2024 CIDTA Summer School

Robotics Workshop

David Vernon
Carnegie Mellon University Africa

www.vernon.eu

www.vernon.eu/talks/CIDTA_Summer_School-Robotics_Workshop_2024.pdf

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

Principles

Demonstrations

1. Locomotion and navigation using odometry
2. Pick-and-place application using a Lynxmotion AL5D robot arm
3. Gesture execution by a social robot using biological motion
4. Visual and aural attention by a social robot

Practice

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

Principles

What is a Robot?

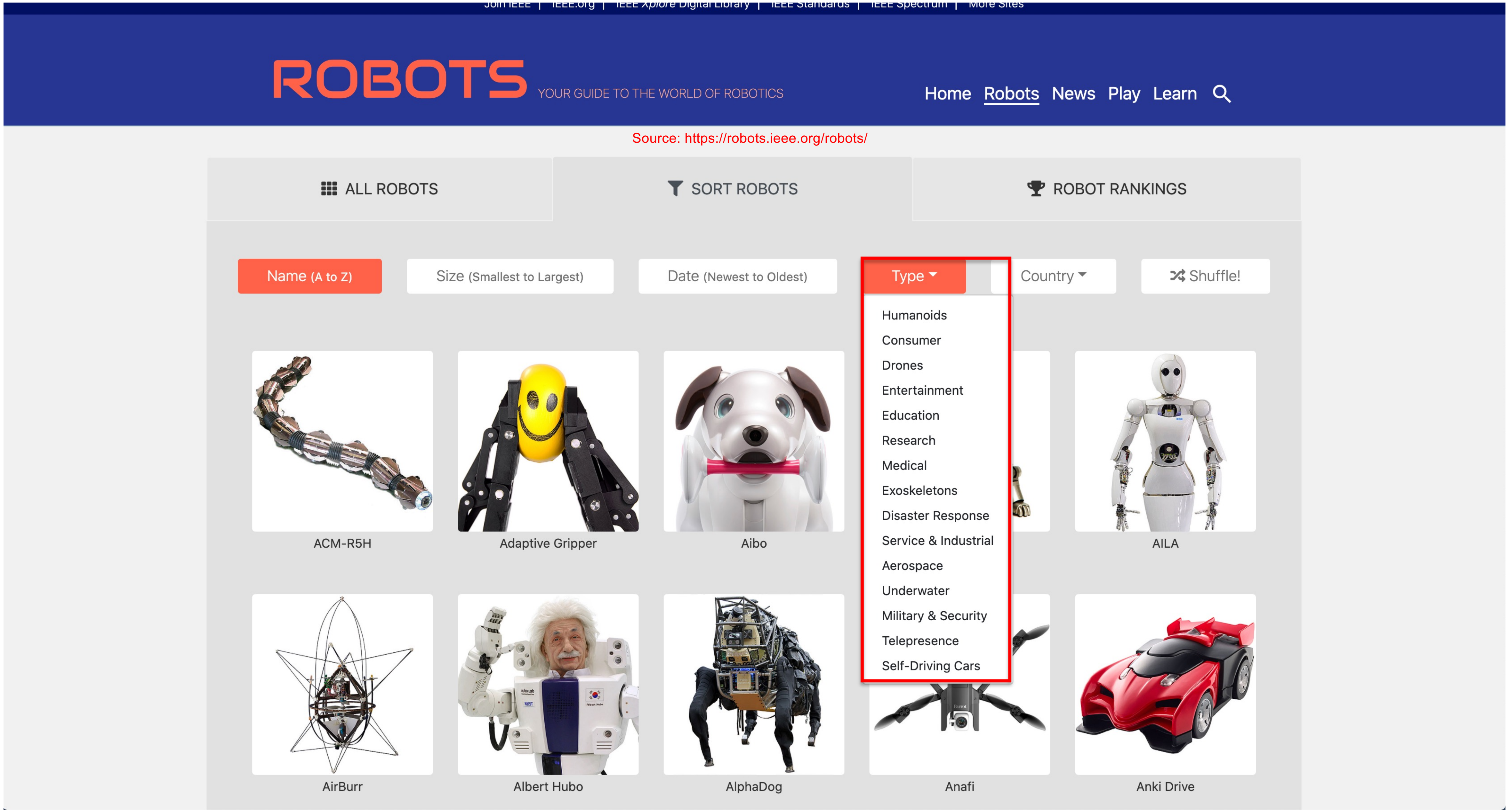
"A robot is an autonomous system	←	Not teleoperated (self-controlled & has controllers)
which exists in the physical world,	←	Subject to the physical laws (has a physical body)
can sense its environment,	←	Estimate the state of the world (uses sensors)
and can act on it	←	Physically affect the world (uses actuators & effectors)
to achieve some goals"	←	Purposeful, useful, possibly intelligent behaviour

M. Mataric, The Robotics Primer, MIT Press, 2007.

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

Principles



Source: https://robots.ieee.org/robots/

Types of Robot

Humanoids

Research

Armar

Armar is a robot created to be a helper in industrial environments. Its humanoid form lets it use human tools like power drills and hammers. Earlier versions were home helpers that could clean tables and load the dishwasher.

