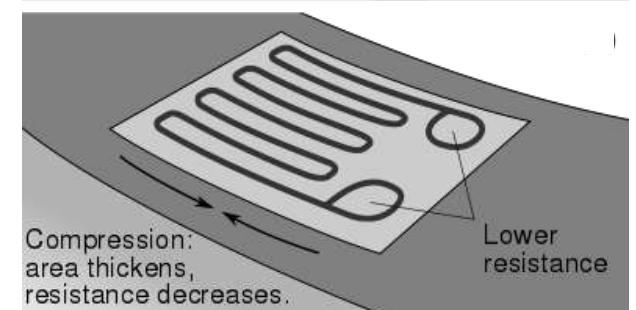
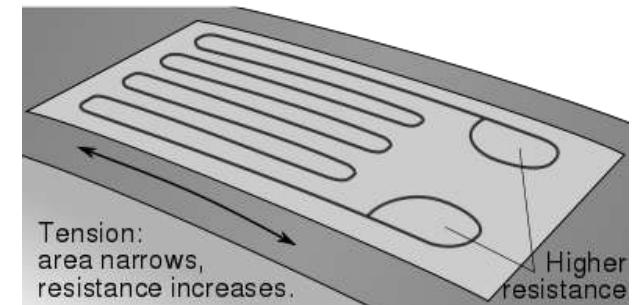
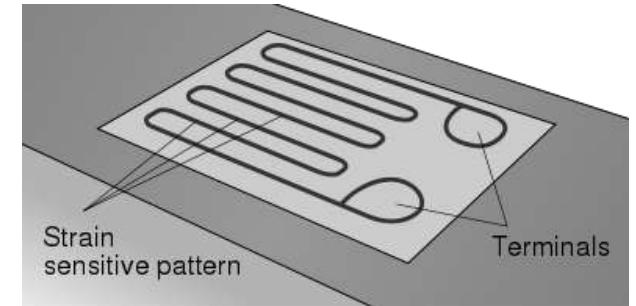


Force/torque and deformation

- indirect information obtained from the measure of **deformation** of an elastic element subject to the force or torque to be measured
- basic component is a **strain gauge**: it uses the variation of the resistance R of a metal conductor when its length L and/or cross-section S vary

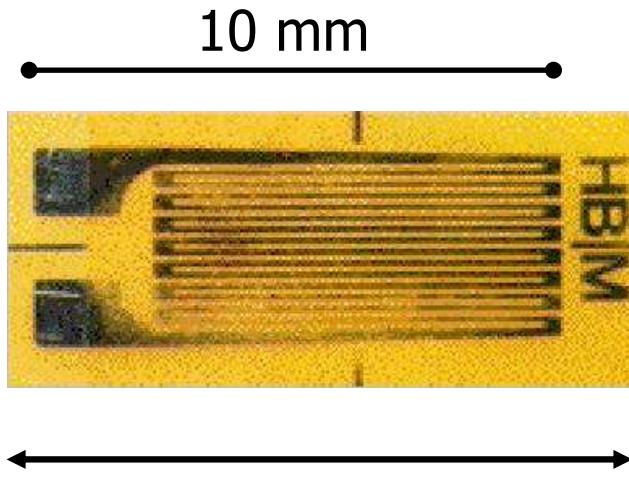
$$\frac{\partial R}{\partial L} > 0 \quad \frac{\partial R}{\partial S} < 0$$

$$\frac{\partial R}{\partial T} \xleftarrow{\text{small}}$$





Strain gauges



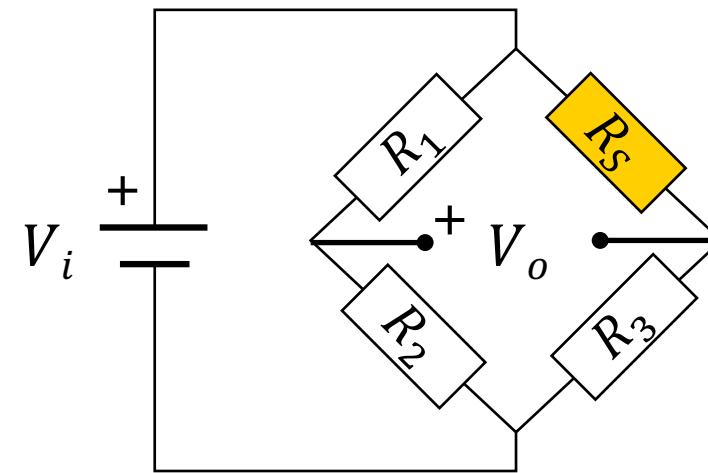
principal measurement axis

$$\text{Gauge-Factor} = \text{GF} = \frac{\Delta R / R}{\Delta L / L} \quad \text{strain } \varepsilon$$

(typically GF ≈ 2 , i.e., small sensitivity)

if R_1 has the same dependence on T of R_S
thermal variations are automatically
compensated

Wheatstone **single-point** bridge connection
(for accurately measuring resistance)



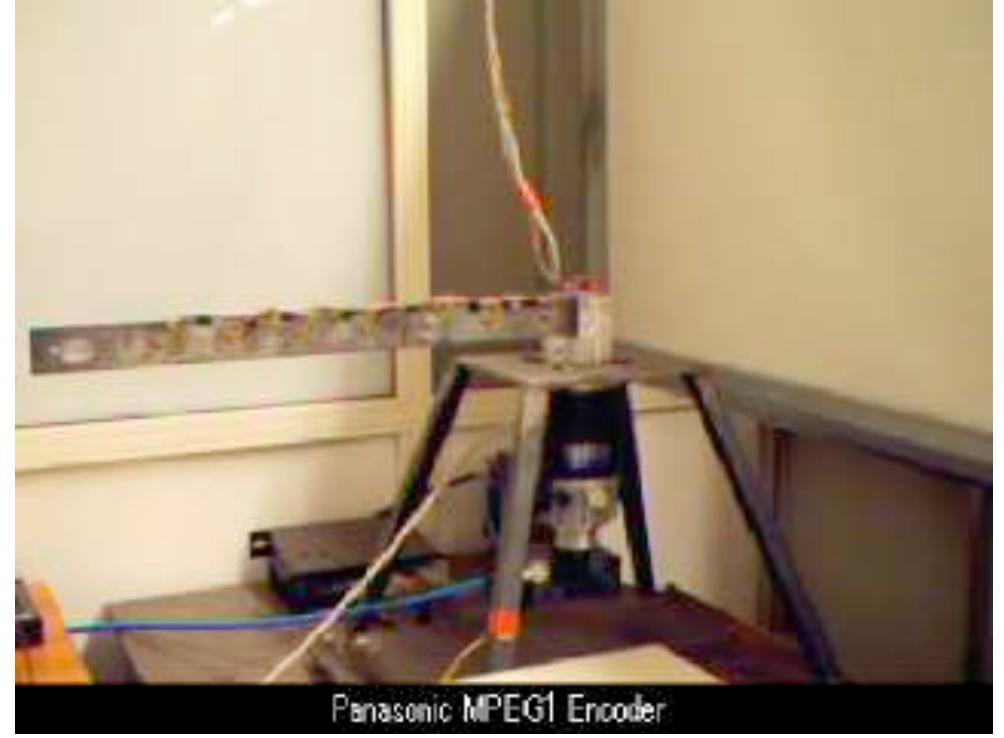
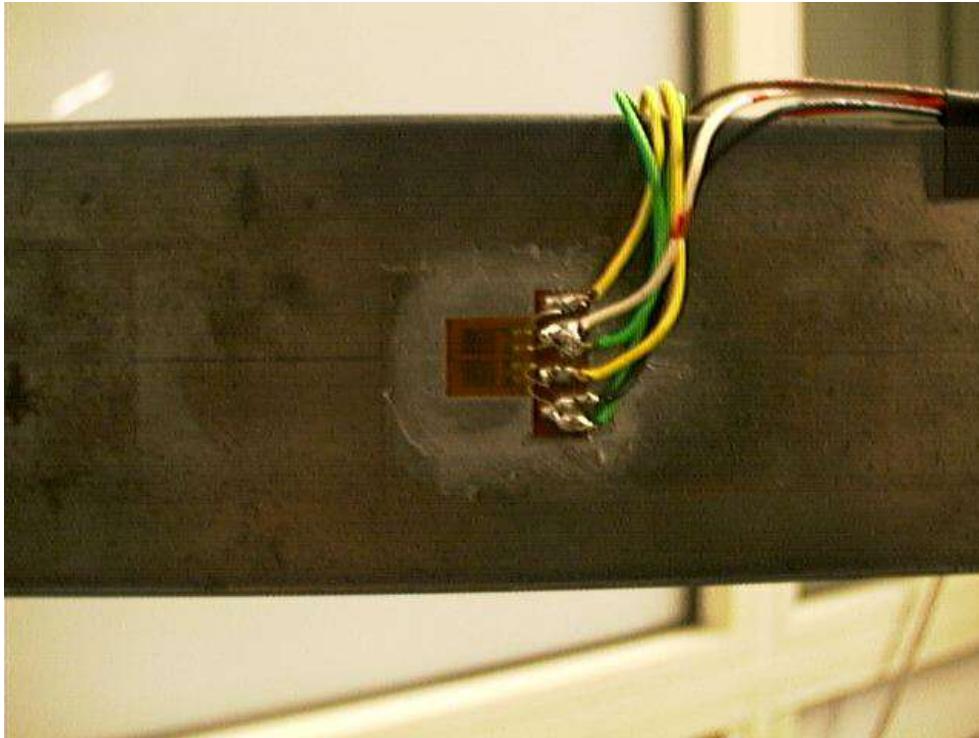
- R_1, R_2, R_3 very well matched ($\approx R$)
- $R_S \approx R$ at rest (no stress)
- **two-point** bridges have 2 strain gauges connected oppositely (\rightarrow sensitivity)

$$V_0 = \left(\frac{R_2}{R_1 + R_2} - \frac{R_3}{R_3 + R_S} \right) V_i$$



Strain gauges in flexible arms

[video](#)

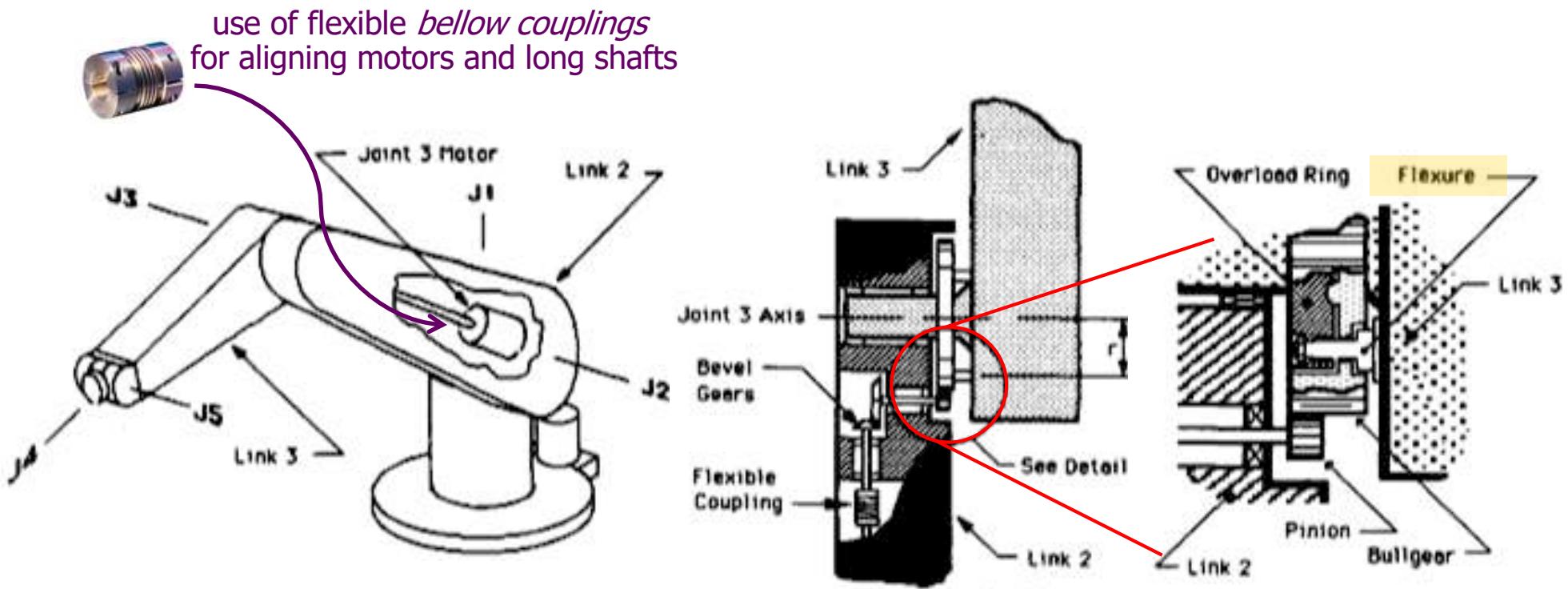


7 strain gauges glued⁽¹⁾ to a flexible aluminum beam (a robot “link”) measuring its local “curvature” in dynamic bending during slew motions (a **proprioceptive** use of these sensors)

⁽¹⁾ by cyanoacrylic glue



Torque sensor at robot joints



strain gauge mounted to “sense” the axial deformation
of the transmission shaft of joint #3 (elbow) in a PUMA 500 robot
(again, a **proprioceptive** use of this sensor)

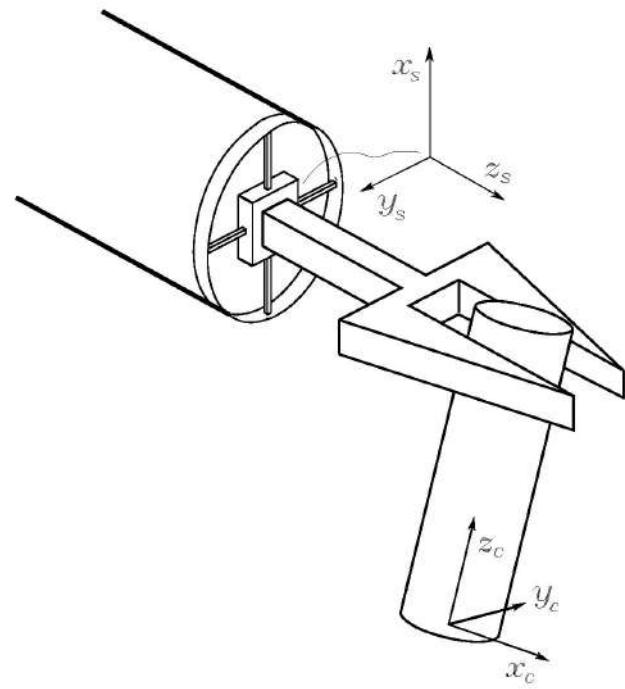


Force/torque sensor at robot wrist

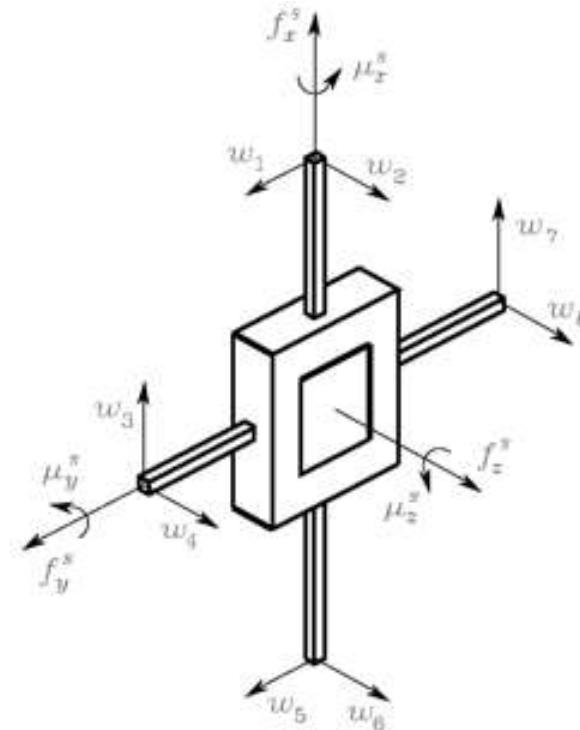
- a device (with the outer form of a cylinder), typically located between the last robot link and its end-effector
- top and bottom plates are mechanically connected by a number of **deformable elements** subject to **strain** under the action of forces and moments
- there should be at least one such element in any direction along/around which a force or torque measure is needed
- since a complete “decoupling” of these measurements is hard to obtain, there are $N \geq 6$ such deformable elements
- on each element, a **pair of strain gauges** is glued so as to undergo opposite deformations (e.g., traction/compression) along the main axis of measurement



Maltese-cross configuration



- diameter ≈ 10 cm
- height ≈ 5 cm
- $50 \div 500$ N (resolution 0.1%)
- $5 \div 70$ Nm (resolution 0.05%)
- sample frequency ≈ 1 KHz



- 4 deformable elements
- two pairs of strain gauges are mounted on opposite sides of each element (8 pairs)
- the two gauges of each pair are placed adjacent on the same Wheatstone bridge



6D force/torque sensors

- ATI series
- cost (in 2016): about 6 K€ for Mini45 model + 700 € DAQ card

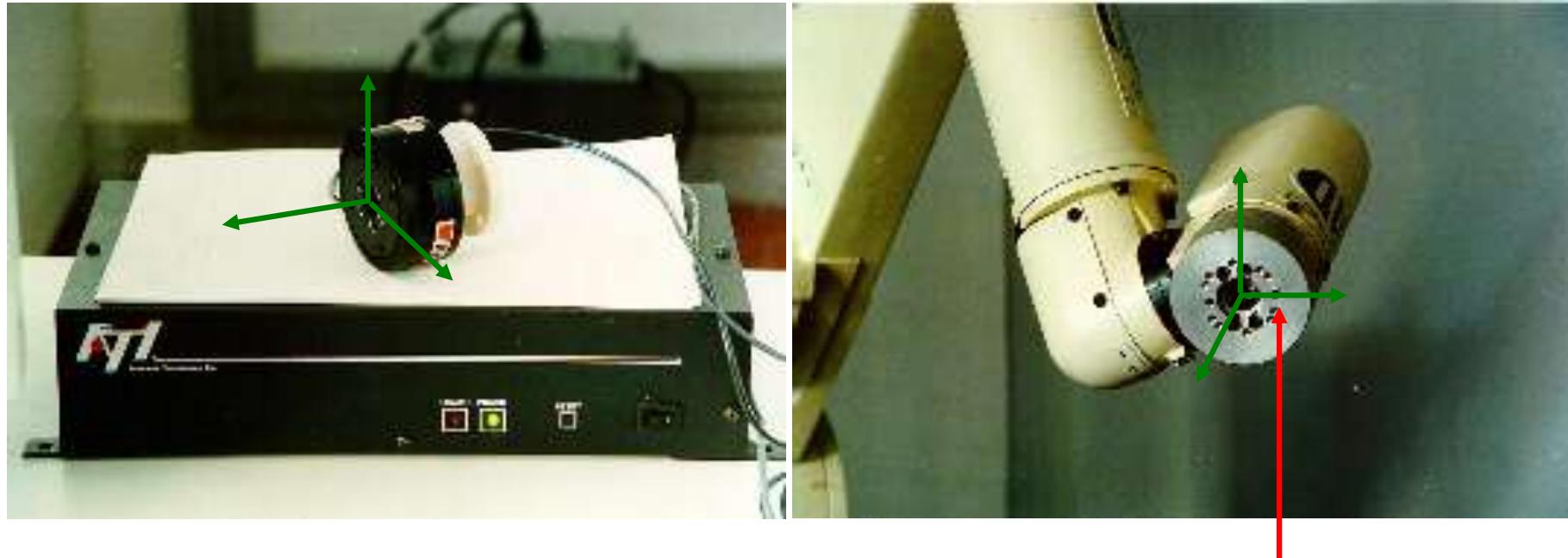


Model	Max Fx,Fy*	Max Tx,Ty*	Weight**	Diameter**	Height**
Nano17	±50 N	±500 N-mm	0.0091 kg	17 mm	14 mm
Nano25	±250 N	±6 N-m	0.064 kg	25 mm	22 mm
Nano43	±36 N	±500 N-mm	0.041 kg	43 mm	11 mm
Mini40	±80 N	±4 N-m	0.05 kg	40 mm	12 mm
Mini45	±580 N	±20 N-m	0.091 kg	45 mm	16 mm
Gamma	±130 N	±10 N-m	0.25 kg	75 mm	33 mm
Delta	±660 N	±60 N-m	0.91 kg	94 mm	33 mm
Theta	±2500 N	±400 N-m	5 kg	150 mm	61 mm
Omega160	±2500 N	±400 N-m	2.7 kg	160 mm	56 mm
Omega190	±7200 N	±1400 N-m	6.4 kg	190 mm	56 mm



6D force/torque sensor

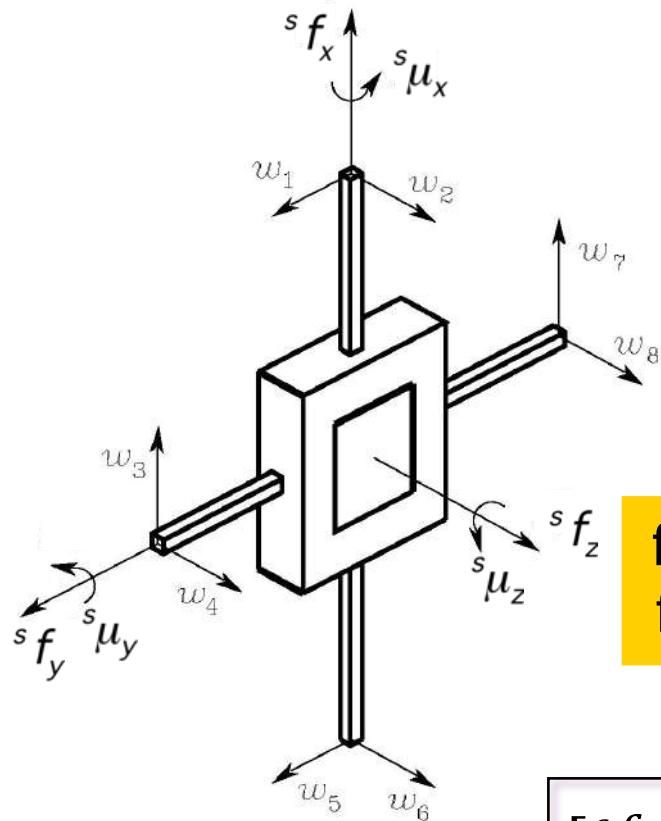
- electronic processing unit and mounting on an industrial robot (Comau Smart 3 robot, 6R kinematics)



mounting flange
(on link 6 of the manipulator arm)



6D F/T sensor calibration



$$\begin{bmatrix} {}^s f_x \\ {}^s f_y \\ {}^s f_z \\ {}^s \mu_x \\ {}^s \mu_y \\ {}^s \mu_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & c_{13} & 0 & 0 & 0 & c_{17} & 0 \\ c_{21} & 0 & 0 & 0 & c_{25} & 0 & 0 & 0 \\ 0 & c_{32} & 0 & c_{34} & 0 & c_{36} & 0 & c_{38} \\ 0 & 0 & 0 & c_{44} & 0 & 0 & 0 & c_{48} \\ 0 & c_{52} & 0 & 0 & 0 & c_{56} & 0 & 0 \\ c_{61} & 0 & c_{63} & 0 & c_{65} & 0 & c_{67} & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \end{bmatrix}$$

force/torque measured in the frame attached to the sensor

calibration matrix

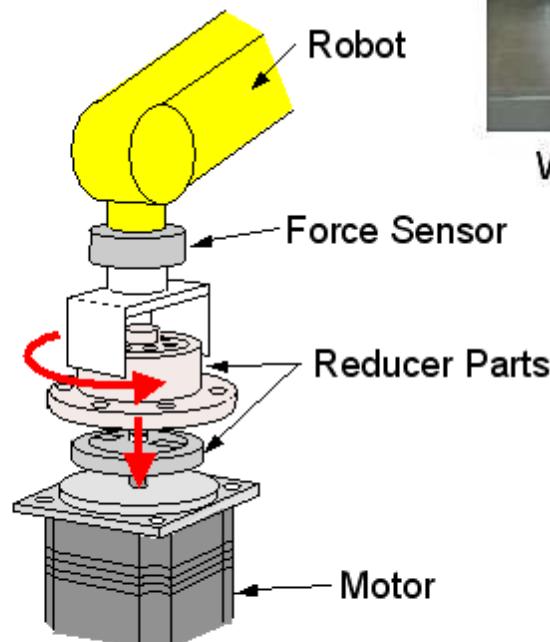
output of Wheatstone bridges

$$\begin{bmatrix} {}^c f_c \\ {}^c \mu_c \end{bmatrix} = \begin{bmatrix} {}^c R_s & O \\ S({}^c r_{cs}) {}^c R_s & {}^c R_s \end{bmatrix} \begin{bmatrix} {}^s f_s \\ {}^s \mu_s \end{bmatrix}$$

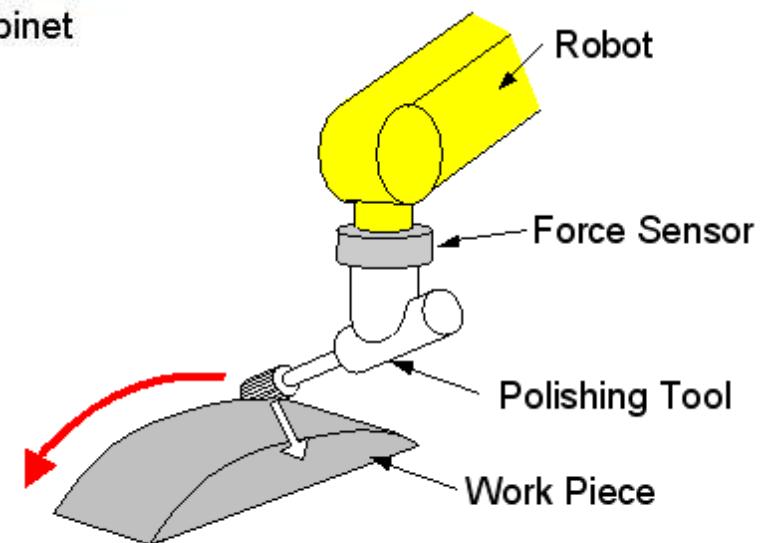
transformation from the sensor frame to the load/contact frame (at TCP)



Typical uses of a F/T sensor



Phase matching by force sensing



Following with constant pushing force



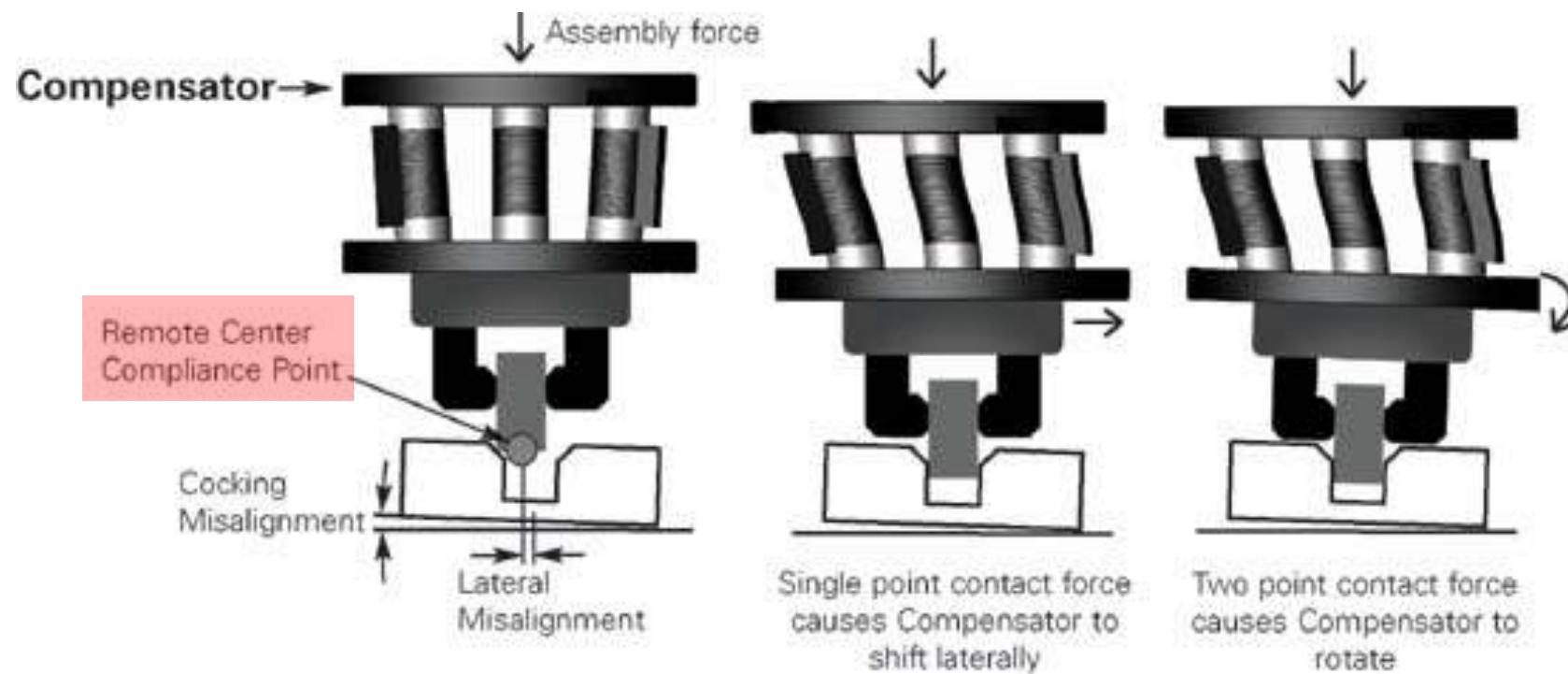
Passive RCC device

- RCC = Remote Center of Compliance
- placed on the wrist so as to introduce **passive “compliance”** to the robot end-effector, in response to static forces and moments applied from the environment at the contact area
- mechanical construction yields “**decoupled**” linear/angular motion responses **if** contact occurs at or near the RCC point





Assembly with RCC





Passive assembly with RCC



video

RCC by ATI Industrial Automation
<http://www.ati-ia.com>



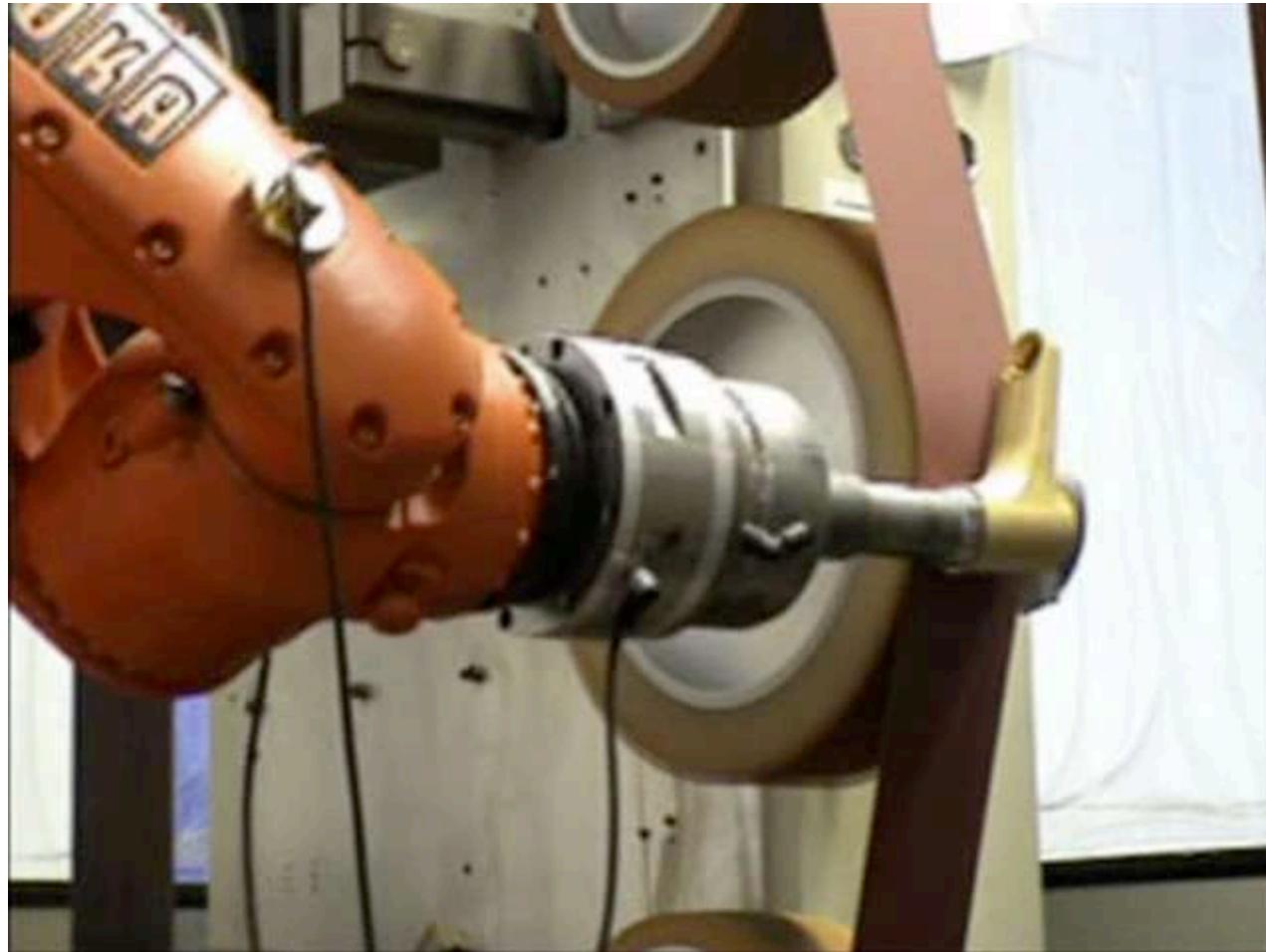
Active assembly with F/T sensor



ABB robot with ATI F/T sensor



Surface finishing with F/T sensor



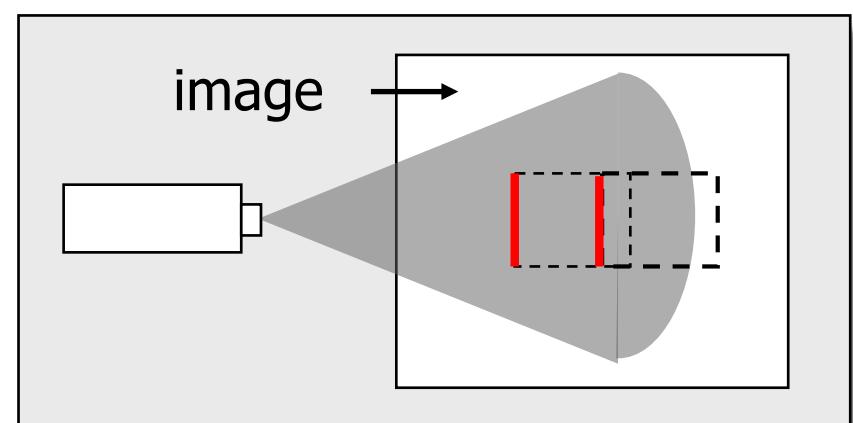
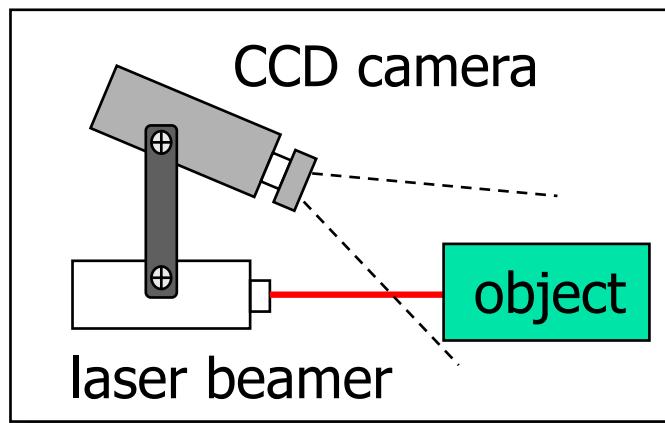
video

KUKA robot with F/T sensor

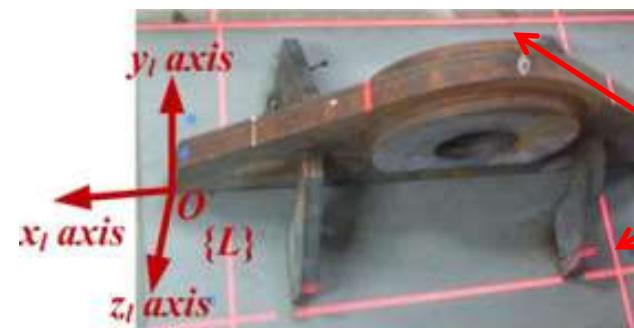


Proximity/distance sensors - 4

- **structured light:** a laser beam (coherent light source) is projected on the environment, and its planar intersection with surrounding objects is detected by a (tilted) camera
- the position of the “red pixels” on the camera image plane is in **trigonometric** relation with the object distance from the sensor



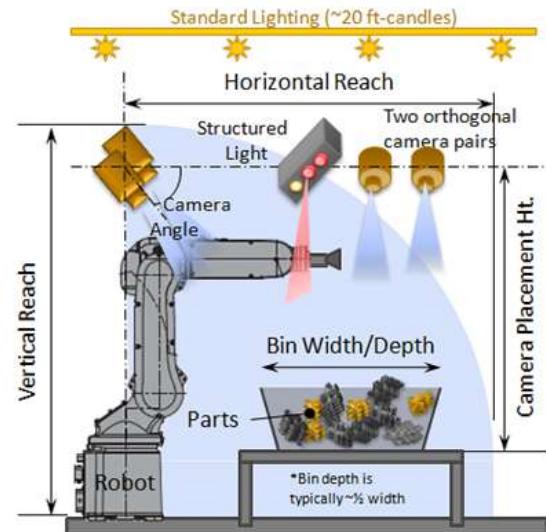
side view



top view



Use of structured light sensors



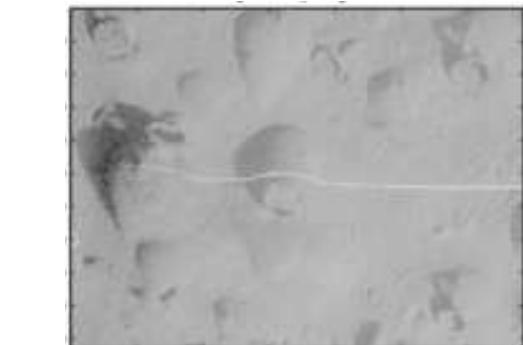
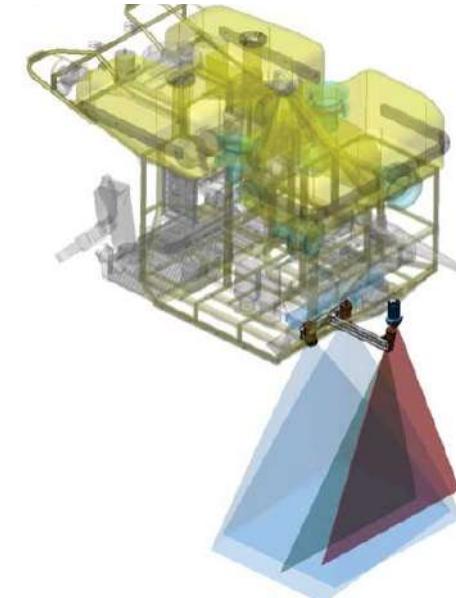
Random [bin picking](#) of 10-30 parts/minute (with surface inspection) with a 6R industrial robot, two pairs of cameras and a structured light sensor [Universal Robotics]



Structured light approach to best fit and [finish car bodies](#) (down to 0.1 mm) for reducing wind noise [Ford Motor Co.]



[Virtobot](#) system for post-mortem 3D optical scanning of human body & image-guided needle placement [Univ. Zürich]



[Hercules ROV](#) + structured-laser-light imaging system for high-resolution bathymetric underwater maps [Univ. Rhode Island]



Robotic bin picking using vision and structured light

video

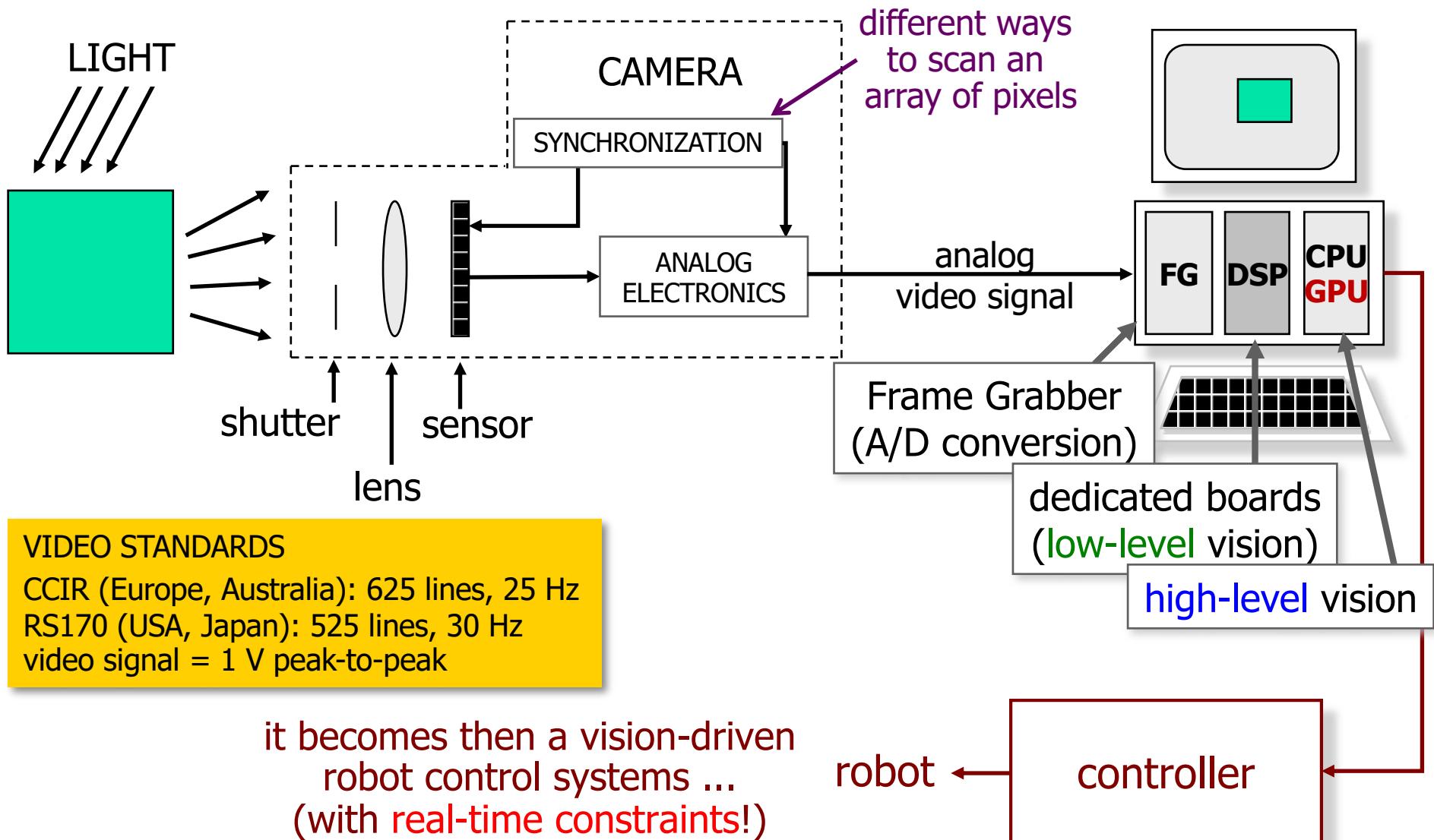


video





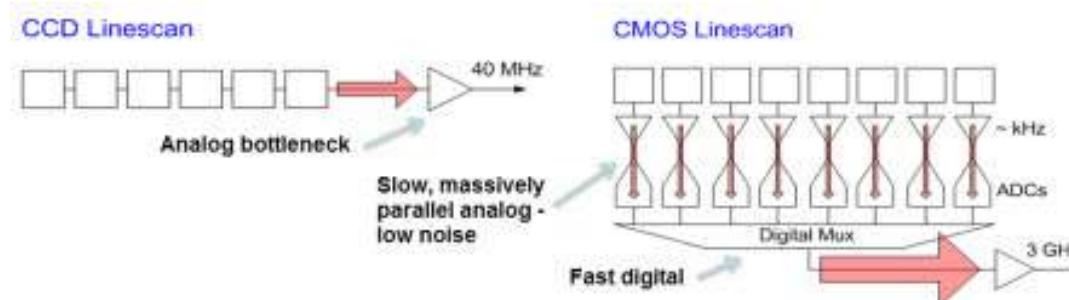
Vision systems





Sensors for vision

- arrays (spatial sampling) of photosensitive elements (**pixel**) converting light energy into electrical energy
- **CCD** (Charge Coupled Device): each pixel surface is made by a semiconductor device, **accumulating** free charge when hit by photons (**photoelectric effect**); “integrated” charges “read-out” by a sequential process (external circuitry) and transformed into voltage levels
- **CMOS** (Complementary Metal Oxide Semiconductor): each pixel is a **photodiode**, directly providing a voltage or current proportional to the **instantaneous** light intensity, with possibility of random access to each pixel





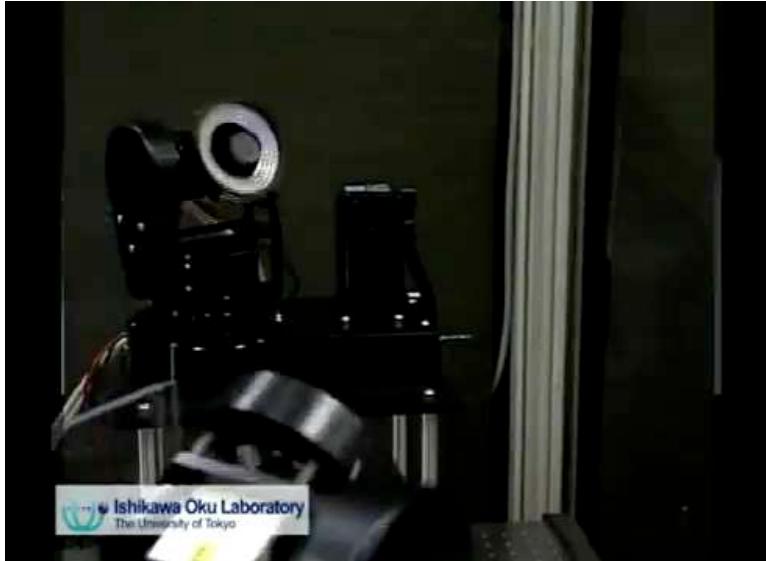
CMOS versus CCD

- reduction of fabrication costs of CMOS imagers
- better spatial resolution of elementary sensors
 - CMOS: 1M pixel, CCD: 768×576 pixel
- faster processing speed
 - 1000 vs. 25 fps (frames per second)
- possibility of integrating “intelligent” functions on single chip
 - sensor + frame grabber + low-level vision
- random access to each pixel or area
 - flexible handling of ROI (Region Of Interest)
- possibly lower image quality w.r.t. CCD imagers
 - sensitivity, especially for applications with low S/N signals
- customization for small volumes is more expensive
 - CCD cameras have been since much longer time on the market



Fast image processing for fast motion control

video



video



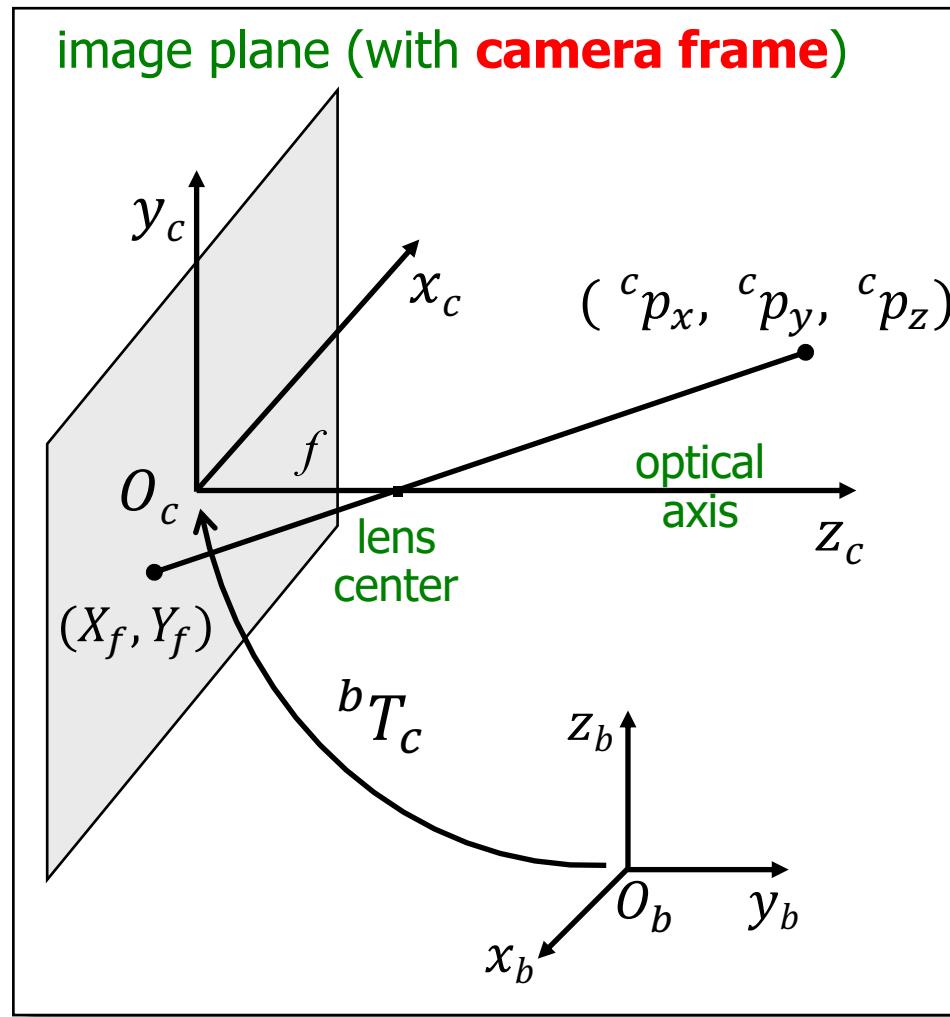
video

- 1 KHz vision frame rate
- 1 KHz robot control rate
@ Ishikawa Lab – U Tokyo
(2007-09)





Perspective transformation with pinhole camera model



1. in metric units

$$X_f = \frac{f \ ^c p_x}{f - ^c p_z} \quad Y_f = \frac{f \ ^c p_y}{f - ^c p_z}$$

2. in pixel

$$X_I = \frac{\alpha_x f \ ^c p_x}{f - ^c p_z} + X_0$$

offsets of pixel coordinate system w.r.t. optical axis

$$Y_I = \frac{\alpha_y f \ ^c p_y}{f - ^c p_z} + Y_0$$

pixel/metric scaling factor

3. LINEAR MAP in homogeneous coordinates

$$X_I = \frac{x_I}{z_I} \quad Y_I = \frac{y_I}{z_I} \quad \rightarrow \quad \begin{bmatrix} x_I \\ y_I \\ z_I \end{bmatrix} = \Omega \begin{bmatrix} ^c p_x \\ ^c p_y \\ ^c p_z \\ 1 \end{bmatrix}$$

$$\Omega = \begin{bmatrix} \alpha_x & 0 & X_0 & 0 \\ 0 & \alpha_y & Y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/f & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

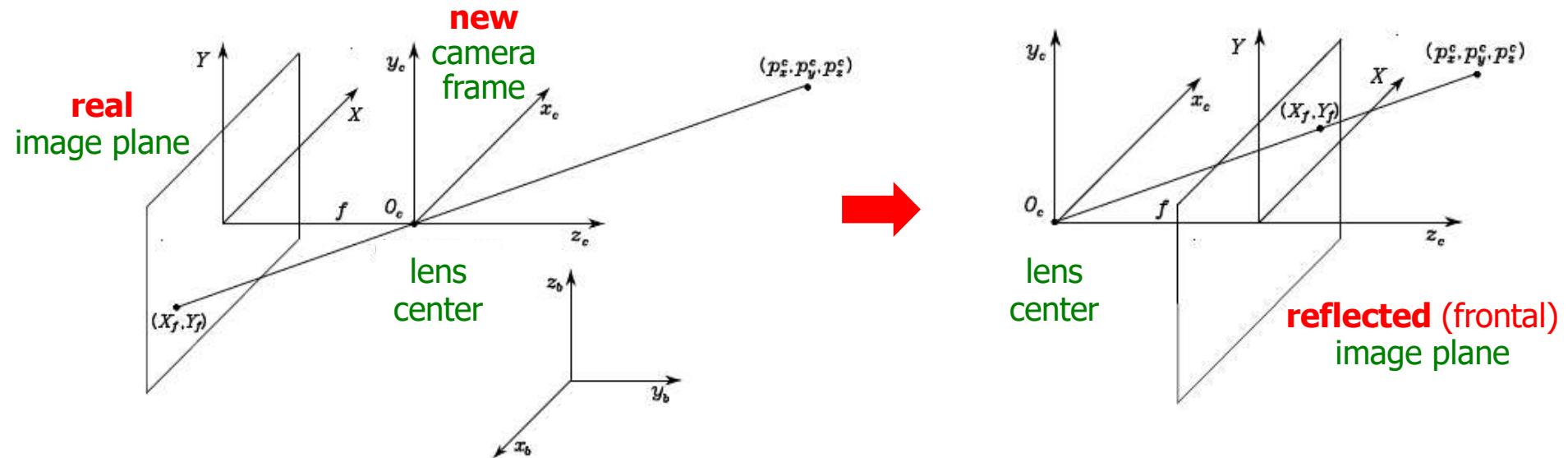
calibration matrix

$$H = \Omega \cdot {}^c T_b$$

intrinsic and extrinsic parameters



Perspective transformation with camera frame at the lens center



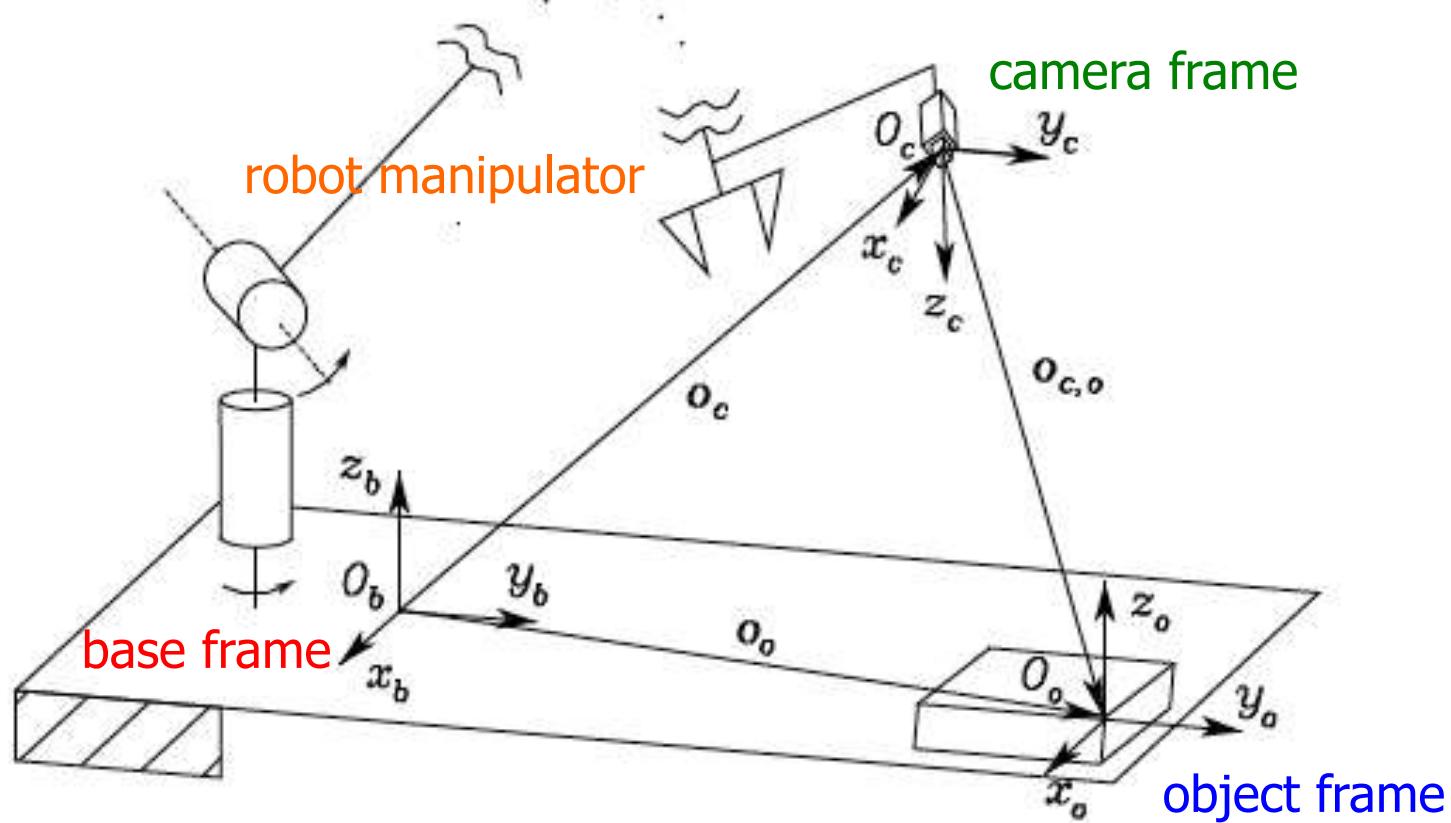
1. in metric units $X_f = -\frac{f}{c} \frac{p_x}{p_z}$ $Y_f = -\frac{f}{c} \frac{p_y}{p_z}$ \rightarrow $X_f = \frac{f}{c} \frac{p_x}{p_z}$ $Y_f = \frac{f}{c} \frac{p_y}{p_z}$

2. in pixel \dots \rightarrow $X_I = \frac{\alpha_x f}{c} \frac{p_x}{p_z} + X_0$ $Y_I = \frac{\alpha_y f}{c} \frac{p_y}{p_z} + Y_0$

3. LINEAR MAP in homogeneous coordinates \dots \rightarrow $\begin{bmatrix} x_I \\ y_I \\ z_I \\ 1 \end{bmatrix} = \Omega \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$ $\Omega = \begin{bmatrix} \alpha_x f & 0 & X_0 & 0 \\ 0 & \alpha_y f & Y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$



Eye-in-hand camera



Relevant reference frames for visual-based tasks

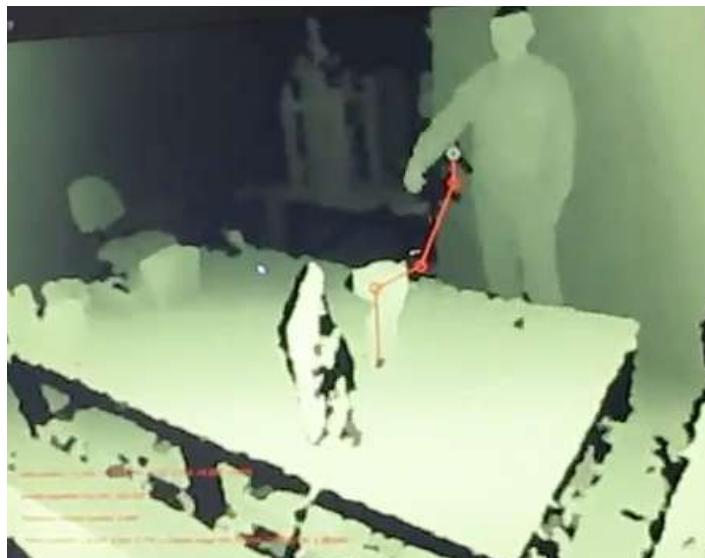


Kinect

camera + structured light 3D sensor



- RGB camera (with 640×480 pixel)
- depth sensor (by PrimeSense)
 - infrared laser emitter
 - infrared camera (with 320×240 pixel)
- 30 fps data rate
- range: $0.5 \div 5$ m
- depth resolution: 1cm@2m; 7cm@5m
- cost: < 90 €

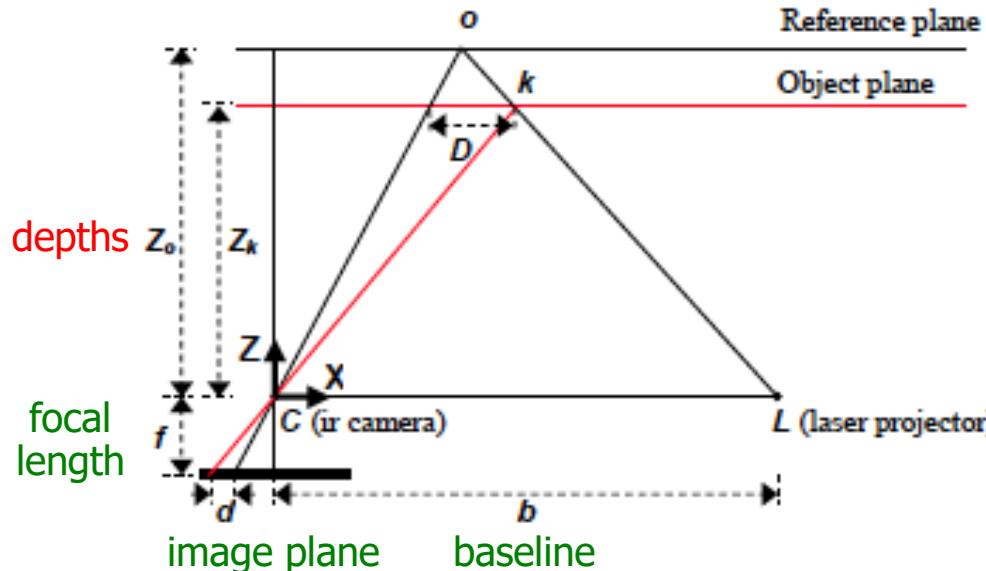


“skeleton” extraction and
human motion tracking



Kinect

Depth sensor operation



- stereo triangulation based on IR source emitting pseudo-random patterns
- reference pattern on IR camera image plane acquired in advance from a plane at known distance and coded in H/W
- correlating the disparity d (10 bits) of reference and received object patterns provides the object depth z_k

1. triangulation equations (by similarity of triangles)

$$\frac{D}{b} = \frac{z_0 - z_k}{z_0} \quad \& \quad \frac{d}{f} = \frac{D}{z_k}$$

$$z_k = \frac{z_0}{1 + \frac{d}{fb} z_0} \quad \begin{matrix} \rightarrow \\ \rightarrow \end{matrix} \quad \begin{aligned} x_k &= -\frac{z_k}{f} (X_k - X_0 + \delta X) \\ y_k &= -\frac{z_k}{f} (Y_k - Y_0 + \delta Y) \end{aligned}$$

2. accurate calibration of sensor

baseline length b , depth of reference z_0 + camera intrinsic parameters
(focal length f , lens distortion coefficients $\delta X, \delta Y$, center offsets X_0, Y_0)



How Kinect works (a 2-minute illustration...)

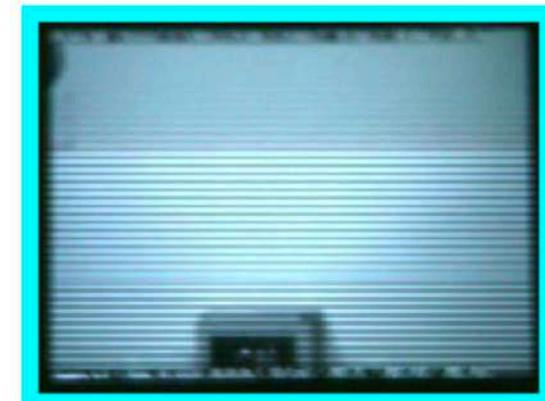
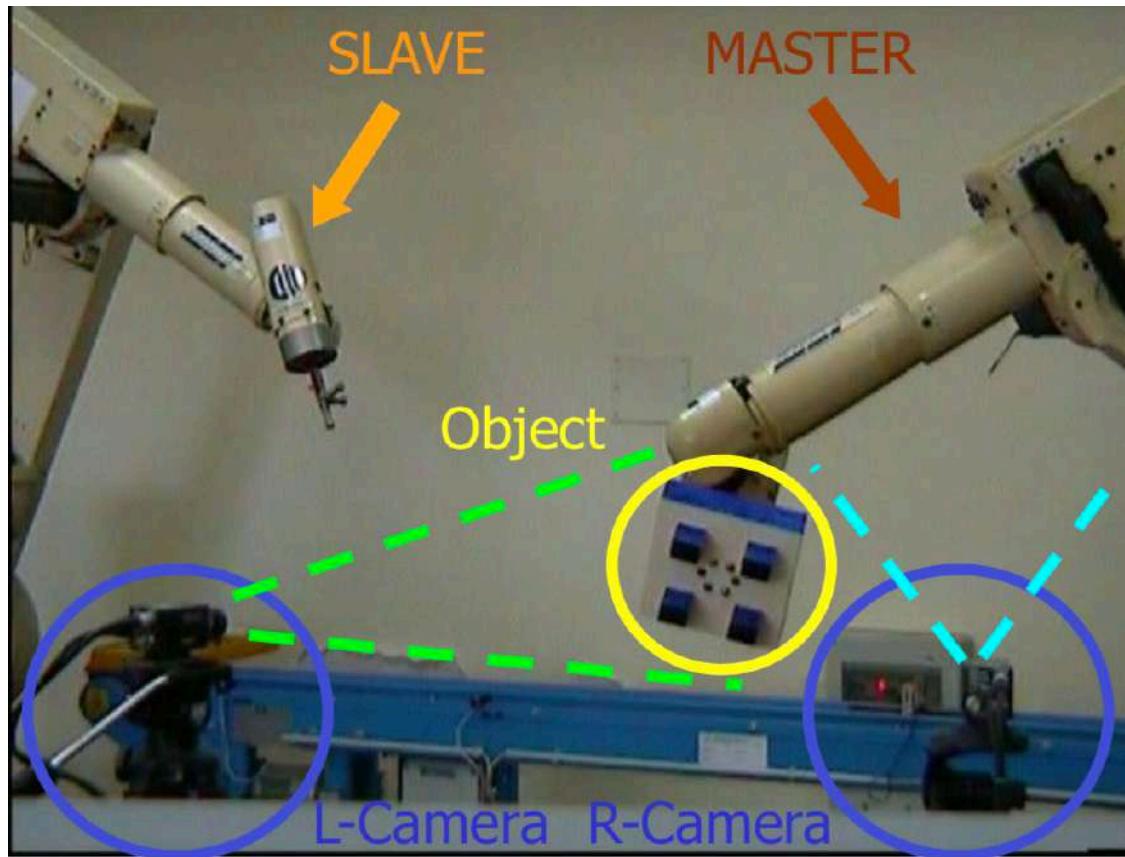


<http://youtu.be/uq9SEJxZiUg>



Manipulators and vision systems

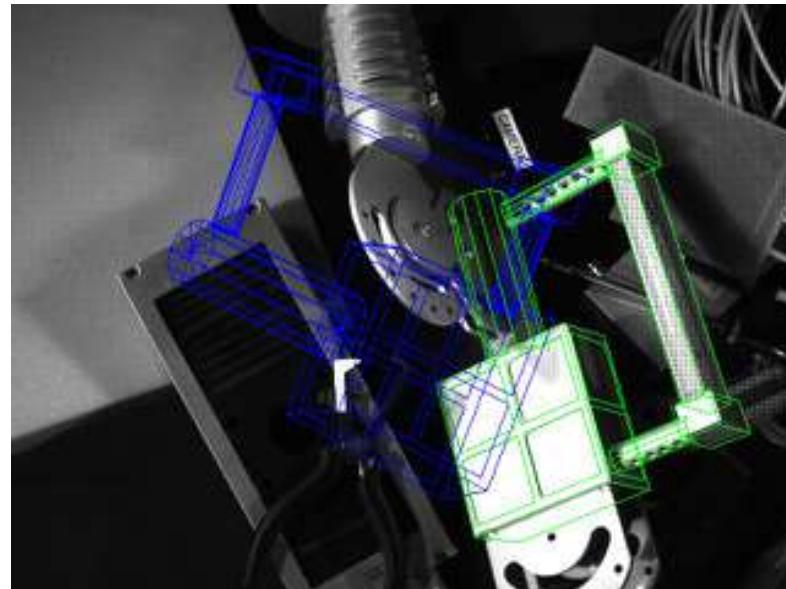
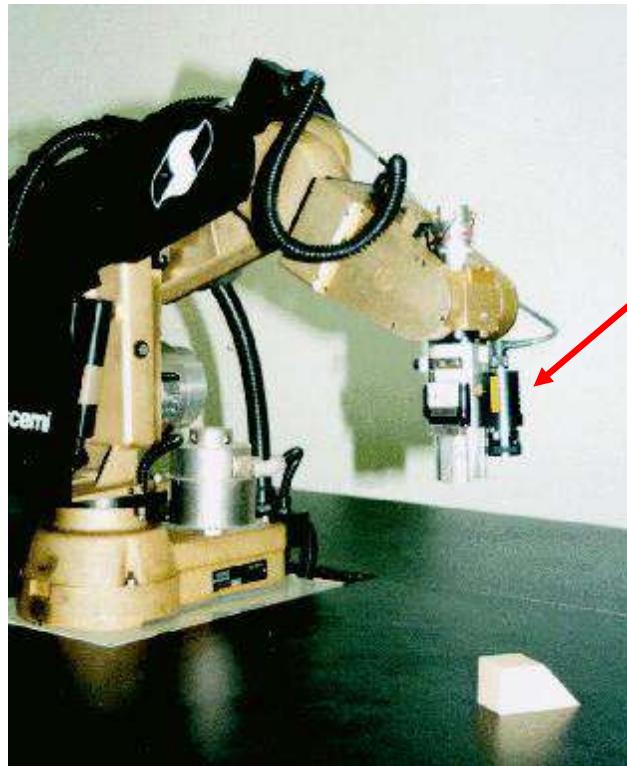
- stereovision with two external cameras, fixed in the environment (**eye-to-hand**)





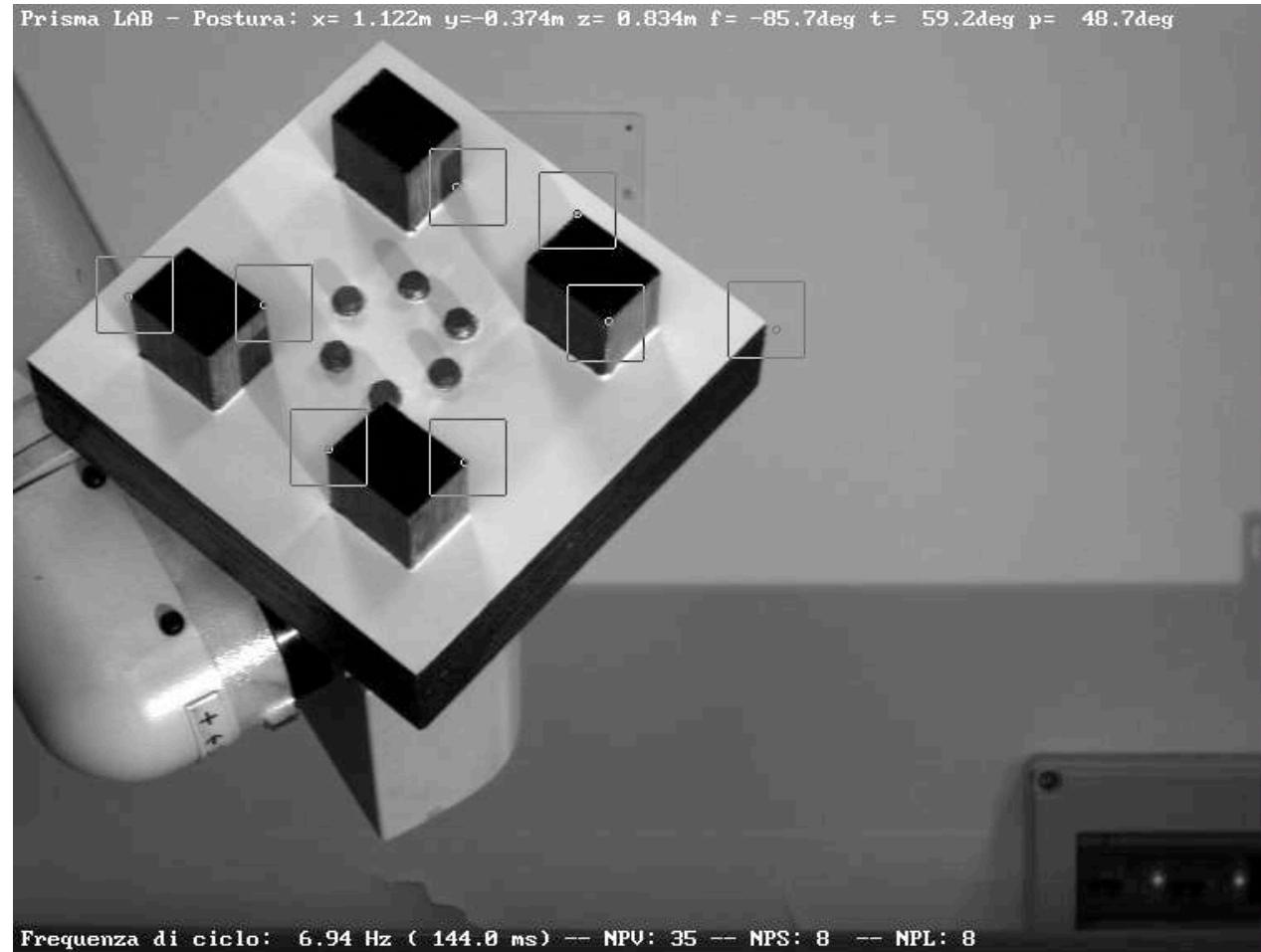
Manipulators and vision systems

- CCD camera mounted on the robot for controlling the end-effector positioning (**eye-in-hand**)





Visual tracking eye-to-hand



COMAU robot with position-based 6D tracking from external camera
(DIS, Università di Napoli Federico II)



Visual servoing eye-in-hand

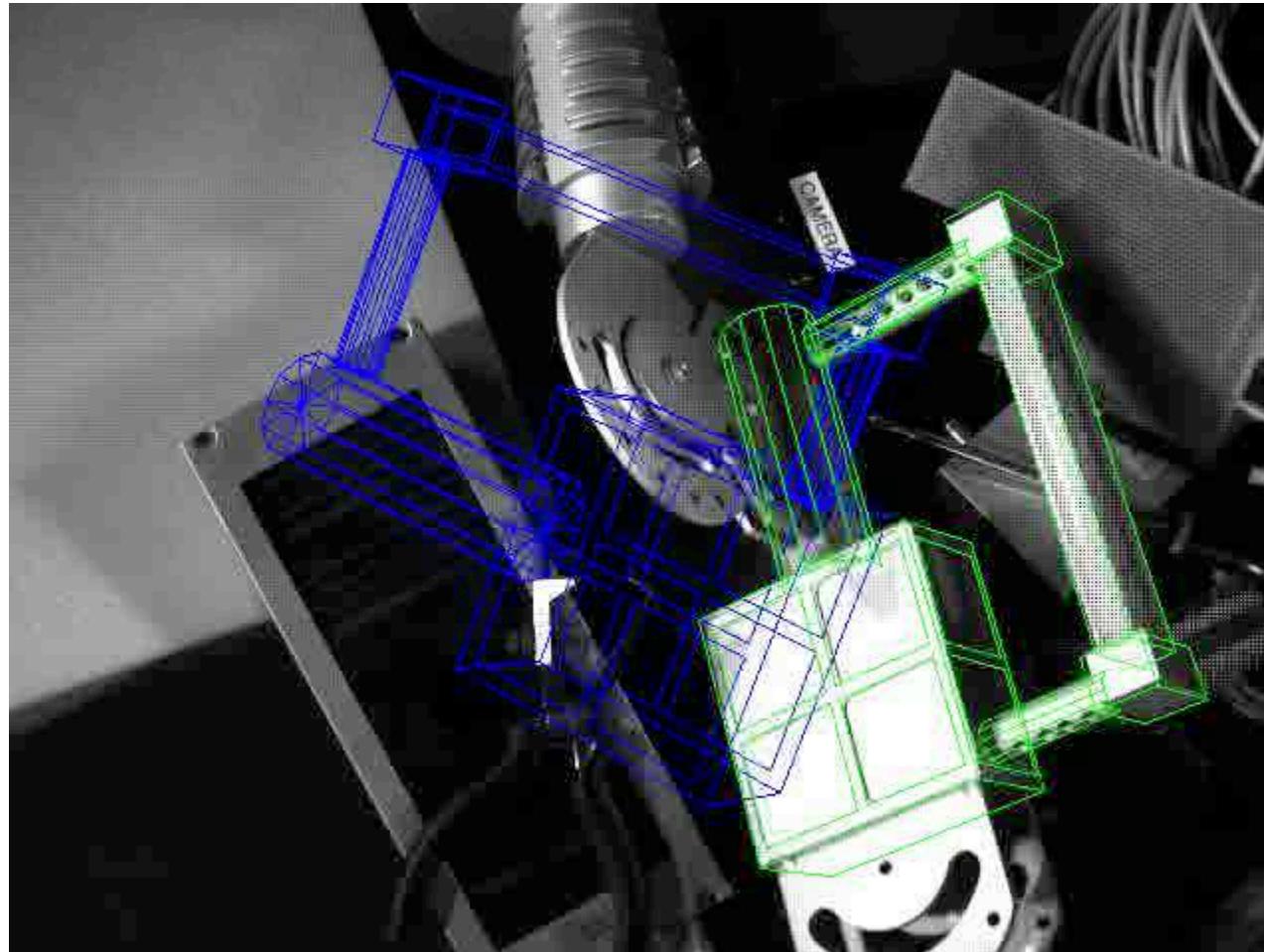
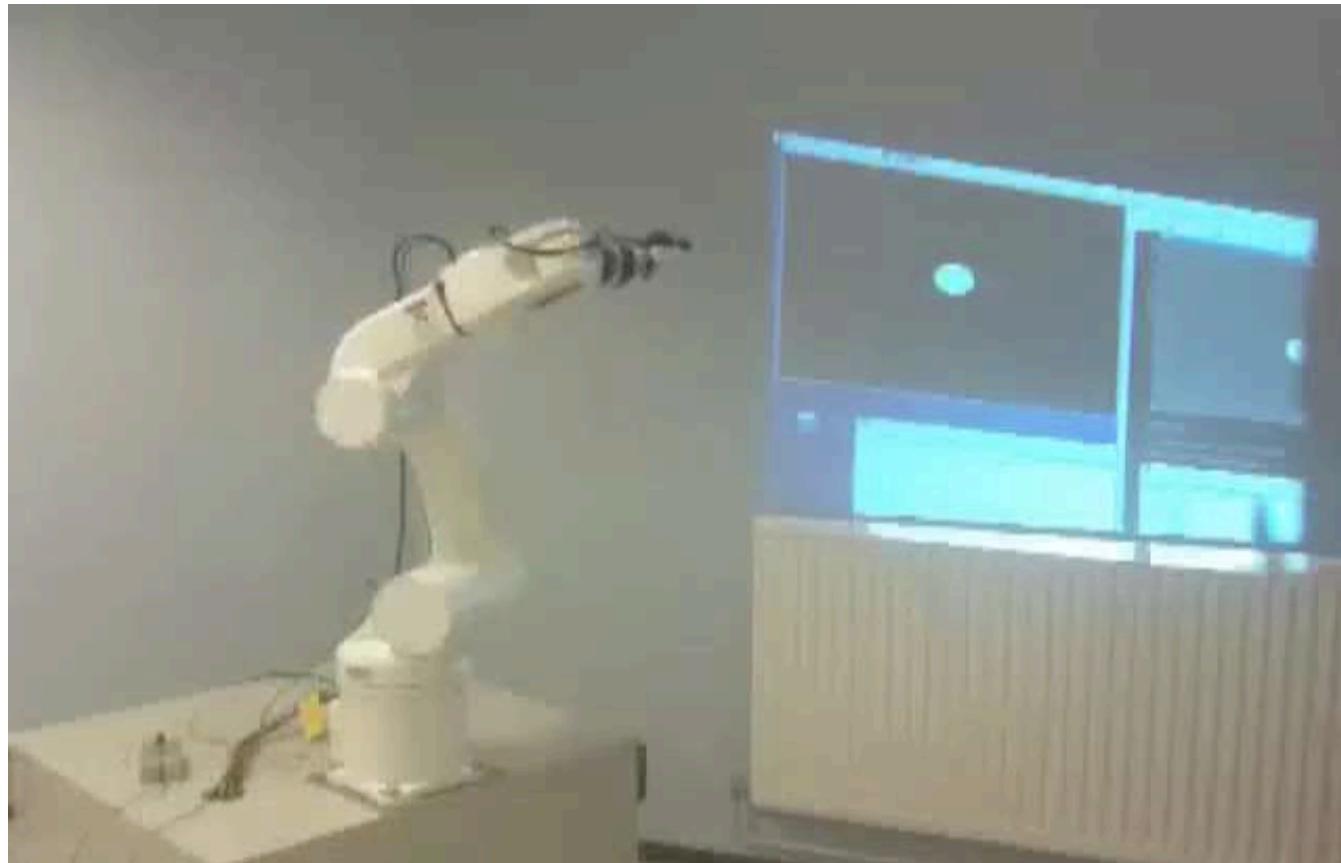


Image-based servoing with camera mounted on the robot end-effector
(IRISA/INRIA, Rennes)



Visual servoing and redundancy



[video](#)

visual servoing of circle feature ($m = 3: p_x, p_y, r$) by Adept Viper robot ($n = 6$):
redundancy is used for avoiding joint range limits (IRISA/INRIA, Rennes)



Combined visual/force assembly



[video](#)

KUKA LWR with eye-in-hand camera and F/T sensor
(DLR, IEEE ICRA'07 demo in Roma)

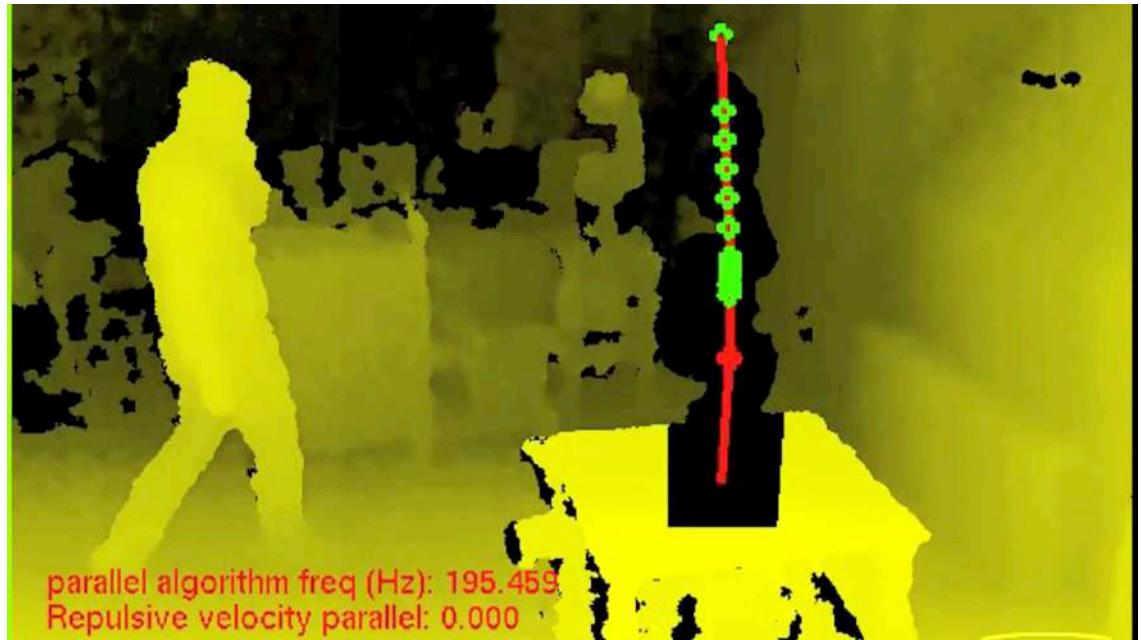
On-line distance computation and human-robot coexistence



video



monitoring **left-** and **right-hand**
distance to the robot (at same time)



several **control points** on robot **skeleton**
used to compute distances and control motion

KUKA LWR with a Kinect monitoring its workspace
(DIAG Robotics Laboratory, EU project SAPHARI, 2013)



Robotics 1

Programming Supervision and control architectures

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





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)



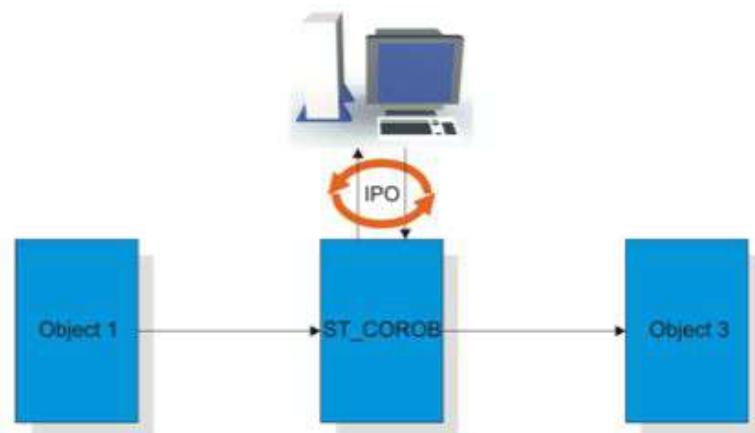
Robot-oriented programming

- “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



Fig. 4-1: Front view of KCP

1	Mode selector switch	10	Numeric keypad
2	Drives ON	11	Softkeys
3	Drives OFF / SSB GUI	12	Start backwards key
4	EMERGENCY STOP button	13	Start key
5	Space Mouse	14	STOP key
6	Right-hand status keys	15	Window selection key
7	Enter key	16	ESC key
8	Arrow keys	17	Left-hand status keys
9	Keypad	18	Menu keys



KRL language

- basic **instruction** set:

Variables and declarations	
DECL	(>>> 10.4.1 "DECL" page 138)
ENUM	(>>> 10.4.2 "ENUM" page 140)
IMPORT ... IS	(>>> 10.4.3 "IMPORT ... IS" page 141)
STRUCT	(>>> 10.4.4 "STRUCT" page 141)

Motion programming	
CIRC	(>>> 10.5.1 "CIRC" page 143)
CIRC_REL	(>>> 10.5.2 "CIRC_REL" page 144)
LIN	(>>> 10.5.3 "LIN" page 146)
LIN_REL	(>>> 10.5.4 "LIN_REL" page 146)
PTP	(>>> 10.5.5 "PTP" page 148)
PTP_REL	(>>> 10.5.6 "PTP_REL" page 148)

Program execution control	
CONTINUE	(>>> 10.6.1 "CONTINUE" page 150)
EXIT	(>>> 10.6.2 "EXIT" page 150)
FOR ... TO ... ENDFOR	(>>> 10.6.3 "FOR ... TO ... ENDFOR" page 150)
GOTO	(>>> 10.6.4 "GOTO" page 151)
HALT	(>>> 10.6.5 "HALT" page 152)
IF ... THEN ... ENDIF	(>>> 10.6.6 "IF ... THEN ... ENDIF" page 152)
LOOP ... ENDOOP	(>>> 10.6.7 "LOOP ... ENDOOP" page 153)
REPEAT ... UNTIL	(>>> 10.6.8 "REPEAT ... UNTIL" page 153)
SWITCH ... CASE ... ENDSWITCH	(>>> 10.6.9 "SWITCH ... CASE ... ENDSWITCH" page 154)
WAIT ... FOR	(>>> 10.6.10 "WAIT FOR" page 155)
WAIT ... SEC	(>>> 10.6.11 "WAIT SEC" page 156)
WHILE ... ENDO WHILE	(>>> 10.6.12 "WHILE ... ENDO WHILE" page 156)

Inputs/outputs	
ANIN	(>>> 10.7.1 "ANIN" page 157)
ANOUT	(>>> 10.7.2 "ANOUT" page 158)
DIGIN	(>>> 10.7.3 "DIGIN" page 159)
PULSE	(>>> 10.7.4 "PULSE" page 160)
SIGNAL	(>>> 10.7.5 "SIGNAL" page 164)

Subprograms and functions	
RETURN	(>>> 10.8.1 "RETURN" page 165)

Interrupt programming	
BRAKE	(>>> 10.9.1 "BRAKE" page 166)
INTERRUPT	(>>> 10.9.2 "INTERRUPT" page 166)
INTERRUPT ... DECL ... WHEN ... DO	(>>> 10.9.3 "INTERRUPT ... DECL ... WHEN ... DO" page 167)
RESUME	(>>> 10.9.4 "RESUME" page 169)

Path-related switching actions (=Trigger)	
TRIGGER WHEN DISTANCE	(>>> 10.10.1 "TRIGGER WHEN DISTANCE" page 170)
TRIGGER WHEN PATH	(>>> 10.10.2 "TRIGGER WHEN PATH" page 173)

Communication	
(>>> 10.11 "Communication" page 176)	

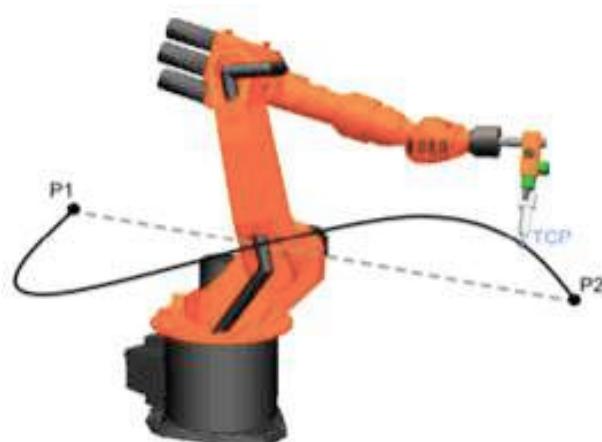
System functions	
VARSTATE()	(>>> 10.12.1 "VARSTATE()" page 176)

- basic **data** set: frames, vectors + DECLaration



KRL language

- typical motion primitives



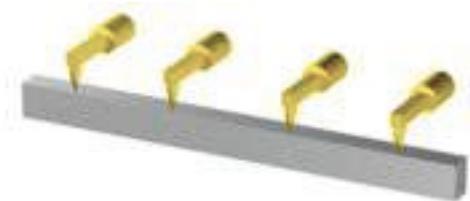
PTP motion
(point-to-point, linear
in joint space)



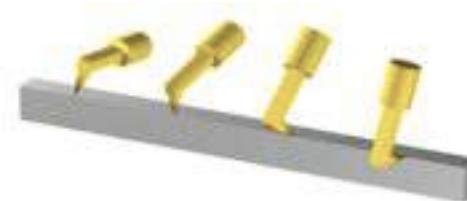
LIN motion
(linear in
Cartesian space)



CIRC motion
(circular in
Cartesian space)



CONST orientation

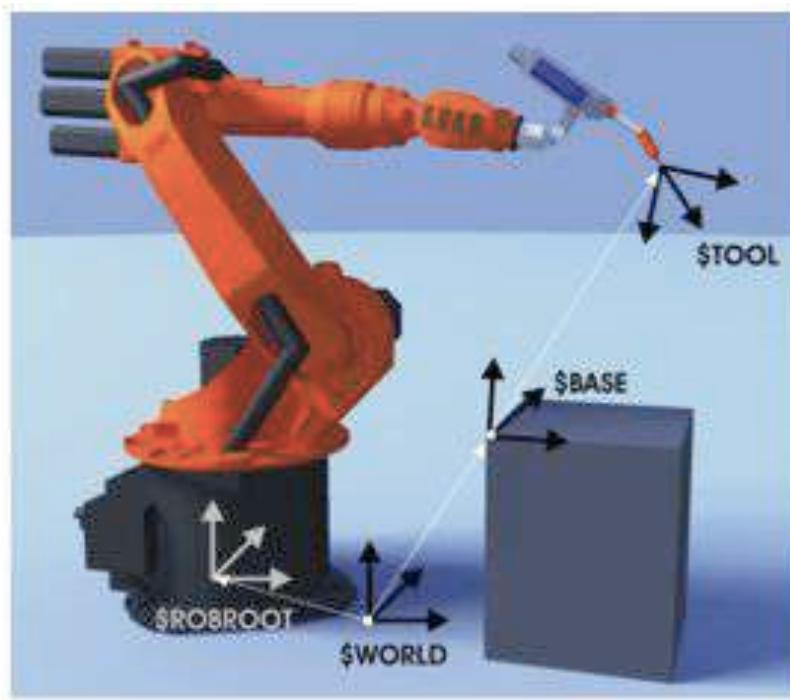


PTP motion
(linear in RPY angles)



KRL language

- multiple coordinate frames (in Cartesian space) and jogging of robot joints

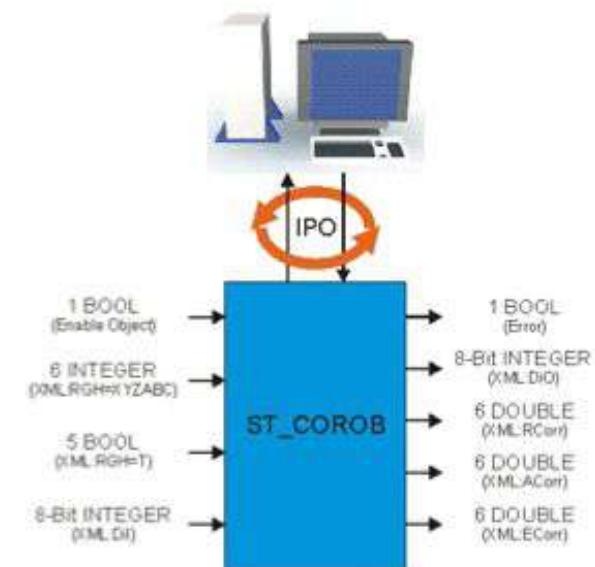


KUKA Ethernet RSI

Robot Sensor Interface



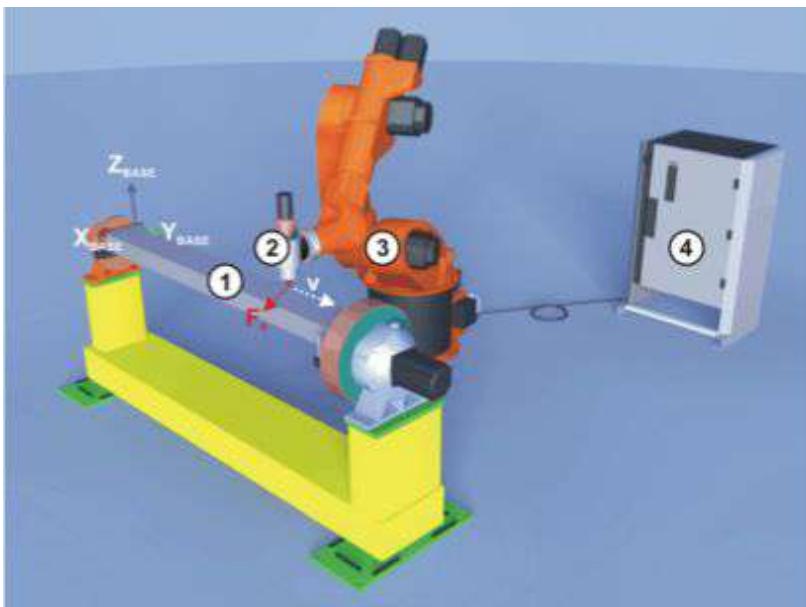
- cyclical data transmission **from** the robot controller **to** an external system (e.g., position data, axis angles, operating mode, etc.) and **vice versa** (e.g., sensor data) in the **interpolation cycle of 12 ms**
- **influencing** the robot in the interpolation cycle by means of an external program
- 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

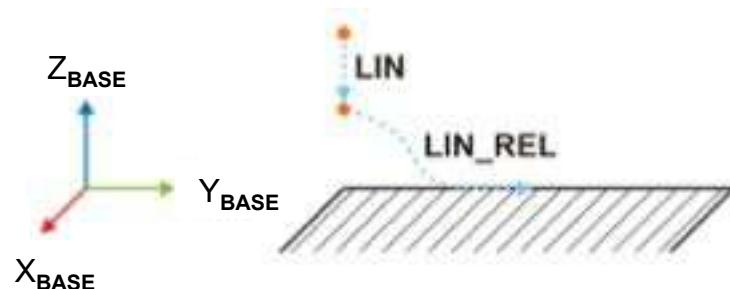
② tool with force sensor

③ robot

④ robot controller

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

v direction of motion



LIN_REL = 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

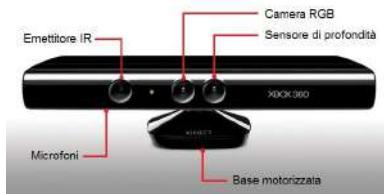


video



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

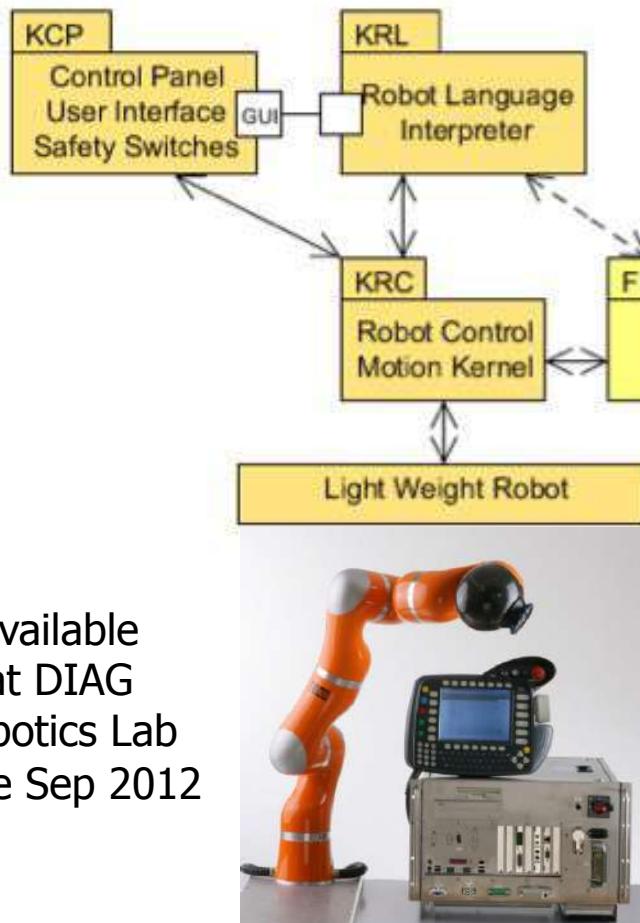


video

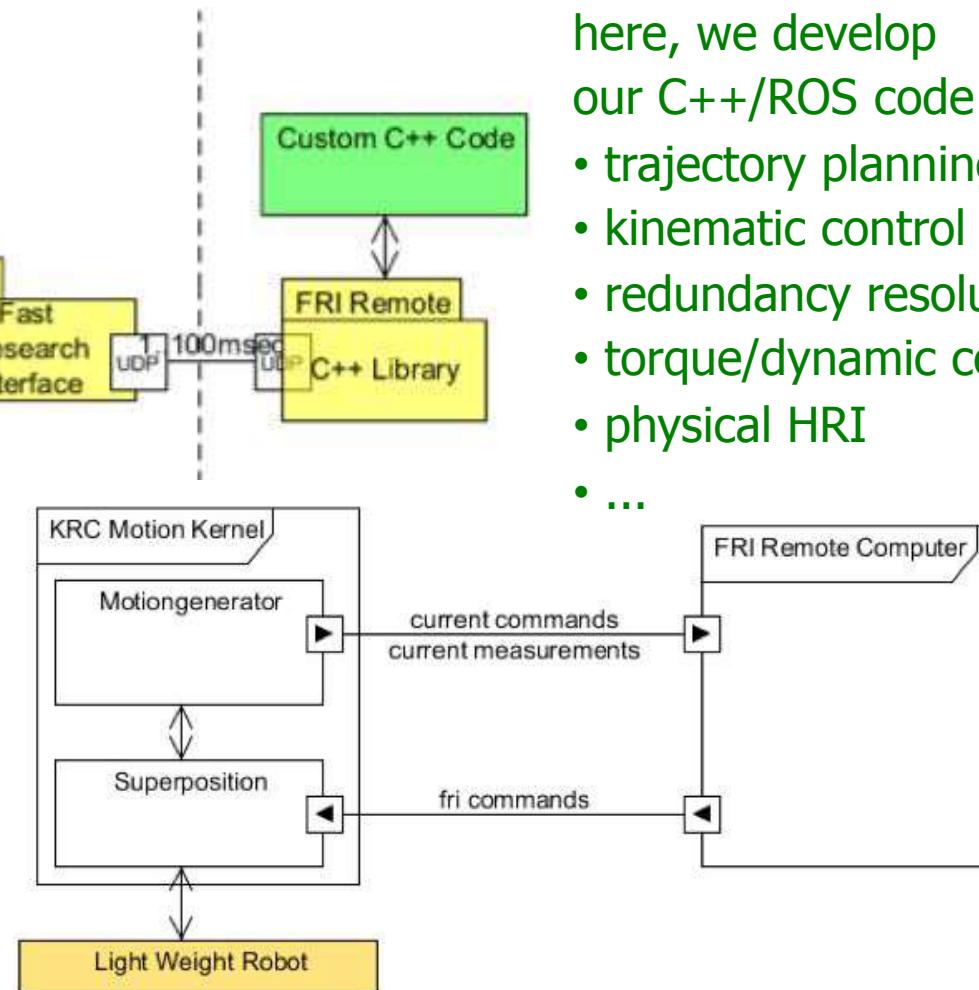


Fast Research Interface (FRI) for KUKA Light Weight Robot (LWR-IV)

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



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**

Discrete-Time Redundancy Resolution
at the Velocity Level with Acceleration/Torque
Optimization Properties

Fabrizio Flacco Alessandro De luca

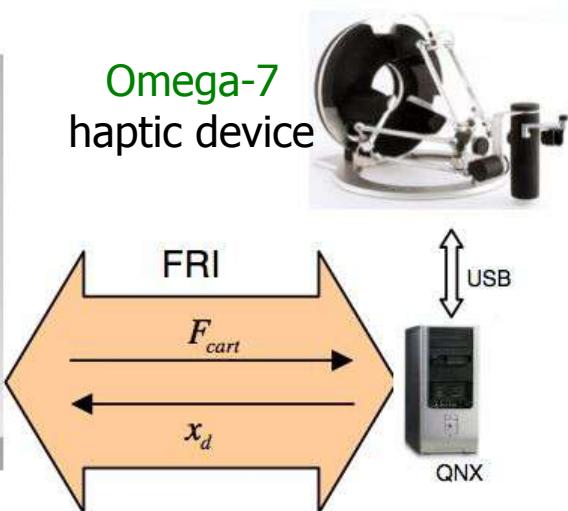
Robotics Lab, DIAG
Sapienza University of Rome

September 2014

[video](#)

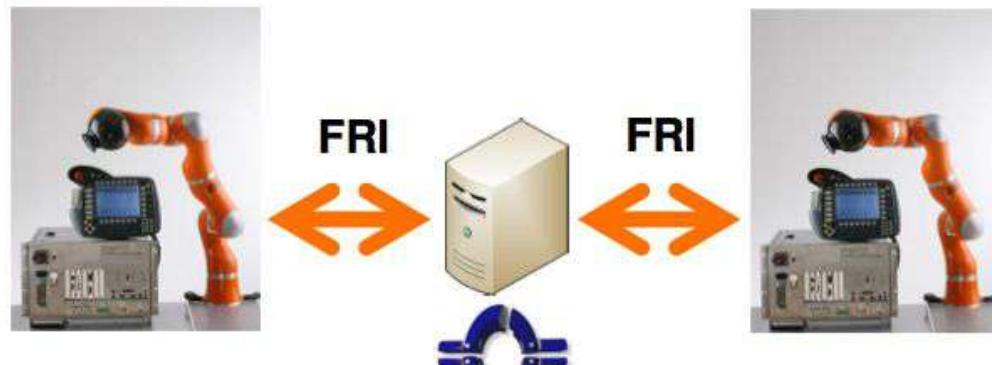


Other uses of the FRI



- haptic feedback to the user

- coordinated dual-arm motion



The Orococos Project
Smarter control in robotics & automation!



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, ...