CREATOR
Karlsruhe Institute of Technology

COUNTRY
Germany 🇩🇪

YEAR
2017

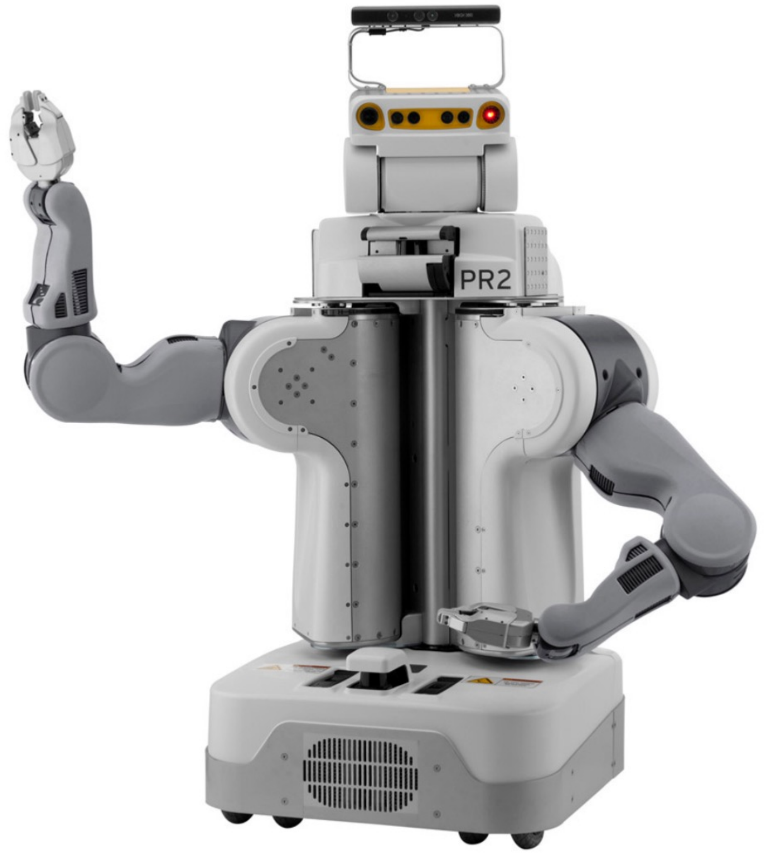
TYPE
Humanoids, Research

Source: https://robots.ieee.org/robots/armar/

Types of Robot

Humanoids

Research



PR2

The PR2 is one of the most advanced research robots ever built. Its powerful hardware and software systems let it do things like clean up tables, fold towels, and fetch you drinks from the fridge.

CREATOR
Willow Garage

COUNTRY
United States 🇺🇸

YEAR
2010

TYPE
Research, Humanoids

Source: https://robots.ieee.org/robots/pr2/

Types of Robot

Humanoids

Consumer

Entertainment

Pepper

Pepper is a friendly humanoid designed to be a companion in the home and help customers at retail stores. It talks, gesticulates, and seems determined to make everyone smile.

CREATOR
SoftBank Robotics
(originally created by Aldebaran Robotics, acquired by SoftBank in 2015)

COUNTRY
Japan 🇯🇵

YEAR
2014

TYPE
Humanoids, Consumer, Entertainment

Source: https://robots.ieee.org/robots/pepper/

Types of Robot

Humanoids

Research

Education

Nao

Nao is a small humanoid robot designed to interact with people. It's packed with sensors (and character) and it can walk, dance, speak, and recognize faces and objects. Now in its sixth generation, it is used in research, education, and healthcare all over the world.

CREATOR
SoftBank Robotics
(originally created by Aldebaran Robotics, acquired by SoftBank in 2015)

COUNTRY
France 🇫🇷

YEAR
2008

TYPE
Humanoids, Research, Education

Source: https://robots.ieee.org/robots/nao/

Types of Robot

Humanoids

Research

HRP-4

HRP-4 is one of the world's most advanced humanoids, the culmination of a decade of R&D. It's designed to collaborate with humans and can perform remarkably natural, human-like movements.

CREATOR
Kawada Industries and AIST

COUNTRY
Japan 🇯🇵

YEAR
2010

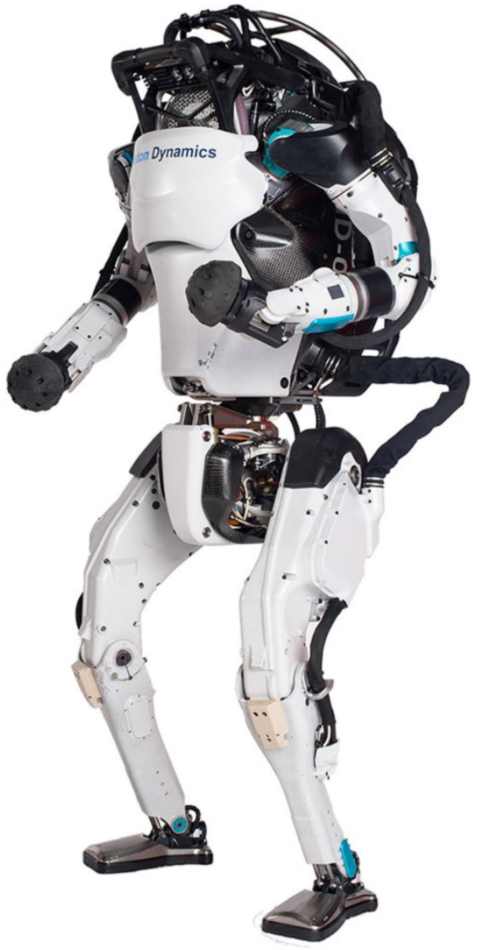
TYPE
Humanoids, Research

Source: https://robots.ieee.org/robots/hrp4/

Types of Robot

Humanoids

Industrial



Atlas

Atlas is the most agile humanoid in existence. It uses whole-body skills to move quickly and balance dynamically. It can lift and carry objects like boxes and crates, but its favorite tricks are running, jumping, and doing backflips.