Robot 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

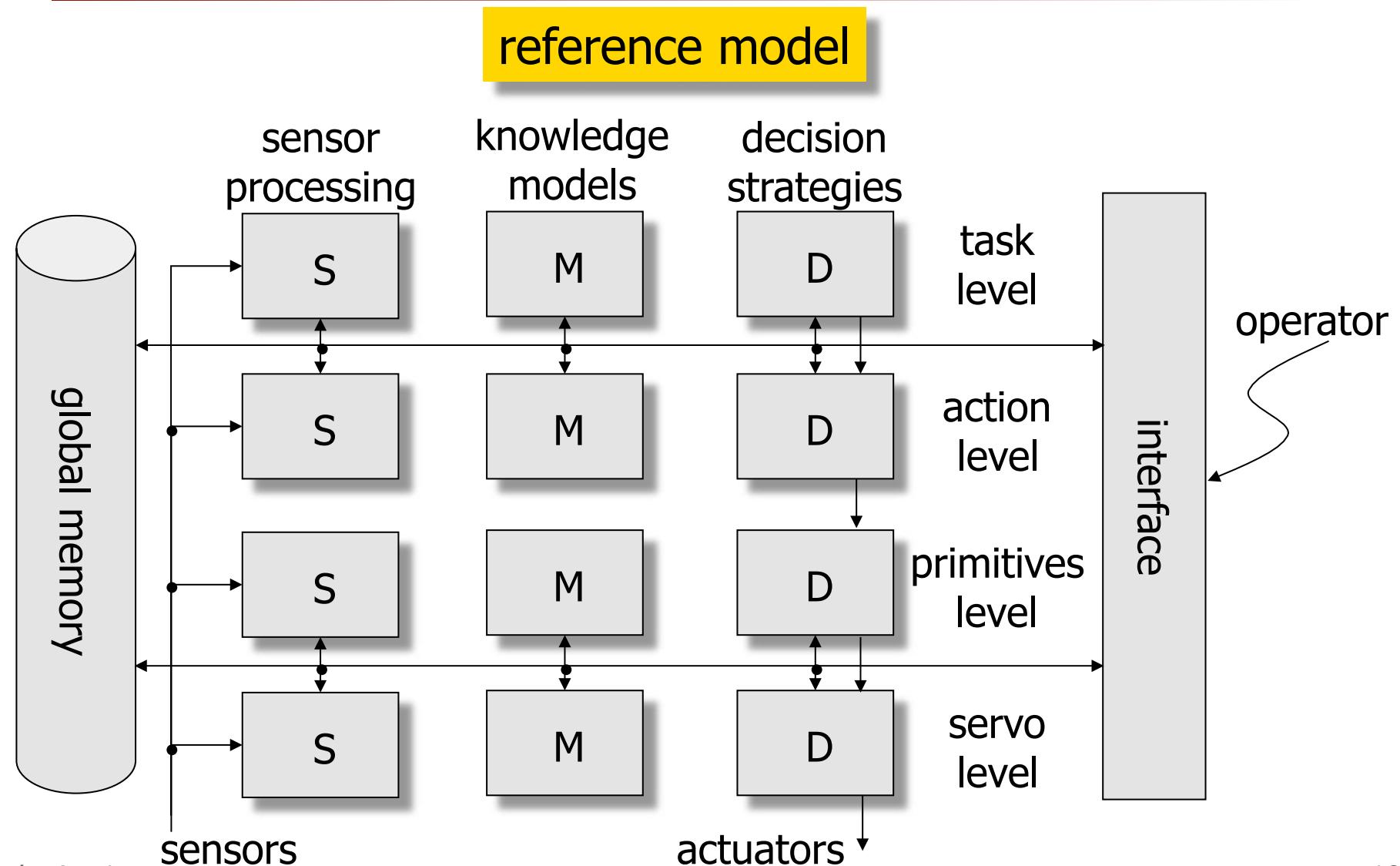


Task-oriented programming

- “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**

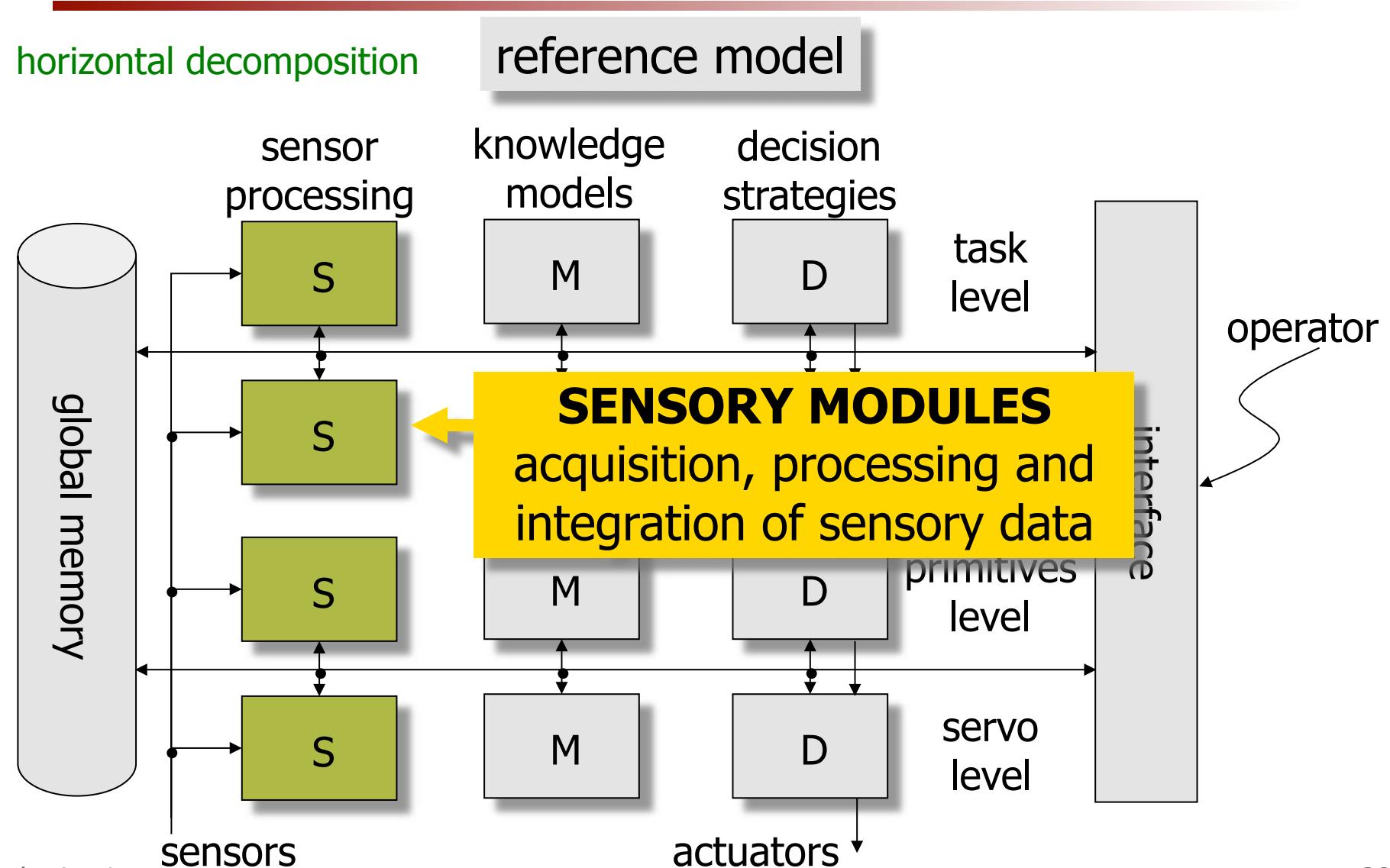


Functional control architecture



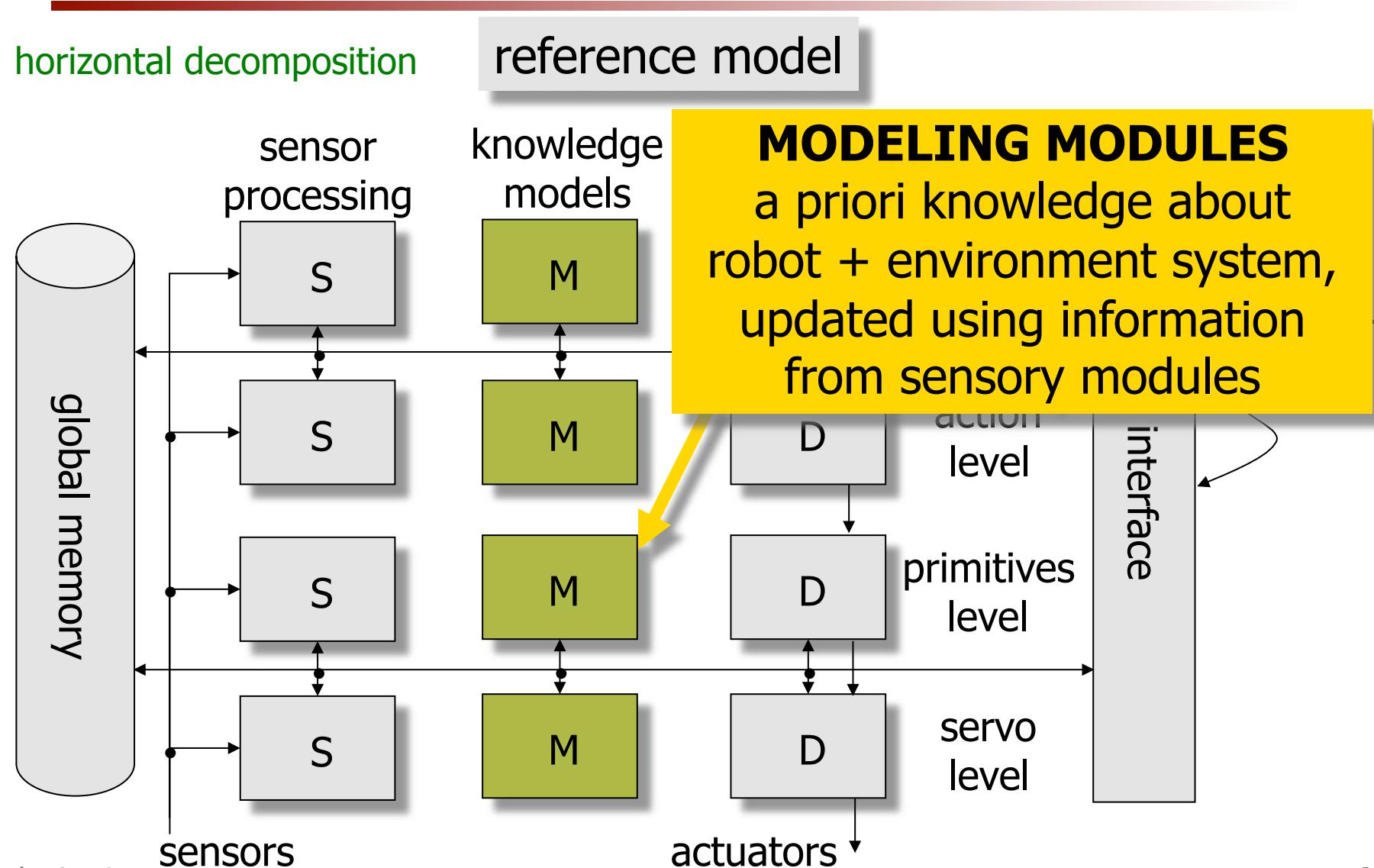


Functional architecture: Modules



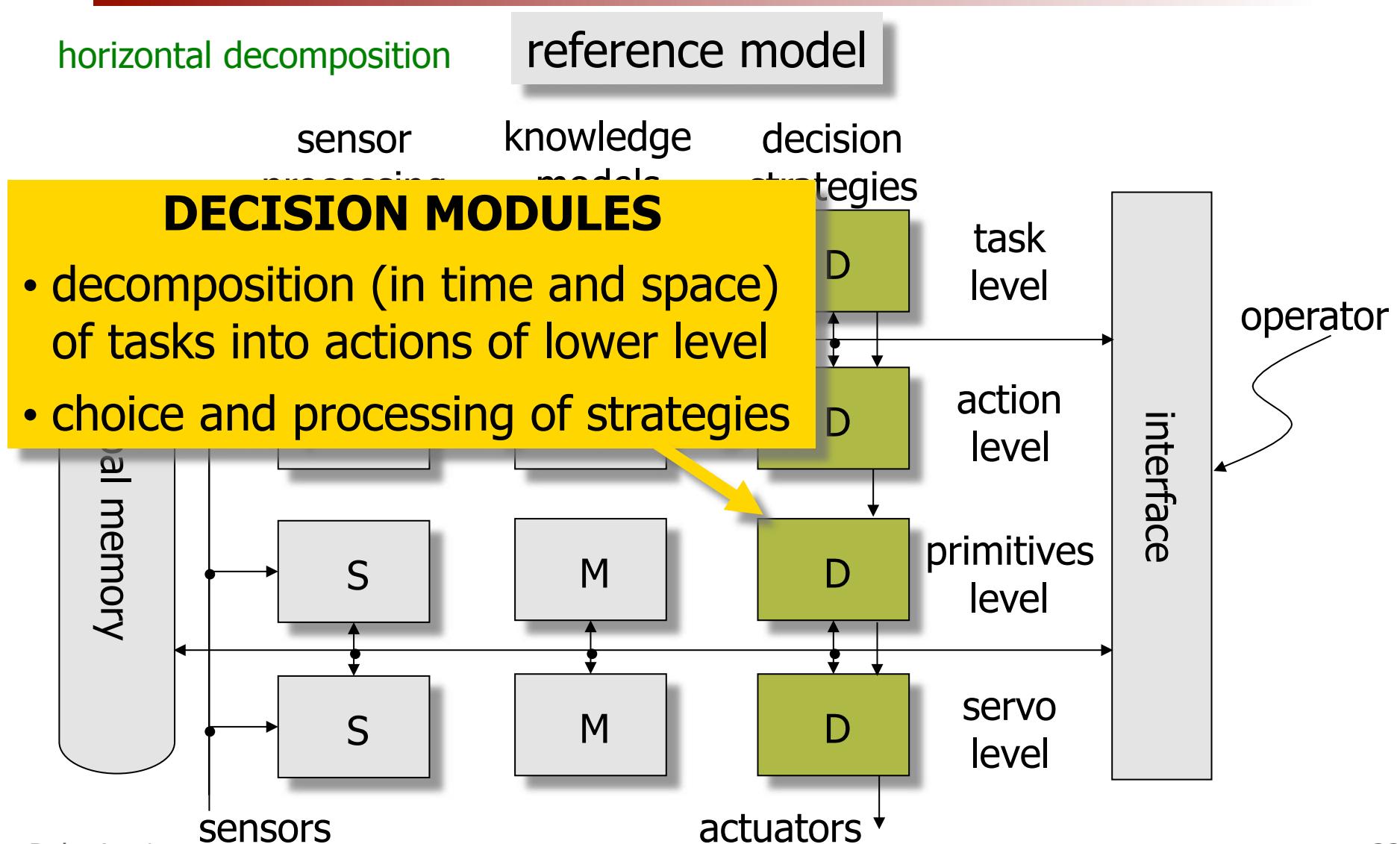


Functional architecture: Modules



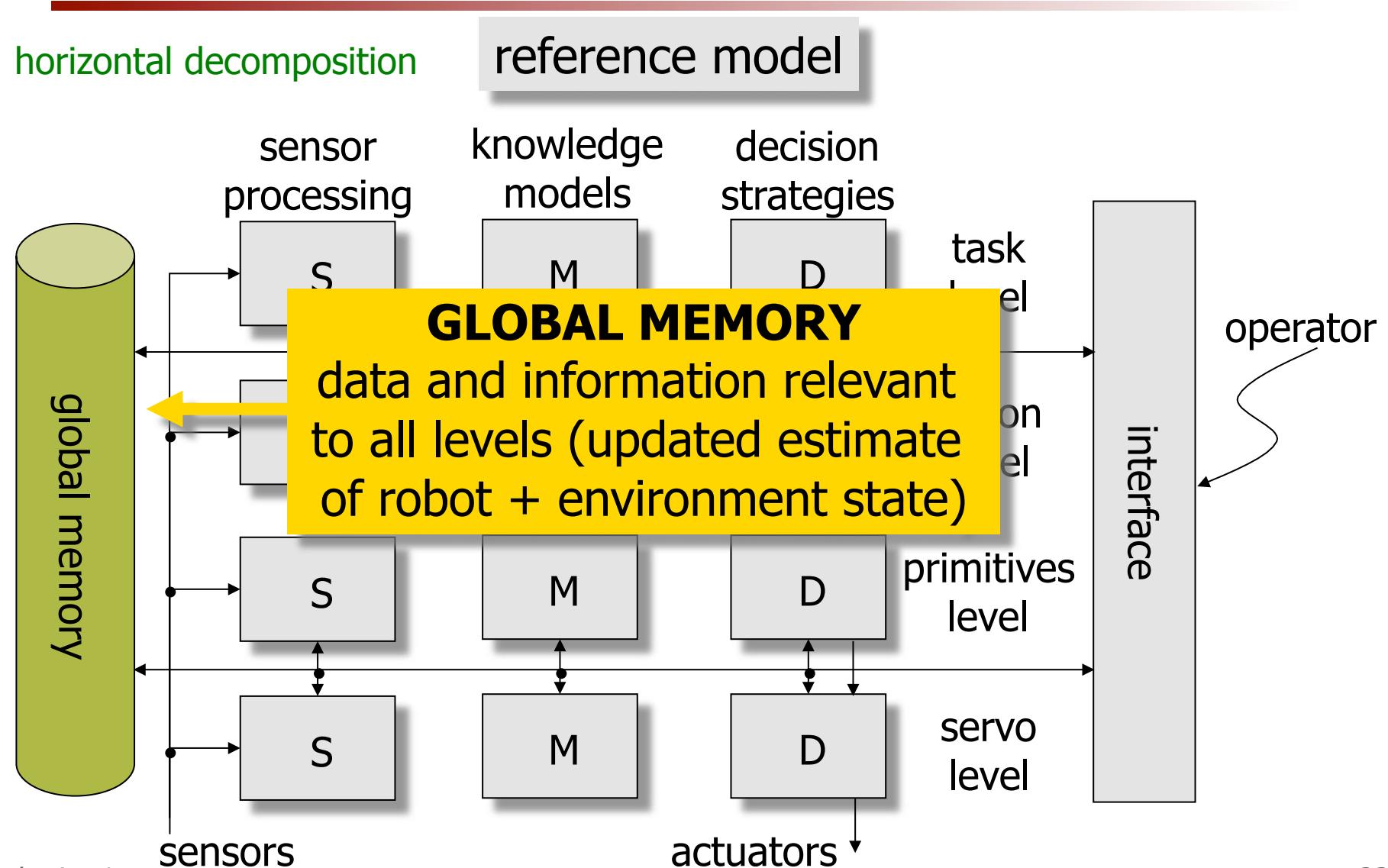


Functional architecture: Modules



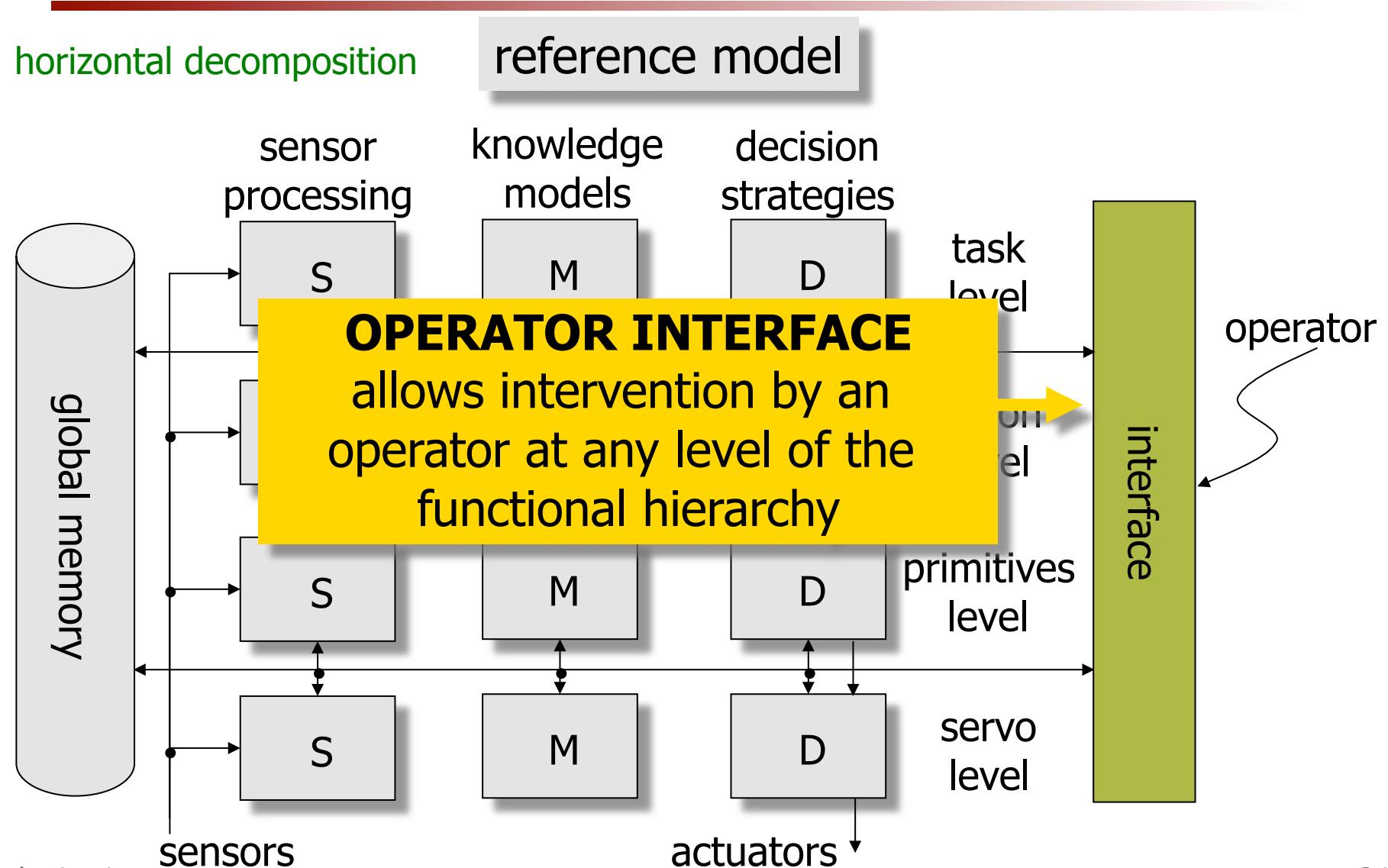


Functional architecture: Modules



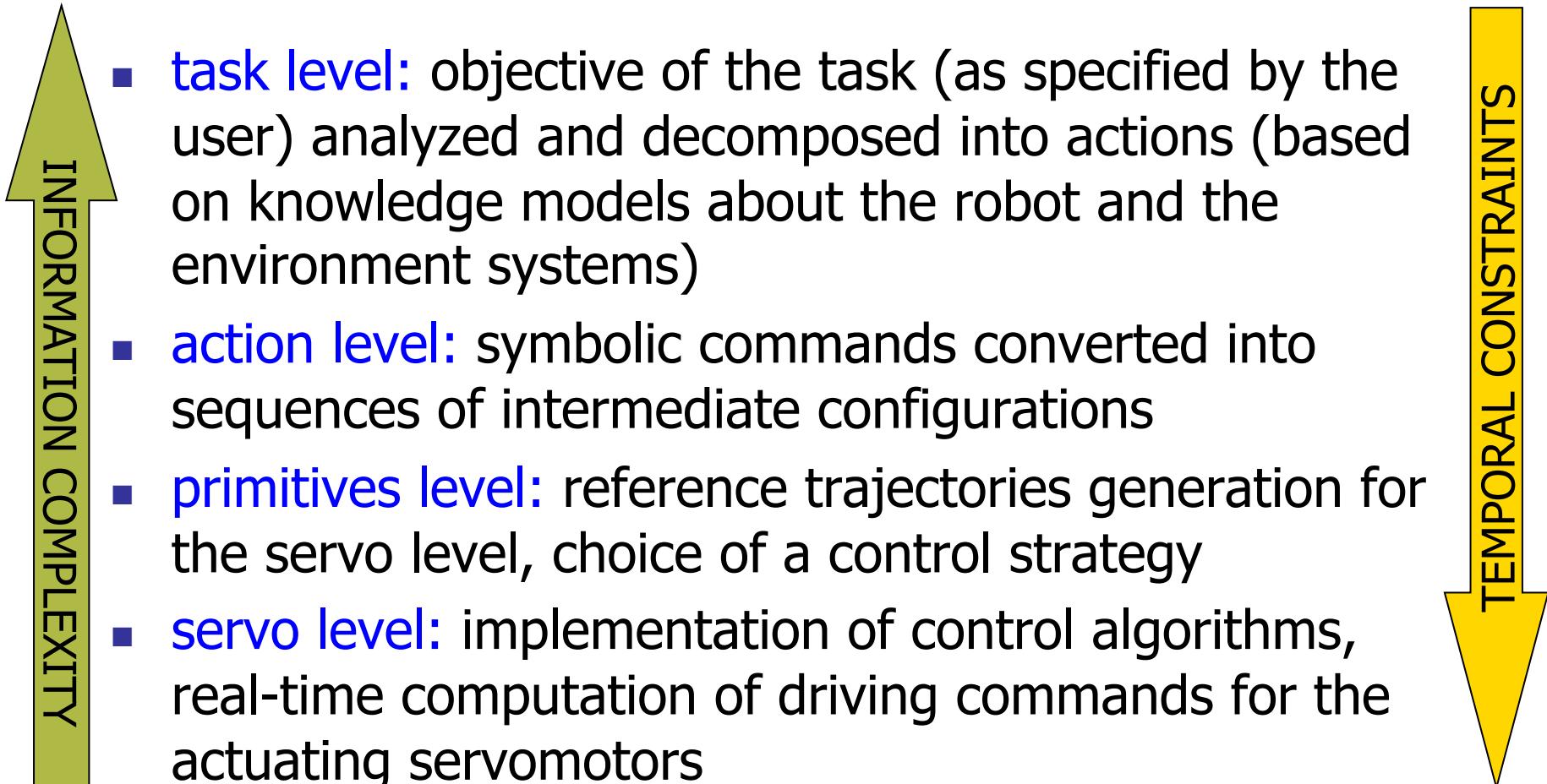


Functional architecture: Modules

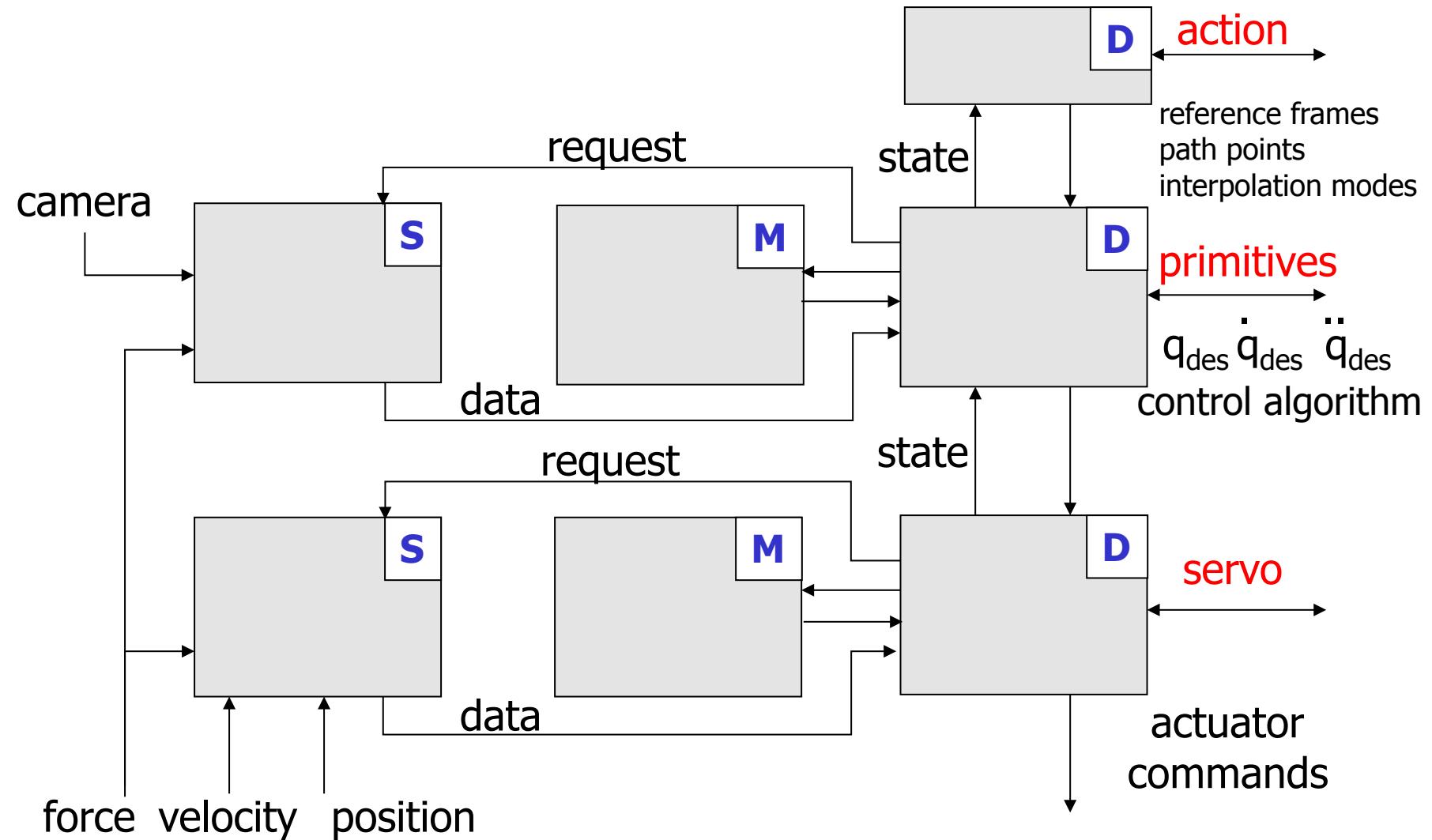




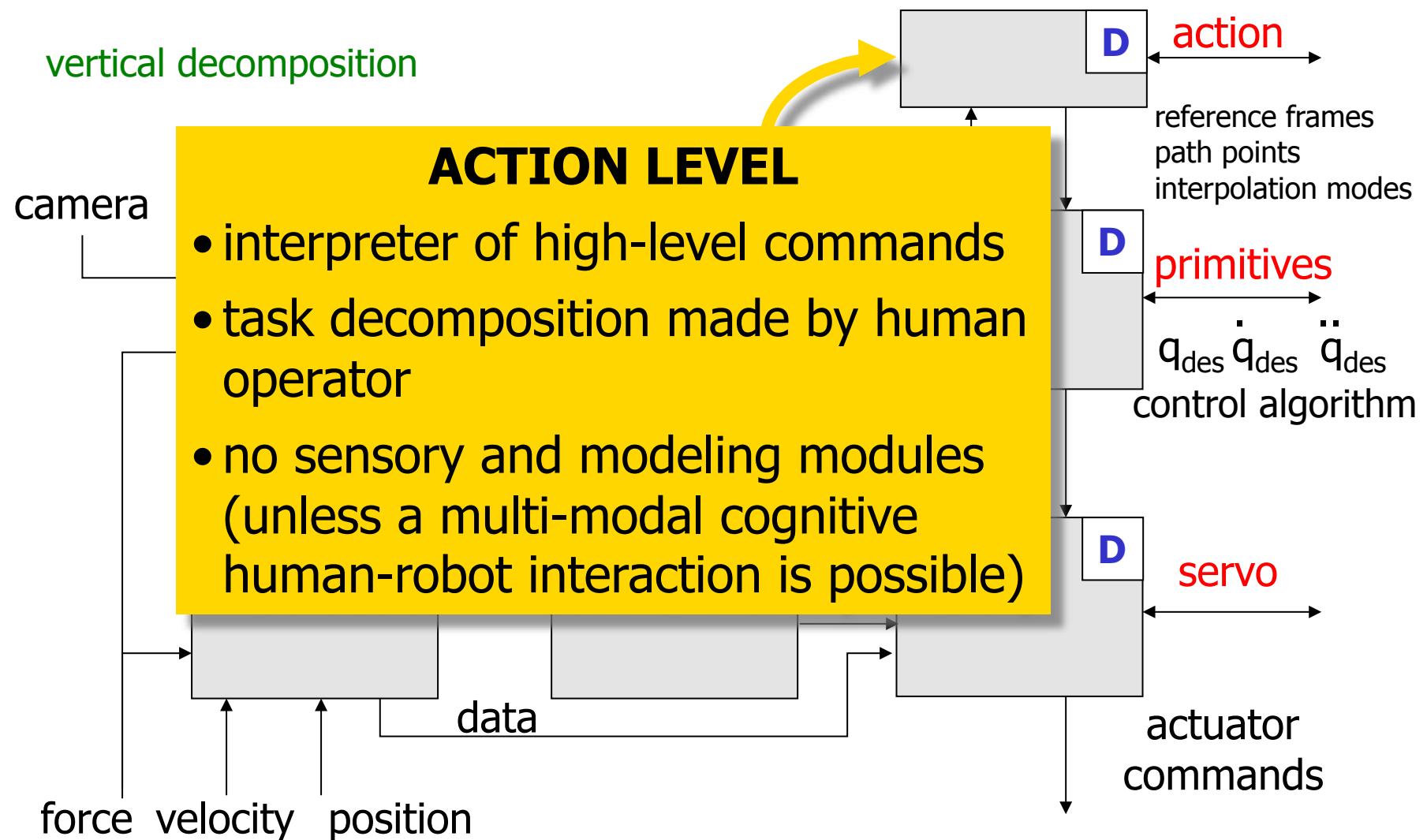
Reference model: Levels



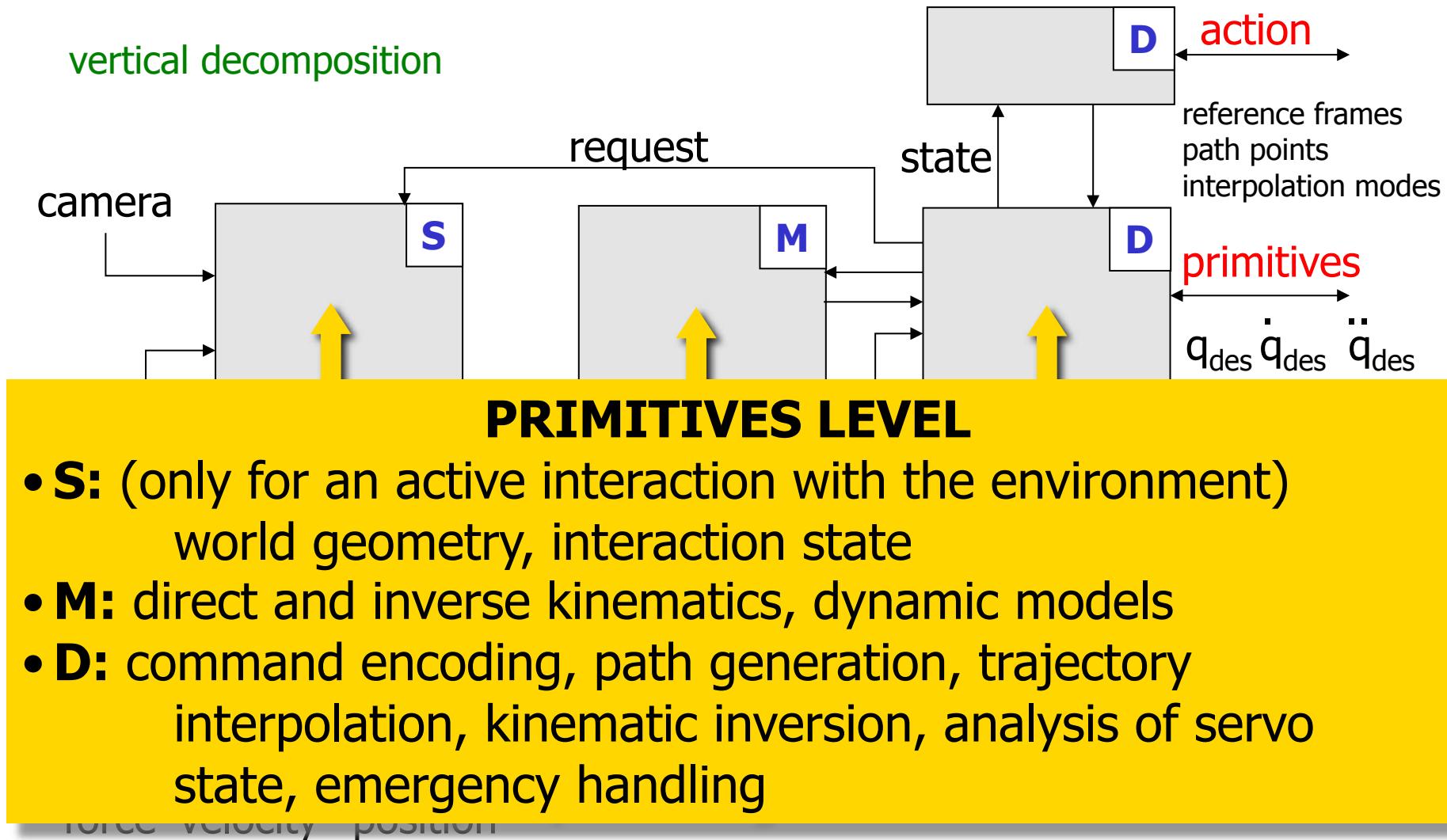
A functional architecture for industrial robots



A functional architecture for industrial robots



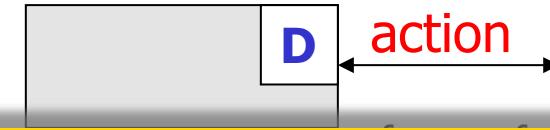
A functional architecture for industrial robots





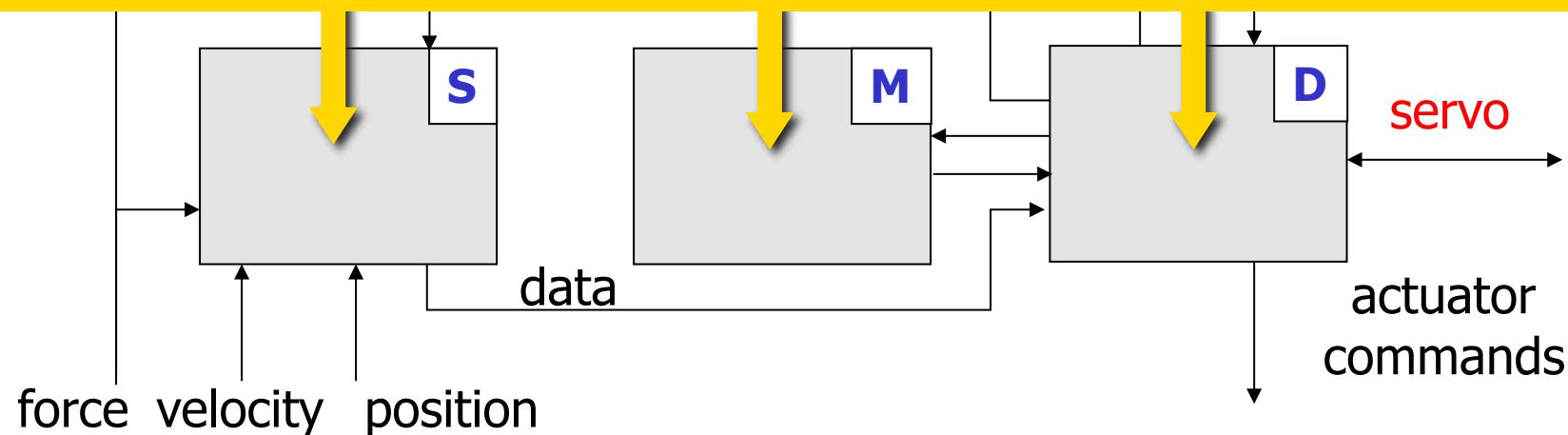
A functional architecture for industrial robots

vertical decomposition



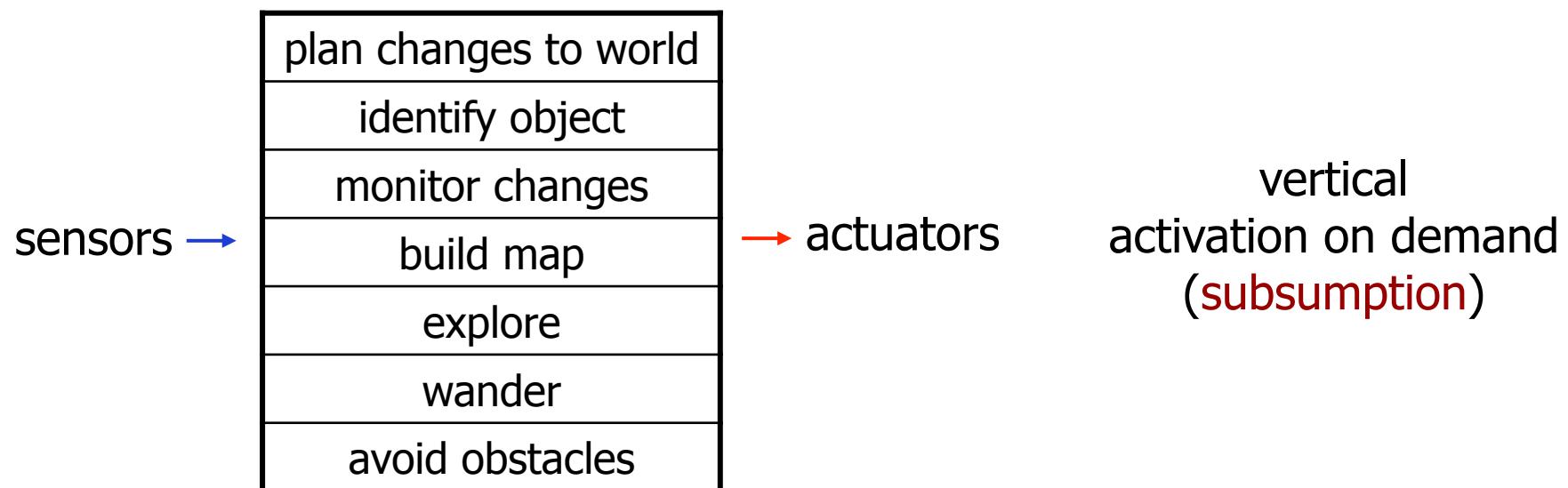
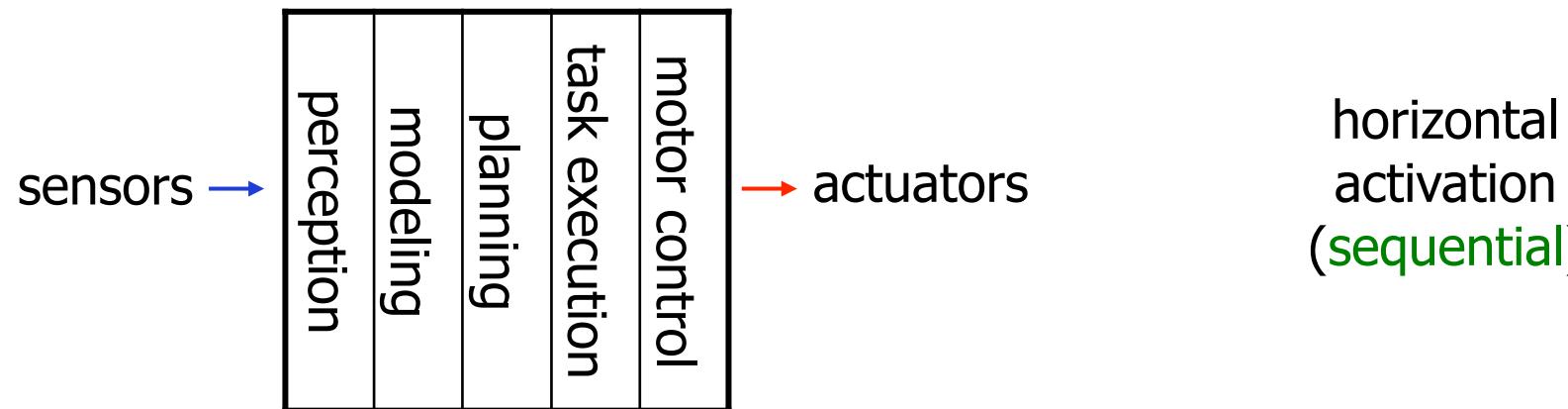
SERVO LEVEL

- **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



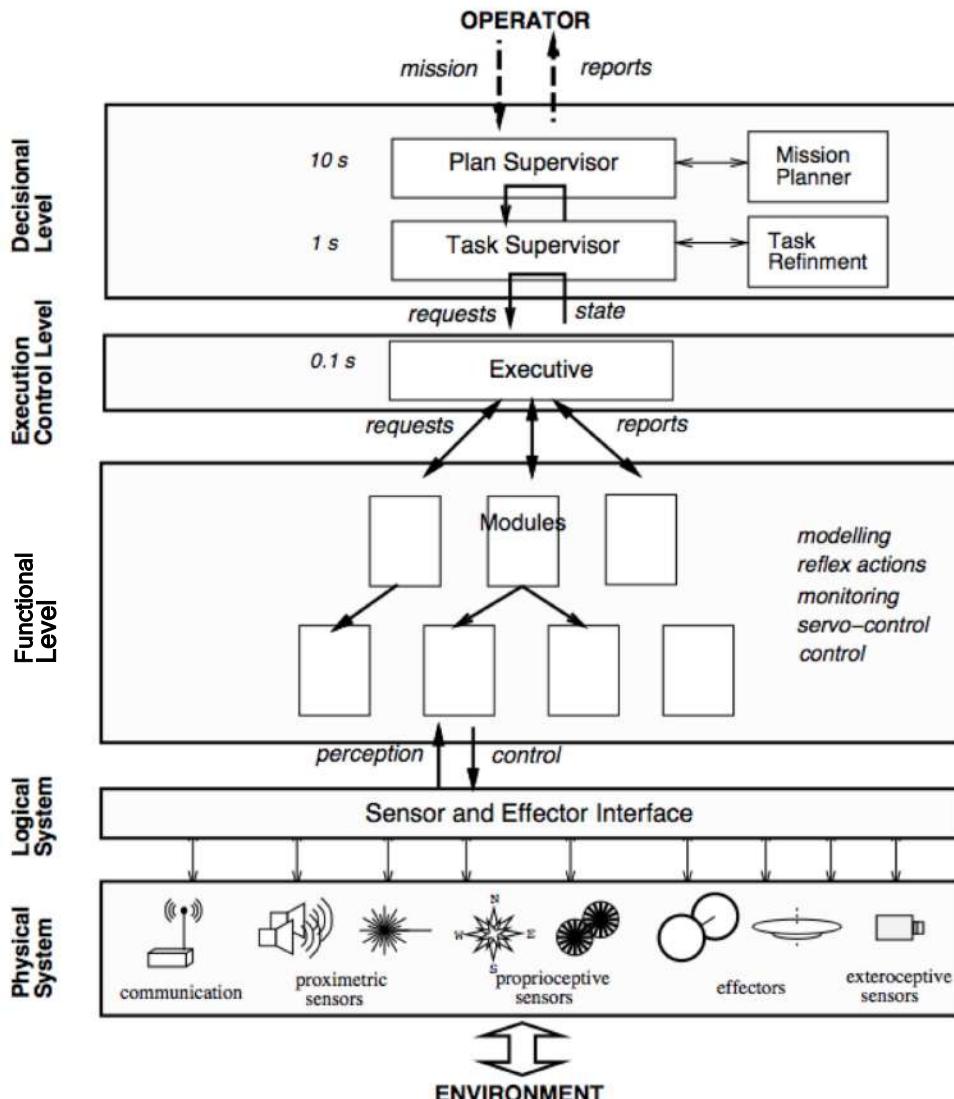


Interaction among modules





LAAS architecture

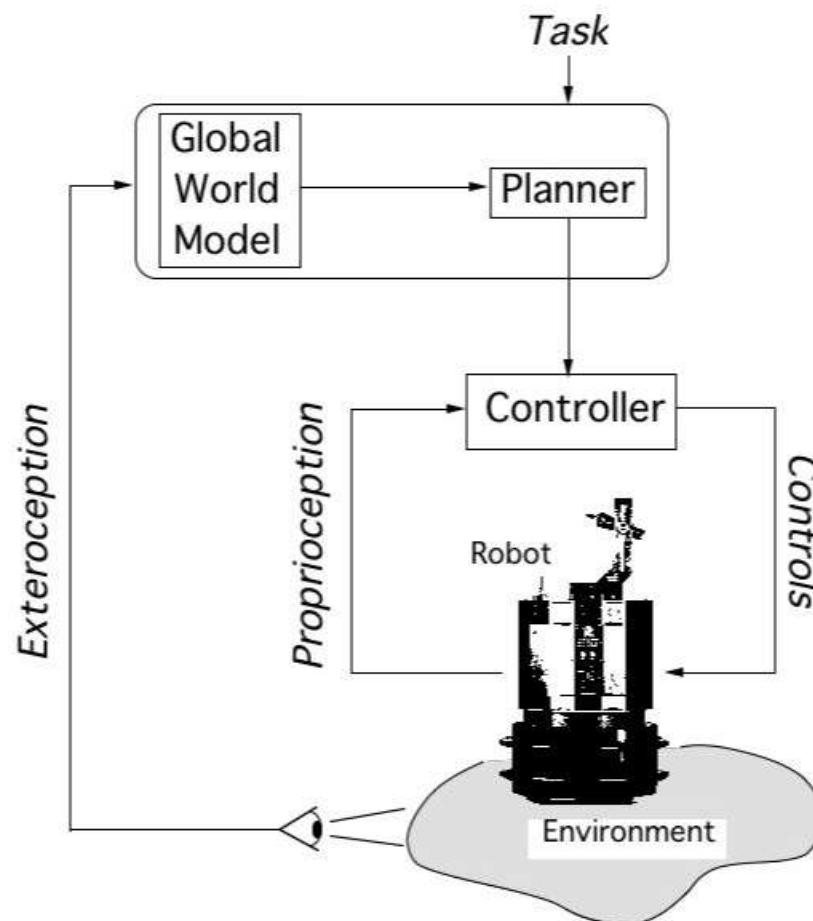


- alternative example by LAAS/CNRS in Toulouse
- five levels
 - decision
 - execution (synchronization)
 - functional (modules)
 - logical for interface
 - physical devices

R. Alami *et al.*
 "An Architecture for Autonomy,"
Int. J. of Robotics Research, 1998



Development of architectures - 1

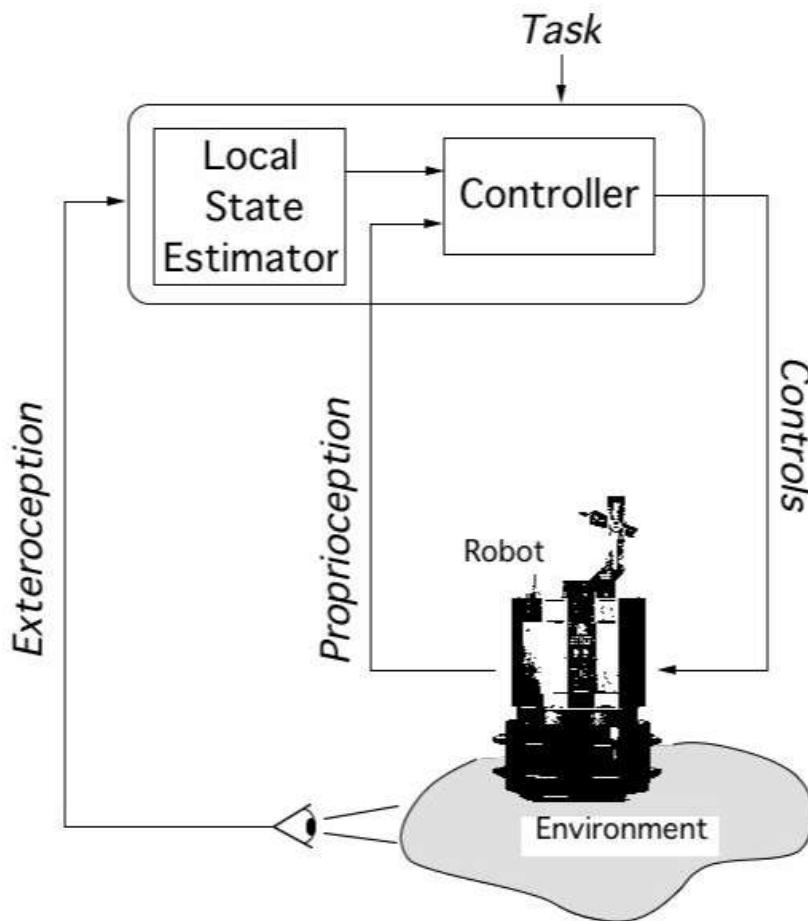


*example: a navigation task
for a wheeled mobile robot*

- 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)



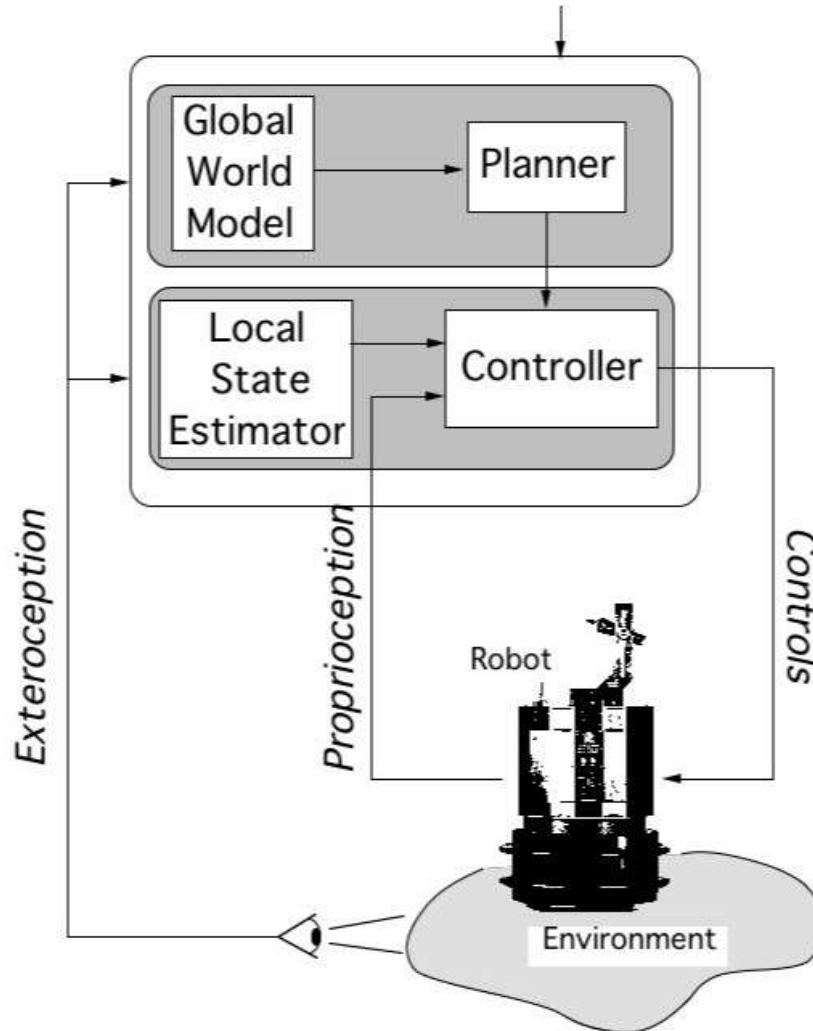
Development of architectures - 2



- 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

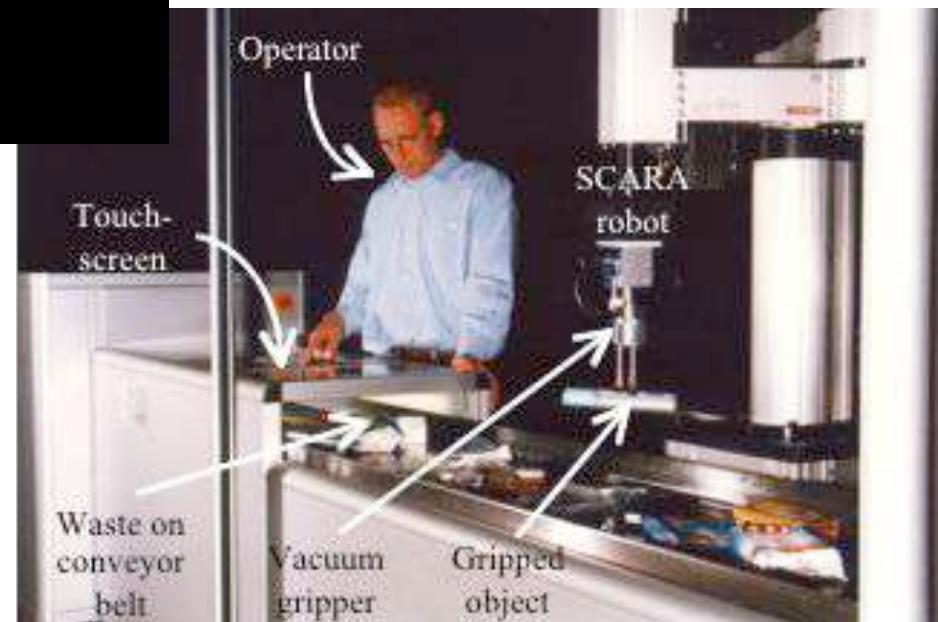


IPA robotic cell for garbage collection and separation for recycling



video

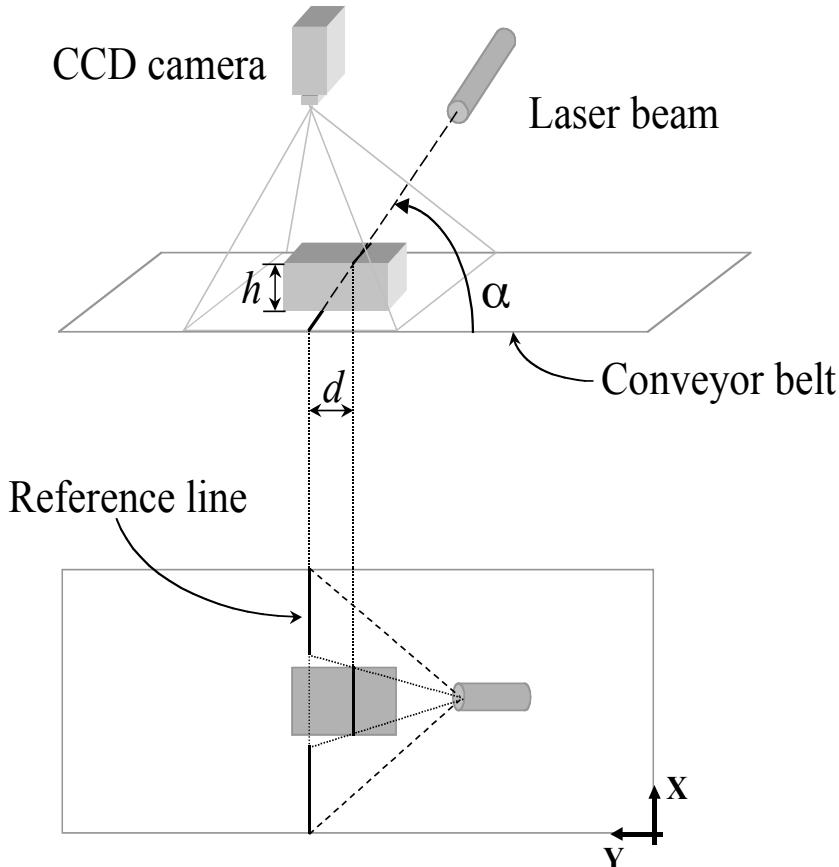
semi-automatic version
at Fraunhofer IPA
Stuttgart, 1997



objective: replace operator



Sensory module in fully automatic version

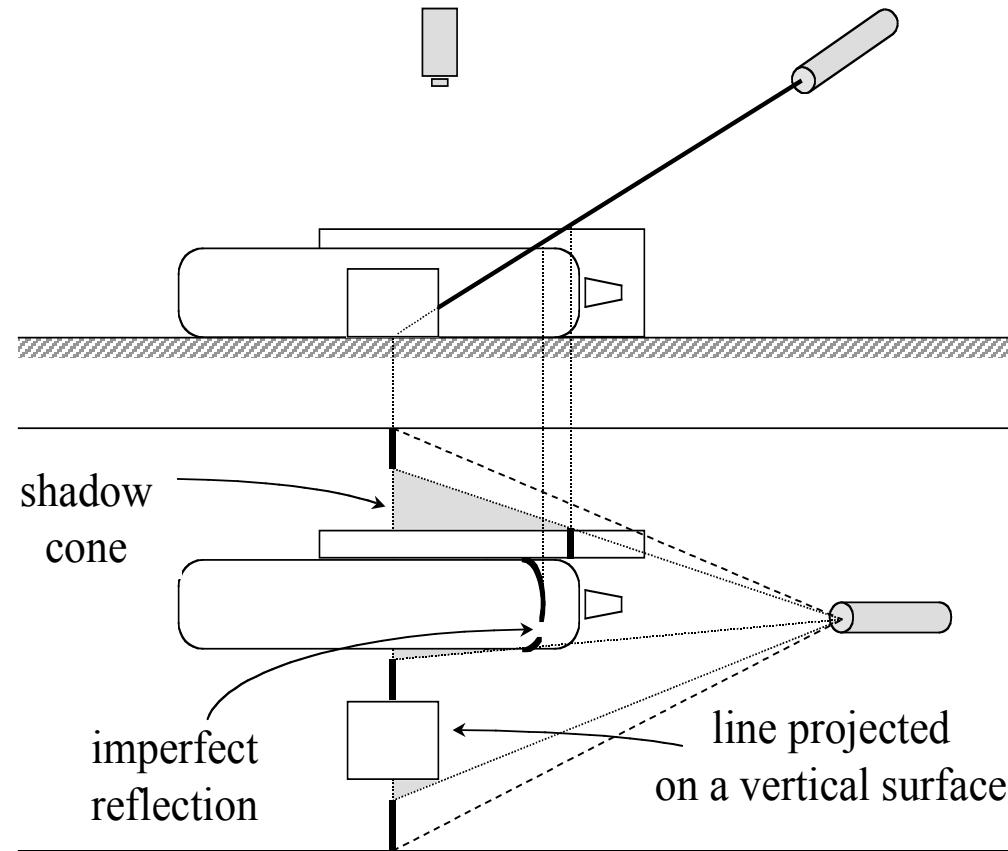


operator
+
touch-screen
replaced by
structured light vision
+
neuro-fuzzy system
for object localization
and classification

operation principle
of the structured light sensor



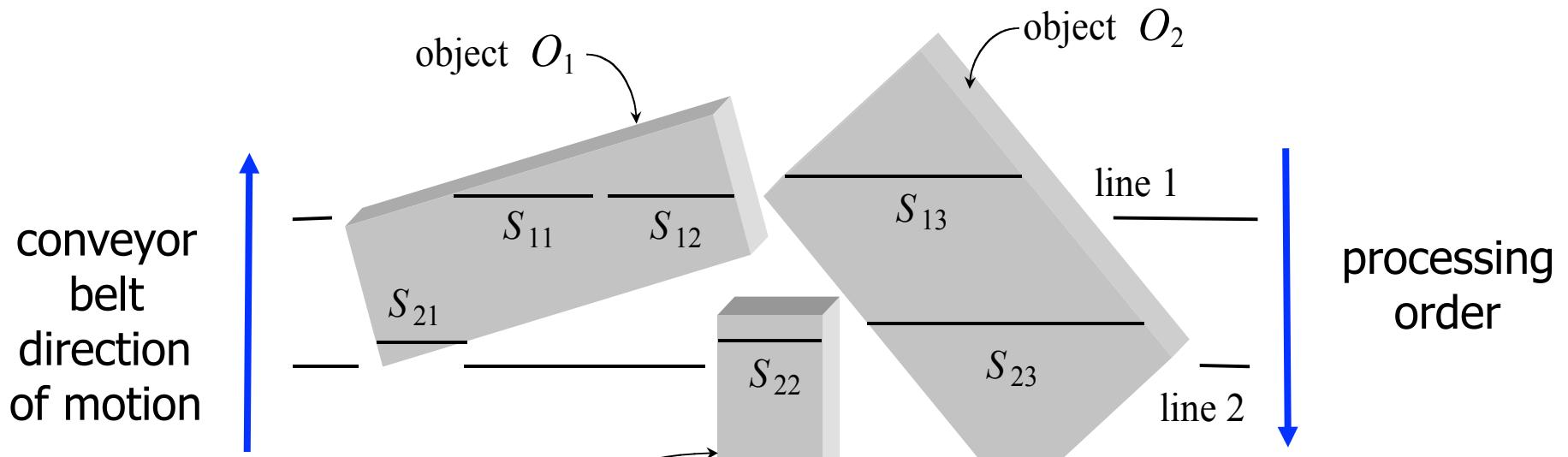
Sensory data interpretation



possible sources of lack of information on a single line scan



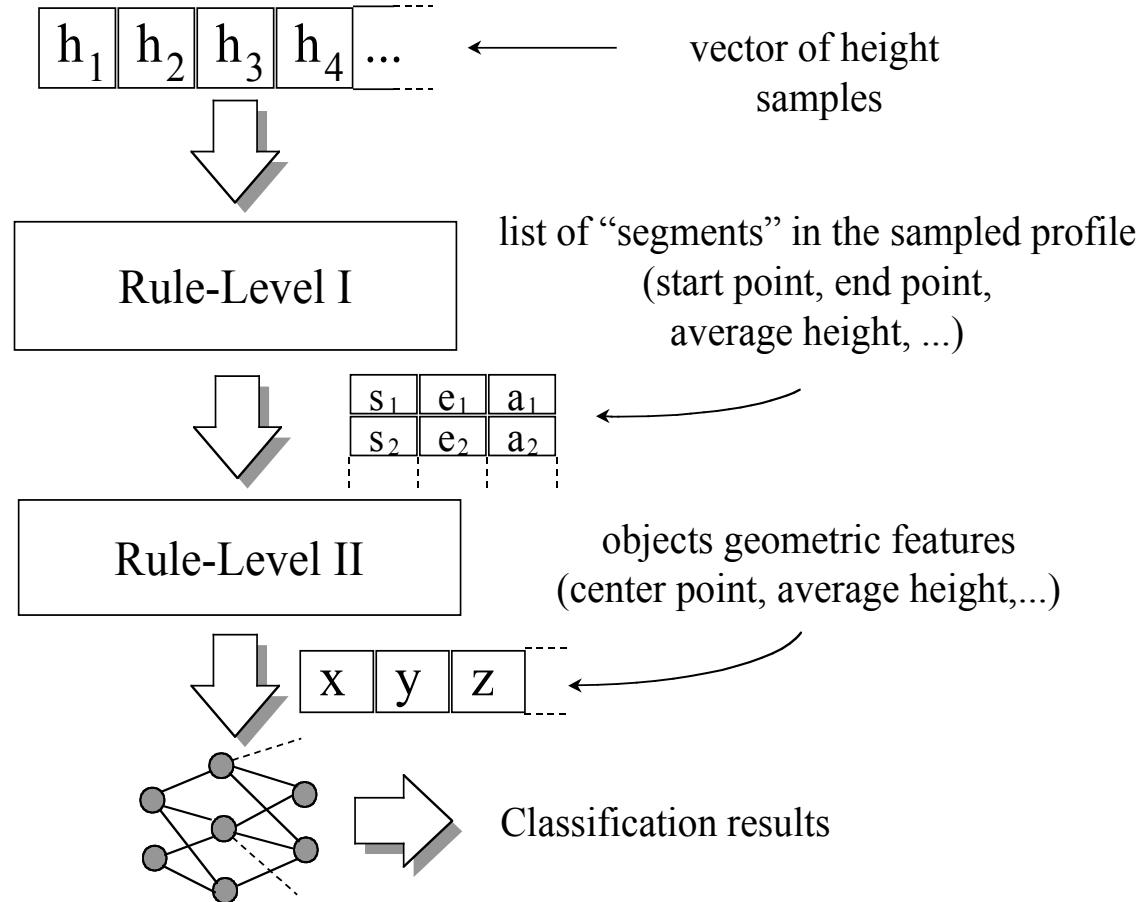
Sensory data interpretation



integration of data collected
in successive sampling instants



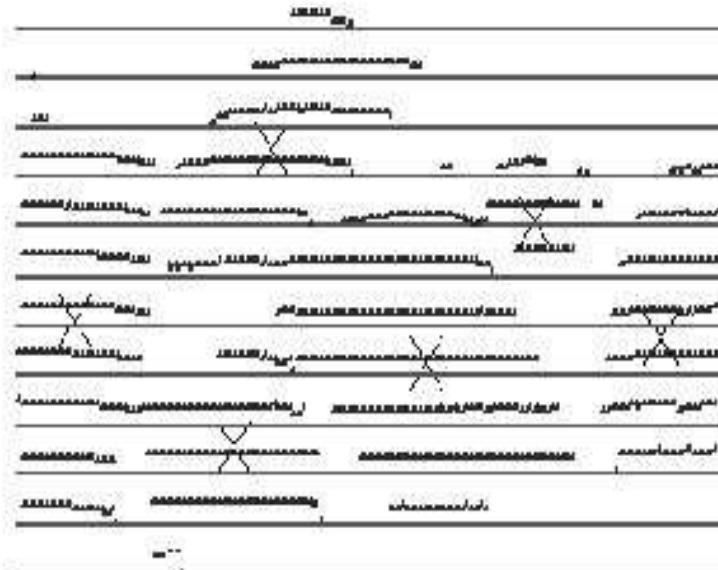
Decision module



structure of the object localization and classification module



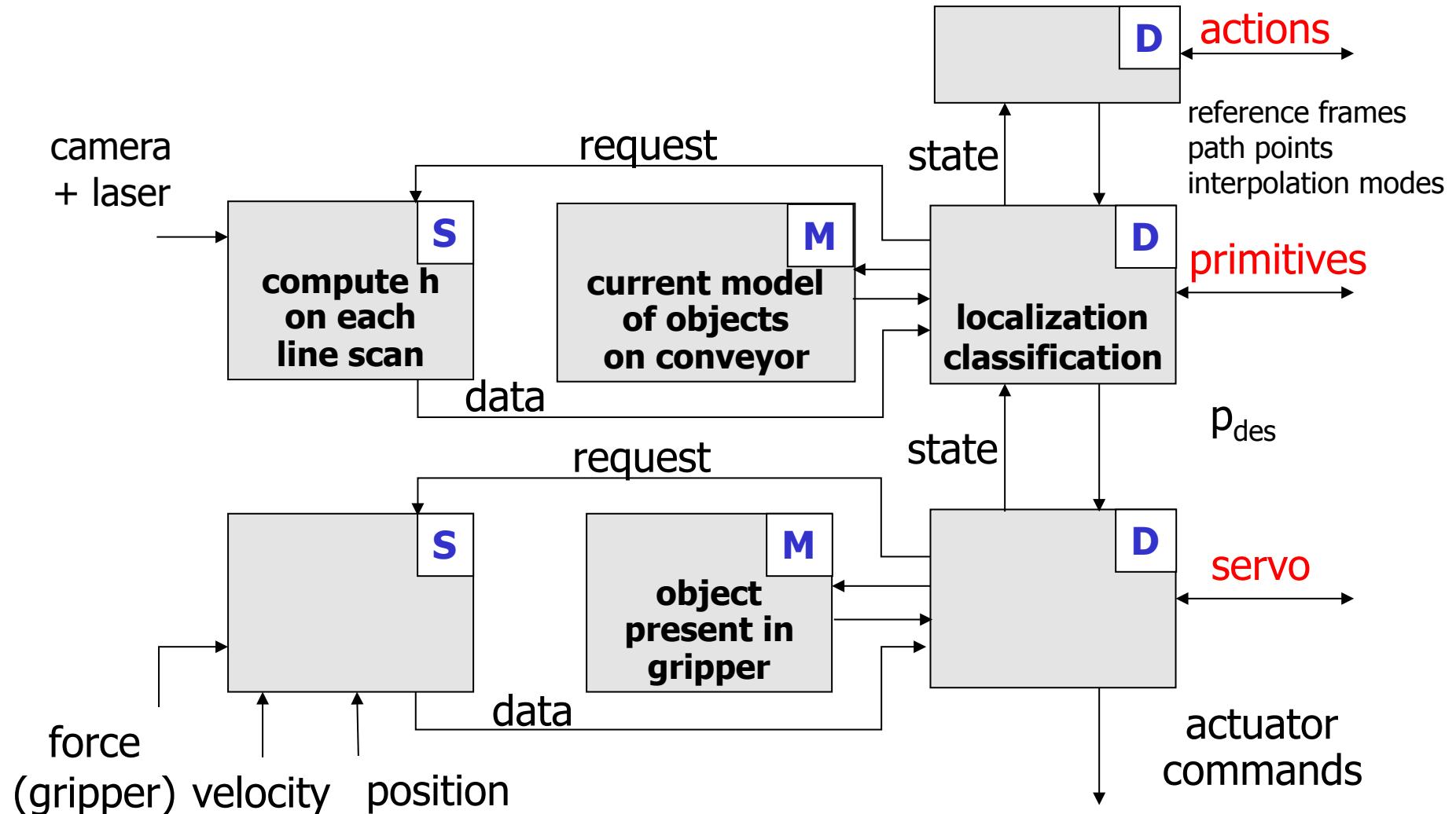
Modeling module



example of models for objects on the conveyor belt



Functional architecture of the IPA cell





Test results

[video](#)

Automatic robotized garbage collection

Raffaella Mattone, Linda Adduci

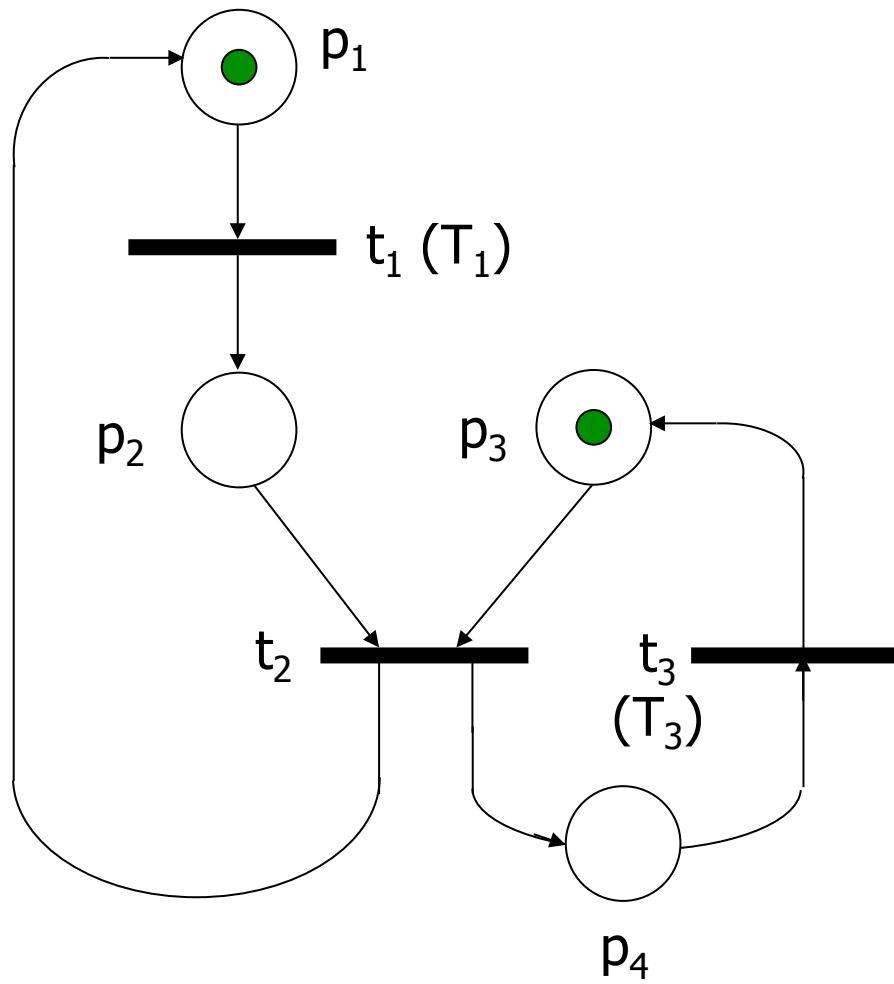
c/o Fraunhofer IPA, Stuttgart, 1997

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

work by Dr. Raffaella Mattone (PhD @ DIS)



Flow diagrams of operation



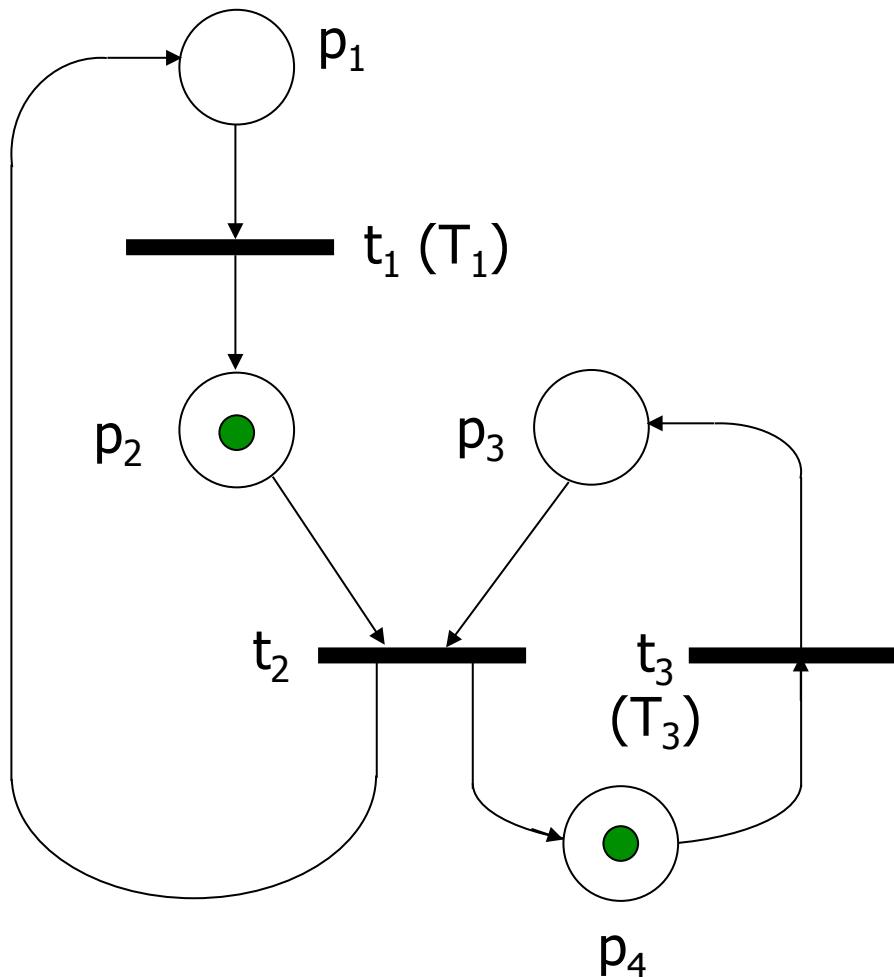
PETRI NETS

oriented graphs with
two types of nodes

- **places** (p_1, \dots, p_4)
states or functional blocks:
active if a “token” is present
(e.g., p_1 and p_3)
- **transitions** (t_1, \dots, 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

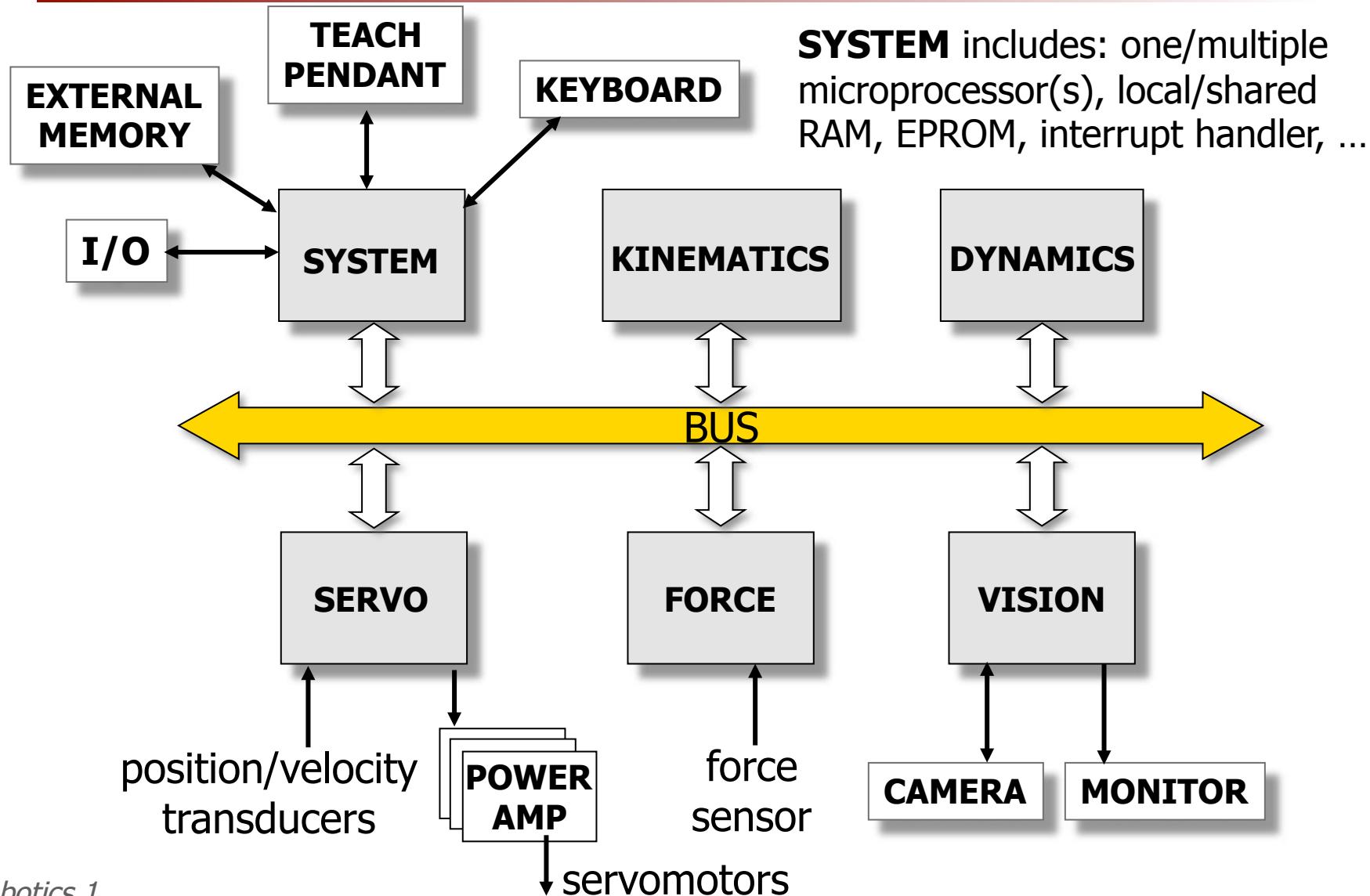


- p_1 : robot picking & placing
 - T_1 : pick & place time
- p_2 : robot ready
- p_3 : new part on conveyor
- p_4 : waiting for a part
 - T_3 (random variable): time interval between two successive parts

initial marking/state:
robot ready, waiting for a part



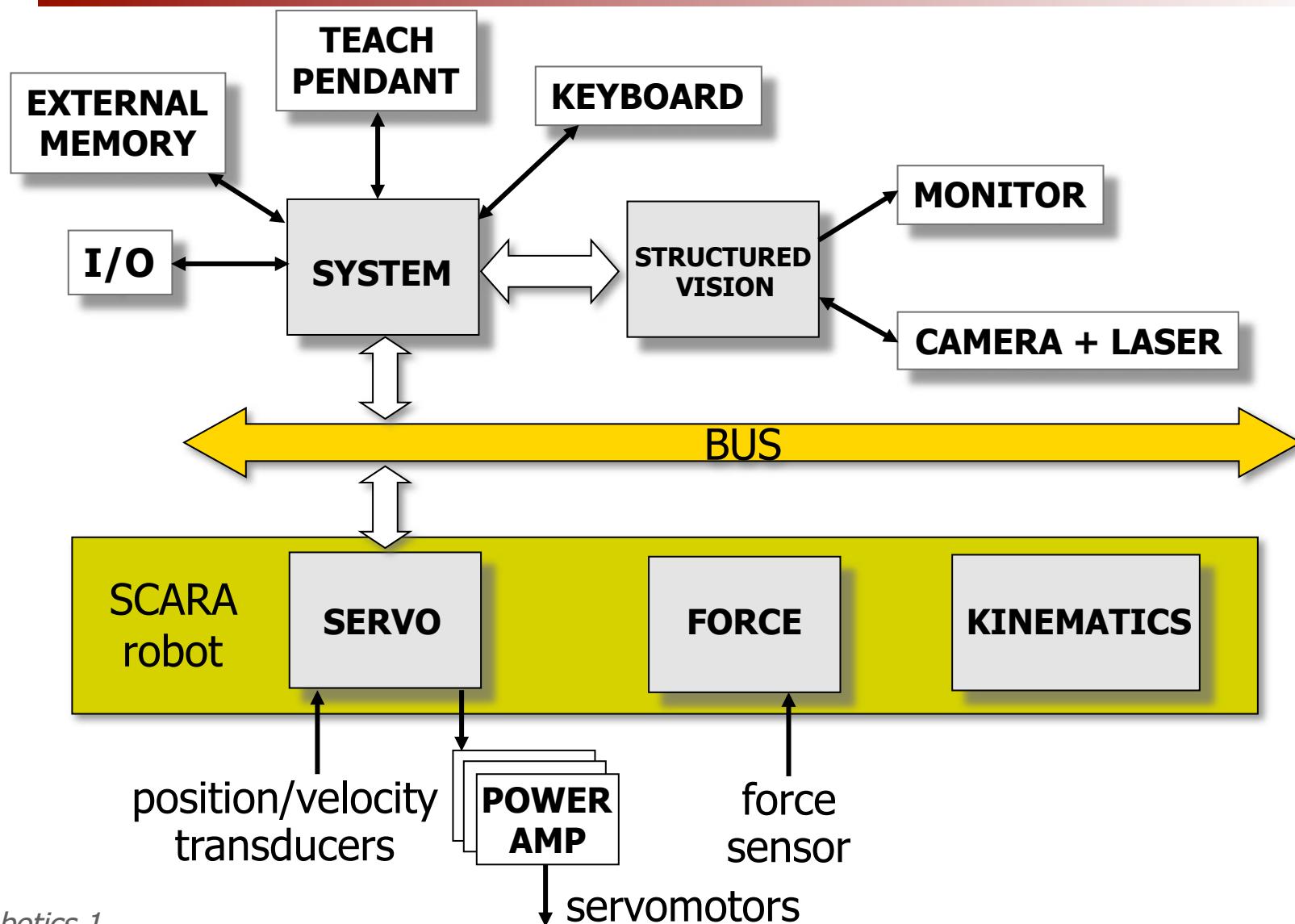
Hardware architecture





Hardware architecture

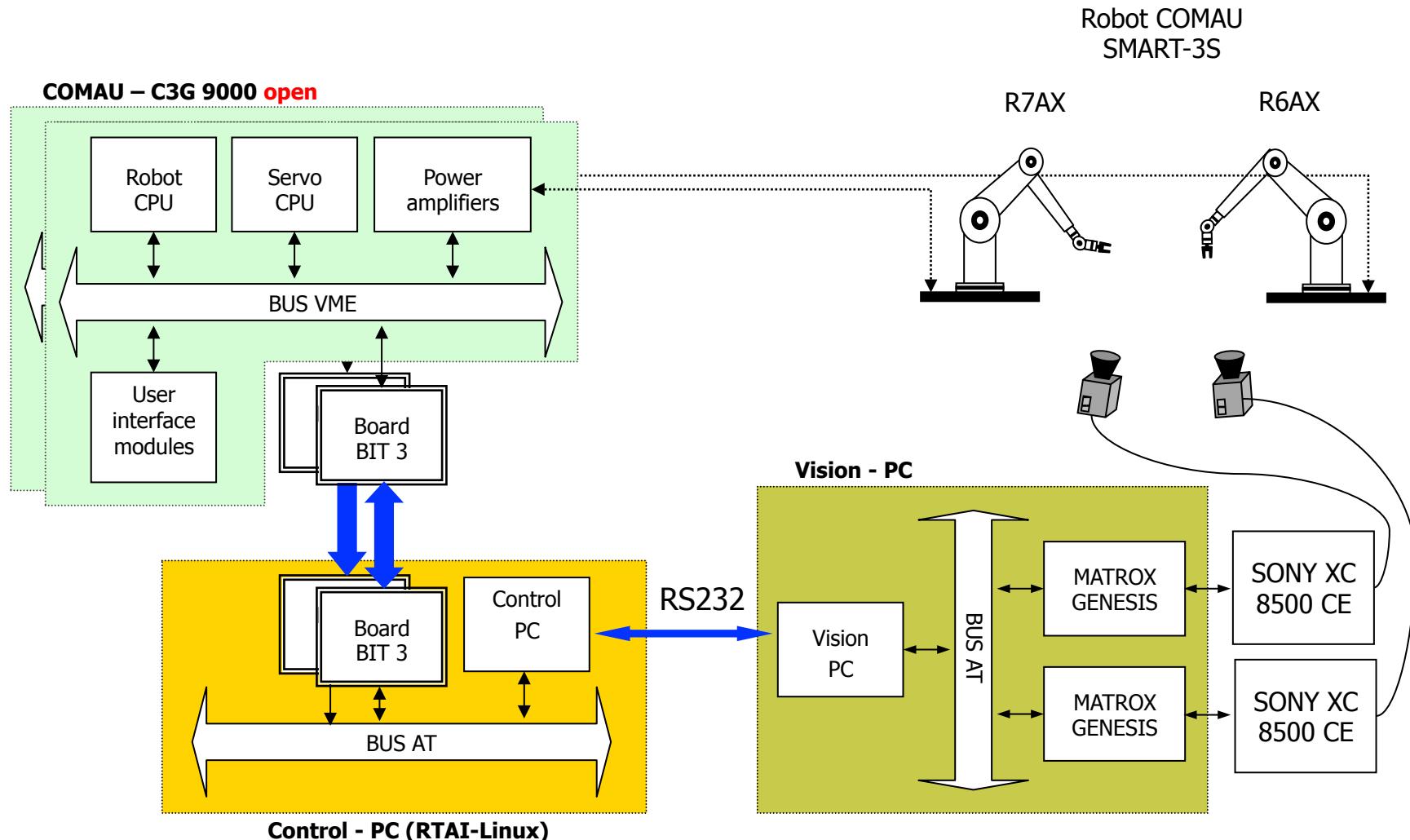
Example of the IPA cell





Hardware architecture

Example including vision in an open controller





Robotics 1

Position and orientation of rigid bodies

Prof. Alessandro De Luca

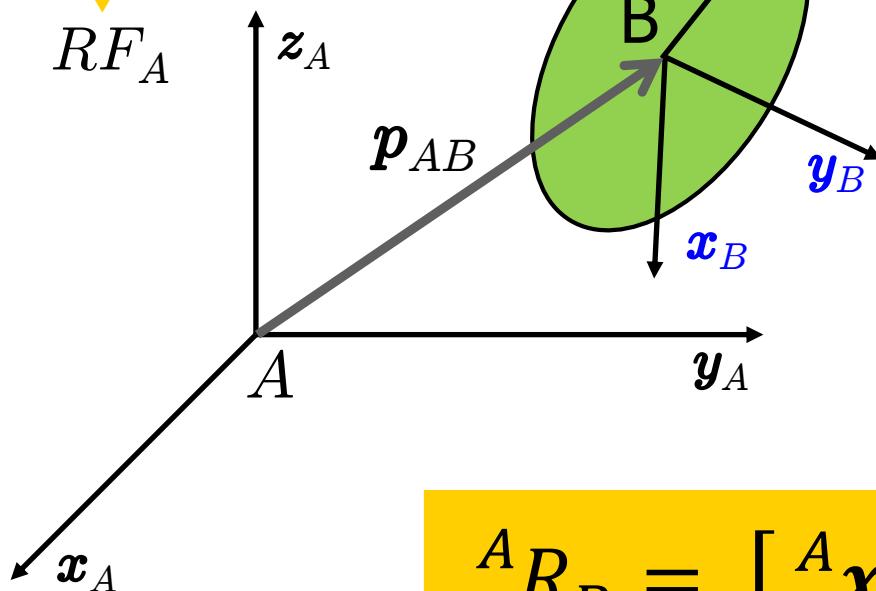
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Position and orientation

right-handed orthogonal
Reference Frames



rigid body

- position: ${}^A\mathbf{p}_{AB}$ (vector $\in \mathbb{R}^3$)
Cartesian coordinates of vector \overrightarrow{AB}
expressed in RF_A
- orientation:
orthonormal 3×3 matrix
($R^T = R^{-1} \Rightarrow R^T R = I$), with $\det = +1$

$${}^A R_B = [{}^A \mathbf{x}_B \quad {}^A \mathbf{y}_B \quad {}^A \mathbf{z}_B]$$

- $\mathbf{x}_A \mathbf{y}_A \mathbf{z}_A$ ($\mathbf{x}_B \mathbf{y}_B \mathbf{z}_B$) are axis vectors (of unitary norm) of frame RF_A (RF_B)
- components in ${}^A R_B$ are the **direction cosines** of the axes of RF_B with respect to (w.r.t.) RF_A



Position of a rigid body

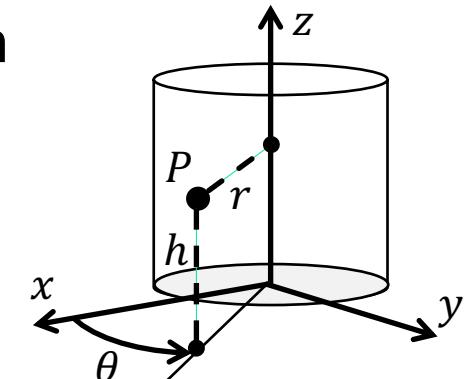
- for position representation, use of other coordinates than the Cartesian ones is possible, e.g., cylindrical or spherical
- direct** transformation from **cylindrical** to Cartesian

$$x = r \cos \theta$$

$$y = r \sin \theta$$

$$z = h$$

is always **well defined**
(with $r > 0$ or $r \gtrless 0$)



- inverse** transformation from **Cartesian** to cylindrical

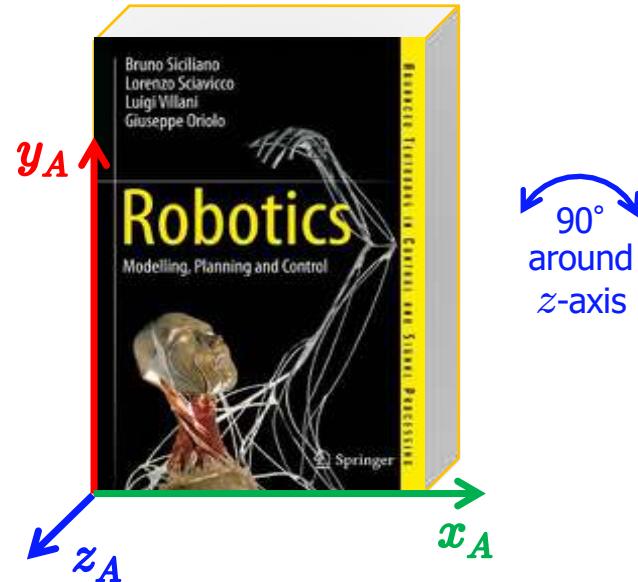
$$\begin{aligned} x^2 + y^2 &= r^2 && \text{assuming } + \\ \frac{y}{x} &= \tan \theta &\Rightarrow& \quad r = \sqrt{x^2 + y^2} \\ &&& \theta = \text{atan2}\{y, x\} \\ &&& h = z \end{aligned}$$

(r > 0 only) four-quadrant arc tangent

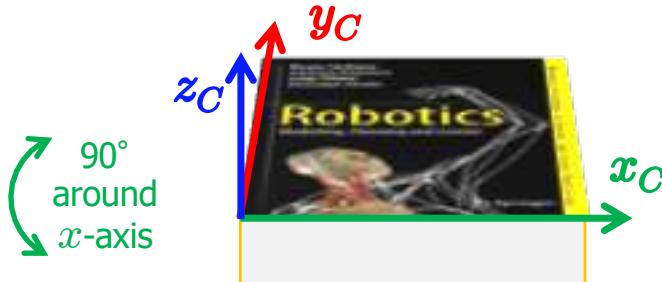
with a **singularity**
for $x = y = 0$



Orientation of a rigid body



$${}^B R_A = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = {}^A R_B^T$$



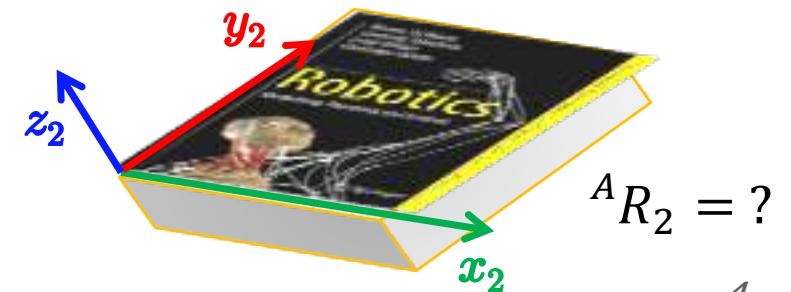
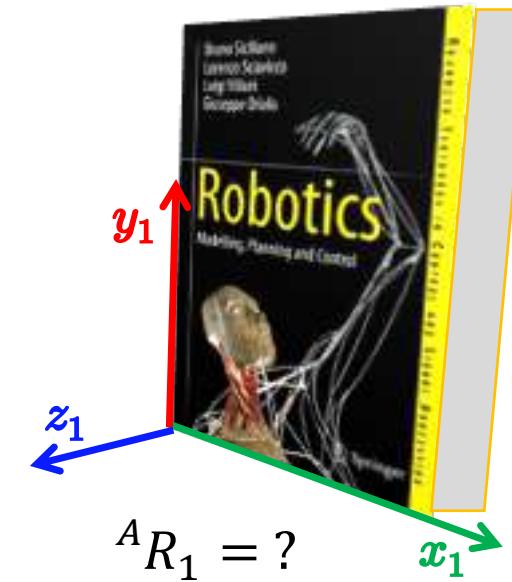
Robotics 1

$${}^A R_B = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$${}^A R_A = {}^A R_B {}^B R_A = I$$

$${}^B R_C = \begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} = {}^B R_A {}^A R_C = {}^A R_B^T {}^A R_C$$

$${}^A R_C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$





Rotation matrix

$${}^A R_B = \begin{bmatrix} \mathbf{x}_A^T \mathbf{x}_B & \mathbf{x}_A^T \mathbf{y}_B & \mathbf{x}_A^T \mathbf{z}_B \\ \mathbf{y}_A^T \mathbf{x}_B & \mathbf{y}_A^T \mathbf{y}_B & \mathbf{y}_A^T \mathbf{z}_B \\ \mathbf{z}_A^T \mathbf{x}_B & \mathbf{z}_A^T \mathbf{y}_B & \mathbf{z}_A^T \mathbf{z}_B \end{bmatrix}$$

orthonormal,
with $\det = +1$

chain rule property

$${}^k R_i {}^i R_j = {}^k R_j$$

orientation of RF_i
w.r.t. RF_k

orientation of RF_j
w.r.t. RF_i

orientation of RF_j
w.r.t. RF_k

direction cosine of
 \mathbf{z}_B w.r.t. \mathbf{x}_A

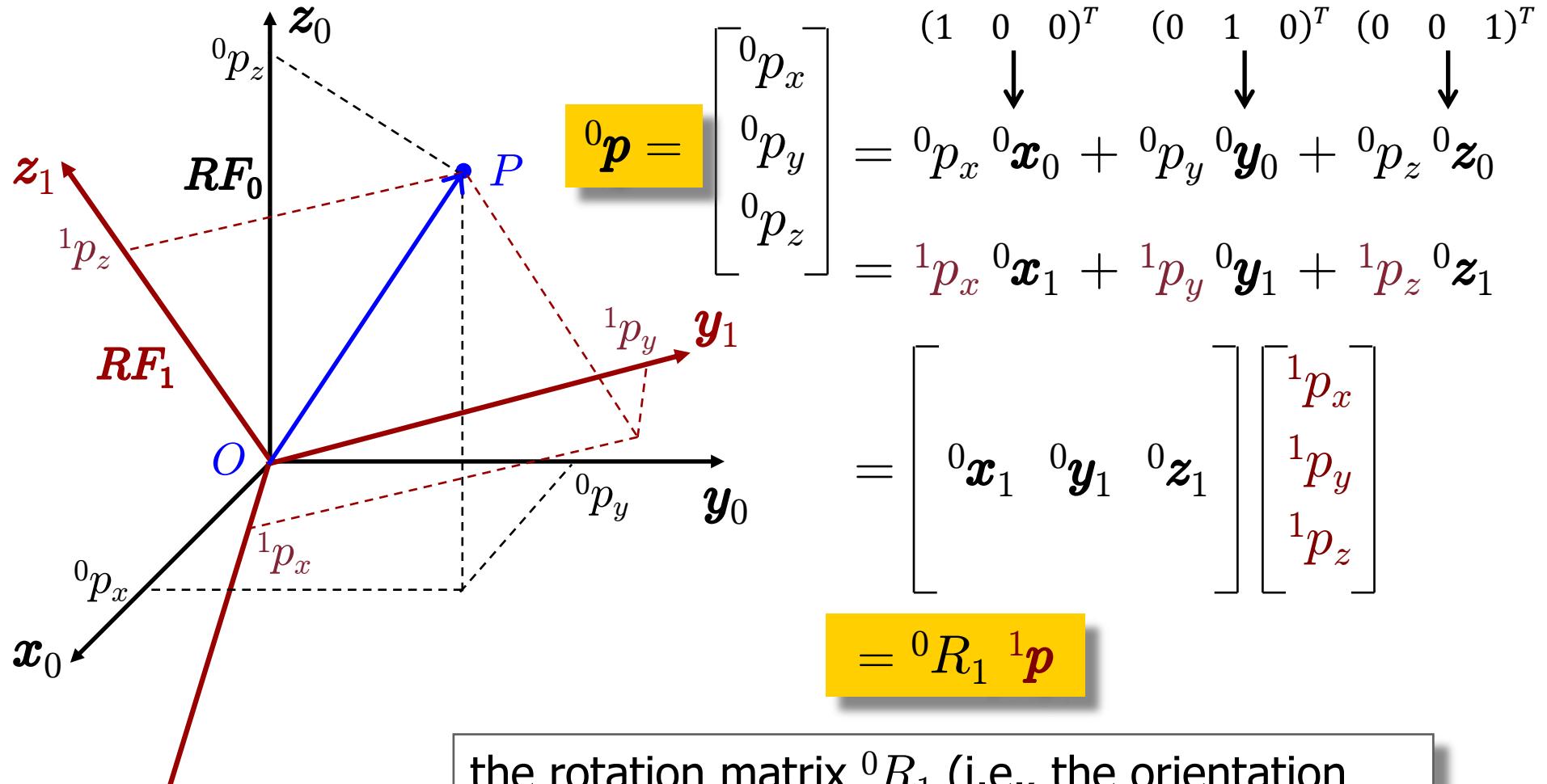
$\mathbf{x}_A^T \mathbf{z}_B = \|\mathbf{x}_A\| \|\mathbf{z}_B\| \cos \beta$
 $= \cos \beta$

algebraic structure
of a group $SO(3)$:
neutral element = I ,
inverse element = R^T

NOTE: in general, the product of rotation matrices does **not** commute!