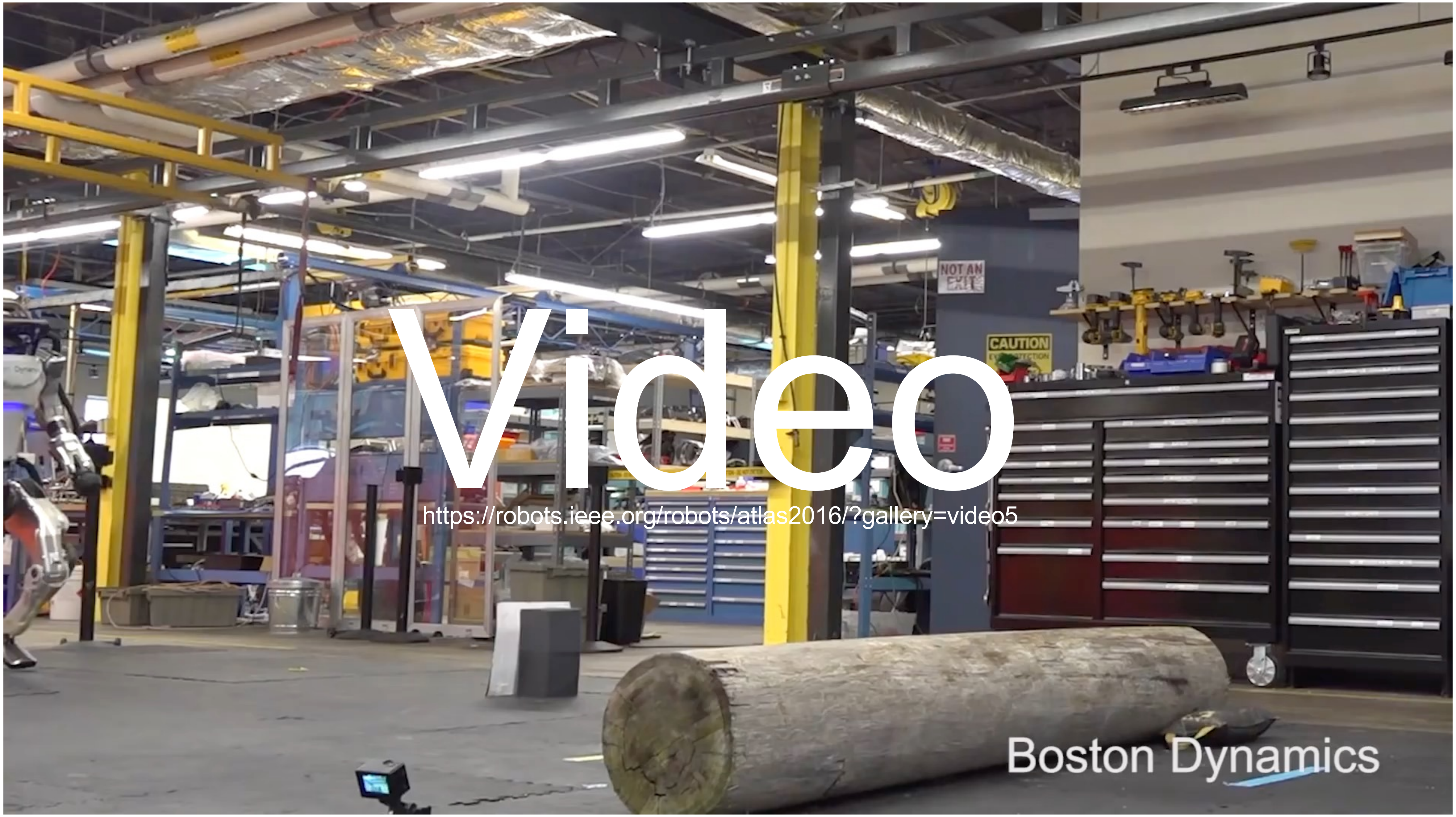
CREATOR
Boston Dynamics

COUNTRY
United States 🇺🇸

YEAR
2016

TYPE
Humanoids, Industrial

Source: https://robots.ieee.org/robots/atlas2016/



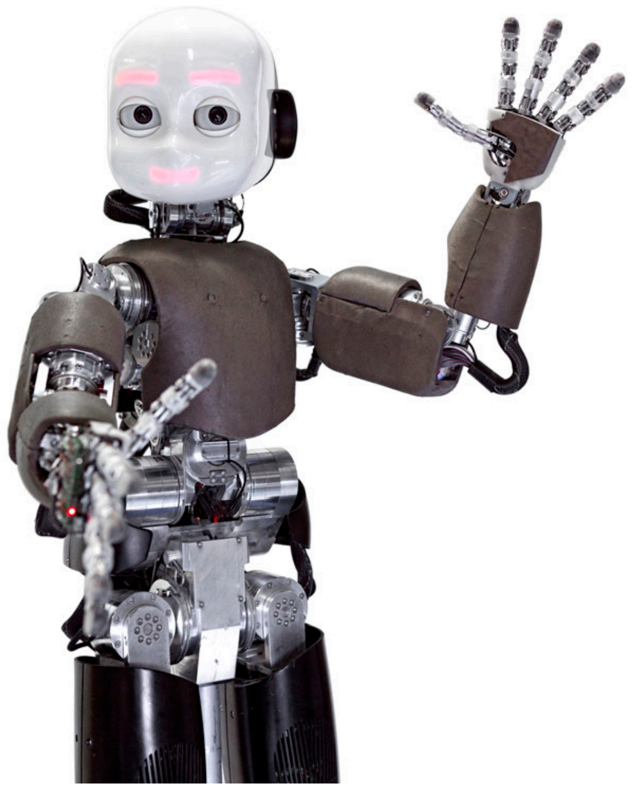
Video

https://robots.ieee.org/robots/atlas2016/?gallery=video5

Types of Robot

Humanoids

Research



iCub

iCub is a child-size humanoid robot capable of crawling, grasping objects, and interacting with people. It's designed as an open source platform for research in robotics, AI, and cognitive science.

CREATOR
RoboCub Consortium and IIT

COUNTRY
Italy 🇮🇹

YEAR
2004

TYPE
Humanoids, Research

Source: https://robots.ieee.org/robots/icub/

Video

https://robots.ieee.org/robots/icub/?gallery=video1

Types of Robot

Consumer

Roomba

Roomba is an autonomous vacuum and one of the most popular consumer robots in existence. It navigates around clutter and under furniture cleaning your floors, and returns to its charging dock when finished.

CREATOR
iRobot

COUNTRY
United States 🇺🇸

YEAR
2002

TYPE
Consumer

Source: https://robots.ieee.org/robots/roomba/

Video

https://robots.ieee.org/robots/roomba/?gallery=video2

Types of Robot

Education

Roomba

Roomba is an autonomous vacuum and one of the most popular consumer robots in existence. It navigates around clutter and under furniture cleaning your floors, and returns to its charging dock when finished.