Change of coordinates



the rotation matrix 0R_1 (i.e., the orientation of RF_1 w.r.t. RF_0) represents **also** the change of coordinates of a **vector** from RF_1 to RF_0



Change of coordinates

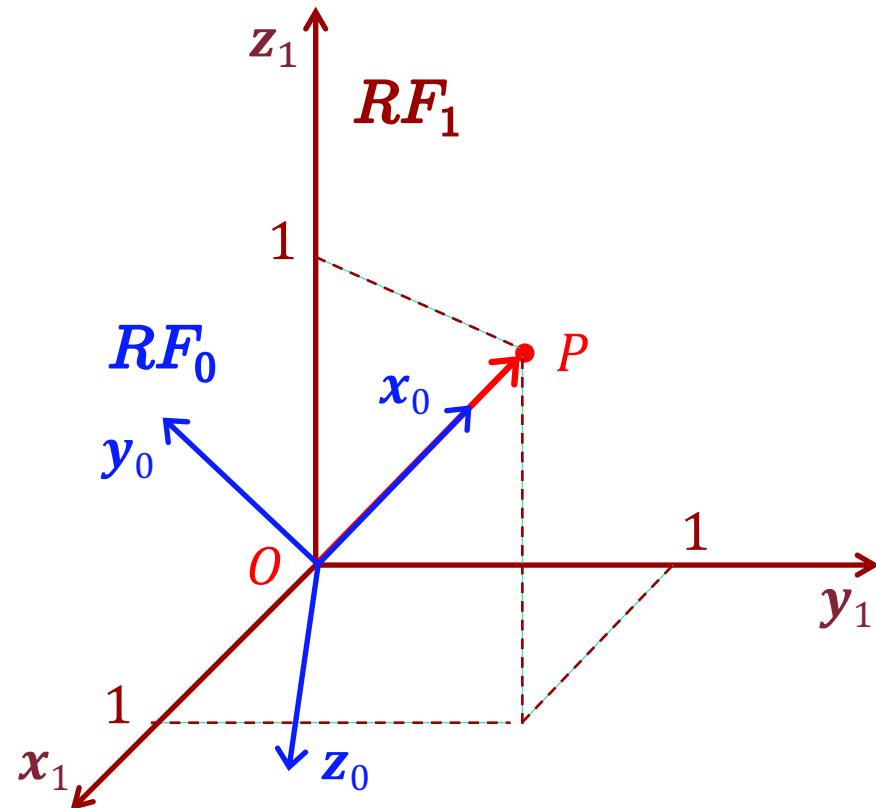
$${}^1\mathbf{p} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$${}^0R_1 = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ 1/\sqrt{6} & -2/\sqrt{6} & 1/\sqrt{6} \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \end{pmatrix}$$

$${}^0\mathbf{p} = {}^0R_1 {}^1\mathbf{p} = \begin{pmatrix} \sqrt{3} \\ 0 \\ 0 \end{pmatrix}$$

$$\|\mathbf{p}\| = \|{}^0\mathbf{p}\| = \|{}^1\mathbf{p}\| = \sqrt{3}$$

... and where is \mathbf{RF}_0 ?

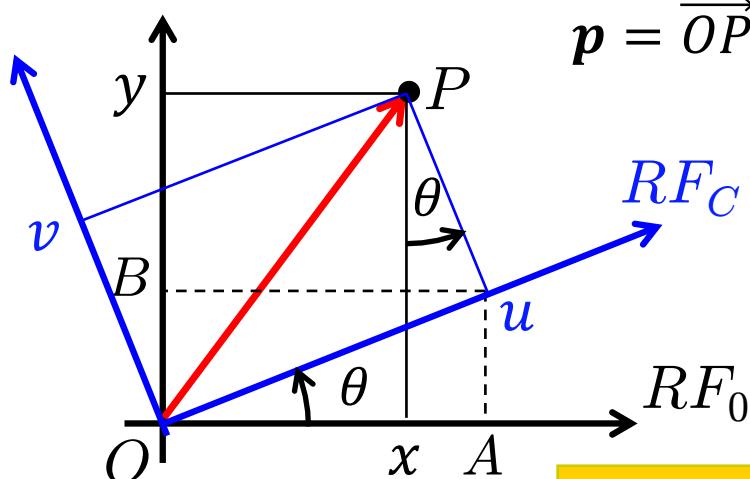


- \mathbf{x}_0 is aligned with $\mathbf{p} = \overrightarrow{OP}$
- \mathbf{z}_0 is orthogonal to \mathbf{y}_1 ($\mathbf{z}_0^T \mathbf{y}_1 = 0$) and is positive on \mathbf{x}_1 ($\mathbf{z}_0^T \mathbf{x}_1 = 1/\sqrt{2}$)
- \mathbf{y}_0 completes a right-handed frame



Orientation of frames in a plane

(elementary rotation around z -axis)



$$x = OA - xA = u \cos \theta - v \sin \theta$$

$$y = OB + By = u \sin \theta + v \cos \theta$$

$$z = w$$

or...

$${}^0\mathbf{p} \rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = R_z(\theta) \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

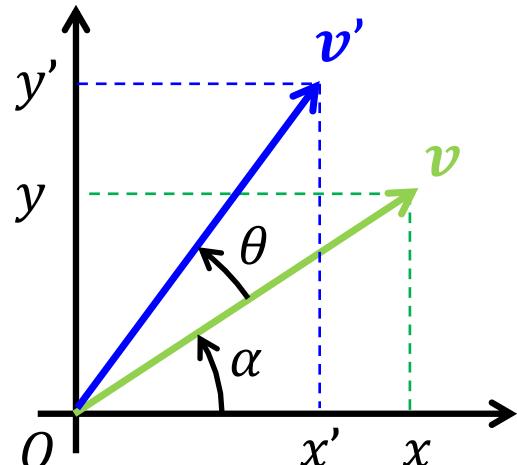
similarly:

$$R_z(-\theta) = R_z^T(\theta)$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$



Rotation of a vector around z



$$x = \|\boldsymbol{v}\| \cos \alpha$$

$$y = \|\boldsymbol{v}\| \sin \alpha$$

$$\begin{aligned} x' &= \|\boldsymbol{v}\| \cos (\alpha + \theta) = \|\boldsymbol{v}\|(\cos \alpha \cos \theta - \sin \alpha \sin \theta) \\ &= x \cos \theta - y \sin \theta \\ y' &= \|\boldsymbol{v}\| \sin (\alpha + \theta) = \|\boldsymbol{v}\|(\sin \alpha \cos \theta + \cos \alpha \sin \theta) \\ &= x \sin \theta + y \cos \theta \end{aligned}$$

$$z' = z$$

or...

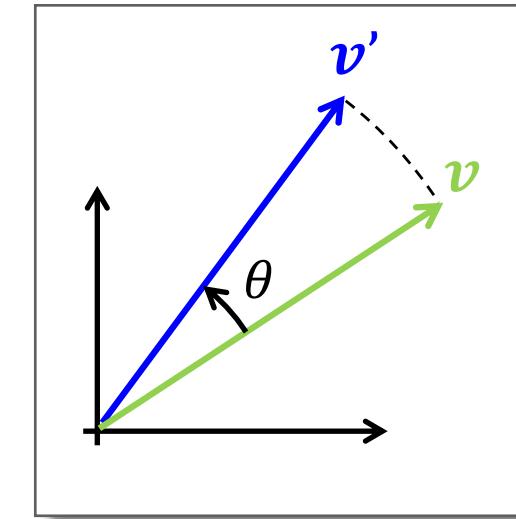
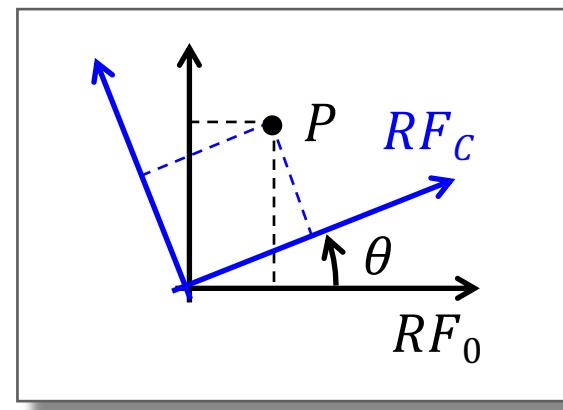
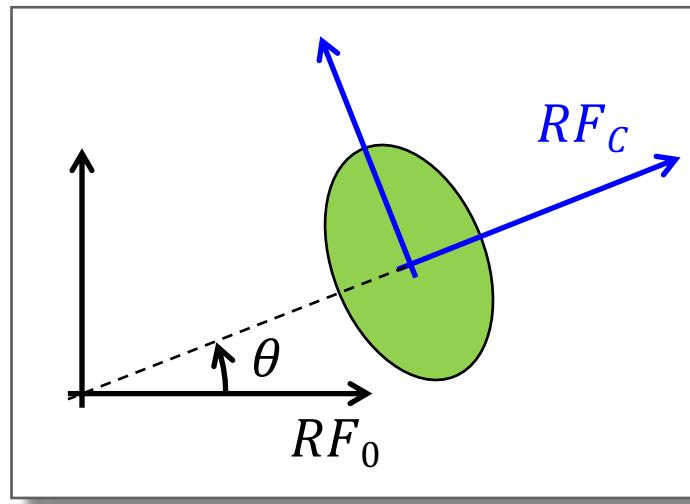
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_z(\theta) \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

... same as before!



Equivalent interpretations of a rotation matrix

the **same** rotation matrix (e.g., $R_z(\theta)$) may represent



the orientation of a rigid body with respect to a reference frame RF_0
e.g., $[{}^0\mathbf{x}_c \ {}^0\mathbf{y}_c \ {}^0\mathbf{z}_c] = R_z(\theta)$

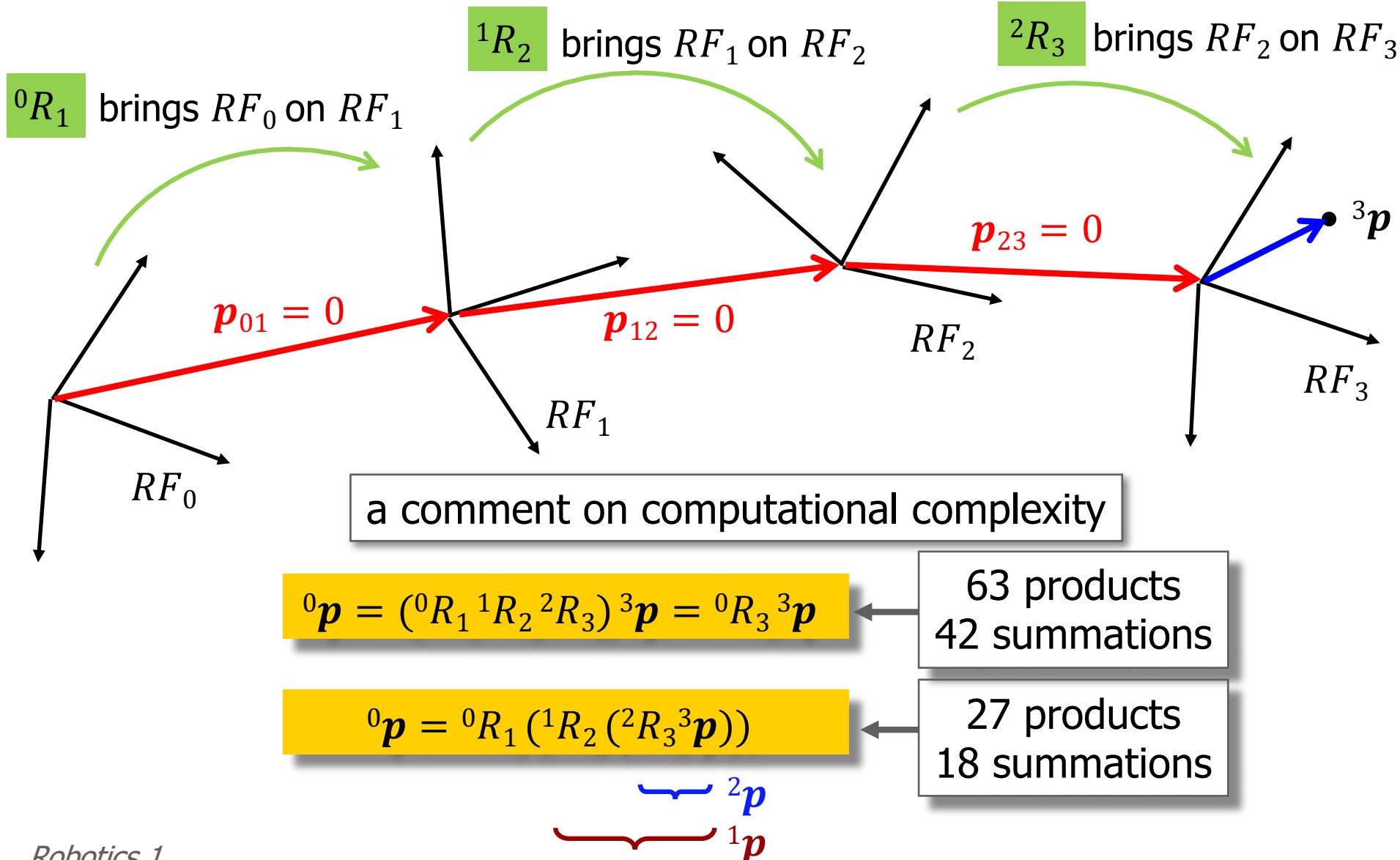
the change of coordinates from RF_C to RF_0
e.g., ${}^0\mathbf{p} = R_z(\theta) {}^C\mathbf{p}$

the rotation operator on vectors
e.g., $\mathbf{v}' = R_z(\theta) \mathbf{v}$

the rotation matrix 0R_C is an operator superposing frame RF_0 to frame RF_C

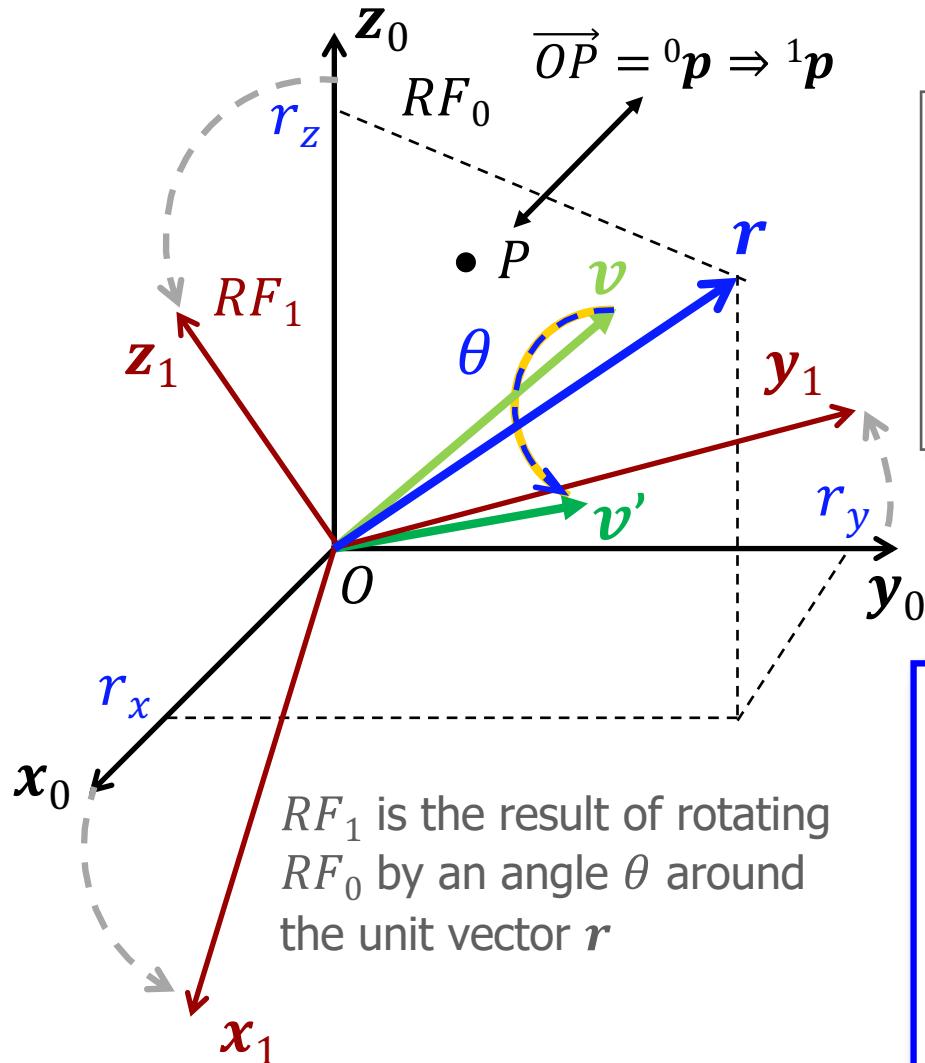


Composition of rotations





Axis/angle representation



DATA

- axis \mathbf{r} (unit vector in \mathbb{R}^3 , $\|\mathbf{r}\| = 1$)
- angle θ , positive **counterclockwise** (as seen from an “observer” oriented like \mathbf{r} with the **head placed on the arrow, looking down to her/his feet**)

DIRECT PROBLEM

parametrized by the given data!

find a rotation matrix $R(\theta, \mathbf{r})$

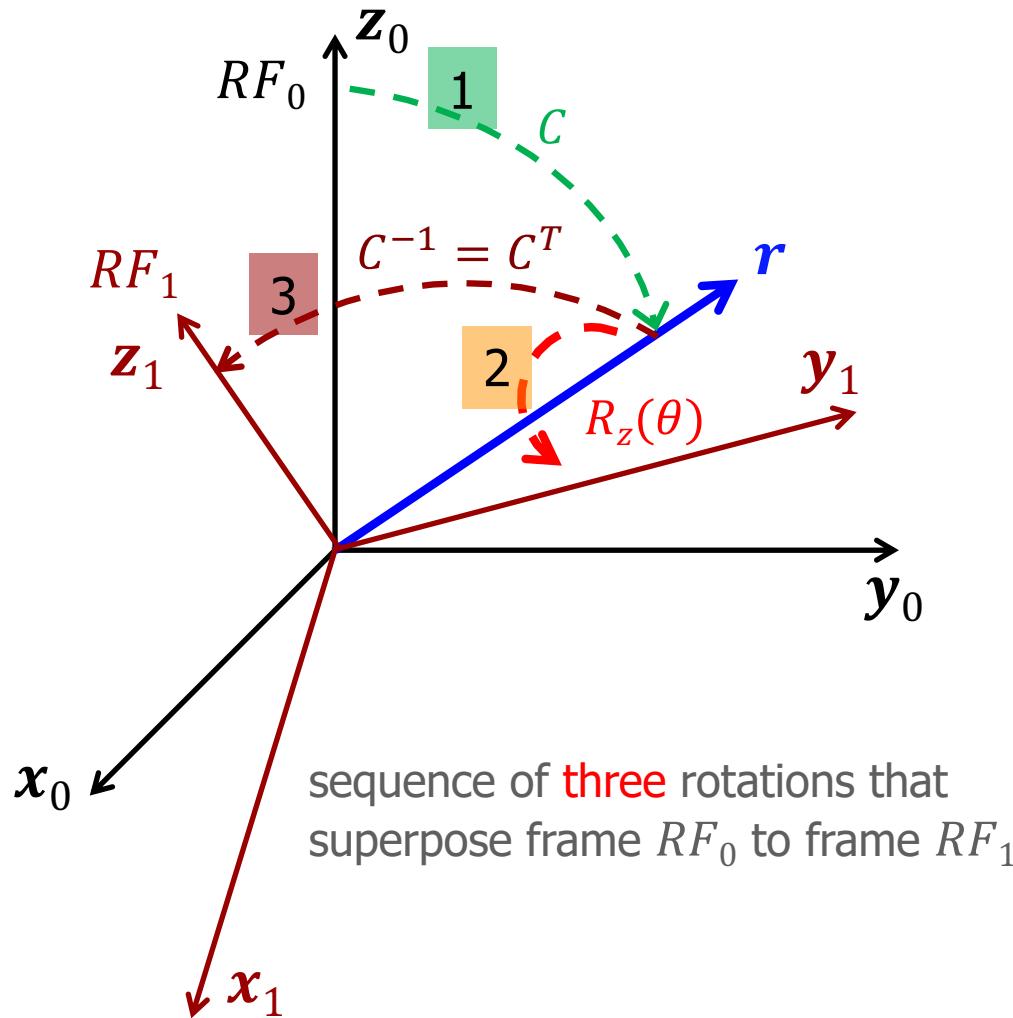
$$R(\theta, \mathbf{r}) = [{}^0\mathbf{x}_1 \ {}^0\mathbf{y}_1 \ {}^0\mathbf{z}_1]$$

such that

$${}^0\mathbf{p} = R(\theta, \mathbf{r}) {}^1\mathbf{p} \quad {}^0\mathbf{v}' = R(\theta, \mathbf{r}) {}^0\mathbf{v}$$



Axis/angle: Direct problem



$$R(\theta, \mathbf{r}) = C R_z(\theta) C^T$$

sequence of three rotations
(one of which is elementary)

$$C = \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{r} \end{bmatrix}$$

after the first rotation
the z-axis coincides with \mathbf{r}

\mathbf{n} and \mathbf{s} are orthogonal unit vectors such that
 $\mathbf{n} \times \mathbf{s} = \mathbf{r}$



Skew-symmetric matrices

whiteboard...

- properties of a **skew-symmetric matrix**
 - a square matrix S is skew-symmetric iff $S^T = -S$
 $\Leftrightarrow s_{ij} = -s_{ji} \Rightarrow s_{ii} = 0$ (zeros on the diagonal)
 - any square matrix A can be decomposed into its symmetric and skew-symmetric parts

$$A = \frac{A+A^T}{2} + \frac{A-A^T}{2} = A_{symm} + A_{skew}$$

- in quadratic forms the skew-symmetric part vanishes (only the symmetric part matters)

$$x^T A x = \frac{1}{2}[x^T A x + (x^T A x)^T] = \frac{1}{2}[x^T A x + x^T A^T x] = x^T \frac{A + A^T}{2} x = x^T A_{symm} x$$

- canonical form of a **3×3 skew-symmetric matrix** also called **vee map** v

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \implies S(\mathbf{v}) = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \quad S = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \stackrel{\mathbf{v} = S\mathbf{v}}{\implies} \mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

- expression of the **vector product** between two vectors $\in \mathbb{R}^3$

$$\mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}, \mathbf{s} = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} \implies \mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \mathbf{n} \times \mathbf{s} = \begin{bmatrix} n_y s_z - s_y n_z \\ n_z s_x - s_z n_x \\ n_x s_y - s_x n_y \end{bmatrix} = S(\mathbf{n}) \mathbf{s}$$

Sarrus rule for
determinant of $\begin{bmatrix} n_x & n_y & n_z \\ s_x & s_y & s_z \\ \vec{i} & \vec{j} & \vec{k} \end{bmatrix}$

$$\mathbf{v}_1 \times \mathbf{v}_2 = S(\mathbf{v}_1) \mathbf{v}_2 = -\mathbf{v}_2 \times \mathbf{v}_1 = -S(\mathbf{v}_2) \mathbf{v}_1 = S^T(\mathbf{v}_2) \mathbf{v}_1$$



Inner and outer products

whiteboard...

- (inner) **row by column** products between two 3×3 matrices

$$C^T C = \begin{bmatrix} \mathbf{n}^T \\ \mathbf{s}^T \\ \mathbf{r}^T \end{bmatrix} \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$$

- **dyadic expansion** of a $n \times n$ matrix

$$\mathbf{e}_i = [0 \quad \dots \quad 1 \quad \dots \quad 0]^T, \quad i = 1, \dots, n \quad \Rightarrow \quad A = \sum_{i,j=1}^n a_{ij} \mathbf{e}_i \mathbf{e}_j^T$$

- **product** of three $n \times n$ matrices **using dyadic form**

$$B = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_1 \dots & \mathbf{b}_{n-1} & \mathbf{b}_n \end{bmatrix} \quad \Rightarrow \quad B A B^T = \sum_{i,j=1}^n a_{ij} \mathbf{b}_i \mathbf{b}_j^T$$

- (outer) **column by row** products between two 3×3 matrices

$$\begin{aligned} CC^T = I \Rightarrow CC^T &= \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{r} \end{bmatrix} \begin{bmatrix} \mathbf{n}^T \\ \mathbf{s}^T \\ \mathbf{r}^T \end{bmatrix} = \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{r} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{n}^T \\ \mathbf{s}^T \\ \mathbf{r}^T \end{bmatrix} \\ &= \mathbf{n}\mathbf{n}^T + \mathbf{s}\mathbf{s}^T + \mathbf{r}\mathbf{r}^T = I \end{aligned}$$



Axis/angle: Direct problem solution

$$R(\theta, \mathbf{r}) = C R_z(\theta) C^T$$

$$\begin{aligned} R(\theta, \mathbf{r}) &= [\mathbf{n} \quad \mathbf{s} \quad \mathbf{r}] \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{n}^T \\ \mathbf{s}^T \\ \mathbf{r}^T \end{bmatrix} \\ &= \mathbf{r}\mathbf{r}^T + (\mathbf{n}\mathbf{n}^T + \mathbf{s}\mathbf{s}^T)^T c\theta + (\mathbf{s}\mathbf{n}^T - \mathbf{n}\mathbf{s}^T) s\theta \end{aligned}$$

taking into account

$$CC^T = \mathbf{n}\mathbf{n}^T + \mathbf{s}\mathbf{s}^T + \mathbf{r}\mathbf{r}^T = I$$

$$\mathbf{s}\mathbf{n}^T - \mathbf{n}\mathbf{s}^T = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} = S(\mathbf{r})$$

depends only
on \mathbf{r} and θ !

$$\rightarrow R(\theta, \mathbf{r}) = \mathbf{r}\mathbf{r}^T + (I - \mathbf{r}\mathbf{r}^T) c\theta + S(\mathbf{r}) s\theta$$



Final expression of $R(\theta, \mathbf{r})$

developing computations...

$$R(\theta, \mathbf{r}) =$$

$$\begin{bmatrix} r_x^2(1 - \cos \theta) + \cos \theta & r_x r_y (1 - \cos \theta) - r_z \sin \theta & r_x r_z (1 - \cos \theta) + r_y \sin \theta \\ r_x r_y (1 - \cos \theta) + r_z \sin \theta & r_y^2(1 - \cos \theta) + \cos \theta & r_y r_z (1 - \cos \theta) - r_x \sin \theta \\ r_x r_z (1 - \cos \theta) - r_y \sin \theta & r_y r_z (1 - \cos \theta) + r_x \sin \theta & r_z^2(1 - \cos \theta) + \cos \theta \end{bmatrix}$$

note that

$$\text{trace } R(\theta, \mathbf{r}) = 1 + 2 \cos \theta$$

$$R(\theta, \mathbf{r}) = R(-\theta, -\mathbf{r}) = R^T(-\theta, \mathbf{r})$$



Axis/angle: a simple example

$$R(\theta, \mathbf{r}) = \mathbf{r}\mathbf{r}^T + (I - \mathbf{r}\mathbf{r}^T) c\theta + S(\mathbf{r}) s\theta$$

$$\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{z}_0$$

$$\begin{aligned} R(\theta, \mathbf{r}) &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} c\theta + \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} s\theta \\ &= \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_z(\theta) \end{aligned}$$



Axis/angle: Rodriguez formula

$$\mathbf{v}' = R(\theta, \mathbf{r})\mathbf{v}$$

$$\mathbf{v}' = \mathbf{v} \cos \theta + (\mathbf{r} \times \mathbf{v}) \sin \theta + (1 - \cos \theta)(\mathbf{r}^T \mathbf{v}) \mathbf{r}$$

proof

$$\begin{aligned} R(\theta, \mathbf{r})\mathbf{v} &= (\mathbf{r} \mathbf{r}^T + (I - \mathbf{r} \mathbf{r}^T) \cos \theta + S(\mathbf{r}) \sin \theta) \mathbf{v} \\ &= \mathbf{r} \mathbf{r}^T \mathbf{v} (1 - \cos \theta) + \mathbf{v} \cos \theta + (\mathbf{r} \times \mathbf{v}) \sin \theta \end{aligned}$$

q.e.d.



Properties of $R(\theta, \mathbf{r})$

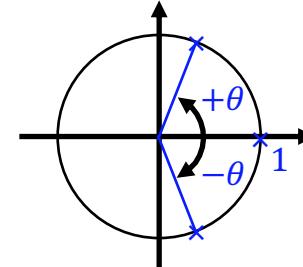
1. $R(\theta, \mathbf{r})\mathbf{r} = \mathbf{r}$ (\mathbf{r} is the **invariant axis** in this rotation)
 2. when \mathbf{r} is one of the coordinate axes, R boils down to one of the known elementary rotation matrices
 3. $(\theta, \mathbf{r}) \rightarrow R$ is **not** an **injective** map: $R(\theta, \mathbf{r}) = R(-\theta, -\mathbf{r})$
 4. $\det(R) = +1 = \prod_i \lambda_i$ (eigenvalues)
 5. $\text{tr}(R) = \text{tr}(\mathbf{r}\mathbf{r}^T) + (I - \mathbf{r}\mathbf{r}^T)c\theta = 1 + 2c\theta = \sum_i \lambda_i$
- identities
in green
hold for
any matrix

$$1. \Rightarrow \lambda_1 = 1$$

$$4. \& 5. \Rightarrow \lambda_2 + \lambda_3 = 2 c\theta \Rightarrow \lambda^2 - 2 c\theta \lambda + 1 = 0$$

$$\Rightarrow \lambda_{2,3} = c\theta \pm \sqrt{c^2\theta^2 - 1} = c\theta \pm i s\theta = e^{\pm i\theta}$$

all eigenvalues λ have unitary module ($\Leftarrow R$ orthonormal)





Axis/angle: Inverse problem

GIVEN a rotation matrix R ,
FIND a unit vector r and an angle θ such that

$$R = rr^T + (I - rr^T) \cos \theta + S(r) \sin \theta = R(\theta, r)$$

note first that $\text{tr}(R) = R_{11} + R_{22} + R_{33} = 1 + 2 \cos \theta$; so, one could solve

$$\theta = \arccos \frac{R_{11} + R_{22} + R_{33} - 1}{2}$$

but

- this formula provides only values in $[0, \pi]$ (thus, never negative angles θ)
- loss of numerical accuracy for $\theta \rightarrow 0$ (sensitivity of $\cos \theta$ is low around 0)



Axis/angle: Inverse problem solution

from the **data**



from $R(\theta, \mathbf{r})$

$$\mathbf{R} - \mathbf{R}^T = \begin{bmatrix} 0 & R_{12} - R_{21} & R_{13} - R_{31} \\ R_{21} - R_{12} & 0 & R_{23} - R_{32} \\ R_{31} - R_{13} & R_{32} - R_{23} & 0 \end{bmatrix} = 2 \sin \theta \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

it follows

$$\|\mathbf{r}\| = 1 \Rightarrow \sin \theta = \pm \frac{1}{2} \sqrt{(R_{12} - R_{21})^2 + (R_{13} - R_{31})^2 + (R_{23} - R_{32})^2} \quad (*)$$

thus

(**)

$$\theta = \text{atan2} \left\{ \pm \sqrt{(R_{12} - R_{21})^2 + (R_{13} - R_{31})^2 + (R_{23} - R_{32})^2}, R_{11} + R_{22} + R_{33} - 1 \right\}$$

see next slide

$$\mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \frac{1}{2 \sin \theta} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

can be used **only** if

$$\sin \theta \neq 0$$

test is made on (*)
using the data $\{R_{ij}\}$



atan2 function

- arctangent with output values “in the four quadrants”
 - two input arguments
 - takes values in $[-\pi, +\pi]$
 - undefined only for $(0, 0)$
- uses the sign of both arguments to define the output quadrant
- based on **arctan** function with output values in $[-\pi/2, +\pi/2]$
- available in main languages (C++, Matlab, ...)

$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \pi + \arctan\left(\frac{y}{x}\right) & y \geq 0, x < 0 \\ -\pi + \arctan\left(\frac{y}{x}\right) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$



Singular cases

(use when $\sin \theta = 0$)

- if $\theta = 0$ from (**), there is no solution for \mathbf{r} (rotation axis undefined)
- if $\theta = \pm\pi$ from (**), then set $\sin \theta = 0, \cos \theta = -1$ and solve

$$\Rightarrow \mathbf{R} = 2\mathbf{r}\mathbf{r}^T - I$$

$$\mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} \pm \sqrt{(R_{11} + 1)/2} \\ \pm \sqrt{(R_{22} + 1)/2} \\ \pm \sqrt{(R_{33} + 1)/2} \end{bmatrix}$$

$$\text{with } \begin{aligned} r_x r_y &= R_{12}/2 \\ r_x r_z &= R_{13}/2 \\ r_y r_z &= R_{23}/2 \end{aligned}$$

used to resolve sign ambiguities
 \Leftrightarrow two solutions of opposite sign

homework: write a code that determines the two solutions (θ, \mathbf{r})

$$\text{for } \mathbf{R} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$



Unit quaternion

- to eliminate non-uniqueness and singular cases of the axis/angle (θ, \mathbf{r}) representation, the **unit quaternion** can be used

$$Q = \{\eta, \boldsymbol{\epsilon}\} = \{\cos(\theta/2), \sin(\theta/2) \mathbf{r}\}$$

a scalar 3-dim vector

- $\eta^2 + \|\boldsymbol{\epsilon}\|^2 = 1$ (thus, “unit ...”)
- (θ, \mathbf{r}) and $(-\theta, -\mathbf{r})$ are associated to the **same** quaternion Q
- the **rotation** matrix R associated to a given quaternion Q is

$$R(\eta, \boldsymbol{\epsilon}) = \begin{bmatrix} 2(\eta^2 + \epsilon_x^2) - 1 & 2(\epsilon_x \epsilon_y - \eta \epsilon_z) & 2(\epsilon_x \epsilon_z + \eta \epsilon_y) \\ 2(\epsilon_x \epsilon_y + \eta \epsilon_z) & 2(\eta^2 + \epsilon_y^2) - 1 & 2(\epsilon_y \epsilon_z - \eta \epsilon_x) \\ 2(\epsilon_x \epsilon_z - \eta \epsilon_y) & 2(\epsilon_y \epsilon_z + \eta \epsilon_x) & 2(\eta^2 + \epsilon_z^2) - 1 \end{bmatrix}$$

- no** rotation is $Q = \{1, \mathbf{0}\}$, while the **inverse** rotation is $Q = \{\eta, -\boldsymbol{\epsilon}\}$
- unit quaternions are **composed** with special rules

$$Q_1 * Q_2 = \{\eta_1 \eta_2 - \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2, \eta_1 \boldsymbol{\epsilon}_2 + \eta_2 \boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_1 \times \boldsymbol{\epsilon}_2\}$$



Robotics 1

Minimal representations of orientation (Euler and roll-pitch-yaw angles) Homogeneous transformations

Prof. Alessandro De Luca

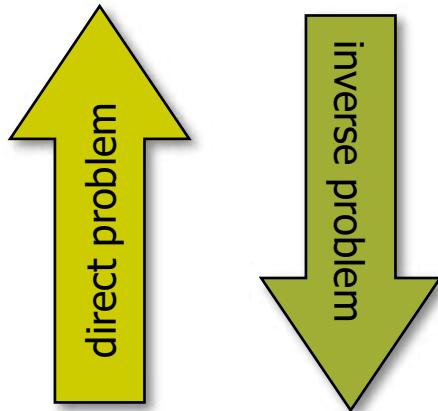
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





“Minimal” representations

- rotation matrices:



9 elements

$$\begin{aligned} & - 3 \text{ orthogonality relationships} \\ & - 3 \text{ unitary relationships} \\ = & \underline{\quad\quad\quad} \\ = & \text{3 independent variables} \end{aligned}$$

- sequence of **3 rotations** w.r.t. independent axes

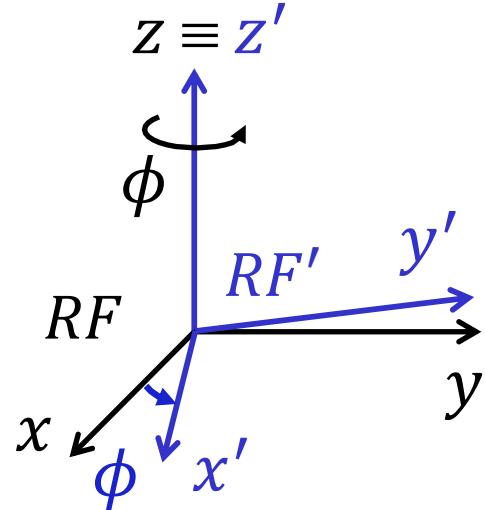
- by angles $\alpha_i, i = 1, 2, 3$, around **fixed** (a_i) or **moving/current** (a'_i) axes
 - generically called **Roll-Pitch-Yaw** (fixed axes) or **Euler** (moving axes) angles
- $12 + 12$ possible different sequences (e.g., XYX)
 - **without** contiguous repetitions of axes (e.g., no XXZ nor YZ'Z')
- actually, only 12 sequences are different since we shall see that

$$\{(a_1, \alpha_1), (a_2, \alpha_2), (a_3, \alpha_3)\} \equiv \{(a'_3, \alpha_3), (a'_2, \alpha_2), (a'_1, \alpha_1)\}$$

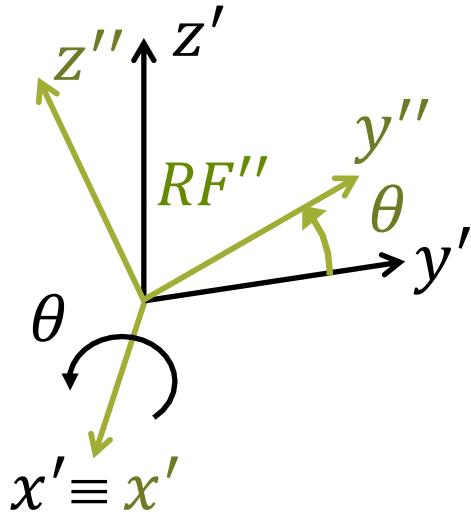


ZX'Z'' Euler angles

1



2

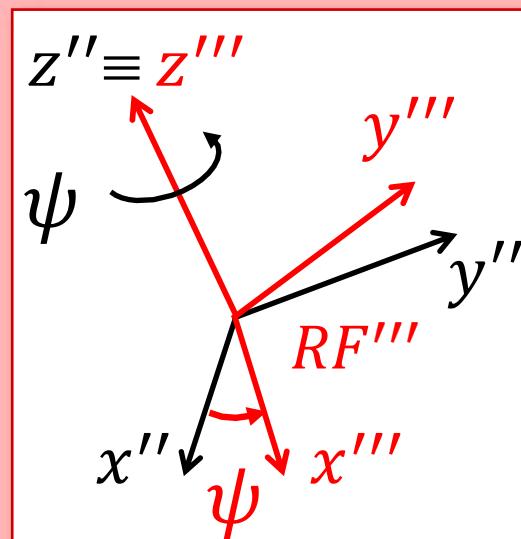


$$R_{x'}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

$$R_z(\phi) = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3

$$R_{z''}(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$





$ZX'Z''$ Euler angles

- direct problem: given ϕ, θ, ψ , find R

$$R_{ZX'Z''}(\phi, \theta, \psi) = R_Z(\phi)R_{X'}(\theta)R_{Z''}(\psi)$$

order of definition
in concatenation $\xrightarrow{ }$

$$= \begin{bmatrix} c\phi c\psi - s\phi c\theta s\psi & -c\phi s\psi - s\phi c\theta c\psi & s\phi s\theta \\ s\phi c\psi + c\phi c\theta s\psi & -s\phi s\psi + c\phi c\theta c\psi & -c\phi s\theta \\ s\theta s\psi & s\theta c\psi & c\theta \end{bmatrix}$$

- given a vector $v''' = (x''', y''', z''')$ expressed in RF''' , its expression in the coordinates of RF is

$$v = R_{ZX'Z''}(\phi, \theta, \psi)v'''$$

- the orientation of RF''' is the **same** that would be obtained with the sequence of rotations

ψ around z , θ around x (fixed), ϕ around z (fixed)



ZX'Z'' Euler angles

- inverse problem: given $R = \{r_{ij}\}$, find ϕ, θ, ψ

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\phi c\psi - s\phi c\theta s\psi & -c\phi s\psi - s\phi c\theta c\psi & s\phi s\theta \\ s\phi c\psi + c\phi c\theta s\psi & -s\phi s\psi + c\phi c\theta c\psi & -c\phi s\theta \\ s\theta s\psi & s\theta c\psi & c\theta \end{bmatrix}$$

- $r_{13}^2 + r_{23}^2 = s^2\theta, r_{33} = c\theta \Rightarrow$

$$\theta = \text{atan2} \left\{ \pm \sqrt{r_{13}^2 + r_{23}^2}, r_{33} \right\}$$

two values differing just for the sign

- if $r_{13}^2 + r_{23}^2 \neq 0$ (i.e., $s\theta \neq 0$)

$$r_{31}/s\theta = s\psi, r_{32}/s\theta = c\psi \Rightarrow$$

$$\psi = \text{atan2}\{r_{31}/s\theta, r_{32}/s\theta\}$$

- similarly...

$$\phi = \text{atan2}\{r_{13}/s\theta, -r_{23}/s\theta\}$$

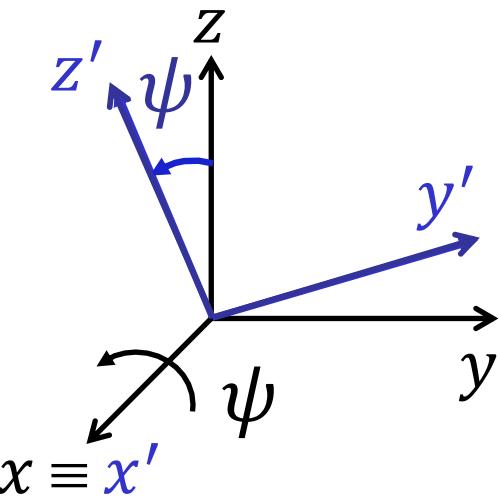
- there is always a **pair** of solutions in the regular case
- there are always **singularities** (here $\theta = 0$ or $\pm\pi$) \Rightarrow only the **sum** $\phi + \psi$ or the **difference** $\phi - \psi$ can be determined



Roll-Pitch-Yaw angles (fixed XYZ)

1

ROLL

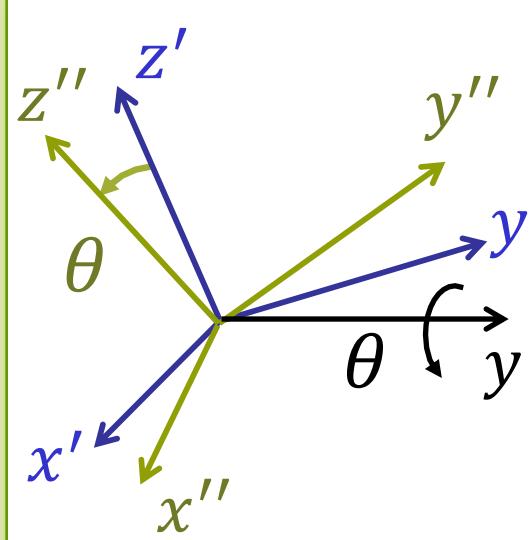


$$R_X(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi \\ 0 & \sin \psi & \cos \psi \end{bmatrix}$$

$C_1 R_Y(\theta) C_1^T$

with $R_Y(\theta) =$

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$



PITCH

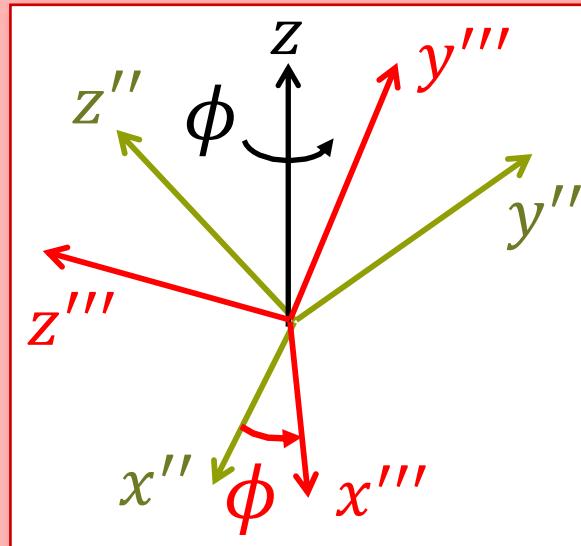
$$C_1 R_Y(\theta) C_1^T$$

with $R_Y(\theta) =$

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

3

YAW





Roll-Pitch-Yaw angles (fixed XYZ)

- direct problem: given ψ, θ, ϕ , find R

$$R_{RPY}(\psi, \theta, \phi) = R_Z(\phi)R_Y(\theta)R_X(\psi) \quad \Leftarrow \text{note the order of products!}$$

order of definition

$$= \begin{bmatrix} c\phi c\theta & c\phi s\theta s\psi - s\phi c\psi & c\phi s\theta c\psi + s\phi s\psi \\ s\phi c\theta & s\phi s\theta s\psi + c\phi c\psi & s\phi s\theta c\psi - c\phi s\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix}$$

- inverse problem: given $R = \{r_{ij}\}$, find ψ, θ, ϕ

- $r_{32}^2 + r_{33}^2 = c^2\theta, r_{31} = -s\theta \Rightarrow$

$$\theta = \text{atan2} \left\{ -r_{31}, \pm \sqrt{r_{32}^2 + r_{33}^2} \right\}$$

- if $r_{32}^2 + r_{33}^2 \neq 0$ (i.e., $c\theta \neq 0$)

for $r_{31} < 0$, two symmetric values w.r.t. $\pi/2$

- $r_{32}/c\theta = s\psi, r_{33}/c\theta = c\psi \Rightarrow$

$$\psi = \text{atan2}\{r_{32}/c\theta, r_{33}/c\theta\}$$

- similarly ...

$$\phi = \text{atan2}\{r_{21}/c\theta, r_{11}/c\theta\}$$

- singularities for $\theta = \pm \pi/2 \Rightarrow$ only $\phi + \psi$ or $\phi - \psi$ are defined



...why this order in the product?

$$R_{RPY}(\psi, \theta, \phi) = R_Z(\phi)R_Y(\theta)R_X(\psi)$$

$\xrightarrow{\text{order of definition}}$ “reverse” order in the product
(pre-multiplication...)

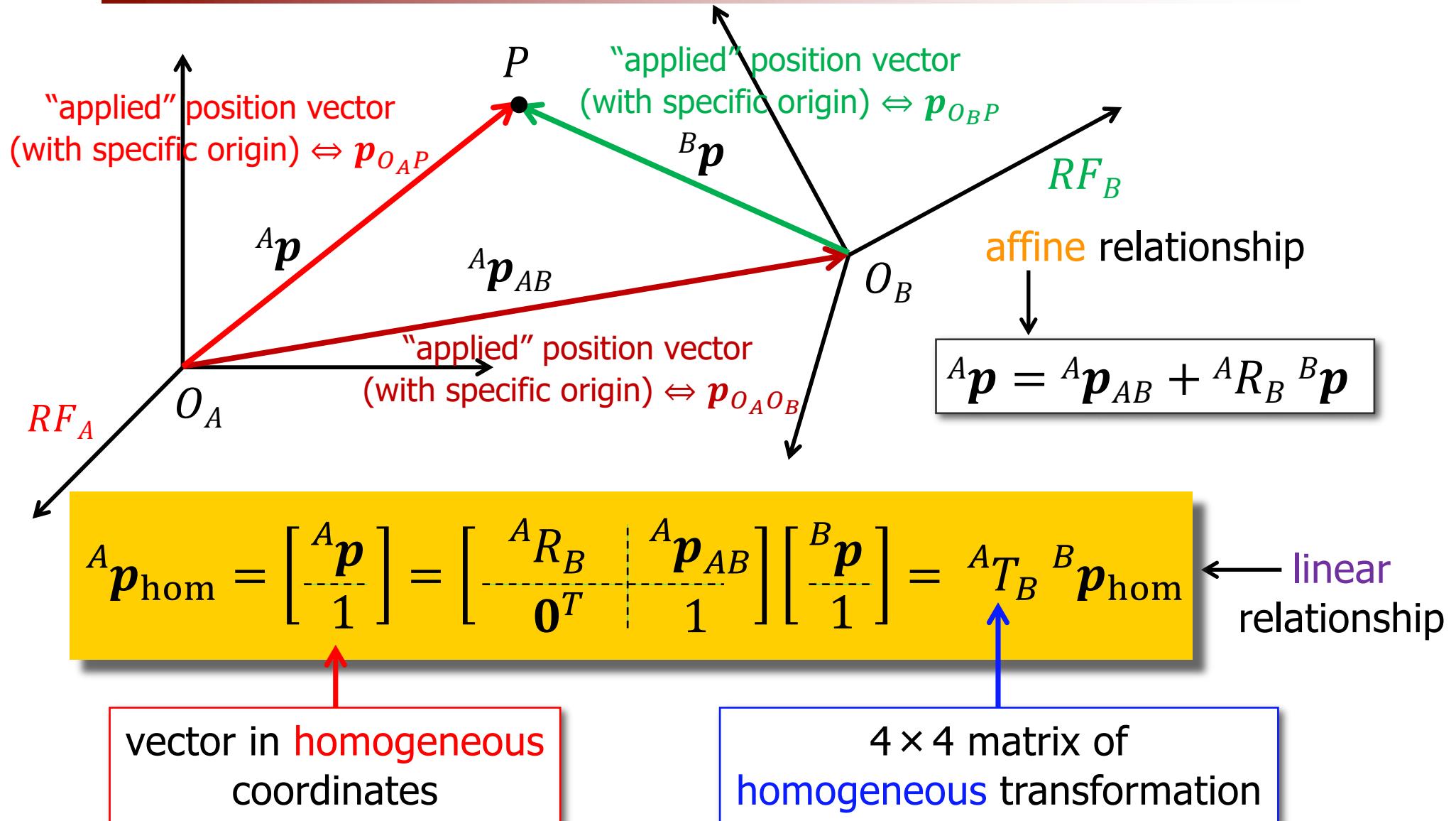
- need to refer each rotation in the sequence to one of the original **fixed** axes
 - use the angle/axis technique for each rotation in the sequence:
 $C R(\alpha) C^T$, with C being the rotation matrix **reverting** the previously made rotations (= “go back” to the original axes)

concatenating three rotations: [] [] [] (post-multiplication...)

$$\begin{aligned} R_{RPY}(\psi, \theta, \phi) &= [R_X(\psi)] [R_X^T(\psi) R_Y(\theta) R_X(\psi)] \\ &\quad [R_X^T(\psi) R_Y^T(\theta) R_Z(\phi) R_Y(\theta) R_X(\psi)] \\ &= R_Z(\phi)R_Y(\theta)R_X(\psi) \end{aligned}$$



Homogeneous transformations





Use of homogeneous transformation T

- describes the relation between two reference frames
(relative **pose** = position & orientation)
- transforms the representation of a **position** vector
(**applied** vector starting from the **origin** of the frame)
from one frame to another frame
- it is a **roto-translation** operator on vectors in the three-dimensional space
- it is always invertible $({}^A T_B)^{-1} = {}^B T_A$
- can be composed, i.e., ${}^A T_B {}^B T_C = {}^A T_C$ \leftarrow note: it does not commute in general!



Affine vs linear computations

whiteboard...

$${}^1 p = {}^1 p_{01} + {}^1 R_0 {}^0 p$$

$${}^2 p = {}^2 p_{12} + {}^2 R_1 {}^1 p = {}^2 p_{12} + {}^2 R_1 {}^1 p_{01} + {}^2 R_1 {}^1 R_0 {}^0 p$$

$${}^3 p = {}^3 p_{23} + {}^3 R_2 {}^2 p = \dots = {}^2 p_{23} + {}^3 R_2 {}^2 p_{12} + {}^3 R_2 {}^2 R_1 {}^1 p_{01} + {}^3 R_2 {}^2 R_1 {}^1 R_0 {}^0 p$$

$${}^4 p = {}^4 p_{34} + {}^4 R_3 {}^3 p = \dots \quad \text{heavy on notation (and not only!)}$$

$${}^1 T_0 = \begin{bmatrix} {}^1 R_0 & {}^1 p_{01} \\ 0^T & 1 \end{bmatrix} \Rightarrow {}^1 p_{hom} = {}^1 T_0 {}^0 p_{hom}$$

$${}^2 T_1 = \begin{bmatrix} {}^2 R_1 & {}^2 p_{12} \\ 0^T & 1 \end{bmatrix} \Rightarrow {}^2 p_{hom} = {}^2 T_1 {}^1 T_0 {}^0 p_{hom} = {}^2 T_0 {}^0 p_{hom}$$

$${}^3 T_2 = \begin{bmatrix} {}^3 R_1 & {}^3 p_{23} \\ 0^T & 1 \end{bmatrix} \Rightarrow {}^3 p_{hom} = {}^3 T_2 {}^2 T_1 {}^1 T_0 {}^0 p_{hom} = {}^3 T_0 {}^0 p_{hom}$$

$${}^4 T_3 = \begin{bmatrix} {}^4 R_3 & {}^4 p_{34} \\ 0^T & 1 \end{bmatrix} \Rightarrow {}^4 p_{hom} = {}^4 T_3 {}^3 T_2 {}^2 T_1 {}^1 T_0 {}^0 p_{hom} = {}^4 T_0 {}^0 p_{hom}$$



Inverse of a homogeneous transformation

$${}^A\mathbf{p} = {}^A\mathbf{p}_{AB} + {}^A R_B {}^B\mathbf{p}$$

exchange $A \leftrightarrow B$

$${}^B\mathbf{p} = {}^B\mathbf{p}_{BA} + {}^B R_A {}^A\mathbf{p} = - {}^A R_B^T {}^A\mathbf{p}_{AB} + {}^A R_B^T {}^A\mathbf{p}$$

... with the original vectors/matrices ...

$$\begin{bmatrix} {}^A R_B & {}^A\mathbf{p}_{AB} \\ \hline 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} {}^B R_A & {}^B\mathbf{p}_{BA} \\ \hline 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^A R_B^T & - {}^A R_B^T {}^A\mathbf{p}_{AB} \\ \hline 0 & 1 \end{bmatrix}$$

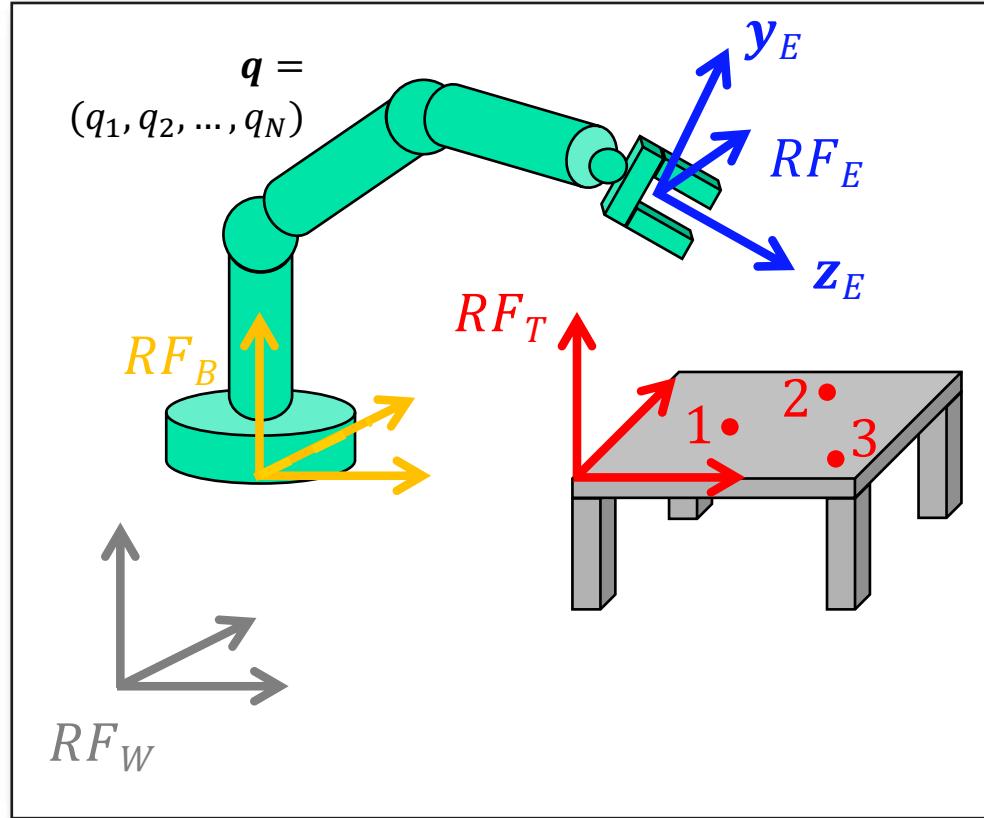
$${}^A T_B$$

$${}^B T_A$$

$$({}^A T_B)^{-1}$$



Defining a robotic task



solve for q
(inverse
kinematics)

absolute definition
of task

task definition relative
to the robot end-effector

$$WT_T = WT_B {}^B T_E {}^E T_T$$

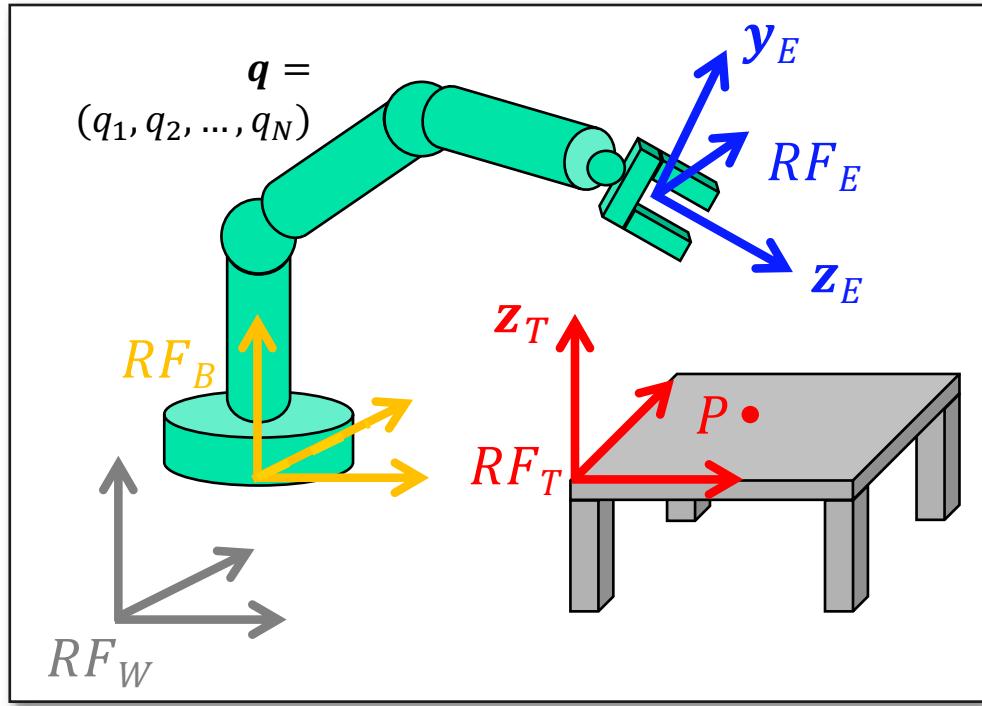
known, once
the robot
is placed

direct kinematics of the
robot arm (function of q)

$${}^B T_E(q) = WT_B^{-1} WT_T {}^E T_T^{-1} = \text{constant}$$



Example of task definition



Q: where is the EE frame w.r.t. the table frame?

$${}^T T_E = \begin{bmatrix} {}^T R_E & {}^T p_{TE} \\ 0^T & 1 \end{bmatrix} = {}^E T_T^{-1}$$

with ${}^T R_E = ({}^E R_T)^T = {}^E R_T$

$${}^T p_{TE} = {}^T p - {}^T R_E {}^E p = \begin{bmatrix} p_x \\ p_y \\ h \end{bmatrix}$$

- the robot carries a **depth camera** (e.g., a Kinect) on the end-effector
- the end-effector should go to a pose above the point P on the table, pointing its approach axis \mathbf{z}_E **downward** and being **aligned** with the table sides

$${}^E R_T = [{}^E \mathbf{x}_T \quad {}^E \mathbf{y}_T \quad {}^E \mathbf{z}_T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

- point P is known in the table frame RF_T
- the robot proceeds by **centering point P** in its camera image until it senses a **depth h** from the table (in RF_E)

$${}^E p = \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix}$$

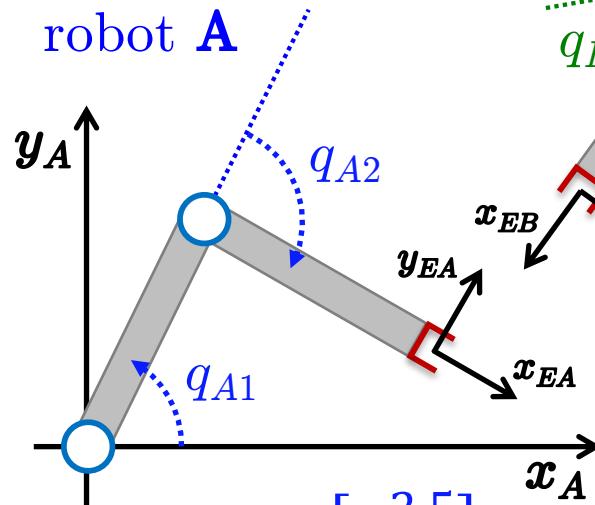


A robotic problem with T matrices

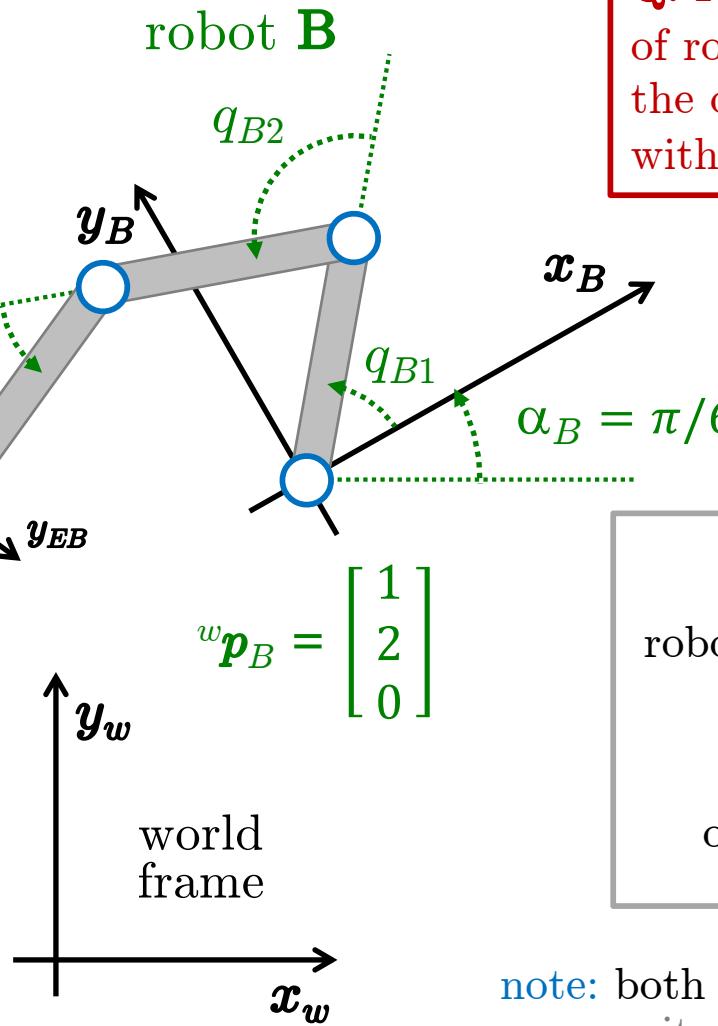
Task: 2R planar robot **A** should hand over an object at a given location to 3R planar robot **B**

configuration of robot **A**
with which the object is being held

$$\mathbf{q}_A = \begin{bmatrix} \pi/3 \\ -\pi/2 \end{bmatrix}$$

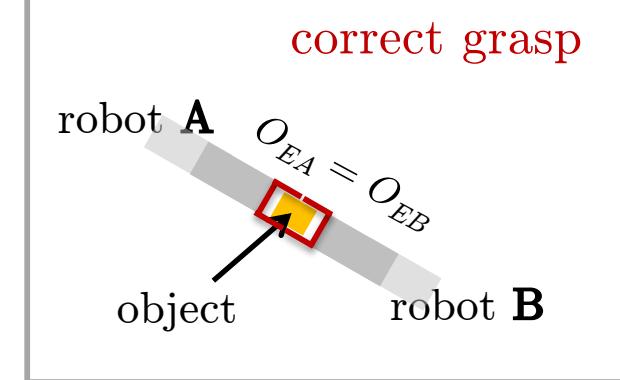


$${}^w\mathbf{p}_A = \begin{bmatrix} -2.5 \\ 1 \\ 0 \end{bmatrix}$$



Q: Find a configuration \mathbf{q}_B of robot **B** so as to grasp the object held by robot **A** with the right orientation

Ex #3, Robotics 1
exam of Sep 11, 2020



note: both robots have unitary link lengths



Solution procedure

$${}^w\mathbf{T}_A = \begin{pmatrix} {}^w\mathbf{R}_A & {}^w\mathbf{p}_A \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I}_{3 \times 3} & -2.5 \\ \mathbf{0}^T & 1 \end{pmatrix} \quad \text{base frame of robot A w.r.t. world}$$

$${}^w\mathbf{T}_B = \begin{pmatrix} {}^w\mathbf{R}_B & {}^w\mathbf{p}_B \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \cos \alpha_B & -\sin \alpha_B & 0 & 1 \\ \sin \alpha_B & \cos \alpha_B & 0 & 2 \\ 0 & 0 & 1 & 0 \\ \mathbf{0}^T & & 1 \end{pmatrix} = \begin{pmatrix} 0.8660 & -0.5 & 0 & 1 \\ 0.5 & 0.8660 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ \mathbf{0}^T & & 1 \end{pmatrix} \quad \text{base frame of robot B w.r.t. world}$$

$$\begin{aligned} {}^A\mathbf{T}_{EA} &= \begin{pmatrix} {}^A\mathbf{R}_{EA} & {}^A\mathbf{p}_{EA} \\ \mathbf{0}^T & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos(q_{A1} + q_{A2}) & -\sin(q_{A1} + q_{A2}) & 0 & \cos q_{A1} + \cos(q_{A1} + q_{A2}) \\ \sin(q_{A1} + q_{A2}) & \cos(q_{A1} + q_{A2}) & 0 & \sin q_{A1} + \sin(q_{A1} + q_{A2}) \\ 0 & 0 & 1 & 0 \\ \mathbf{0}^T & & & 1 \end{pmatrix} \quad \text{end-effector frame of robot A w.r.t. its base frame (uses } \mathbf{q}_A \text{)} \\ &= \begin{pmatrix} 0.8660 & 0.5 & 0 & 1.3660 \\ -0.5 & 0.8660 & 0 & 0.3660 \\ 0 & 0 & 1 & 0 \\ \mathbf{0}^T & & 1 \end{pmatrix} \quad \text{= direct kinematics of robot A!} \end{aligned}$$

$${}^{EA}\mathbf{T}_{EB} = \begin{pmatrix} {}^{EA}\mathbf{R}_{EB} & {}^{EA}\mathbf{p}_{EB} \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathbf{0}^T & & 1 \end{pmatrix} \quad \text{end-effector frame of robot B w.r.t. end-effector frame of robot A to realize the right grasp for correct handover}$$



Solution procedure

$${}^w\mathbf{T}_A{}^A\mathbf{T}_{EA}{}^{EA}\mathbf{T}_{EB} = {}^w\mathbf{T}_B{}^B\mathbf{T}_{EB}$$

kinematic equation defining the task

end-effector frame of robot **B**
w.r.t. world passing via the
given configuration of robot **A**

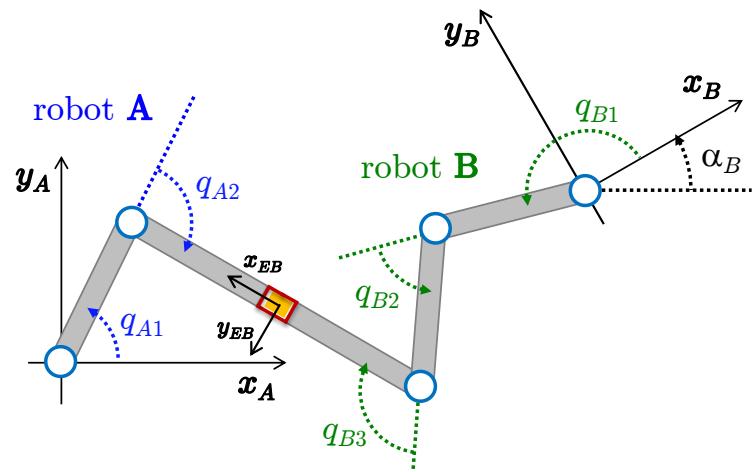
end-effector frame of robot **B**
w.r.t. world passing via its
base frame

$$\begin{aligned} {}^B\mathbf{T}_{EB,d} &= \begin{pmatrix} {}^B\mathbf{R}_{EB,d} & {}^B\mathbf{p}_{EB,d} \\ \mathbf{0}^T & 1 \end{pmatrix} = ({}^w\mathbf{T}_B)^{-1} {}^w\mathbf{T}_A{}^A\mathbf{T}_{EA}{}^{EA}\mathbf{T}_{EB} \\ &= \begin{pmatrix} -0.5 & -0.8660 & 0 & -2.1651 \\ 0.8660 & -0.5 & 0 & 0.5179 \\ 0 & 0 & 1 & 0 \\ \mathbf{0}^T & & & 1 \end{pmatrix} = \end{aligned}$$

desired end-effector frame
of robot **B** w.r.t. its base
= input for the
inverse kinematics of robot **B**!

one solution \mathbf{q}_B (out of 2) of the
inverse kinematics of robot **B**

$$\mathbf{q}_B = \begin{bmatrix} q_{B1} \\ q_{B2} \\ q_{B3} \end{bmatrix} = \begin{bmatrix} 2.7939 \\ 1.1076 \\ -1.8071 \end{bmatrix} [\text{rad}] = \begin{bmatrix} 160.08^\circ \\ 63.46^\circ \\ -103.54^\circ \end{bmatrix}$$





Remarks on homogeneous matrices

- the main tool used for computing the **direct kinematics** of robot manipulators
- relevant in many other applications (in robotics and beyond)
 - in positioning/orienting a vision camera (matrix bT_c with extrinsic parameters of the camera pose)
 - in computer graphics, for the real-time visualization of 3D solid objects when changing the observation point

$${}^A T_B = \begin{bmatrix} {}^A R_B & {}^A p_{AB} \\ \alpha_x & \alpha_y & \alpha_z \\ \end{bmatrix}$$

all zero
in robotics

coefficients of
perspective
deformation

scaling
coefficient

always unitary
in robotics



Robotics 1

Direct kinematics

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Kinematics of robot manipulators

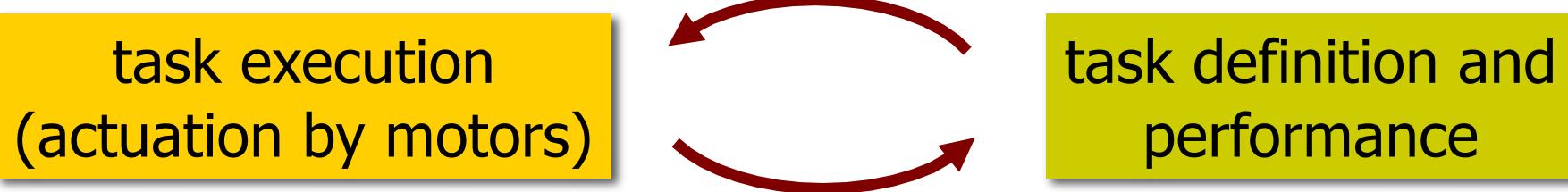
- study of ...
geometric and timing aspects of **robot motion**,
without reference to the causes producing it

- robot seen as ...
an (open) **kinematic chain** of rigid bodies
interconnected by (revolute or prismatic) joints



Motivations

- functional aspects
 - definition of robot workspace
 - calibration
- operational aspects



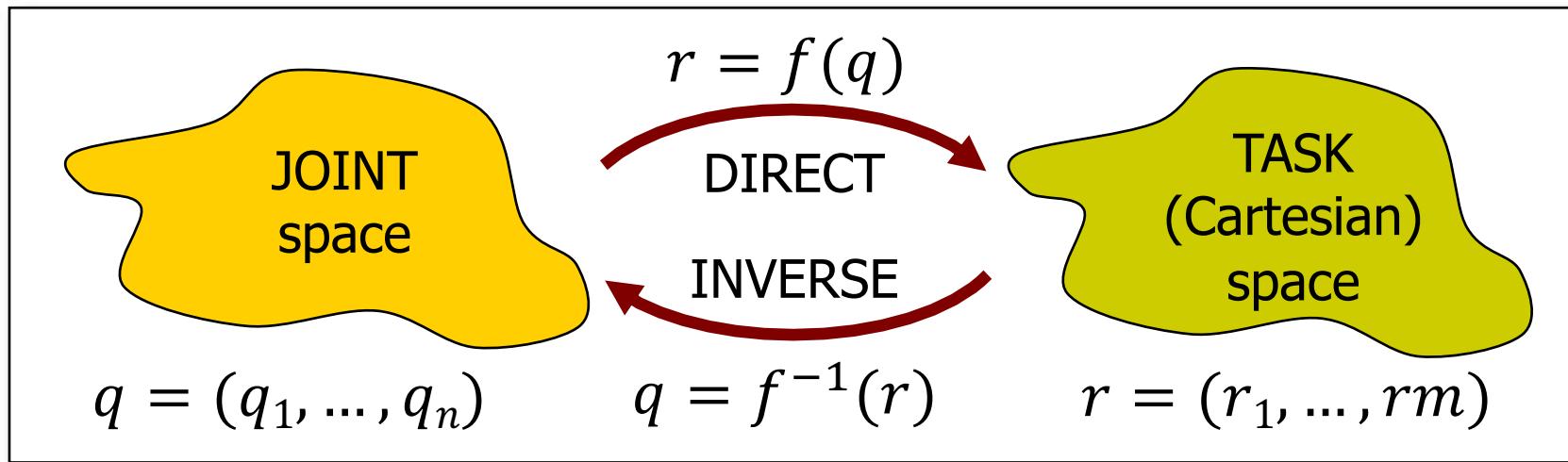
two different “spaces” related by kinematic (and dynamic) maps

- trajectory planning
- programming
- motion control



Kinematics

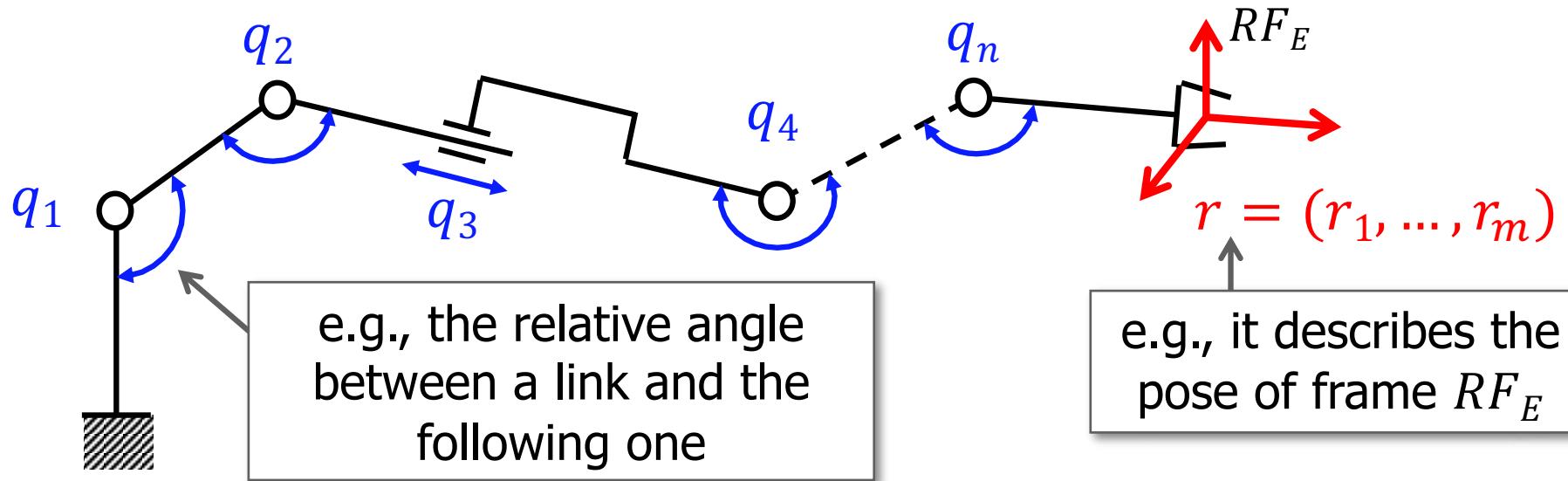
formulation and parameterizations



- choice of parameterization q
 - unambiguous and minimal characterization of robot configuration
 - $n = \#$ degrees of freedom (dof) = $\#$ robot joints (rotational or translational)
- choice of parameterization r
 - compact description of position and/or orientation (pose) variables of interest to the required task
 - usually, $m \leq n$ and $m \leq 6$ (but none of these is strictly necessary)



Open kinematic chains

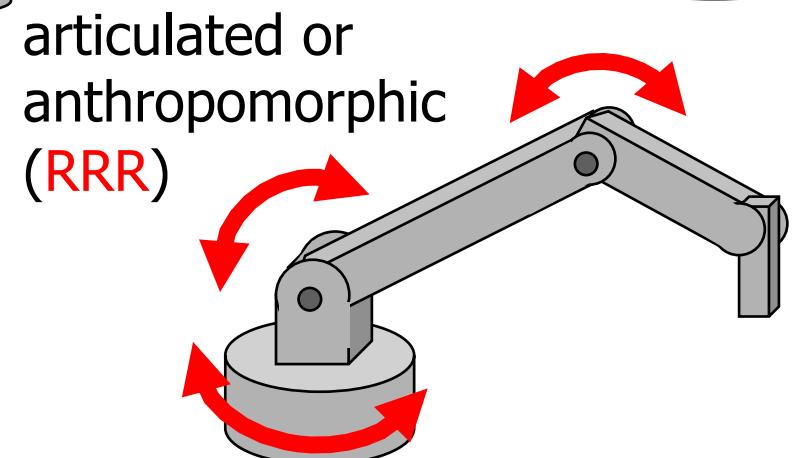
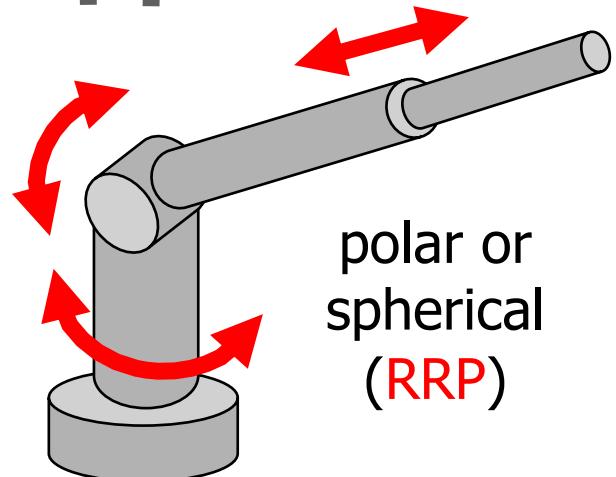
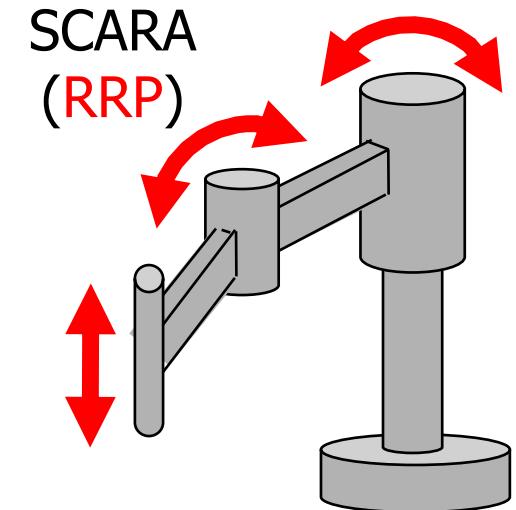
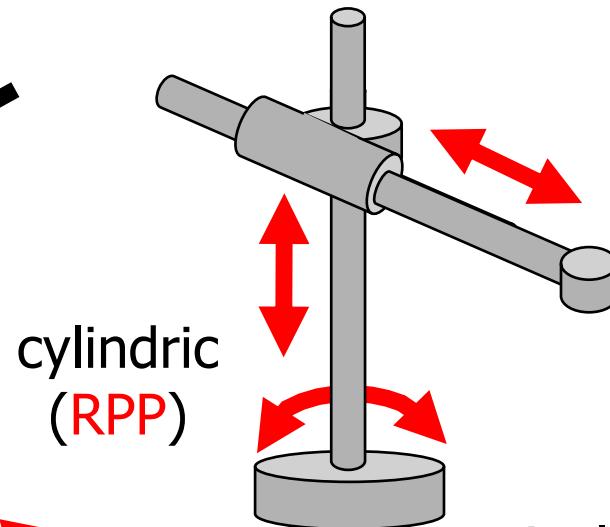
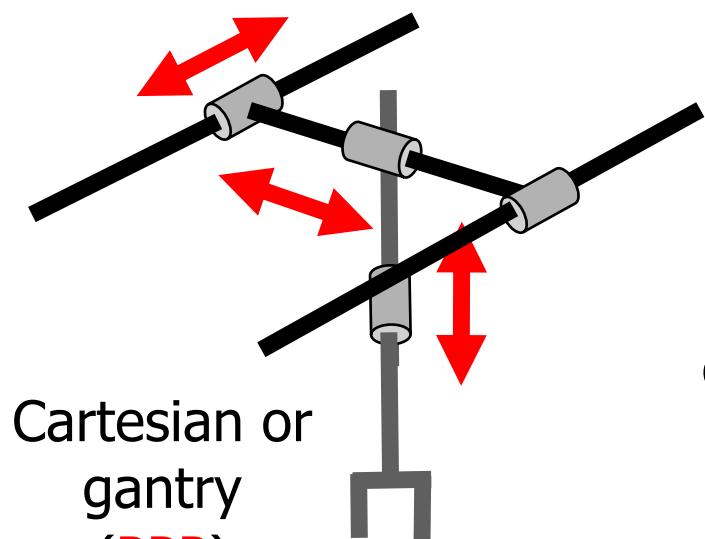


- $m = 2$
 - pointing in space
 - positioning in the plane
- $m = 3$
 - orientation in space
 - positioning and orientation in the plane
- $m = 5$
 - positioning and pointing in space (like for spot welding)
- $m = 6$
 - positioning and orientation in space
 - positioning of two points in space (e.g., end-effector and elbow)



Classification by kinematic type

first 3 dofs only



R = 1-dof rotational (revolute) joint
P = 1-dof translational (prismatic) joint



Direct kinematic map

- the structure of the **direct kinematics** function depends on the chosen r

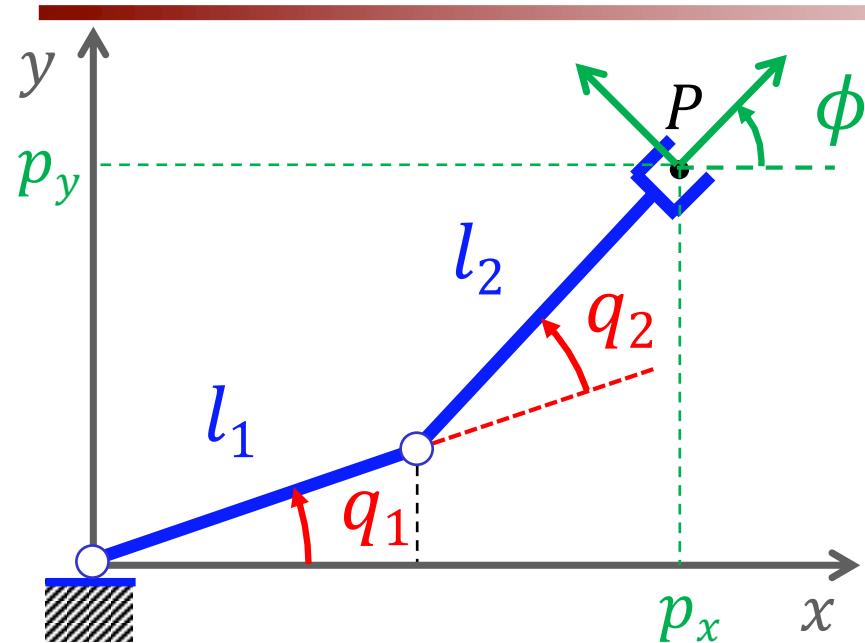
$$r = f_r(q)$$

- methods for computing $f_r(q)$
 - geometric/by inspection
 - systematic: assigning **frames attached to the robot links** and using homogeneous transformation matrices



Direct kinematics of 2R planar robot

just using inspection...



$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

$$n = 2$$

$$r = \begin{bmatrix} p_x \\ p_y \\ \phi \end{bmatrix}$$

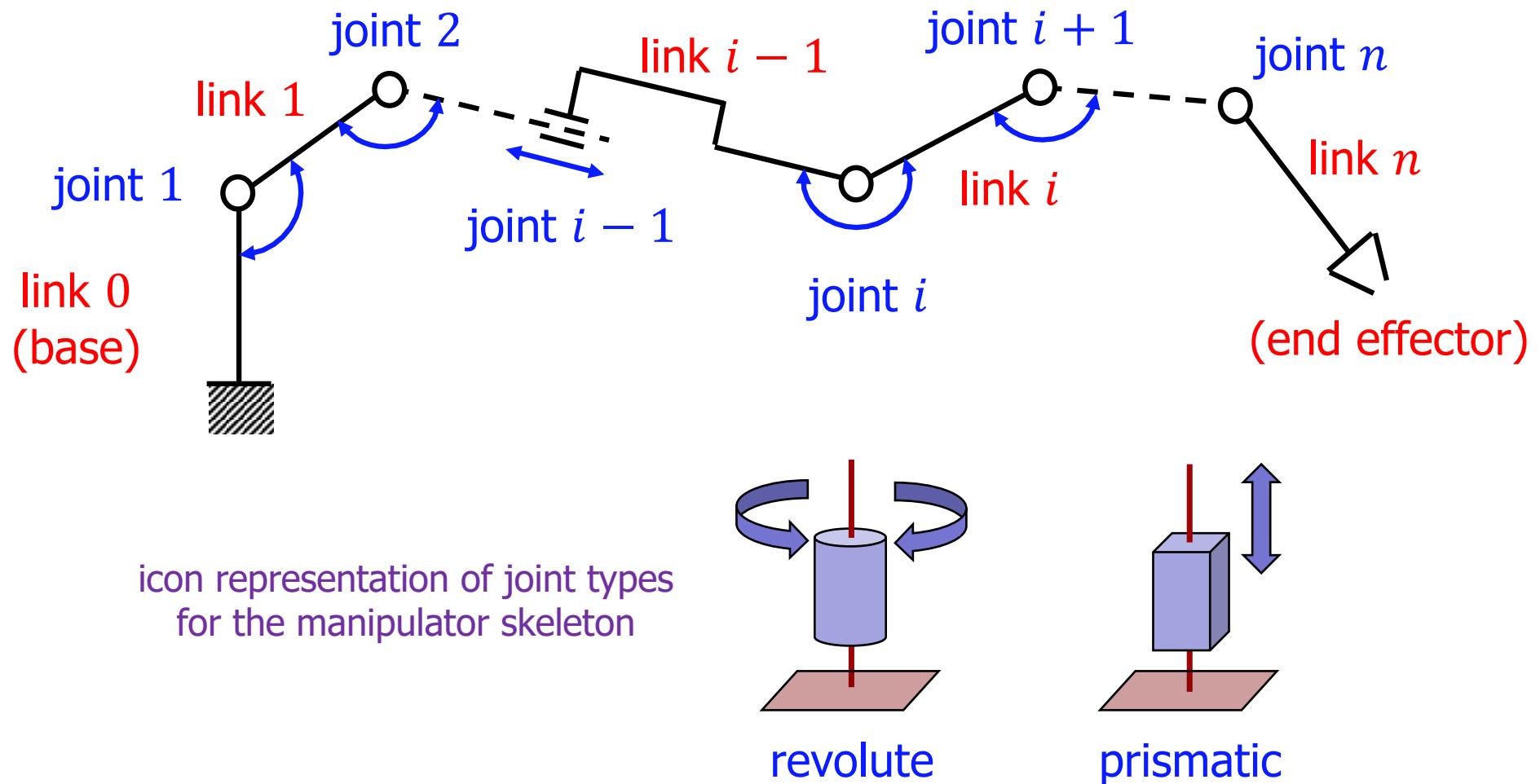
$$m = 3$$

$$p_x = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$
$$p_y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$
$$\phi = q_1 + q_2$$

for more general cases, we need a 'method'!

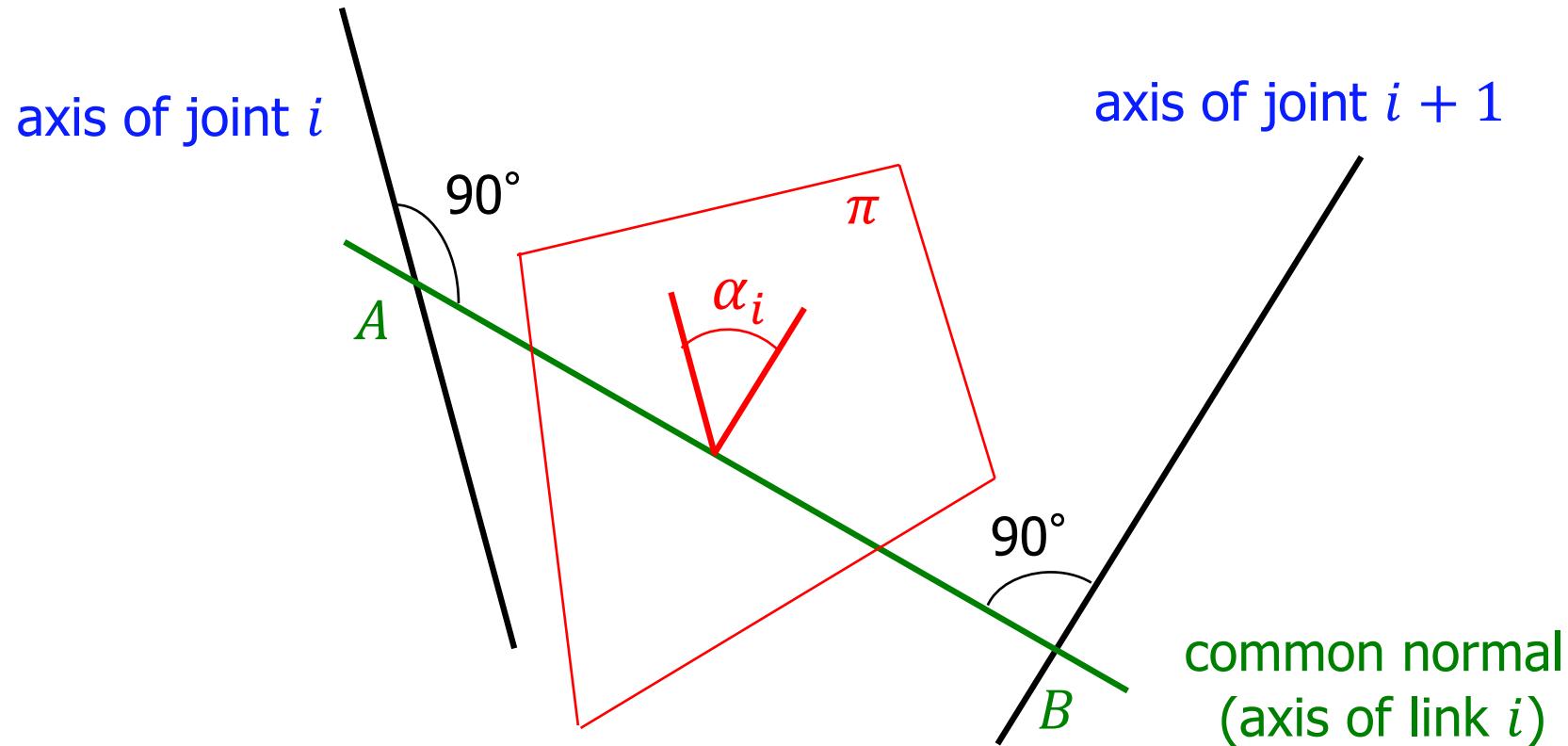


Numbering links and joints





Spatial relation between joint axes

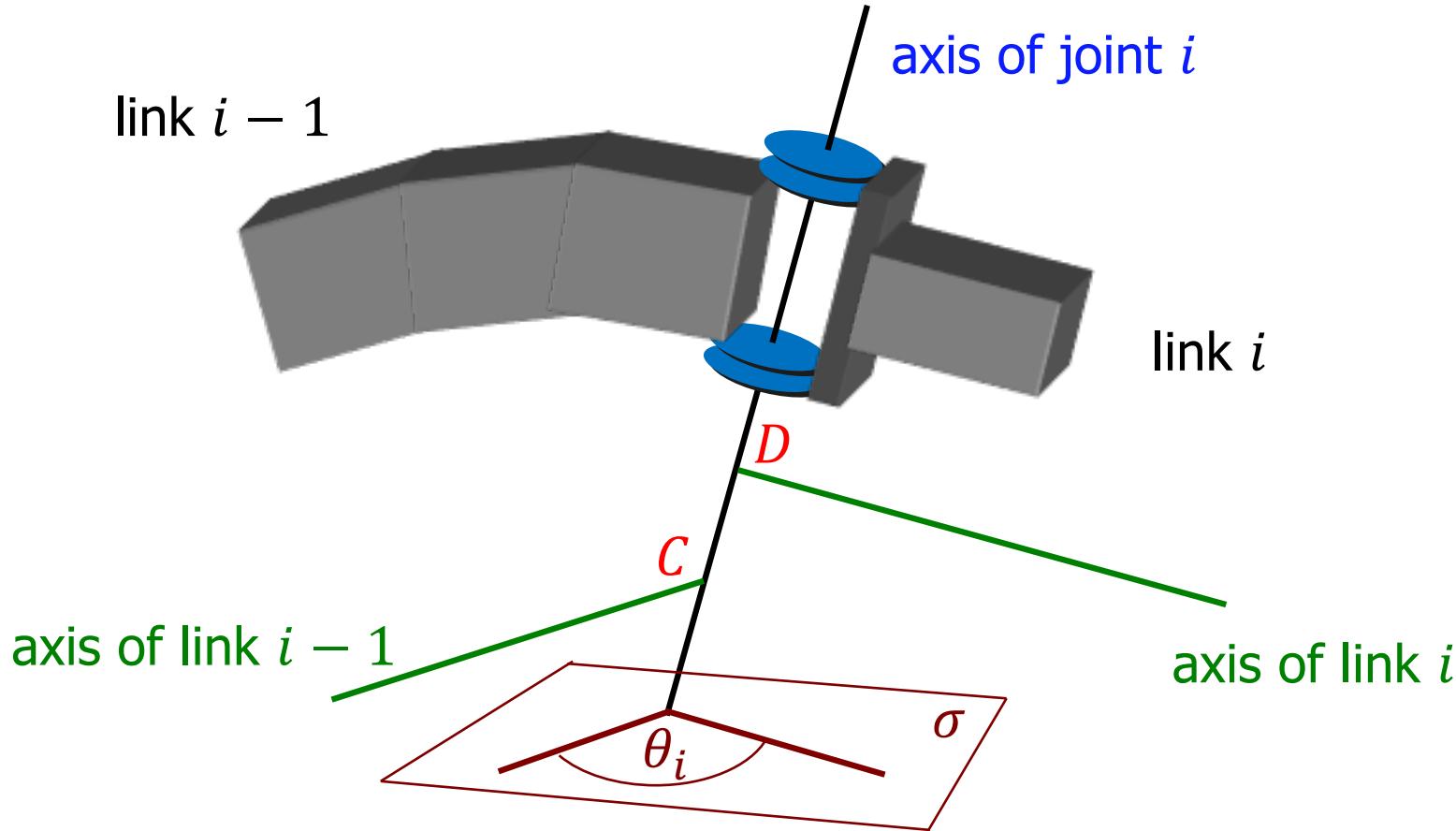


a_i = **displacement** AB between joint axes (always well defined)
 α_i = **twist angle** between joint axes
 — projected on a plane π orthogonal to the link axis

} with sign (pos/neg)!



Spatial relation between link axes



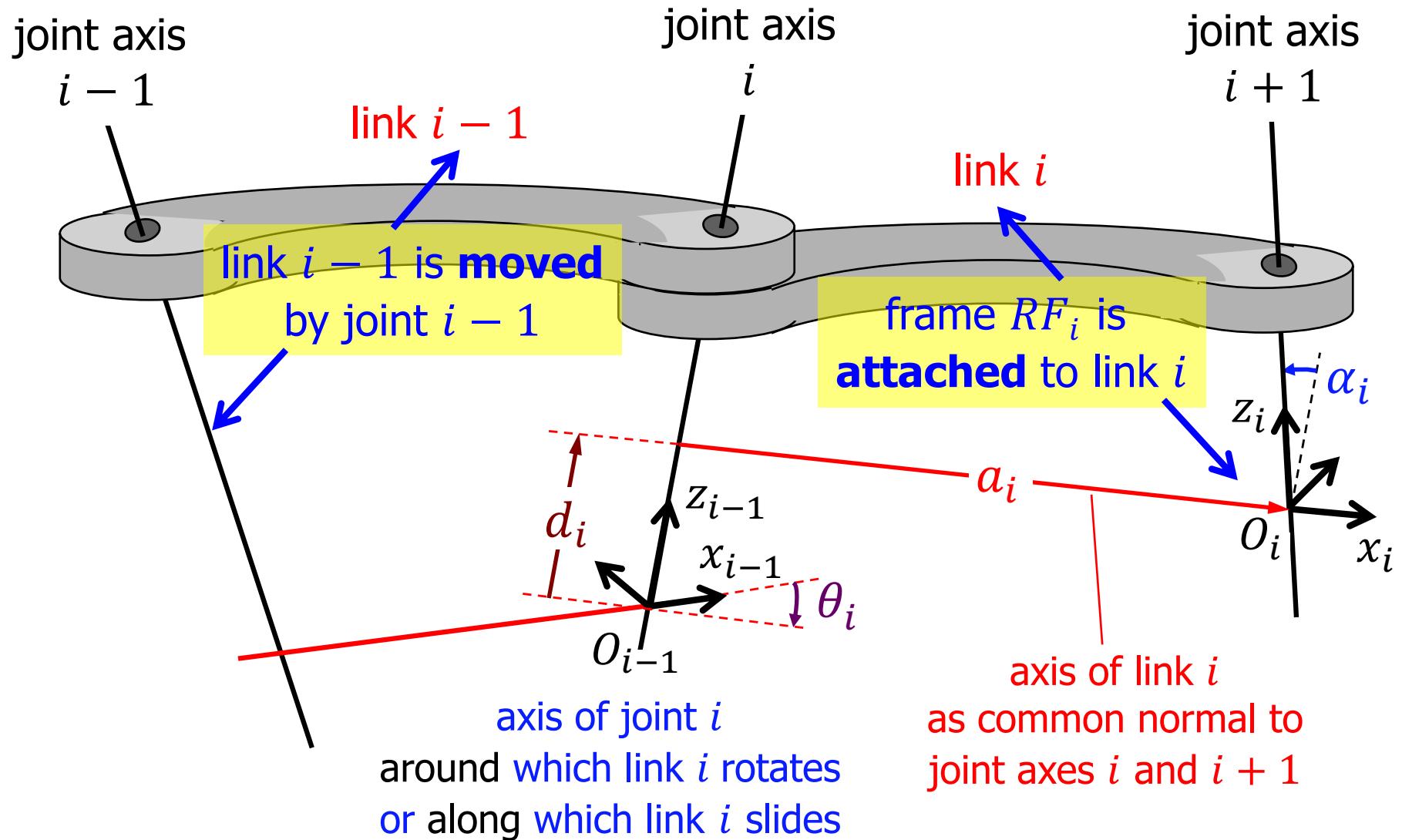
d_i = **displacement** CD (a variable if joint i is prismatic)

θ_i = **angle between link axes** (a variable if joint i is revolute)
— projected on a plane σ orthogonal to the joint axis

} with sign
(pos/neg)!

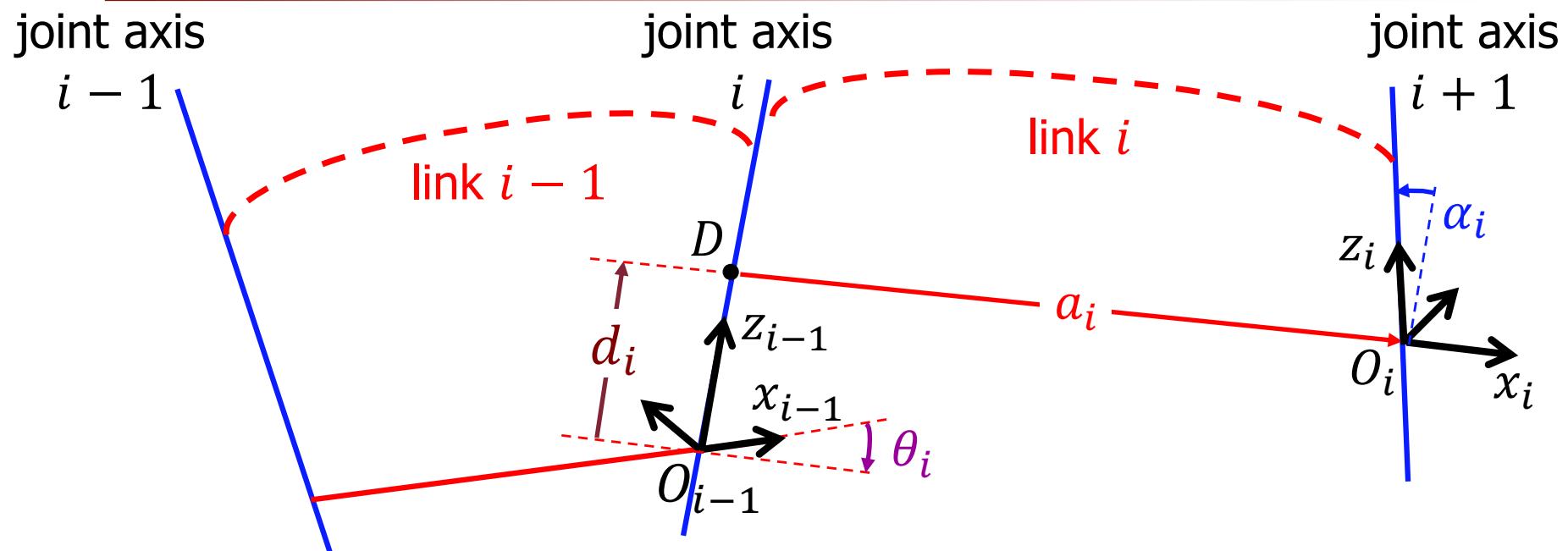


Denavit-Hartenberg (DH) frames





Definition of DH parameters



- unit vector z_i along **axis** of joint $i + 1$
- unit vector x_i along the **common normal** to joint i and $i + 1$ axes ($i \rightarrow i + 1$)
- a_i = distance DO_i , + if oriented as x_i , always constant (= '**length**' of link i)
- d_i = distance $O_{i-1}D$, + if oriented as z_{i-1} , **variable** if joint i is **PRISMATIC**
- α_i = **twist** angle from z_{i-1} to z_i around x_i , + if CCW, always constant
- θ_i = angle from x_{i-1} to x_i around z_{i-1} , + if CCW, **variable** if joint i is **REVOLUTE**



DH layout made simple

a popular 3-minute illustration...

video



<https://www.youtube.com/watch?v=rA9tm0gTln8>

- **note:** the author of this video uses r in place of a , and does not add subscripts!



Homogeneous transformation

between successive DH frames (from frame $i - 1$ to frame i)

- roto-translation (screw motion) around and along z_{i-1}

$${}^{i-1}A_i'(\boldsymbol{q}_i) = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the product of these two matrices commutes!

rotational joint $\Rightarrow q_i = \theta_i$

prismatic joint $\Rightarrow q_i = d_i$

- roto-translation (screw motion) around and along x_i

$${}^iA'_i = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos \alpha_i & -\sin \alpha_i & 0 \\ 0 & \sin \alpha_i & \cos \alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

← always a
constant matrix



Denavit-Hartenberg matrix

J. Denavit and R.S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices,"
Trans. ASME J. Applied Mechanics, **23**: 215–221, 1955

$${}^{i-1}A_i(q_i) = {}^{i-1}A_{i'}(q_i) \ {}^{i'}A_i = \begin{bmatrix} \cos \theta_i & -\cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cos \theta_i & -\sin \alpha_i \cos \theta_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

compact notation: $c = \cos$, $s = \sin$

super-compact notation (if feasible): $c_i = \cos q_i$, $s_i = \sin q_i$



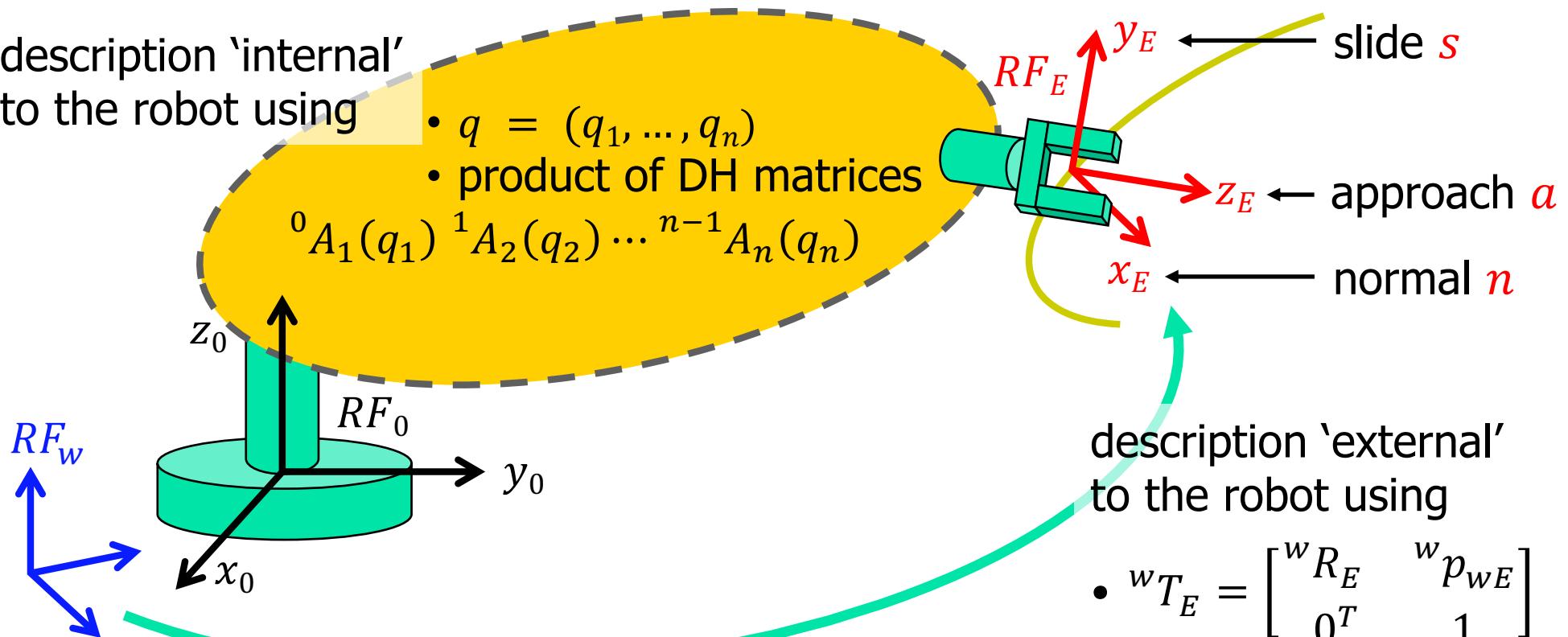
Ambiguities in defining DH frames

- **frame 0:** origin and x_0 axis are arbitrary
- **frame n :** z_n axis is not specified
 - however, x_n **must** intersect and be chosen orthogonal to z_{n-1}
- **positive** direction of z_{i-1} (up/down on axis of joint i) is arbitrary
 - choose one, and try to '**avoid flipping over**' to the next one
- **positive** direction of x_i (back/forth on axis of link i) is arbitrary
 - if successive joint axes are incident, we often take $x_i = z_{i-1} \times z_i$
 - when natural, follow the direction '**from base to tip**'
- if z_{i-1} and z_i are **parallel** (common normal not uniquely defined)
 - O_i chosen arbitrarily along z_i , still trying to '**zero out**' parameters
- if z_{i-1} and z_i are **coincident**, normal x_i axis can be chosen at will
 - this case occurs **only** if the two joints are of different kind (P/R or R/P)
 - again, try using '**simple values**' (e.g., 0 or $\pm\pi/2$) for constant angles



Direct kinematics of robot manipulators

description 'internal'
to the robot using



$$\begin{aligned} {}^wT_E &= {}^wT_0 {}^0A_1(q_1) {}^1A_2(q_2) \dots {}^{n-1}A_n(q_n) {}^nT_E \\ r &= f_r(q) \end{aligned}$$

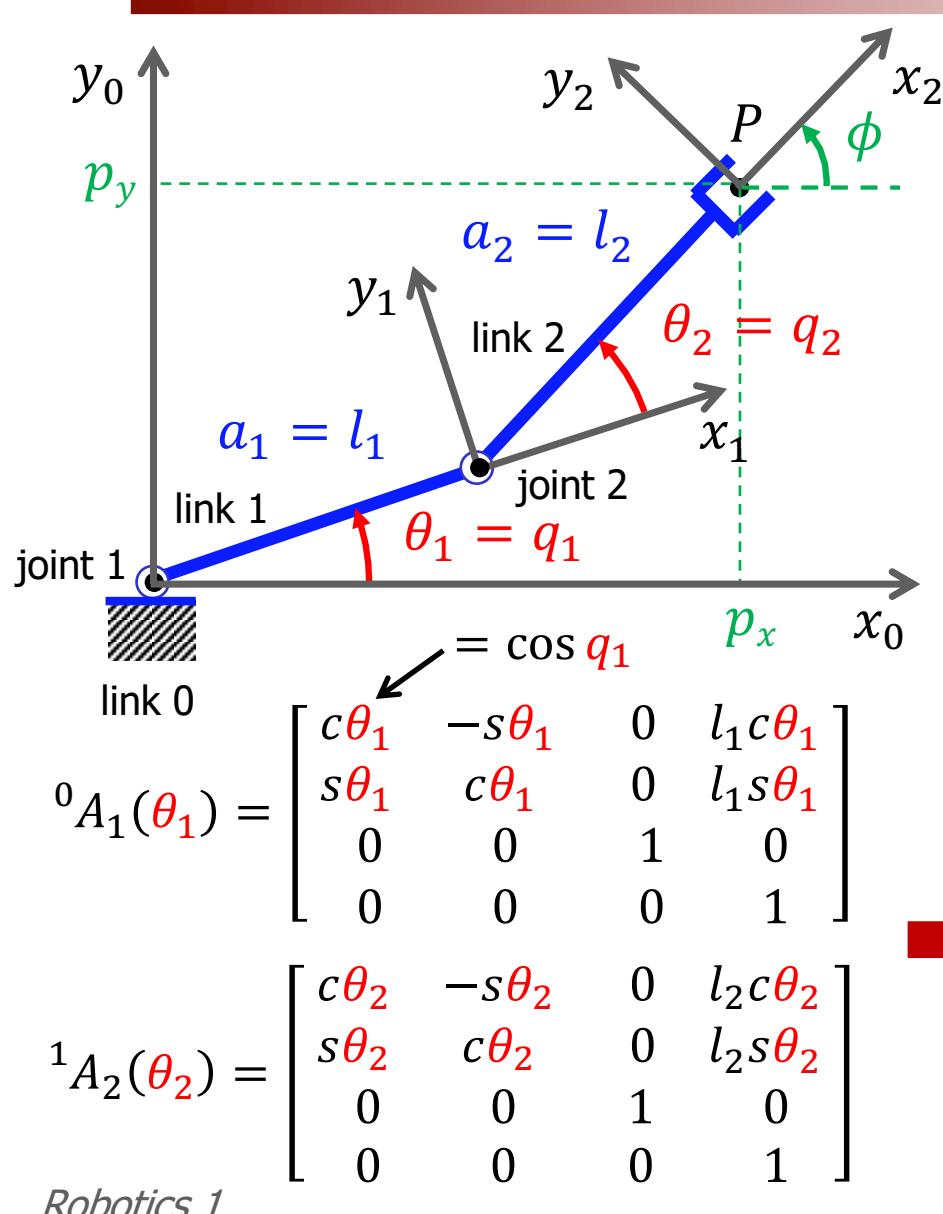
description 'external'
to the robot using

$$\begin{aligned} \bullet \quad {}^wT_E &= \begin{bmatrix} {}^wR_E & {}^wp_{wE} \\ 0^T & 1 \end{bmatrix} \\ &= \begin{bmatrix} n & s & a & p \\ 0^T & 1 \end{bmatrix} \\ \bullet \quad r &= (r_1, \dots, r_m) \end{aligned}$$

alternative representations of the **direct kinematics**



Direct kinematics of 2R planar robot using DH frame assignment...



z_0, z_1, z_2 outgoing from plane

i	α_i	a_i	d_i	θ_i
1	0	l_1	0	q_1
2	0	l_2	0	q_2

$$\cos(q_1 + q_2)$$

$${}^0A_2(q) = \begin{bmatrix} c_{12} & -s_{12} & 0 & l_1c_1 + l_2c_{12} \\ s_{12} & c_{12} & 0 & l_1s_1 + l_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

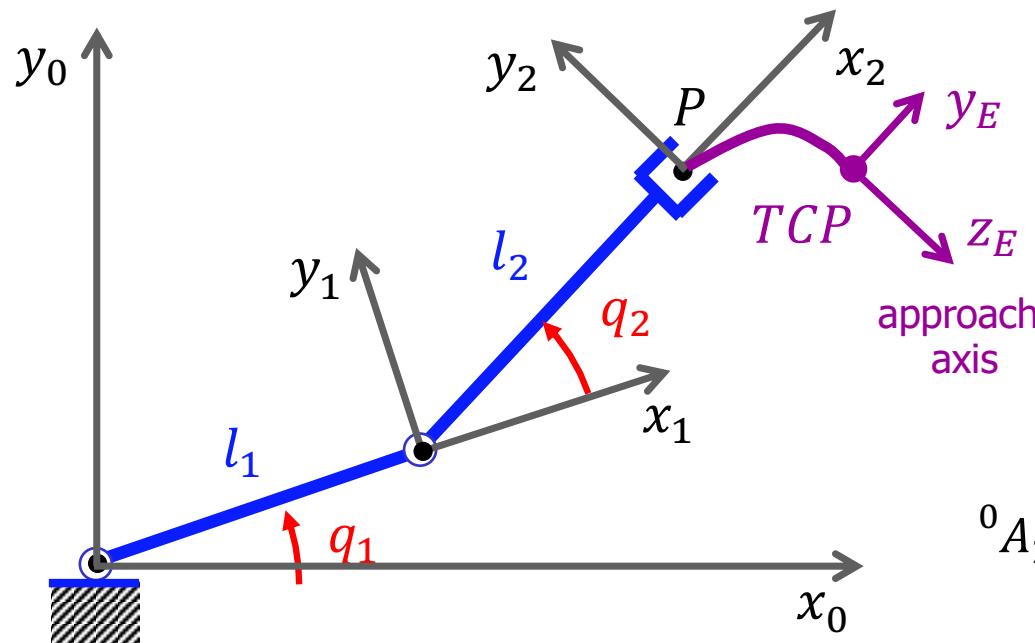
$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = {}^0A_2(q) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} l_1c_1 + l_2c_{12} \\ l_1s_1 + l_2s_{12} \\ 0 \\ 1 \end{bmatrix}$$

$$\phi = q_1 + q_2 \quad (\text{extracted from } {}^0R_2(q))$$



Direct kinematics of 2R planar robot

TCP location on the robot end effector



i	α_i	a_i	d_i	θ_i
1	0	l_1	0	q_1
2	0	l_2	0	q_2

$${}^0A_2(q) = \begin{bmatrix} c_{12} & -s_{12} & 0 & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & 0 & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tool Center Point TCP and associated end-effector frame RF_E

$${}^2T_E = \begin{bmatrix} 0 & 1 & 0 & {}^2TCP_x \\ 0 & 0 & -1 & {}^2TCP_y \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^0TCP(q) \\ 1 \end{bmatrix} = \begin{bmatrix} {}^0TCP_x(q) \\ {}^0TCP_y(q) \\ 0 \\ 1 \end{bmatrix} = {}^0A_2(q) \begin{bmatrix} {}^2TCP_x \\ {}^2TCP_y \\ 0 \\ 1 \end{bmatrix} = {}^0T_E(q) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = {}^0A_2(q) {}^2T_E$$

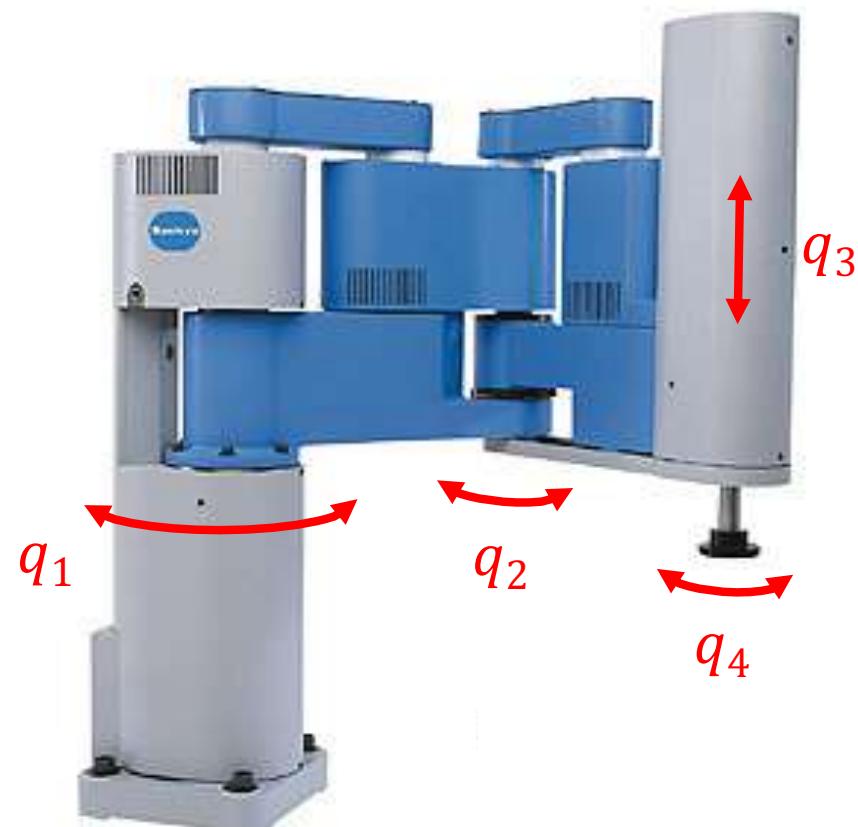


DH assignment for a SCARA robot

video



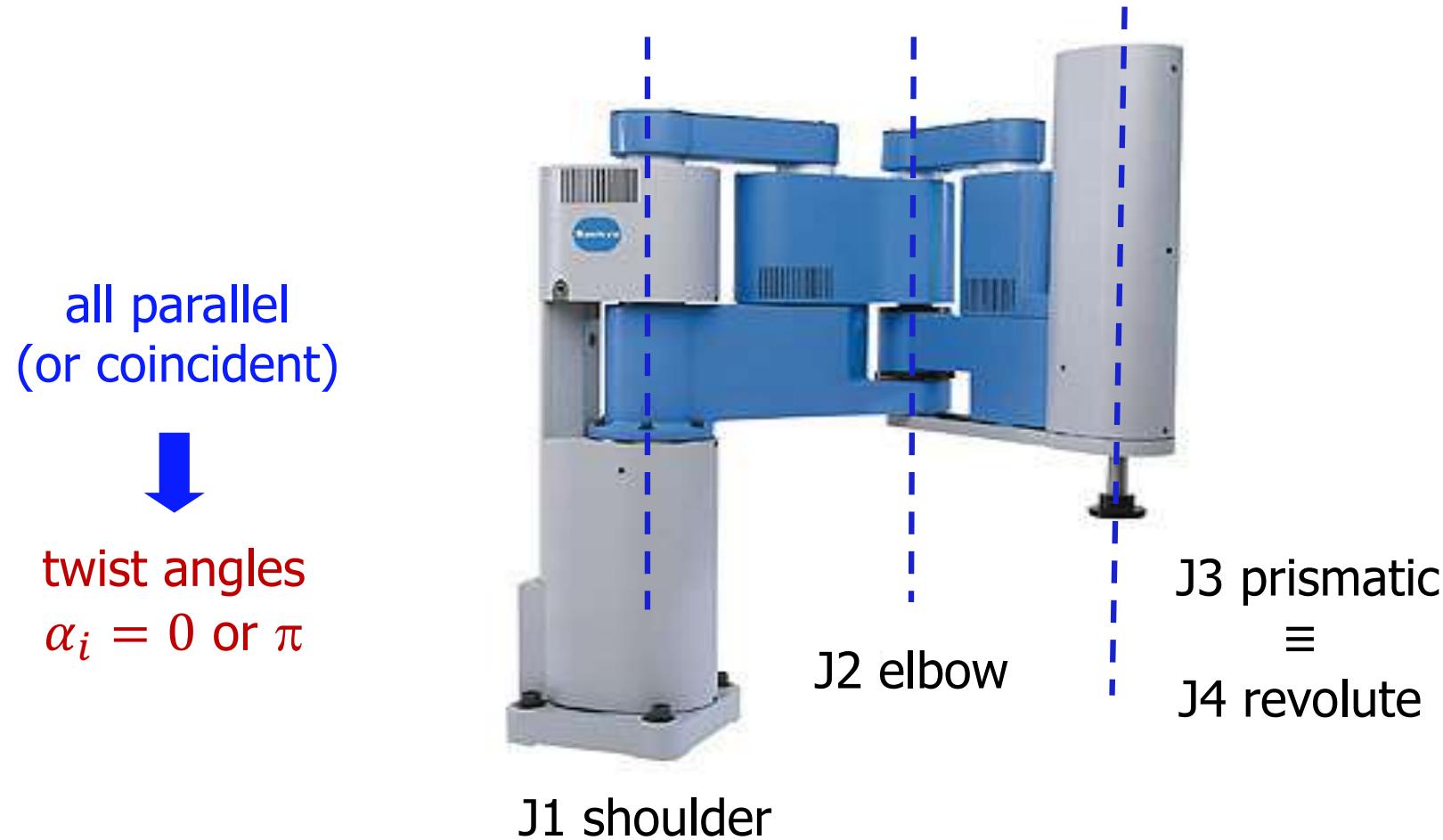
Sankyo SCARA 8438



Sankyo SCARA SR 8447



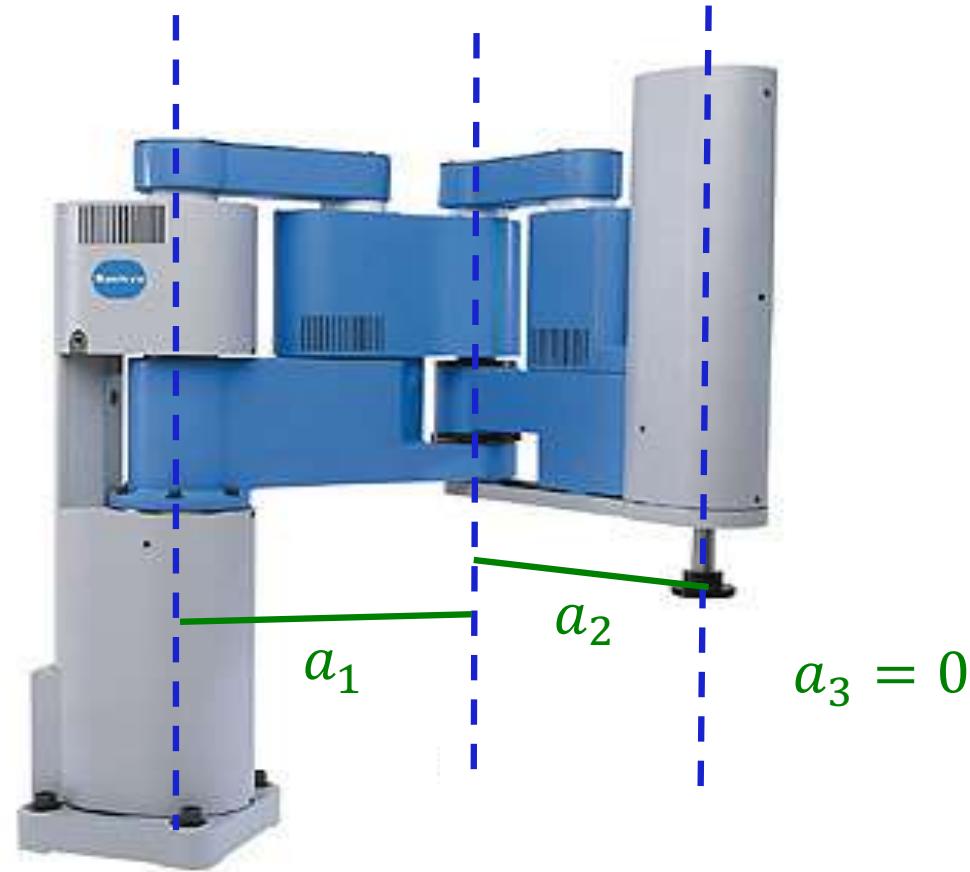
Step 1: joint axes





Step 2: link axes

the vertical 'heights'
of the **link axes**
are arbitrary
(for the time being)

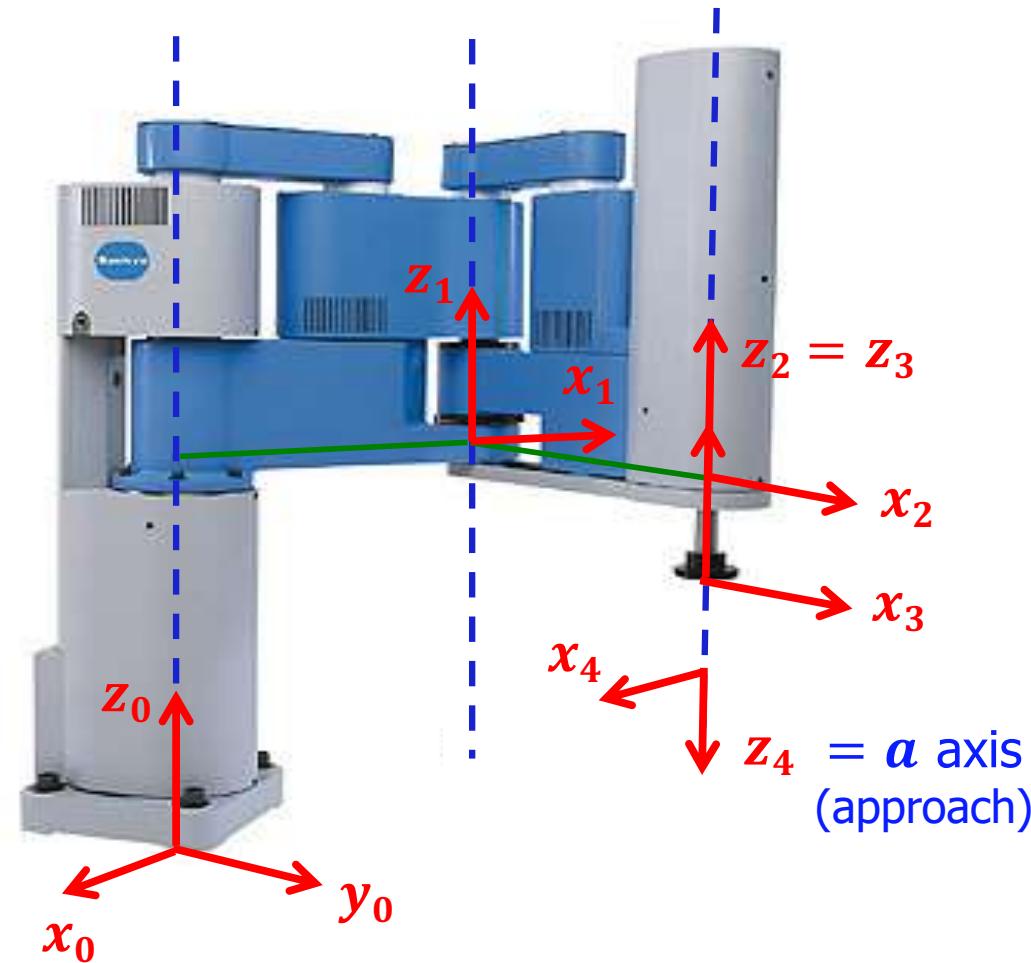




Step 3: frames

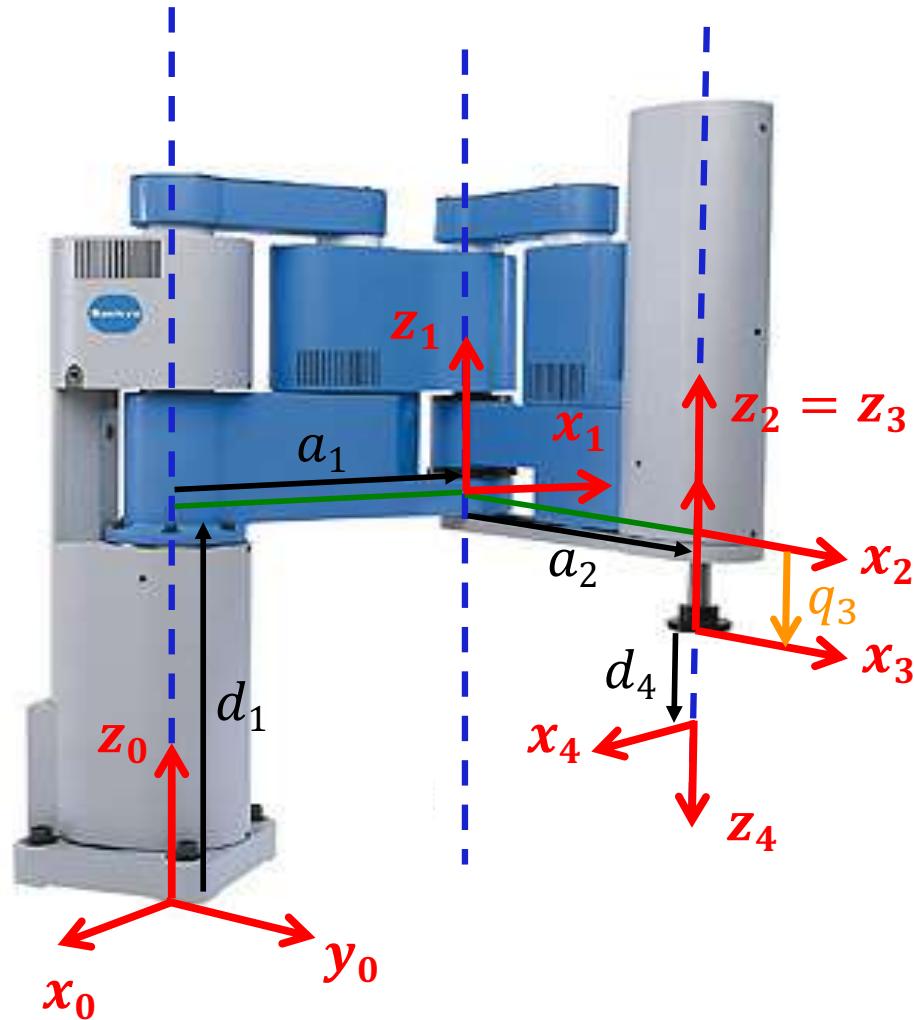
axes y_i for $i > 0$
are not shown

(nor needed; they form
right-handed frames)





Step 4: DH table of parameters



i	α_i	a_i	d_i	θ_i
1	0	a_1	d_1	q_1
2	0	a_2	0	q_2
3	0	0	q_3	0
4	π	0	d_4	q_4

note that

- d_1 and d_4 could be set = 0
- $d_4 < 0$ (opposite to z_3)
- also, $q_3 < 0$ in this configuration



Step 5: DH transformation matrices

$${}^0A_1(q_1) = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & a_1 c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & a_1 s\theta_1 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1A_2(q_2) = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2 s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2A_3(q_3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} q &= (q_1, q_2, q_3, q_4) \\ &= (\theta_1, \theta_2, d_3, \theta_4) \end{aligned}$$

$${}^3A_4(q_4) = \begin{bmatrix} c\theta_4 & s\theta_4 & 0 & 0 \\ s\theta_4 & -c\theta_4 & 0 & 0 \\ 0 & 0 & -1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Step 6a: direct kinematics

homogeneous matrix wT_E as product of the ${}^{i-1}A_i(q_i)$'s

$${}^0A_2(q_1, q_2) = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12} & c_{12} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0A_3(q_1, q_2, q_3) = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12} & c_{12} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & d_1 + q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^wT_E = {}^0A_4(q_1, q_2, q_3, q_4) = \begin{bmatrix} R(q_1, q_2, q_4) = [n \ s \ a] & p = p(q_1, q_2, q_3) \\ \boxed{\begin{bmatrix} c_{124} & s_{124} & 0 \\ s_{124} & -c_{124} & 0 \\ 0 & 0 & -1 \end{bmatrix}} & \boxed{\begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ d_1 + q_3 + d_4 \end{bmatrix}} \\ ({}^wT_0 = {}^4T_E = I) & \begin{bmatrix} 0 & 0 & 0 \\ & & 1 \end{bmatrix} \end{bmatrix}$$



Step 6b: direct kinematics

as task vector $r \in \mathbb{R}^m$

$${}^0A_4(q_1, q_2, q_3, q_4) = \begin{bmatrix} c_{124} & s_{124} & 0 \\ s_{124} & -c_{124} & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ d_1 + q_3 + d_4 \\ 1 \end{bmatrix}$$

extract α_z
from
 $R(q_1, q_2, q_4)$

\downarrow

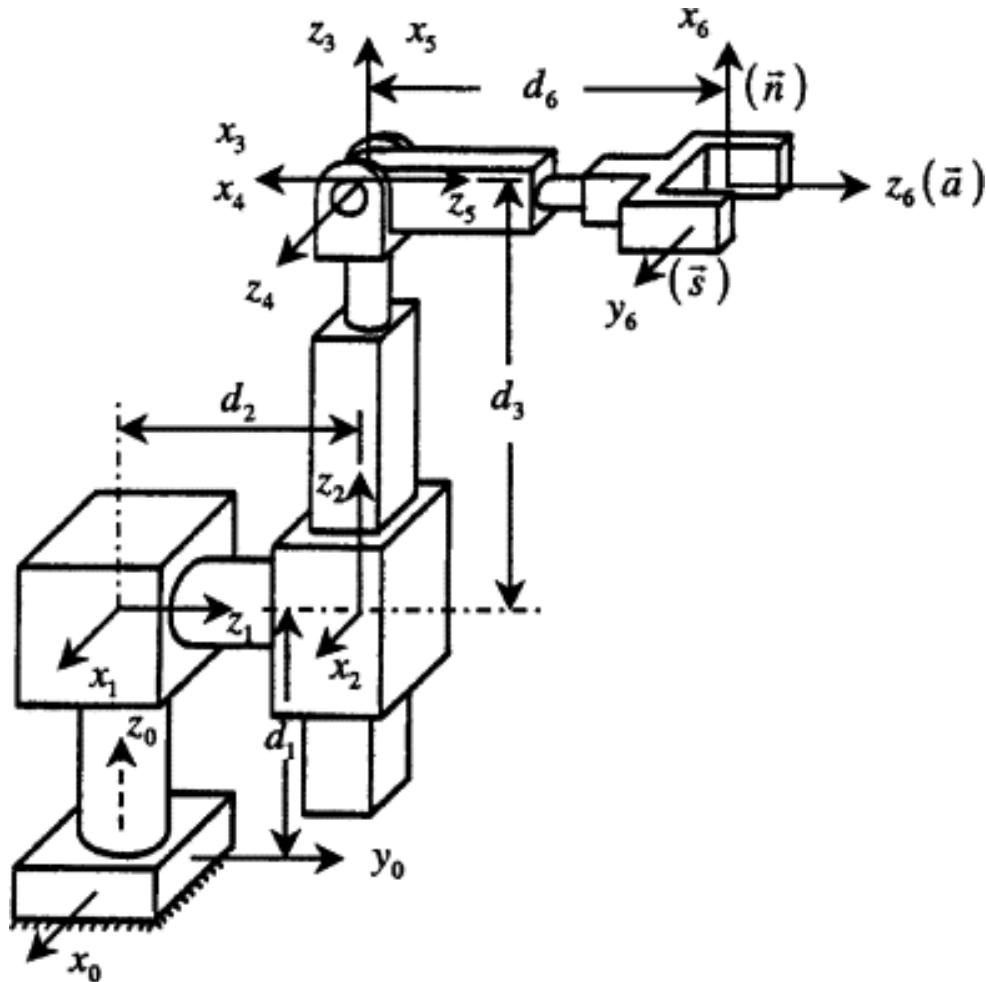
take $p \in \mathbb{R}^4$
as such from
 $p(q_1, q_2, q_3)$

$$r = \begin{bmatrix} p_x \\ p_y \\ p_z \\ \alpha_z \end{bmatrix} = f_r(q) = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ d_1 + q_3 + d_4 \\ q_1 + q_2 + q_4 \end{bmatrix} \in \mathbb{R}^4$$



Stanford manipulator

- 6-dof: 2R-1P-3R (spherical wrist)

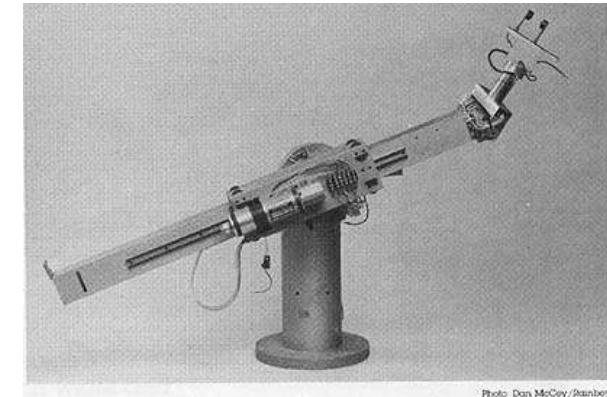
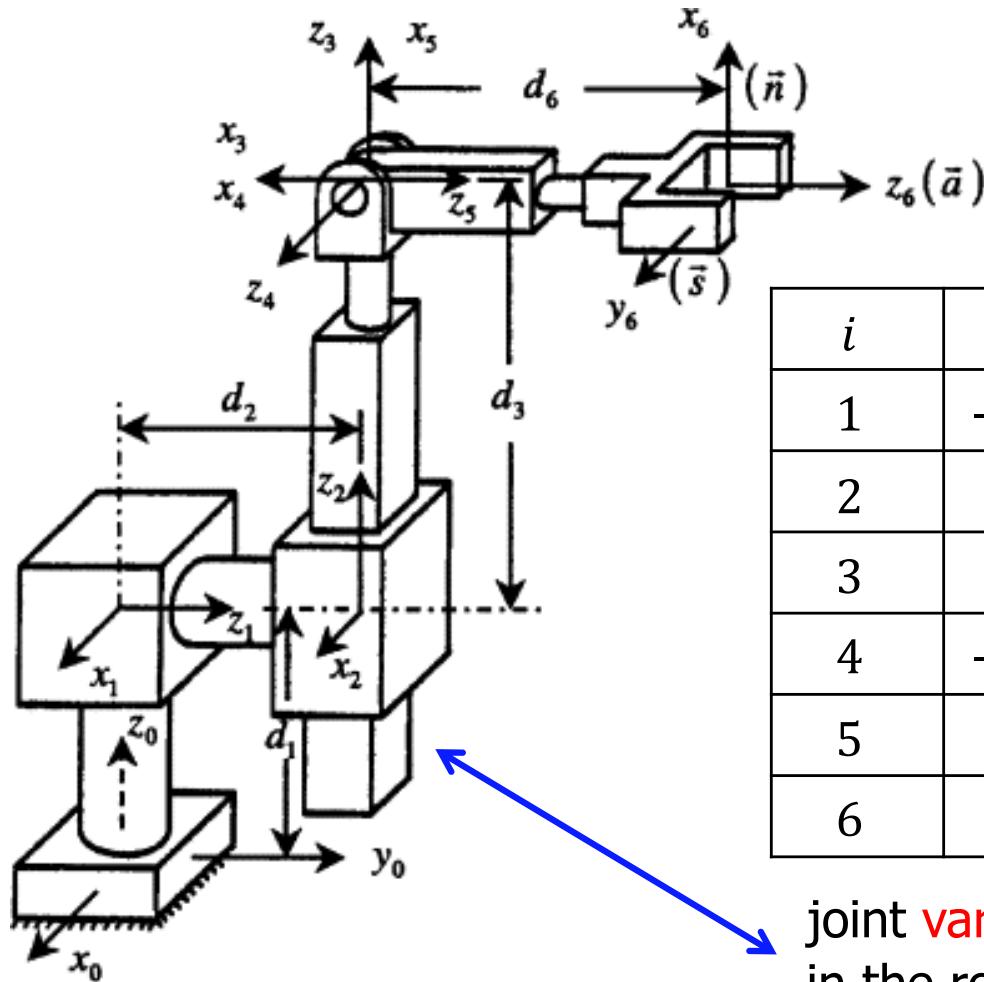


- robot with **shoulder** offset
- 'one possible' DH assignment of frames is shown
- determine the associated
 - [table of DH parameters](#)
 - homogeneous transformation matrices
 - direct kinematics
- write a program for computing the direct kinematics
 - [numerically](#) (Matlab), given a q
 - [symbolically](#) (Mathematica, Maple, Symbolic Manipulation Toolbox of Matlab, ...)



DH table for Stanford manipulator

- 6-dof: 2R-1P-3R (spherical wrist)



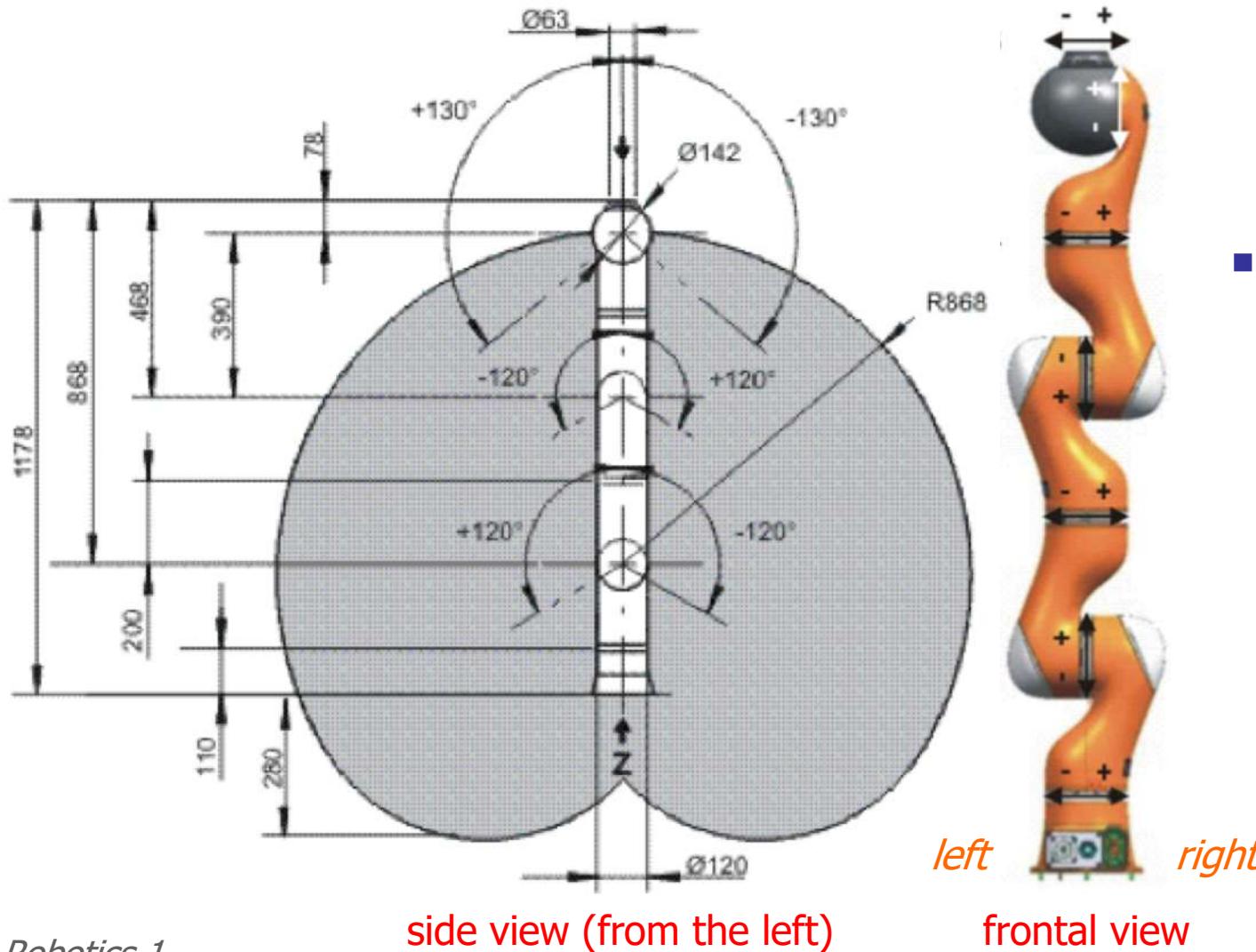
i	α_i	a_i	d_i	θ_i
1	$-\pi/2$	0	$d_1 > 0$	$q_1 = 0$
2	$\pi/2$	0	$d_2 > 0$	$q_2 = 0$
3	0	0	$q_3 > 0$	$-\pi/2$
4	$-\pi/2$	0	0	$q_4 = 0$
5	$\pi/2$	0	0	$q_5 = -\pi/2$
6	0	0	$d_6 > 0$	$q_6 = 0$

joint variables are in red, while their values in the robot configuration shown are in blue



KUKA LWR 4+

- 7R (no offsets, spherical shoulder and spherical wrist)



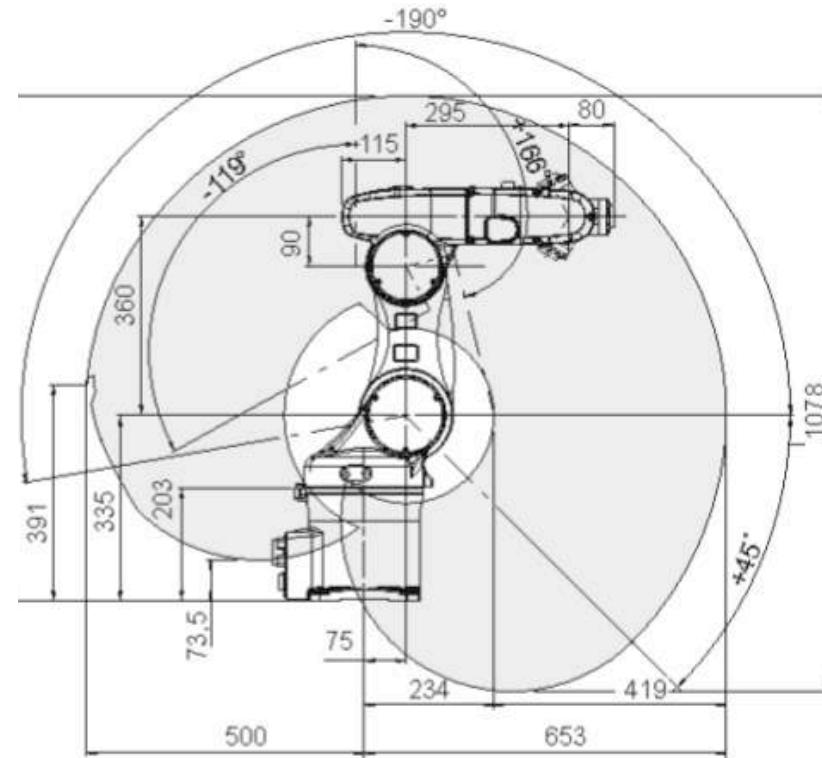
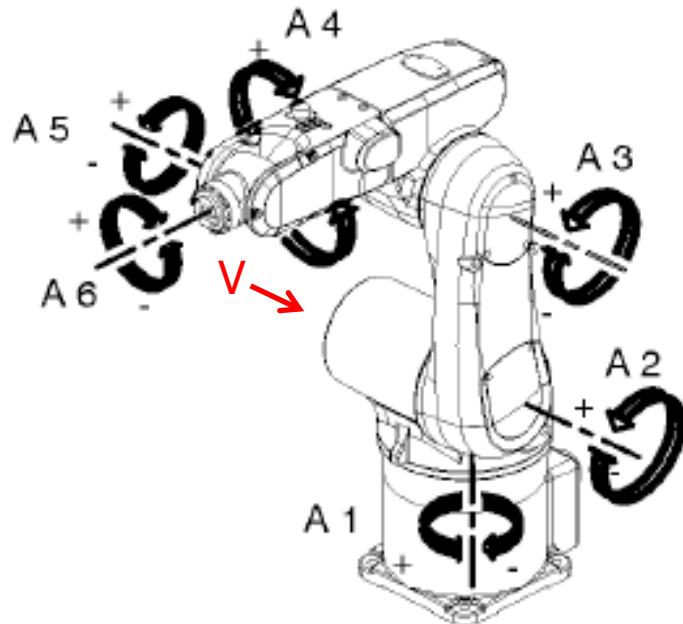
available at
DIAG Robotics Lab

- determine
 - frames and table of DH parameters
 - homogeneous transformation matrices
 - direct kinematics
 - d_1 and d_7 can be set = 0 or not (as needed)

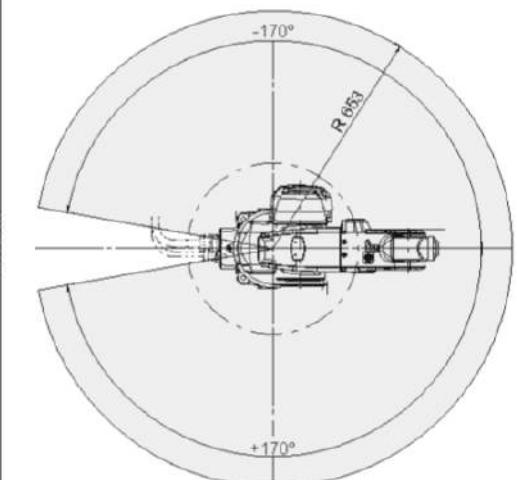


KUKA KR5 Sixx R650

- 6R (offsets at shoulder and elbow, spherical wrist)



side view (from observer in V)



top view

- determine
 - frames and table of DH parameters
 - homogeneous transformation matrices
 - direct kinematics

available at
DIAG Robotics Lab



Appendix: Modified DH convention

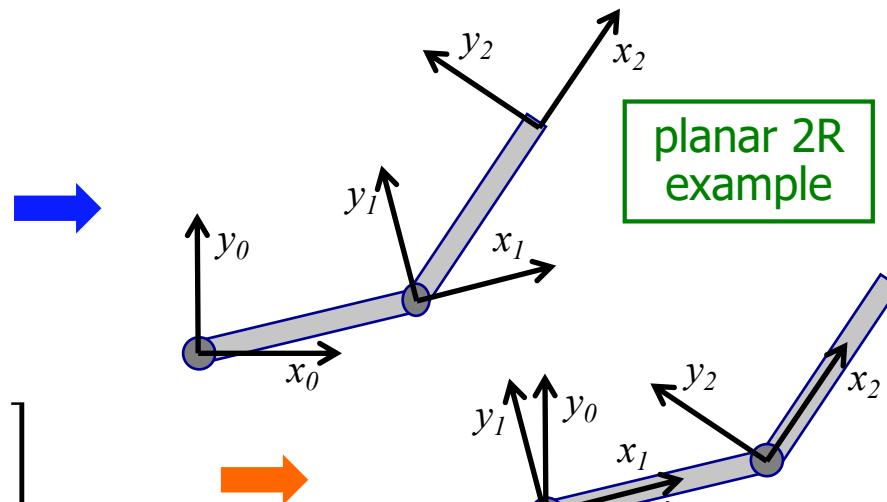
- a **modified** version introduced in J. Craig's book "Introduction to Robotics" (1986) and aligned for the indexing by Khalil and Kleinfinger (ICRA, 1986)
 - has z_i axis on joint i
 - a_i & α_i = distance & twist angle from z_{i-1} to z_i , measured along & about x_{i-1}
 - d_i & θ_i = distance & angle from x_{i-1} to x_i , measured along & about z_i
 - **source of much confusion...** if you are not aware of it (or don't mention it!)
 - convenient with link flexibility: a rigid frame at the base, another at the tip...

classical
(or distal)

$${}^{i-1}A_i = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

modified
(or proximal)

$${}^{i-1}A_i^{\text{mod}} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_i \\ c\alpha_i s\theta_i & c\alpha_i c\theta_i & -s\alpha_i & -d_i s\alpha_i \\ s\alpha_i s\theta_i & s\alpha_i c\theta_i & c\alpha_i & d_i c\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



modified DH tends to place frames
'at the base' of each link



Robotics 1

Inverse kinematics

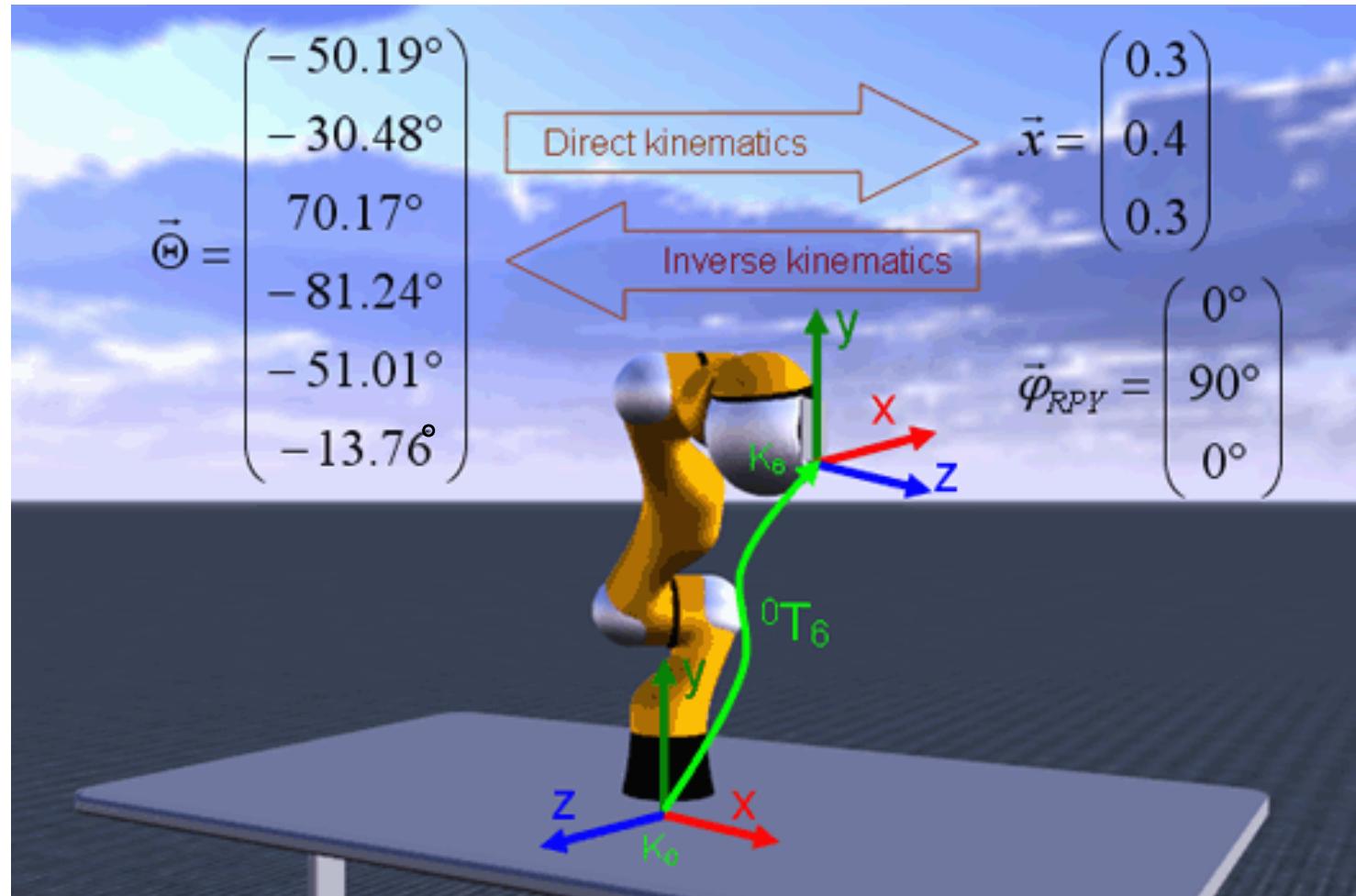
Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Inverse kinematics what are we looking for?



direct kinematics is always unique;
how about inverse kinematics for this 6R robot?



Inverse kinematics problem

- given a desired end-effector pose (position + orientation), **find** the values of the joint variables q that will realize it
- a **synthesis** problem, with **input** data in the form
 - $T = \begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix} = {}^0A_n(q)$
 - $r = f_r(q)$, for a task function

classical formulation:
inverse kinematics for a given end-effector pose T generalized formulation:
inverse kinematics for a given value r of task variables

- a typical **nonlinear** problem
 - **existence** of a solution (**workspace** definition)
 - uniqueness/**multiplicity** of solutions ($r \in \mathbb{R}^m$, $q \in \mathbb{R}^n$)
 - solution **methods**



Solvability and robot workspace

for tasks related to a desired end-effector Cartesian pose

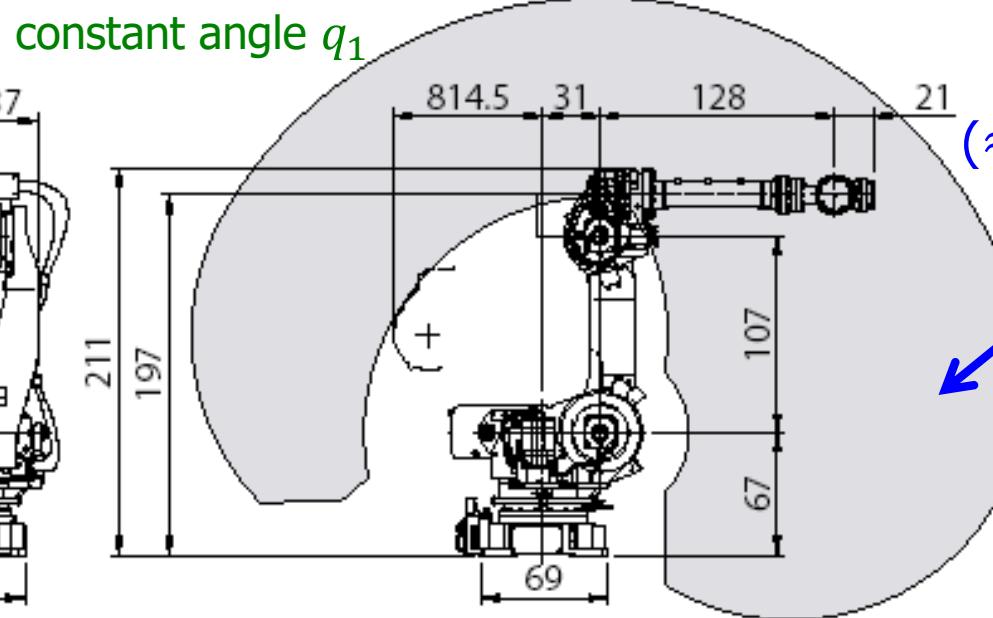
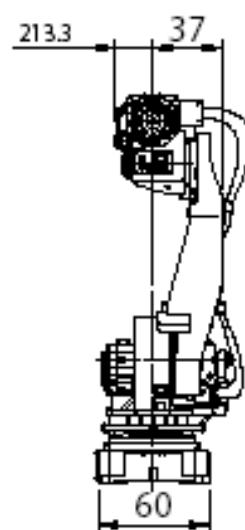
- primary workspace WS_1 : set of all positions p that can be reached with at least one orientation (ϕ or R)
 - out of WS_1 there is no solution to the problem
 - if $p \in WS_1$, there is a suitable ϕ (or R) for which a solution exists
- secondary (or dexterous) workspace WS_2 : set of positions p that can be reached with any orientation (among those feasible for the robot direct kinematics)
 - if $p \in WS_2$, there exists a solution for any feasible ϕ (or R)
- $WS_2 \subseteq WS_1$



Workspace of Fanuc R-2000i/165F

Area di lavoro
Operating Space

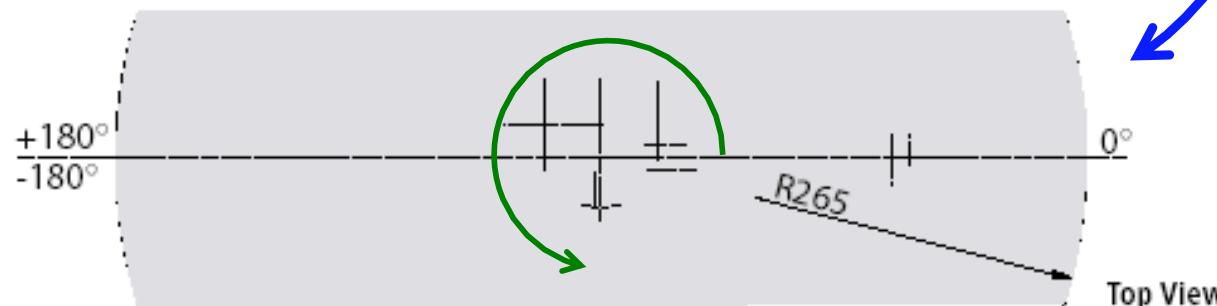
section for a
constant angle q_1



$$WS_1 \subset \mathbb{R}^3$$

($\approx WS_2$ for spherical wrist
without joint limits)

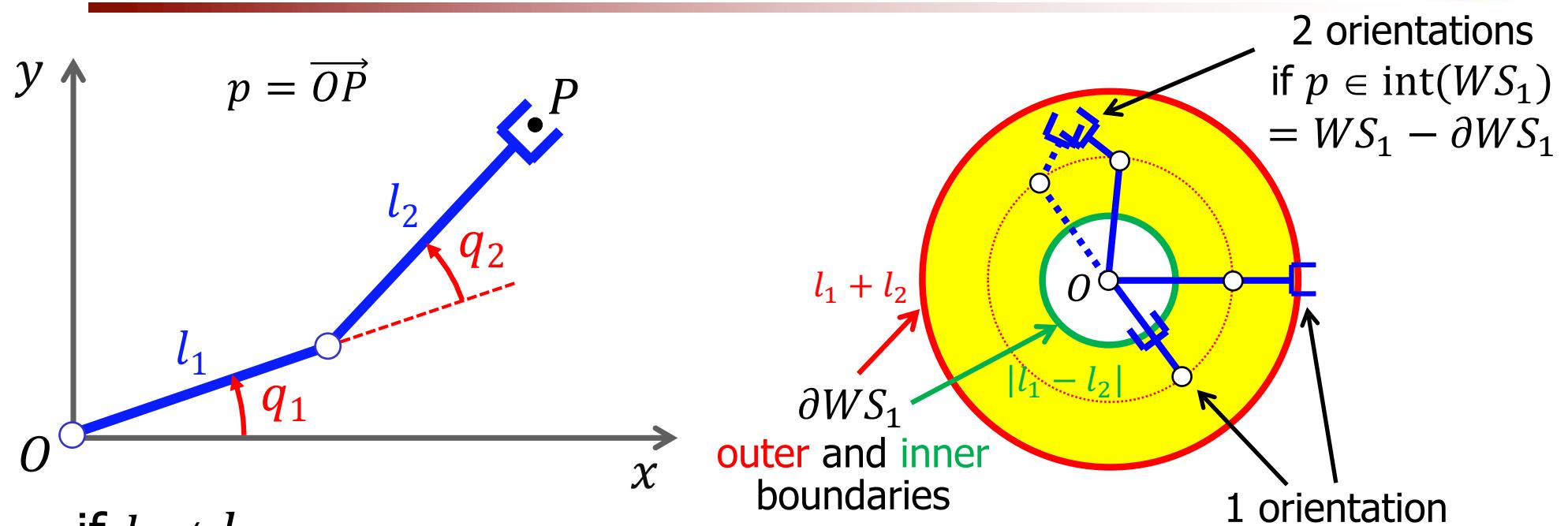
Side View



rotating
the
base joint angle q_1



Workspace of a planar 2R arm

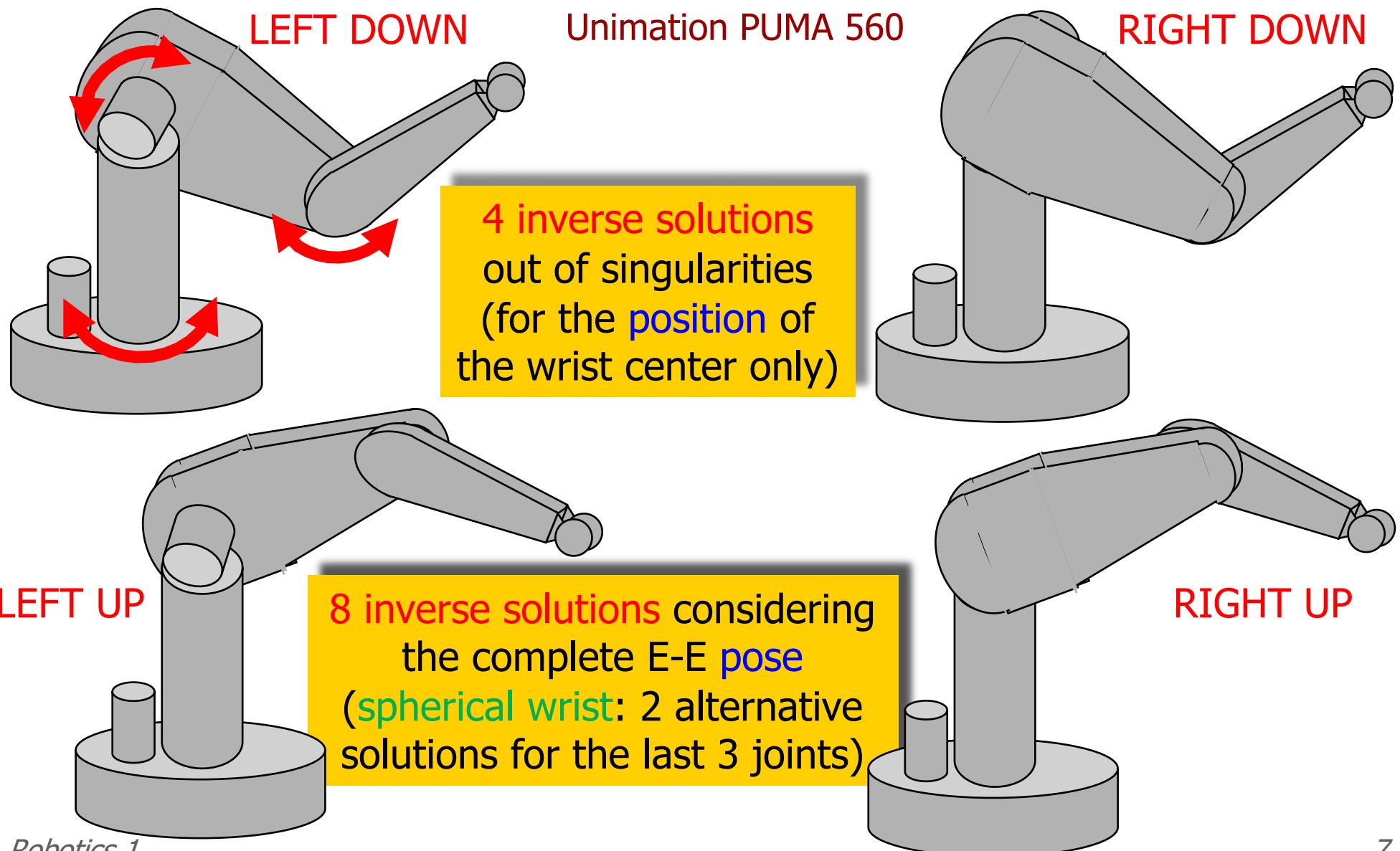


- if $l_1 \neq l_2$
 - $WS_1 = \{p \in \mathbb{R}^2 : |l_1 - l_2| \leq \|p\| \leq l_1 + l_2\} \subset \mathbb{R}^2$
 - $WS_2 = \emptyset$
- if $l_1 = l_2 = l$
 - $WS_1 = \{p \in \mathbb{R}^2 : \|p\| \leq 2l\} \subset \mathbb{R}^2$
 - $WS_2 = \{p = 0\}$ (all **feasible** orientations at the origin!... an **infinite** number)



Wrist position and E-E pose

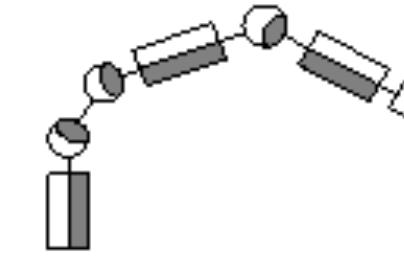
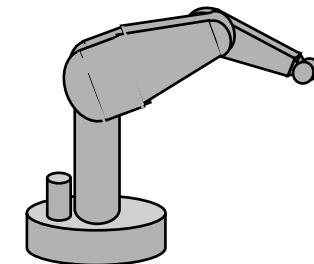
inverse solutions for an articulated 6R robot



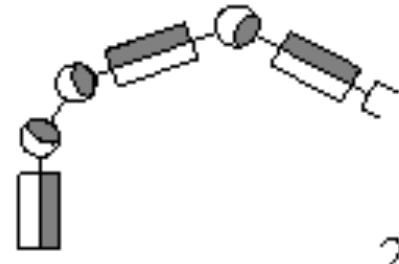


Counting/visualizing the 8 solutions of the inverse kinematics for a Unimation Puma 560

RIGHT UP

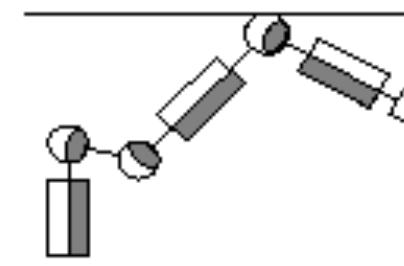
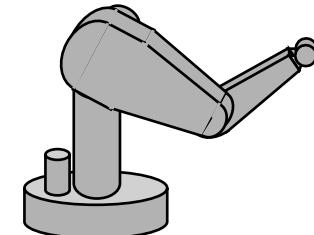


1

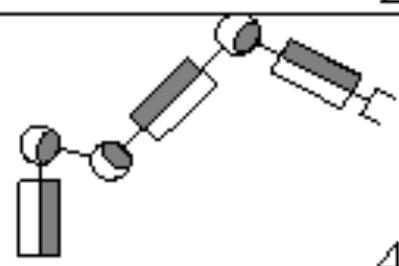


2

RIGHT DOWN

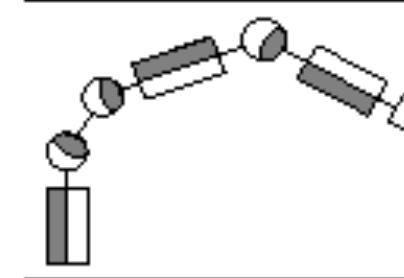
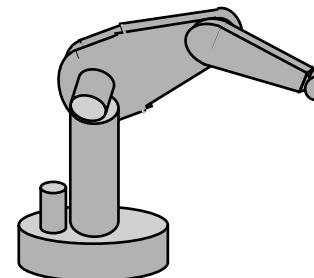


3

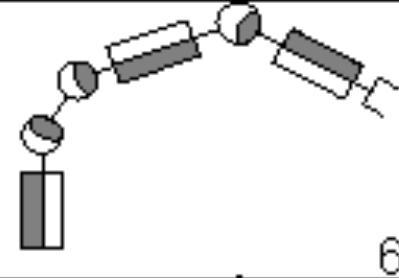


4

LEFT UP

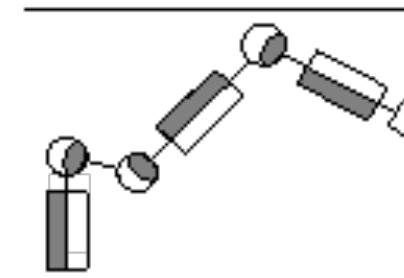
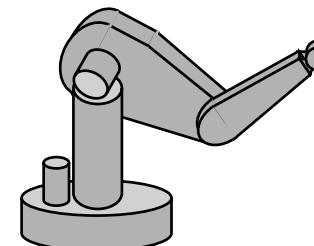


5

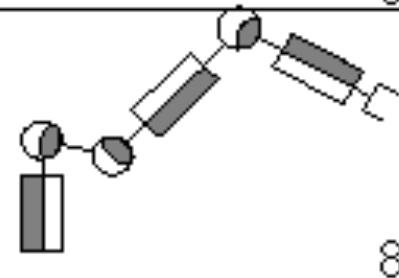


6

LEFT DOWN



7



8



Inverse kinematic solutions of UR10 6-dof Universal Robot UR10, with non-spherical wrist



video (slow motion)

desired pose

$$\mathbf{p} = \begin{pmatrix} -0.2373 \\ -0.0832 \\ 1.3224 \end{pmatrix} [\text{m}]$$

$$\mathbf{R} = \begin{pmatrix} \sqrt{3}/2 & 0.5 & 0 \\ -0.5 & \sqrt{3}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

home configuration at start

$$\mathbf{q} = (0 \quad -\pi/2 \quad 0 \quad -\pi/2 \quad 0 \quad 0)^T \quad [\text{rad}]$$





8 inverse kinematic solutions of UR10



shoulderRight
wristDown
elbowUp

$$q = \begin{pmatrix} 1.0472 \\ -1.2833 \\ -0.7376 \\ -2.6915 \\ -1.5708 \\ 3.1416 \end{pmatrix}$$



shoulderRight
wristDown
elbowDown

$$q = \begin{pmatrix} 1.0472 \\ -1.9941 \\ 0.7376 \\ 2.8273 \\ -1.5708 \\ 3.1416 \end{pmatrix}$$



shoulderRight
wristUp
elbowUp

$$q = \begin{pmatrix} 1.0472 \\ -1.5894 \\ -0.5236 \\ 0.5422 \\ 1.5708 \\ 0 \end{pmatrix}$$



shoulderRight
wristUp
elbowDown

$$q = \begin{pmatrix} 1.0472 \\ -2.0944 \\ 0.5236 \\ 0 \\ 1.5708 \\ 0 \end{pmatrix}$$



shoulderLeft
wristDown
elbowDown

$$q = \begin{pmatrix} 2.7686 \\ -1.0472 \\ -0.5236 \\ 3.1416 \\ -1.5708 \\ 1.4202 \end{pmatrix}$$



shoulderLeft
wristDown
elbowUp

$$q = \begin{pmatrix} 2.7686 \\ -1.5522 \\ 0.5236 \\ 2.5994 \\ -1.5708 \\ 1.4202 \end{pmatrix}$$



shoulderLeft
wristUp
elbowDown

$$q = \begin{pmatrix} 2.7686 \\ -1.1475 \\ -0.7376 \\ 0.3143 \\ 1.5708 \\ -1.7214 \end{pmatrix}$$



shoulderLeft
wristUp
elbowUp

$$q = \begin{pmatrix} 2.7686 \\ -1.8583 \\ 0.7376 \\ -0.4501 \\ 1.5708 \\ -1.7214 \end{pmatrix}$$



Multiplicity of solutions

few examples

- E-E positioning ($m = 2$) of a planar 2R robot
 - 2 **regular** solutions in $\text{int}(WS_1)$
 - 1 solution on ∂WS_1
 - for $l_1 = l_2$: ∞ solutions in WS_2
- } **singular** solutions
- E-E positioning ($m = 3$) of an elbow-type spatial 3R robot
 - 4 **regular** solutions in WS_1 (with **singular** cases yet to be investigated ...)
- spatial 6R robot arms
 - **≤ 16 distinct solutions**, out of singularities: this “upper bound” of solutions was shown to be attained by a particular instance of “orthogonal” robot, i.e., with twist angles $\alpha_i = 0$ or $\pm\pi/2$ ($\forall i$)
 - analysis based on **algebraic transformations** of robot kinematics
 - transcendental equations are transformed into a single polynomial equation in one variable (number of roots = degree of the polynomial)
 - seek for a transformed polynomial equation of the least possible degree



Algebraic transformations

whiteboard ...

start with some **trigonometric equation** in the joint angle θ to be solved ...

$$a \sin \theta + b \cos \theta = c \quad (*)$$

introduce the algebraic transformation (... and the related inverse formulas)

$$u = \tan(\theta/2)$$

$$\Rightarrow \sin \theta = \frac{2u}{1+u^2} \quad \cos \theta = \frac{1-u^2}{1+u^2} \quad (\Rightarrow \sin^2 \theta + \cos^2 \theta = 1)$$

$$\tan \theta = \tan 2(\theta/2) = \frac{2 \tan(\theta/2)}{1 - \tan^2(\theta/2)} = \frac{2u}{1-u^2} \quad (\text{using the duplication formula})$$

substituting in $(*)$

$$a \frac{2u}{1+u^2} + b \frac{1-u^2}{1+u^2} = c \quad \Rightarrow \quad \text{polynomial equation of second degree in } u$$
$$(b+c)u^2 - 2au - (b-c) = 0$$

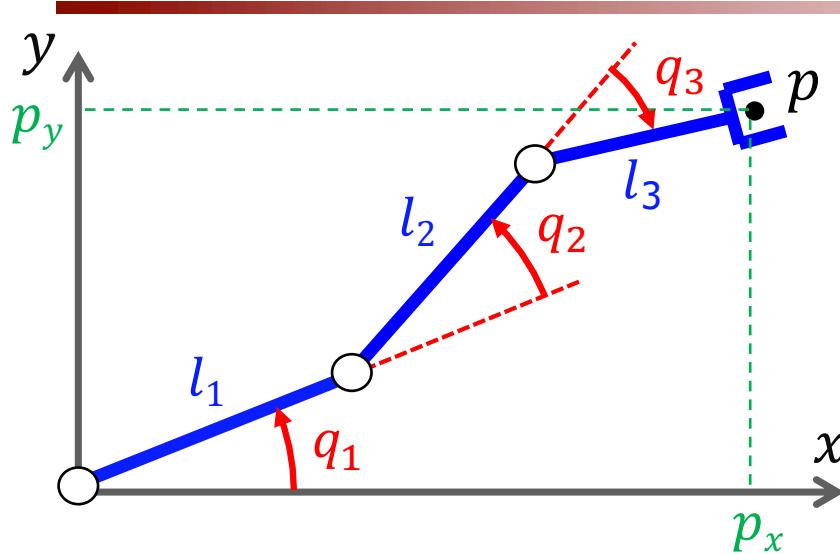
$$\Rightarrow u_{1,2} = \frac{a \pm \sqrt{a^2 + b^2 - c^2}}{b+c} \quad \Rightarrow \quad \theta_{1,2} = 2 \arctan(u_{1,2})$$

only if argument is real, else no solution



A planar 3R arm

workspace and number/type of inverse solutions



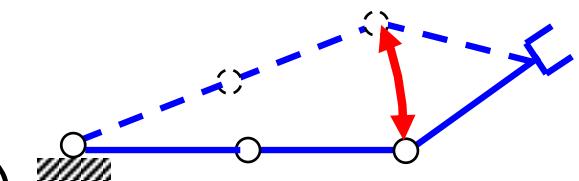
$$l_1 = l_2 = l_3 = l \quad n = 3, m = 2$$

$$WS_1 = \{p \in \mathbb{R}^2 : \|p\| \leq 3l\} \subset \mathbb{R}^2$$

$$WS_2 = \{p \in \mathbb{R}^2 : \|p\| \leq l\} \subset \mathbb{R}^2$$

any planar orientation is feasible in WS_2

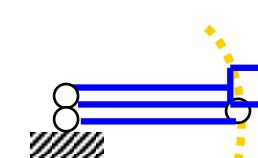
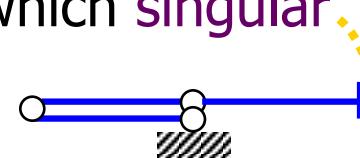
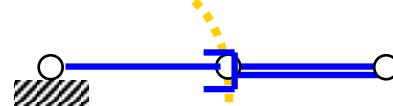
1. in $\text{int}(WS_1)$: ∞^1 **regular** (except for 3.) solutions,
at which the E-E can take a **continuum** of
 ∞ orientations (but **not all** orientations in the plane!)



2. if $\|p\| = 3l$: only 1 solution, **singular**



3. if $\|p\| = l$: ∞^1 solutions, 3 of which **singular**



4. if $\|p\| < l$: ∞^1 **regular** solutions (that are **never singular**)



Workspace of a planar 3R arm with generic link lengths

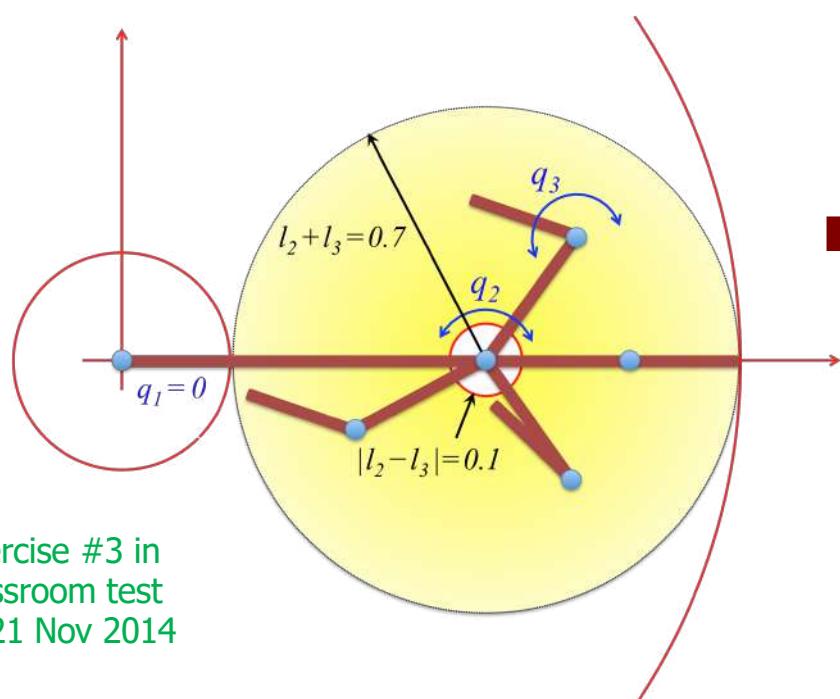
$$l_{max} = \max \{l_i, i = 1, 2, 3\}$$

$$l_{min} = \min \{l_i, i = 1, 2, 3\}$$

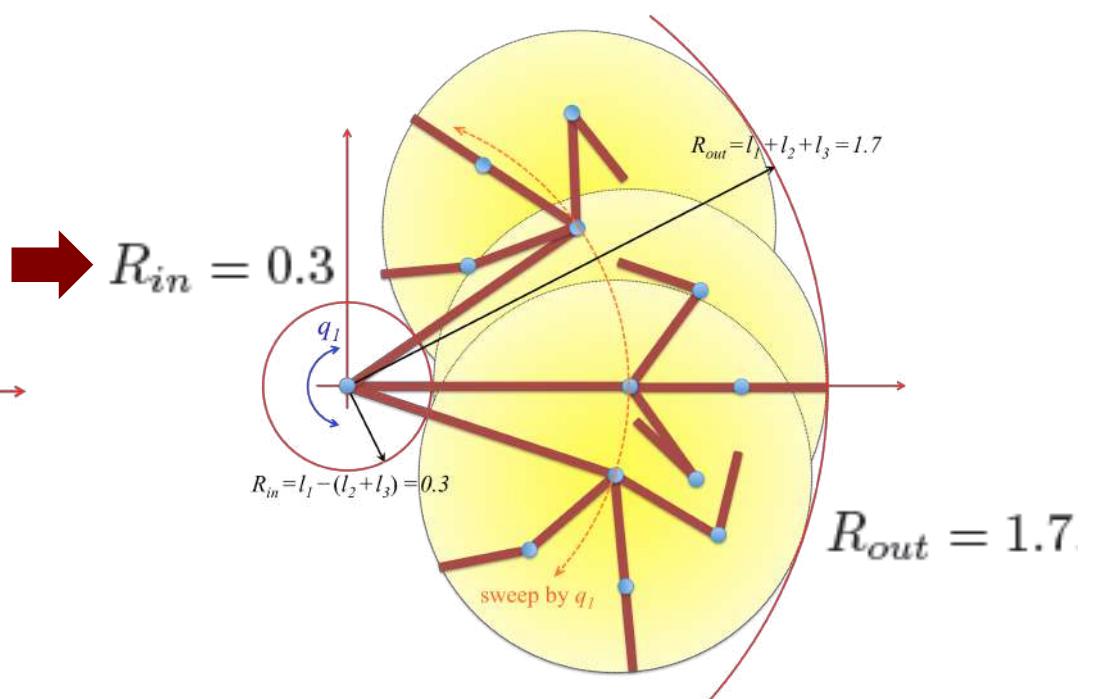
$$R_{out} = l_{min} + l_{med} + l_{max} = l_1 + l_2 + l_3$$

$$R_{in} = \max \{0, l_{max} - (l_{med} + l_{min})\}$$

a) $l_1 = 1, l_2 = 0.4, l_3 = 0.3$ [m] $\Rightarrow l_{max} = l_1 = 1, l_{med} = l_2 = 0.4, l_{min} = l_3 = 0.3$



Exercise #3 in
classroom test
of 21 Nov 2014



b) $l_1 = 0.5, l_2 = 0.7, l_3 = 0.5$ [m] $\Rightarrow l_{max} = l_2 = 0.7, l_{med} = l_{min} = l_1(\text{or } l_3) = 0.5$

$$\rightarrow R_{in} = 0, R_{out} = 1.7$$



Multiplicity of solutions

summary of the general cases

- if $m = n$
 - \emptyset solutions
 - a finite number of solutions (**regular/generic case**)
 - “degenerate” solutions: infinite or finite set, but anyway **different in number** from the generic case (**singularity**)
- if $m < n$ (robot is kinematically **redundant** for the task)
 - \emptyset solutions
 - ∞^{n-m} solutions (**regular/generic case**)
 - a finite or infinite number of **singular** solutions
- use of the term **singularity** will become clearer when dealing with differential kinematics
 - instantaneous velocity mapping from joint to task velocity
 - **lack of full rank** of the associated $m \times n$ Jacobian matrix $J(q)$



Dexter 8R robot arm

- $m = 6$ (position and orientation of E-E)
- $n = 8$ (all revolute joints)
- ∞^2 inverse kinematic solutions (redundancy degree = $n - m = 2$)

video

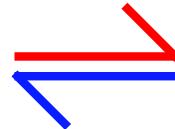


exploring inverse kinematic solutions by a robot self-motion



Solution methods

ANALYTICAL solution
(in closed form)



NUMERICAL solution
(in iterative form)

- preferred, if it can be found*
- use ad-hoc geometric inspection
- algebraic methods (solution of polynomial equations)
- systematic ways for generating a reduced set of equations to be solved

- * sufficient conditions for 6-dof arms
- 3 consecutive rotational joint axes are incident (e.g., spherical wrist), or
 - 3 consecutive rotational joint axes are parallel

D. Pieper, PhD thesis, Stanford University, 1968

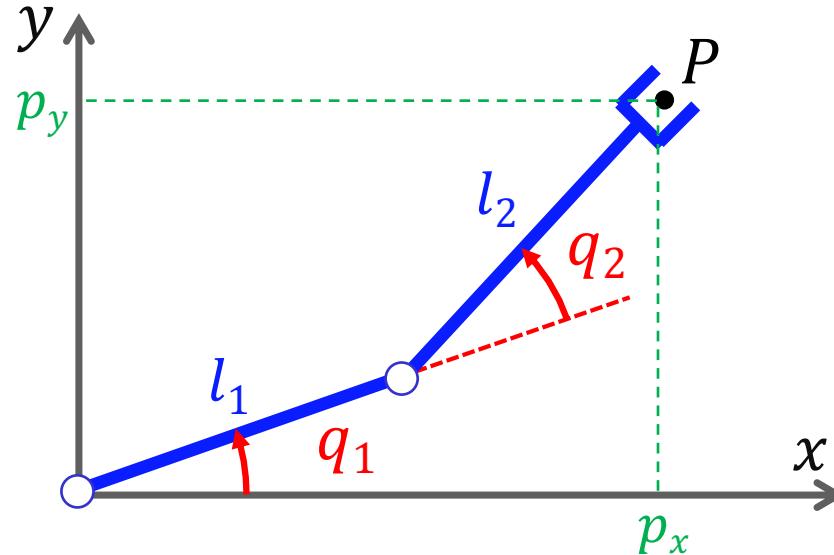
- certainly needed if $n > m$ (redundant case) or at/close to singularities
- slower, but easier to be set up
- in its basic form, it uses the (analytical) **Jacobian matrix** of the direct kinematics map

$$J_r(q) = \frac{\partial f_r(q)}{\partial q}$$

- **Newton** method, **Gradient** method, and so on...



Inverse kinematics of planar 2R arm



direct kinematics

$$p_x = l_1 c_1 + l_2 c_{12}$$

$$p_y = l_1 s_1 + l_2 s_{12}$$



data

q_1, q_2 unknowns

“squaring and summing” the equations of the direct kinematics

$$p_x^2 + p_y^2 - (l_1^2 + l_2^2) = 2l_1l_2(c_1c_{12} + s_1s_{12}) = 2l_1l_2c_2$$

and from this

$$c_2 = (p_x^2 + p_y^2 - (l_1^2 + l_2^2)) / 2l_1l_2, \quad s_2 = \pm\sqrt{1 - c_2^2}$$



must be in $[-1,1]$ (else, point P is outside robot workspace!)

$$q_2 = \text{atan2}\{s_2, c_2\}$$

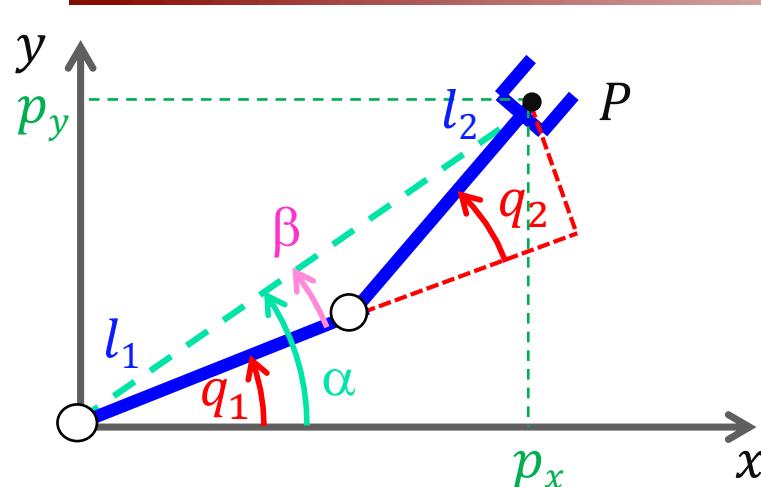
2 solutions



in analytical form



Inverse kinematics of 2R arm (cont'd)



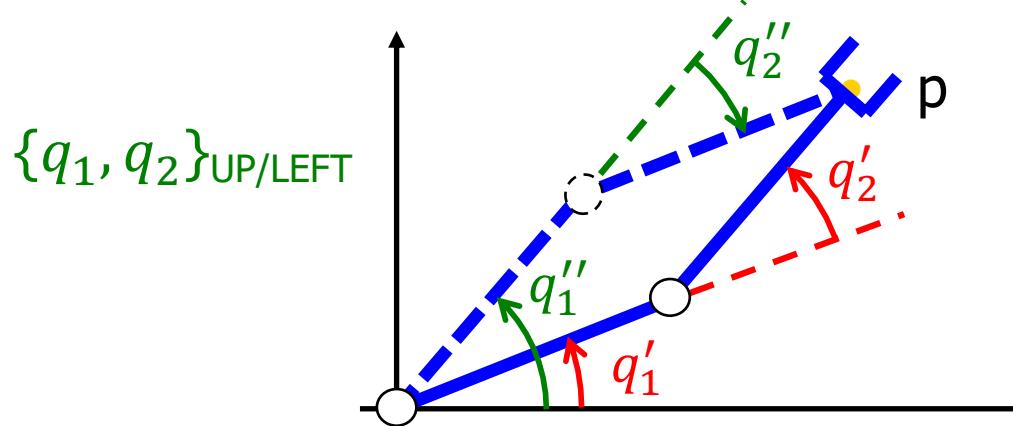
2 solutions
(one for each value of s_2)

by geometric inspection
 $q_1 = \alpha - \beta$



$$q_1 = \text{atan2}\{p_y, p_x\} - \text{atan2}\{l_2 s_2, l_1 + l_2 c_2\}$$

note: difference of atan2's needs to be re-expressed in $(-\pi, \pi]$!



$\{q_1, q_2\}_{\text{DOWN/RIGHT}}$

q'_2 and q''_2 have same absolute value, but opposite signs



Algebraic solution for q_1

another
solution
method...

$$\left. \begin{array}{l} p_x = l_1 c_1 + l_2 c_{12} = l_1 c_1 + l_2 (c_1 c_2 - s_1 s_2) \\ p_y = l_1 s_1 + l_2 s_{12} = l_1 s_1 + l_2 (s_1 c_2 + c_1 s_2) \end{array} \right\} \text{linear in } s_1 \text{ and } c_1$$

$$\underbrace{\begin{bmatrix} l_1 + l_2 c_2 & -l_2 s_2 \\ l_2 s_2 & l_1 + l_2 c_2 \end{bmatrix}}_{\det = l_1^2 + l_2^2 + 2l_1 l_2 c_2 > 0} \begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

except if $l_1 = l_2$ and $c_2 = -1$
being then q_1 undefined
(singular case: ∞^1 solutions)

$$q_1 = \text{atan2}\{s_1, c_1\}$$

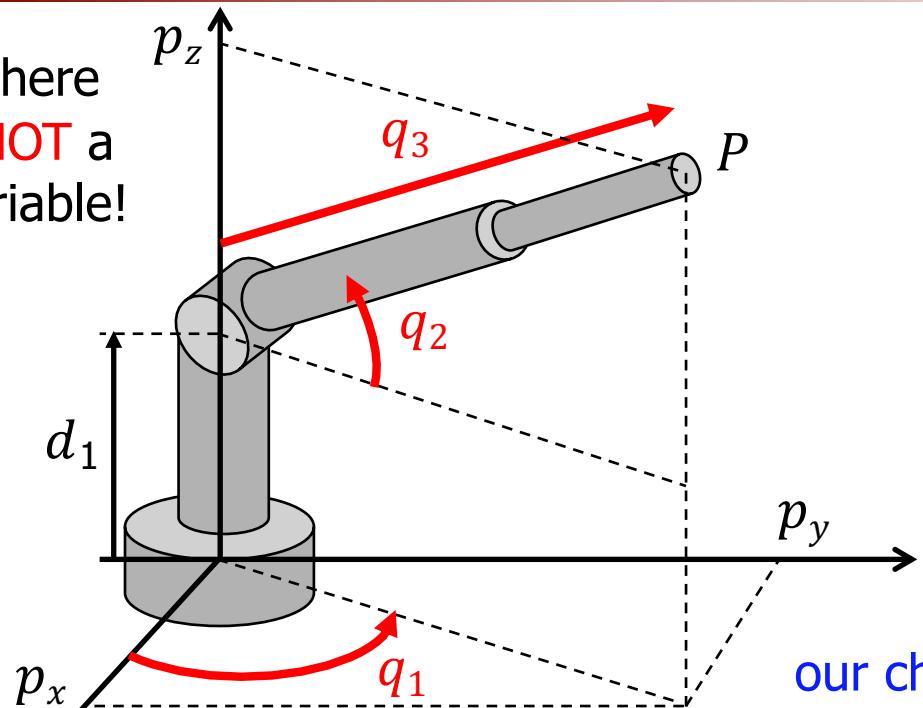
$$= \text{atan2}\{(p_y(l_1 + l_2 c_2) - p_x l_2 s_2)/\det, (p_x(l_1 + l_2 c_2) + p_y l_2 s_2)/\det\}$$

notes: a) this method provides directly the result in $(-\pi, \pi]$
b) when evaluating atan2, $\det > 0$ can be in fact eliminated
from the expressions of s_1 and c_1 (not changing the result)



Inverse kinematics of polar (RRP) arm

note: here
 q_2 is NOT a
DH variable!



direct
kinematics

$$p_x = q_3 c_2 c_1$$

$$p_y = q_3 c_2 s_1$$

$$p_z = d_1 + q_3 s_2$$

$$p_x^2 + p_y^2 + (p_z - d_1)^2 = q_3^2$$

$$q_3 = + \sqrt{p_x^2 + p_y^2 + (p_z - d_1)^2}$$

our choice: take here only the positive value...

if $q_3 = 0$, then q_1 and q_2 remain both undefined (stop); else

$$q_2 = \text{atan2} \left\{ (p_z - d_1)/q_3, \pm \sqrt{p_x^2 + p_y^2}/q_3 \right\}$$

(if we stop, it is
a singular case:
 ∞^2 or ∞^1
solutions)

if $p_x^2 + p_y^2 = 0$, then q_1 remains undefined (stop); else

$$q_1 = \text{atan2} \left\{ p_y/c_2, p_x/c_2 \right\}$$

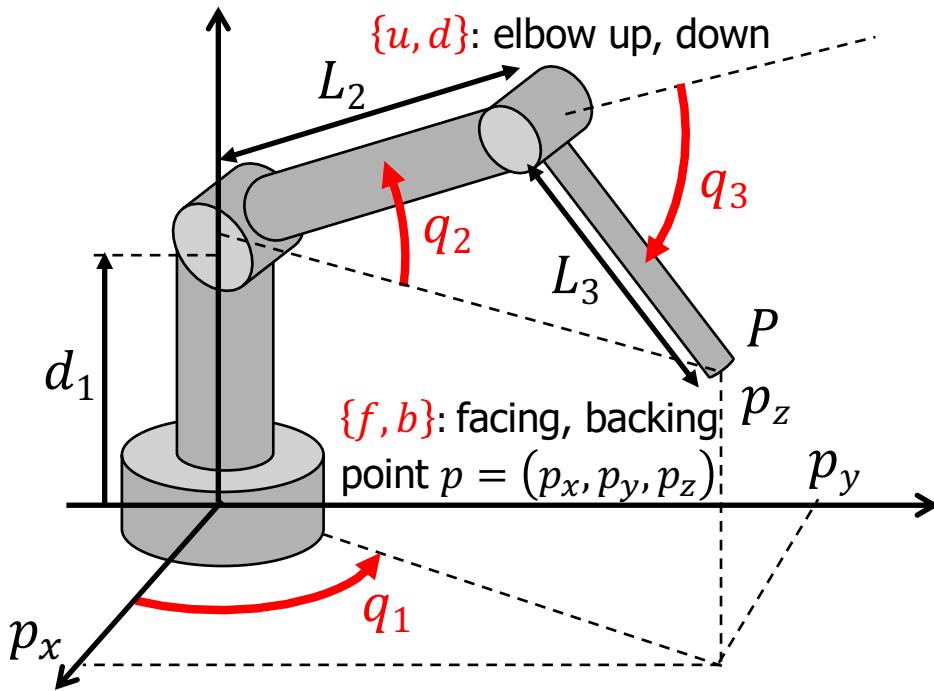
(2 regular solutions $\{q_1, q_2, q_3\}$)



eliminating $q_3 > 0$ from both arguments 21



Inverse kinematics of 3R elbow-type arm



symmetric structure **without** offsets
e.g., first 3 joints of Mitsubishi PA10 robot

$WS_1 = \{\text{spherical shell centered at } (0,0,d_1),$
 $\text{with outer radius } R_{out} = L_2 + L_3$
 $\text{and inner radius } R_{in} = |L_2 - L_3|\}$



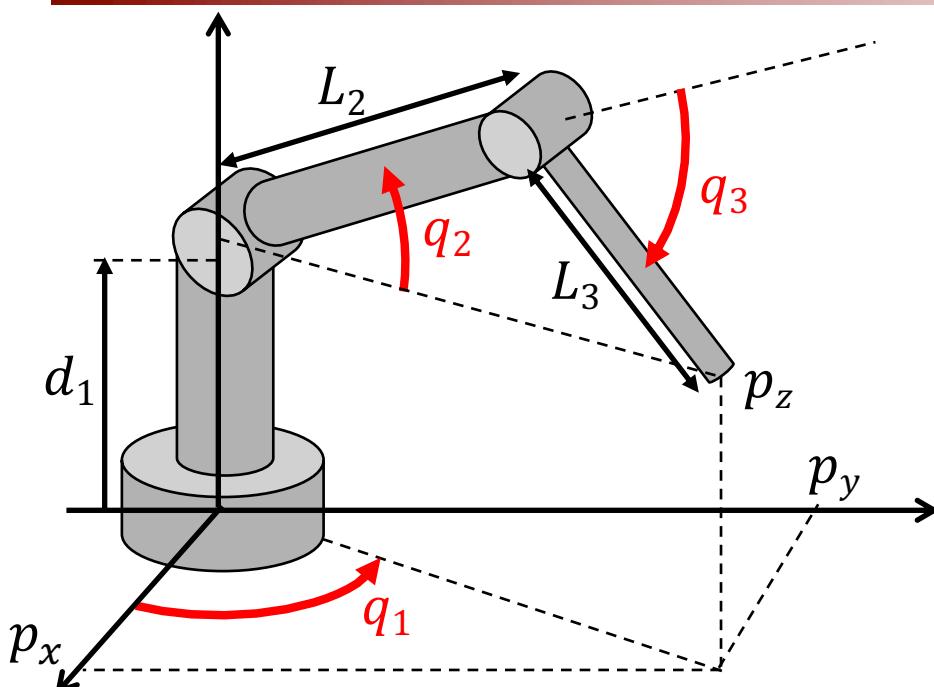
4 **regular** inverse
kinematics solutions in WS_1

more details (e.g., full handling of singular cases)
can be found in the solution of Exercise #1
in written exam of 11 Apr 2017



Inverse kinematics of 3R elbow-type arm

step 1



direct
kinematics

$$p_x = c_1(L_2 c_2 + L_3 c_{23})$$

$$p_y = s_1(L_2 c_2 + L_3 c_{23})$$

$$p_z = d_1 + L_2 s_2 + L_3 s_{23}$$

$$\begin{aligned} p_x^2 + p_y^2 + (p_z - d_1)^2 &= c_1^2(L_2 c_2 + L_3 c_{23})^2 + c_1^2(L_2 c_2 + L_3 c_{23})^2 + (L_2 s_2 + L_3 s_{23})^2 \\ &= \dots = L_2^2 + L_3^2 + 2L_2 L_3(c_2 c_{23} + s_2 s_{23}) = L_2^2 + L_3^2 + 2L_2 L_3 c_3 \end{aligned}$$

$$c_3 = (p_x^2 + p_y^2 + (p_z - d_1)^2 - L_2^2 - L_3^2) / 2L_2 L_3 \in [-1, +1] \text{ (else, } p \text{ is out of workspace!)}$$

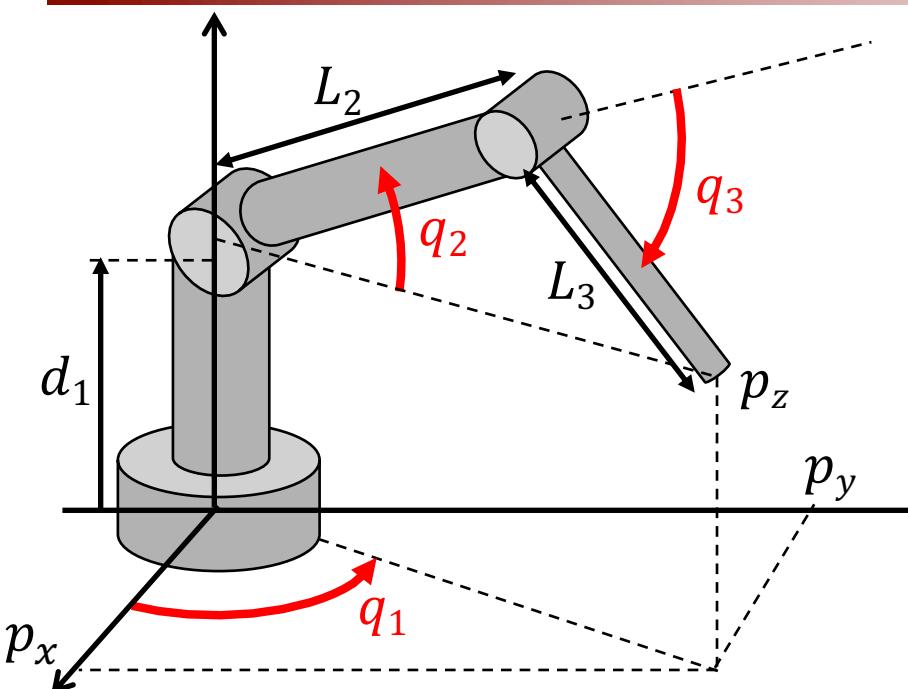
$$\downarrow$$

$$\pm s_3 = \pm \sqrt{1 - c_3^2} \quad \rightarrow \text{two solutions} \quad \left\{ \begin{array}{l} q_3^{(+)} = \text{atan2}\{s_3, c_3\} \\ q_3^{(-)} = \text{atan2}\{-s_3, c_3\} = -q_3^{(+)} \end{array} \right.$$



Inverse kinematics of 3R elbow-type arm

step 2



direct
kinematics

$$p_x = c_1(L_2 c_2 + L_3 c_{23})$$

$$p_y = s_1(L_2 c_2 + L_3 c_{23})$$

$$p_z = d_1 + L_2 s_2 + L_3 s_{23}$$

... being $p_x^2 + p_y^2 = (L_2 c_2 + L_3 c_{23})^2 > 0$

only when $p_x^2 + p_y^2 > 0$...
(else q_1 is undefined —infinite solutions!)