CREATOR
iRobot

COUNTRY
United States 🇺🇸

YEAR
2002

TYPE
Consumer

Source: https://robots.ieee.org/robots/roomba/

Types of Robot

Consumer

Research

Education

TurtleBot

TurtleBot is a low-cost personal robot designed for hobbyists and researchers. It's open source, runs the ROS operating system, and combines a netbook with a Kinect 3D sensor and a mobile base.

CREATOR
Willow Garage

COUNTRY
United States 🇺🇸

YEAR
2011

TYPE
Consumer, Research, Education

Source: https://robots.ieee.org/robots/turtlebot/

Types of Robot

Drones

Military & Security

Global Hawk

The Global Hawk is an unmanned aerial vehicle that's used for high-altitude, long-duration surveillance. You tell it what to do, and it can take off, fly, spy, and return without any human input.

CREATOR
Northrop Grumman

COUNTRY
United States 🇺🇸

YEAR
2001

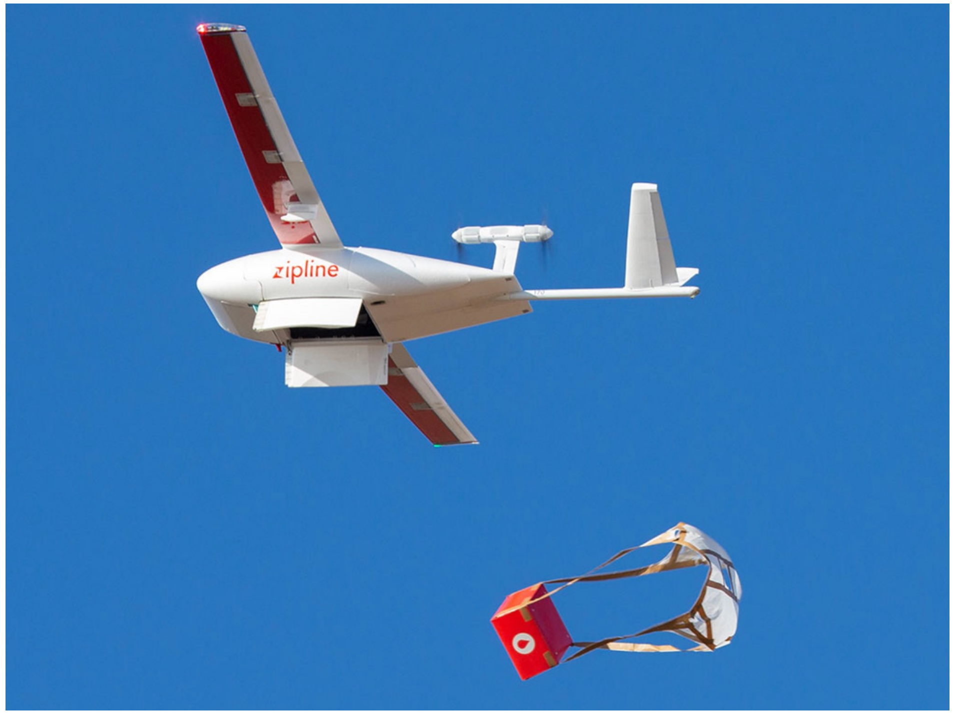
TYPE
Aerospace, Military & Security, Drones

Source: https://robots.ieee.org/robots/globalhawk/

Types of Robot

Drones

Medical



Zipline

Zipline is an autonomous fixed-wing aircraft drone used to carry blood and medicine from a distribution center to wherever it's needed. It can launch within minutes, and travel in any weather.

CREATOR
Zipline

COUNTRY
United States 🇺🇸

YEAR
2016

TYPE
Drones, Medical

Source: https://robots.ieee.org/robots/zipline/

Video

http://www.vernon.eu/videos/Zipline_hero.mp4

Video

https://www.youtube.com/watch?v=QWglZKVP26c

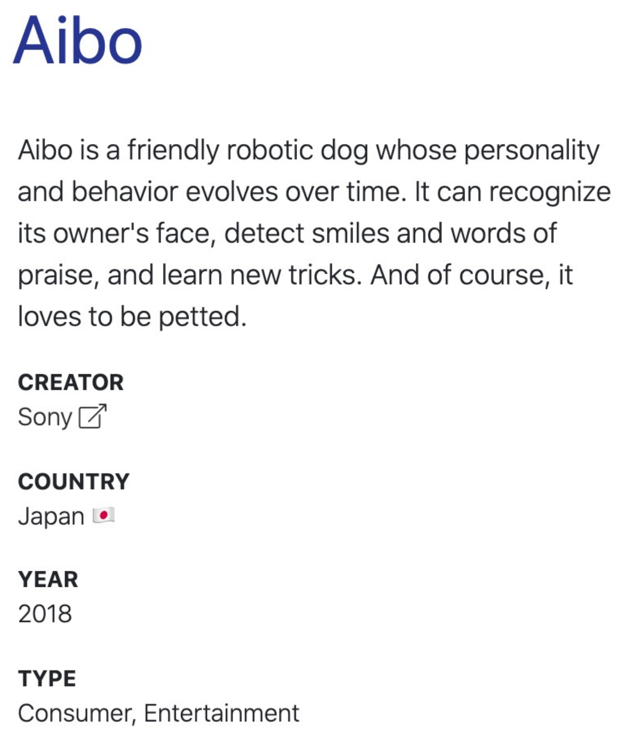
Video

http://www.vernon.eu/videos/Zipline_drop.mp4

Types of Robot

Entertainment

Consumer



Aibo

Aibo is a friendly robotic dog whose personality and behavior evolves over time. It can recognize its owner's face, detect smiles and words of praise, and learn new tricks. And of course, it loves to be petted.

CREATOR
Sony

COUNTRY
Japan 🇯🇵

YEAR
2018

TYPE
Consumer, Entertainment

Source: https://robots.ieee.org/robots/aibo2018/

Video

https://www.youtube.com/watch?v=5ifwGc-0mAY

Types of Robot

Industrial

Picker Robots

Picker Robots are mobile machines designed to autonomously retrieve and carry products in a warehouse. The robots are directed through AI-powered software that identifies the most efficient paths for them to pick, replenish, return, and count goods.