→

$$\begin{cases} c_1 = p_x / \pm \sqrt{p_x^2 + p_y^2} \\ s_1 = p_y / \pm \sqrt{p_x^2 + p_y^2} \end{cases}$$

again, two solutions

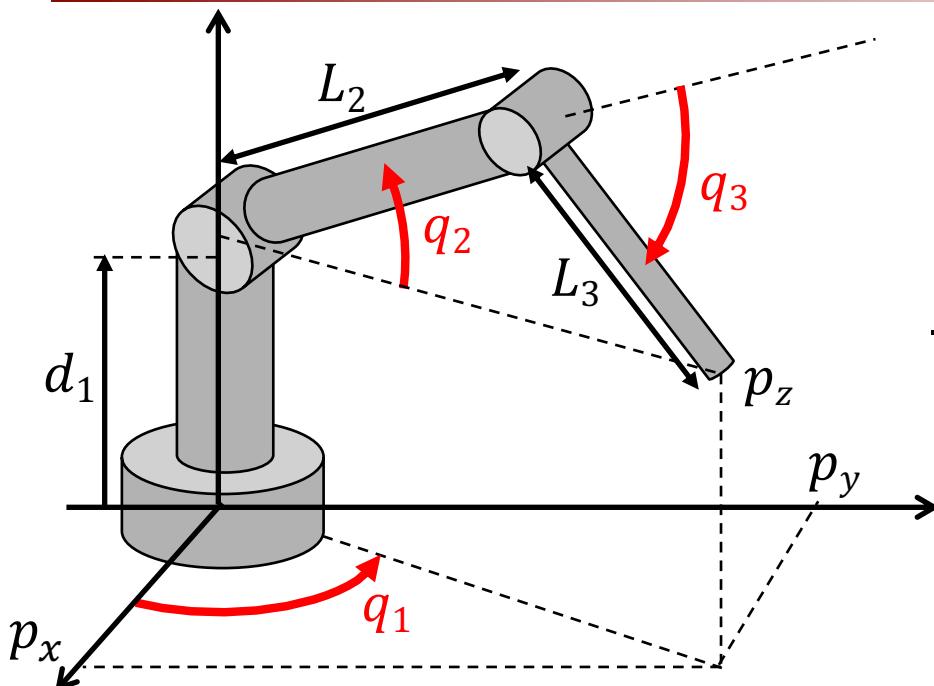


$$\begin{cases} q_1^{(+)} = \text{atan2}\{p_y, p_x\} \\ q_1^{(-)} = \text{atan2}\{-p_y, -p_x\} \end{cases}$$



Inverse kinematics of 3R elbow-type arm

step 3



$$\begin{bmatrix} L_2 + L_3 c_3 & -L_3 s_3^{\{+,-\}} \\ L_3 s_3^{\{+,-\}} & L_2 + L_3 c_3 \end{bmatrix} \begin{bmatrix} c_2 \\ s_2 \end{bmatrix} = \begin{bmatrix} c_1^{\{+,-\}} p_x + s_1^{\{+,-\}} p_y \\ p_z - d_1 \end{bmatrix}$$

coefficient matrix A

known vector b

provided $\det A = p_x^2 + p_y^2 + (p_z - d_1)^2 \neq 0$

(else q_2 is undefined —infinite solutions!)

combine first the two equations of direct kinematics and rearrange the last one

$$\left\{ \begin{array}{l} c_1 p_x + s_1 p_y = L_2 c_2 + L_3 c_{23} \\ \quad = (L_2 + L_3 c_3) c_2 - L_3 s_3 s_2 \\ p_z - d_1 = L_2 s_2 + L_3 s_{23} \\ \quad = L_3 s_3 c_2 + (L_2 + L_3 c_3) s_2 \end{array} \right.$$

define and solve a **linear system** $Ax = b$
in the **algebraic** unknowns $x = (c_2, s_2)$

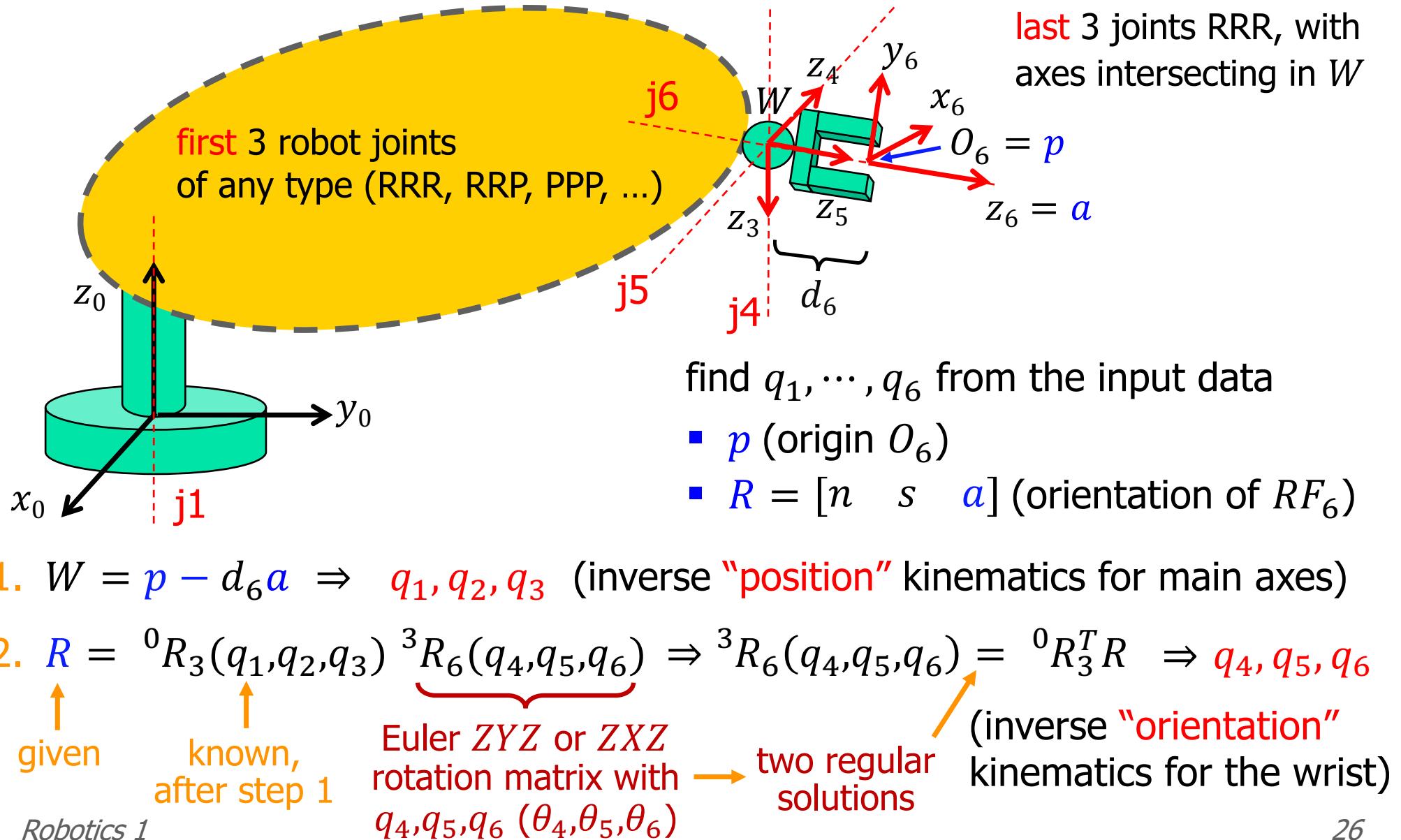
4 **regular** solutions for q_2 ,
depending on the combinations
of $\{+,-\}$ from q_1 and q_3

↓

$$q_2^{\{\{f,b\},\{u,d\}\}} = \text{atan2} \left\{ s_2^{\{\{f,b\},\{u,d\}\}}, c_2^{\{\{f,b\},\{u,d\}\}} \right\}$$



Inverse kinematics for robots with spherical wrist





6R robot Unimation PUMA 600

spherical
wrist

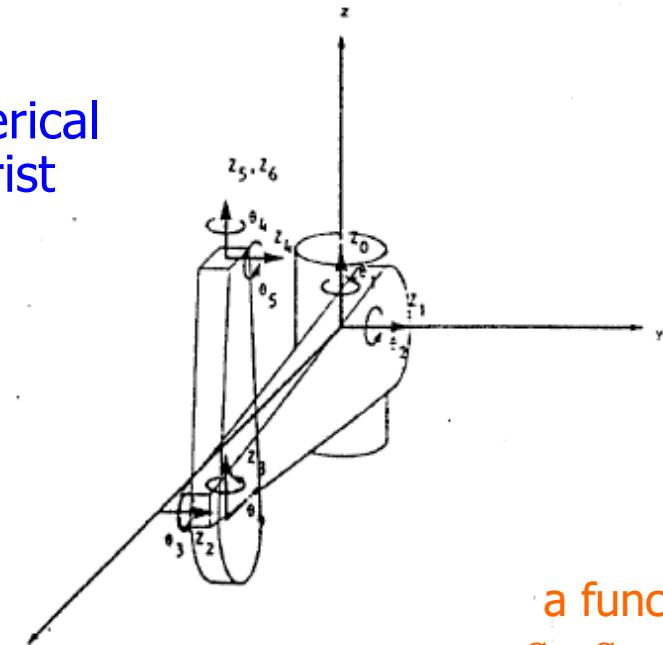


TABLE I
LINK PARAMETERS FOR PUMA ARM

Joint	a°	θ°	d	a	Range
1	-90°	θ_1	0	0	$\theta_1: +/- 160^\circ$
2	0	θ_2	0	a_2	$\theta_2: +45^\circ \rightarrow -225^\circ$
3	90°	θ_3	d_3	a_3	$\theta_3: 225^\circ \rightarrow -45^\circ$
4	-90°	θ_4	d_4	0	$\theta_4: +/- 170^\circ$
5	90°	θ_5	0	0	$\theta_5: +/- 135^\circ$
6	0	θ_6	0	0	$\theta_6: +/- 170^\circ$

$a_2 = 17.000$ $a_3 = 0.75$
 $d_3 = 4.937$ $d_4 = 17.000$

here $d_6 = 0$,

so that $O_6 = W$ directly

a function of
 q_1, q_2, q_3 only!

$$\left. \begin{aligned}
 n_x &= C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] \\
 &\quad - S_1[S_4C_5C_6 + C_4S_6] \\
 n_y &= S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] \\
 &\quad + C_1[S_4C_5C_6 + C_4S_6] \\
 n_z &= -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6 \\
 o_x &= C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] \\
 &\quad - S_1[-S_4C_5S_6 + C_4C_6] \\
 o_y &= S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] \\
 &\quad + C_1[-S_4C_5S_6 + C_4C_6] \\
 o_z &= S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6 \\
 a_x &= C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5 \\
 a_y &= S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5 \\
 a_z &= -S_{23}C_4S_5 + C_{23}C_5
 \end{aligned} \right\} \begin{aligned}
 n &= {}^0x_6(q) \\
 o &= {}^0y_6(q) \\
 a &= {}^0z_6(q) \\
 p &= O_6(q)
 \end{aligned}$$

$p_x = C_1(d_4S_{23} + a_3C_{23} + a_2C_2) - S_1d_3$
 $p_y = S_1(d_4S_{23} + a_3C_{23} + a_2C_2) + C_1d_3$
 $p_z = -(-d_4C_{23} + a_3S_{23} + a_2S_2)$.

8 different (regular) inverse solutions
that can be found in closed form



Finding nice kinematic relations

whiteboard ...

- the most complex inverse kinematics that could be solved in principle in closed form (i.e., **analytically**) is that of a **6R serial manipulator**, with arbitrary DH table
 - ways to systematically generate equations from the direct kinematics that could be easier to solve \Rightarrow some scalar equations may contain perhaps **a single unknown variable!**

method used for the
Unimation PUMA 600 in (*)

$$\begin{aligned} {}^0T_6 &= {}^0A_1(\theta_1) {}^1A_2(\theta_2) \cdots {}^5A_6(\theta_6) = U_0 \\ {}^0A_1^{-1} {}^0T_6 &= U_1 (= {}^1A_2 \cdots {}^5A_6) \\ {}^1A_2^{-1} {}^0A_1^{-1} {}^0T_6 &= U_2 (= {}^2A_3 \cdots {}^5A_6) \\ &\dots \\ {}^4A_5^{-1} \cdots {}^1A_2^{-1} {}^0A_1^{-1} {}^0T_6 &= U_5 (= {}^5A_6) \end{aligned}$$

or also ...

$$\begin{aligned} {}^0T_6 {}^5A_6^{-1} &= V_5 (= {}^1A_2 \cdots {}^4A_5) \\ {}^0T_6 {}^5A_6^{-1} {}^4A_5^{-1} &= V_4 (= {}^1A_2 \cdots {}^3A_4) \\ &\dots \\ {}^0T_6 {}^5A_6^{-1} {}^4A_5^{-1} \cdots {}^1A_2^{-1} &= V_1 (= {}^0A_1) \end{aligned}$$

(*) Paul, Shimano, and Mayer: IEEE Transactions on Systems, Man, and Cybernetics, 1981

- generating from the direct kinematics a reduced set of equations to be solved (setting w.l.o.g. $d_1 = d_6 = 0$) \Rightarrow **4 compact scalar equations** in the 4 unknowns $\theta_2, \dots, \theta_5$

$${}^0T_6 = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0A_6(\theta) \longrightarrow \begin{aligned} a_z &= a^T(\theta) z & \|p\|^2 &= p^T(\theta) p(\theta) && \text{solved analytically or numerically ...} \\ p_z &= p^T(\theta) z & p^T a &= p^T(\theta) a(\theta) \end{aligned}$$

$z = [0 \ 0 \ 1]^T$

... then solve easily for the remaining θ_1 and θ_6

Manseur and Doty: International Journal of Robotics Research, 1988

Numerical solution of inverse kinematics problems



- use when a closed-form solution q to $r_d = f_r(q)$ does not exist or is “too hard” to be found
- all methods are **iterative** and need the matrix $J_r(q) = \frac{\partial f_r(q)}{\partial q}$ (analytical Jacobian)
- **Newton method** (here only for $m = n$, at the **k th iteration**)
 - $r_d = f_r(q) = f_r(q^k) + J_r(q^k)(q - q^k) + o(\|q - q^k\|)$ ← neglected

$$q^{k+1} = q^k + J_r^{-1}(q^k) [r_d - f_r(q^k)]$$

 - convergence for q^0 (initial guess) close enough to some q^* : $f_r(q^*) = r_d$
 - problems near **singularities** of the Jacobian matrix $J_r(q)$
 - in case of robot redundancy ($m < n$), use the pseudo-inverse $J_r^\#(q)$
 - has **quadratic** convergence rate when near to a solution (fast!)



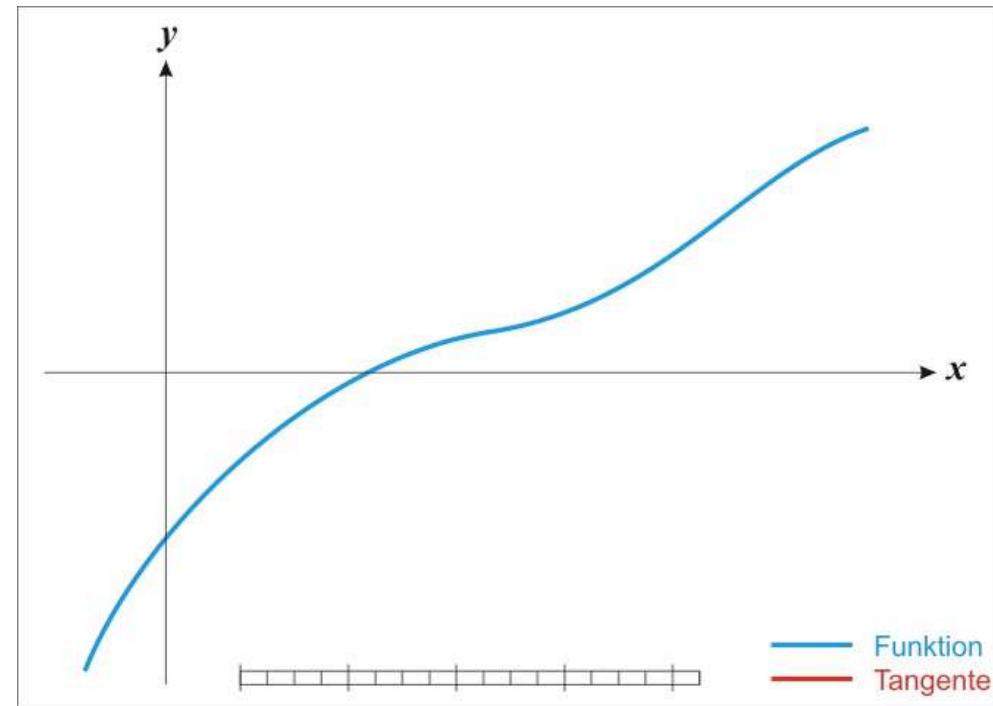
Operation of Newton method

- in the **scalar** case, also known as “method of the tangent”
- for a differentiable function $f(x)$, find a root x^* of $f(x^*) = 0$ by iterating as

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

an approximating sequence

$$\{x_1, x_2, x_3, x_4, x_5, \dots\} \rightarrow x^*$$



animation from
http://en.wikipedia.org/wiki/File:NewtonIteration_Ani.gif



Numerical solution of inverse kinematics problems (cont'd)

- Gradient method (max descent)

- minimize the **error** function

$$H(q) = \frac{1}{2} \|r_d - f_r(q)\|^2 = \frac{1}{2} (r_d - f_r(q))^T (r_d - f_r(q))$$

$$q^{k+1} = q^k - \alpha \nabla_q H(q^k)$$

from

$$\nabla_q H(q) = (\partial H(q)/\partial q)^T = - \left((r_d - f_r(q))^T (\partial f_r(q)/\partial q) \right)^T = -J_r^T(q)(r_d - f_r(q))$$

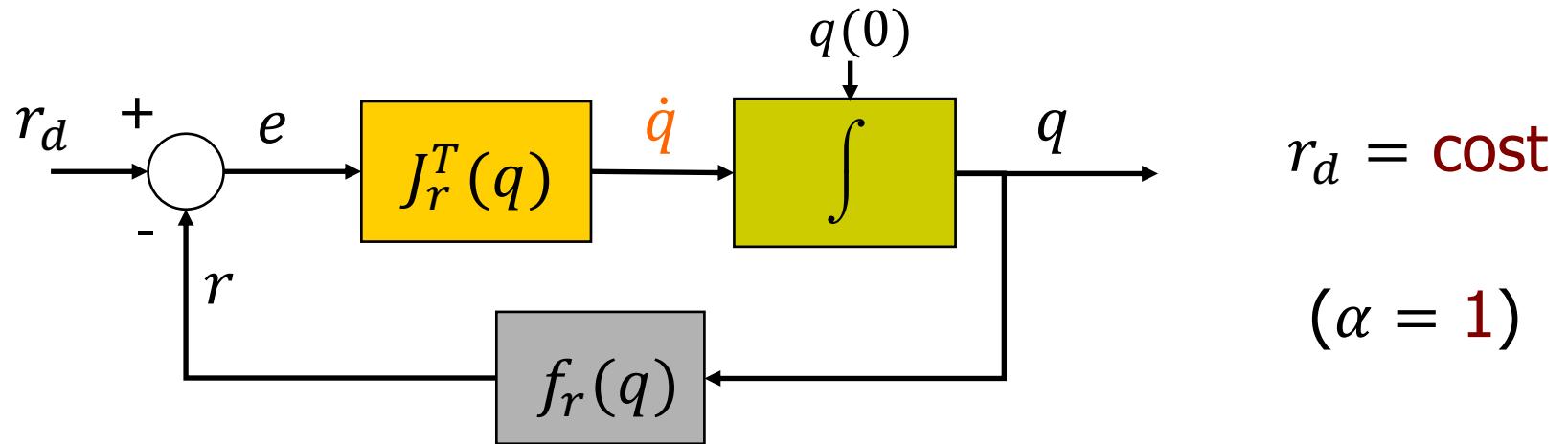
we get

$$q^{k+1} = q^k + \alpha J_r^T(q^k)(r_d - f_r(q^k))$$

- the scalar **step size** $\alpha > 0$ should be chosen so as to guarantee a decrease of the error function at each iteration: too large values for α may lead the method to “miss” the minimum
 - when the step size is too small, convergence is extremely **slow**



Revisited as a feedback scheme



$e = r_d - f_r(q) \rightarrow 0 \Leftrightarrow$ closed-loop **equilibrium** $e = 0$
is **asymptotically stable**

$V = \frac{1}{2} e^T e \geq 0$ is a **Lyapunov** candidate function

$$\dot{V} = e^T \dot{e} = e^T \frac{d}{dt} (r_d - f_r(q)) = -e^T J_r(q) \dot{q} = -e^T J_r(q) J_r^T(q) e \leq 0$$

$$\dot{V} = 0 \Leftrightarrow e \in \mathcal{N}(J_r^T(q))$$

↑
null space

in particular, $e = 0$

asymptotic stability



Properties of Gradient method

- computationally simpler: use the **Jacobian transpose**, rather than its (pseudo)-inverse
- same use also for robots that are **redundant** ($n > m$) for the task
- may not converge to a solution, but it **never diverges**
- the **discrete-time** evolution of the continuous scheme

$$q^{k+1} = q^k + \Delta T J_r^T(q^k)(r_d - f_r(q^k)), \quad \alpha = \Delta T$$

is equivalent to an iteration of the Gradient method

- the scheme can be accelerated by using a gain matrix $K > 0$

$$\dot{q} = J_r^T(q) K e = J_r^T(q) K(r_d - f_r(q))$$

note: $K \rightarrow K + K_s$, with K_s skew-symmetric, can be used also to “escape” from being stuck in a **stationary point** of $V = \frac{1}{2} e^T K e$, by **rotating** the error $K e$ out of the null space of J_r^T (when a **singularity** is encountered)



A case study

analytic expressions of Newton and gradient iterations

- 2R robot with $l_1 = l_2 = 1$, desired end-effector position $r_d = p_d = (1,1)$
- direct kinematic function and error

$$f_r(q) = \begin{pmatrix} c_1 + c_{12} \\ s_1 + s_{12} \end{pmatrix} \quad e = p_d - f_r(q) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - f_r(q)$$

- Jacobian matrix

$$J_r(q) = \frac{\partial f_r(q)}{\partial q} = \begin{pmatrix} -(s_1 + s_{12}) & -s_{12} \\ c_1 + c_{12} & c_{12} \end{pmatrix}$$

- Newton versus Gradient iteration

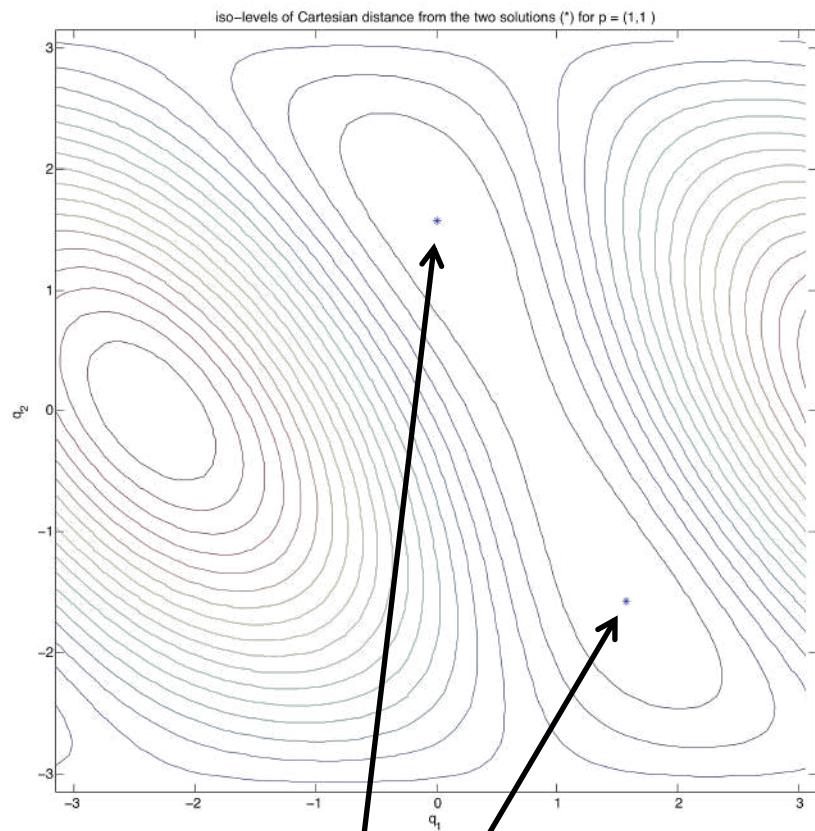
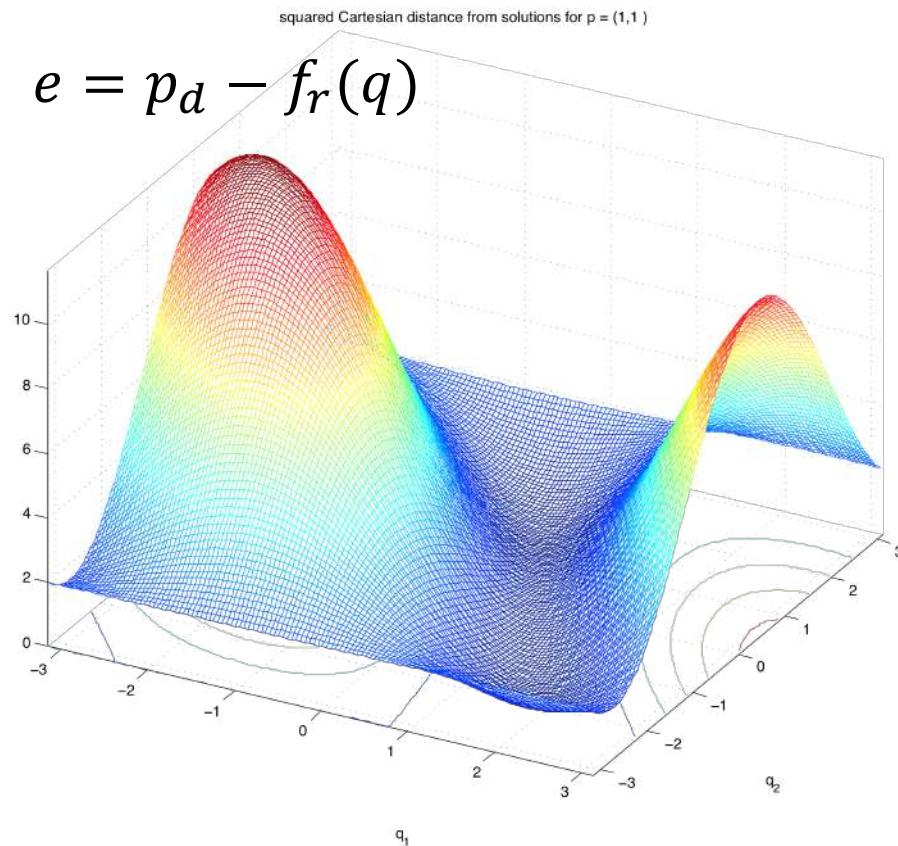
$$q^{k+1} = q^k + \left\{ \begin{array}{c} J_r^{-1}(q^k) \\ \det J_r(q) \end{array} \right. \times \left. \begin{array}{c} e_k \\ J_r^T(q^k) \end{array} \right|_{q=q^k}$$

$$q^{k+1} = q^k + \left\{ \begin{array}{c} \frac{1}{s_2} \begin{pmatrix} c_{12} & s_{12} \\ -(c_1 + c_{12}) & -(s_1 + s_{12}) \end{pmatrix}_{|q=q^k} \\ \alpha \begin{pmatrix} -(s_1 + s_{12}) & c_1 + c_{12} \\ -s_{12} & c_{12} \end{pmatrix}_{|q=q^k} \end{array} \right. \times \left. \begin{array}{c} e_k \\ J_r^T(q^k) \end{array} \right|_{q=q^k}$$



Error function

- 2R robot with $l_1 = l_2 = 1$ and desired end-effector position $p_d = (1,1)$



two local minima
(inverse kinematic solutions)

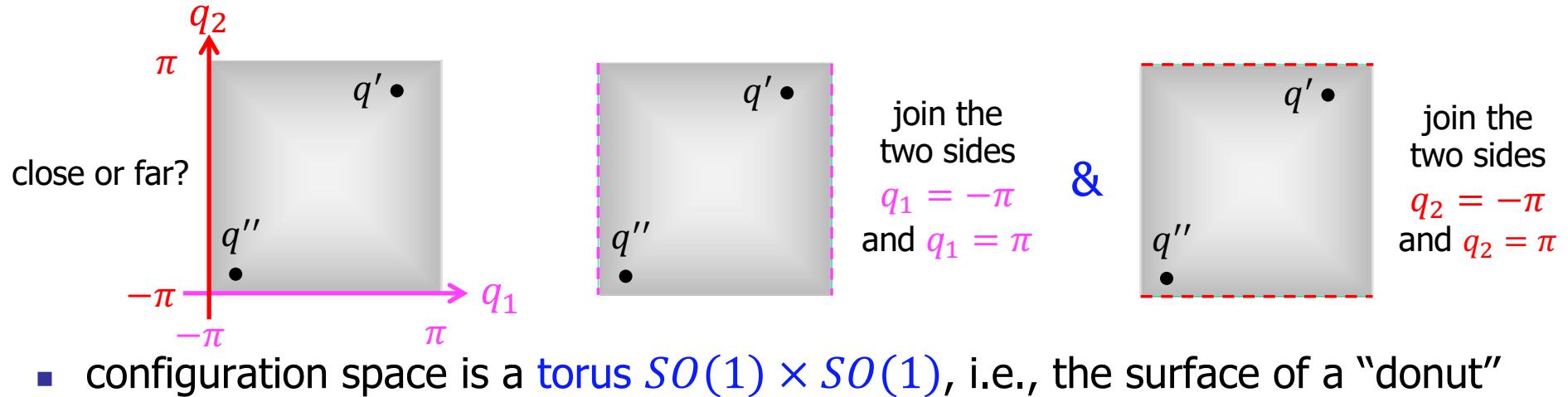
plot of $\|e\|^2$ as a function of $q = (q_1, q_2)$



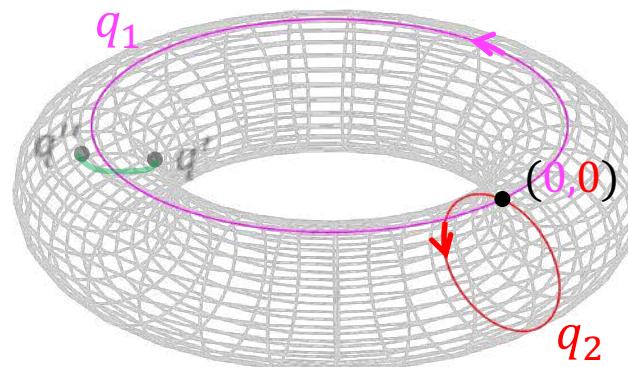
Configuration space of 2R robot

whiteboard ...

- can we represent the correct “distance” between two configurations q' and q'' of this robot on a (square) region in \mathbb{R}^2 ?



- configuration space is a torus $SO(1) \times SO(1)$, i.e., the surface of a “donut”

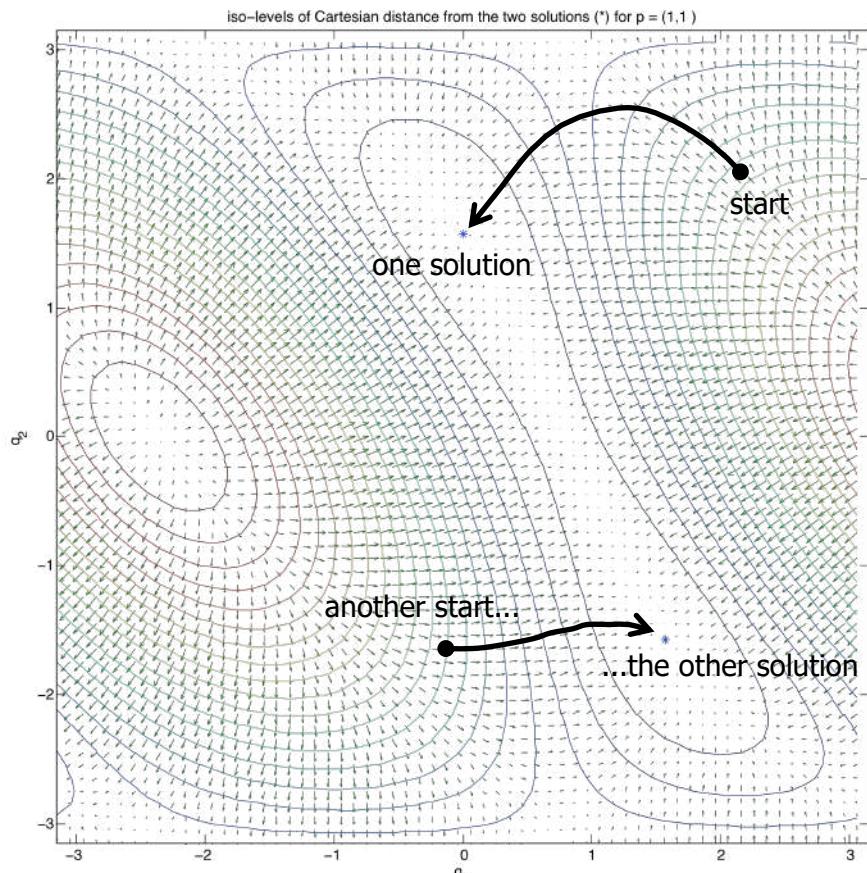


- the right metric is a **geodesic** on the torus ...



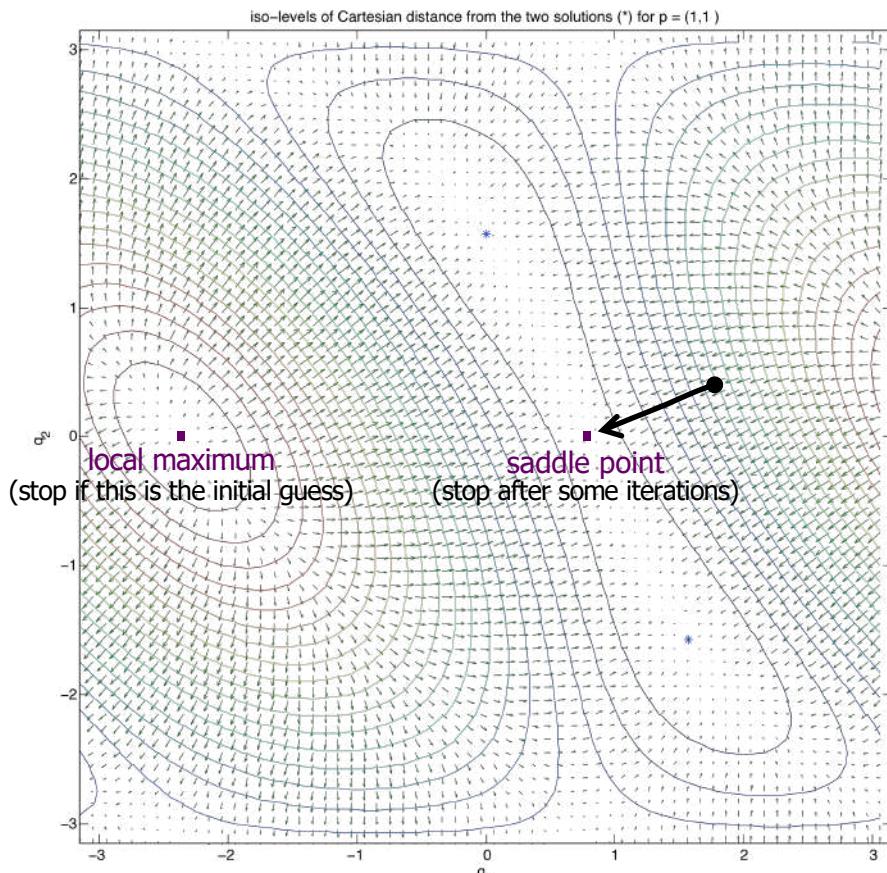
Error reduction by Gradient method

- flow of iterations along the **negative** (or anti-) gradient
- two possible cases: convergence or stuck (at **zero gradient**)



$$(q_1, q_2)' = (0, \pi/2)$$

$$(q_1, q_2)'' = (\pi/2, -\pi/2)$$



$$(q_1, q_2)_{max} = (-3\pi/4, 0) \quad (q_1, q_2)_{saddle} = (\pi/4, 0)$$

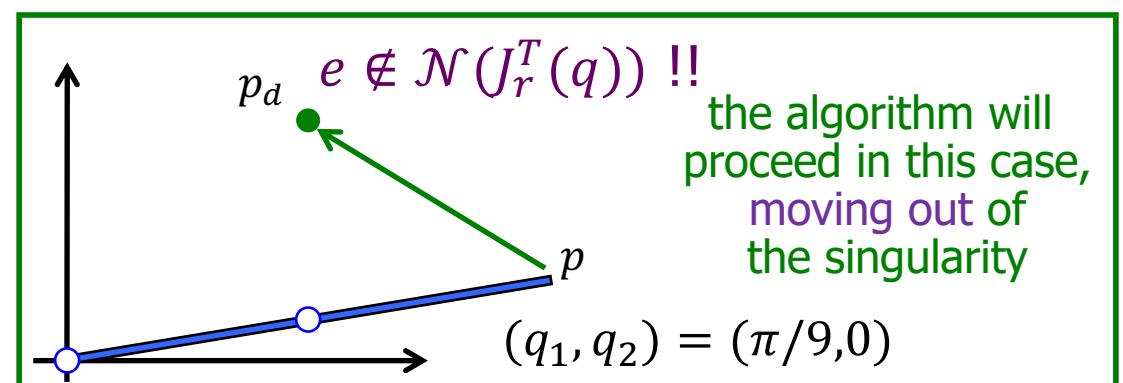
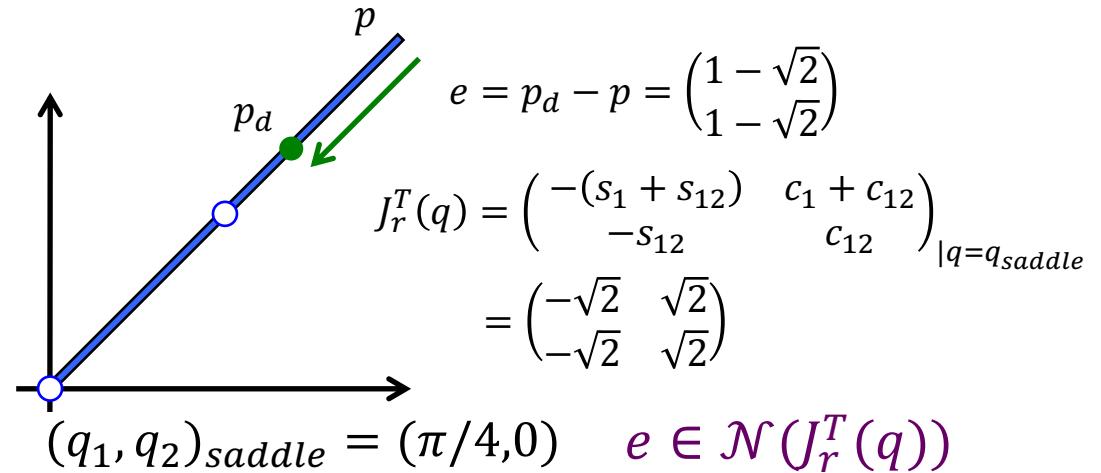
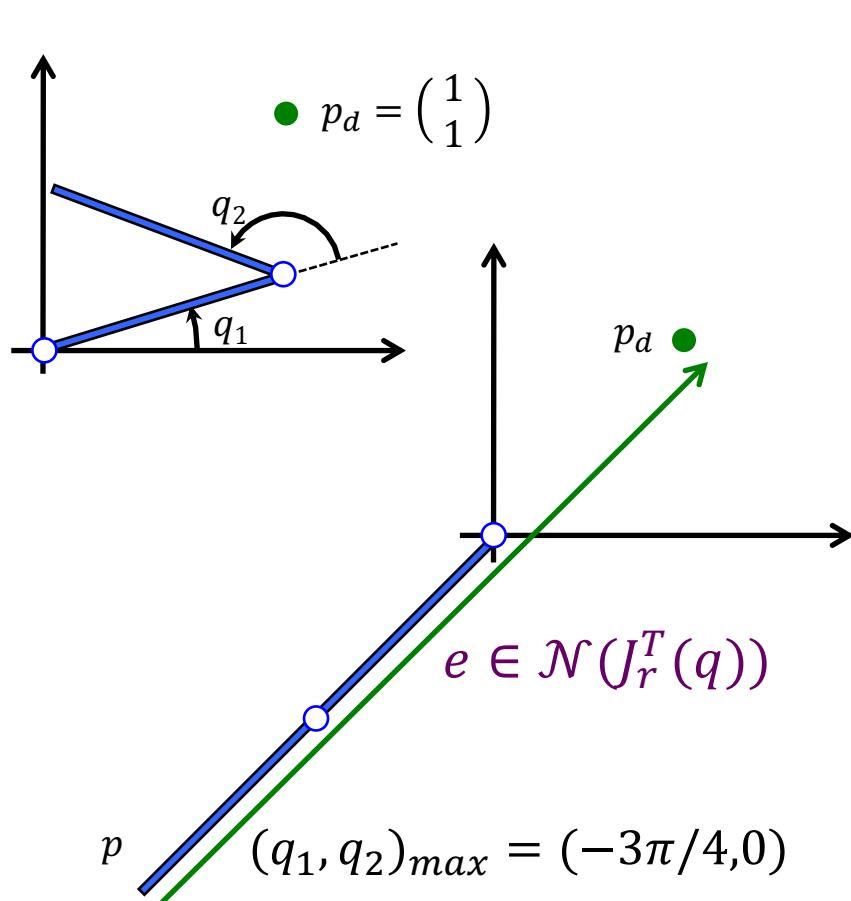
$e \in \mathcal{N}(J_r^T(q)) !$



Convergence analysis

when does the gradient method get stuck?

- lack of convergence occurs when
 - the Jacobian matrix $J_r(q)$ is not full rank (the robot is in a “singular configuration”)
 - AND** the error e is in the null space of $J_r^T(q)$





Issues in implementation

- initial guess q^0
 - only **one** inverse solution is generated for each guess
 - multiple initializations for obtaining other solutions
- optimal step size $\alpha > 0$ in Gradient method
 - a constant step may work good initially, but not close to the solution (or vice versa)
 - an **adaptive** one-dimensional line search (e.g., Armijo's rule) could be used to choose the best α at each iteration

- stopping criteria

Cartesian error
(possibly, separate for position and orientation)

$$\|r_d - f_r(q^k)\| \leq \varepsilon$$

algorithm increment

$$\|q^{k+1} - q^k\| \leq \varepsilon_q$$

- understanding closeness to singularities

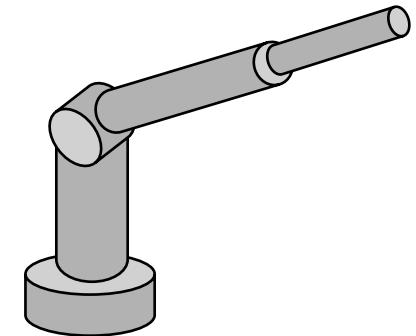
$$\sigma_{min}\{J_r(q^k)\} \geq \sigma_0$$

good numerical conditioning
of Jacobian matrix (SVD)

(or a simpler test on its determinant, for $m = n$)



Numerical tests on RRP robot

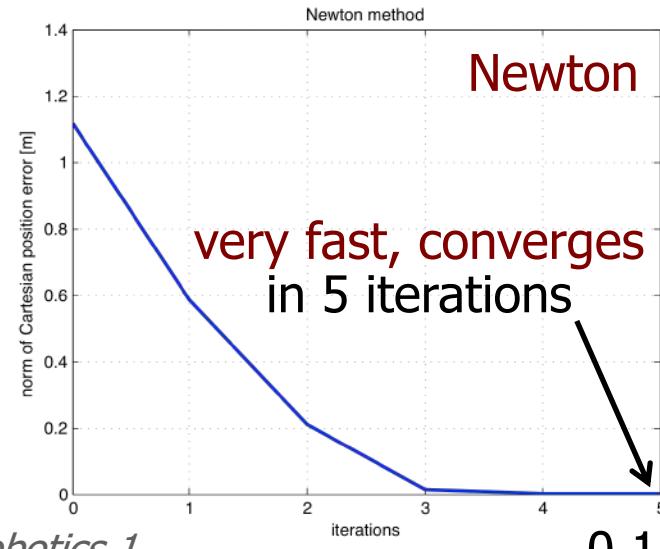
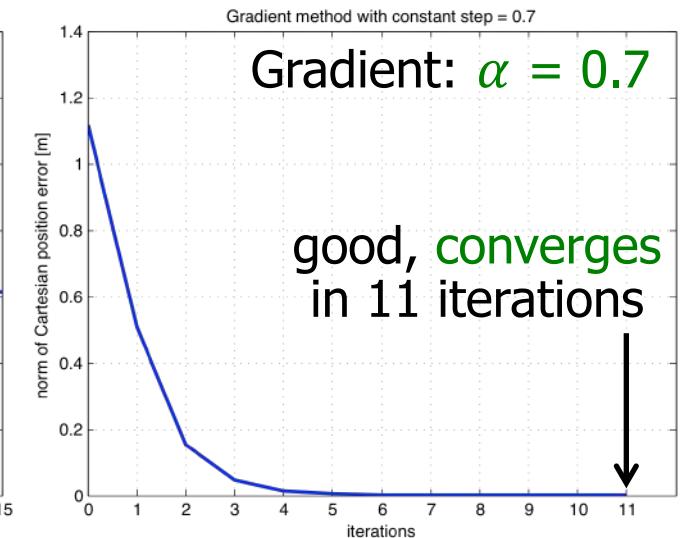
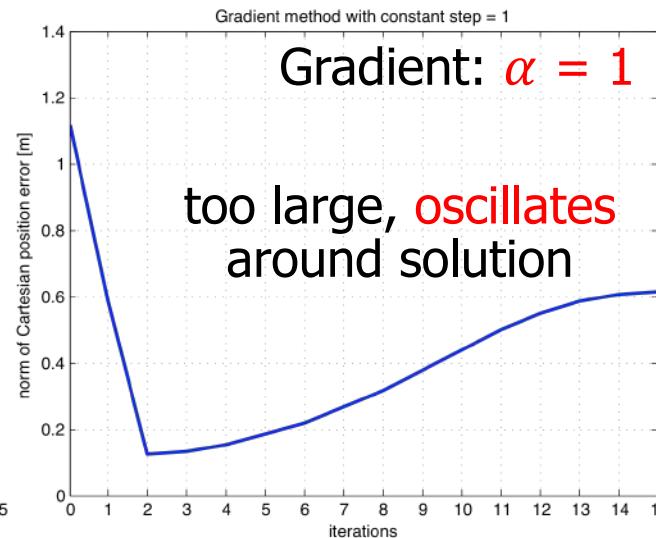
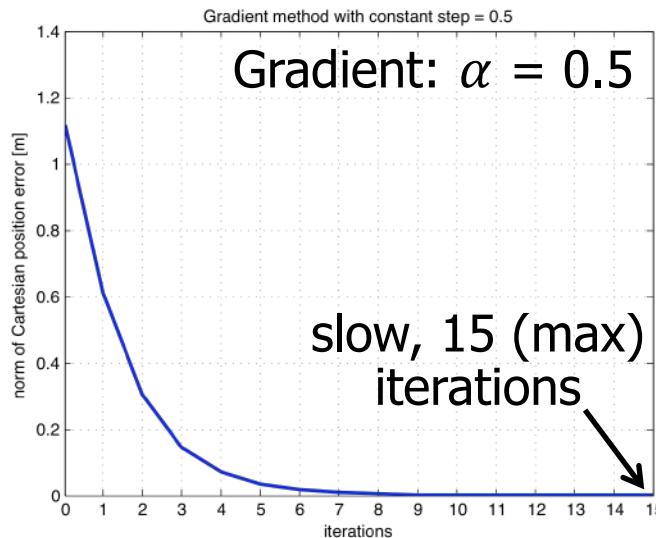


- RRP/polar robot: desired E-E position $r_d = p_d = (1, 1, 1)$
 - see slide 21, with $d_1 = 0.5$
- the two (known) analytical solutions, with $q_3 \geq 0$, are
 - $q^* = (0.7854, 0.3398, 1.5)$
 - $q^{**} = (q_1^* - \pi, \pi - q_2^*, q_3^*) = (-2.3562, 2.8018, 1.5)$
- norms $\varepsilon = 10^{-5}$ (max Cartesian error), $\varepsilon_q = 10^{-6}$ (min joint increment)
- $k_{max} = 15$ (max # iterations), $|\det J_r(q)| \leq 10^{-4}$ (singularity closeness)
- numerical performance of Gradient (with different steps α) vs. Newton
 - test 1: $q^0 = (0, 0, 1)$ as initial guess
 - test 2: $q^0 = (-\pi/4, \pi/2, 1)$ — “singular” start, since $c_2 = 0$ (see slide 21)
 - test 3: $q^0 = (0, \pi/2, 0)$ — “doubly singular” start, since also $q_3 = 0$
 - solution and plots with Matlab code

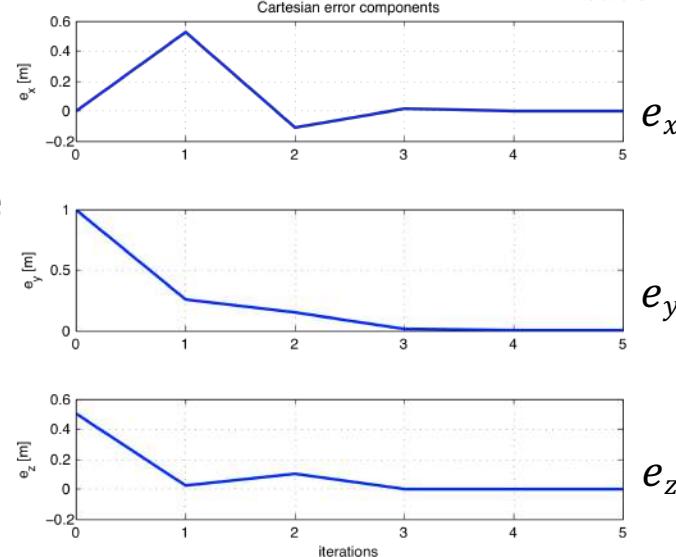


Numerical test - 1

- test 1: $q^0 = (0, 0, 1)$ as initial guess; evolution of error norm



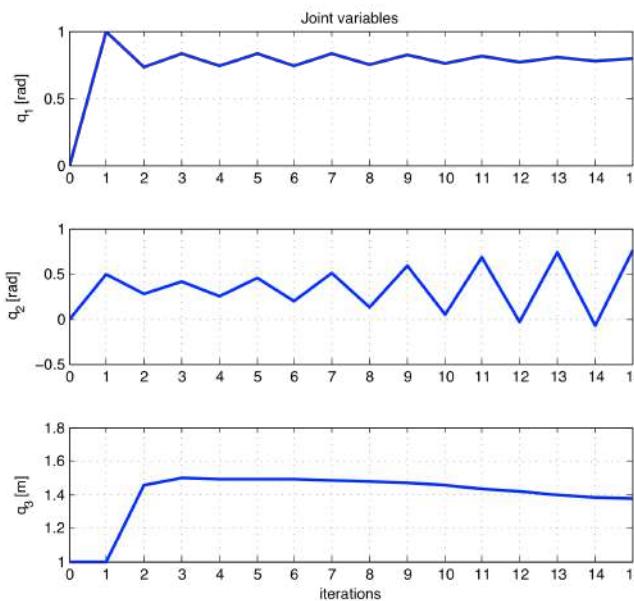
Cartesian errors component-wise





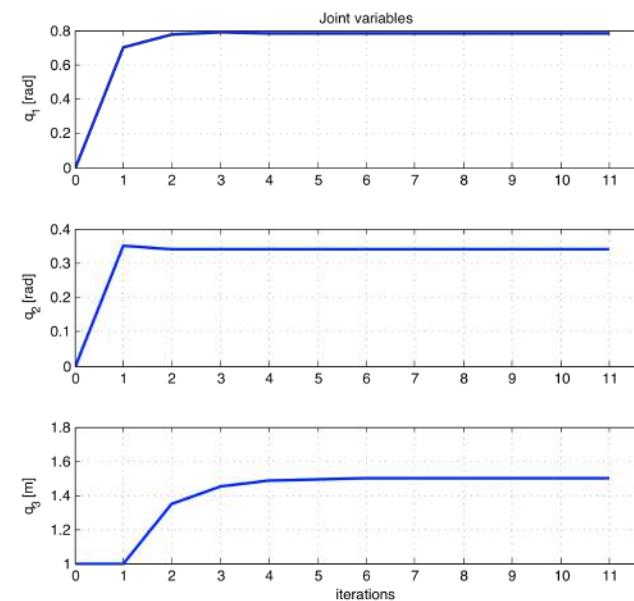
Numerical test - 1

- **test 1:** $q^0 = (0, 0, 1)$ as initial guess; evolution of joint variables



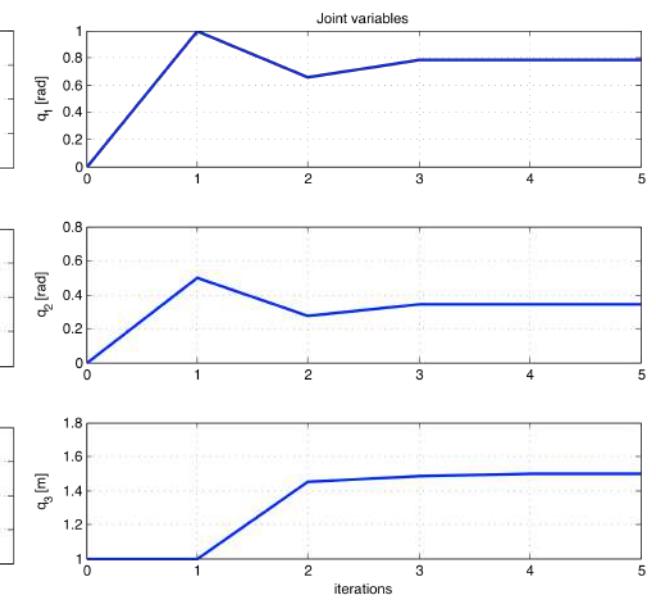
Gradient: $\alpha = 1$

not converging
to a solution



Gradient: $\alpha = 0.7$

converges in
11 iterations



Newton

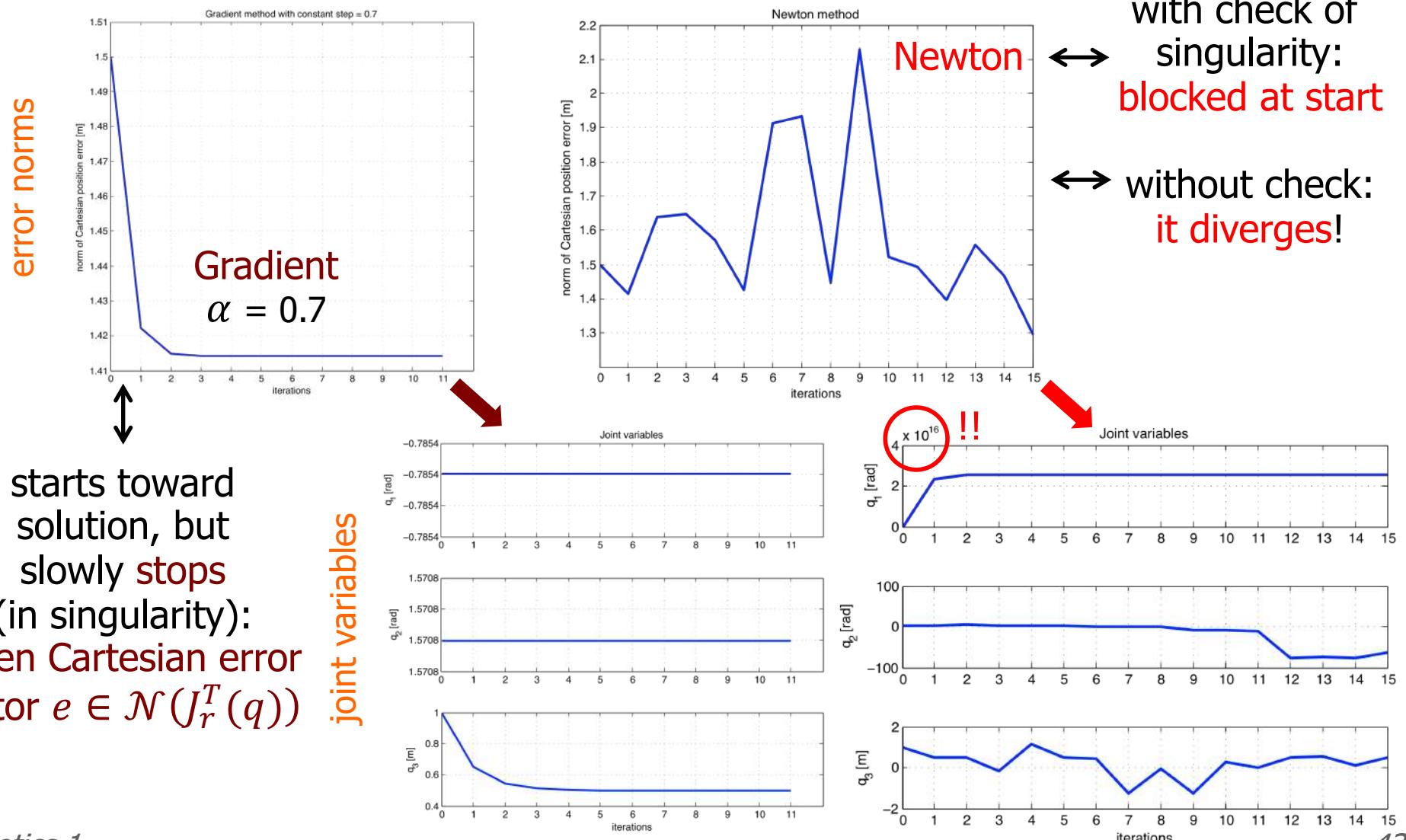
converges in
5 iterations

both to the same solution $q^* = (0.7854, 0.3398, 1.5)$



Numerical test - 2

- test 2: $q^0 = (-\pi/4, \pi/2, 1)$: singular start

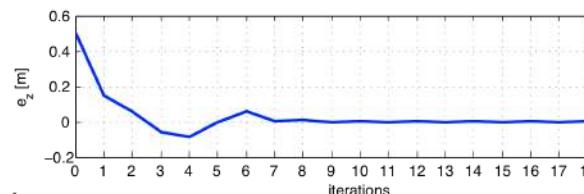
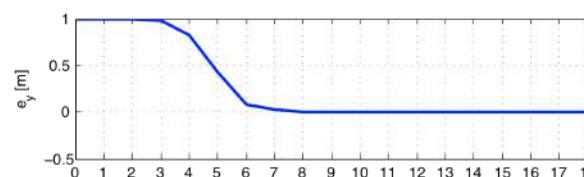
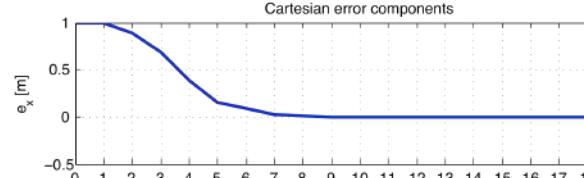
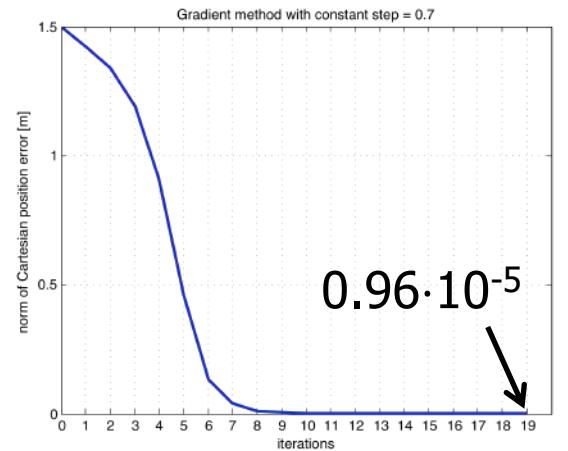




Numerical test - 3

- **test 3:** $q^0 = (-\pi/4, \pi/2, 1)$: doubly singular start

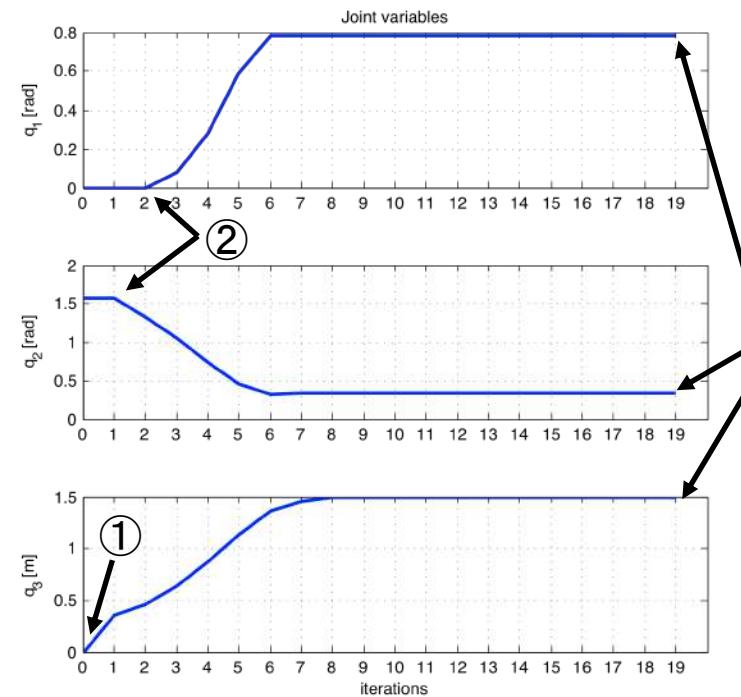
Cartesian errors



Gradient (with $\alpha = 0.7$)

- ① starts toward solution
 - ② exits the double singularity
 - ③ slowly converges in 19 iterations to the solution
- $q^* = (0.7854, 0.3398, 1.5)$

joint variables



Newton is either
blocked at start
or (w/o check)
explodes!
⇒ "NaN" in Matlab



Final remarks

- an **efficient** iterative scheme can be devised by combining
 - **initial iterations** using Gradient ("sure but slow", linear convergence rate)
 - **switch then** to Newton method (quadratic terminal convergence rate)
- **joint range limits** are considered only at the end
 - check if the solution found is **feasible**, as for analytical methods
- in alternative, an **optimization** criterion can be included in the search
 - drives iterations toward an inverse kinematic solution with nicer properties
- if the problem has to be solved **on-line**
 - execute iterations and associate an actual robot motion: **repeat steps** at times $t_0, t_1 = t_0 + T, \dots, t_k = t_{k-1} + T$ (e.g., every $T = 40$ ms)
 - a "good" choice for the initial guess q^0 at t_k is the solution of the previous problem at t_{k-1} (provides continuity, requires only 1-2 Newton iterations)
 - crossing of singularities and handling of joint range limits need special care
- Jacobian-based inversion schemes are used also for **kinematic control**, moving along a continuous task trajectory $r_d(t)$



Robotics 1

Differential kinematics

Prof. Alessandro De Luca

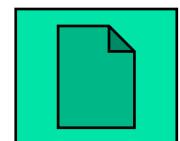
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





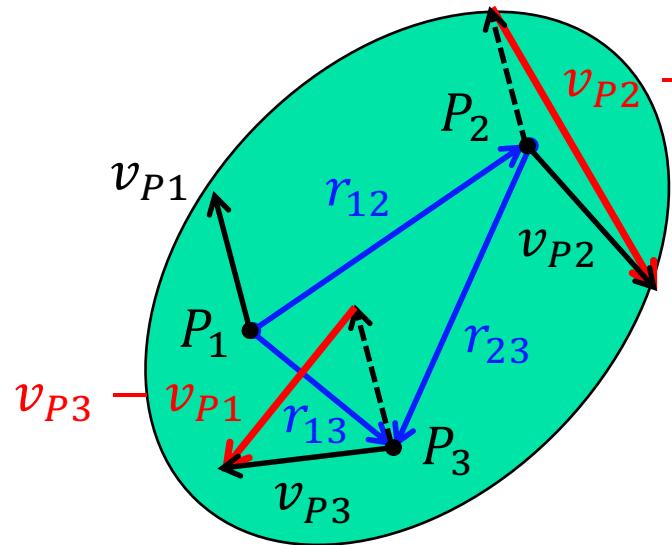
Differential kinematics

- relations between motion (velocity) in **joint** space and motion (linear/angular velocity) in **task** space (e.g., Cartesian space)
- **instantaneous** velocity mappings can be obtained through **time differentiation** of the direct kinematics or in a **geometric** way, directly at the differential level
 - different treatments arise for **rotational** quantities
 - establish the relation between **angular velocity** and
 - time **derivative** of a **rotation matrix**
 - time **derivative** of the angles in a **minimal representation** of orientation





Angular velocity of a rigid body



“rigidity” constraint on distances among points:

$$\|r_{ij}\| = \text{constant}$$

$v_{Pi} - v_{Pj}$ orthogonal to r_{ij}

1 $v_{P2} - v_{P1} = \omega_1 \times r_{12}$

2 $v_{P3} - v_{P1} = \omega_1 \times r_{13}$

3 $v_{P3} - v_{P2} = \omega_2 \times r_{23}$

$\forall P_1, P_2, P_3$

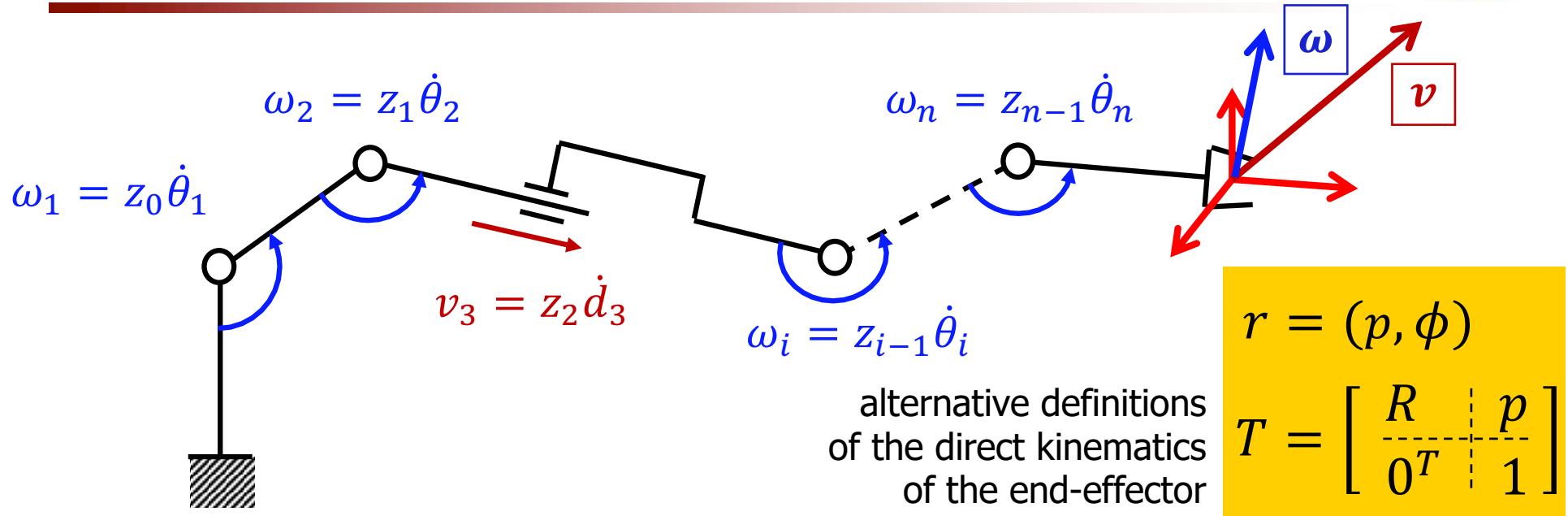
$2 - 1 = 3 \quad \rightarrow \quad \omega_1 = \omega_2 = \omega$

aka, “(fundamental)
kinematic equation”
of rigid bodies

$$v_{Pj} = v_{Pi} + \omega \times r_{ij} = v_{Pi} + S(\omega) r_{ij} \quad \leftrightarrow \quad \dot{r}_{ij} = \omega \times r_{ij}$$

- the angular velocity ω is associated to the **whole body** (**not** to a point)
- if $\exists P_1, P_2: v_{P1} = v_{P2} = 0 \Rightarrow$ **pure rotation** (circular motion of all $P_j \notin$ line P_1P_2)
- $\omega = 0 \Rightarrow$ **pure translation** (**all** points have the same velocity v_p)

Linear and angular velocity of the robot end-effector



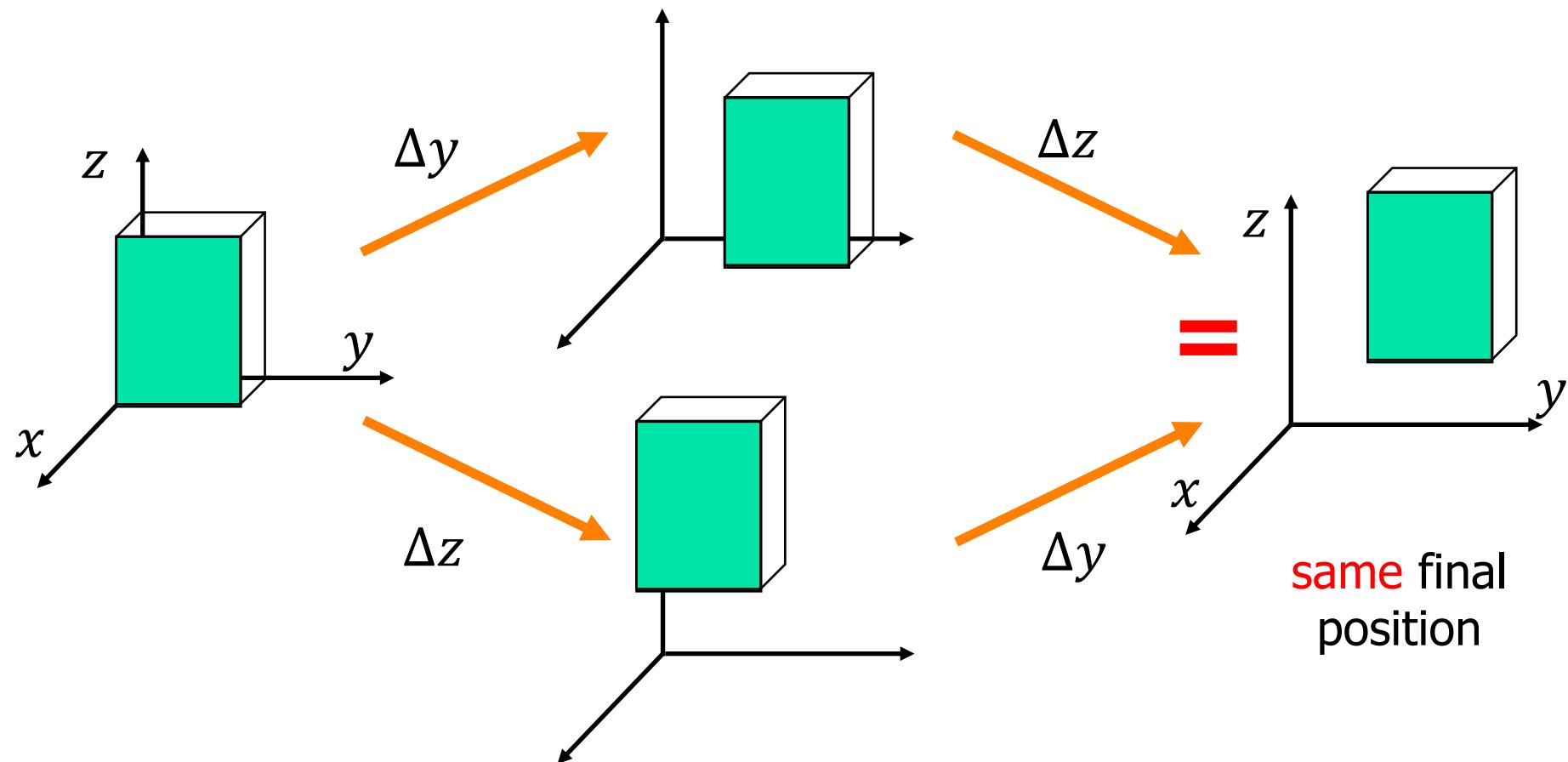
- v and ω are “vectors”, namely are elements of **vector spaces**
 - they can be obtained as the sum of single contributions (in any order)
 - such contributions will be given by the single (linear or angular) joint velocities
- on the other hand, ϕ (and $\dot{\phi}$) is **not** an element of a vector space
 - a minimal representation of a **sequence** of two rotations is **not** obtained summing the corresponding minimal representations (accordingly, for their time derivatives)

in general, $\omega \neq \dot{\phi}$



Finite and infinitesimal translations

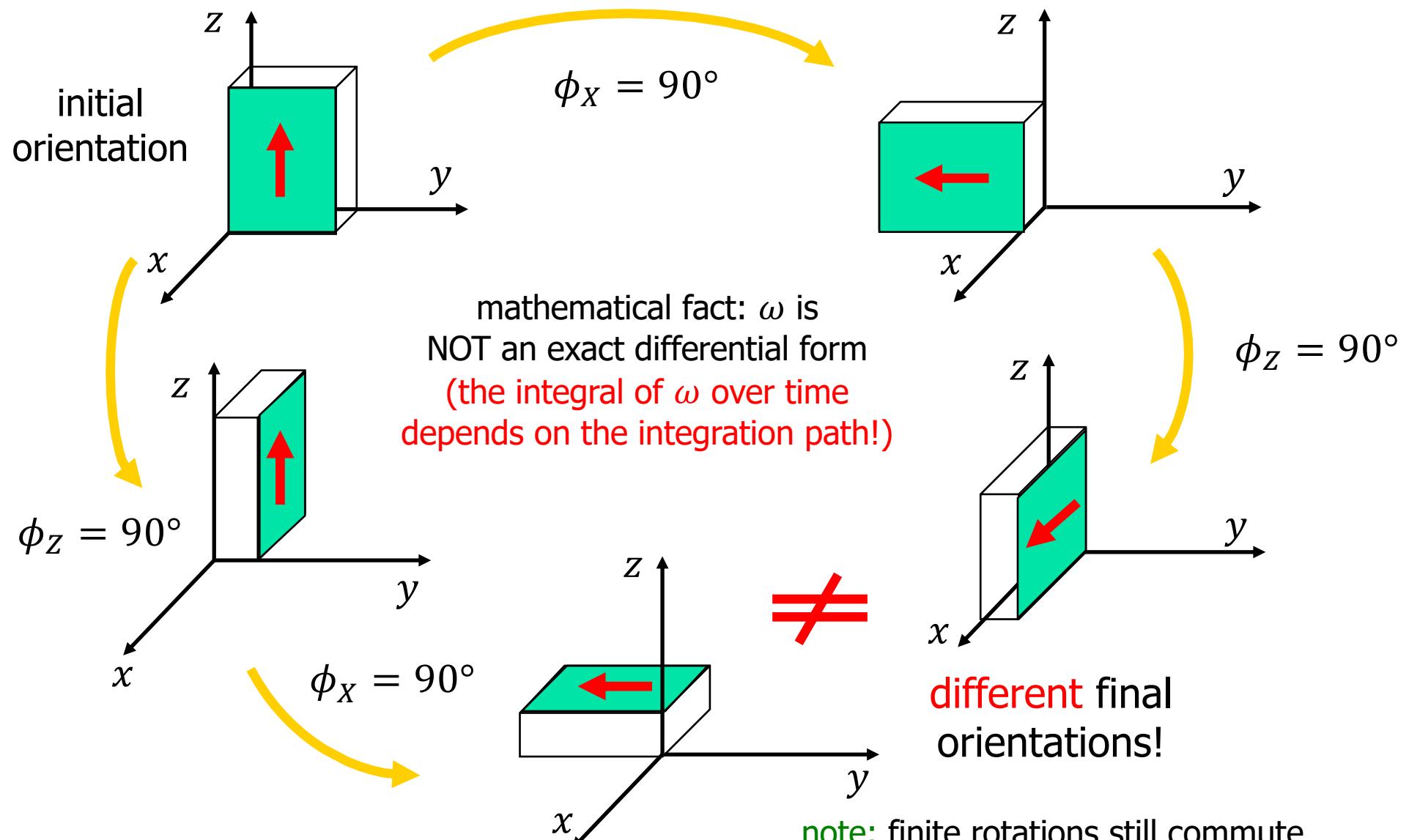
- finite $\Delta x, \Delta y, \Delta z$ or infinitesimal dx, dy, dz translations (linear displacements) always commute





Finite rotations do not commute

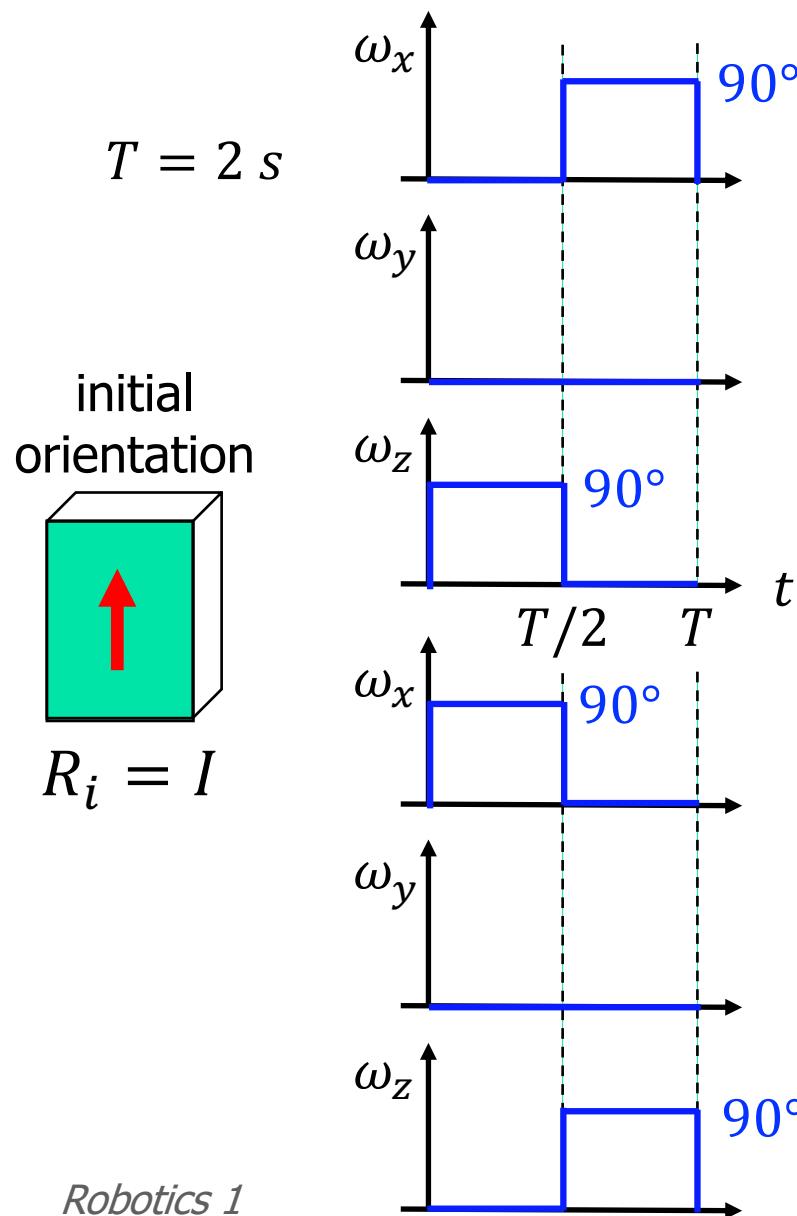
example





ω is not an exact differential

whiteboard ...



$$\int_0^T \omega(t) dt = \int_0^T \begin{pmatrix} \omega_x(t) \\ \omega_y(t) \\ \omega_z(t) \end{pmatrix} dt = \begin{pmatrix} 90^\circ \\ 0 \\ 90^\circ \end{pmatrix}$$

$$\int_0^T \dot{\phi}(t) dt = \int_0^T \frac{d\phi}{dt} dt = \int_{\phi(0)}^{\phi(T)} d\phi = \phi_f - \phi_i$$

an exact differential form

$$\int_0^T \omega(t) dt = \dots = \begin{pmatrix} 90^\circ \\ 0 \\ 90^\circ \end{pmatrix}$$

...the same value
but a different...

first final
orientation

$R_{f,ZX}$

$R_{f,XZ}$

...final
orientation



Infinitesimal rotations commute!

- infinitesimal rotations $d\phi_X, d\phi_Y, d\phi_Z$ around x, y, z axes

$$R_X(\phi_X) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi_X & -\sin \phi_X \\ 0 & \sin \phi_X & \cos \phi_X \end{bmatrix} \rightarrow R_X(d\phi_X) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\phi_X \\ 0 & d\phi_X & 1 \end{bmatrix}$$

$$R_Y(\phi_Y) = \begin{bmatrix} \cos \phi_Y & 0 & \sin \phi_Y \\ 0 & 1 & 0 \\ -\sin \phi_Y & 0 & \cos \phi_Y \end{bmatrix} \rightarrow R_Y(d\phi_Y) = \begin{bmatrix} 1 & 0 & d\phi_Y \\ 0 & 1 & 0 \\ -d\phi_Y & 0 & 1 \end{bmatrix}$$

$$R_Z(\phi_Z) = \begin{bmatrix} \cos \phi_Z & -\sin \phi_Z & 0 \\ \sin \phi_Z & \cos \phi_Z & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow R_Z(d\phi_Z) = \begin{bmatrix} 1 & -d\phi_Z & 0 \\ d\phi_Z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\blacksquare R(d\phi) = R(d\phi_X, d\phi_Y, d\phi_Z) = \begin{bmatrix} 1 & -d\phi_Z & d\phi_Y \\ d\phi_Z & 1 & -d\phi_X \\ -d\phi_Y & d\phi_X & 1 \end{bmatrix}$

↑
in **any** order

$= I + S(d\phi)$

neglecting
second- and
third-order
(infinitesimal)
terms



Time derivative of a rotation matrix

- let $R = R(t)$ be a rotation matrix, given as a function of time
- since $I = R(t)R^T(t)$, taking the time derivative of both sides yields

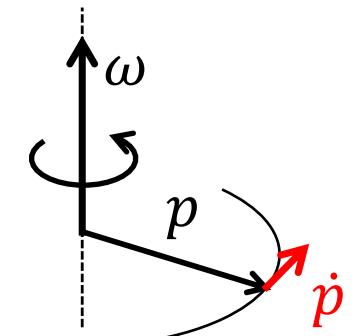
$$\begin{aligned} 0 &= d(R(t)R^T(t))/dt = (dR(t)/dt)R^T(t) + R(t)(dR^T(t)/dt) \\ &= (dR(t)/dt)R^T(t) + ((dR(t)/dt) R^T(t))^T \end{aligned}$$

thus $(dR(t)/dt) R^T(t) = S(t)$ is a **skew-symmetric** matrix

- let $p(t) = R(t)p'$ a vector (with constant norm) rotated over time
- comparing

$$\begin{aligned} \dot{p}(t) &= (dR(t)/dt)p' = S(t)R(t)p' = S(t)p(t) \\ \dot{p}(t) &= \omega(t) \times p(t) = S(\omega(t))p(t) \end{aligned}$$

we get $S = S(\omega)$



$$\boxed{\dot{R} = S(\omega)R} \quad \leftrightarrow \quad \boxed{S(\omega) = \dot{R} R^T}$$



Example

Time derivative of an elementary rotation matrix

$$R_X(\phi(t)) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi(t) & -\sin \phi(t) \\ 0 & \sin \phi(t) & \cos \phi(t) \end{bmatrix}$$

$$\begin{aligned} \dot{R}_X(\phi) R_X^T(\phi) &= \dot{\phi} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin \phi & -\cos \phi \\ 0 & \cos \phi & -\sin \phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\dot{\phi} \\ 0 & \dot{\phi} & 0 \end{bmatrix} = S(\omega) \quad \xrightarrow{\text{blue arrow}} \quad \omega = \omega_X = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

more in general, for the **axis/angle** rotation matrix

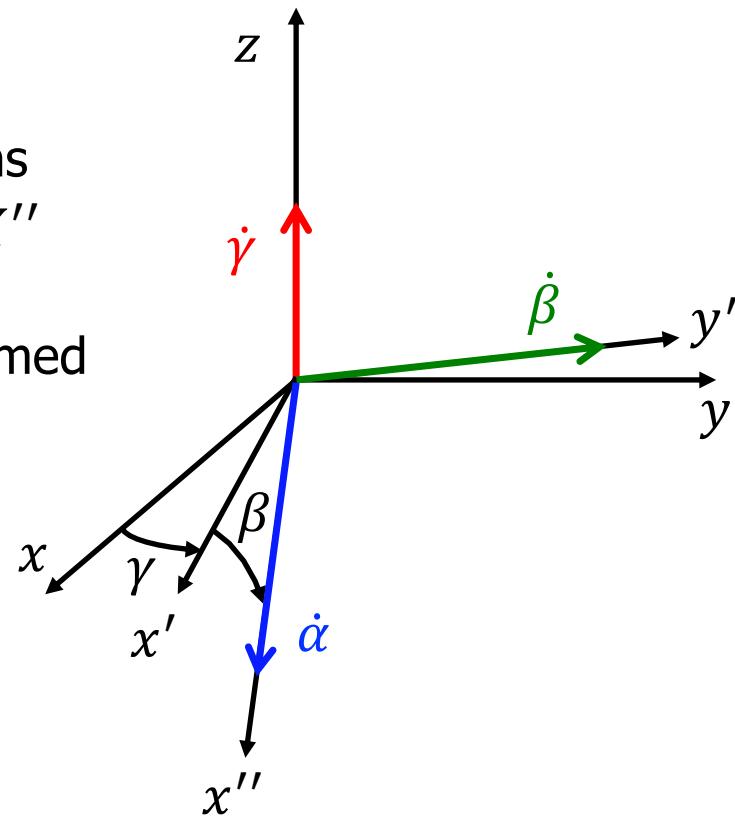
$$R(r, \theta(t)) \Rightarrow \dot{R}(r, \theta) R^T(r, \theta) = S(\omega) \quad \xrightarrow{\text{blue arrow}} \quad \omega = \omega_r = \dot{\theta} r = \dot{\theta} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$



Time derivative of RPY angles and ω

$$R_{RPY}(\alpha_X, \beta_Y, \gamma_Z) = R_{ZY'X''}(\gamma_Z, \beta_Y, \alpha_X) = R_Z(\gamma)R_{Y'}(\beta)R_{X''}(\alpha)$$

the three contributions $\dot{\gamma}Z, \dot{\beta}Y', \dot{\alpha}X''$ to ω are simply summed as vectors



$$\omega = \underbrace{\begin{bmatrix} c\beta c\gamma & -s\gamma & 0 \\ c\beta s\gamma & c\gamma & 0 \\ -s\beta & 0 & 1 \end{bmatrix}}_{T_{RPY}(\beta, \gamma)} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix}$$

$X'' \quad Y' \quad Z$
 ↑ ↑
 1st col in 2nd col in
 $R_Z(\gamma)R_{Y'}(\beta) \quad R_Z(\gamma)$

$\det T_{RPY}(\beta, \gamma) = \cos \beta = 0$
 for $\beta = \pm \pi/2$
 (singularity of the RPY representation)

similar treatment for the other 11 minimal representations...



Robot Jacobian matrices

- **analytical** Jacobian (obtained by **time differentiation**)

$$r = \begin{pmatrix} p \\ \phi \end{pmatrix} = f_r(q) \quad \rightarrow \quad \dot{r} = \begin{pmatrix} \dot{p} \\ \dot{\phi} \end{pmatrix} = \frac{\partial f_r(q)}{\partial q} \dot{q} = J_r(q) \dot{q}$$

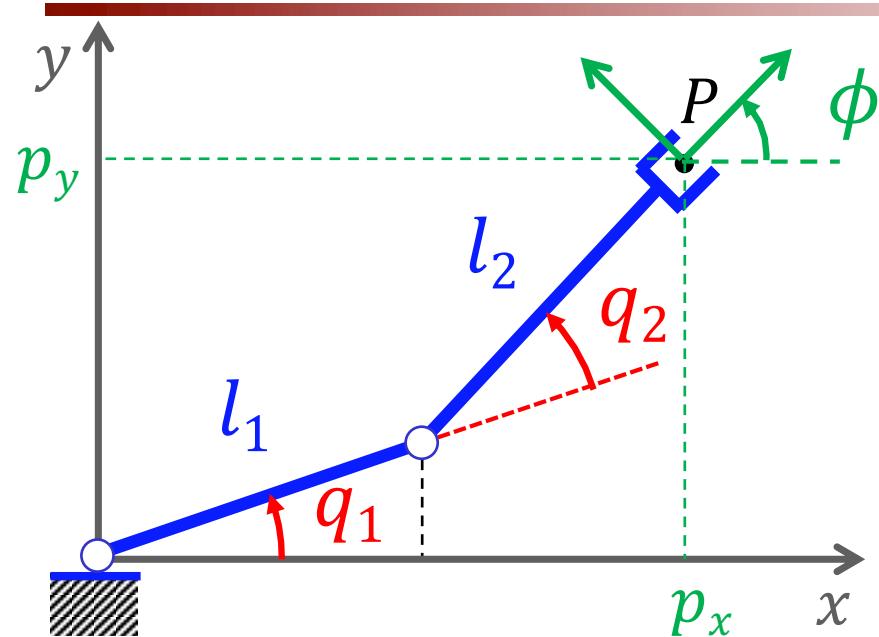
- **geometric** or basic Jacobian (**no** derivatives)

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} J_L(q) \\ J_A(q) \end{pmatrix} \dot{q} = J(q) \dot{q}$$

- in both cases, the Jacobian matrix **depends** on the **(current) configuration** of the robot



Analytical Jacobian of planar 2R arm



direct kinematics

$$r \left\{ \begin{array}{l} p_x = l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ p_y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ \phi = q_1 + q_2 \end{array} \right.$$

$$\dot{p}_x = -l_1 s_1 \dot{q}_1 - l_2 s_{12}(\dot{q}_1 + \dot{q}_2)$$

$$\dot{p}_y = l_1 c_1 \dot{q}_1 + l_2 c_{12}(\dot{q}_1 + \dot{q}_2)$$

$$\dot{\phi} = \omega_z = \dot{q}_1 + \dot{q}_2$$



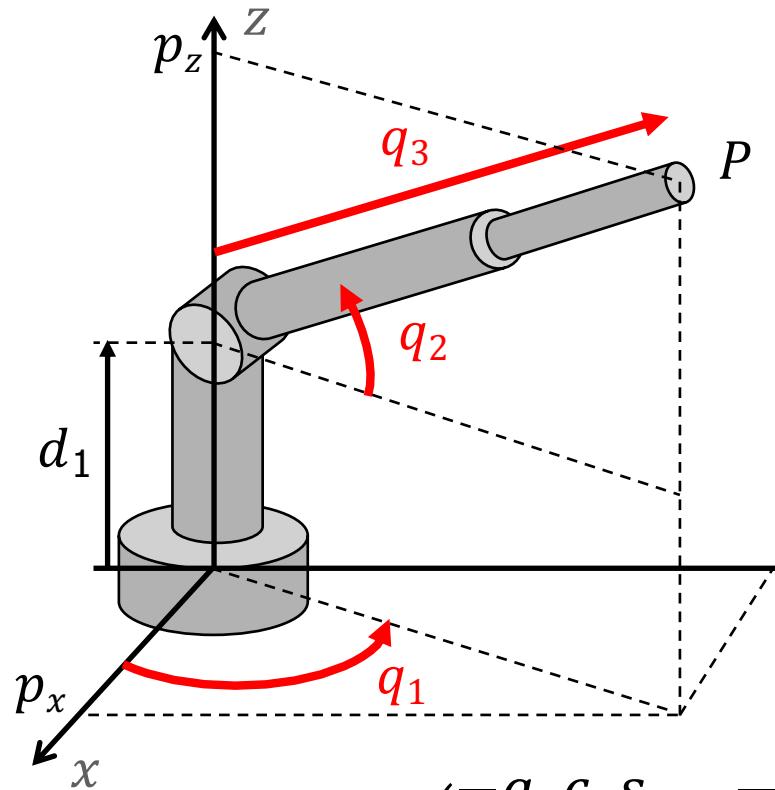
$$J_r(q) = \begin{pmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 1 & 1 \end{pmatrix}$$

given r , this is a 3×2 matrix

here, all rotations occur around the same fixed axis z (normal to the plane of motion)



Analytical Jacobian of polar (RRP) robot



direct kinematics (here, $r = p$)

$$\begin{aligned} p_x &= q_3 c_2 c_1 \\ p_y &= q_3 c_2 s_1 \\ p_z &= d_1 + q_3 s_2 \end{aligned} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} f_r(q)$$

taking the time derivative

$$v = \dot{p} = \underbrace{\begin{pmatrix} -q_3 c_2 s_1 & -q_3 s_2 c_1 & c_2 c_1 \\ q_3 c_2 c_1 & -q_3 s_2 s_1 & c_2 s_1 \\ 0 & q_3 c_2 & s_2 \end{pmatrix}}_{\frac{\partial f_r(q)}{\partial q}} \dot{q} = J_r(q) \dot{q}$$



Geometric Jacobian

end-effector
instantaneous
velocity

always a $6 \times n$ matrix

$$\begin{pmatrix} v_E \\ \omega_E \end{pmatrix} = \begin{pmatrix} J_L(q) \\ J_A(q) \end{pmatrix} \dot{q} = \begin{pmatrix} J_{L1}(q) & \cdots & J_{Ln}(q) \\ J_{A1}(q) & \cdots & J_{An}(q) \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{pmatrix}$$

superposition of effects

$$v_E = J_{L1}(q)\dot{q}_1 + \cdots + J_{Ln}(q)\dot{q}_n$$

contribution to the linear e-e velocity due to \dot{q}_1

$$\omega_E = J_{A1}(q)\dot{q}_1 + \cdots + J_{An}(q)\dot{q}_n$$

contribution to the angular e-e velocity due to \dot{q}_1

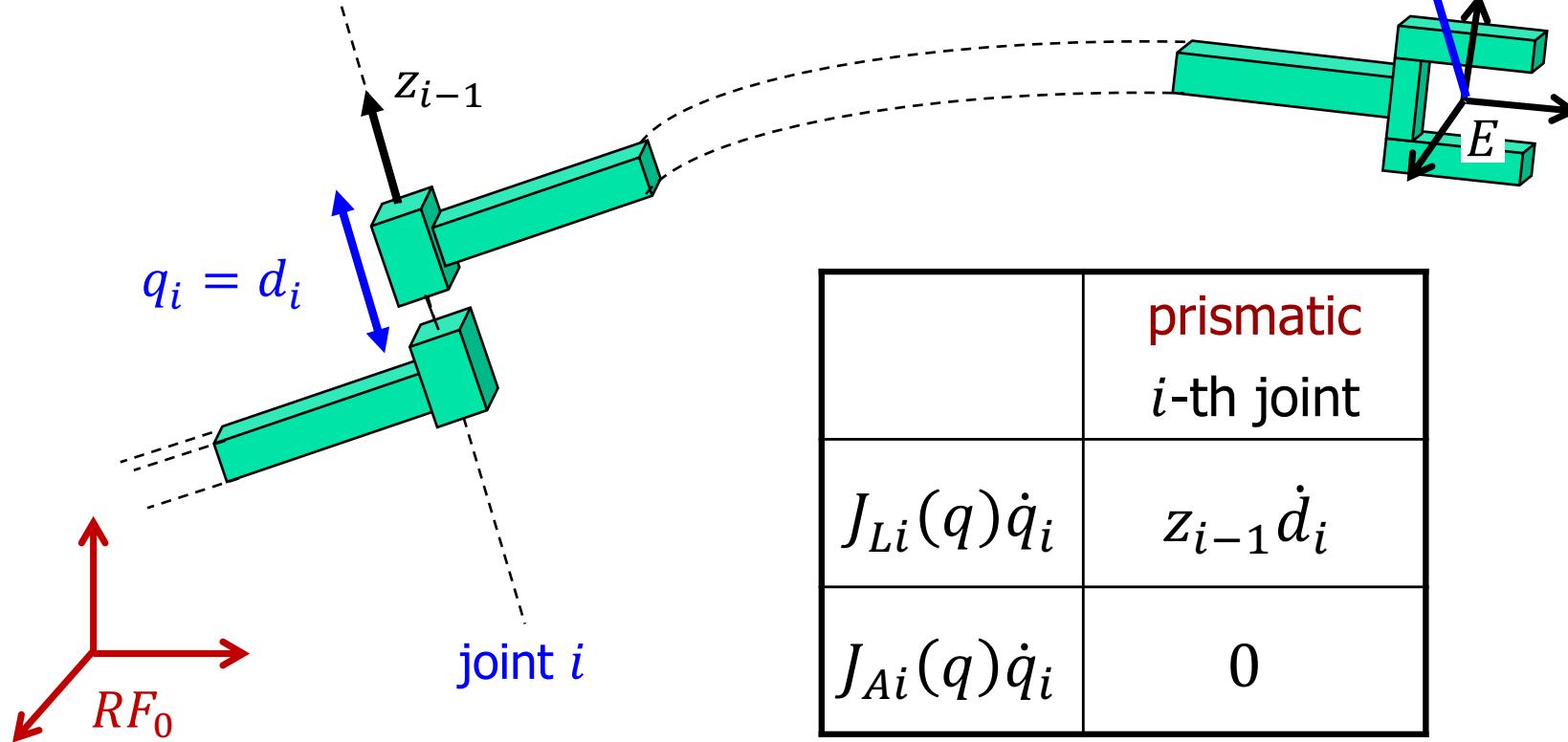
linear and angular velocity belong to (linear) vector spaces in \mathbb{R}^3



Contribution of a prismatic joint

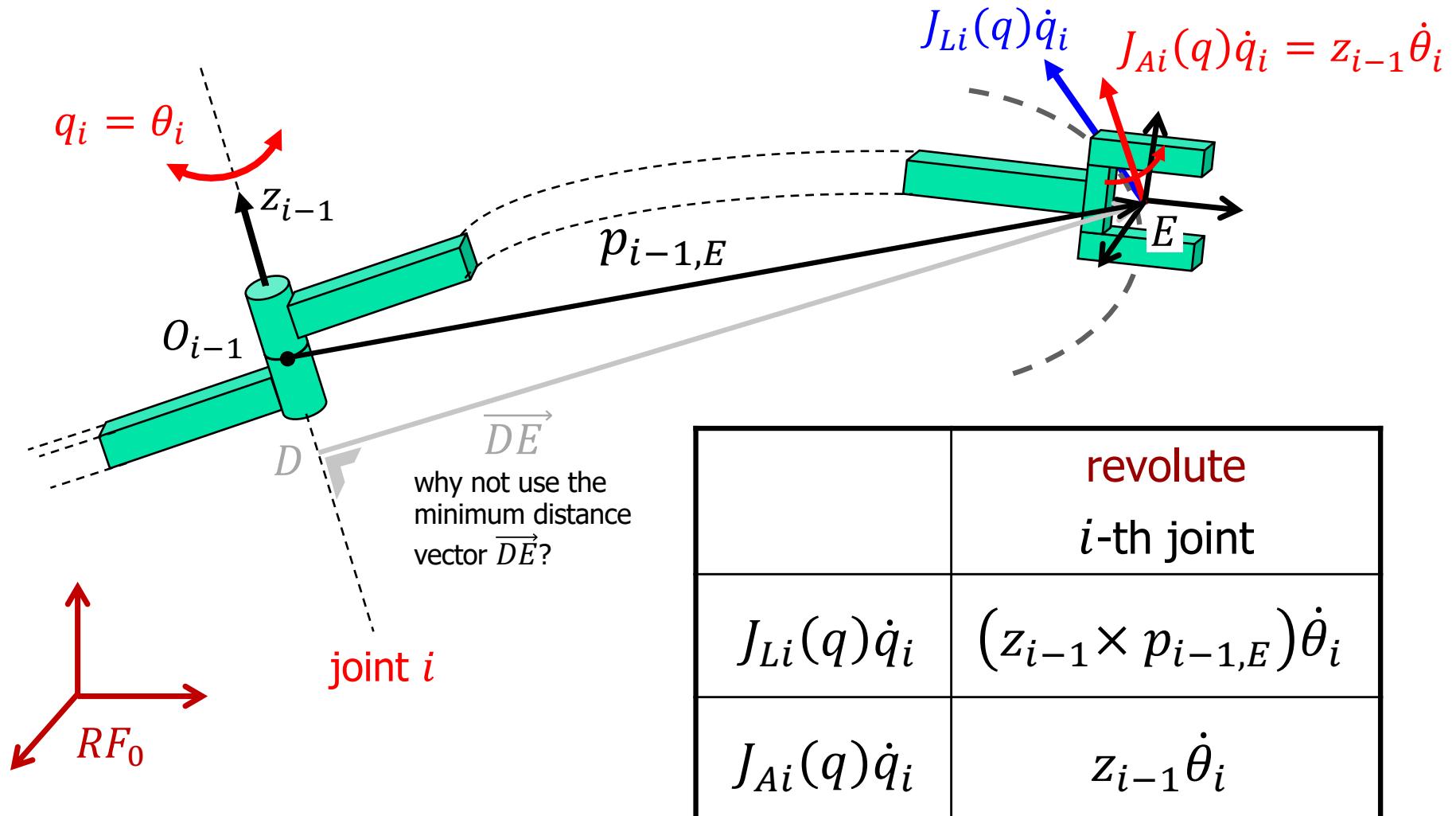
note: joints beyond the i -th one are considered to be “frozen”, so that the distal part of the robot is a **single rigid body**

$$J_{Li}(q)\dot{q}_i = z_{i-1}\dot{d}_i$$





Contribution of a revolute joint





Expression of geometric Jacobian

$$\begin{pmatrix} \dot{p}_{0,E} \\ \omega_E \end{pmatrix} = \begin{pmatrix} v_E \\ \omega_E \end{pmatrix} = \begin{pmatrix} J_L(q) \\ J_A(q) \end{pmatrix} \dot{q} = \begin{pmatrix} J_{L1}(q) & \cdots & J_{Ln}(q) \\ J_{A1}(q) & \cdots & J_{An}(q) \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{pmatrix}$$

	prismatic <i>i</i> -th joint	revolute <i>i</i> -th joint
$J_{Li}(q)$	z_{i-1}	$z_{i-1} \times p_{i-1,E}$
$J_{Ai}(q)$	0	z_{i-1}

this can be also computed as

$$= \frac{\partial p_{0,E}(q)}{\partial q_i}$$

$$z_{i-1} = {}^0R_1(q_1) \cdots {}^{i-2}R_{i-1}(q_{i-1}) {}^{i-1}z_{i-1}$$

$$p_{i-1,E} = p_{0,E}(q_1, \dots, q_n) - p_{0,i-1}(q_1, \dots, q_{i-1})$$

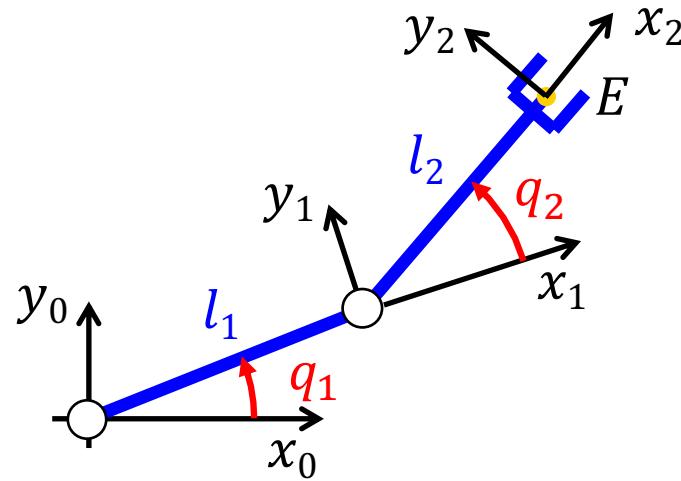
complete kinematics
for e-e position

partial kinematics
for O_{i-1} position

all vectors should be expressed in the same reference frame
(here, the **base frame RF_0**)



Geometric Jacobian of planar 2R arm



$$J(q) = \begin{pmatrix} z_0 \times p_{0,E} & z_1 \times p_{1,E} \\ z_0 & z_1 \end{pmatrix}$$

$$z_0 = z_1 = z_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Denavit-Hartenberg table

joint	α_i	d_i	a_i	θ_i
1	0	0	l_1	q_1
2	0	0	l_2	q_2

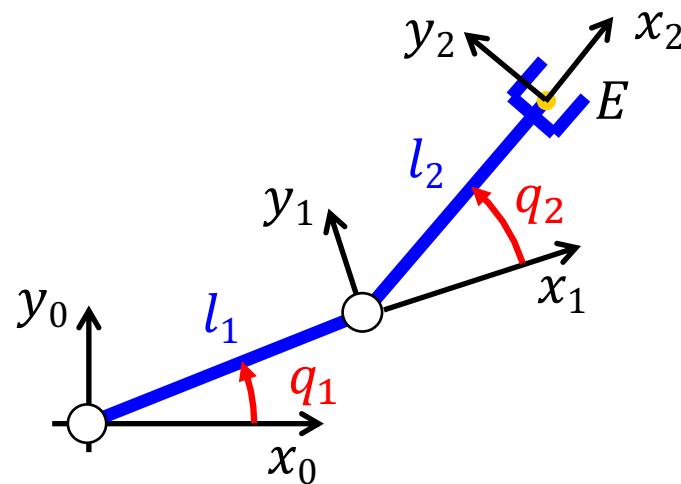
$${}^0A_1 = \begin{pmatrix} c_1 & -s_1 & 0 & l_1 c_1 \\ s_1 & c_1 & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \leftarrow p_{0,1}$$

$${}^0A_2 = \begin{pmatrix} c_{12} & -s_{12} & 0 & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & 0 & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \leftarrow p_{0,E}$$

$$p_{1,E} = p_{0,E} - p_{0,1}$$



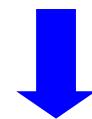
Geometric Jacobian of planar 2R arm



$$J(q) = \begin{pmatrix} z_0 \times p_{0,E} & z_1 \times p_{1,E} \\ z_0 & z_1 \end{pmatrix}$$

$$= \begin{pmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix}$$

note: the Jacobian is here a 6×2 matrix,
thus its **maximum rank** is **2**

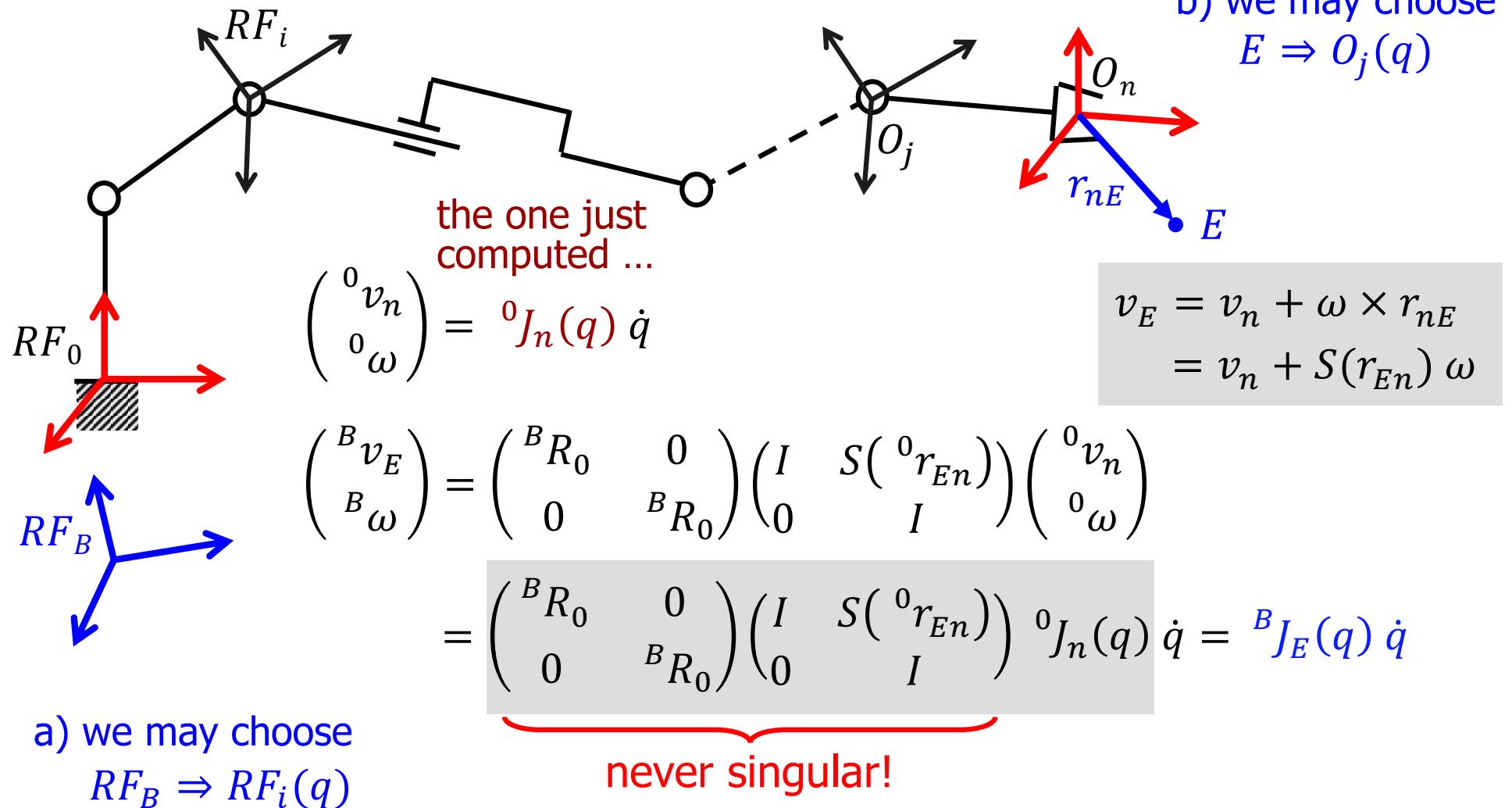


at most 2 components of the linear/angular
end-effector velocity can be **independently** assigned

compare rows 1, 2, and 6
with the analytical
Jacobian in slide #13!



Transformations of Jacobian matrix





Example: Dexter robot

- 8R robot manipulator with transmissions by pulleys and steel cables (joints 3 to 8)
 - lightweight: only 15 kg in motion
 - motors located in second link
 - incremental encoders (homing)
 - **redundancy degree for e-e pose task:** $n - m = 2$
 - compliant in the interaction with environment



i	a (mm)	d (mm)	α (rad)	range θ (deg)
0	0	0	$-\pi/2$	[-12.56, 179.89]
1	144	450	$-\pi/2$	[-83, 84]
2	0	0	$\pi/2$	[7, 173]
3	100	350	$\pi/2$	[65, 295]
4	0	0	$-\pi/2$	[-174, -3]
5	24	250	$-\pi/2$	[57, 265]
6	0	0	$-\pi/2$	[-129.99, -45]
7	100	0	π	[-55.05, 30]



Mid-frame Jacobian of Dexter robot

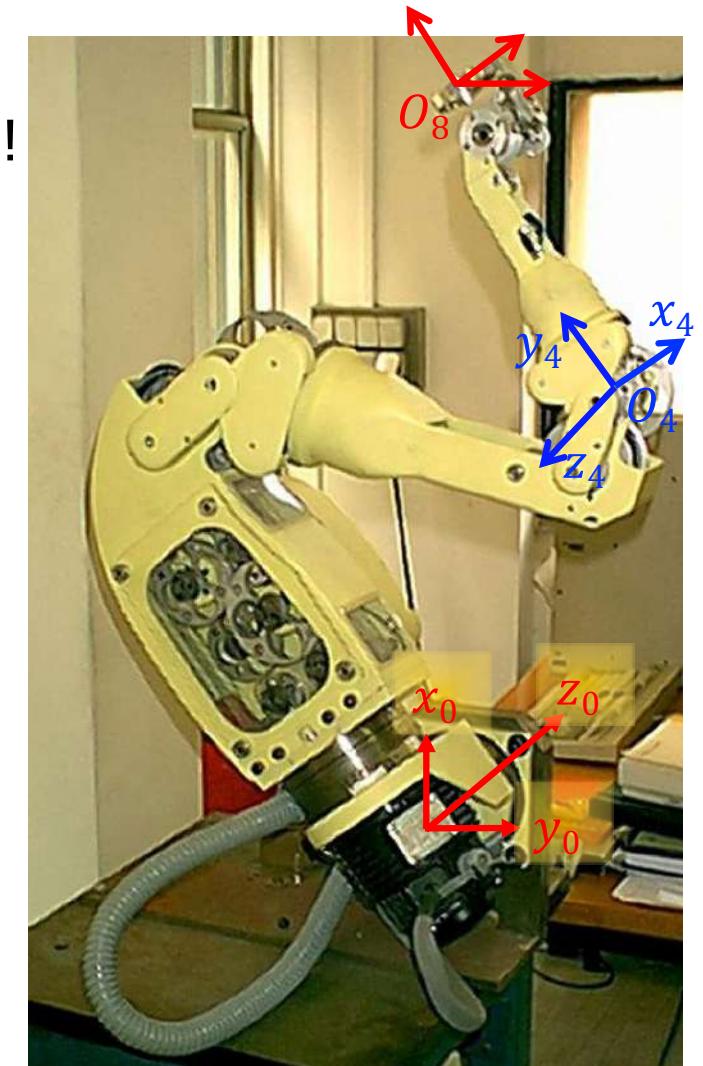
- geometric Jacobian ${}^0J_8(q)$ is very complex
- “mid-frame” Jacobian ${}^4J_4(q)$ is relatively simple!

$${}^4\hat{J}_4 = \begin{bmatrix} d_1 s_1 s_3 + d_3 s_3 c_2 s_1 - a_1 c_3 c_1 s_2 - d_1 c_3 c_1 c_2 - d_3 c_1 c_3 \\ -a_3 s_3 c_2 s_1 + a_3 c_3 c_1 + a_1 c_1 c_2 - d_1 c_1 s_2 \\ -d_3 c_3 c_2 s_1 - a_1 s_3 c_1 s_2 - d_1 s_3 c_1 c_2 - d_3 s_3 c_1 - d_1 s_1 c_3 + a_3 s_2 s_1 \\ -c_3 c_2 s_1 - s_3 c_1 \\ -s_2 s_1 \\ -s_3 c_2 s_1 + c_3 c_1 \end{bmatrix}$$

6 rows,
8 columns

$$\begin{bmatrix} a_1 s_3 + d_3 s_3 s_2 & d_3 c_3 & 0 & 0 & 0 \\ -a_3 s_3 s_2 & -a_3 c_3 & 0 & 0 & 0 \\ -a_1 c_3 - d_3 c_3 s_2 - a_3 c_2 & d_3 s_3 & -a_3 & 0 & 0 \\ -c_3 s_2 & s_3 & 0 & 0 & -s_4 \\ c_2 & 0 & 1 & 0 & c_4 \\ -s_3 s_2 & -c_3 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -a_5 s_4 - d_5 c_5 c_4 & -a_5 s_5 c_4 c_6 + d_5 s_5 s_6 c_4 \\ -d_5 c_5 s_4 + a_5 c_4 & d_5 s_5 s_6 s_4 - a_5 s_5 s_4 c_6 \\ d_5 s_5 & -a_5 c_6 c_5 + d_5 c_5 s_6 \\ -c_4 s_5 & -c_4 c_5 s_6 + s_4 c_6 \\ -s_4 s_5 & -s_4 c_5 s_6 - c_4 c_6 \\ -c_5 & s_5 s_6 \end{bmatrix}$$





Summary of differential relations

$$\dot{p} \rightleftharpoons v \quad \dot{p} = v$$

$$\dot{R} \rightleftharpoons \omega \quad \dot{R} = S(\omega)R \quad \longleftrightarrow \quad \text{for each (unit) column } r_i \text{ of } R \text{ (a frame): } \dot{r}_i = \omega \times r_i$$

$$S(\omega) = \dot{R}R^T$$

$$\dot{\phi} \rightleftharpoons \omega \quad \omega = \omega_{\dot{\phi}_1} + \omega_{\dot{\phi}_2} + \omega_{\dot{\phi}_3} = a_1 \dot{\phi}_1 + a_2(\phi_1) \dot{\phi}_2 + a_3(\phi_1, \phi_2) \dot{\phi}_3$$

$$= T(\phi) \dot{\phi}$$

(moving) axes of definition for the sequence of rotations ϕ_i , $i = 1, 2, 3$

if the task vector r is

$$r = \begin{pmatrix} p \\ \phi \end{pmatrix} \quad \rightarrow \quad J_r(q) = \begin{pmatrix} I & 0 \\ 0 & T^{-1}(\phi) \end{pmatrix} J(q) \quad \longleftrightarrow \quad J(q) = \begin{pmatrix} I & 0 \\ 0 & T(\phi) \end{pmatrix} J_r(q)$$

$T(\phi)$ has always \Leftrightarrow singularity of the **specific** minimal representation of orientation

a singularity



Acceleration relations (and beyond...)

Higher-order differential kinematics

- differential relations between motion in the joint space and motion in the task space can be established at the second order, third order, ...
- the analytical Jacobian always “weights” the highest-order derivative



velocity

$$\dot{r} = \boxed{J_r(q)} \dot{q} \quad \text{matrix function } N_2(q, \dot{q})$$

acceleration

$$\ddot{r} = J_r(q) \ddot{q} + \dot{J}_r(q) \dot{q} \quad \text{matrix function } N_3(q, \dot{q}, \ddot{q})$$

jerk

$$\dddot{r} = J_r(q) \ddot{\ddot{q}} + 2\dot{J}_r(q) \ddot{q} + \ddot{J}_r(q) \dot{q}$$

snap

$$\ddot{\ddot{r}} = J_r(q) \ddot{\ddot{q}} + \dots$$

- the same holds true also for the geometric Jacobian $J(q)$



Primer on linear algebra

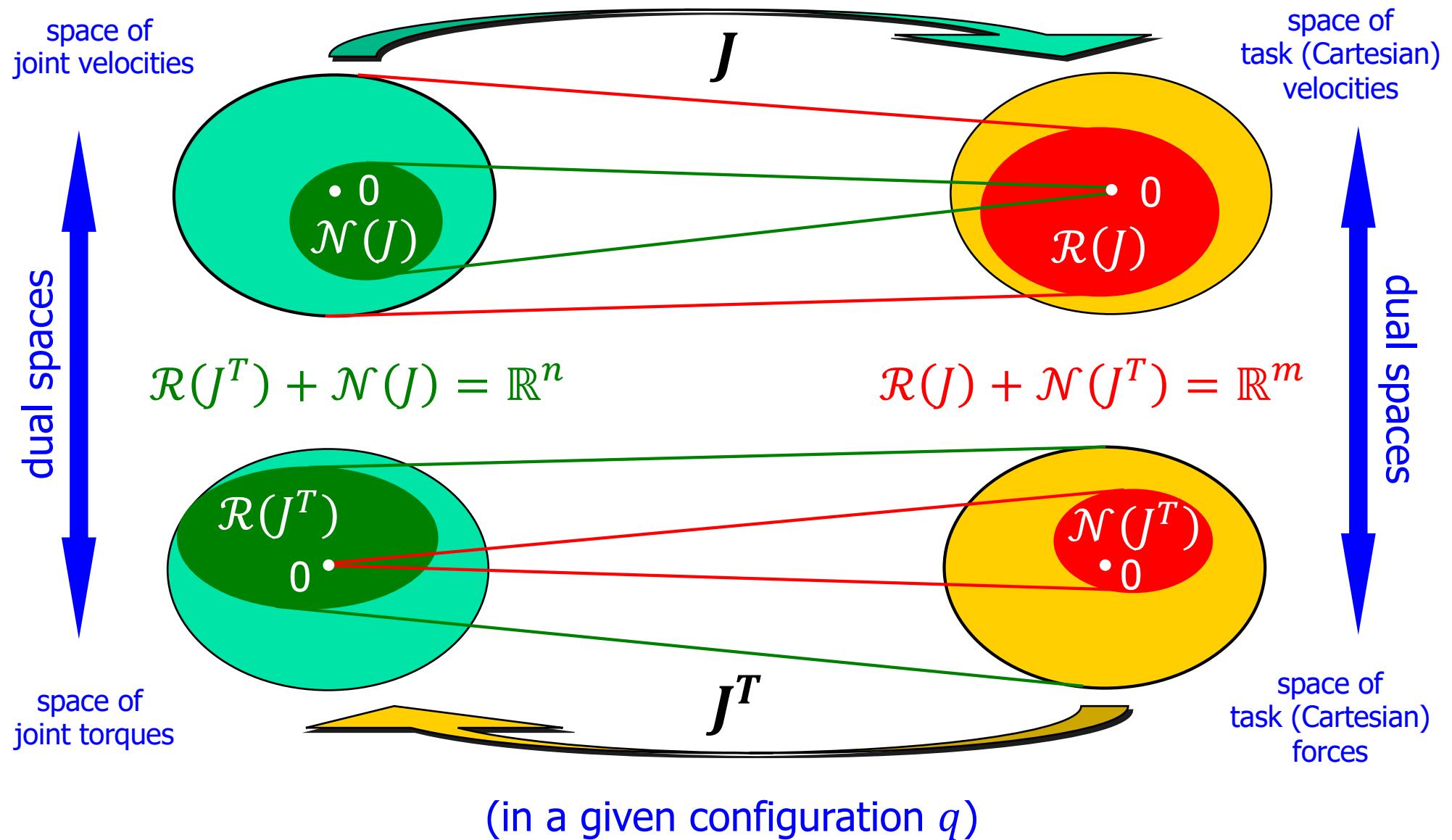
given a matrix J : $m \times n$ (m rows, n columns)

- **rank** $\rho(J) = \max$ # of rows or columns that are linearly independent
 - $\rho(J) \leq \min(m, n) \Leftarrow$ if equality holds, J has full rank
 - if $m = n$ and J has full rank, J is nonsingular and the inverse J^{-1} exists
 - $\rho(J) =$ dimension of the largest nonsingular square submatrix of J
- **range** space $\mathcal{R}(J) =$ subspace of all linear combinations of the columns of J
$$\mathcal{R}(J) = \{v \in \mathbb{R}^m : \exists \xi \in \mathbb{R}^n, v = J\xi\} \quad \leftarrow \text{also called } \text{image of } J$$
 - $\dim(\mathcal{R}(J)) = \rho(J)$
- **null** space $\mathcal{N}(J) =$ subspace of all vectors that are zeroed by matrix J
$$\mathcal{N}(J) = \{\xi \in \mathbb{R}^n : J\xi = 0 \in \mathbb{R}^m\} \quad \leftarrow \text{also called } \text{kernel of } J$$
 - $\dim(\mathcal{N}(J)) = n - \rho(J)$
- $\mathcal{R}(J) + \mathcal{N}(J^T) = \mathbb{R}^m$ and $\mathcal{R}(J^T) + \mathcal{N}(J) = \mathbb{R}^n$ (sum of vector subspaces)
 - any element $v \in V = V_1 + V_2$ can be written as $v = v_1 + v_2$, $v_1 \in V_1, v_2 \in V_2$



Robot Jacobian

decomposition in linear subspaces and duality





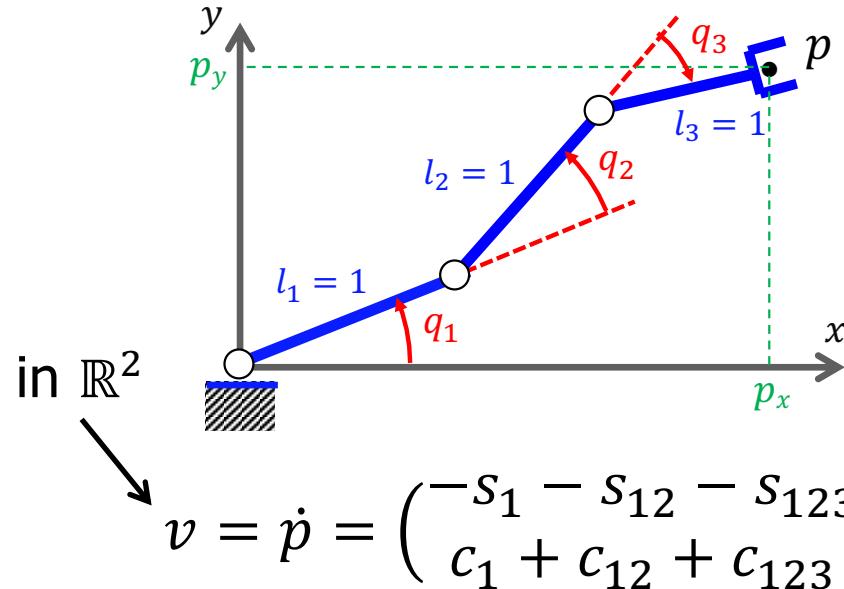
Mobility analysis in the task space

- $\rho(J) = \rho(J(q))$, $\mathcal{R}(J) = \mathcal{R}(J(q))$, $\mathcal{N}(J^T) = \mathcal{N}(J^T(q))$, etc. are **locally** defined, i.e., they depend on the **current configuration q**
- $\mathcal{R}(J(q))$ is the subspace of all “generalized” velocities (with linear and/or angular components) that can be **instantaneously** realized by the robot end-effector when varying the joint velocities \dot{q} at the current q
- if $\rho(J(q)) = m$ at q ($J(q)$ has **max rank**, with $m \leq n$), the end-effector can be **moved in any direction** of the task space \mathbb{R}^m
- if $\rho(J(q)) < m$, there are directions in \mathbb{R}^m in which the end-effector **cannot move** (at least, not instantaneously!)
 - these directions $\in \mathcal{N}(J^T(q))$, the complement of $\mathcal{R}(J(q))$ to task space \mathbb{R}^m , which is of dimension $m - \rho(J(q))$
- if $\mathcal{N}(J(q)) \neq \{0\}$, there are **non-zero** joint velocities \dot{q} that produce **zero** end-effector velocity (“**self motions**”)
 - this happens **always** for $m < n$, i.e., when the robot is redundant for the task



Mobility analysis for a planar 3R robot

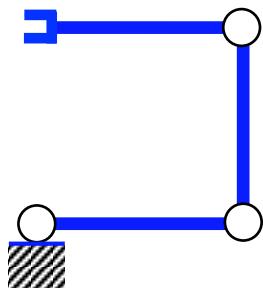
whiteboard ...



case 1)

$$q = (0, \pi/2, \pi/2)$$

$$J = \begin{pmatrix} -1 & -1 & 0 \\ 0 & -1 & -1 \end{pmatrix}$$



$$l_1 = l_2 = l_3 = 1 \quad n = 3, \quad m = 2$$

$$WS_1 = \{p \in \mathbb{R}^2 : \|p\| \leq 3\} \subset \mathbb{R}^2$$

$$WS_2 = \{p \in \mathbb{R}^2 : \|p\| \leq 1\} \subset \mathbb{R}^2$$

$$p = \begin{pmatrix} c_1 + c_{12} + c_{123} \\ s_1 + s_{12} + s_{123} \end{pmatrix}$$

in \mathbb{R}^3



case 2)

$$q = (\pi/2, 0, \pi)$$

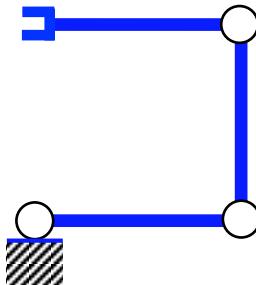
$$J = \begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

- run the Matlab code **subspaces_3Rplanar.m** available in the course material



Mobility analysis for a planar 3R robot

whiteboard ...



$$q = (0, \pi/2, \pi/2)$$

case 1)

$$J = \begin{pmatrix} -1 & -1 & 0 \\ 0 & -1 & -1 \end{pmatrix} \quad J^T = \begin{pmatrix} -1 & 0 \\ -1 & -1 \\ 0 & -1 \end{pmatrix}$$

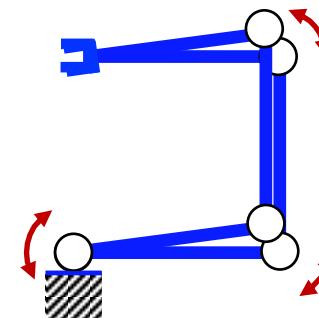
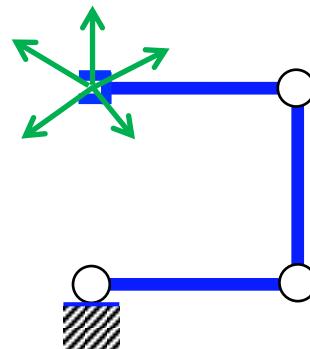
$$\rho(J) = 2 = m$$

$$\rho(J^T) = \rho(J) = 2$$

$$\mathcal{R}(J) = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} = \mathbb{R}^2$$

$$\mathcal{N}(J) = \left\{ \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \right\}$$

$$\begin{aligned} \dim \mathcal{N}(J) &= 1 \\ &= n - \rho(J) = n - m \end{aligned}$$



$$\mathcal{R}(J^T) = \left\{ \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \right\}$$

$$\dim \mathcal{R}(J^T) = 2 = m$$

$$\mathcal{N}(J^T) = 0$$

$$\mathcal{R}(J) + \mathcal{N}(J^T) = \mathbb{R}^2$$

$$\mathcal{R}(J^T) + \mathcal{N}(J) = \mathbb{R}^3$$



Mobility analysis for a planar 3R robot

whiteboard ...

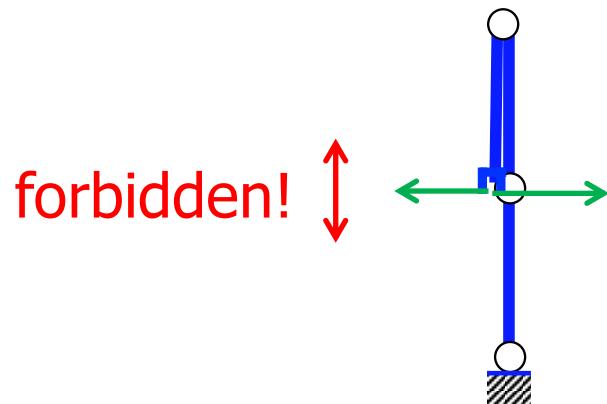


$$q = (\pi/2, 0, \pi)$$

case 2)

$$\mathcal{R}(J) = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$$

$$\dim \mathcal{R}(J) = 1 = \rho(J)$$



$$\mathcal{R}(J^T) = \left\{ \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

$$\dim \mathcal{R}(J^T) = 1 = m - \rho(J)$$

$$J = \begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

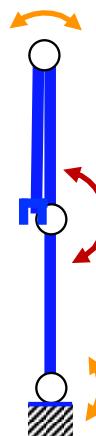
$$\rho(J) = 1 < m$$

$$\mathcal{N}(J) = \left\{ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

$$J^T = \begin{pmatrix} -1 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}$$

$$\rho(J^T) = \rho(J) = 1$$

$$\dim \mathcal{N}(J) = 2 = n - \rho(J)$$



$$\mathcal{R}(J) + \mathcal{N}(J^T) = \mathbb{R}^2$$

$$\mathcal{R}(J^T) + \mathcal{N}(J) = \mathbb{R}^3$$

$$\mathcal{N}(J^T) = \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

$$\dim \mathcal{N}(J^T) = 1 = n - \rho(J)$$

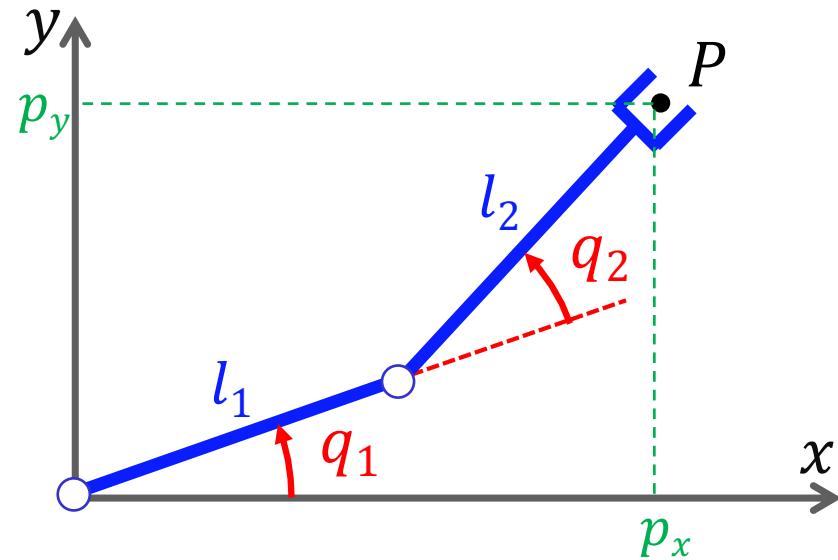


Kinematic singularities

- **configurations where the Jacobian loses rank**
 \Leftrightarrow **loss of instantaneous mobility** of the robot end-effector
- for $m = n$, they correspond to Cartesian poses at which the number of solutions of the **inverse kinematics** problem **differs** from the generic case
- “in” a **singular configuration**, we **cannot** find any joint velocity that realizes a desired end-effector velocity in **some** directions of the task space
- “close” to a **singularity**, **large joint velocities** may be needed to realize even a small velocity of the end-effector in **some** directions of the task space
- finding and analyzing in advance the mobility of a robot helps in **singularity avoidance** during **trajectory planning** and **motion control**
 - when $m = n$: find the configurations q such that $\det J(q) = 0$
 - when $m < n$: find the configurations q such that **all** $m \times m$ minors of $J(q)$ are singular (or, equivalently, such that $\det(J(q)J^T(q)) = 0$)
- finding all singular configurations of a robot with a **large** number of joints, or the actual “distance” from a singularity, is a **complex computational** task



Singularities of planar 2R robot



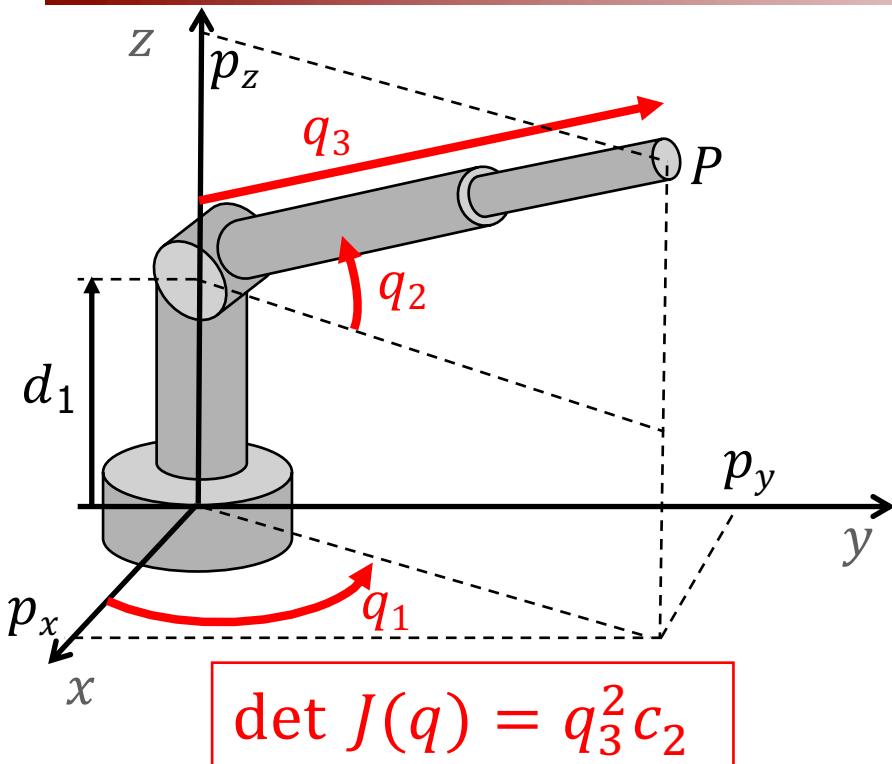
$$\det J(q) = l_1 l_2 s_2$$

$$\dot{p} = \begin{pmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{pmatrix} \dot{q} = J(q) \dot{q}$$

- **singularities**: robot arm is stretched ($q_2 = 0$) or folded ($q_2 = \pi$)
- singular configurations correspond **here** to Cartesian points that are **on the boundary** of the primary workspace
- **here, and in many cases**, singularities **separate** configuration space regions with **distinct** inverse kinematic solutions (e.g., elbow “up” or “down”)



Singularities of polar (RRP) robot



direct kinematics

$$p_x = q_3 c_2 c_1$$

$$p_y = q_3 c_2 s_1$$

$$p_z = d_1 + q_3 s_2$$

analytical Jacobian

$$\begin{aligned} \dot{p} &= \begin{pmatrix} -q_3 s_1 c_2 & -q_3 c_1 s_2 & c_1 c_2 \\ q_3 c_1 c_2 & -q_3 s_1 s_2 & s_1 c_2 \\ 0 & q_3 c_2 & s_2 \end{pmatrix} \dot{q} \\ &= J(q) \dot{q} \end{aligned}$$

- singularities
 - E-E is along the z axis ($q_2 = \pm\pi/2$): simple singularity $\Rightarrow \text{rank } \rho(J) = 2$
 - third link is fully retracted ($q_3 = 0$): double singularity $\Rightarrow \text{rank } \rho(J)$ drops to 1
- all singular configurations correspond here to Cartesian points internal to the workspace (supposing no range limits for the prismatic joint)



Singularities of robots with spherical wrist

- $n = 6$, last three joints are **revolute** and their axes **intersect** at a point
- without loss of generality, we set $O_6 = W$ = center of **spherical wrist** (i.e., choose $d_6 = 0$ in DH table) and obtain for the geometric Jacobian

$$J(q) = \begin{pmatrix} J_{11} & 0 \\ J_{12} & J_{22} \end{pmatrix}$$

- since $\det J(q_1, \dots, q_5) = \det J_{11} \cdot \det J_{22}$, there is a **decoupling** property
 - $\det J_{11}(q_1, q_2, q_3) = 0$ provides the **arm singularities**
 - $\det J_{22}(q_4, q_5) = 0$ provides the **wrist singularities**
- being in the geometric Jacobian $J_{22} = (z_3 \ z_4 \ z_5)$, **wrist** singularities correspond to when z_3, z_4 and z_5 become **linearly dependent vectors**
⇒ when either $q_5 = 0$ or $q_5 = \pm\pi/2$ (see Euler angles singularities!)
- inversion of $J(q)$ is simpler (block triangular structure)
- the determinant of $J(q)$ will **never** depend on q_1 : **why?**



Robotics 1

Trajectory planning

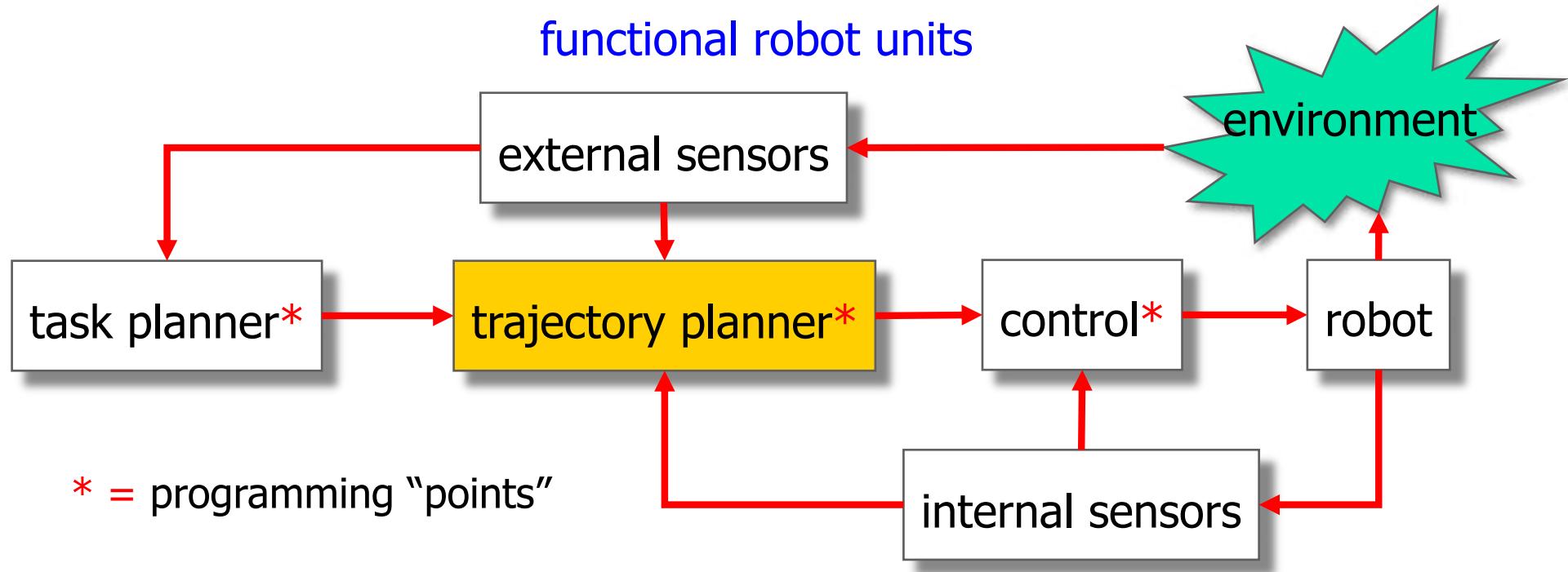
Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Trajectory planner interfaces



robot action described
as a sequence of poses
or configurations
(with possible exchange
of contact forces)



TRAJECTORY
PLANNER



reference profile/values
(continuous or discrete)
for the robot controller



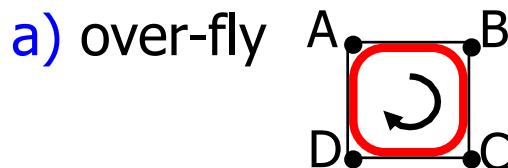
Trajectory definition

a standard procedure for industrial robots



1. define Cartesian pose points (position+orientation) using the teach-box
2. program an (average) velocity between these points, as a 0-100% of a maximum system value (different for Cartesian- and joint-space motion)
3. linear interpolation in the joint space between points sampled from the built trajectory

examples of additional features



b) sensor-driven STOP

c) circular path
through 3 points

main drawbacks

- semi-manual programming (as in “first generation” robot languages)
- limited visualization of motion

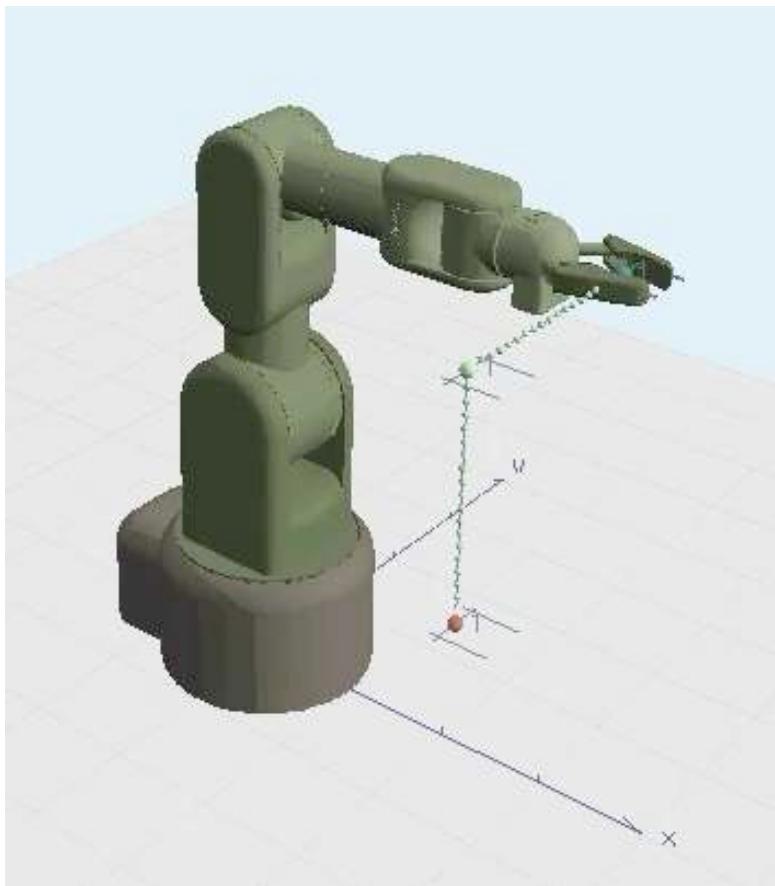


a mathematical formalization of trajectories is useful/needed

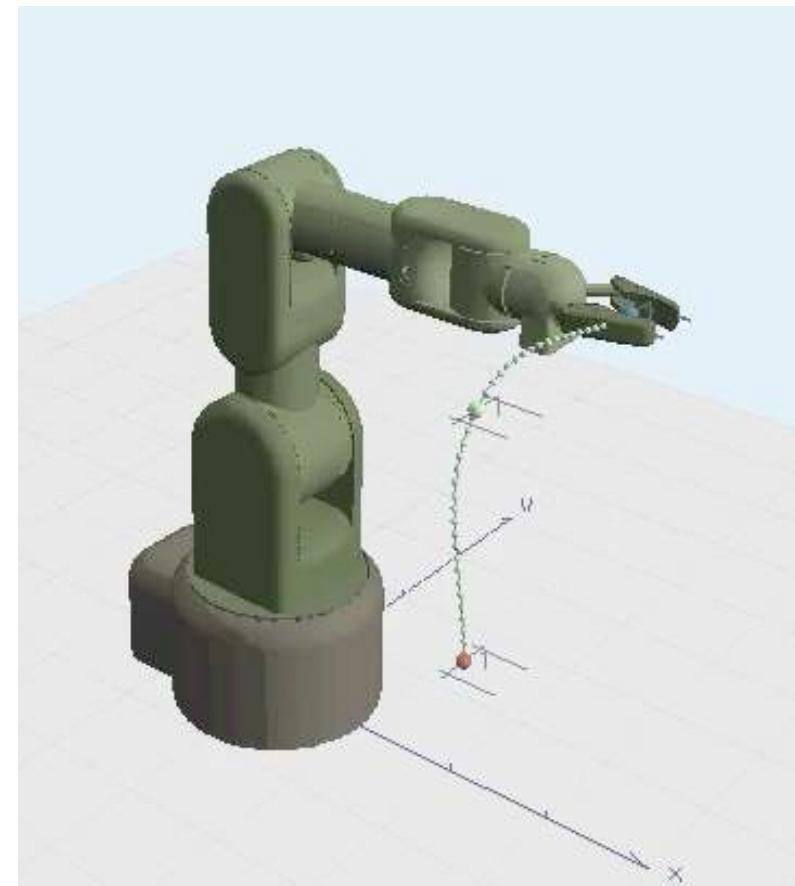


Some typical trajectories

- Point-to-point Cartesian motion with an **intermediate** point



Straight lines as Cartesian path

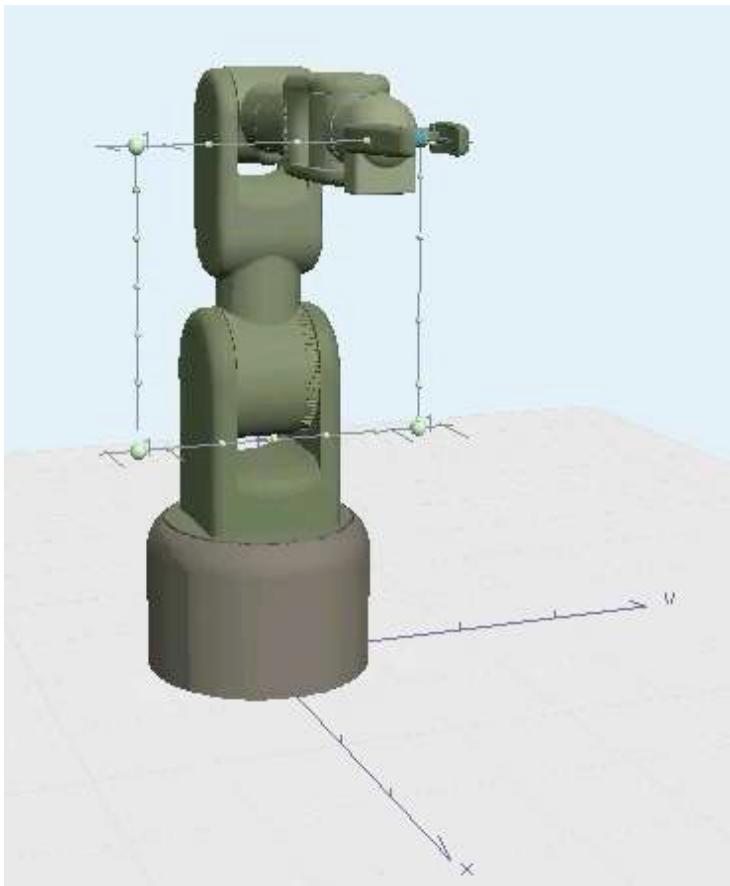


Interpolation with Bezier curves



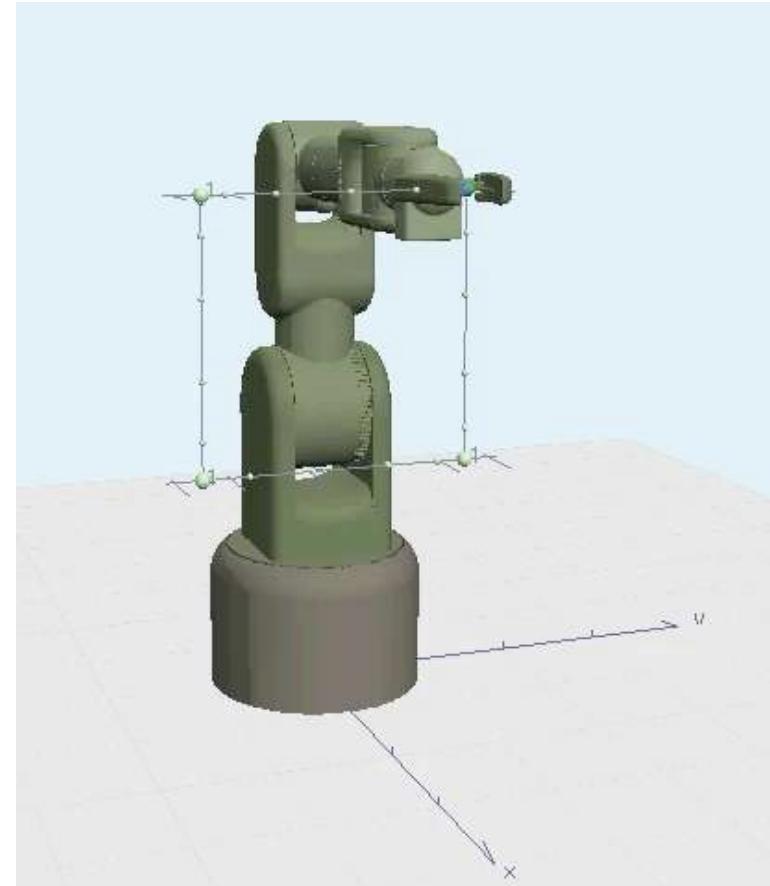
Some typical trajectories

- Timing laws: Cartesian path with (dis-)continuous tangent



video

Square path at constant speed



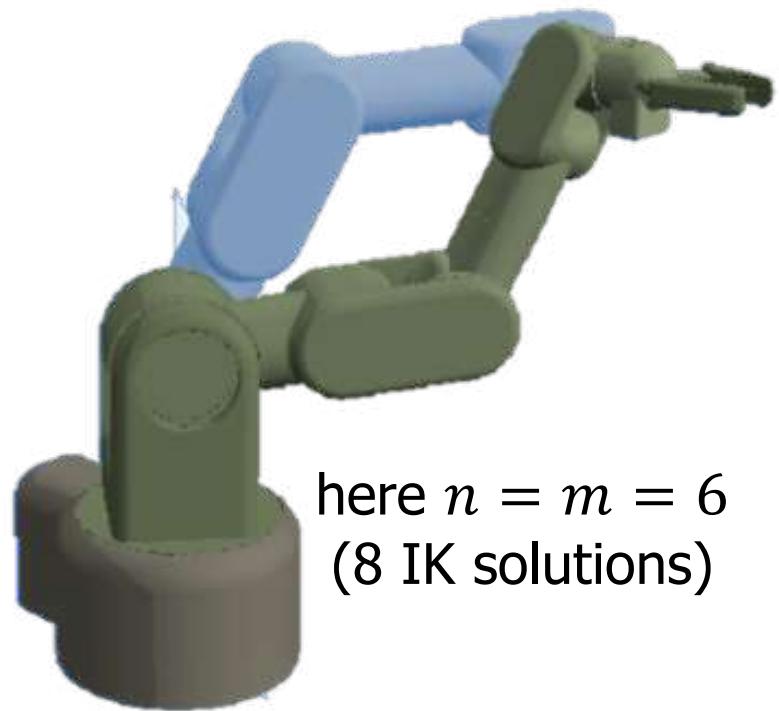
video

Square path with
trapezoidal speed profile



Joint and Cartesian trajectories

- assigned task: arm **reconfiguration** between two inverse kinematic solutions associated to a **given end-effector pose**



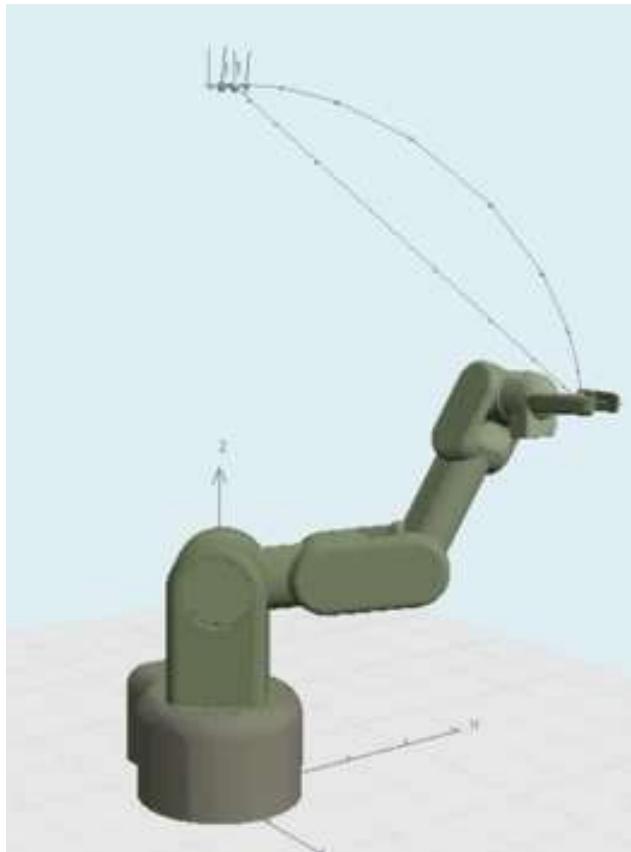
- **initial** and **final** configuration
- same Cartesian pose (no change!): the motion cannot be fully specified in the Cartesian space
- to perform this task, the robot should leave the given end-effector pose and then return to it
- a self-motion could be sufficient
 - if there is (task) redundancy ($m < n$)
 - if the robot starts in a singularity

for “simple” manipulators (e.g., all industrial robots) and $m = n$, the execution of these tasks will require the **passage through a singular configuration**

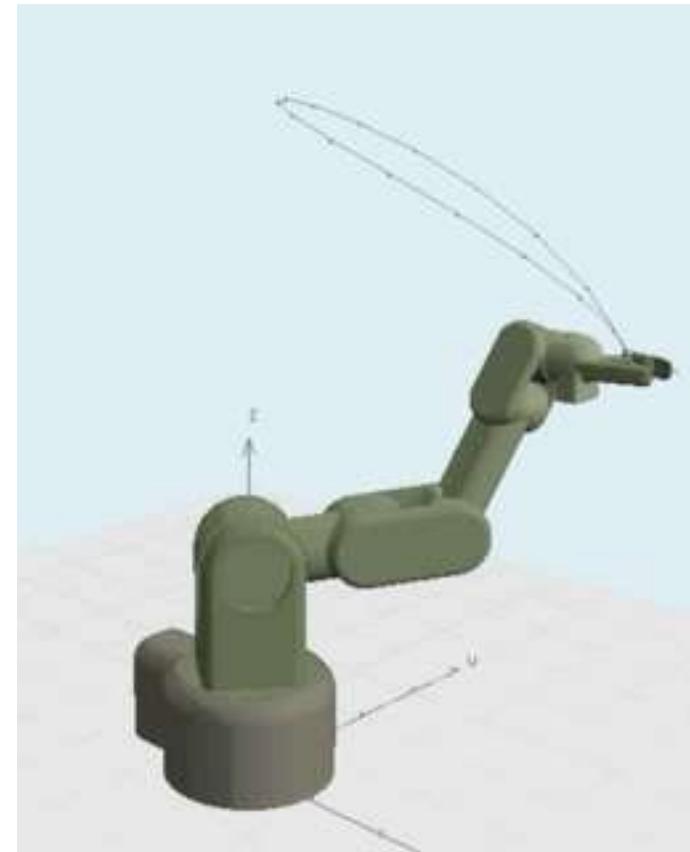


Joint and Cartesian trajectories

- a reconfiguration task (or... passing through singularity)



three-phase trajectory:
circular path + self-motion + linear path



single-phase trajectory
in the joint space (no stops)



From task to trajectory

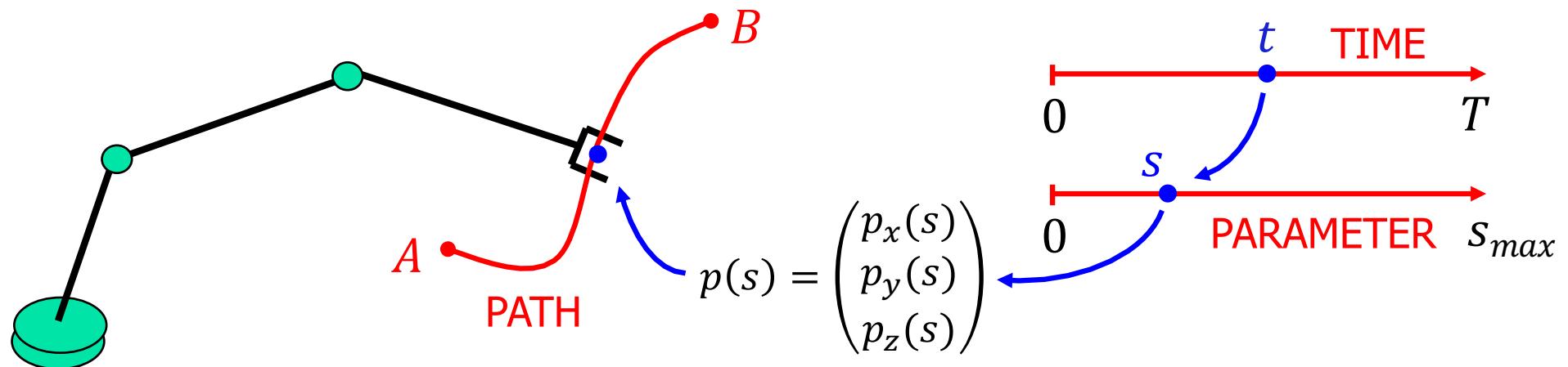
TRAJECTORY
||
GEOMETRIC PATH + TIMING LAW

of motion $p_d(t)$ (or $q_d(t)$)
of interaction $F_d(t)$

parameterized by s : $p = p(s)$
(e.g., s is the arc length)

describes the time evolution of $s = s(t)$

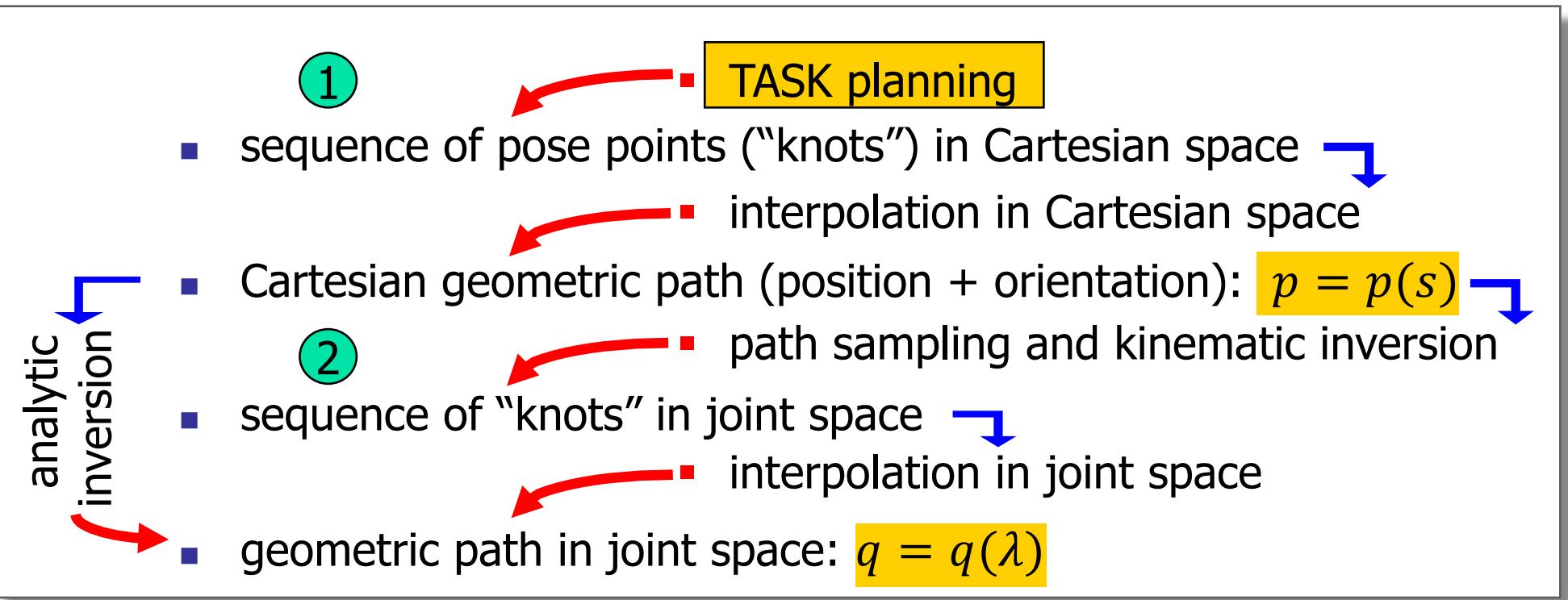
$p(s(t))$



example: TASK planner provides A, B
TRAJECTORY planner generates $p(t)$



Trajectory planning operative sequence

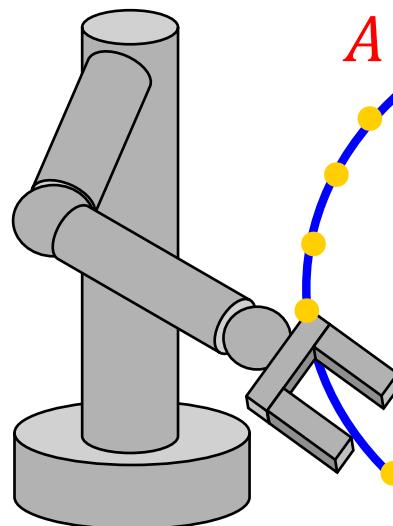


additional issues to be considered in the planning process

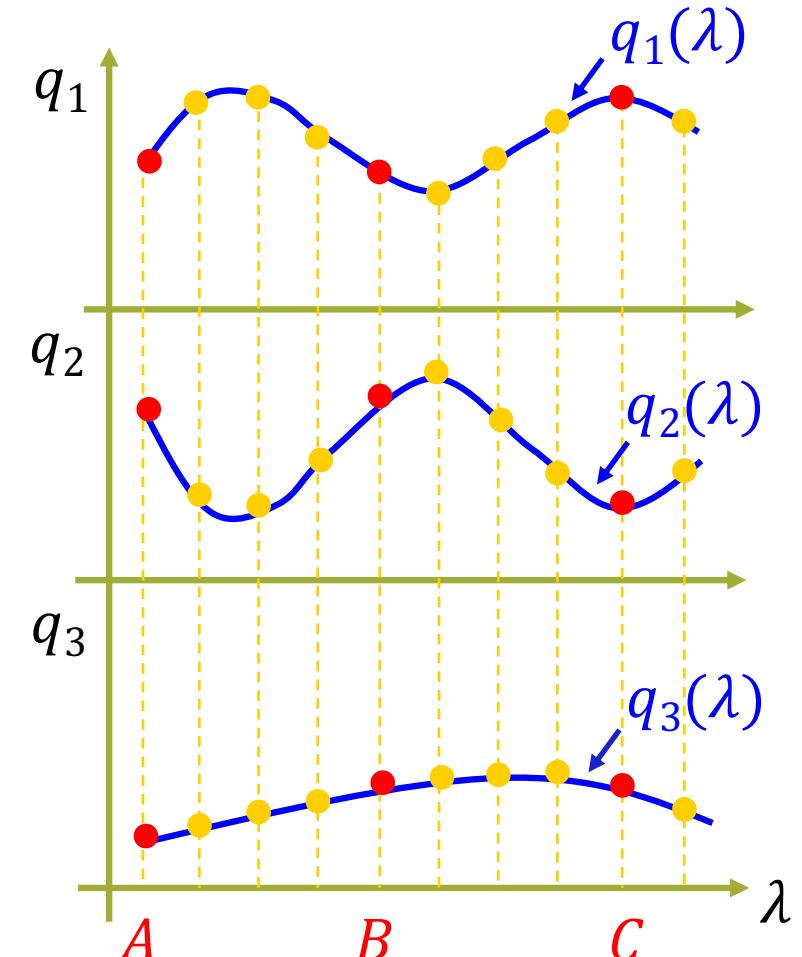
- obstacle avoidance
- on-line/off-line computational load
- sequence ② is more "dense" than ①



Example



Cartesian space



joint space



Trajectory classification

- space of definition
 - Cartesian, joint
- task type
 - point-to-point (PTP), multiple points (knots), continuous, concatenated
- path geometry
 - rectilinear, polynomial, exponential, cycloid, ...
- timing law
 - bang-bang in acceleration, trapezoidal in velocity, polynomial, ...
- coordinated or independent
 - motion of all joints (or of all Cartesian components) start and ends at the same instants (say, $t = 0$ and $t = T$) = single timing law
or
 - motions are timed independently (according to the requested displacement and robot capabilities) – mostly only in joint space



Path and timing law

- after choosing a **path**, the trajectory definition is completed by the choice of a **timing law**

$$p = p(s) \quad \Rightarrow s = s(t) \quad (\text{Cartesian space})$$

$$q = q(\lambda) \quad \Rightarrow \lambda = \lambda(t) \quad (\text{joint space})$$

- if $s(t) = t$, path parameterization is the **natural** one given by time
- the **timing law**
 - is chosen based on **task specifications** (stop in a point, move at constant velocity, and so on)
 - may consider **optimality criteria** (min transfer time, min energy,...)
 - **constraints** are imposed by actuator capabilities (max torque, max velocity,...) and/or by the task (e.g., max acceleration on payload)

note: on parameterized paths, a **space-time decomposition** takes place

e.g., in Cartesian space $\dot{p}(t) = \frac{dp}{ds} \dot{s}$ $\ddot{p}(t) = \frac{dp}{ds} \ddot{s} + \frac{d^2p}{ds^2} \dot{s}^2$



Cartesian vs. joint trajectory planning

- planning in **Cartesian space**
 - allows a more direct visualization of the generated path
 - obstacle avoidance, lack of “wandering”
- planning in **joint space**
 - does not need on-line kinematic inversion
- issues in kinematic inversion
 - \dot{q} and \ddot{q} (or higher-order derivatives) may also be needed
 - Cartesian task specifications involve the geometric path, but also bounds on the associated timing law
 - for redundant robots, choice among ∞^{n-m} inverse solutions, based on optimality criteria or additional auxiliary tasks
 - off-line planning in advance is not always feasible
 - e.g., when **environment interaction** occurs or when **sensor-based** motion is needed

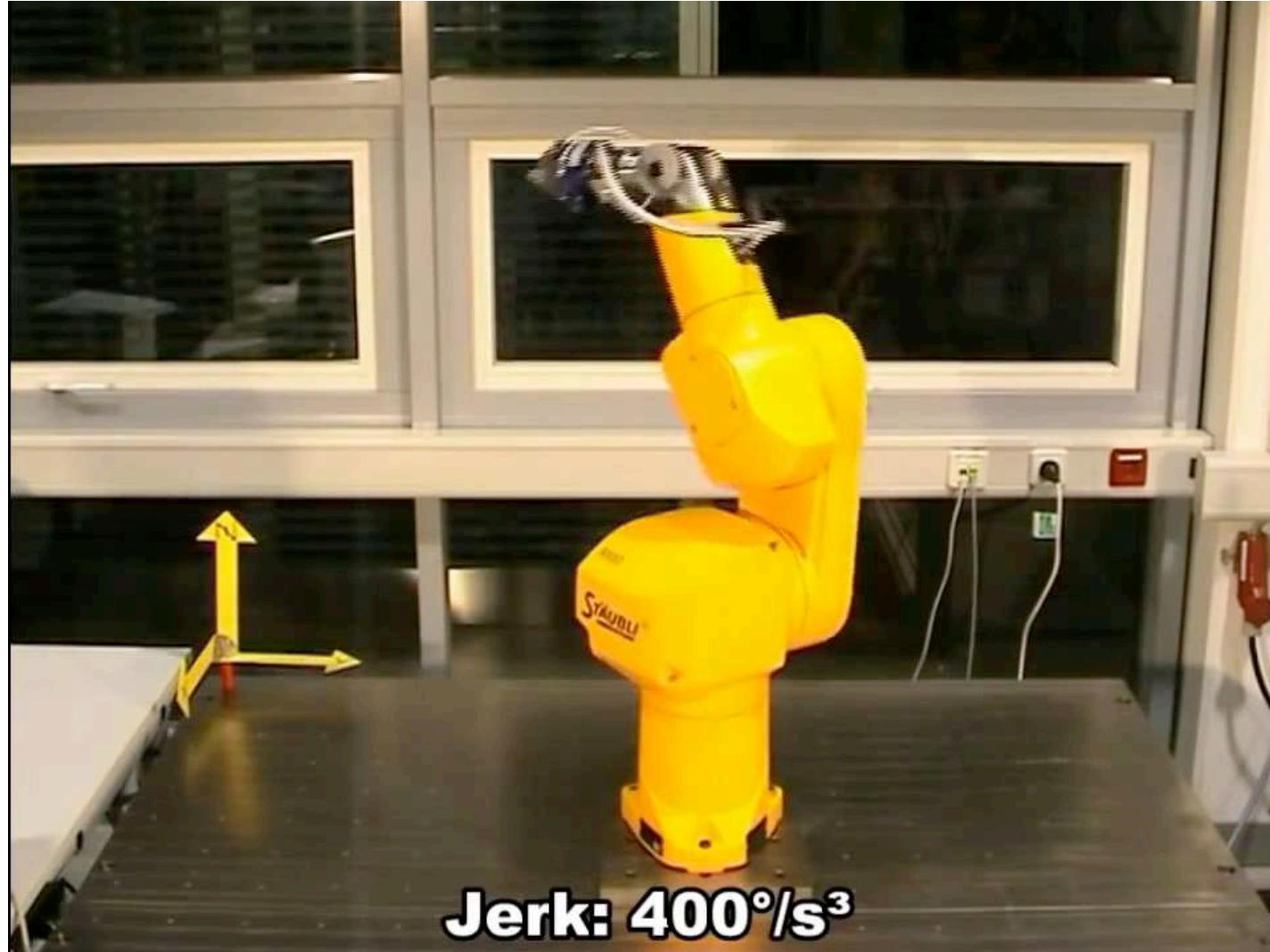


Relevant characteristics

- computational **efficiency** and **memory space**
 - e.g., store only the coefficients of a polynomial function
- **predictability** and **accuracy**
 - vs. “wandering” out of the knots
 - vs. “overshoot” on final position
- **flexibility**
 - allowing concatenation of primitive segments
 - over-fly
 - ...
- **continuity**
 - in space and/or in time
 - at least C^1 , but also up to jerk = third derivative in time



A robot trajectory with bounded jerk





Trajectory planning in joint space

- $q = q(t)$ in **time** or $q = q(\lambda)$ in **space** (then with $\lambda = \lambda(t)$)
- it is sufficient to work **component-wise** (q_i in vector q)
- an **implicit** definition of the trajectory, by solving a problem with specified **boundary conditions** in a given **class of functions**
- typical classes: **polynomials** (cubic, quintic,...), trigonometric (cosine, sines, combined, ...), clothoids, ...
- **imposed conditions**
 - passage through points = interpolation
 - initial, final, intermediate velocity (or **geometric tangent for paths**)
 - initial, final acceleration (or **geometric curvature**)
 - continuity up to the k -th order time (or **space**) derivative: class C^k

many of the following methods and remarks can be directly applied also to Cartesian trajectory planning (and vice versa)!



Cubic polynomial in space

$$q(0) = q_0 \quad q(1) = q_1 \quad q'(0) = v_0 \quad q'(1) = v_1 \quad \leftarrow \text{4 conditions}$$

$$q(\lambda) = q_0 + \Delta q \underbrace{(a\lambda^3 + b\lambda^2 + c\lambda + d)}_{\lambda \in [0,1]} \quad \begin{aligned} \Delta q &= q_1 - q_0 \\ \lambda &\in [0,1] \end{aligned}$$

4 coefficients → “doubly normalized” polynomial $q_N(\lambda)$

$$q_N(0) = 0 \Leftrightarrow d = 0 \quad q_N(1) = 1 \Leftrightarrow a + b + c = 1$$

$$q'_N(0) = dq_N/d\lambda|_{\lambda=0} = c = v_0/\Delta q \quad q'_N(1) = dq_N/d\lambda|_{\lambda=1} = 3a + 2b + c = v_1/\Delta q$$

special case: $v_0 = v_1 = 0$ (zero tangent)

$$q'_N(0) = 0 \Leftrightarrow c = 0$$

$$\left. \begin{array}{l} q_N(1) = 1 \Leftrightarrow a + b = 1 \\ q'_N(1) = 0 \Leftrightarrow 3a + 2b = 0 \end{array} \right\} \Leftrightarrow \begin{array}{l} a = -2 \\ b = 3 \end{array}$$



Cubic polynomial in time

$$q(0) = q_{in}$$

$$q(T) = q_{fin}$$

$$\dot{q}(0) = v_{in}$$

$$\dot{q}(T) = v_{fin}$$

← 4 conditions

$$q(\tau) = q_{in} + \Delta q \underbrace{\left(a\tau^3 + b\tau^2 + c\tau + d \right)}_{}$$

$$\Delta q = q_{fin} - q_{in}$$

$$\tau = t/T \in [0,1]$$

4 coefficients → “doubly normalized” polynomial $q_N(\tau)$

$$q_N(0) = 0 \Leftrightarrow d = 0$$

$$q_N(1) = 1 \Leftrightarrow a + b + c = 1$$

$$q'_N(0) = dq_N/d\tau|_{\tau=0} = c = \frac{v_{in}T}{\Delta q} \quad q'_N(1) = dq_N/d\tau|_{\tau=1} = 3a + 2b + c = \frac{v_{fin}T}{\Delta q}$$

special case: $v_{in} = v_{fin} = 0$ (rest-to-rest)

$$q'_N(0) = 0 \Leftrightarrow c = 0$$

$$q_N(1) = 1 \Leftrightarrow a + b = 1$$

$$q'_N(1) = 0 \Leftrightarrow 3a + 2b = 0$$

$$\left. \begin{array}{l} \\ \\ \end{array} \right\} \Leftrightarrow \begin{array}{l} a = -2 \\ b = 3 \end{array}$$



Quintic polynomial

$$q(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f \quad \text{6 coefficients}$$

$$\tau \in [0, 1]$$

allows to satisfy **6 conditions**, for example (in normalized time $\tau = t/T$)

$$q(0) = q_0$$

$$q(1) = q_1$$

$$q'(0) = v_0 T$$

$$q'(1) = v_1 T$$

$$q''(0) = a_0 T^2$$

$$q''(1) = a_1 T^2$$

$$q(\tau) = (1 - \tau)^3(q_0 + (3q_0 + v_0 T)\tau + (a_0 T^2 + 6v_0 T + 12q_0)\tau^2/2) \\ + \tau^3(q_1 + (3q_1 - v_1 T)(1 - \tau) + (a_1 T^2 - 6v_1 T + 12q_1)(1 - \tau)^2/2)$$

special case: $v_0 = v_1 = a_0 = a_1 = 0$

$$q(\tau) = q_0 + \Delta q(6\tau^5 - 15\tau^4 + 10\tau^3)$$

$$\Delta q = q_1 - q_0$$



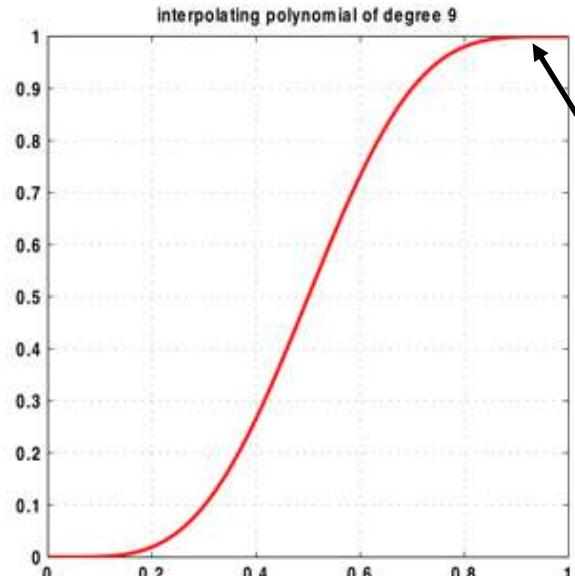
Higher-order polynomials

- a suitable solution class for satisfying **symmetric** boundary conditions (in a PTP motion) that **impose zero** values on higher-order derivatives
 - the interpolating polynomial is always of **odd** degree
 - the coefficients of such (**doubly normalized**) polynomial are always **integers**, **alternate in sign**, sum up to unity, and are zero for all terms up to the power = (degree-1)/2
- in all other cases (e.g., for interpolating a large number N of points), their use is **not** recommended
 - N -th order polynomials have $N - 1$ maximum and minimum points
 - oscillations arise out of the interpolation points (**wandering**)



Numerical examples

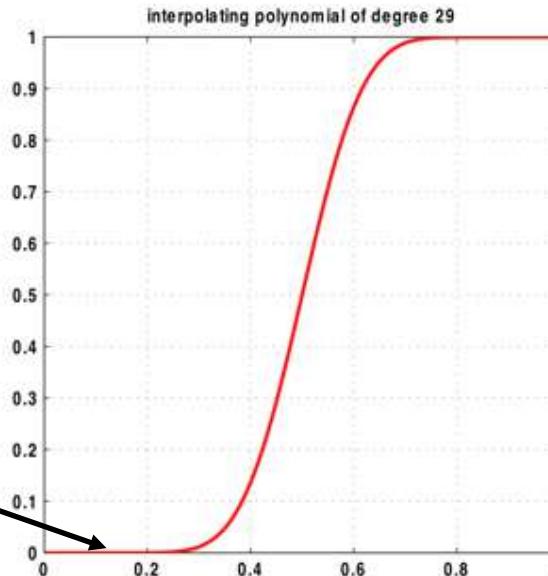
9th
degree



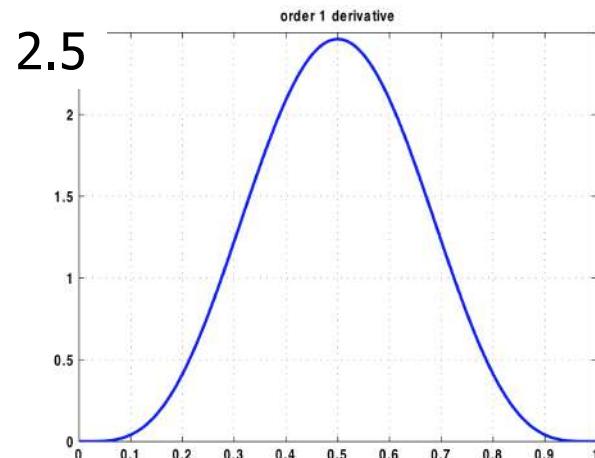
4 derivatives
are zero

14 derivatives
are zero!

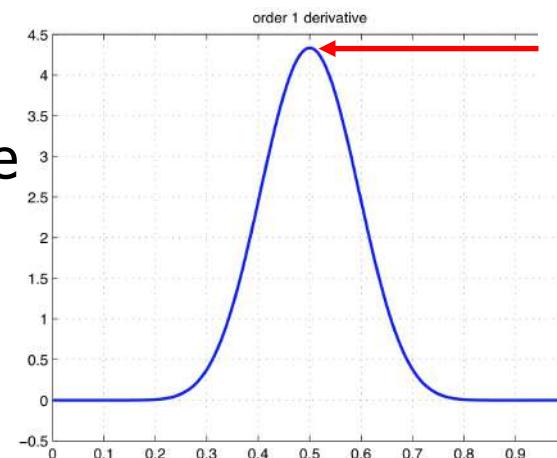
29th
degree



no
overshoot
nor
wandering



normalized
first derivative
(velocity
in time)

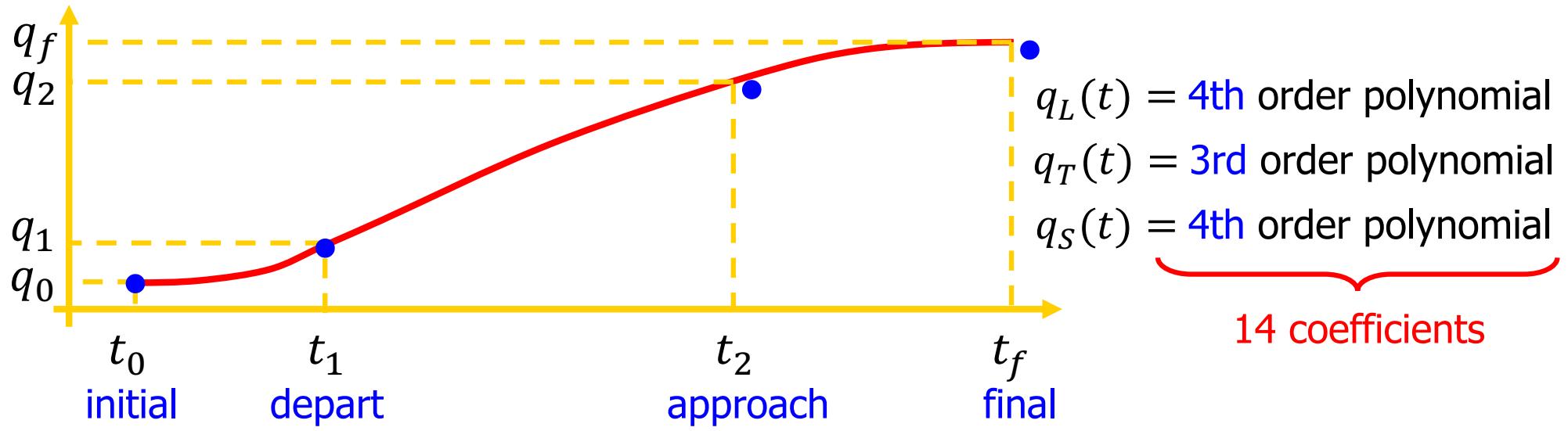


peaking
at midpoint



4-3-4 polynomials

three phases (Lift off, Travel, Set down) in a pick-and-place operation in time



$$q(t_0) = q_0 \quad q(t_1^-) = q(t_1^+) = q_1 \quad q(t_2^-) = q(t_2^+) = q_2 \quad q(t_f) = q_f \quad } \quad 6 \text{ passages}$$

$$\dot{q}(t_0) = \dot{q}(t_f) = 0 \quad \ddot{q}(t_0) = \ddot{q}(t_f) = 0 \quad } \quad 4 \text{ initial/final velocity/acceleration}$$

$$\dot{q}(t_i^-) = \dot{q}(t_i^+) \quad \ddot{q}(t_i^-) = \ddot{q}(t_i^+) \quad i = 1,2 \quad } \quad 4 \text{ continuity up to acceleration}$$

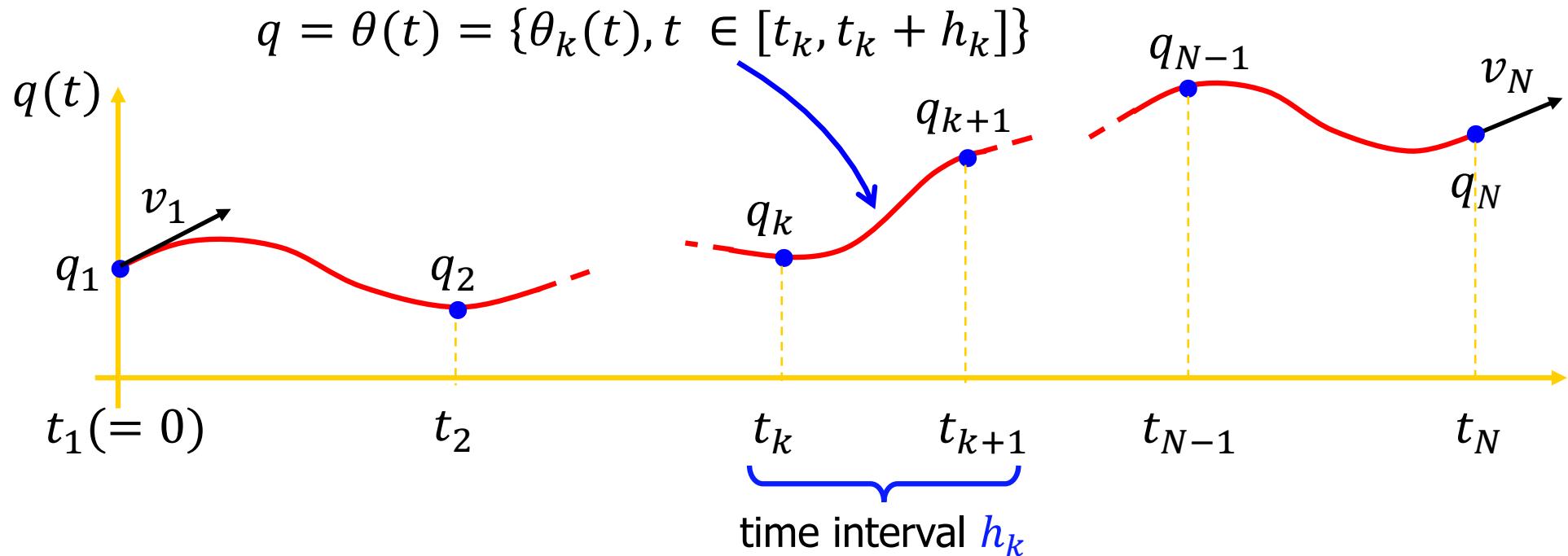


Interpolation using splines

- **problem**
interpolate N knots, with continuity up to the second derivative
- **solution**
spline: $N - 1$ cubic polynomials, concatenated so to pass through N knots, and continuous up to the second derivative at the $N - 2$ internal knots
- **$4(N - 1)$ coefficients**
- **$4(N - 1) - 2$ conditions**, or
 - $2(N - 1)$ of passage (for each cubic, in the two knots at its ends)
 - $N - 2$ of continuity for first derivative (at the internal knots)
 - $N - 2$ of continuity for second derivative (at the internal knots)
- **2 free parameters** are still left over
 - can be used, e.g., to assign initial and final derivatives, v_1 and v_N
- presented next in terms of **time t** , but similar in terms of **space λ**
 - **then**: first derivative = velocity, second derivative = acceleration



Building a cubic spline



$$\theta_k(\tau) = a_{k0} + a_{k1}\tau + a_{k2}\tau^2 + a_{k3}\tau^3 \quad \tau = t - t_k \in [0, h_k] \quad (k = 1, \dots, N-1)$$

continuity conditions
for velocity and acceleration



$$\begin{aligned} \dot{\theta}_k(h_k) &= \dot{\theta}_{k+1}(0) & k &= 1, \dots, N-2 \\ \ddot{\theta}_k(h_k) &= \ddot{\theta}_{k+1}(0) \end{aligned}$$



An efficient algorithm

- if all **velocities** v_k at **internal knots** were known, then each cubic in the spline would be uniquely determined by

$$\begin{aligned}\theta_k(0) &= q_k = a_{k0} & \begin{pmatrix} h_k^2 & h_k^3 \\ 2h_k & 3h_k^2 \end{pmatrix} \begin{pmatrix} a_{k2} \\ a_{k3} \end{pmatrix} &= \begin{pmatrix} q_{k+1} - q_k - v_k h_k \\ v_{k+1} - v_k \end{pmatrix} \quad 1 \\ \dot{\theta}_k(0) &= v_k = a_{k1} \end{aligned}$$

- impose the **continuity for accelerations** ($N - 2$ conditions)

$$\ddot{\theta}_k(h_k) = 2a_{k2} + 6a_{k3}h_k = 2a_{k+1,2} = \ddot{\theta}_{k+1}(0)$$

- expressing the coefficients $a_{k2}, a_{k3}, a_{k+1,2}$ in terms of the **still unknown** knot velocities (see step 1.) yields a linear system of equations that is always solvable

$$\left(\begin{array}{c} \text{tri-diagonal matrix} \\ \text{always invertible} \end{array} \right) \left(\begin{array}{c} v_2 \\ v_3 \\ \vdots \\ v_{N-1} \end{array} \right) = \left(\begin{array}{c} \text{known vector} \\ \text{to be substituted then back in } 1 \end{array} \right)$$

$\text{A}(h_1, \dots, h_{N-1})$

unknown



Structure of $A(\mathbf{h})$

$$\begin{pmatrix} 2(h_1 + h_2) & h_1 & & & \\ h_3 & 2(h_2 + h_3) & h_2 & & \\ & \dots & & & \\ & & \dots & & \\ & & & h_{N-2} & 2(h_{N-3} + h_{N-2}) & h_{N-3} \\ & & & & h_{N-1} & & 2(h_{N-2} + h_{N-1}) \end{pmatrix}$$

diagonally dominant matrix (for $h_k > 0$)
[the same tridiagonal matrix for all joints]



Structure of $b(\mathbf{h}, \mathbf{q}, \mathbf{v}_1, \mathbf{v}_N)$

$$\left(\begin{array}{c} \frac{3}{h_1 h_2} (h_1^2 (q_3 - q_2) + h_2^2 (q_2 - q_1)) - h_2 v_1 \\ \frac{3}{h_2 h_3} (h_2^2 (q_4 - q_3) + h_3^2 (q_3 - q_2)) \\ \vdots \\ \vdots \\ \frac{3}{h_{N-3} h_{N-2}} (h_{N-3}^2 (q_{N-1} - q_{N-2}) + h_{N-2}^2 (q_{N-2} - q_{N-3})) \\ \frac{3}{h_{N-2} h_{N-1}} (h_{N-2}^2 (q_N - q_{N-1}) + h_{N-1}^2 (q_{N-1} - q_{N-2})) - h_{N-2} v_N \end{array} \right)$$



Properties of splines

- a spline (in **space**) is the solution with **minimum curvature** among all interpolating functions having continuous second derivative
- for **cyclic** tasks ($q_1 = q_N$), it is preferable to simply impose continuity of first and second derivatives (i.e., velocity and acceleration in time) at the first/last knot as “squaring” conditions
 - choosing $v_1 = v_N = v$ (for a given v) doesn't guarantee in general the continuity up to the second derivative (in time, of the acceleration)
 - in this way, the first = last knot will be handled as all other internal knots
- a spline is **uniquely** determined from the set of data q_1, \dots, q_N ,
 $h_1, \dots, h_{N-1}, v_1, v_N$
- in **time**, the total motion occurs in $T = \sum_k h_k = t_N - t_1$
- the time intervals h_k can be chosen so as to **minimize T** (linear objective function) under (nonlinear) **bounds** on velocity and acceleration in $[0, T]$
- in **time**, the spline construction can be suitably **modified** when the **acceleration** is also assigned at the initial and final knots



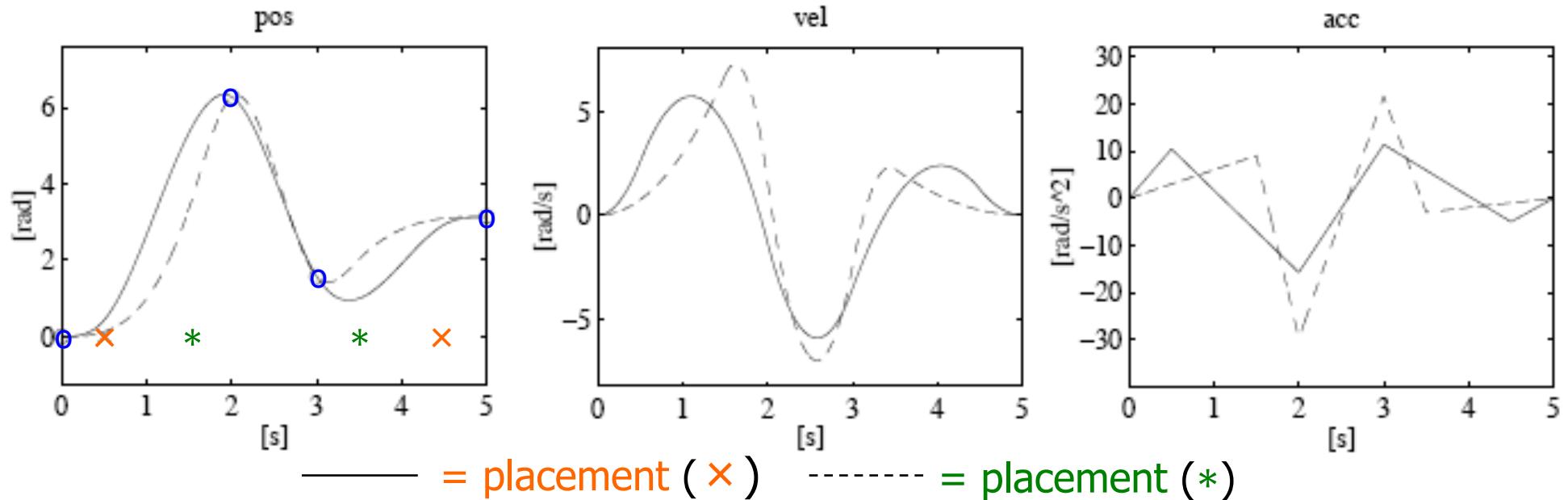
A modification handling assigned initial and final accelerations

- two more parameters are needed in order to impose also the initial acceleration α_1 and final acceleration α_N
- two “**fictitious knots**” are inserted in the first and the last original intervals, increasing the number of cubic polynomials from $N - 1$ to $N + 1$
- in these two knots **only continuity** conditions on **position**, **velocity** and **acceleration** are imposed
 - ⇒ **two** free parameters are left over (one in the first cubic and the other in the last cubic), which are used to satisfy the boundary conditions on acceleration
- depending on the (time) placement of the two additional knots, the resulting spline changes



A numerical example

- $N = 4$ knots (o) \Rightarrow 3 cubic polynomials
 - joint values $q_1 = 0, q_2 = 2\pi, q_3 = \pi/2, q_4 = \pi$
 - at $t_1 = 0, t_2 = 2, t_3 = 3, t_4 = 5 \Rightarrow h_1 = 2, h_2 = 1, h_3 = 2$
 - boundary velocities $v_1 = v_4 = 0$
- 2 added knots to impose accelerations at both ends (5 cubic polynomials)
 - boundary accelerations $\alpha_1 = \alpha_2 = 0$
 - two placements: at $t'_1 = 0.5$ and $t'_3 = 4.5$ (x); or at $t''_1 = 1.5$ and $t''_4 = 3.5$ (*)





Robotics 1

Trajectory planning in Cartesian space

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



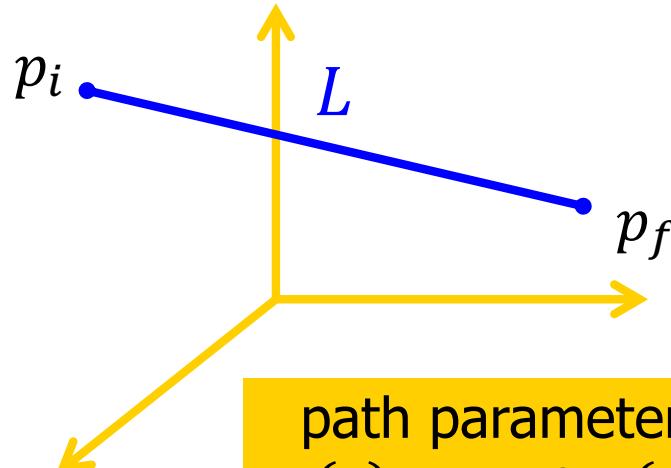


Trajectories in Cartesian space

- in general, the trajectory planning methods proposed in the joint space can be applied also in the Cartesian space
 - consider **independently** each component of the task vector (i.e., a position or an angle of a minimal representation of orientation)
- however, when planning a trajectory for the three orientation angles, the resulting global motion cannot be intuitively **visualized** in advance
- if possible, we still prefer to plan Cartesian trajectories **separately** for **position** and **orientation**
- the number of knots to be interpolated in the Cartesian space is typically low (e.g., 2 knots for a PTP motion, 3 if a “via point” is added) ⇒ use **simple** interpolating paths, such as straight lines, arc of circles, ...



Planning a linear Cartesian path (position only)



GIVEN
 $p_i, p_f \in \mathbb{R}^3; v_i, v_f \in \mathbb{R}$ (typically = 0);
 bounds $v_{max}, a_{max} \in \mathbb{R}^+$

path parameterization
 $p(s) = p_i + s(p_f - p_i)$

$s \in [0,1]$

$L = \|p_f - p_i\|$
 $\frac{p_f - p_i}{\|p_f - p_i\|}$ = unit vector of directional cosines of the line

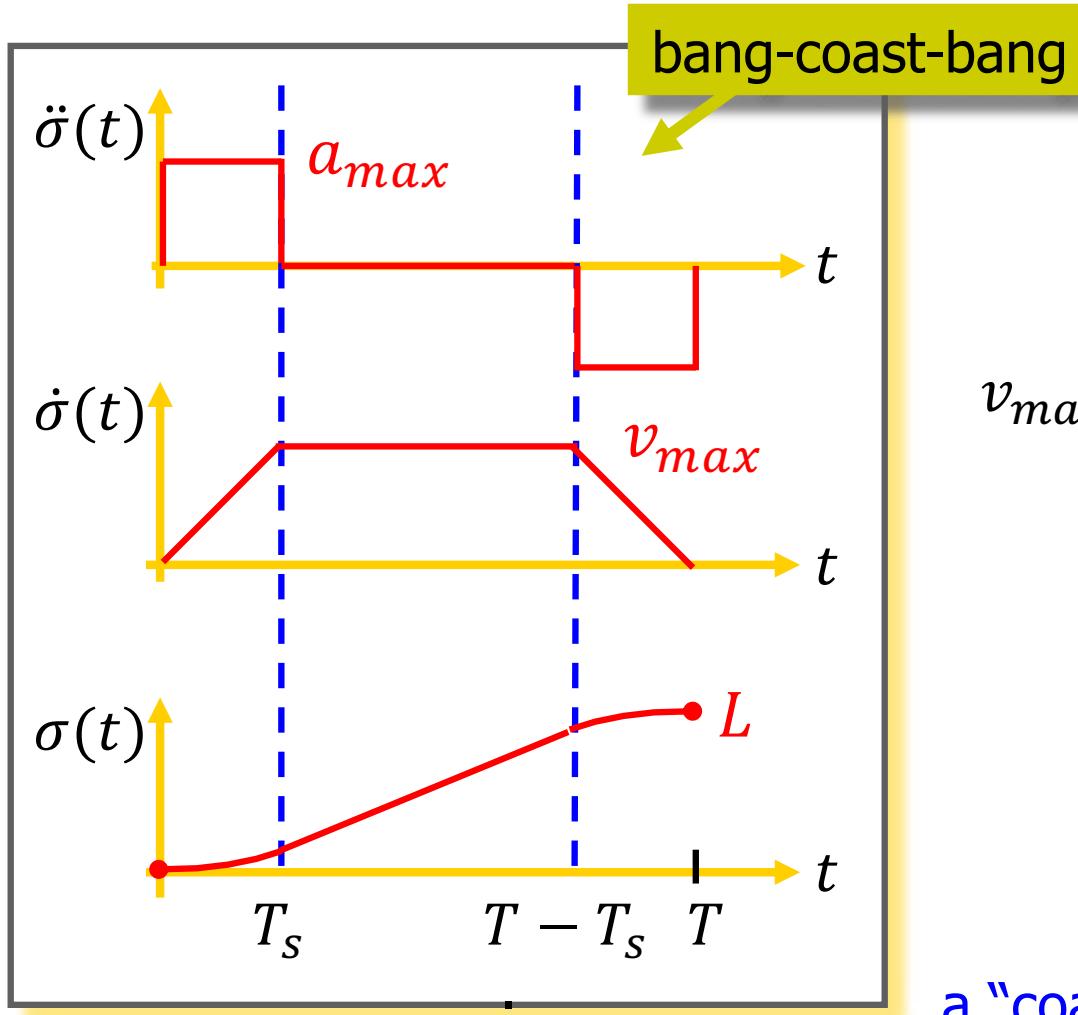
may also use $s = \sigma/L$, where $\sigma \in [0, L]$ is the arc length (gives the current length of the path)

$$\begin{aligned}\dot{p}(s) &= \frac{dp}{ds} \dot{s} = (p_f - p_i) \dot{s} \\ &= \frac{p_f - p_i}{L} \dot{\sigma}\end{aligned}$$

$$\begin{aligned}\ddot{p}(s) &= \frac{d^2p}{ds^2} \dot{s}^2 + \frac{dp}{ds} \ddot{s} = (p_f - p_i) \ddot{s} \\ &= \frac{p_f - p_i}{L} \ddot{\sigma}\end{aligned}$$



Timing law with trapezoidal speed - 1



given*: L, v_{max}, a_{max}
find: T_s, T

$$v_{max} (T - T_s) = L \quad (= \text{area of the speed profile})$$

$$T_s = \frac{v_{max}}{a_{max}}$$

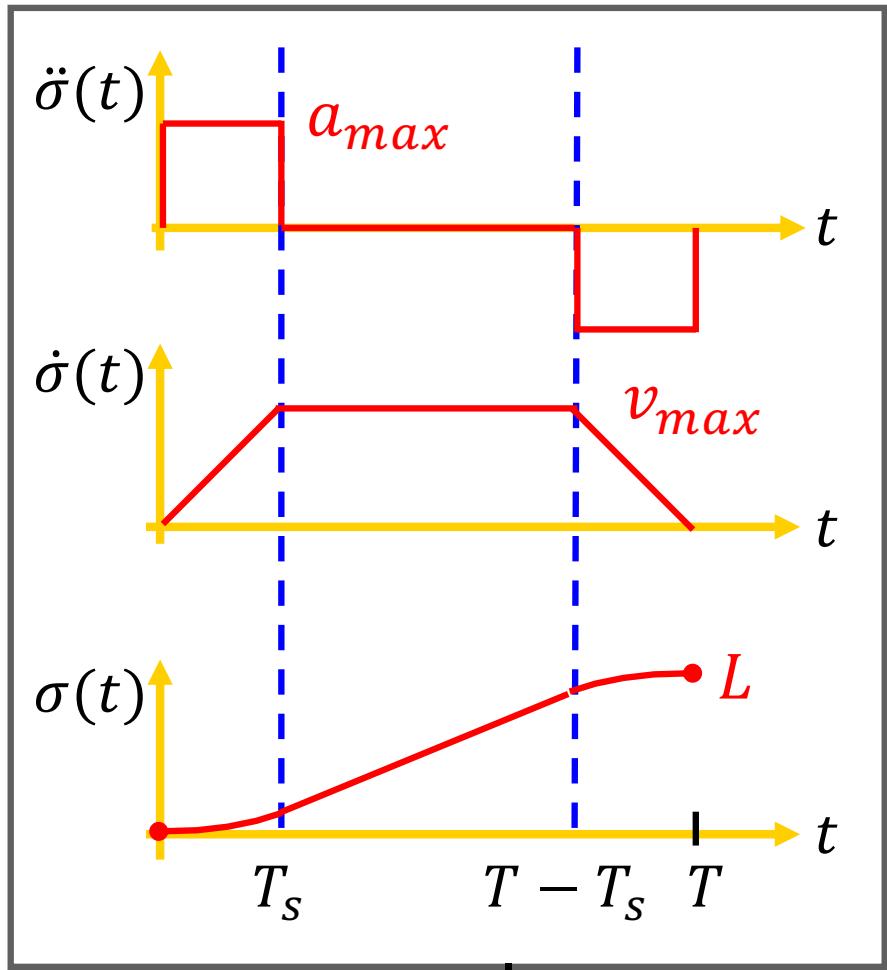
$$T = \frac{La_{max} + v_{max}^2}{a_{max}v_{max}}$$

a “coast” phase exists iff $L > v_{max}^2/a_{max}$

* = other input data combinations are possible (see textbook)



Timing law with trapezoidal speed - 2



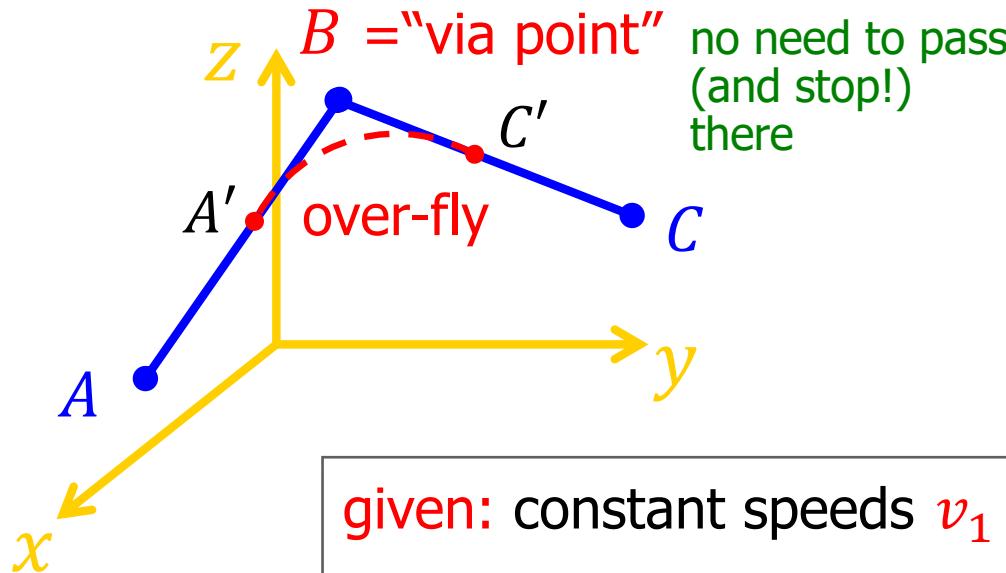
$$\sigma(T) = \begin{cases} \frac{a_{max}t^2}{2}, & t \in [0, T_s] \\ v_{max}t - \frac{v_{max}^2}{2a_{max}}, & t \in [T_s, T - T_s] \\ -\frac{a_{max}(t-T)^2}{2} + v_{max}T - \frac{v_{max}^2}{a_{max}}, & t \in [T - T_s, T] \end{cases}$$

discontinuous acceleration profile!
if needed, use for instance a
a rest-to-rest quintic polynomial timing

can be used also in the joint space!