CREATOR
inVia Robotics

COUNTRY
United States 🇺🇸

YEAR
2015

TYPE
Industrial

Source: https://robots.ieee.org/robots/invia/

Video

https://robots.ieee.org/robots/invia/?gallery=video5

Types of Robot

Industrial

Freight

Freight is an autonomous mobile base for use in warehouses to transport materials from point A to point B. The robot platforms come in three zippy flavors – 100, 500 and 1500, all of which represent the payload it can handle in kilograms.

CREATOR
Fetch Robotics

COUNTRY
United States 🇺🇸

YEAR
2014

TYPE
Industrial

Source: https://robots.ieee.org/robots/freight/

Types of Robot

Industrial

Sawyer

Sawyer is an industrial collaborative robot designed to help out with manufacturing tasks and work alongside humans. You can teach it new tasks by demonstrating what to do using the robot's own arm.

CREATOR
Rethink Robotics

COUNTRY
United States 🇺🇸

YEAR
2015

TYPE
Industrial

Source: https://robots.ieee.org/robots/sawyer/

Video

https://robots.ieee.org/robots/sawyer/?gallery=video1

Types of Robot

Industrial

Meca500

Meca500 is the world's smallest, most compact six-axis industrial robot arm. It's also one of the most precise. And with an embedded controller it can easily be transported and set up in confined spaces.

CREATOR
Mecademic

COUNTRY
Canada 🇨🇦

YEAR
2015

TYPE
Industrial

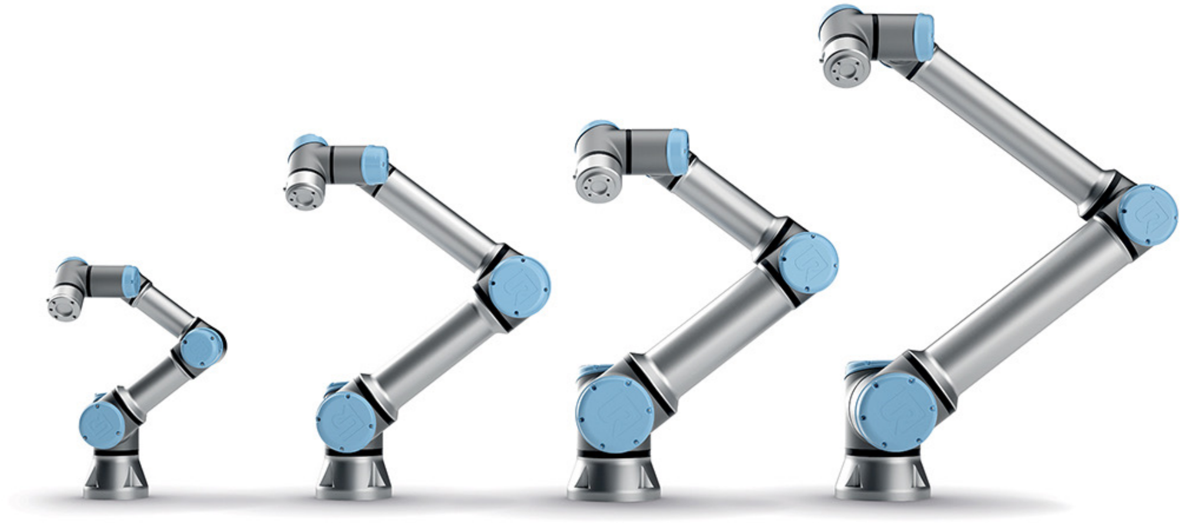
Source: https://robots.ieee.org/robots/meca/

Video

https://robots.ieee.org/robots/meca500/?gallery=video1

Types of Robot

Industrial



UR

Universal Robots cobots are versatile, lightweight collaborative robotic arms designed to work safely alongside humans. Users program it through an intuitive touch-screen interface and by positioning the robot with their hands.

CREATOR
Universal Robots

COUNTRY
Denmark 🇩🇰

YEAR
2008

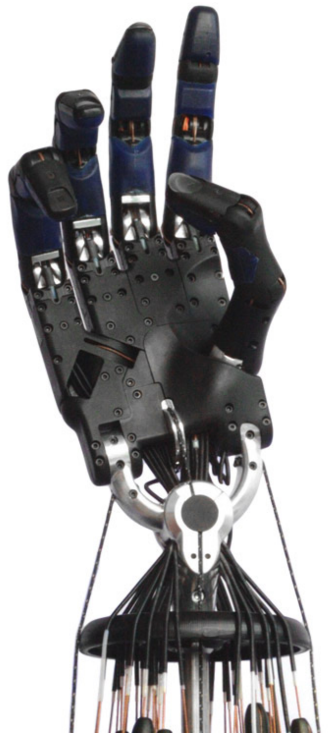
TYPE
Industrial

Source: https://robots.ieee.org/robots/ur/

Types of Robot

Research

Industrial



Shadow Hand

The Shadow Dexterous Hand is one of the most advanced robot hands in the world. It's designed to replicate as much of the functionality, dimensions, and range of motion of the human hand as possible.

CREATOR
Shadow Robot Company

COUNTRY
United Kingdom 🇬🇧

YEAR
2004

TYPE
Industrial, Telepresence, Research

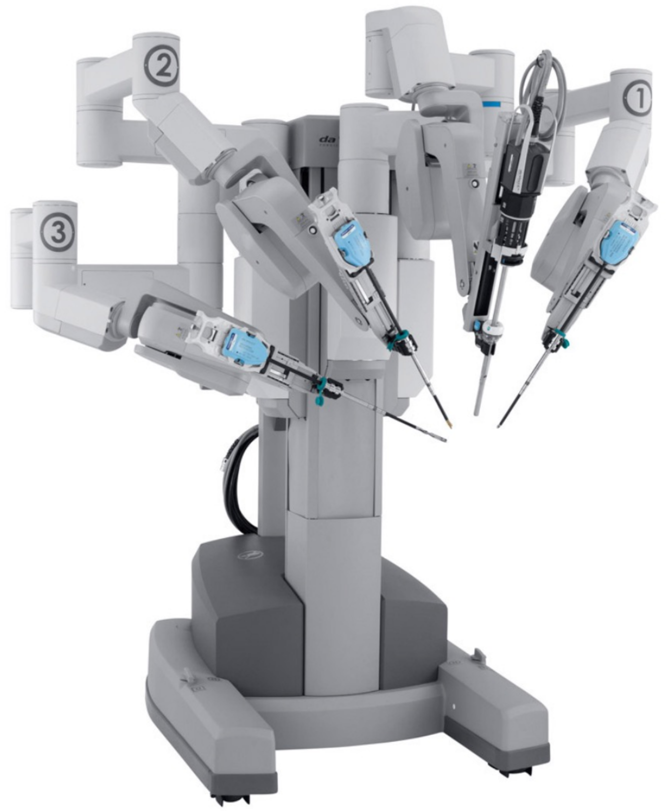
Source: https://robots.ieee.org/robots/shadow/

Video

https://robots.ieee.org/robots/shadow/?gallery=video4

Types of Robot

Medical



Da Vinci

The da Vinci is a surgical robot designed for minimally invasive procedures. It has four arms equipped with surgical instruments and cameras that a physician controls remotely from a console.

CREATOR
Intuitive Surgical

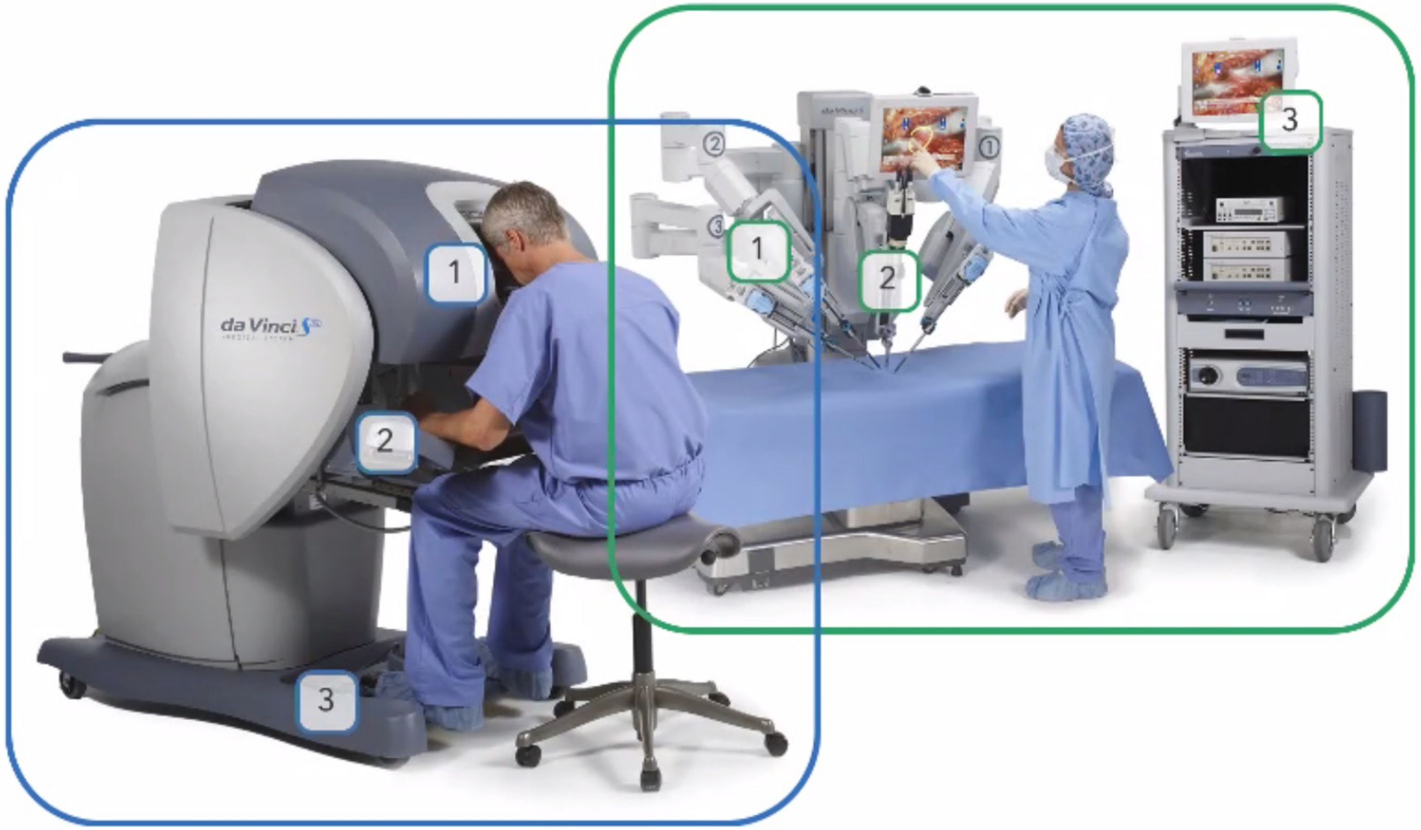
COUNTRY
United States 🇺🇸

YEAR
1999

TYPE
Medical

Source: https://robots.ieee.org/robots/davinci/

THE DA VINCI SURGICAL SYSTEM



SURGEON SIDE

1	High Resolution Stereo Viewers (HRSVs)
2	Master Tool Manipulators (MTMs)
3	Foot pedal tray

PATIENT SIDE

1	Patient Side Manipulators (PSMs)
2	Endoscopic Camera Manipulator (ECM)
3	Vision Cart

Patient Side Manipulators: robotic arms teleoperated by the Master Tool Manipulators, they mount the surgical tools.

Endoscopic Camera Manipulator: robotic arm that is also teleoperated by the Master Tool Manipulators, it holds the endoscope.

Video

https://www.youtube.com/watch?v=961E6Nx9Pok

Types of Robot

Consumer

Telepresence

Beam

Beam is a telepresence robotic system that can "teleport" you to a remote location, allowing you to move around and interact with people. It is easy to drive and has a large display to improve face-to-face, or screen-to-face, communication.