Concatenation of linear paths



$$\frac{B - A}{\|B - A\|} = K_{AB}$$

$$\frac{C - B}{\|C - B\|} = K_{BC}$$

unit vectors of directional cosines

given: constant speeds v_1 on linear path AB
 v_2 on linear path BC

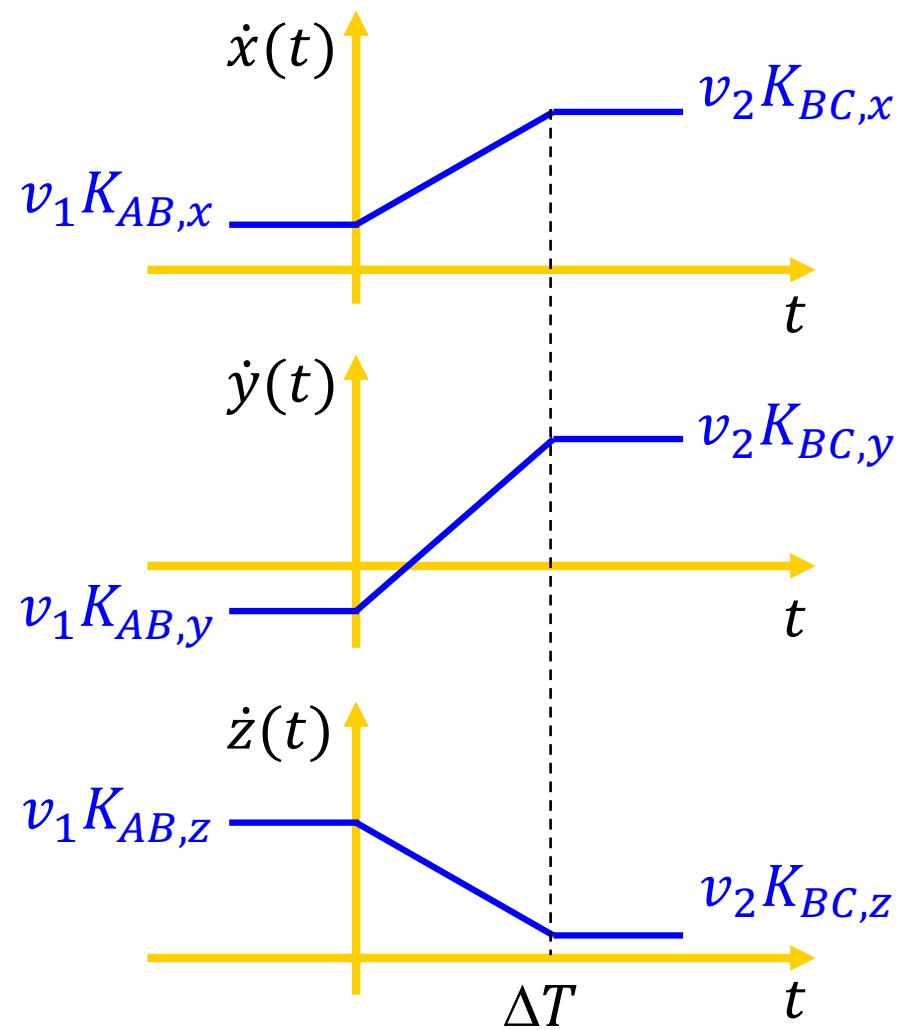
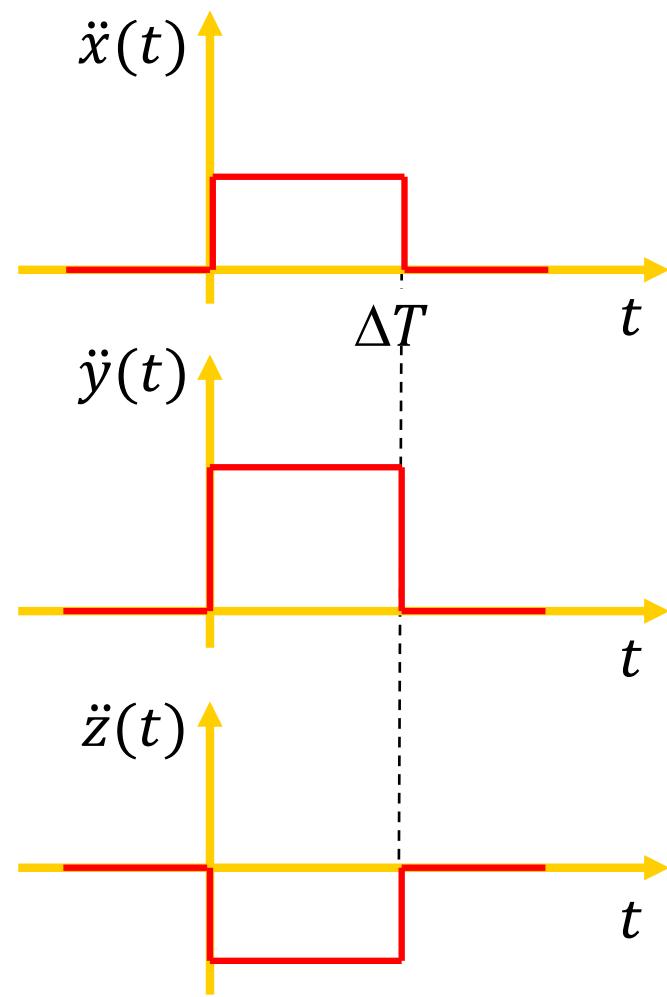
desired transition: with constant acceleration for a time ΔT

$$p(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} \quad t \in [0, \Delta T] \quad (\text{transition starts at } t = 0)$$

note: during over-fly, the path remains always in the plane specified by the two lines intersecting at B (in essence, it is a **planar problem**)

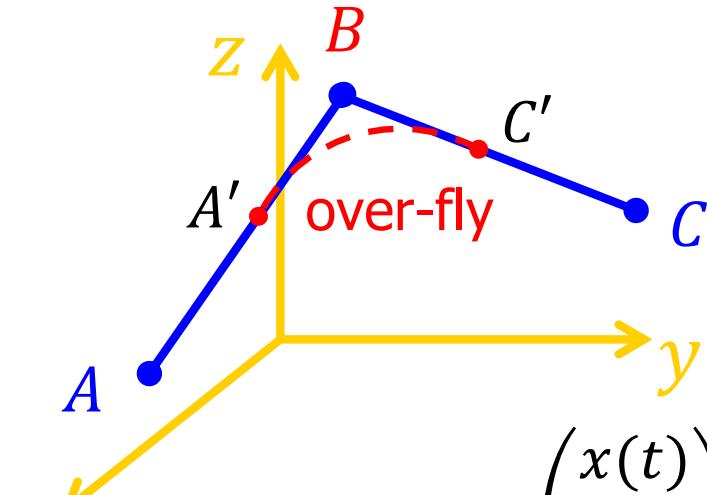


Time profiles on components





Timing law during transition



$$p(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$

$$\frac{B - A}{\|B - A\|} = K_{AB}$$

$$\frac{C - B}{\|C - B\|} = K_{BC}$$

unit vectors of
directional cosines

$t \in [0, \Delta T]$ (transition starts at $t = 0$)

$$\ddot{p}(t) = (\nu_2 K_{BC} - \nu_1 K_{AB})/\Delta T \quad \boxed{\int} \rightarrow \dot{p}(t) = \nu_1 K_{AB} + (\nu_2 K_{BC} - \nu_1 K_{AB})t/\Delta T$$

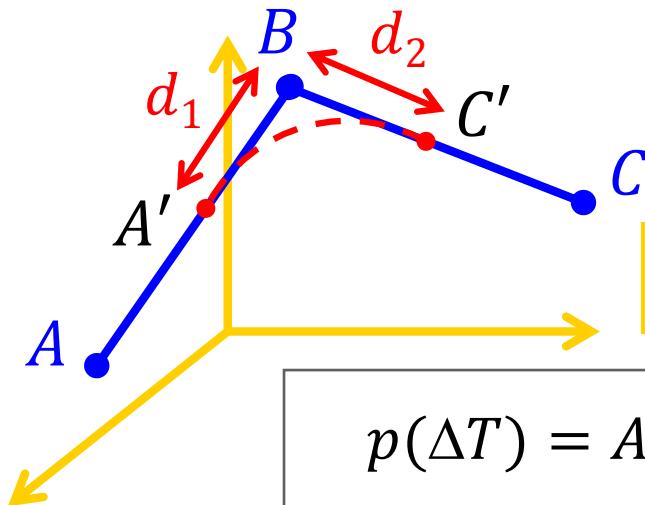
$$\boxed{\int}$$

$$p(t) = A' + \nu_1 K_{AB} t + (\nu_2 K_{BC} - \nu_1 K_{AB})t^2/(2\Delta T)$$

thus, we obtain a
parabolic blending
(see textbook
for this same approach
in the joint space)



Solution (various options)



$$B - A' = d_1 K_{AB}$$

$$C' - B = d_2 K_{BC}$$

1

$$p(t) = A' + v_1 K_{AB} t + (v_2 K_{BC} - v_1 K_{AB}) t^2 / (2\Delta T)$$

$$p(\Delta T) = A' + (\Delta T/2)(v_1 K_{AB} + v_2 K_{BC}) = C'$$

$$\rightarrow -B + A' + (\Delta T/2)(v_1 K_{AB} + v_2 K_{BC}) = C' - B$$

1

$$d_1 K_{AB} + d_2 K_{BC} = (\Delta T/2)(v_1 K_{AB} + v_2 K_{BC})$$

$$d_1 = v_1 \Delta T / 2$$

$$d_2 = v_2 \Delta T / 2$$

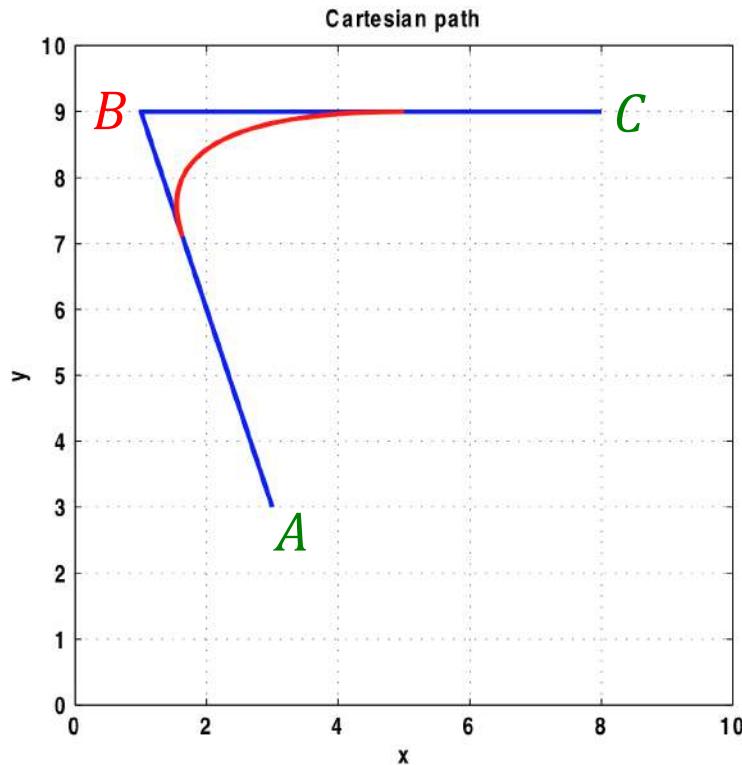
by choosing, e.g., d_1
(namely A')

$$\Delta T = 2d_1/v_1 \rightarrow d_2 = d_1 v_2 / v_1$$

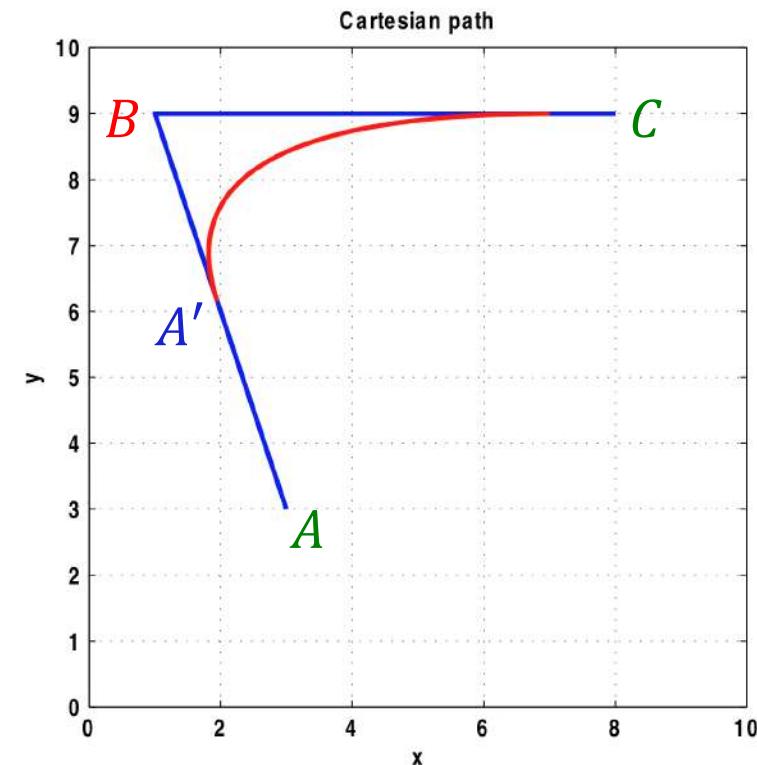


A numerical example

- transition: $A = (3,3)$ to $C = (8,9)$ via $B = (1,9)$, with speed from $v_1 = 1$ to $v_2 = 2$
- exploiting **two options** for solution (resulting in **different paths!**)
 - assign transition time: $\Delta T = 4$ (we re-center it here for $t \in [-\Delta T/2, \Delta T/2]$)
 - assign distance from B for departing: $d_1 = 3$ (assign d_2 for landing is handled similarly)



$$\Delta T = 4$$

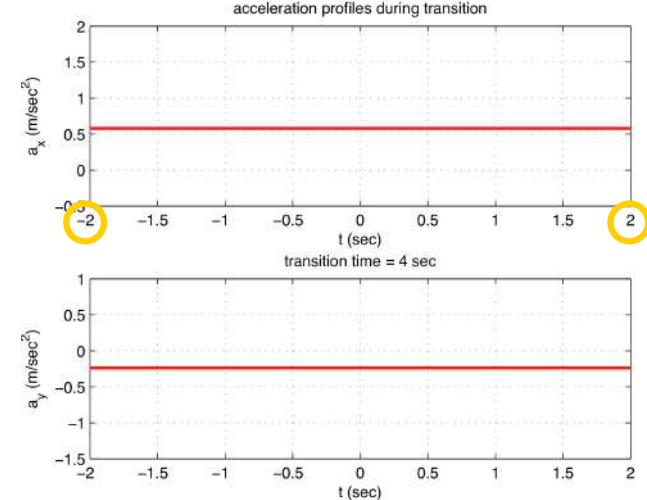
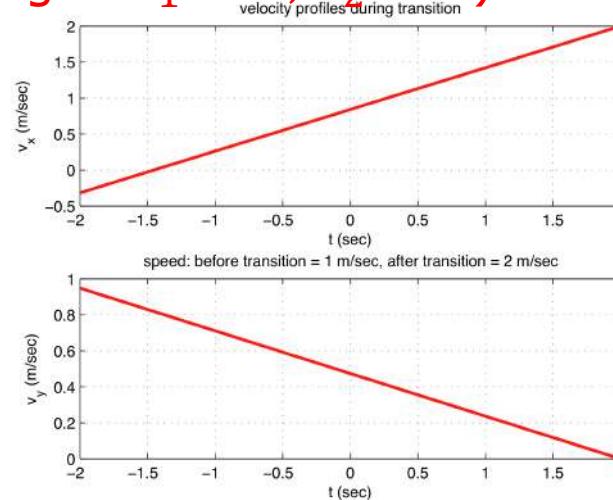
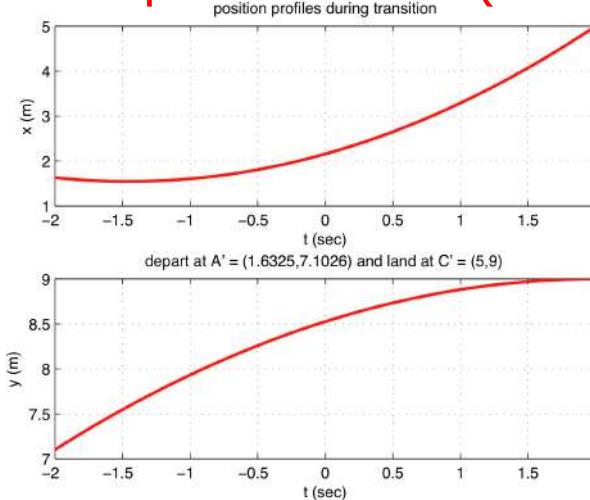


$$d_1 = 3$$

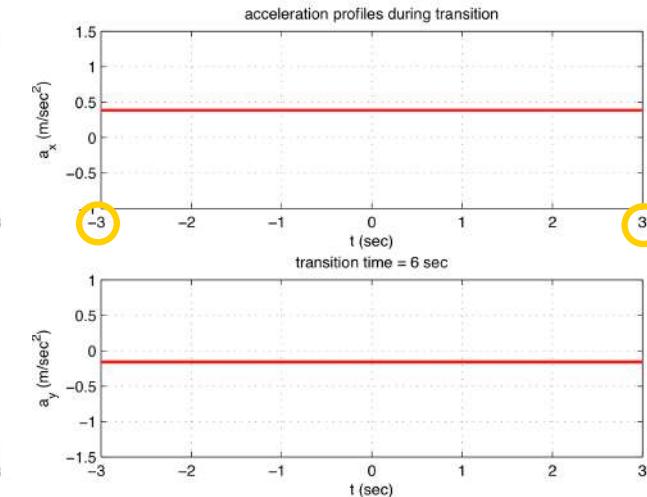
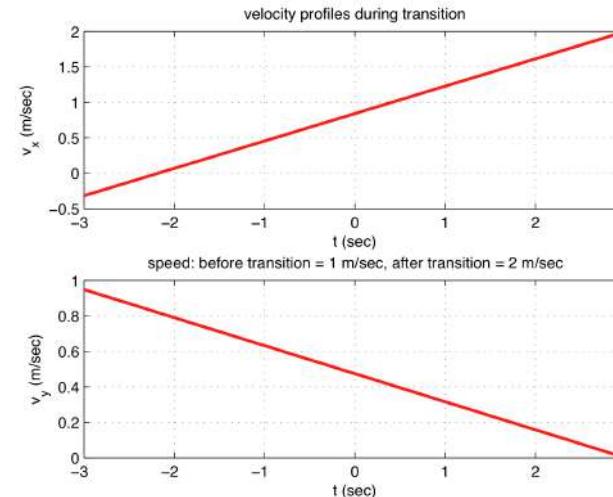
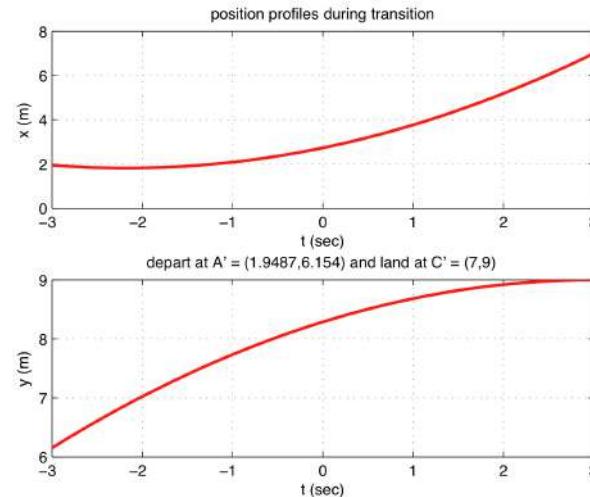


A numerical example (cont'd)

first option: $\Delta T = 4$ (resulting in $d_1 = 2, d_2 = 4$)



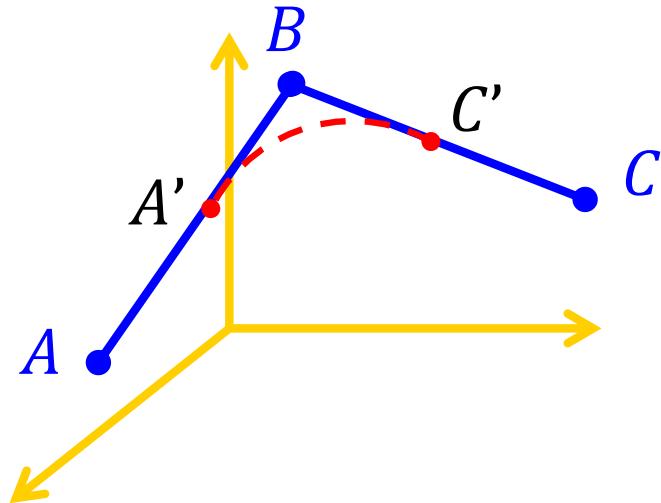
second option: $d_1 = 3$ (resulting in $\Delta T = 6, d_2 = 6$)



actually: the same velocity/acceleration profiles, only with a different time scale!!



Alternative solution (imposing acceleration)



$$\ddot{p}(t) = (v_2 K_{BC} - v_1 K_{AB}) / \Delta T$$

$v_1 = v_2 = v_{max}$ (for simplicity)

$$\|\ddot{p}(t)\| = a_{max}$$

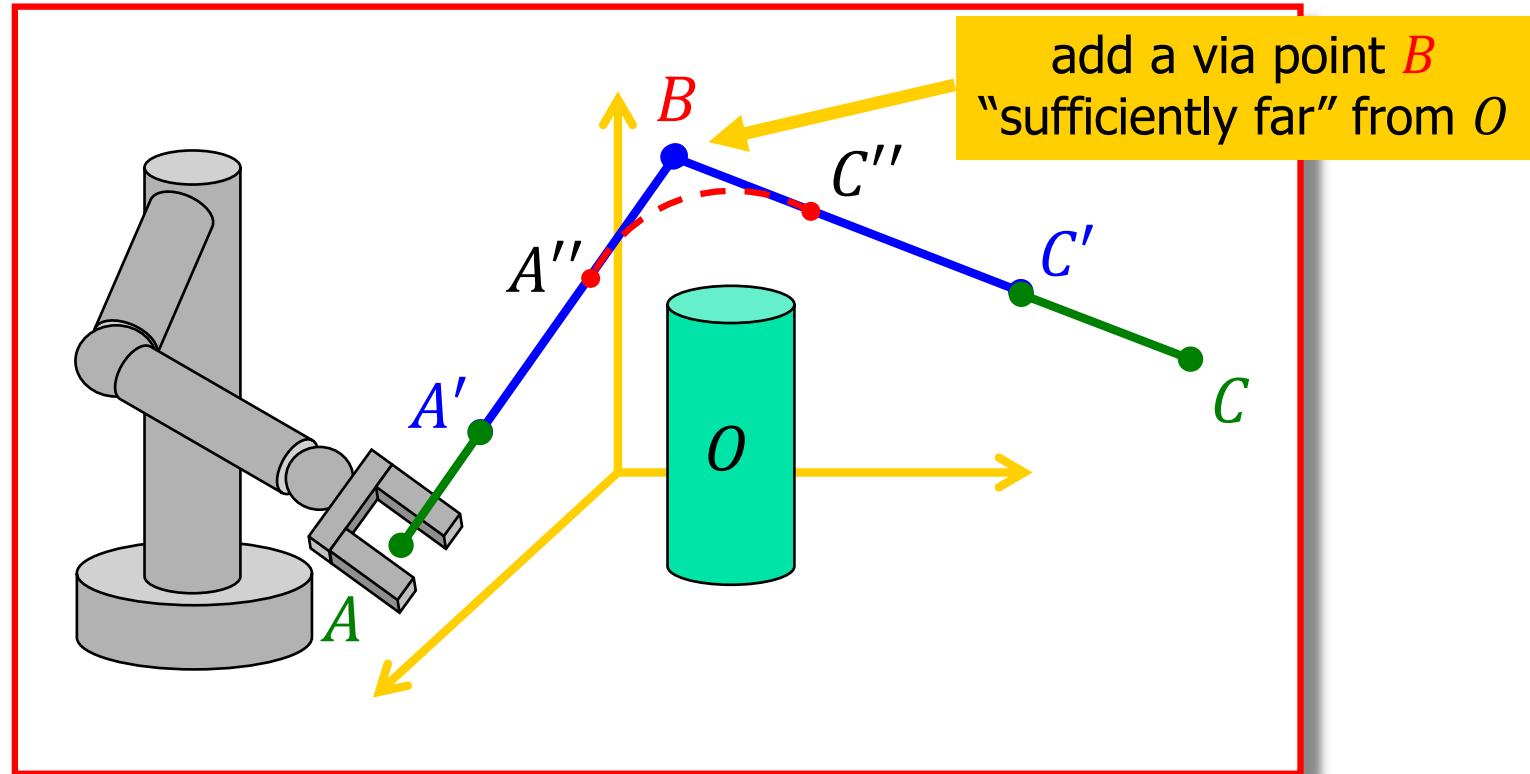
$$\begin{aligned}\Delta T &= (v_{max}/a_{max}) \|K_{BC} - K_{AB}\| \\ &= (v_{max}/a_{max}) \sqrt{2(1 - K_{BC,x}K_{AB,x} - K_{BC,y}K_{AB,y} - K_{BC,z}K_{AB,z})}\end{aligned}$$

$$\text{then, } d_1 = d_2 = v_{max} \Delta T / 2$$



Application example

plan a Cartesian trajectory from A to C (rest-to-rest)
that avoids the obstacle O , with $a \leq a_{max}$ and $v \leq v_{max}$



on $\overline{AA'} \rightarrow a_{max}$; on $\overline{A'B}$ and $\overline{BC'} \rightarrow v_{max}$; on $\overline{C'C} \rightarrow -a_{max}$;
+ over-fly between A'' e C'' (e.g., with a_{max} in norm)



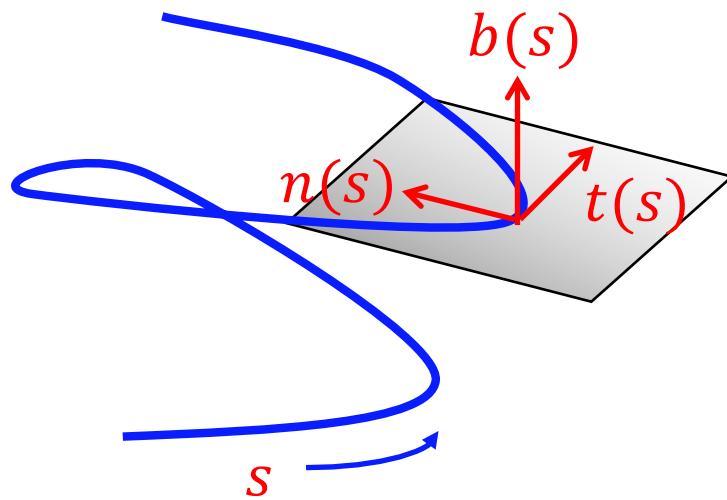
Other Cartesian paths

- **circular path** through 3 points in 3D (often built-in feature)
- linear path for the end-effector with **constant orientation**
- in robots with **spherical wrist**: planning may be **decomposed** into a path for wrist center and one for E-E orientation, with a common timing law
- though more complex in general, it is often **convenient** to parameterize the Cartesian geometric path $p(s)$ in terms of its **arc length** (e.g., with $s = R\theta$ for circular paths), so that the following hold:
 - **velocity** $\dot{p} = dp/dt = (dp/ds)(ds/dt) = p'\dot{s}$
 - p' = unit vector ($\|\cdot\| = 1$) tangent to the path \Rightarrow **tangent** direction $t(s)$
 - $\dot{s} \geq 0$ is the absolute value of the tangential velocity (= **speed**)
 - **acceleration** $\ddot{p} = (d^2p/ds^2)(ds/dt)^2 + (dp/ds)(d^2s/dt^2) = p''\dot{s}^2 + p'\ddot{s}$
 - $\|p''\| =$ **curvature** $\kappa(s)$ (= 1/radius of curvature)
 - $p''\dot{s}^2$ = **centripetal** acceleration \Rightarrow **normal** direction $n(s)$ \perp to the path, on the osculating plane; the **binormal** direction is $b(s) = t(s) \times n(s)$
 - \ddot{s} = scalar value (**with any sign**) of the tangential acceleration



Definition of Frenet frame

- for a generic (smooth) path $p(s) \in \mathbb{R}^3$, parameterized by s (in general, **not** necessarily its arc length), one can define a reference frame as shown



$$p' = dp/ds \quad p'' = d^2p/ds^2$$

derivatives w.r.t. the parameter

$$t(s) = p'(s)/\|p'(s)\| \quad \text{unit tangent vector}$$

$$n(s) = p''(s)/\|p''(s)\| \quad \text{unit normal vector} \\ (\in \text{osculating plane})$$

$$b(s) = t(s) \times n(s) \quad \text{unit binormal vector}$$

- general expression of the path curvature (at a path point $p(s)$)

$$\kappa(s) = \|p'(s) \times p''(s)\|/\|p'(s)\|^3$$

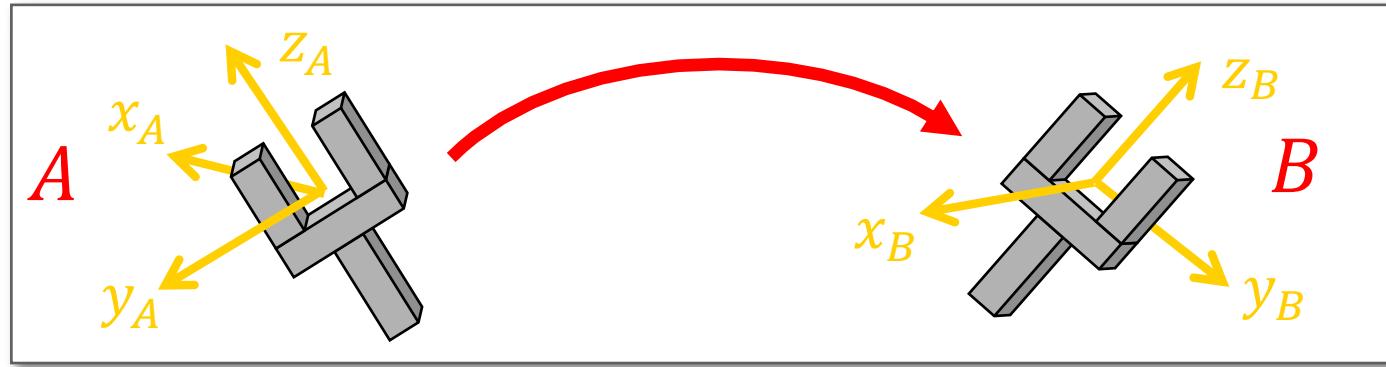


Optimal trajectories

- for Cartesian robots (e.g., PPP joints)
 1. the straight line joining two position points in the Cartesian space is **one** path that can be executed in **minimum time** under velocity/acceleration constraints (but other such paths exist, if (joint) motion is **not coordinated**)
 2. the optimal timing law is of the bang-coast-bang type in **acceleration** (in this special case, also in terms of actuator **torques**)
- for articulated robots (with at least one R joint)
 - 1. e 2. are no longer true in general in the **Cartesian** space, but time-optimality still holds in the **joint** space when assuming **bounds** on **joint velocity/acceleration**
 - straight line paths in the joint space **do not correspond** to straight line paths in the Cartesian space, and vice-versa
 - bounds on joint acceleration are **conservative** (though **kinematically tractable**) w.r.t. actual bounds on actuator torques, which involve the full robot dynamics
 - when changing robot configuration/state, different torque values are needed to impose the same joint accelerations ...



Planning orientation trajectories



- using minimal representations of orientation (e.g., ZXZ Euler angles ϕ, θ, ψ), we can plan a trajectory for each component independently
 - e.g., a linear path in space ϕ, θ, ψ , with a cubic timing law
⇒ but poor prediction/understanding of the resulting intermediate orientations
- alternative method based on the axis/angle representation
 - determine the (neutral) axis r and the angle θ_{AB} : $R(r, \theta_{AB}) = R_A^T R_B$ (rotation matrix changing the orientation from A to B ⇒ inverse axis-angle problem)
 - plan a timing law $\theta(t)$ for the (scalar) angle interpolating $\theta = 0$ with $\theta = \theta_{AB}$ in time T (with possible constraints/boundary conditions on its time derivatives)
 - $\forall t \in [0, T], R_A R(r, \theta(t))$ specifies the actual end-effector orientation at time t



A complete position/orientation Cartesian trajectory

- initial **given** configuration $q(0) = (0 \ \pi/2 \ 0 \ 0 \ 0 \ 0)^T$
- **initial end-effector position** $p(0) = (0.540 \ 0 \ 1.515)^T$
- **initial orientation**

$$R(0) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

linear path
for position

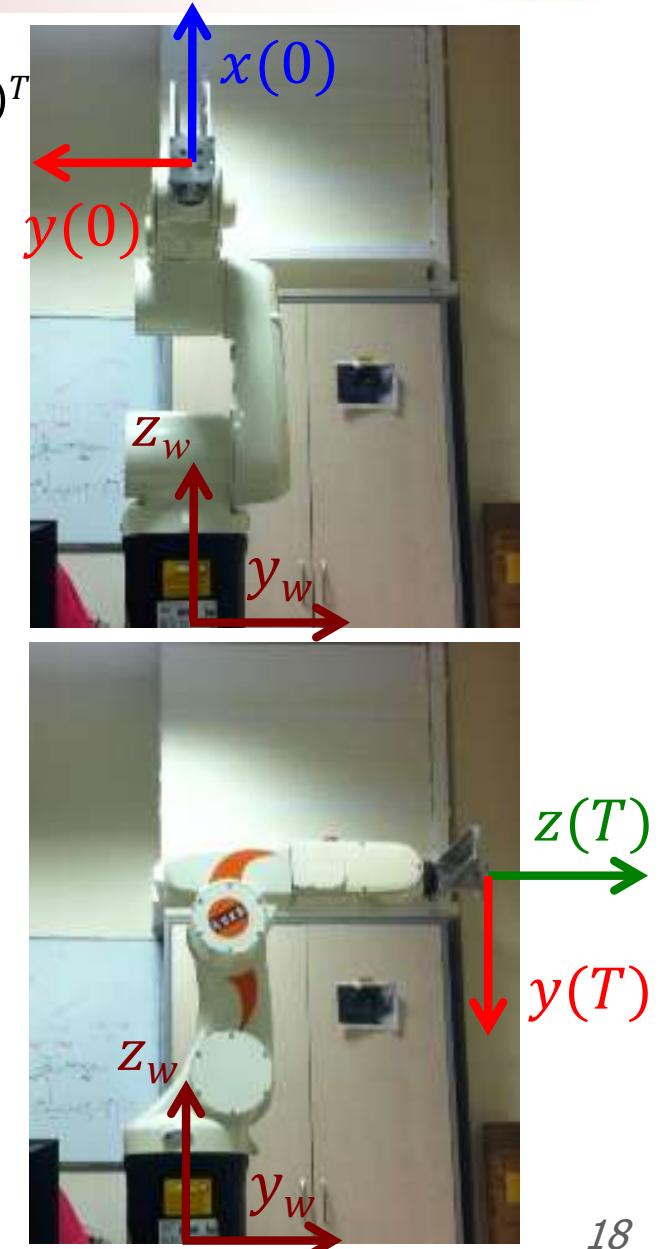


axis-angle method
for orientation

- **final end-effector position** $p(T) = (0 \ 0.540 \ 1.515)^T$
- **final orientation**

$$R(T) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

- the final configuration is **NOT** specified a priori





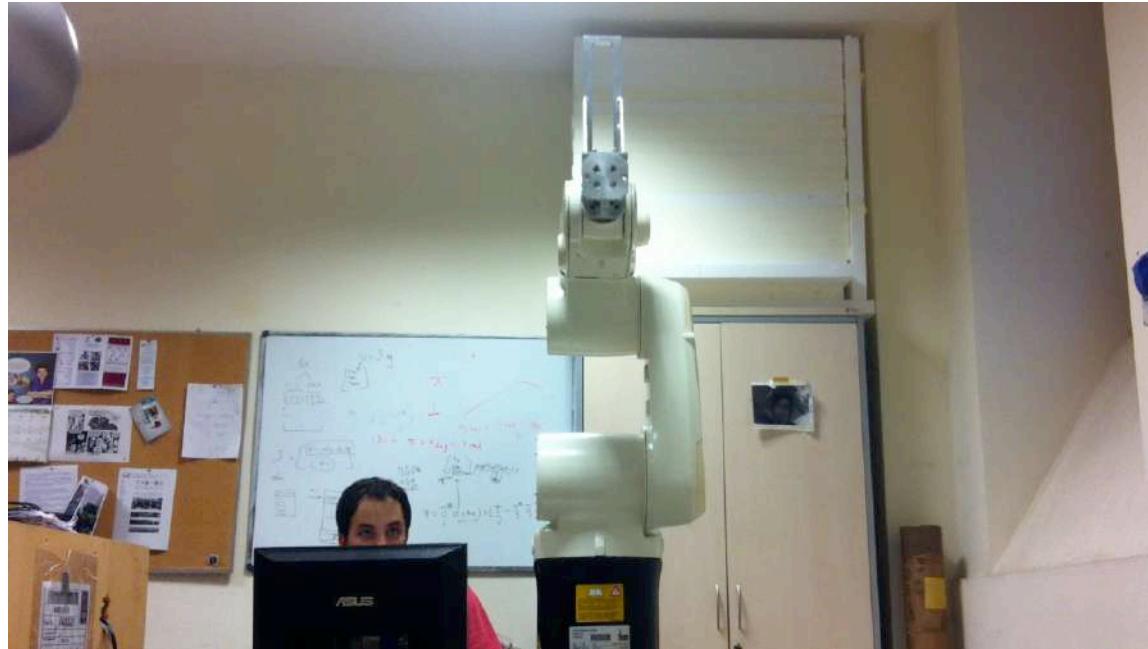
Axis-angle orientation trajectory

video

$$L = \|p_{\text{final}} - p_{\text{init}}\| = 0.763 \text{ [m]}$$

$$\omega = r\dot{\theta} \rightarrow \|\omega\| = |\dot{\theta}|$$

$$\dot{\omega} = r\ddot{\theta} \rightarrow \|\dot{\omega}\| = |\ddot{\theta}|$$



$$p(s) = p_{\text{init}} + s(p_{\text{final}} - p_{\text{init}}) = (0.540 \quad 0 \quad 1.515)^T + s(-0.540 \quad 0.540 \quad 0)^T, \quad s \in [0,1]$$

$$R_{\text{init}} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = R_{\text{init}}^T$$

$$R_{\text{final}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

$$R_{\text{init}}^T R_{\text{final}} = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix} = \text{Rot}(r, \theta_{\text{if}})$$

$$r = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, \theta_{\text{if}} = \frac{2\pi}{3} \text{ [rad]} (= 120^\circ)$$

coordinated
Cartesian motion
with bounds

$$v_{\max} = 0.4 \text{ [m/s]}$$

$$a_{\max} = 0.1 \text{ [m/s}^2]$$

$$\omega_{\max} = \pi/4 \text{ [rad/s]}$$

$$\dot{\omega}_{\max} = \pi/8 \text{ [rad/s}^2]$$



triangular
speed profile $\dot{s}(t)$
with minimum
time $T = 5.52 \text{ s}$

(imposed by the bounds
on linear motion)

$s = s(t), t \in [0, T]$

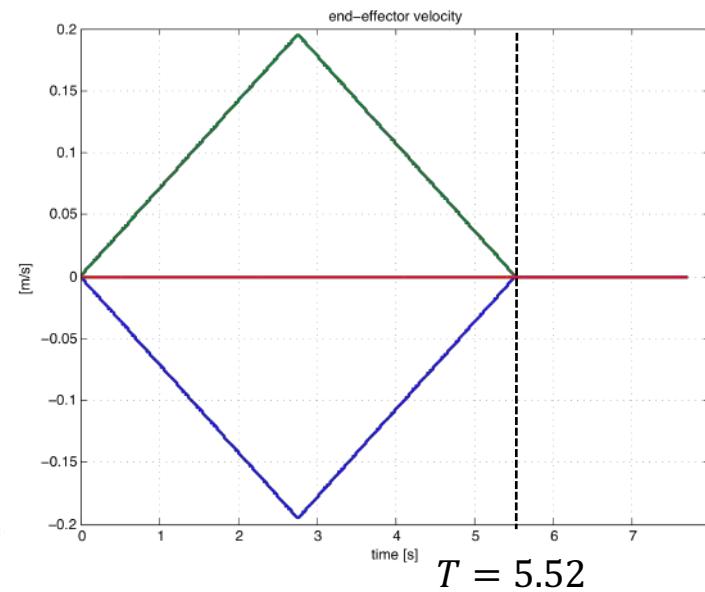
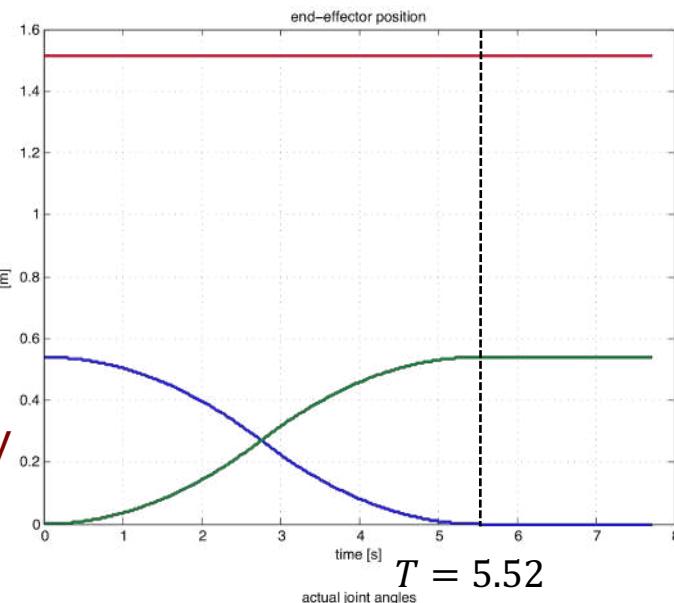
$$R(s) = R_{\text{init}} \text{Rot}(r, \theta(s))$$

$$\theta(s) = s\theta_{\text{if}}, \quad s \in [0,1]$$

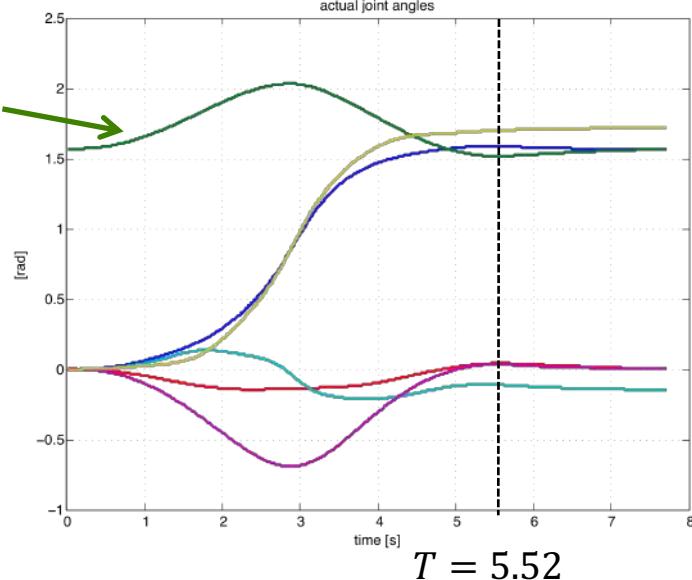


Axis-angle orientation trajectory

planned
motion of
Cartesian
position
and velocity



actual
joint motion



triangular profile for linear speed
 $T = 5.52$ s

- the robot joint velocity was commanded by inversion of the **geometric** Jacobian
- a **user** program, via KUKA RSI interface at $T_c = 12$ ms sampling time (one-way communication)
- robot motion execution is \approx what was planned, but only thanks to an external **kinematic control** loop (at **task** level)



Comparison of orientation trajectories

Euler angles vs. axis-angle method

- initial configuration $q(0) = (0 \quad \pi/2 \quad \pi/2 \quad 0 \quad -\pi/2 \quad 0)^T$
- initial end-effector position $p(0) = (0.115 \quad 0 \quad 1.720)^T$

- initial orientation

$$R(0) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

- initial Euler ZYZ angles $\phi_{ZYX}(0) = (0 \quad \pi/2 \quad \pi)^T$

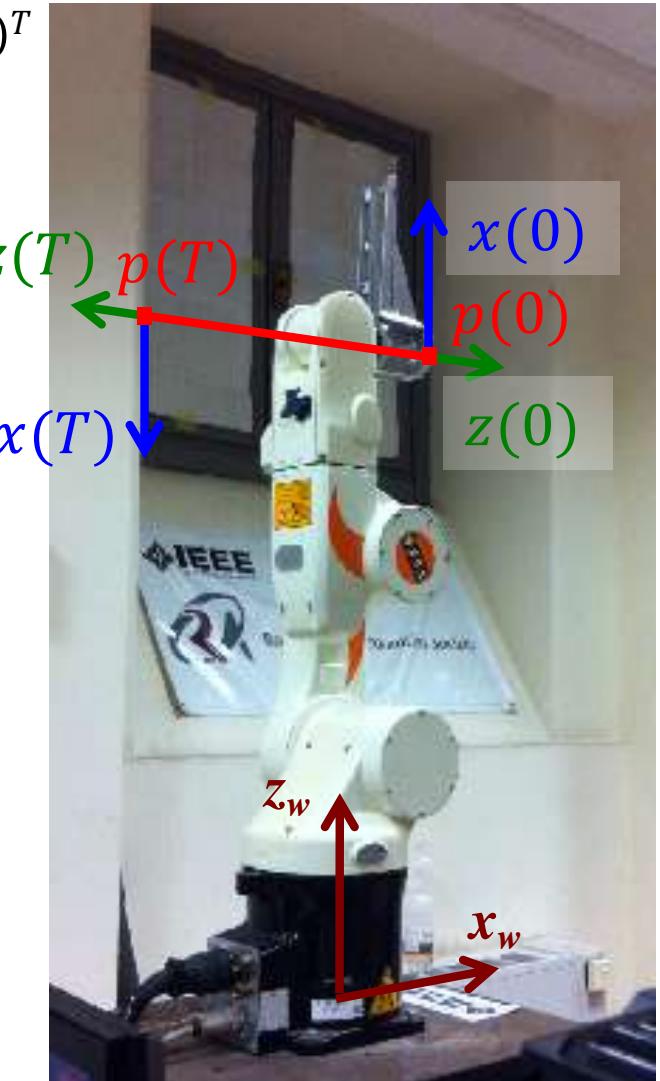
 via a **linear path** (for position)

- final end-effector position $p(T) = (-0.172 \quad 0 \quad 1.720)^T$

- final orientation

$$R(T) = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

- final Euler ZYZ angles $\phi_{ZYX}(T) = (-\pi \quad \pi/2 \quad 0)^T$





Comparison of orientation trajectories

Euler angles vs. axis-angle method

$$R_{\text{init}} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \phi_{ZYX,\text{init}} = \begin{pmatrix} 0 \\ \pi/2 \\ \pi \end{pmatrix}$$

$$R_{\text{final}} = - \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \phi_{ZYX,\text{final}} = \begin{pmatrix} -\pi \\ \pi/2 \\ 0 \end{pmatrix}$$

video



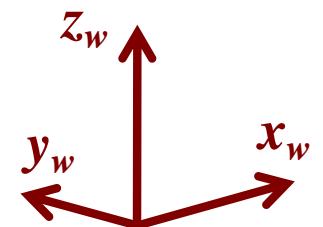
using ZYZ Euler angles



using axis-angle method

$$R_{\text{init}}^T R_{\text{final}} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\Rightarrow r = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}, \quad \theta = \pi$$



video

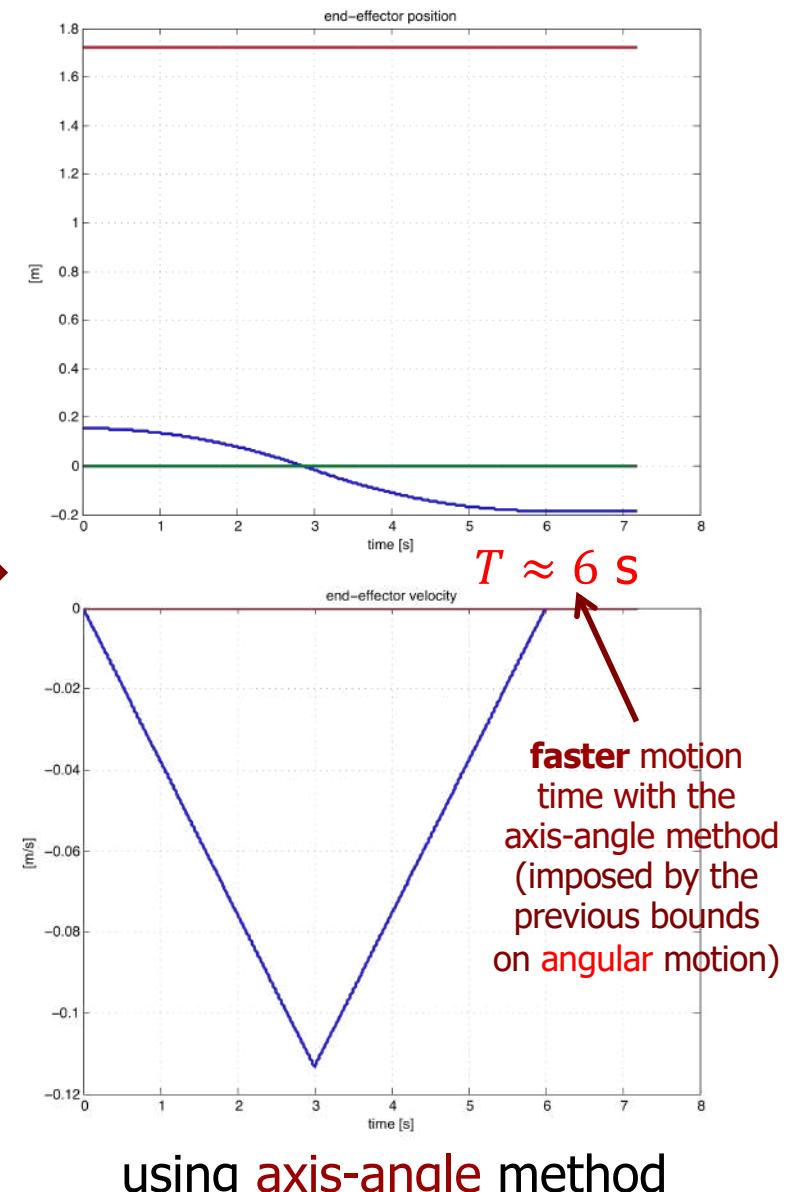
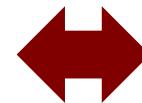
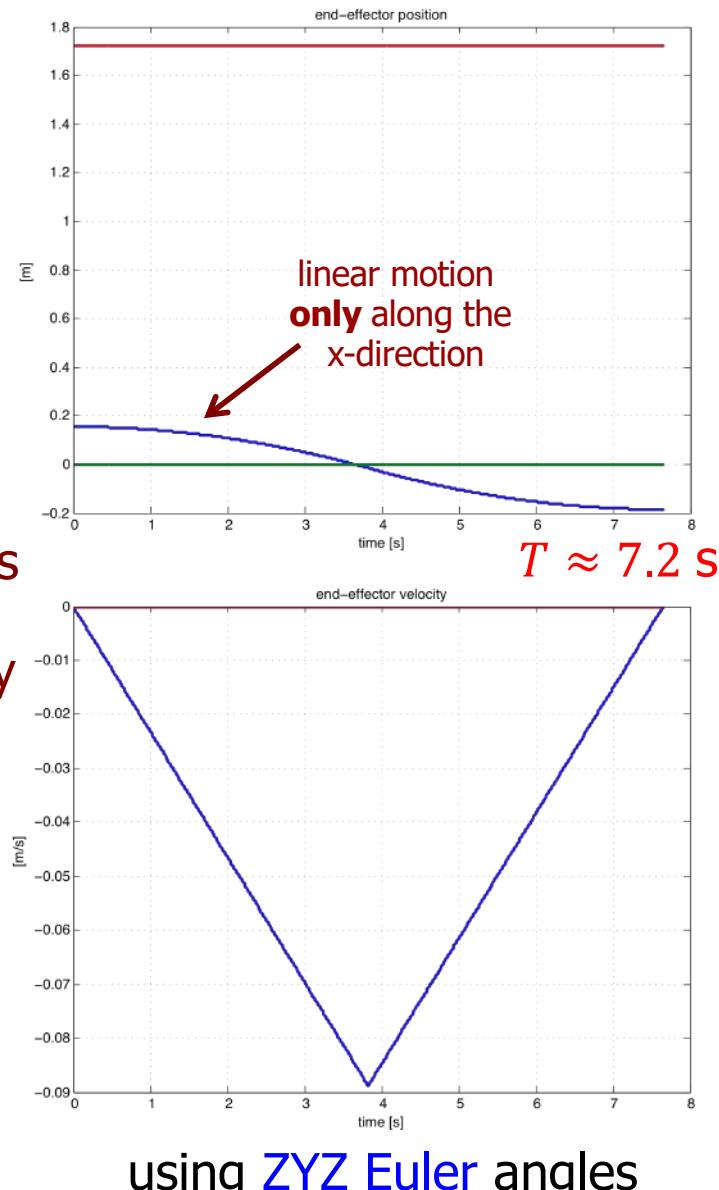


Comparison of orientation trajectories

Euler angles vs. axis-angle method

$x = \text{---}$
 $y = \text{---}$
 $z = \text{---}$

planned
 Cartesian
 components
 of position
 and velocity



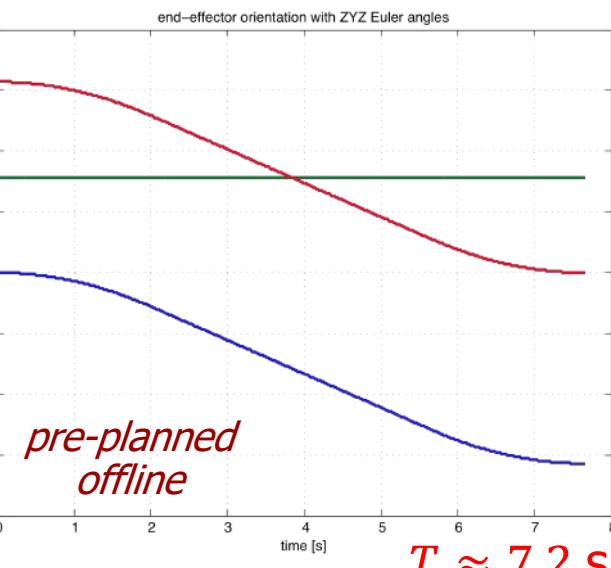


Comparison of orientation trajectories

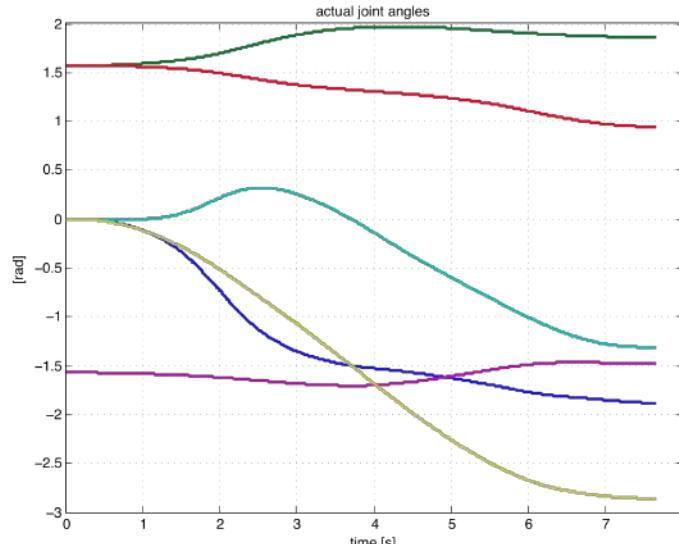
Euler angles vs. axis-angle method

$\alpha = \text{---}$
 $\beta = \text{---}$
 $\gamma = \text{---}$

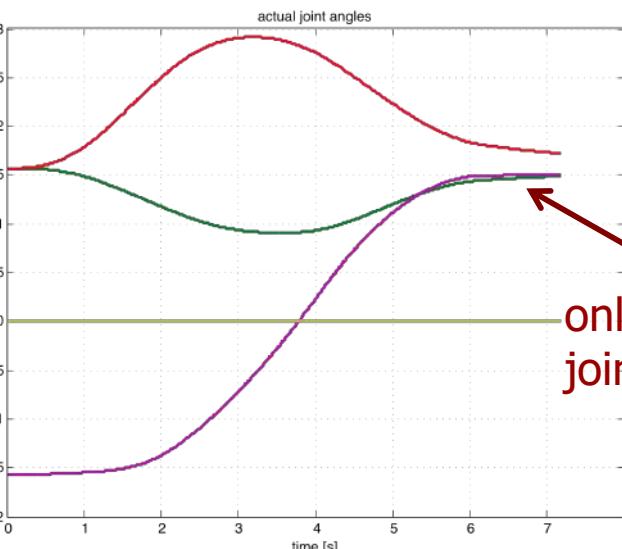
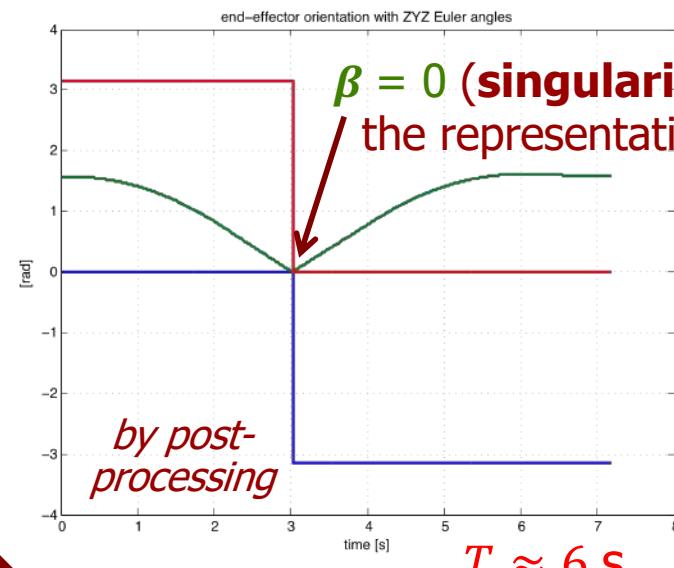
orientation
in terms of ZYZ
Euler angles



actual
joint
motion



using ZYZ Euler angles



using axis-angle method

only three
joints move



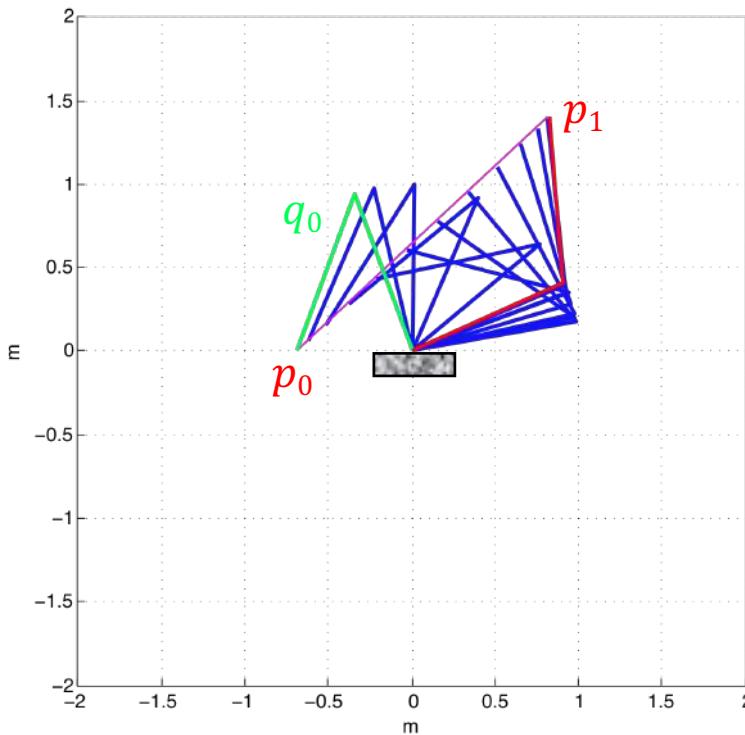
Uniform time scaling

- for a given path $p(s)$ (in joint or Cartesian space) and timing law $s(\tau)$ ($\tau = t/T$, T =“motion time”), we need to **check if existing bounds** v_{max} on (joint) velocity and/or a_{max} on (joint) acceleration **are violated or not**
 - ... unless such constraints have already been taken into account during the trajectory planning, e.g., by using a bang-coast-bang acceleration timing law
- **velocity scales linearly** with motion time
 - $dp/dt = (dp/ds)(ds/d\tau) \cdot 1/T$
- **acceleration scales quadratically** with motion time
 - $d^2p/dt^2 = ((d^2p/ds^2)(ds/d\tau)^2 + (dp/ds)(d^2s/d\tau^2)) \cdot 1/T^2$
- if motion is unfeasible, **scale (increase)** time $T \rightarrow kT$ ($k > 1$), based on the “most violated” constraint (max of the ratios $|v|/v_{max}$ and $|a|/a_{max}$)
- if motion is “too slow” w.r.t. the robot capabilities, **decrease** T ($k < 1$)
 - in both cases, after scaling, there will be (at least) one instant of saturation (for at least one variable)
 - **no need** to re-compute motion profiles from scratch!

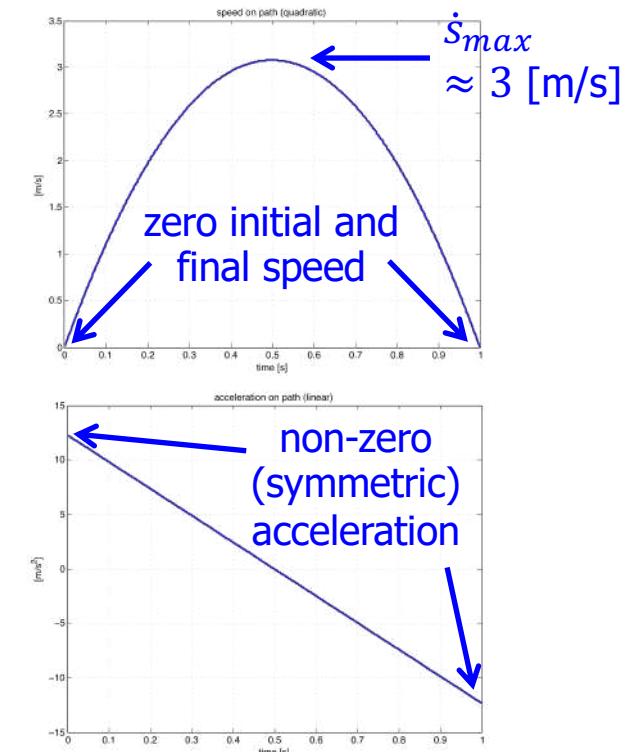
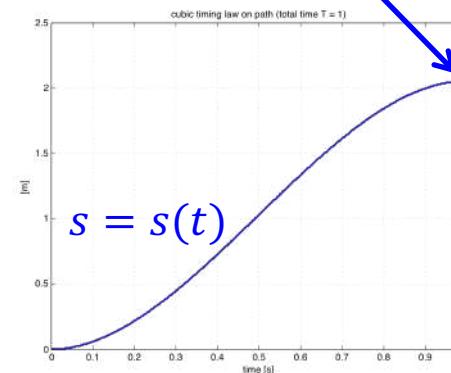


Numerical example - 1

- 2R planar robot with links of unitary length (1 [m])
- linear Cartesian path $p(s)$: $q_0 = (110^\circ, 140^\circ) \Rightarrow p_0 = f(q_0) = (-0.684, 0)$
 $\Rightarrow p_1 = (0.816, 1.4)$ [m], with rest-to-rest cubic timing law $s(t)$, $T = 1$ [s]
- joint space bounds: max (absolute) velocity $v_{max,1} = 2, v_{max,2} = 2.5$ [rad/s],
max (absolute) acceleration $a_{max,1} = 5, a_{max,2} = 7$ [rad/s²]



path length $L = 2.0518$ [m]

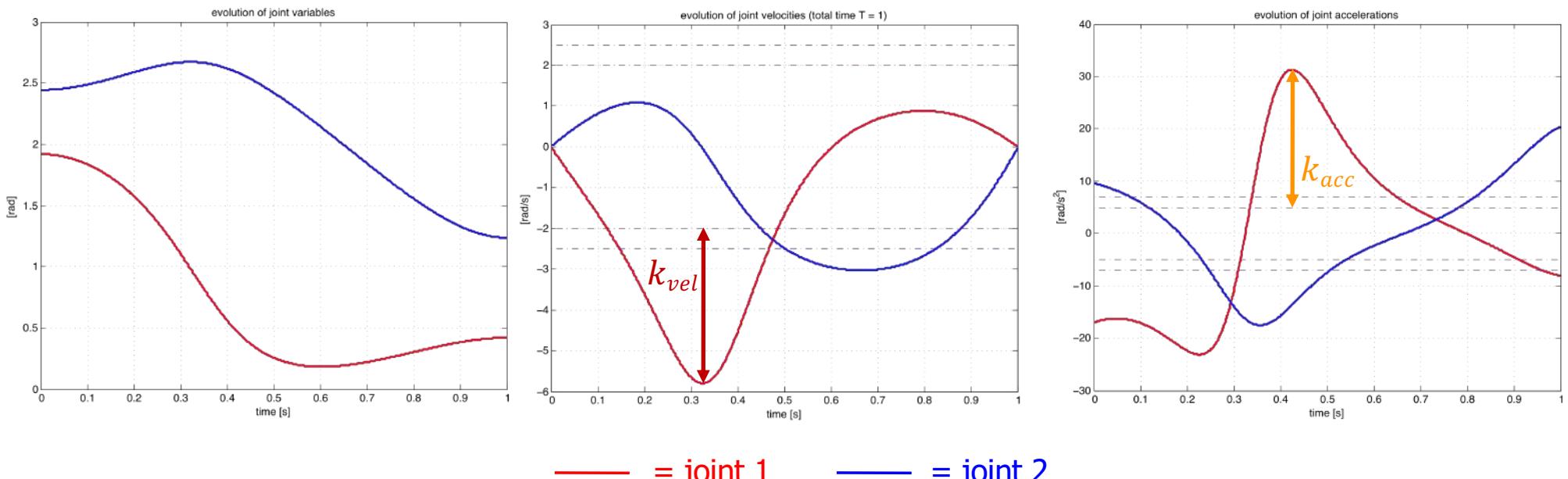




Numerical example - 2

- **violation** of both joint velocity and acceleration bounds with $T = 1$ [s]
 - max relative violation of joint **velocities**: $k_{vel} = 2.898 = \max \{1, |\dot{q}_1|/v_{max,1}, |\dot{q}_2|/v_{max,2}\}$
 - and of joint **accelerations**: $k_{acc} = 6.2567 = \max \{1, |\ddot{q}_1|/a_{max,1}, |\ddot{q}_2|/a_{max,2}\}$
- minimum **uniform time scaling** of Cartesian trajectory to **recover feasibility**

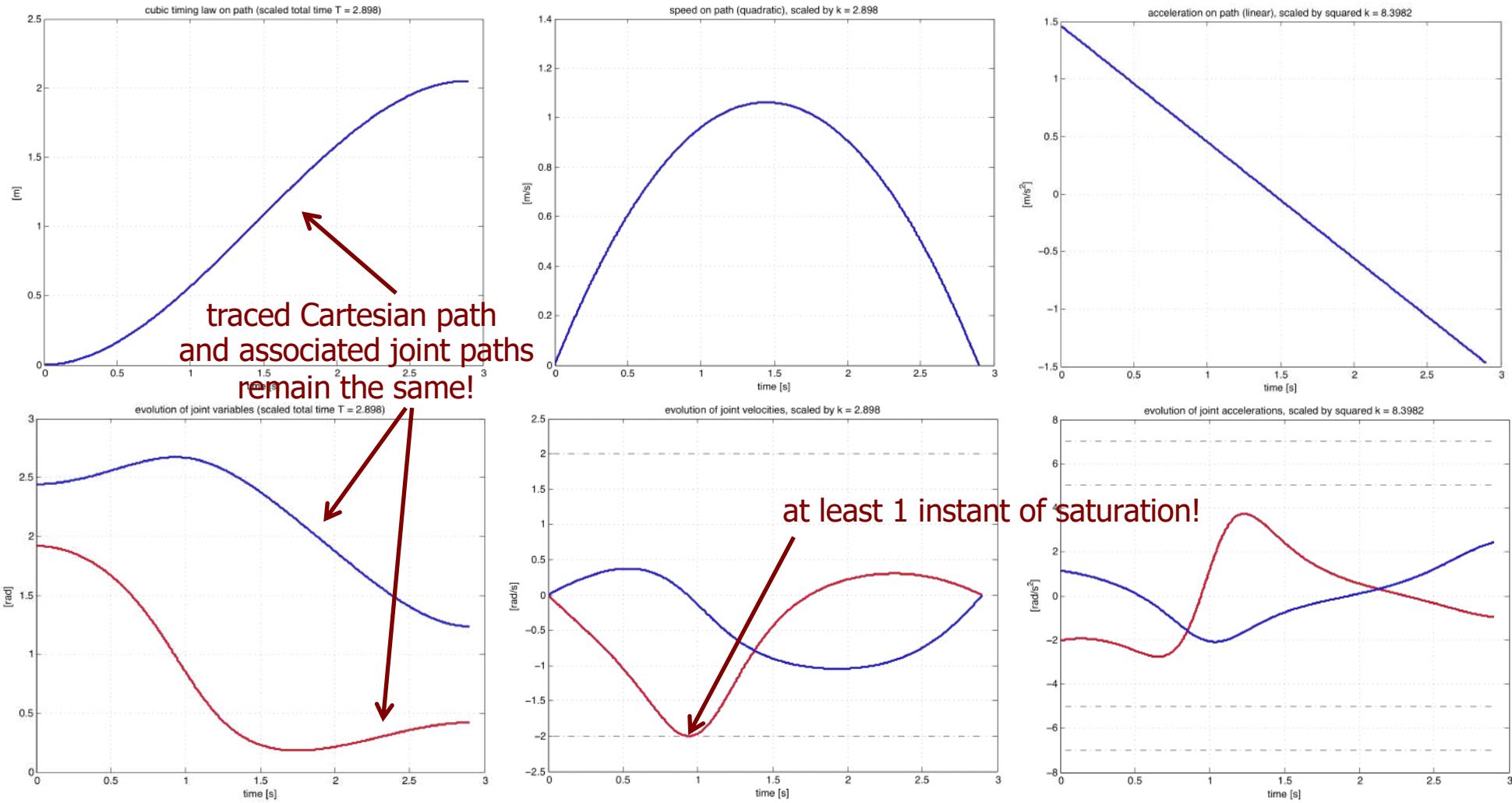
$$k = \max \{1, k_{vel}, \sqrt{k_{acc}}\} = 2.898 \Rightarrow T_{scaled} = kT = 2.898 > T$$





Numerical example - 3

- scaled trajectory with $T_{scaled} = 2.898$ [s]
 - speed [acceleration] on path and joint velocities [accelerations] scale linearly [quadratically]





Robotics 1

Kinematic control

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI





Robot motion control

- need to “actually” realize a desired robot motion task ...
 - regulation of pose/configuration (constant reference)
 - trajectory following/tracking (time-varying reference)
- ... despite the presence of
 - external disturbances and/or unmodeled dynamic effects
 - initial errors (or arising later due to disturbances) w.r.t. desired task
 - discrete-time implementation, uncertain robot parameters, ...
- we use a general control scheme based on
 - feedback (from robot state measures, to impose asymptotic stability)
 - feedforward (nominal commands generated in the planning phase)
- the error driving the feedback part of the control law can be defined either in Cartesian or in joint space
 - control action always occurs at the joint level (where actuators drive the robot), but performance has to be evaluated at the task level



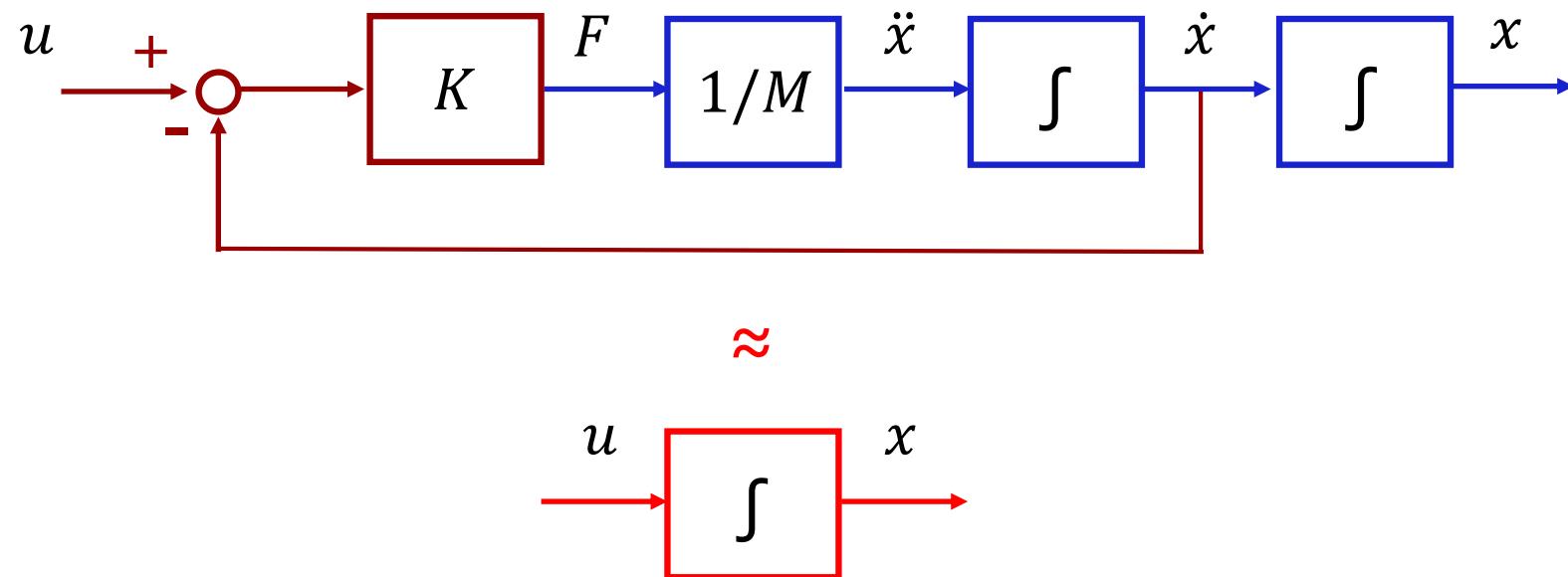
Kinematic control of robots

- a robot is an electro-mechanical system driven by actuating **torques** produced by the motors
- it is possible, however, to consider a **kinematic command** (most often, a **velocity**) as control input to the system...
- ...thanks to the presence of **low-level feedback control** at the robot joints that allows imposing commanded reference velocities (at least, in the “ideal case”)
- these feedback loops are present in industrial robots within a **“closed” control architecture**, where users can only specify reference commands of the kinematic type
- in this way, **performance** can be very satisfactory, provided the desired motion is **not too fast** and/or **does not require large accelerations**



An introductory example

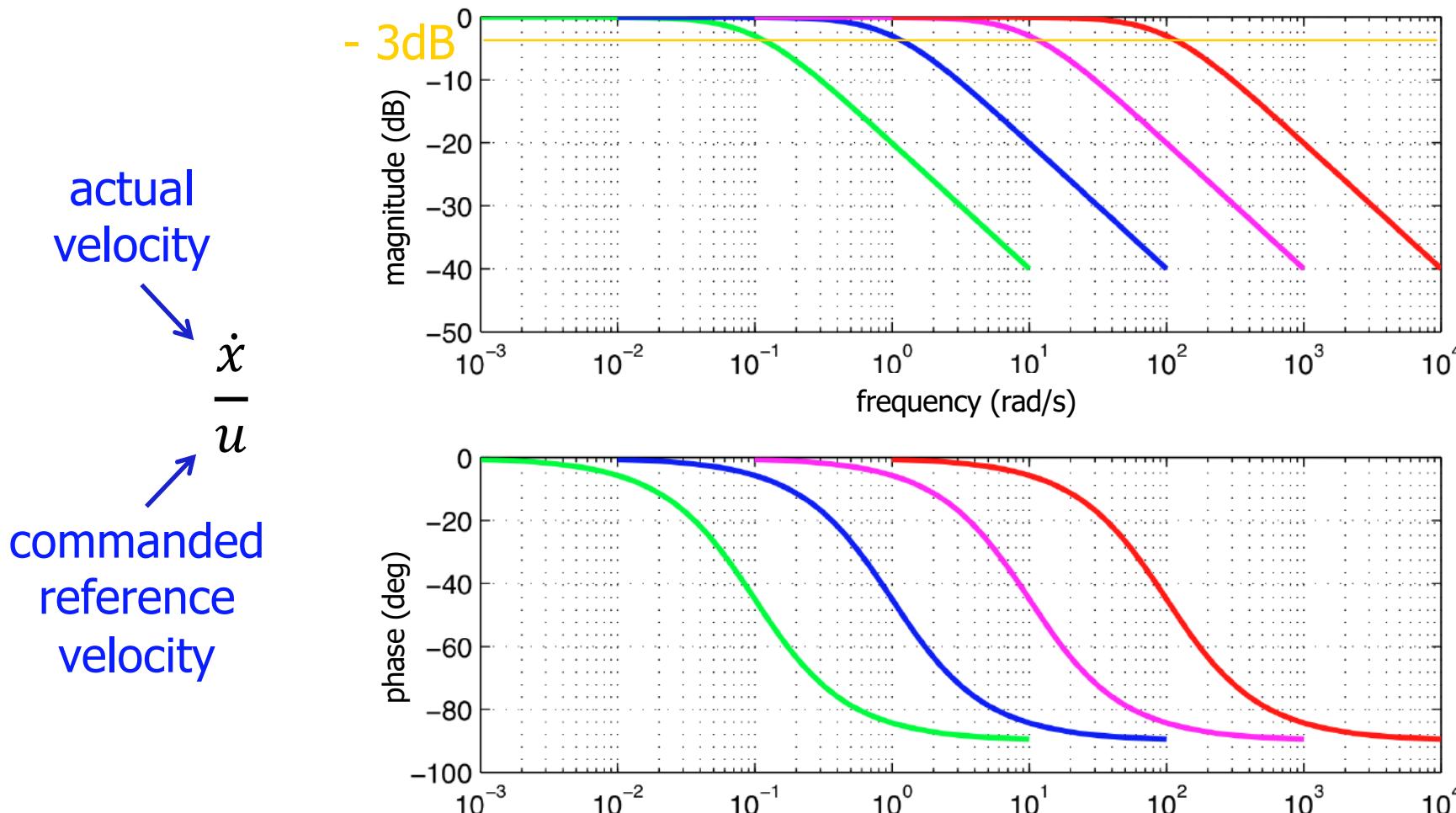
- a mass M in linear motion: $M\ddot{x} = F$
- low-level feedback: $F = K(u - \dot{x})$, with u = reference velocity
- equivalent scheme for $K \rightarrow \infty$: $\dot{x} \approx u$
- in practice, valid in a limited frequency “bandwidth” $\omega \leq K/M$





Frequency response of the closed-loop system

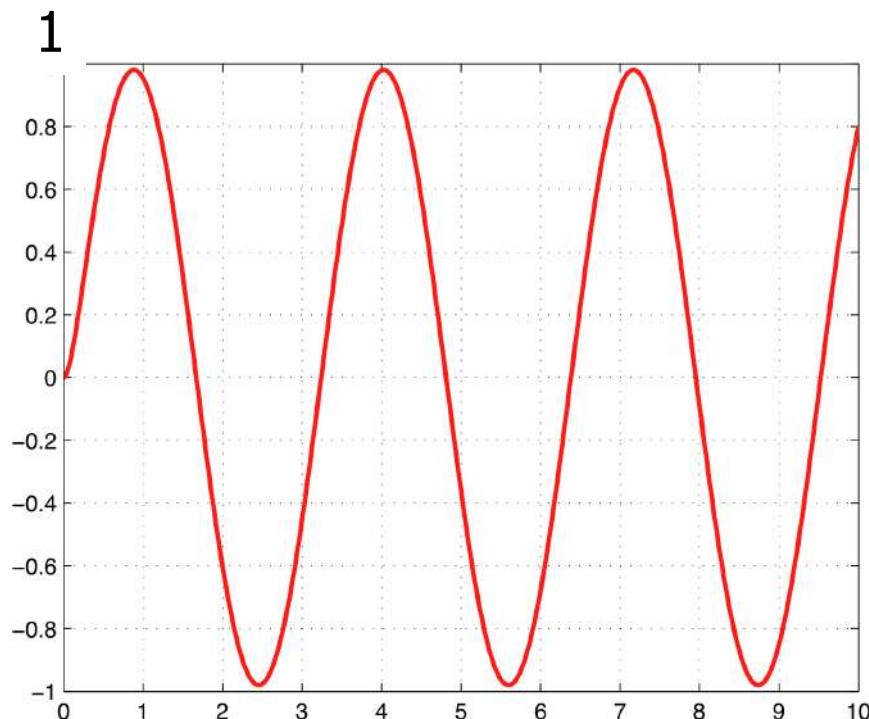
- Bode diagrams of $P(s) = \frac{v(s)}{u(s)} = \frac{sx(s)}{u(s)}$ for $K/M = 0.1, 1, 10, 100$



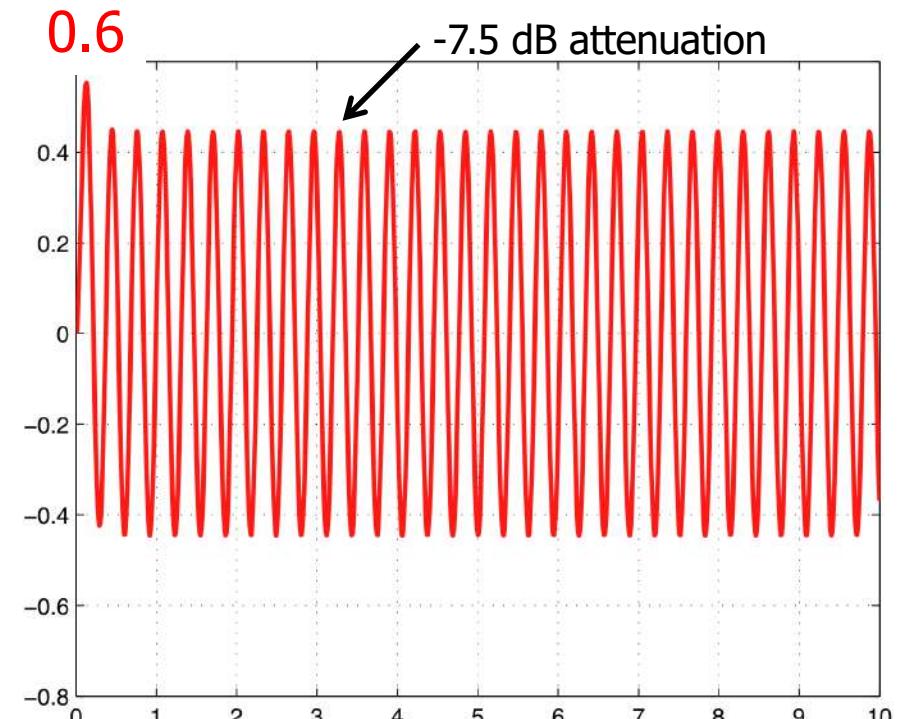


Time response

- setting $K/M = 10$ (bandwidth), we show two possible time responses to unit sinusoidal velocity reference commands at different ω



$\omega = 2 \text{ rad/s}$



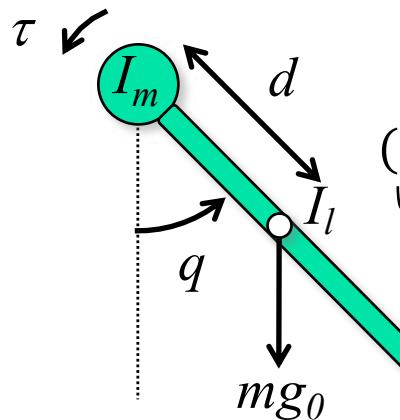
actually realized velocities



A more detailed example

including nonlinear dynamics

- single link (a thin rod) of mass m , center of mass at d from joint axis, inertia M (motor + link) at the joint, rotating in a vertical plane (the gravity torque at the joint is configuration dependent)



dynamic model

$$(I_m + I_l + md^2)\ddot{q} + mg_0d \sin q = \tau$$

$I_m + I_l + md^2$

M

$$\begin{aligned} g_0 &= 9.81 [m/s^2] \\ m &= 10 [kg] \\ d &= l/2 = 0.2 [m] \\ I_l &= ml^2/12 = 0.1333 [kgm^2] \\ I_m &= 0.5333 [kgm^2] \\ &\quad (= I_l + md^2) \\ \Rightarrow M &= 1.0667 [kgm^2] \end{aligned}$$

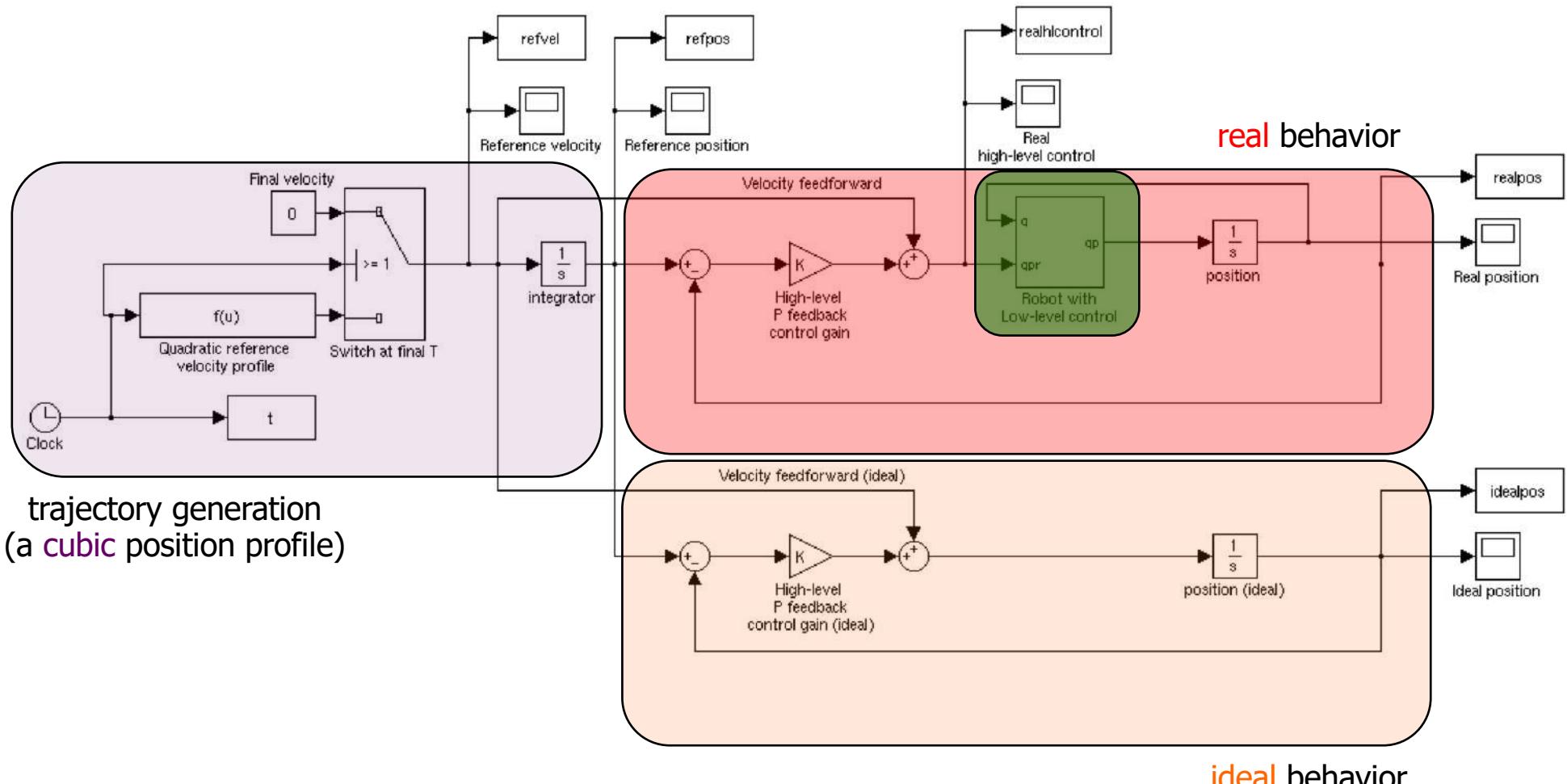
- fast **low-level feedback** control loop based on a PI action on the velocity error + an approximate acceleration feedforward
- kinematic control** loop based on a P feedback action on the position error + feedforward of the velocity reference at the joint level
- evaluation of tracking **performance** for rest-to-rest motion tasks with “increasing dynamics” = higher accelerations



A more detailed example

differences between the ideal and real case

- Simulink scheme

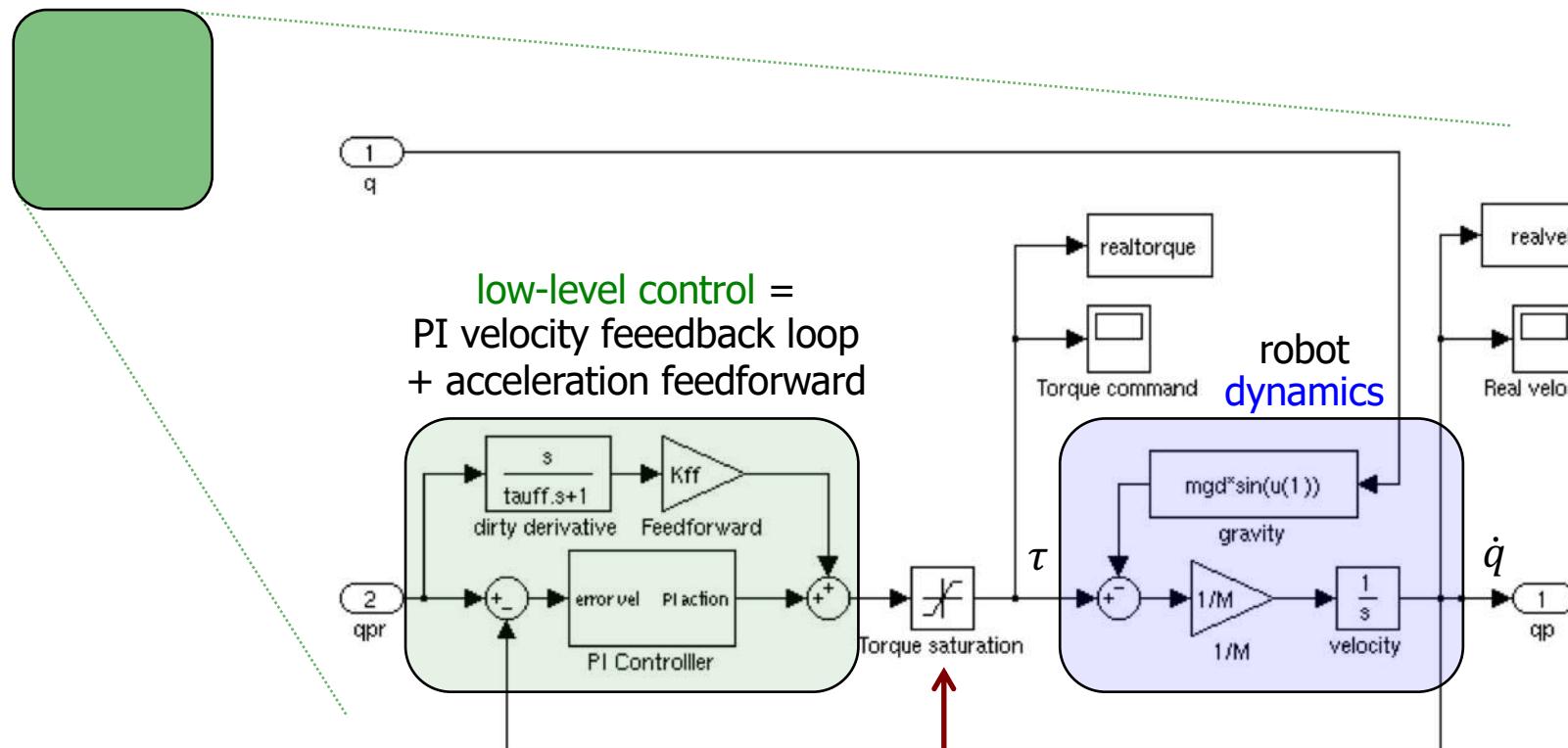




A more detailed example

robot with low-level control

- Simulink scheme



$$M\ddot{q} + mg_o d \sin q = \tau$$

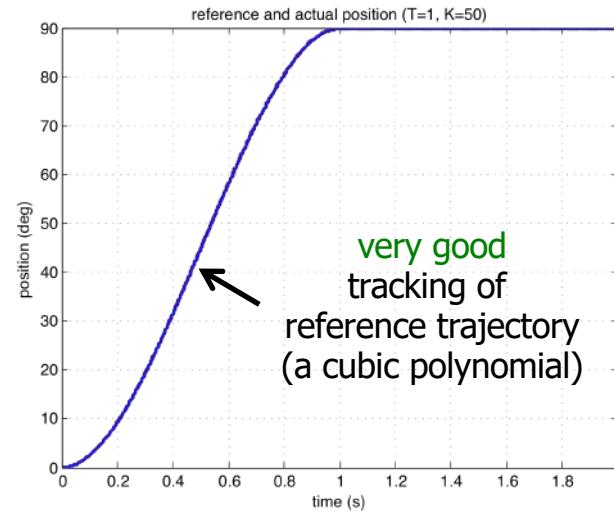
$$\ddot{q} = \frac{1}{M}(\tau - mg_o d \sin q)$$



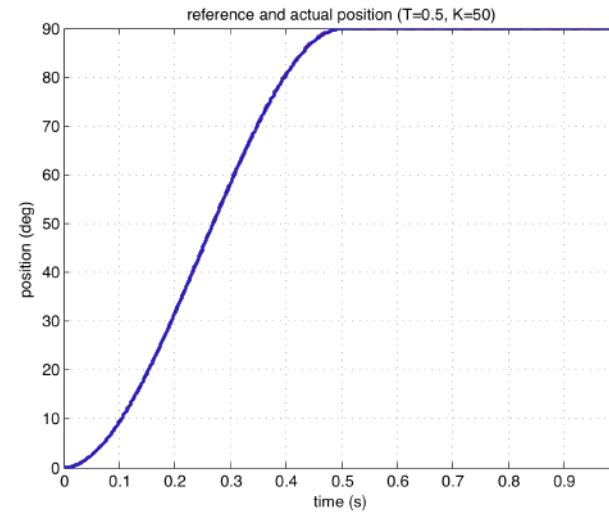
Simulation results

rest-to-rest motion from downward to horizontal position

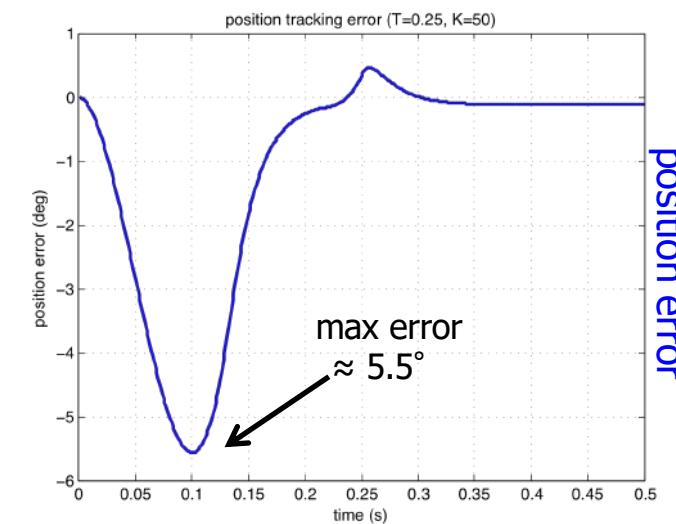
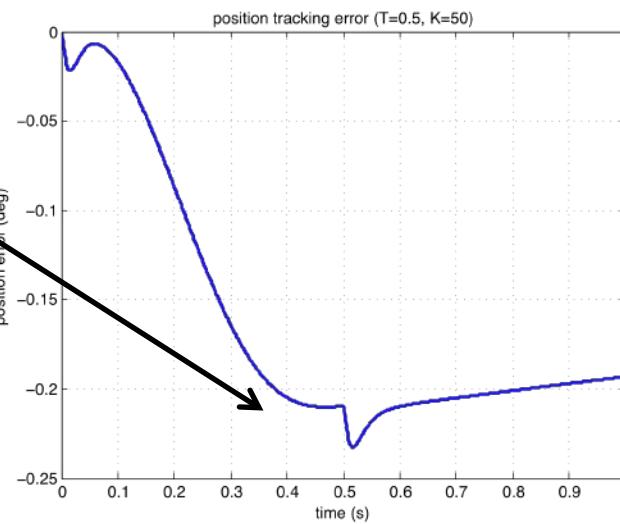
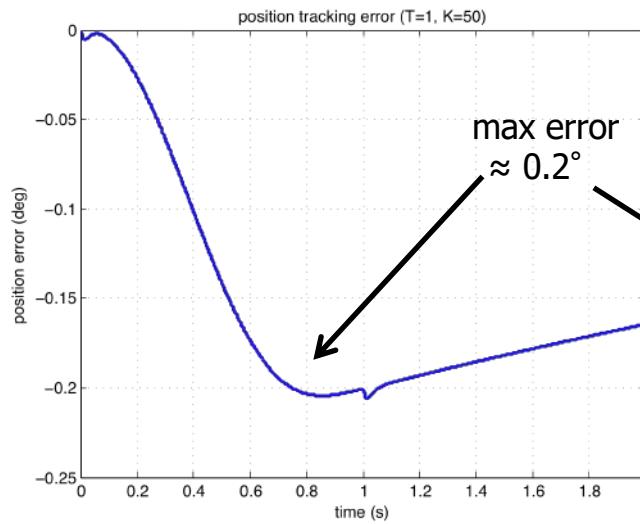
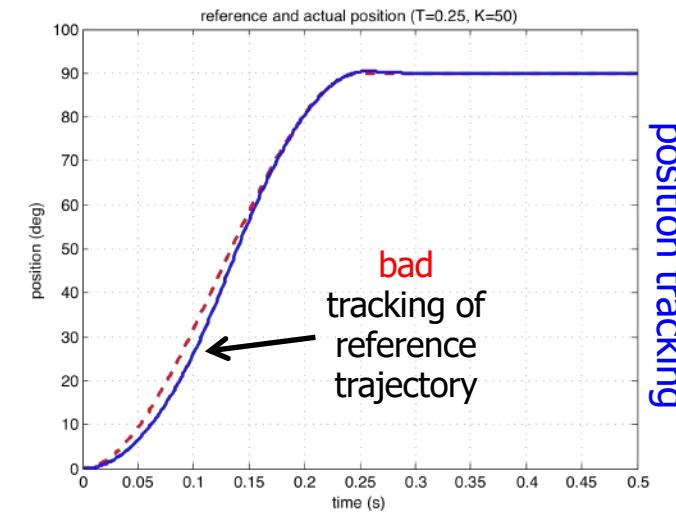
- in $T = 1 \text{ s}$



- in $T = 0.5 \text{ s}$



- in $T = 0,25 \text{ s}$



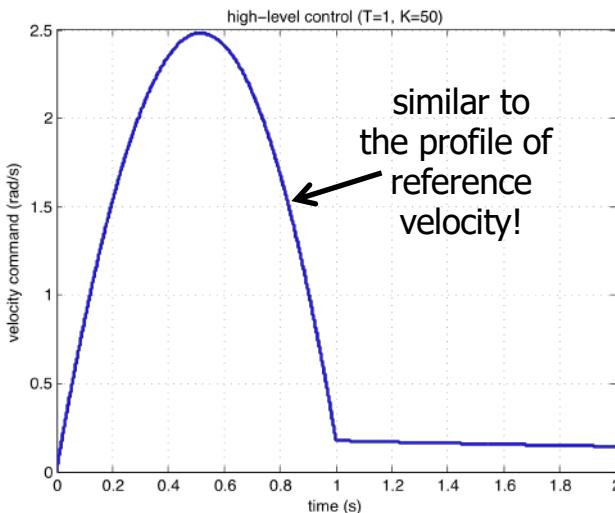
position tracking position error



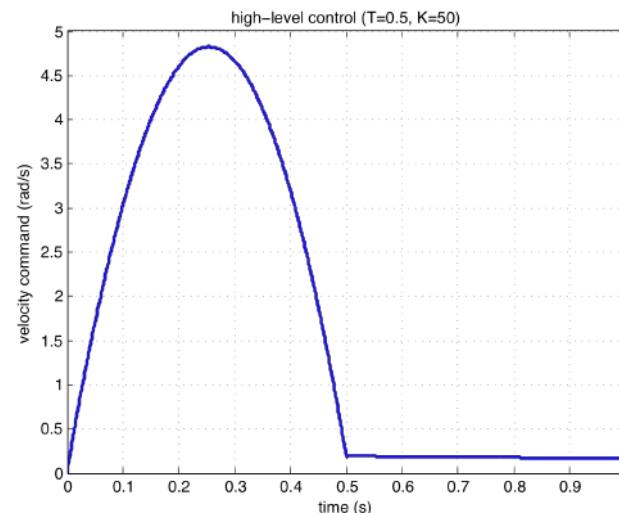
Simulation results

rest-to-rest motion from downward to horizontal position

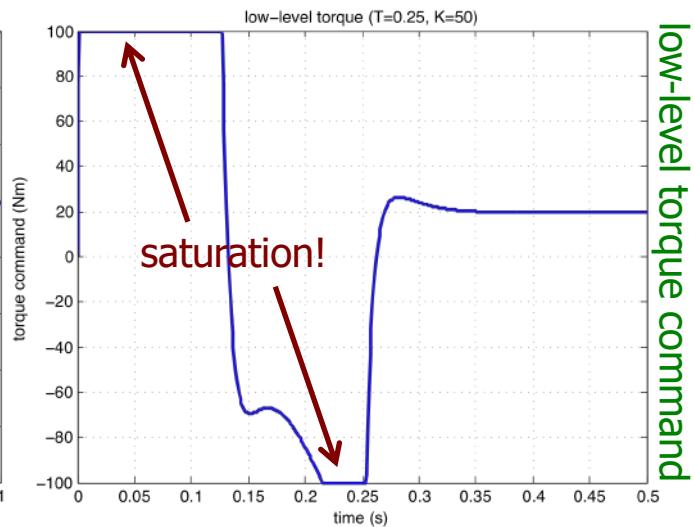
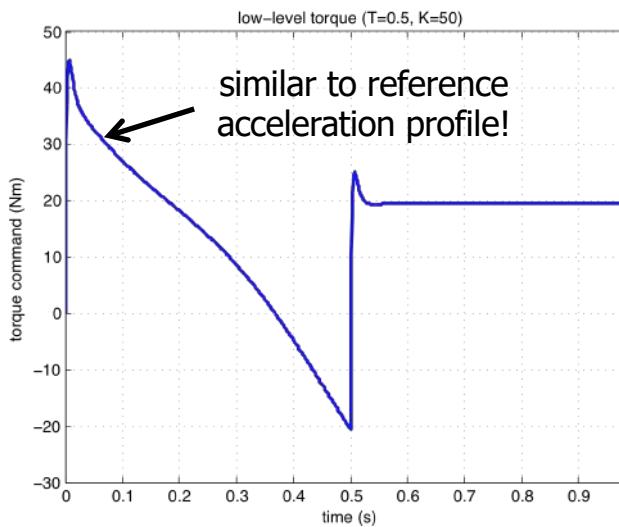
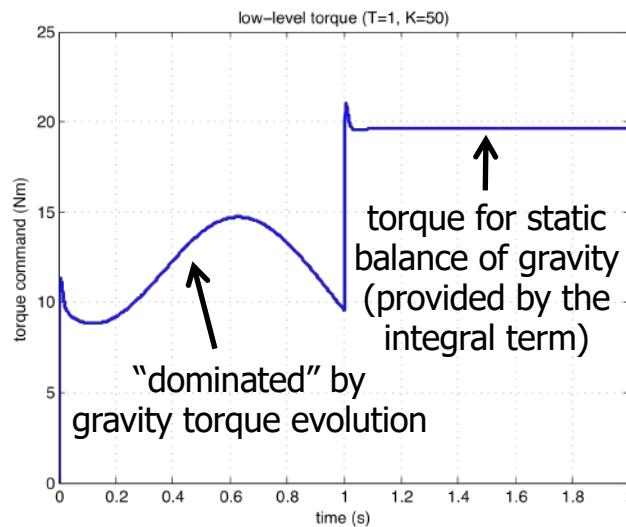
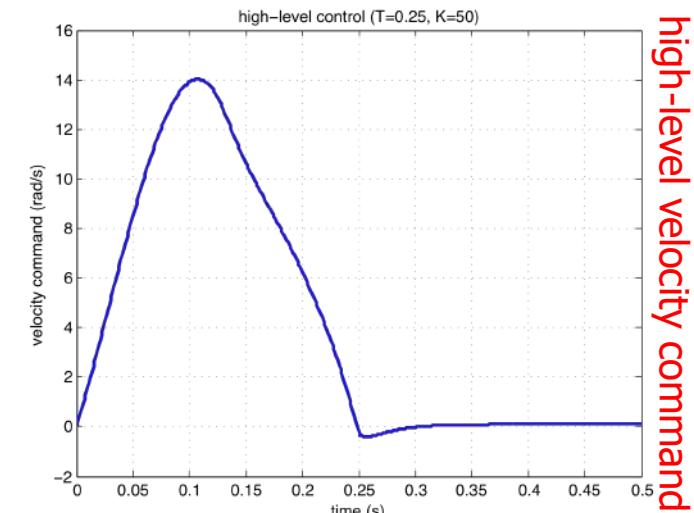
- in $T = 1 \text{ s}$



- in $T = 0.5 \text{ s}$



- in $T = 0.25 \text{ s}$

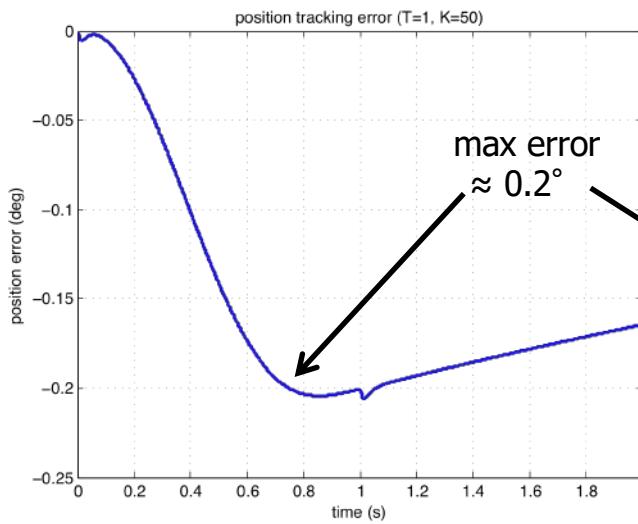




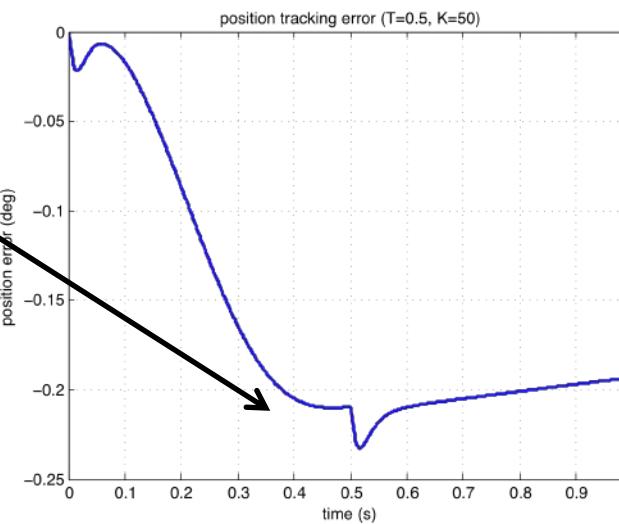
Simulation results

rest-to-rest motion from downward to horizontal position

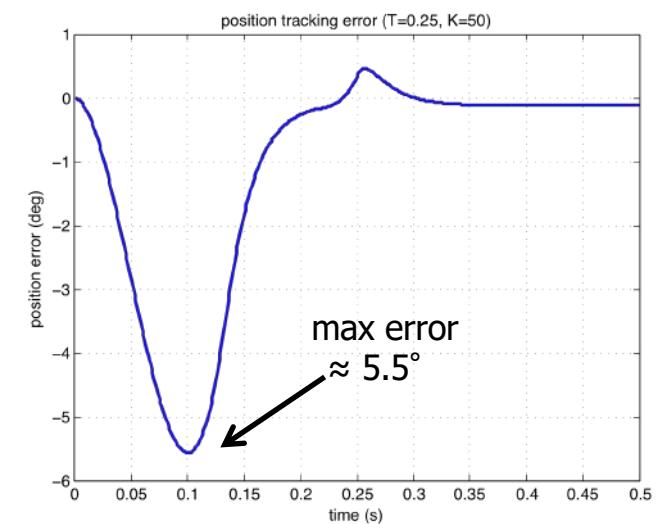
- in $T = 1$ s



- in $T = 0.5$ s



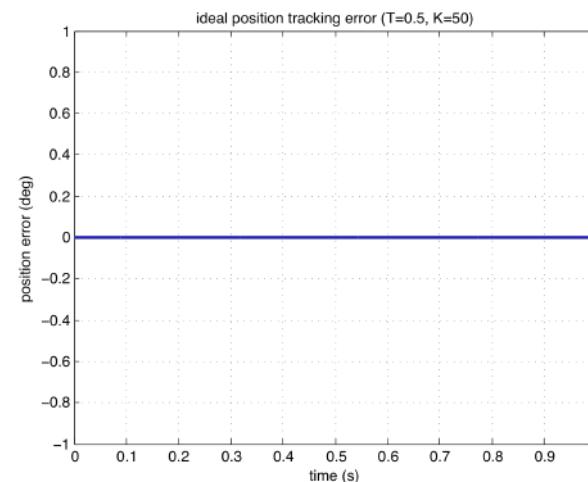
- in $T = 0.25$ s



real position errors increase when reducing too much motion time
(\Rightarrow too high accelerations)

while **ideal** position errors
(based only on kinematics)
remain always the same!!