CREATOR
Suitable Technologies

COUNTRY
United States 🇺🇸

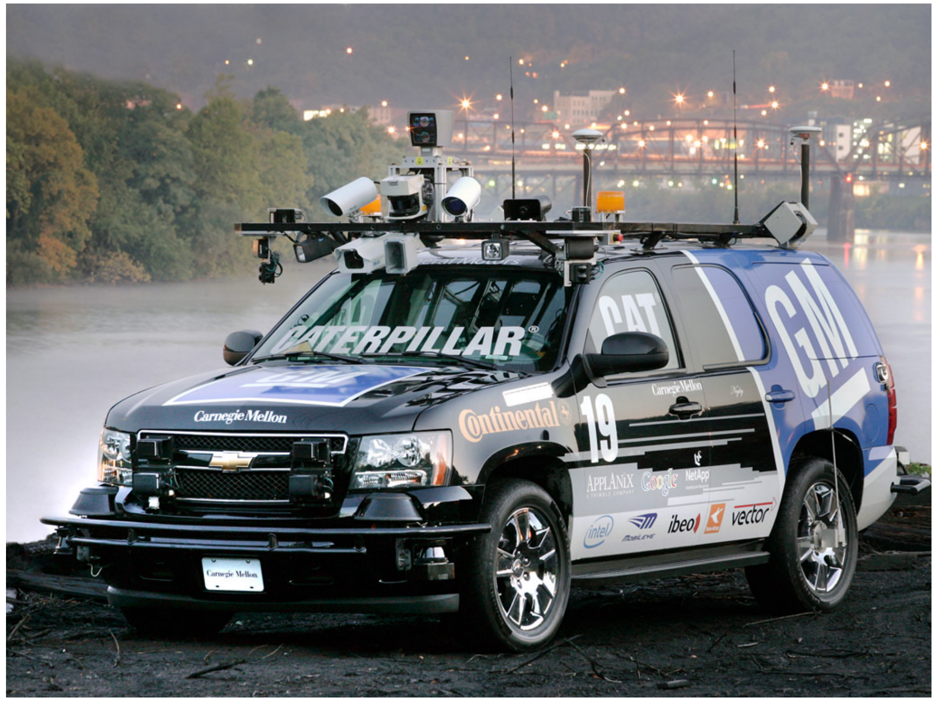
YEAR
2011

TYPE
Telepresence, Consumer

Source: https://robots.ieee.org/robots/beam/

Types of Robot

Autonomous Vehicle Research



Boss

Boss is the world's smartest Chevy Tahoe. In 2007, it won the DARPA Urban Challenge for autonomous vehicles, taking home a $2 million prize for not breaking any traffic laws or running anyone over.

CREATOR
Carnegie Mellon University

COUNTRY
United States 🇺🇸

YEAR
2007

TYPE
Autonomous Vehicle, Research

Source: https://robots.ieee.org/robots/boss/

Types of Robot

Autonomous Vehicle Research

Google Self-Driving Car

Google's self-driving car is a modified Toyota Prius that can autonomously drive in city traffic and on highways. The goal is developing technology to reduce traffic accidents and increase road efficiency.

CREATOR
Google

COUNTRY
United States 🇺🇸

YEAR
2010

TYPE
Autonomous Vehicle, Research

Source: https://robots.ieee.org/robots/beam/

Types of Robot

Industrial

Research

Disaster Response

ANYmal

ANYmal is a rugged, autonomous four-legged robot designed for inspection and manipulation tasks. It uses sensors to scan the terrain and avoid obstacles, and can operate in rain, snow, wind, waterlogged rooms, and dusty environments.

CREATOR
ETH Zurich and ANYbotics

COUNTRY
Switzerland 🇨🇭

YEAR
2016

TYPE
Industrial, Research, Disaster Response

Source: https://robots.ieee.org/robots/anymal/

Types of Robot

Industrial

Research

Spot

Spot is a compact, nimble four-legged robot that can trot around your office, home, or outdoors. It can map its environment, sense and avoid obstacles, climb stairs, and open doors. It can also fetch you a drink.

CREATOR
Boston Dynamics

COUNTRY
United States 🇺🇸

YEAR
2016

TYPE
Industrial, Research

Source: https://robots.ieee.org/robots/spotmini/

Video

https://robots.ieee.org/robots/spotmini/?gallery=video1

Types of Robot

Military & Security

Research

AlphaDog

AlphaDog is a quadruped robot the size of a mule (a big, mean mule). It's powered by a hydraulic actuation system and is designed to assist soldiers in carrying heavy gear over rough terrain.

CREATOR
Boston Dynamics

COUNTRY
United States 🇺🇸

YEAR
2011

TYPE
Military & Security, Research

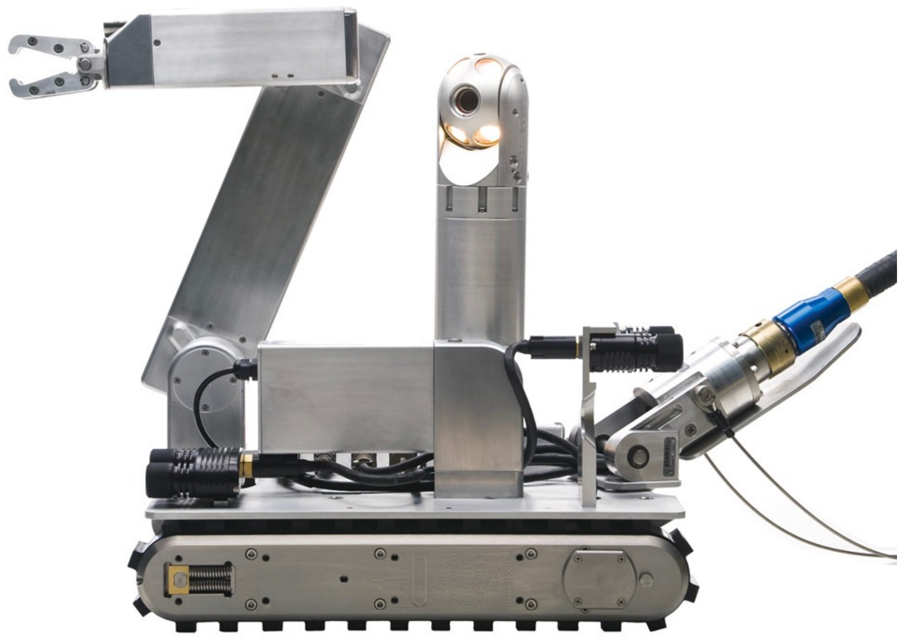
Source: https://robots.ieee.org/robots/alphadog/

Types of Robot

Industrial

Military & Security

Disaster Response



Versatrax

Versatrax 450 TTC is a mobile robot designed for hazardous environments. It allows users to locate, inspect, and safely remove dangerous materials from any site faster than by conventional means.