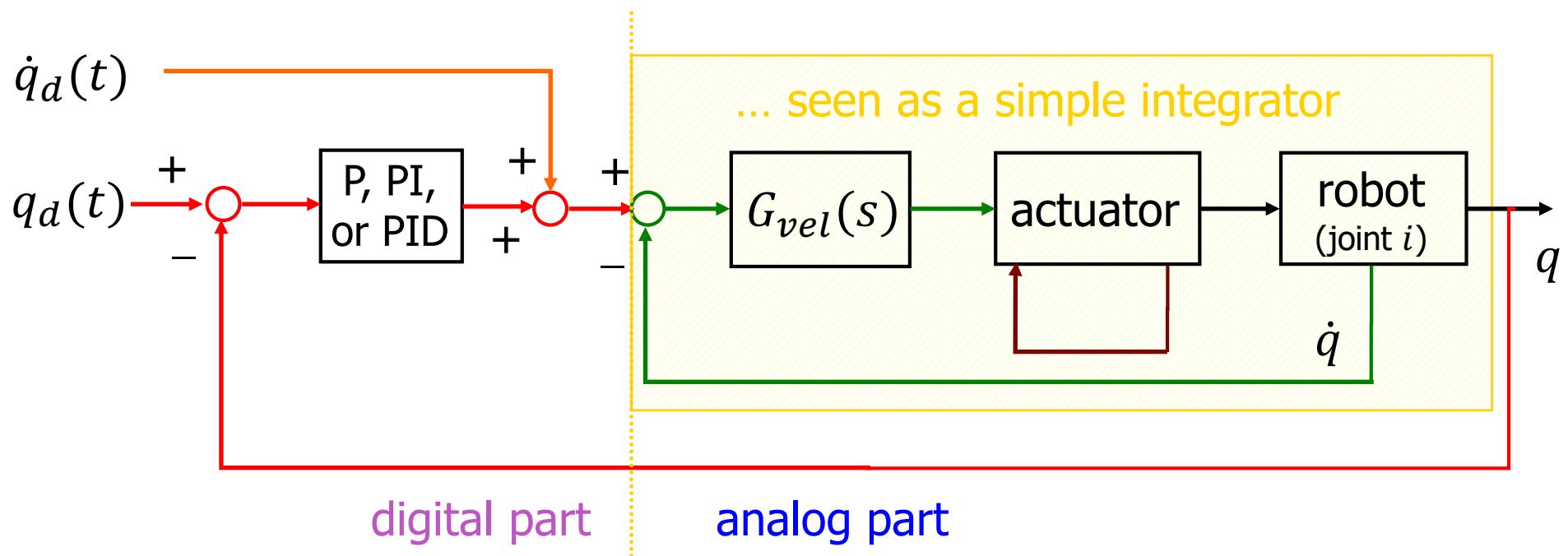
*here $\equiv 0$, thanks to the initial matching
between robot and reference trajectory*





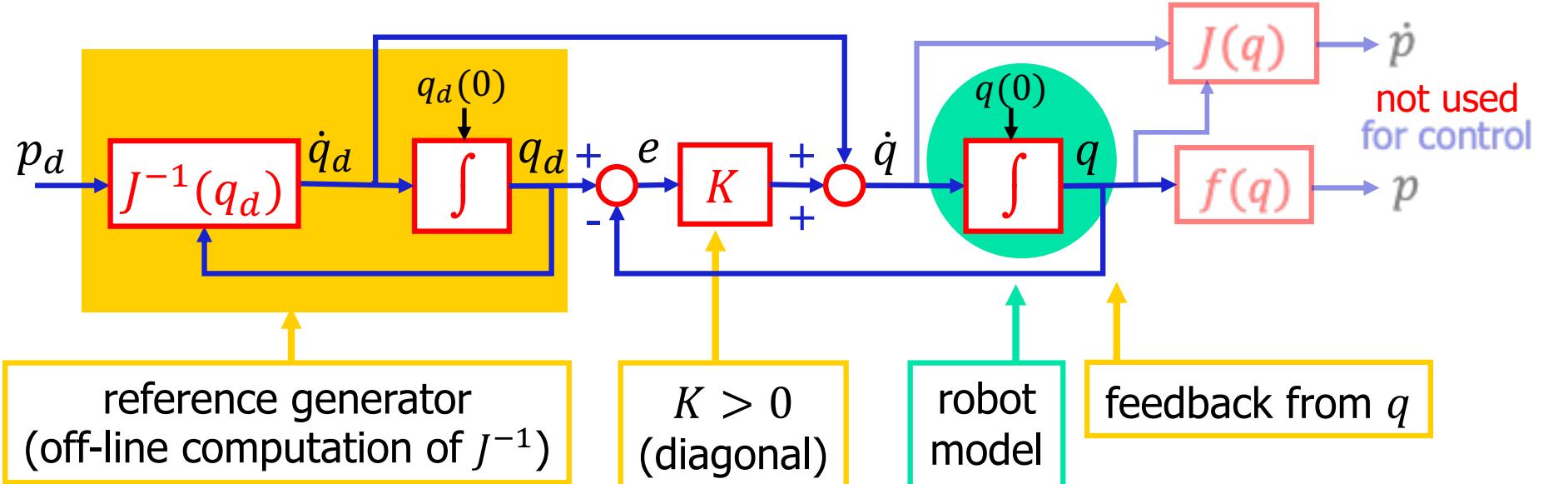
Control loops in industrial robots

- analog loop of large bandwidth on motor current (\propto torque)
- analog loop on velocity ($G_{vel}(s)$, typically a PI)
- digital feedback loop on position, with velocity feedforward
- this scheme is local to each joint (decentralized control)





Kinematic control of joint motion



reference generator
(off-line computation of J^{-1})

$K > 0$
(diagonal)

robot
model

feedback from q

decoupled $e_i \rightarrow 0$
 $(i = 1, \dots, n)$
exponentially,
 $\forall e(0)$

$$e = q_d - q \rightarrow \dot{e} = \dot{q}_d - \dot{q} = \dot{q}_d - (\dot{q}_d + K(q_d - q)) = -Ke$$

$$e_p = p_d - p \rightarrow \dot{e}_p = \dot{p}_d - \dot{p} = J(q_d)\dot{q}_d - J(q)(\dot{q}_d + K(q_d - q))$$

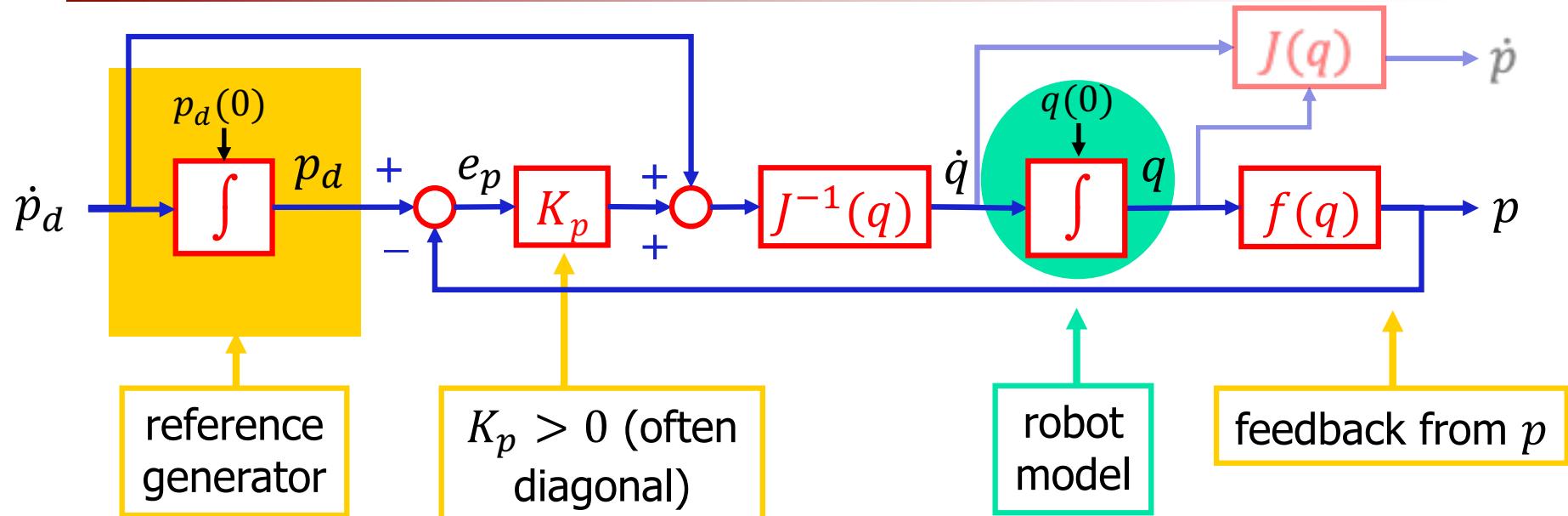
$$\begin{aligned} q &\triangleq q_d \\ e_p &\rightarrow J(q)e \end{aligned}$$

$$\dot{e}_p \approx -J(q)K J^{-1}(q)e_p$$

coupled Cartesian
error dynamics



Kinematic control of Cartesian motion



$$e_p = p_d - p \rightarrow \dot{e}_p = \dot{p}_d - \dot{p} = \dot{p}_d - J(q)J^{-1}(q)(\dot{p}_d + K_p(p_d - p)) = -K_p e_p$$

- decoupled $e_{p,i} \rightarrow 0$ ($i = 1, \dots, m$) exponentially, $\forall e_p(0)$
- needs on-line computation of the inverse^(*) $J^{-1}(q)$
- real-time + singularities issues

(*) or pseudoinverse if $m < n$

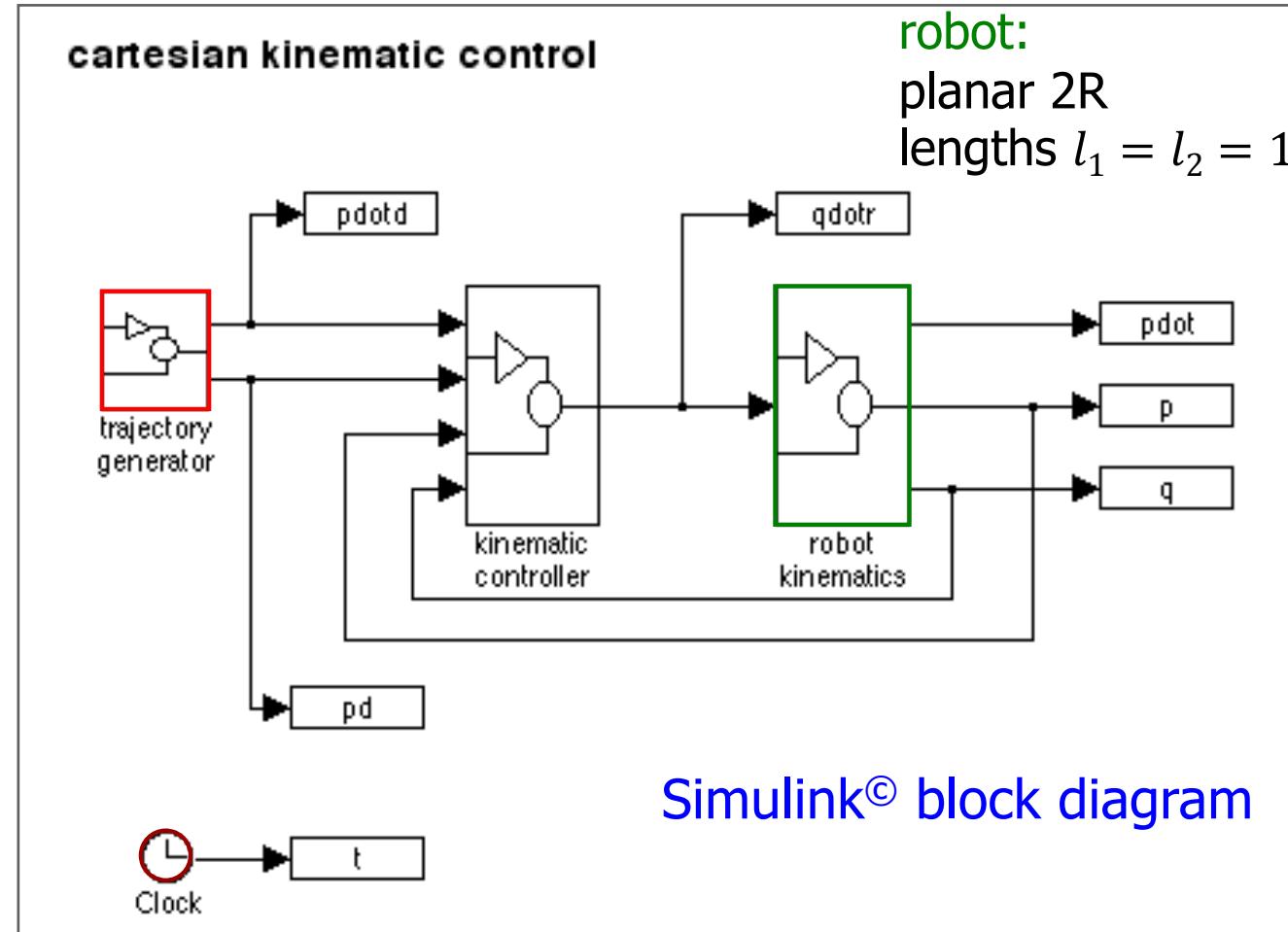
Simulation

features of kinematic control laws



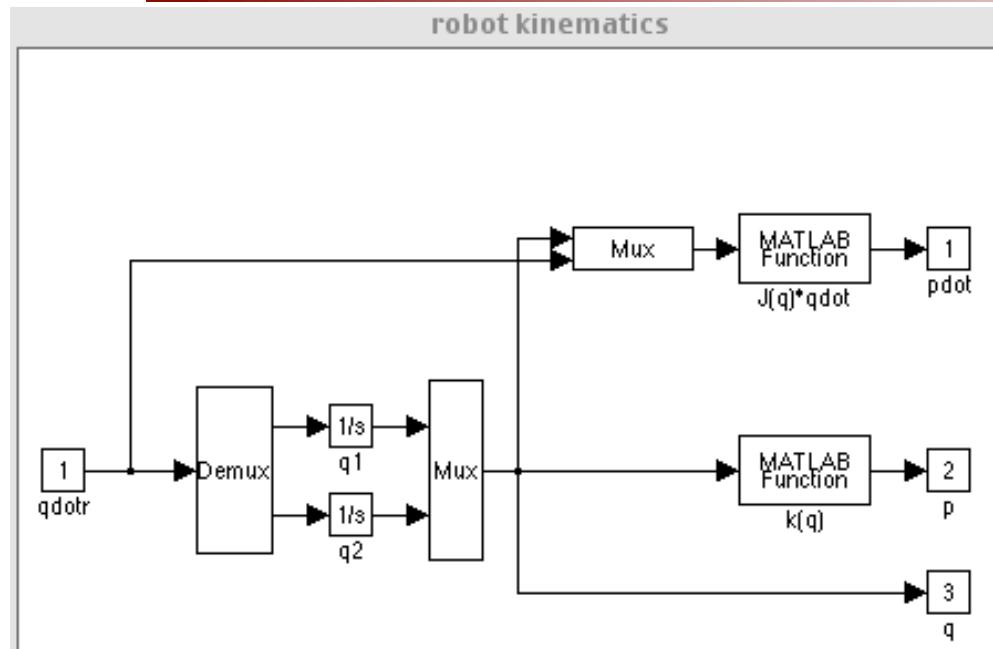
desired reference trajectory:
two types of tasks
1. straight line
2. circular path
both with constant speed

numerical integration method:
fixed step
Runge-Kutta
at 1 msec





Simulink blocks



calls to Matlab functions

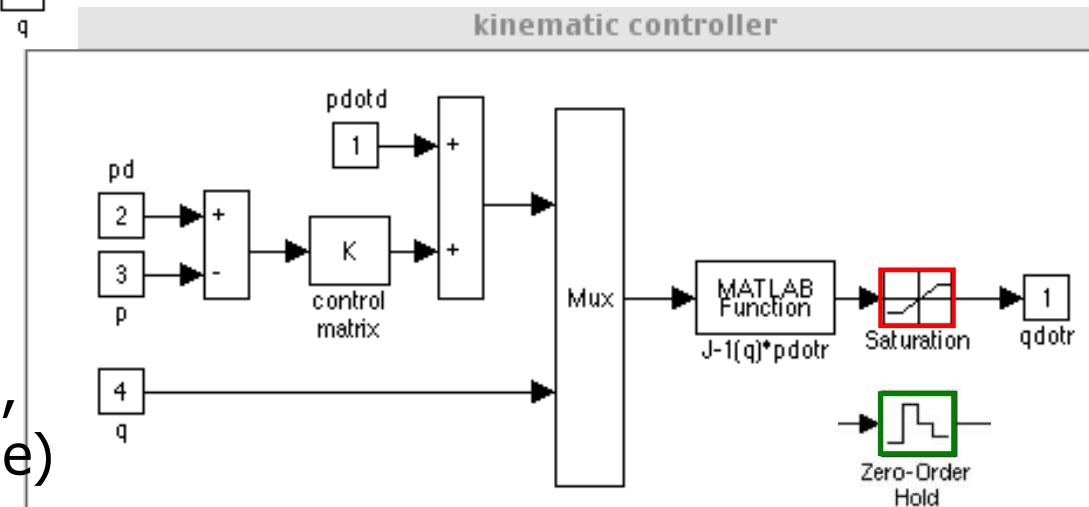
$k(q)=\text{dirkin}$ (user)

$J(q)=\text{jac}$ (user)

$J^{-1}(q)=\text{inv(jac)}$ (library)

- a **saturation** (for task 1.) or a **sample and hold** (for task 2.) added on joint velocity commands
- **system initialization** of kinematics data, desired trajectory, initial state, and control parameters (in **init.m** file)

never put "numbers" inside the blocks !





Matlab functions

dirkin.m

```
function [p] = dirkin(q)  
  
global l1 l2  
  
px=l1*cos(q(1))+l2*cos(q(1)+q(2));  
py=l1*sin(q(1))+l2*sin(q(1)+q(2));
```

jac.m

```
function [J] = jac(q)  
  
global l1 l2  
  
J(1,1)=-l1*sin(q(1))-l2*sin(q(1)+q(2));  
J(1,2)=-l2*sin(q(1)+q(2));  
J(2,1)=l1*cos(q(1))+l2*cos(q(1)+q(2));  
J(2,2)=l2*cos(q(1)+q(2));
```

init.m

```
% controllo cartesiano di un robot 2R  
% initialization  
  
clear all; close all  
global l1 l2  
  
% lunghezze bracci robot 2R  
  
l1=1; l2=1;  
  
% velocità cartesiana desiderata (costante)  
  
vxd=0; vyd=0.5;  
  
% tempo totale  
  
T=2;  
  
% configurazione desiderata iniziale  
  
q1d0=-45*pi/180; q2d0=135*pi/180;  
  
pd0=dirkin([q1d0 q2d0]');  
pxd0=pd0(1); pyd0=pd0(2);  
  
% configurazione attuale del robot  
  
q10=-45*pi/180; q20=90*pi/180;  
  
p0=dirkin([q10 q20]');  
  
% matrice dei guadagni cartesiani  
  
K=[20 20]; K=diag(K);  
  
%saturazioni di velocità ai giunti (input in deg/sec, convertito in rad/sec)  
  
vmax1=120*pi/180; vmax2=90*pi/180;
```

init.m
script
(for task 1.)



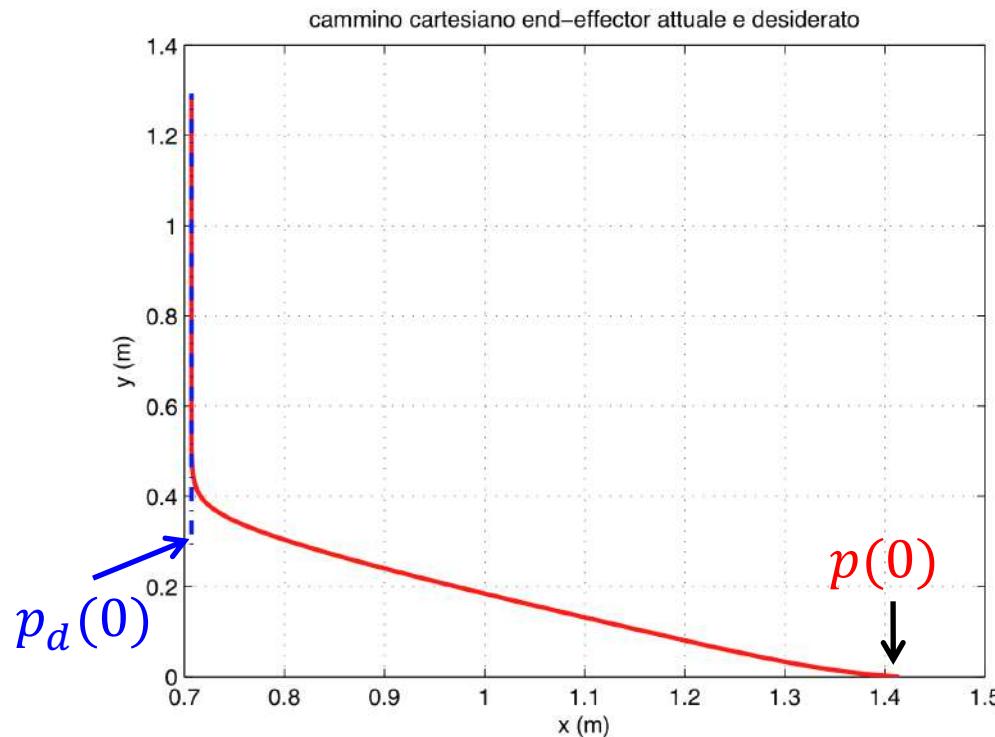
Simulation data for task 1

- straight line path with constant velocity
 - $x_d(0) = 0.7 \text{ m}$, $y_d(0) = 0.3 \text{ m}$; $v_{d,y} = 0.5 \text{ m/s}$, for $T = 2 \text{ s}$
- large initial error on end-effector position
 - $q(0) = (-45^\circ, 90^\circ) \Rightarrow e_p(0) = (-0.7, 0.3) \text{ m}$
- Cartesian control gains
 - $K_p = \text{diag}\{20, 20\}$
- (a) without joint velocity command saturation
- (b) with saturation ...
 - $v_{max,1} = 120^\circ/\text{s}$, $v_{max,2} = 90^\circ/\text{s}$

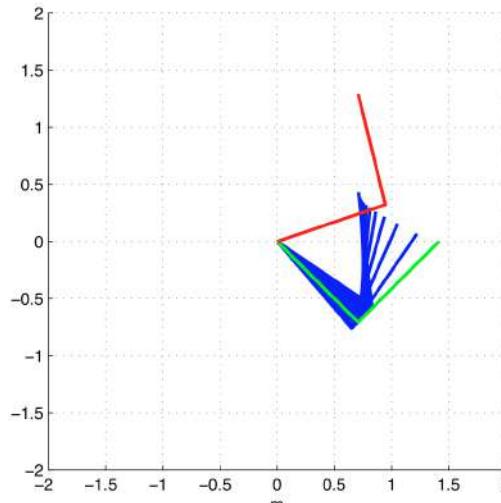


Results for task 1a

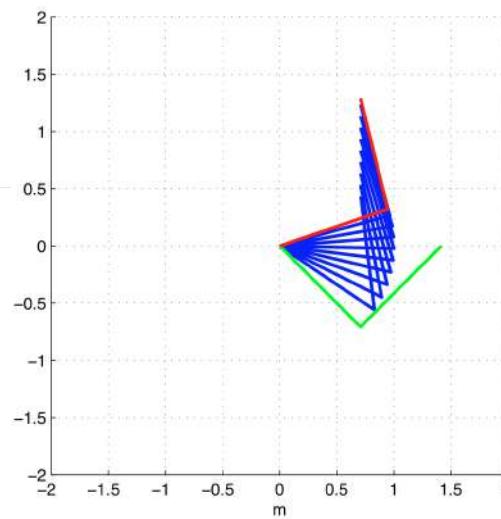
straight line: initial error, **no saturation**



path executed by the
robot end-effector
(**actual** and **desired**)



stroboscopic view of motion
(start and **end** configurations)



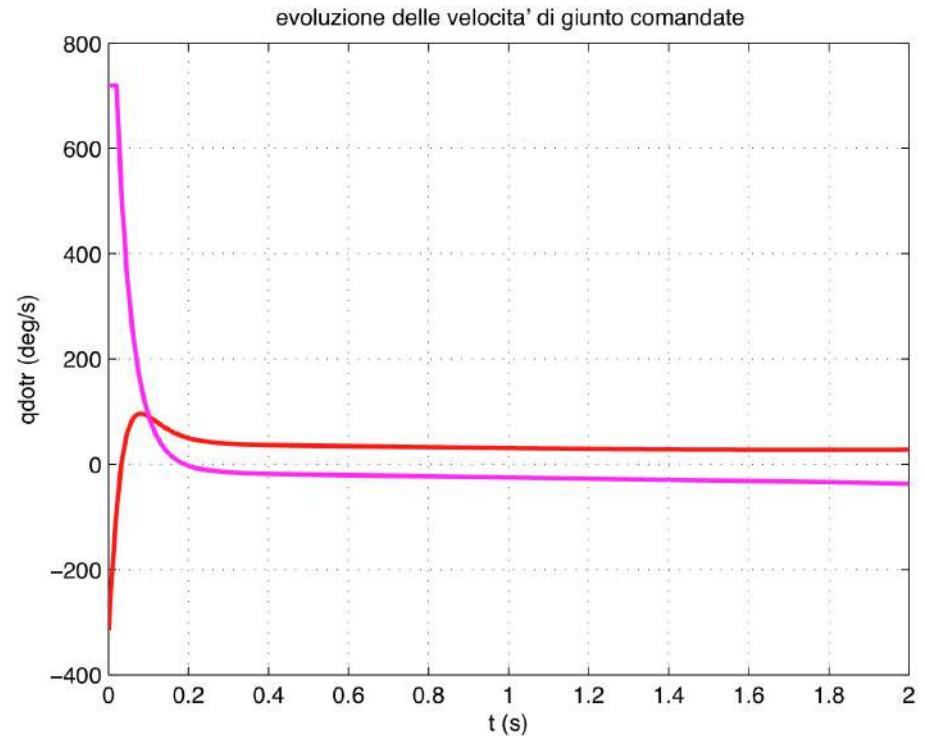
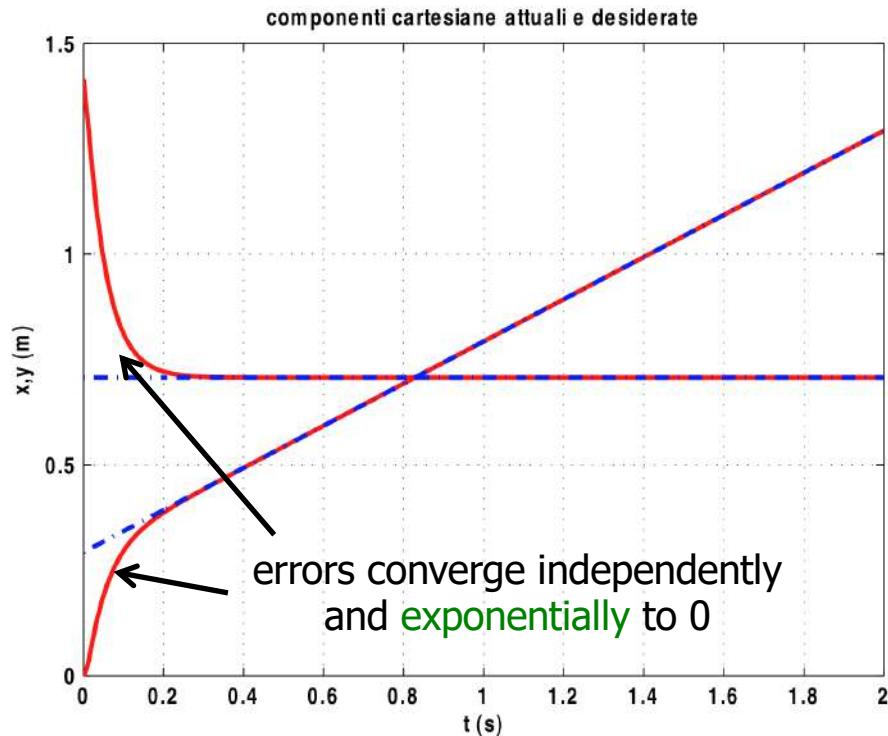
initial
transient
phase
(about 0.2 s)

trajectory
following
phase
(about 1.8 s)



Results for task 1a (cont)

straight line: initial error, **no** saturation



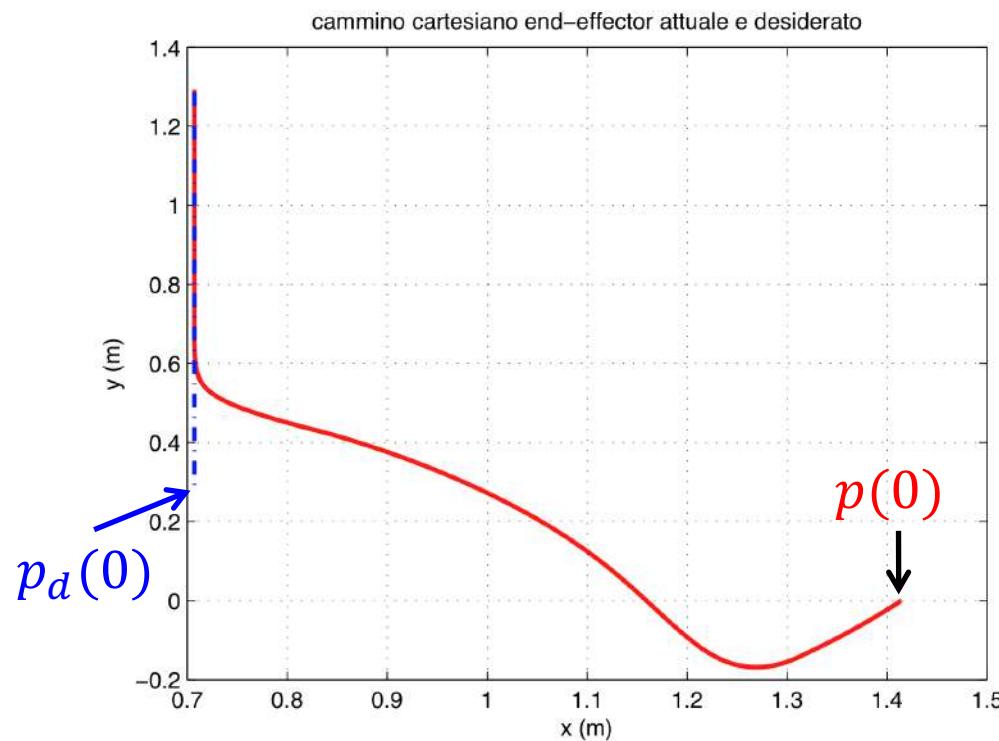
p_x, p_y actual and desired

control inputs $\dot{q}_{r1}, \dot{q}_{r2}$

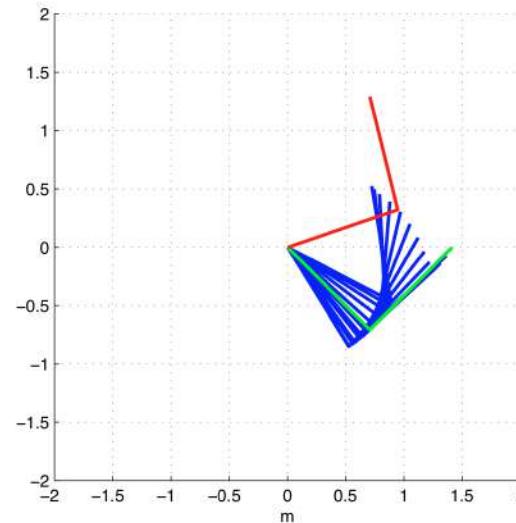


Results for task 1b

straight line: initial error, **with** saturation

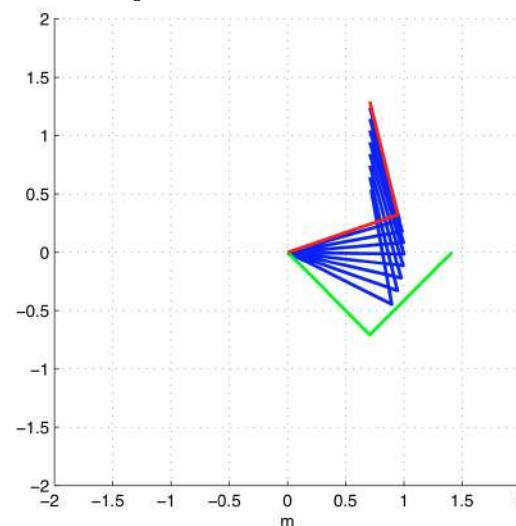


path executed by the
robot end-effector
(**actual** and **desired**)



initial
transient
phase
(about 0.5 s)

stroboscopic view of motion
(start and **end** configurations)

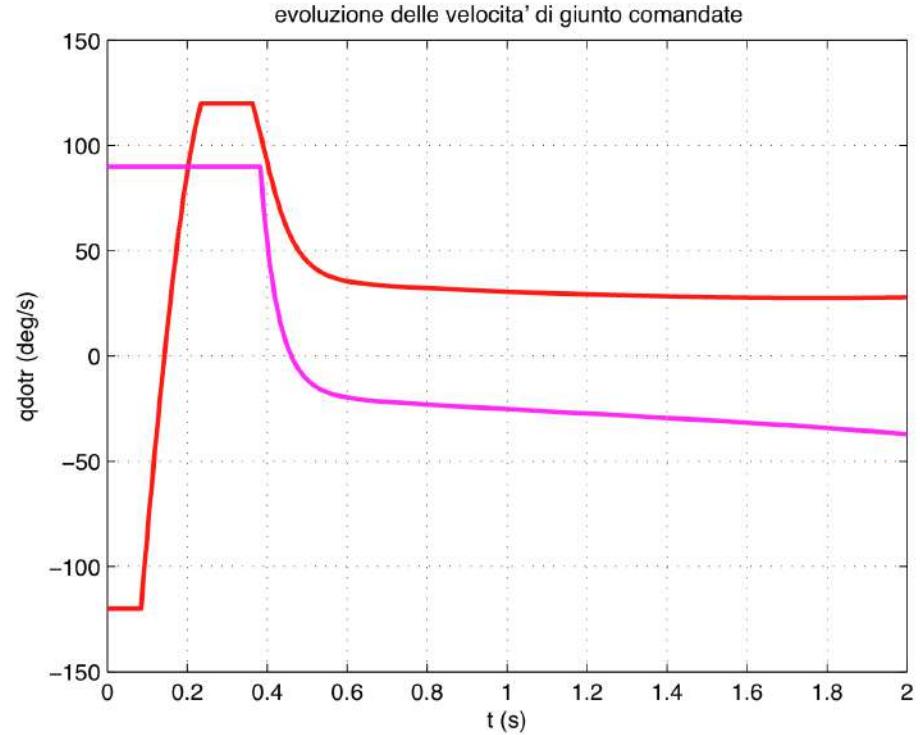
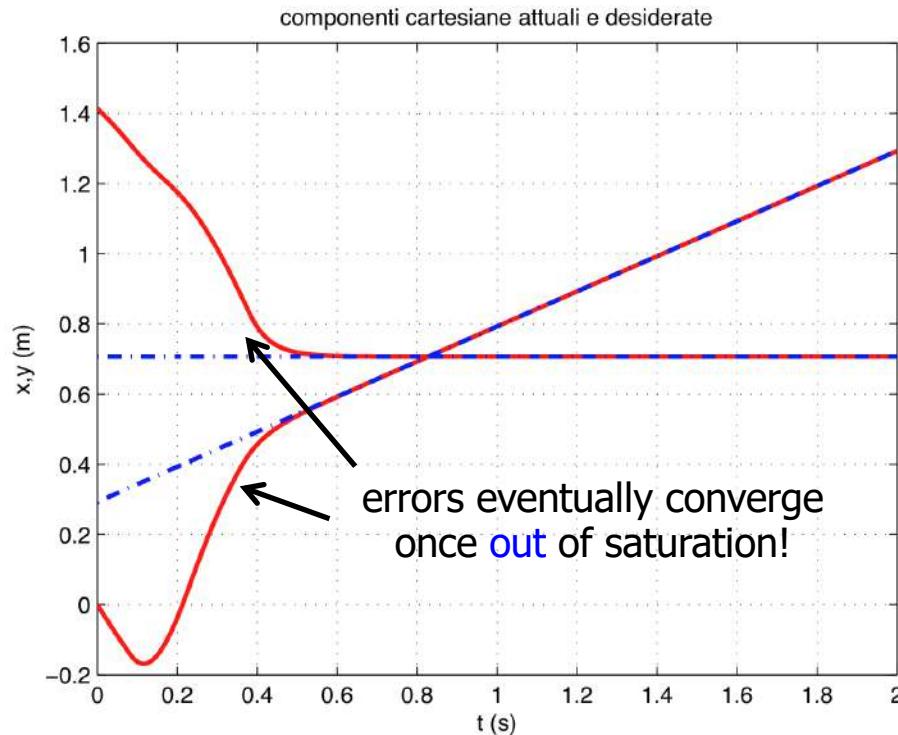


trajectory
following
phase
(about 1.5 s)



Results for task 1b (cont)

straight line: initial error, **with** saturation



p_x, p_y **actual** and **desired**

control inputs $\dot{q}_{r1}, \dot{q}_{r2}$
(saturated at $\pm v_{max,1}, \pm v_{max,2}$)



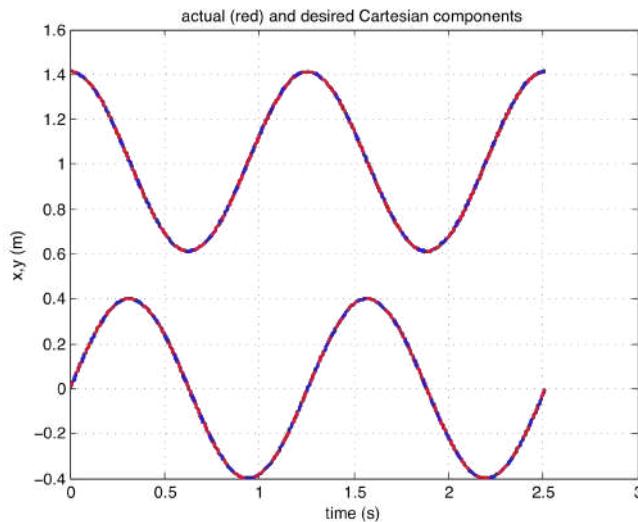
Simulation data for task 2

- **circular** path with constant velocity
 - centered at $(1.014, 0)$ with radius $R = 0.4$ m;
 - $v = 2$ m/s, performing **two** rounds $\Rightarrow T \approx 2.5$ s
- zero initial error on Cartesian position ("match")
 - $q(0) = (-45^\circ, 90^\circ) \Rightarrow e_p(0) = 0$
- (a) ideal **continuous** case (1 kHz), even **without** feedback
- (b) **with** sample and hold (ZOH) of $T_{hold} = 0.02$ s (joint velocity command updated at 50 Hz), but **without** feedback
- (c) as before, but **with** Cartesian feedback using the gains
 - $K_p = \text{diag}\{25, 25\}$

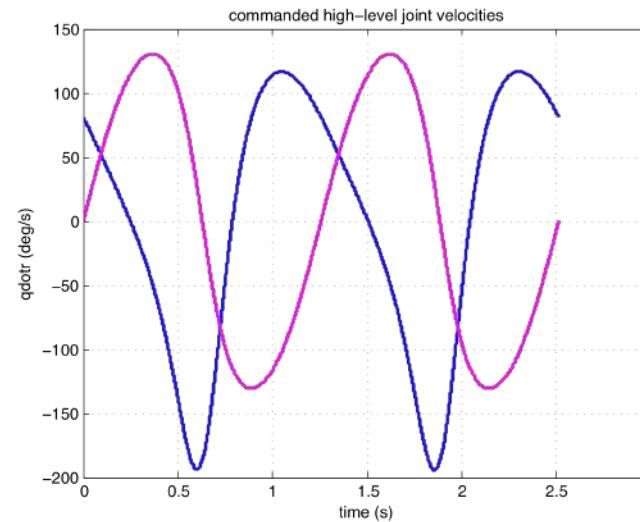


Results for task 2a

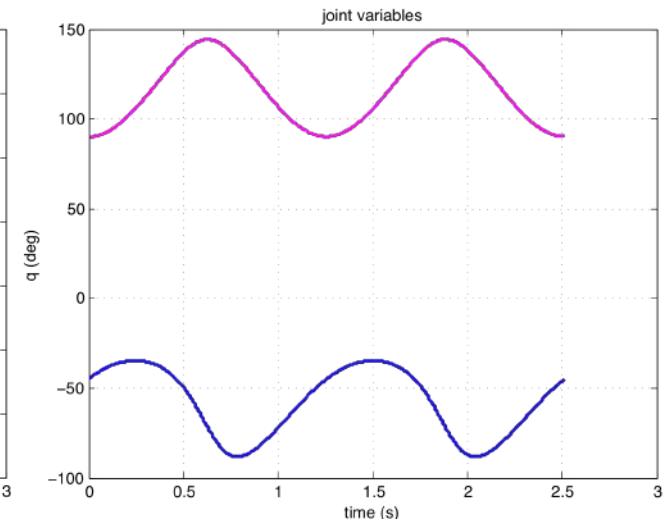
circular path: no initial error, **continuous** control (ideal case)



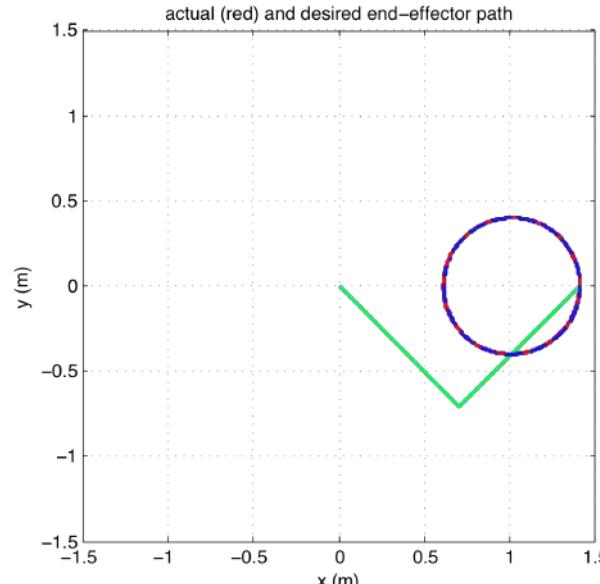
p_x, p_y actual and desired



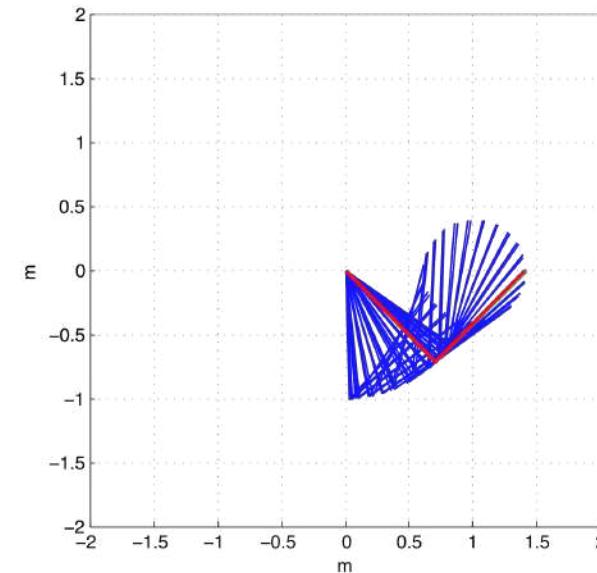
control inputs $\dot{q}_{r1}, \dot{q}_{r2}$



joint variables q_1, q_2



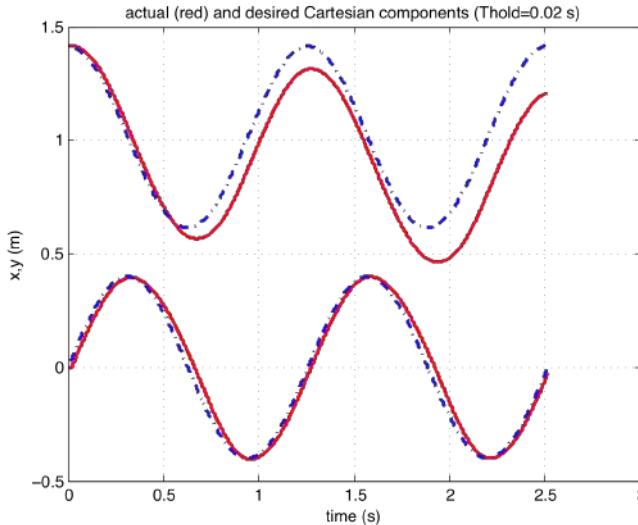
zero tracking error is kept at all times



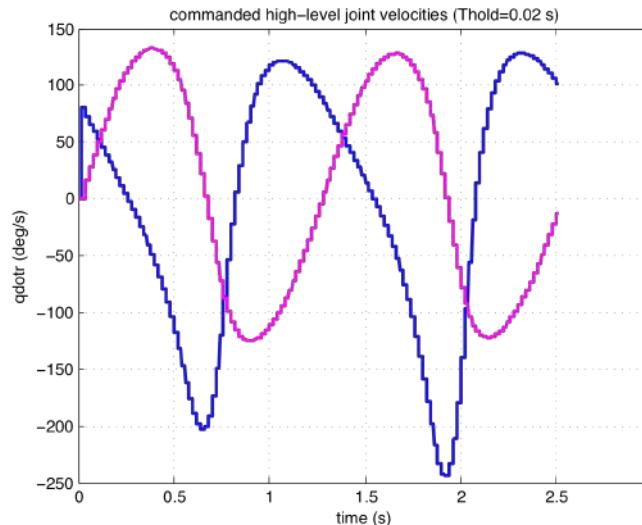


Results for task 2b

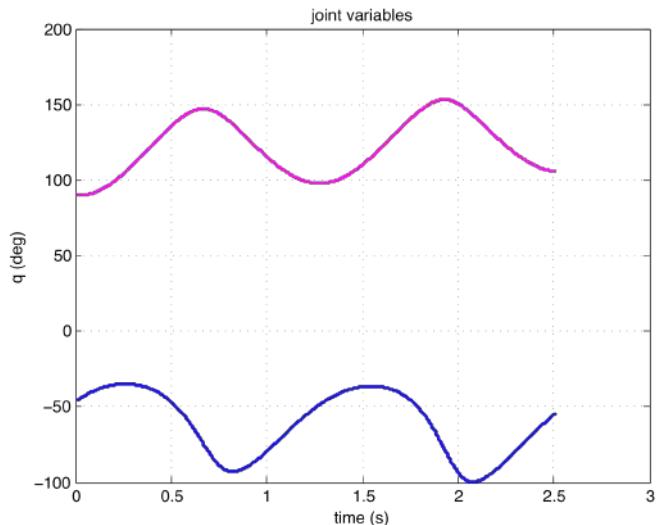
circular path: no initial error, **ZOH** at 50 Hz, **no feedback**



p_x, p_y actual and desired

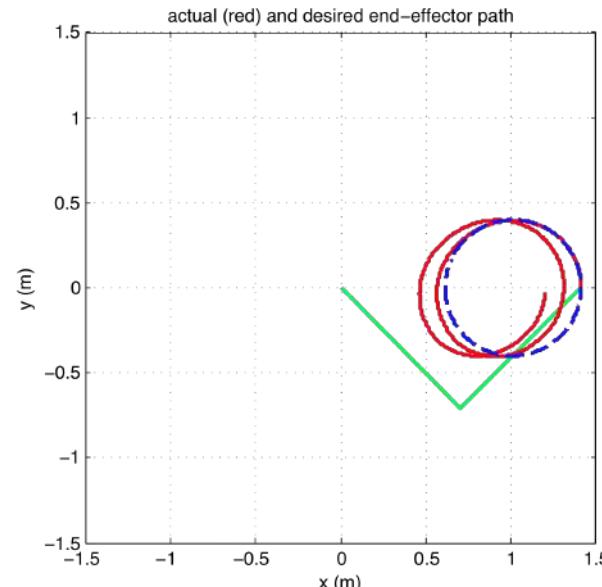


control inputs $\dot{q}_{r1}, \dot{q}_{r2}$

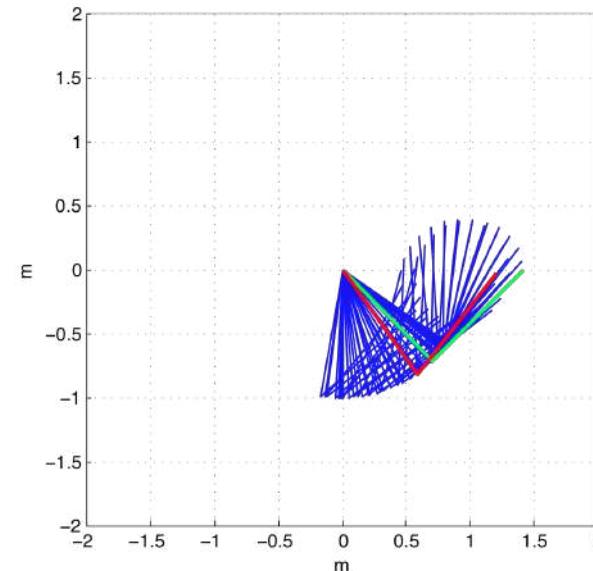


joint variables q_1, q_2

a drift occurs along the path due to the "linearization error" along the path tangent



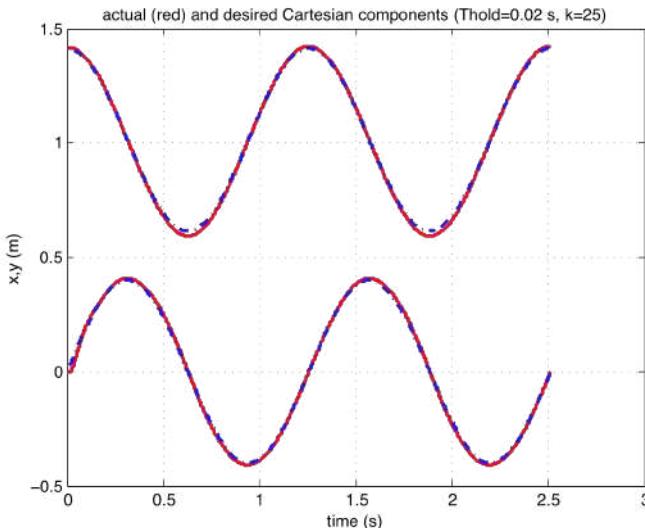
final configuration (after two rounds) differs from initial configuration



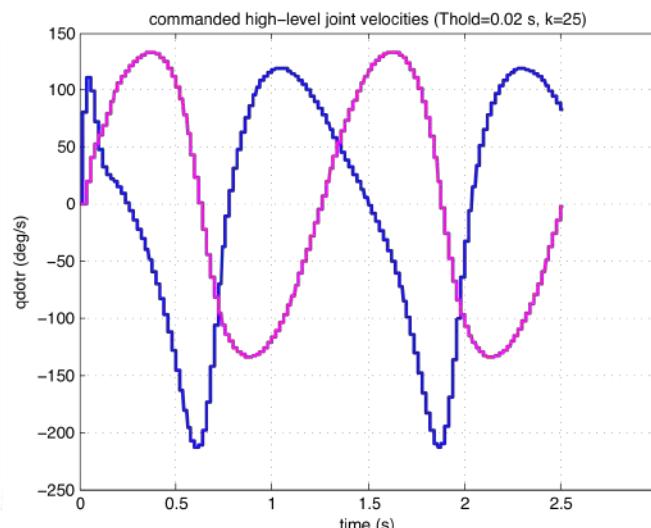


Results for task 2c

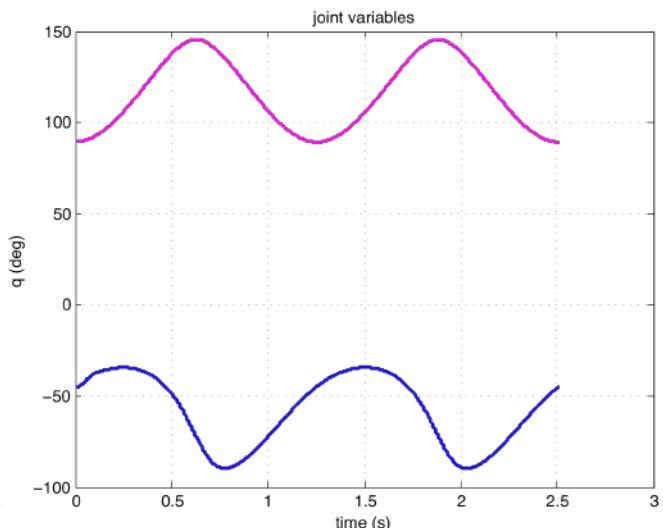
circular path: no initial error, **ZOH** at 50 Hz, **with feedback**



p_x, p_y actual and desired

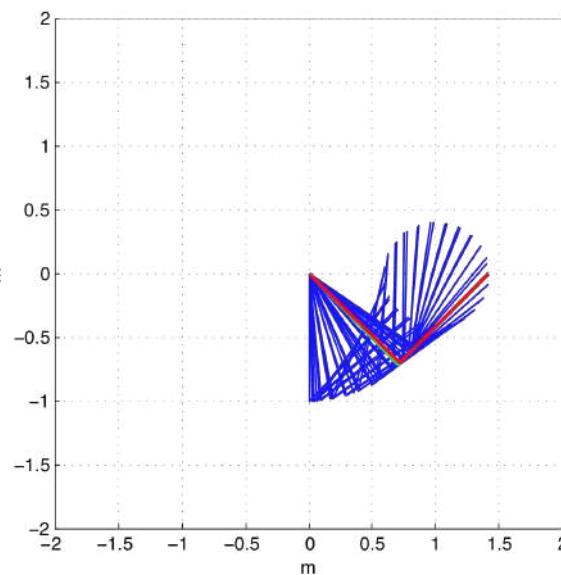
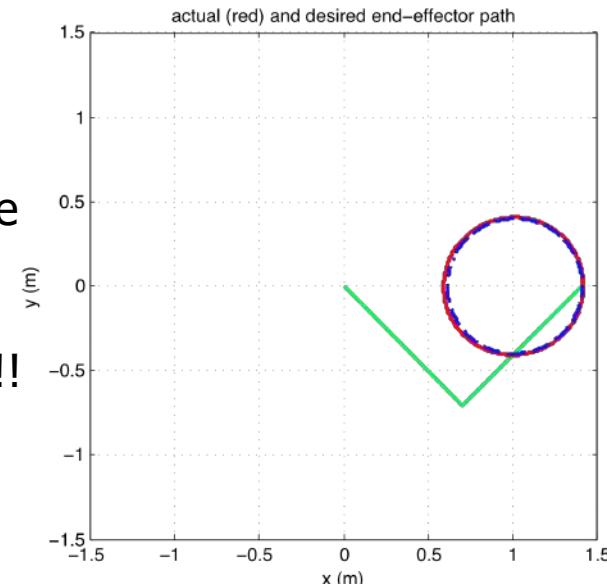


control inputs $\dot{q}_{r1}, \dot{q}_{r2}$



joint variables q_1, q_2

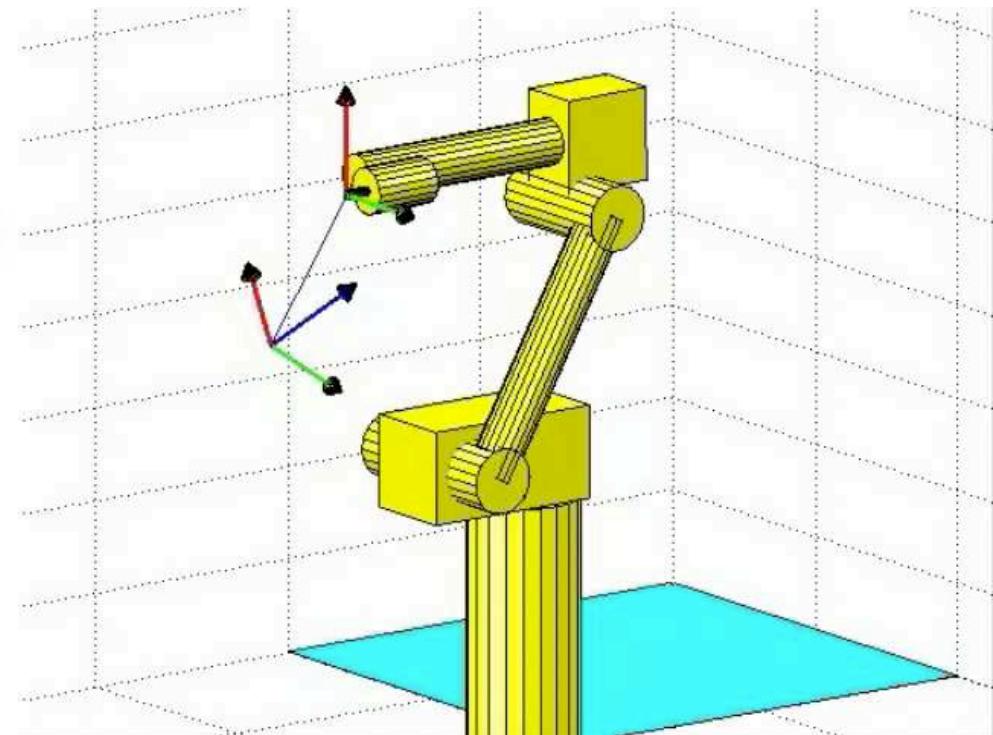
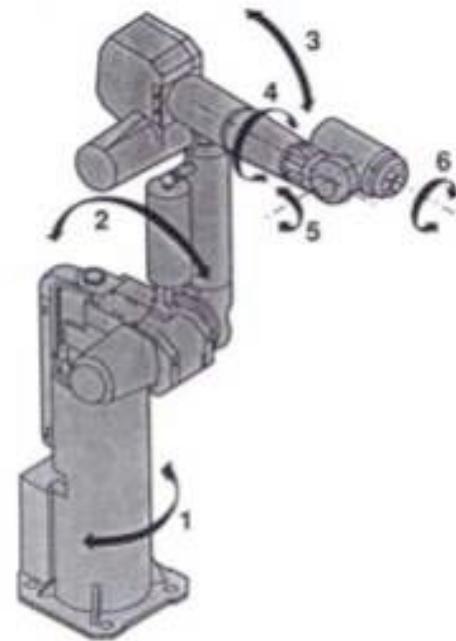
(almost) the same performance of the continuous case is recovered!!



note however that larger P gains will eventually lead to unstable behavior (see: stability problems for discrete-time control systems)



3D simulation



video

kinematic control of Cartesian motion of Fanuc 6R (Arc Mate S-5) robot
simulation and visualization in Matlab



Kinematic control of KUKA LWR

[video](#)



Discrete-Time Redundancy Resolution at the Velocity Level with Acceleration/Torque Optimization Properties

Fabrizio Flacco Alessandro De luca

Robotics Lab, DIAG
Sapienza University of Rome

September 2014

kinematic control of Cartesian motion with redundancy exploitation
velocity vs. acceleration level



Robotics 1

Dynamic control of a single axis

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



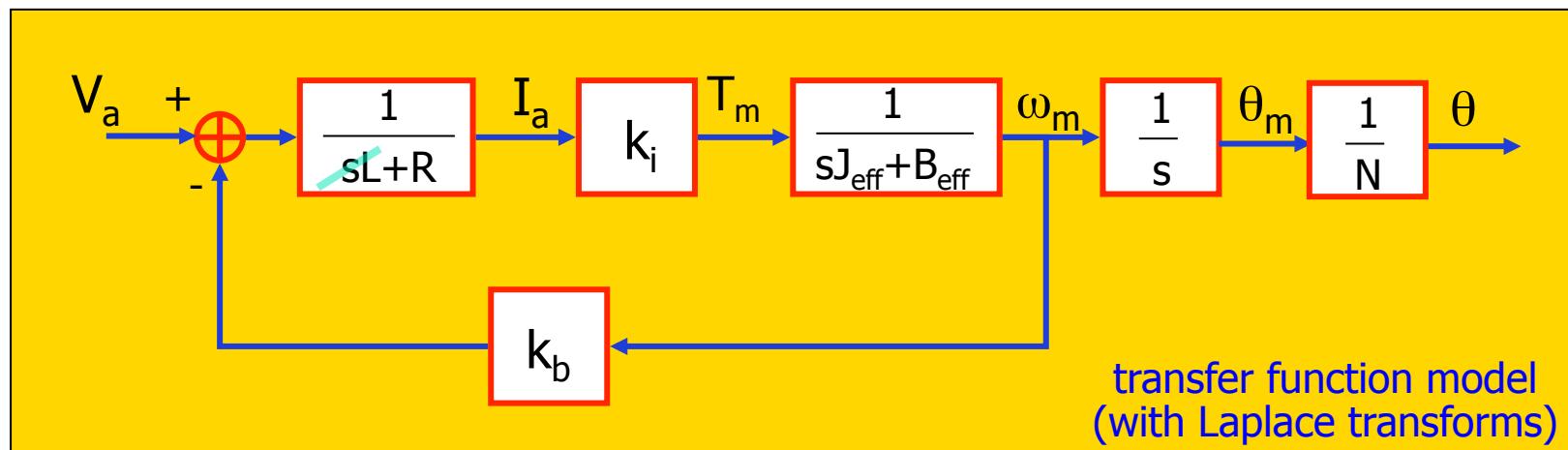
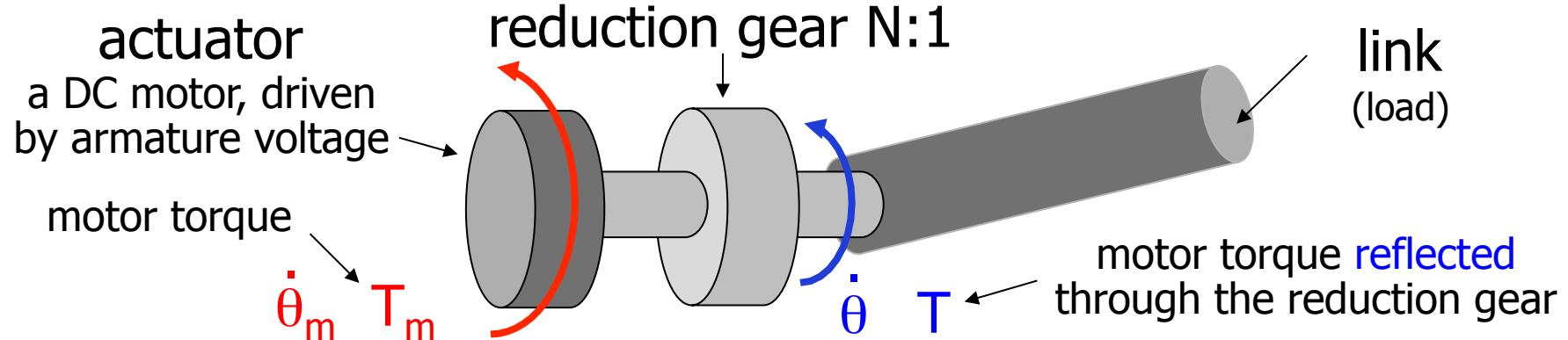


Dynamic control (single axis)

- when **dynamic issues** associated to the desired motion become relevant, one should consider robot mass/inertia and dissipative effects (friction) in the **control design**
- for a multi-dof articulated robot, the dynamics of **each link** is subject also to forces/torques due to
 - motion couplings with other links (**inertial**, **centrifugal**)
 - its own motion simultaneous with that of other links (**Coriolis**)
 - static loads (**gravity**, **contact forces**)
- the effects of these nonlinear couplings and loads can be partly “masked” in the dynamic behavior of a joint axis/motor load
 - if transmissions with **high reduction ratios** ($N \geq 100$) are used
- we will consider next the dynamic control design for a **single joint axis** of a robot (**decentralized** approach)



Dynamic model of a single robot axis



effective inertia

$$J_{\text{eff}} = J_m + \frac{1}{N^2} J_l$$

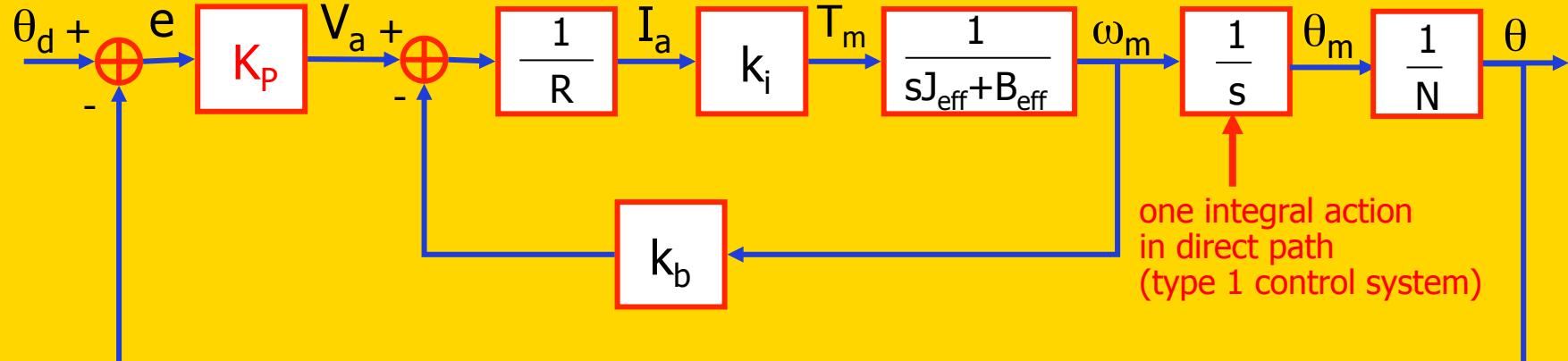
$$\approx 10^{-4}$$

$$B_{\text{eff}} = B_m + \frac{1}{N^2} B_l$$

effective viscous friction



P control (Proportional to the error)



closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{\theta/e}{1+\theta/e} = \frac{K_p k_i}{NR J_{\text{eff}}} \frac{1}{s^2 + \frac{R B_{\text{eff}} + k_i k_b}{R J_{\text{eff}}} s + \frac{K_p k_i}{NR J_{\text{eff}}}}$$

always ASYMPTOTICALLY STABLE for $K_p > 0$



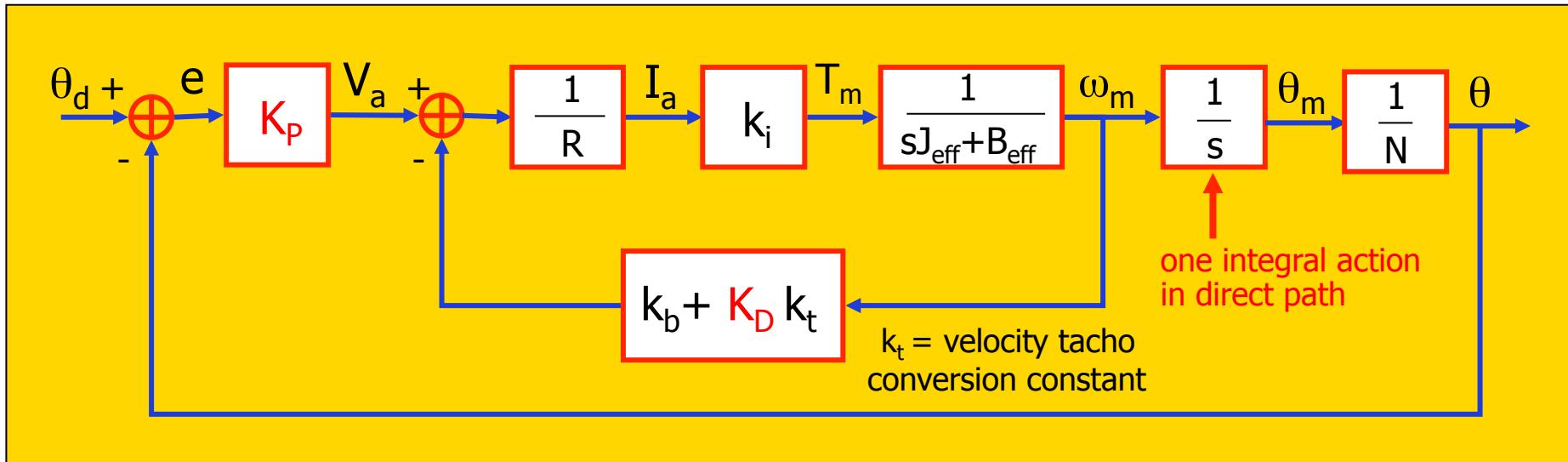
Comments on P controller

- for $\theta_d = \text{constant}$, the **steady-state** error is always **zero**
 - type 1 control system
- just **one** control design parameter (the gain K_p)
 - the (two) closed-loop poles cannot be independently assigned
 - in particular, the **natural frequency** ω_n and **damping ratio** ζ of this (complex) pole pair are coupled
- **transient response** and/or disturbance rejection features may **not** be satisfactory

note: variable measured for feedback is most often the motor position θ_m (where the encoder is usually mounted) $\Rightarrow \theta = \theta_m/N$



PD control (Proportional-Derivative)



closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{\theta/e}{1+\theta/e} = \frac{K_P k_i}{NR J_{\text{eff}}} \frac{1}{s^2 + \frac{RB_{\text{eff}}+k_i(k_b+K_D k_t)}{R J_{\text{eff}}}} s + \frac{K_P k_i}{NR J_{\text{eff}}}$$

always ASYMPTOTICALLY STABLE for $K_P, K_D > 0$



Comments on PD controller

- for $\dot{\theta}_d = \text{constant}$, $\dot{e} = -\dot{\theta}$, this scheme implements a PD action on the position error
- for $\dot{\theta}_d \neq \text{constant}$, in order to obtain a “true” PD action on the position error e (on the load side), the input reference to the control loop should be modified as

$$\theta_d + \dot{\theta}_d (Nk_t K_D) / K_P \quad \text{often neglected for large } K_P$$

- K_P and K_D are chosen so as to yield smooth/fast transients
 - damping ratio $\zeta \geq 0.7$ (at $\zeta = 1$, two coincident negative real poles)
 - natural frequency $\omega_n < 0.5 \omega_r$, where ω_r is the (lowest) resonance frequency of the joint assembly structure (with “braked” motor)
 - such a resonance (caused by the un-modeled **elasticity** of the transmission gears) should non be excited by the control law
 - current industrial robots have typically $f_r = \omega_r / 2\pi = 4 \div 20 \text{ Hz}$



Simulation data

Matlab/Simulink

% Simulation parameters for the first (base) joint of the Stanford robot arm

% motor (U9M4T)

$K_i = 0.043$; % torque/current constant [Nm/A]
 $B_m = 0.00008092$; % viscous friction coefficient [Nm s/rad]
 $K_b = 0.04297$; % back emf constant [V s/rad]
 $L = 0.000100$; % inductance of the equivalent armature circuit [H], negligible
 $R = 1.025$; % resistance of the equivalent armature circuit [Ohm]
 $J_a = 0.000056$; % inertia of motor+tachometer assembly [Nm s^2/rad]

% velocity tachometer (Photocircuits 030/105)

$K_t = 0.02149$; % tachometer conversion constant [V s/rad]

% reduction

$n = 0.01$; % inverse of reduction ratio (= 1/N)

% load

$J_l = 5$; % inertia on the link side [Nm s^2/rad] (varies from 1.4 to 6.17)
 $B_l = 0$; % viscous friction coefficient on the link side (N/A)
 $\omega_{mr} = 25.13$; % resonant frequency (at nominal J_l) [rad/s] (4 Hz)

% computed parameters

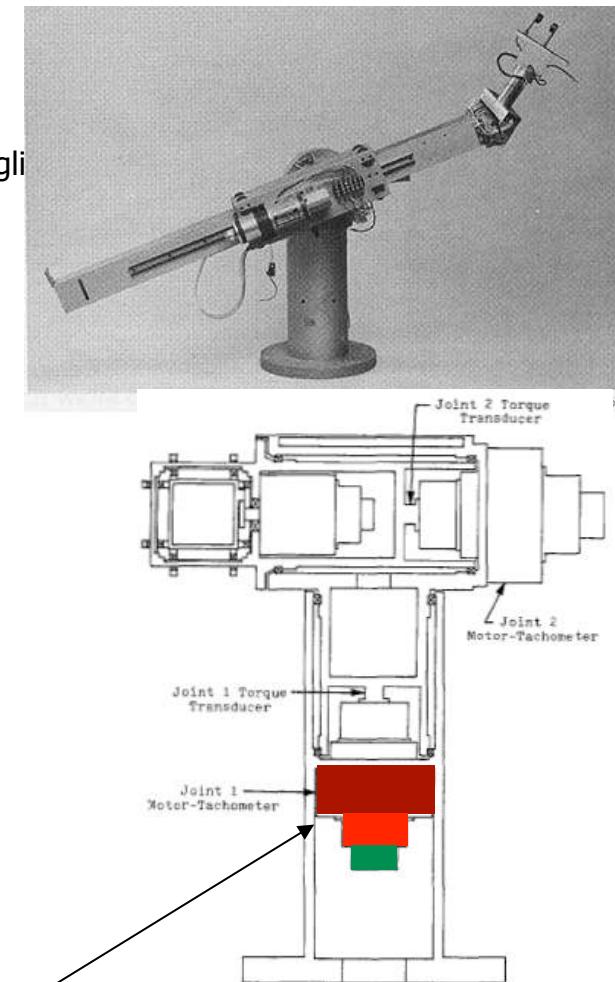
$B_{eff} = B_m + B_l * n^2$; % effective viscous friction coefficient
 $J_{eff} = J_a + J_l * n^2$; % effective inertia

% reference input

$q_{des} = 1$; % desired joint angle value (for step input case) [rad]
 $K_{ram} = 2$; % angular coefficient (for position ramp input) [rad/s]

% possible "hard" nonlinearities

$F_m = 0.042$; % dry friction torque [Nm]
 $D = 0.0087$; % reduction gear backlash [rad] (0.5 deg)
 $T_{max} = 4$; % motor torque saturation level [Nm]

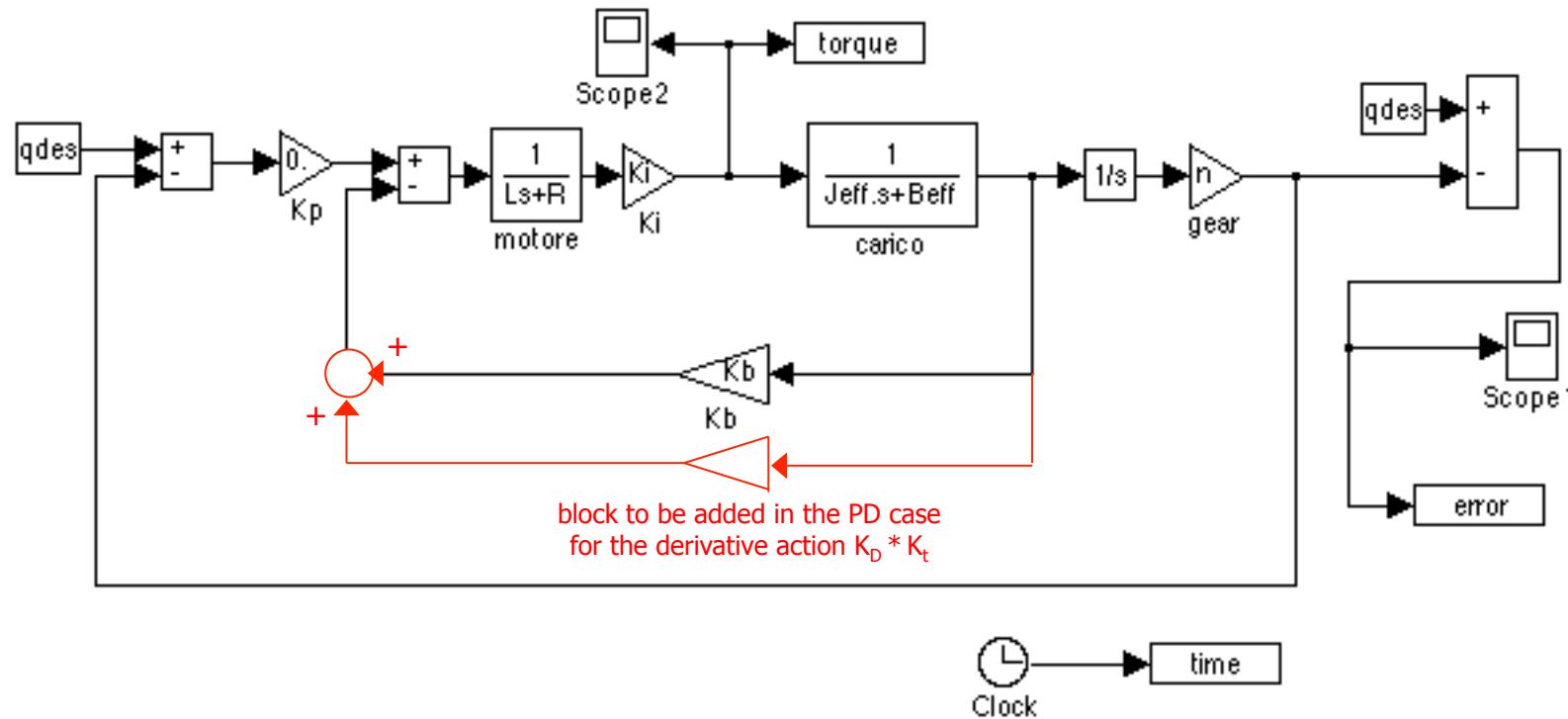


motor, velocity tachometer, optical encoder



Simulink block diagram

dynamic model and P/PD control

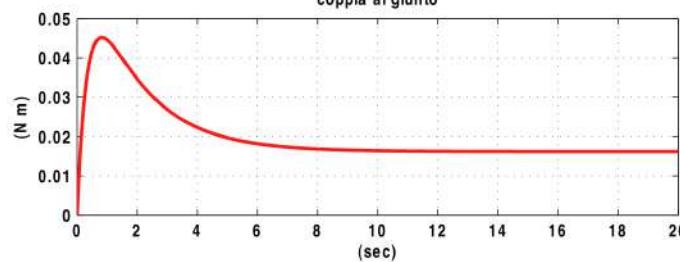
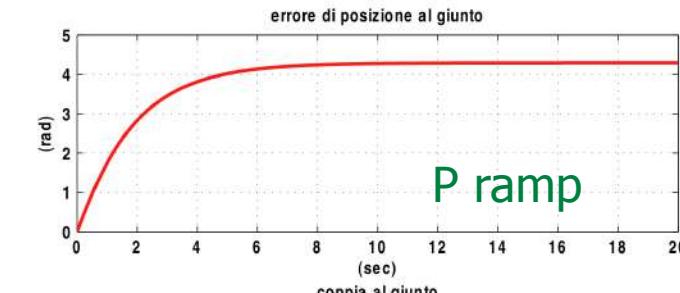
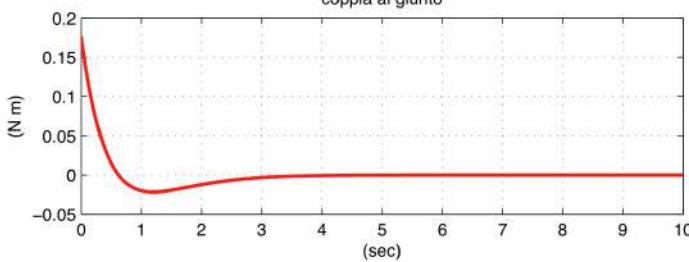
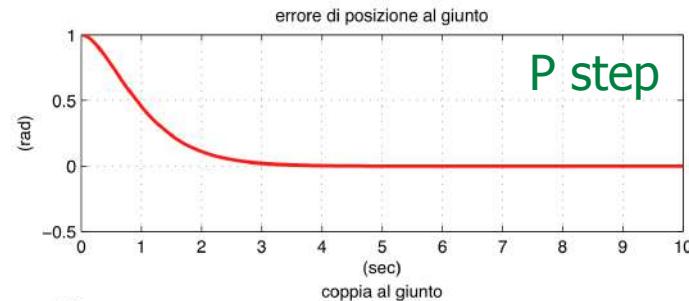


- **P control law:** $K_p = 4.2$ (the **maximum** value that guarantees motion transients **without oscillations**)
- **PD control law:** $K_p = 209$, $K_d = 15.4$ (such as to obtain a \approx **critically damped** transient behavior)



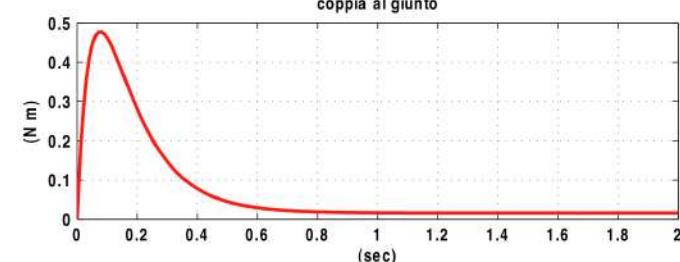
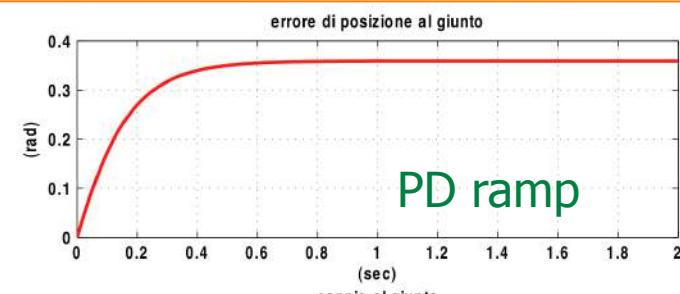
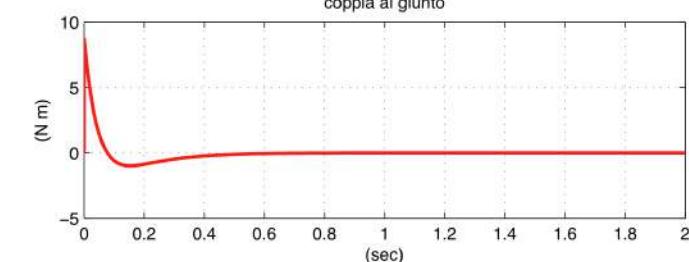
P/PD control results

step (1 rad) and ramp (2 rad/s) responses



position
error

control
torque

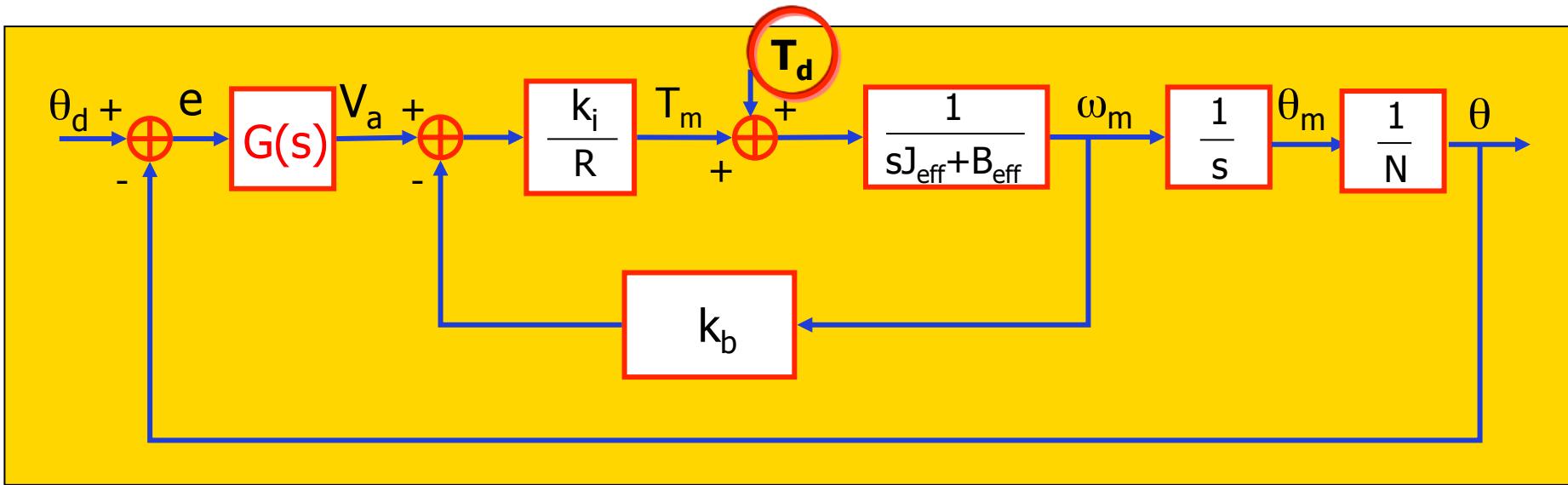


position
error

control
torque



General case (n joints)



T_d = disturbance torque due to **inertial couplings** with other links/axes, **centrifugal/Coriolis terms**, and **gravity** (only position-dependent)

in order to obtain **zero error** at steady state at least for a **constant disturbance** (robot at rest, under gravity), an **integral action** should be added in the direct path **before** the disturbance entry point (**astatic** control behavior)

→ $G(s)$ = PID controller



PID control

(Proportional-Integral-Derivative)

- $G(s) = K_P + K_I/s + K_D s$
 - as usual, the derivative (anticipative) action must be **low-pass filtered** in order to be physically realizable
- closed-loop transfer function

$$\frac{\theta(s)}{\theta_d(s)} = \frac{(K_D s^2 + K_P s + K_I) k_i}{NRJ_{\text{eff}} s^3 + (NRB_{\text{eff}} + Nk_b k_i + K_D k_i)s^2 + k_i K_P s + k_i K_I}$$

- **asymptotic stability** if and only if (Routh criterion)

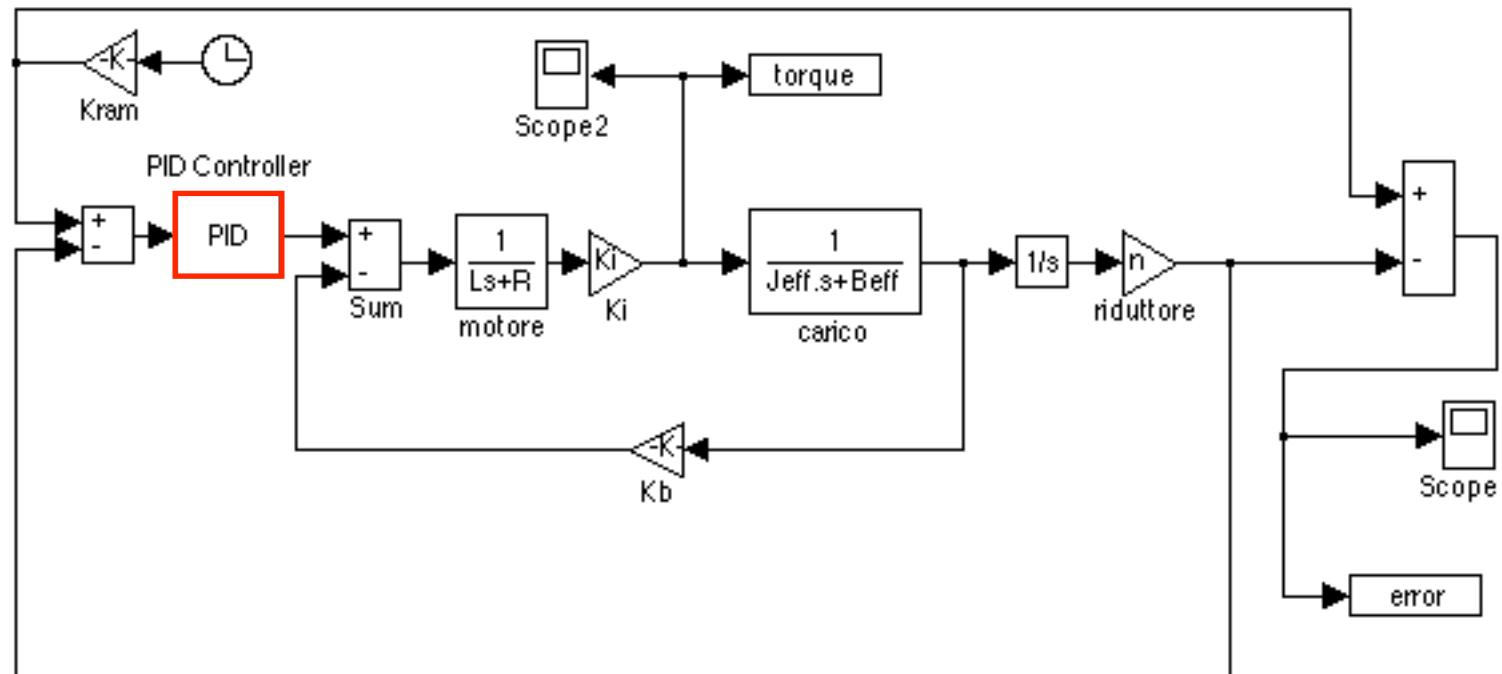
$$0 < K_I < K_P/RJ_{\text{eff}} \underbrace{(RB_{\text{eff}} + K_D k_i/N + k_b k_i)}_{> 0} \quad > 0$$

- control system of **type 2** and **astatic** w.r.t. disturbance



Simulink block diagram

dynamic model and PID control

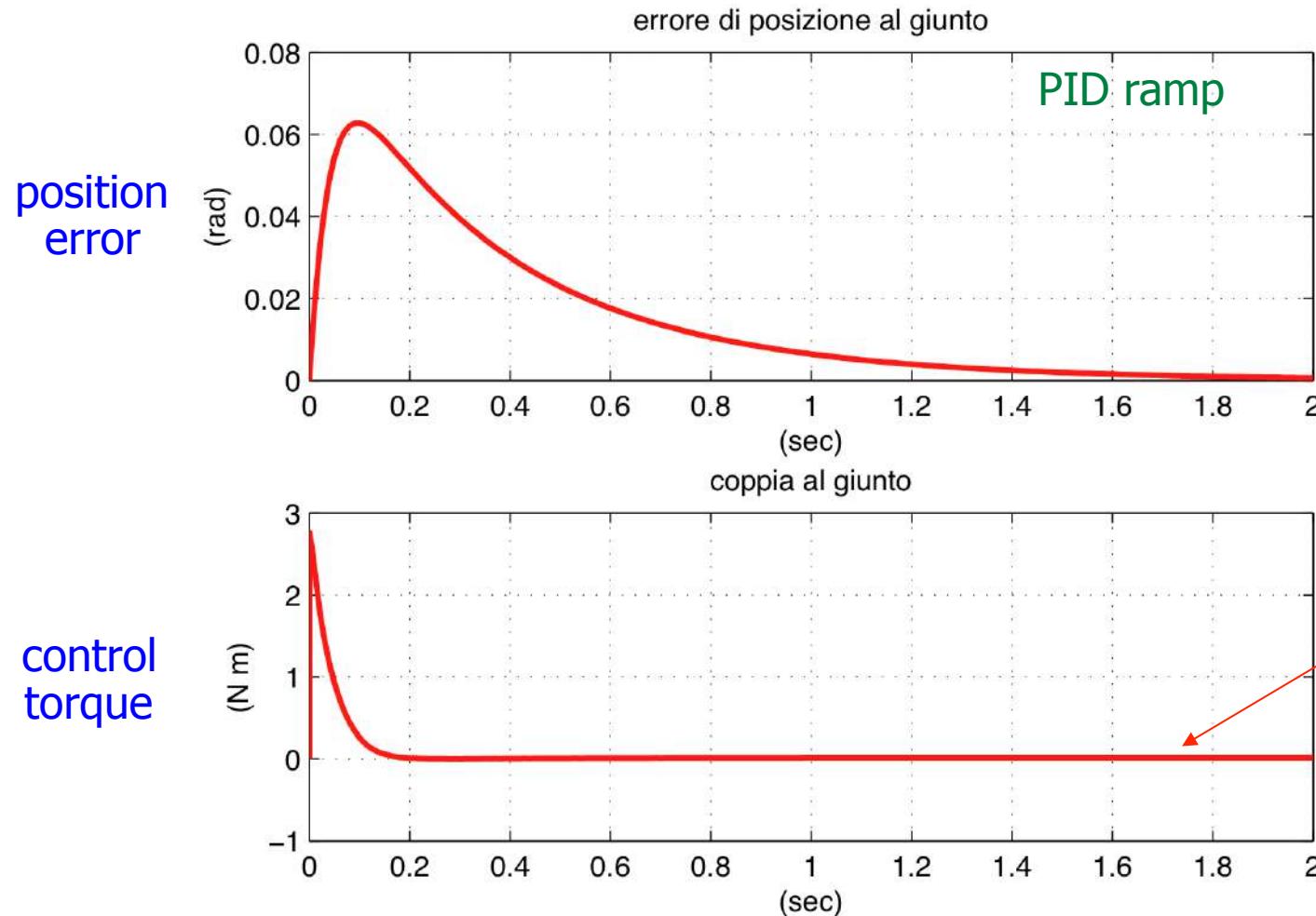


- gain after some **tuning**: $K_p = 209$ (as for PD law), $K_d = 33$, $K_i = 296$
- type 2 control system \Rightarrow **zero** steady-state error on position **ramp** inputs



PID control results

ramp (2 rad/s) response



note: the torque
at steady-state
is NOT zero
(≈ 0.02 Nm)



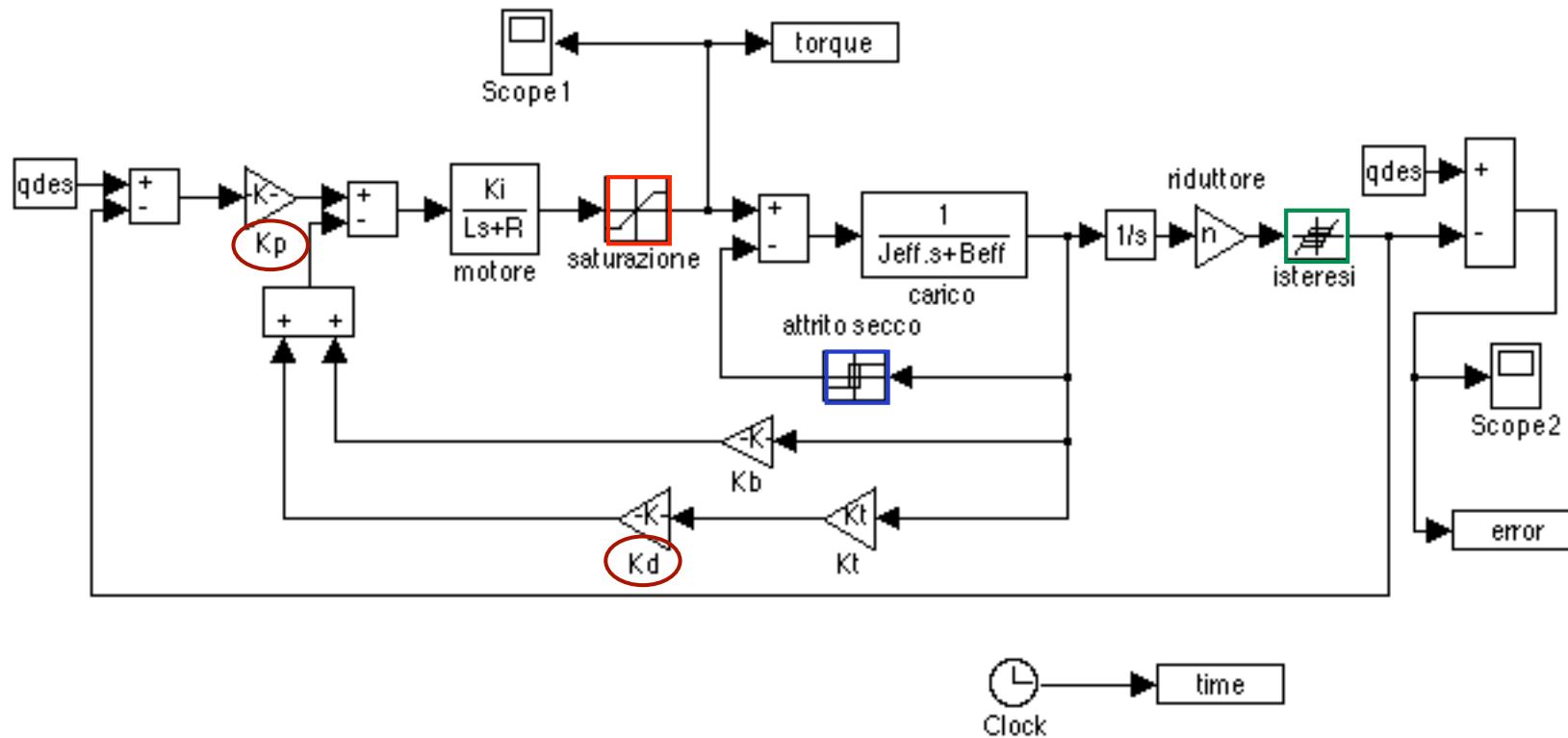
Final remarks

- there are many **non-linear physical phenomena** that cannot be directly considered in control design and analysis based on linear models
 - actuator saturations
 - transmission/gear backlash (delay, hysteresis)
 - dry friction and static friction
 - sensor quantization (encoder)
 - ...
- approximate mathematical **models** can be obtained and then **simulated** in combination with the already designed control law, for a more realistic validation of system behavior and control performance
- similarly, **uncertainties on nominal parameters** of robot kinematics/dynamics can be included in the simulation



Simulink block diagram

dynamic model with nonlinear phenomena and PD control

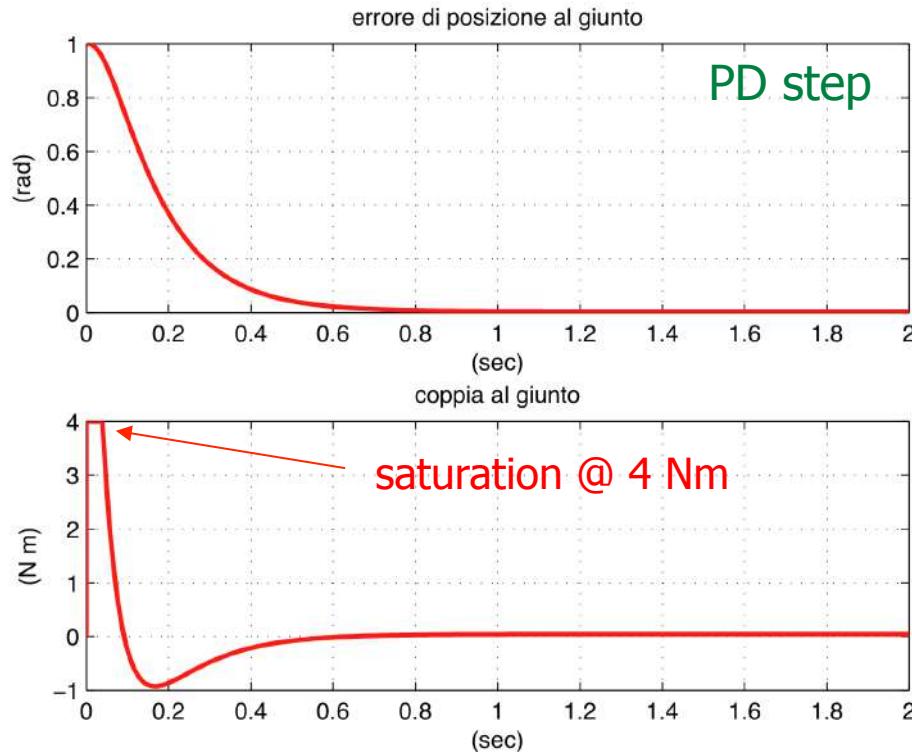


- actuator **saturation**, **dry friction**, **backlash** in reduction gears
- PD control



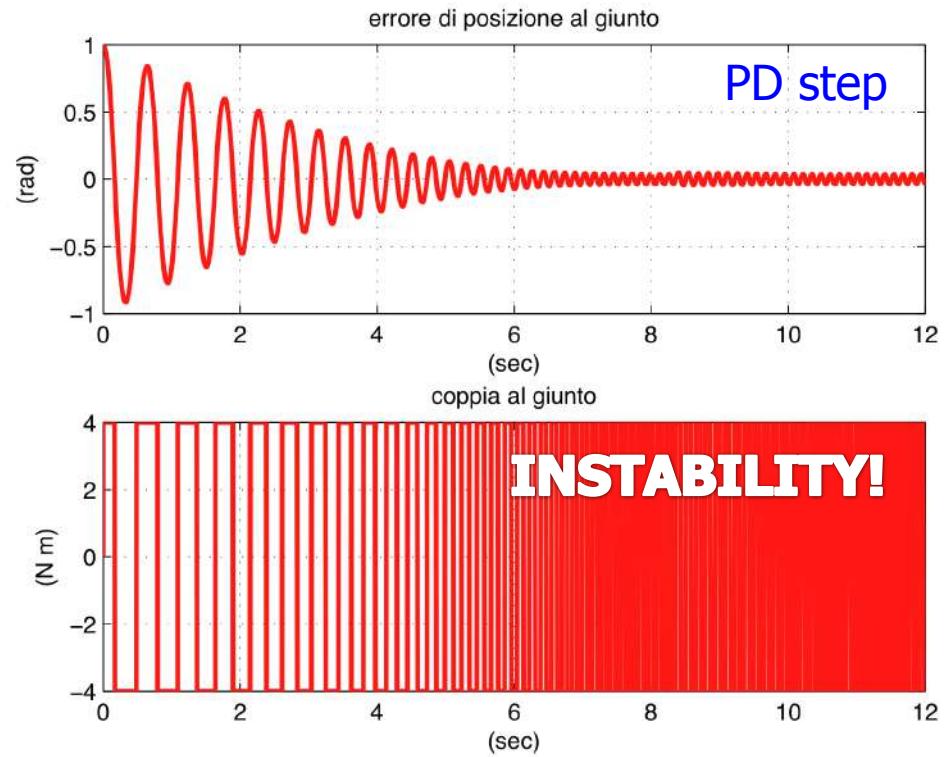
PD control results

step (1 rad) response with non-idealities



same PD gains as before

gears are always engaged
(already when motion starts)



with larger P gain....

gears initially engaged, but not
when velocity inversion occurs
→ “chattering” due to backlash