CREATOR
Inuktun Services

COUNTRY
Canada 🇨🇦

YEAR
2012

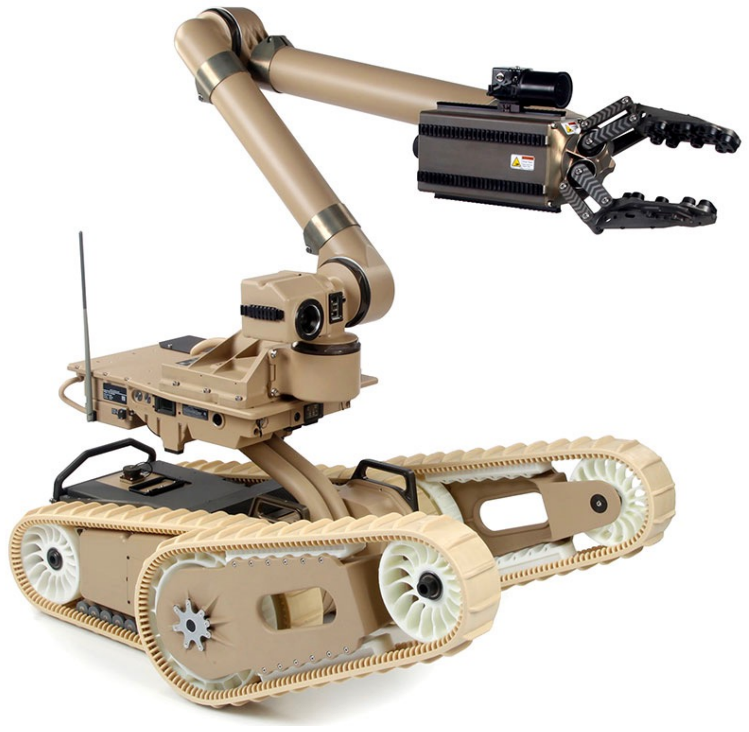
TYPE
Industrial, Military & Security, Disaster Response

Source: https://robots.ieee.org/robots/inuktun/

Types of Robot

Military & Security

Disaster Response



Kobra

Kobra is a rugged, remote control robot designed to search for explosives and carry out reconnaissance missions. It rolls on tank-like treads, and its manipulator arm can lift heavy payloads.

CREATOR
Endeavor Robotics
(Originally created by iRobot)

COUNTRY
United States 🇺🇸

YEAR
2011

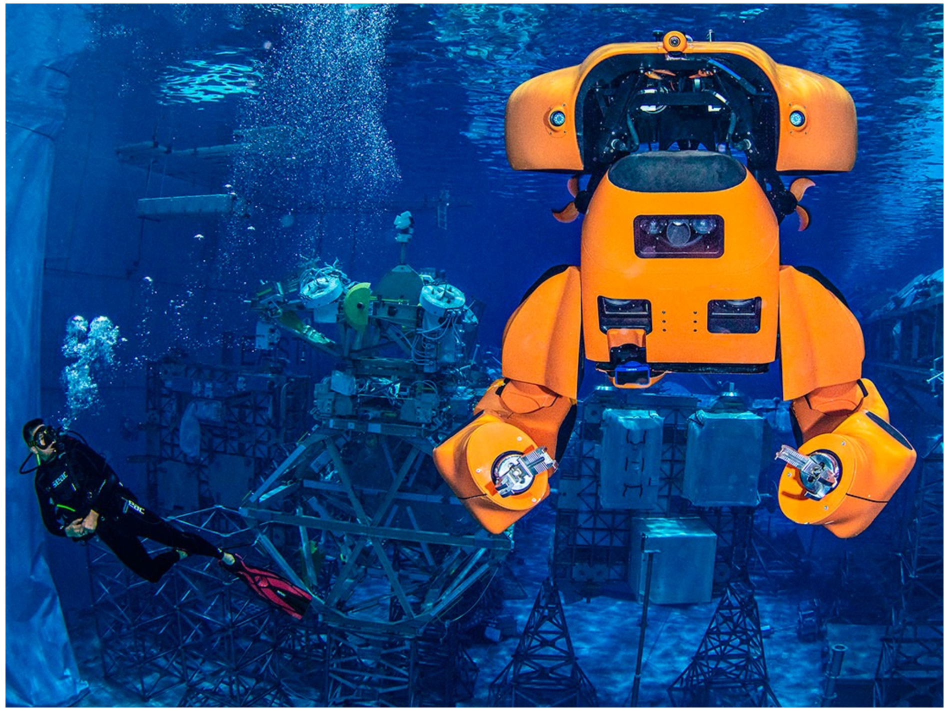
TYPE
Military & Security, Disaster Response

Source: https://robots.ieee.org/robots/kobra/

Types of Robot

Underwater

Industrial



Aquanaut

Aquanaut is an unmanned underwater vehicle that can transform itself from a nimble submarine designed for long-distance cruising into a half-humanoid robot capable of carrying out complex manipulation tasks. It can inspect subsea oil and gas infrastructure, operate valves, and use tools.

CREATOR
Houston Mechatronics Inc.

COUNTRY
United States 🇺🇸

YEAR
2019

TYPE
Underwater, Industrial

Source: https://robots.ieee.org/robots/aquanaut/

Types of Robot

Research

Salamandra robotica II

Salamandra robotica II is an amphibious robot inspired by the salamander's anatomy and nervous system. It's used to study robot locomotion and test neurobiological models in real environments.

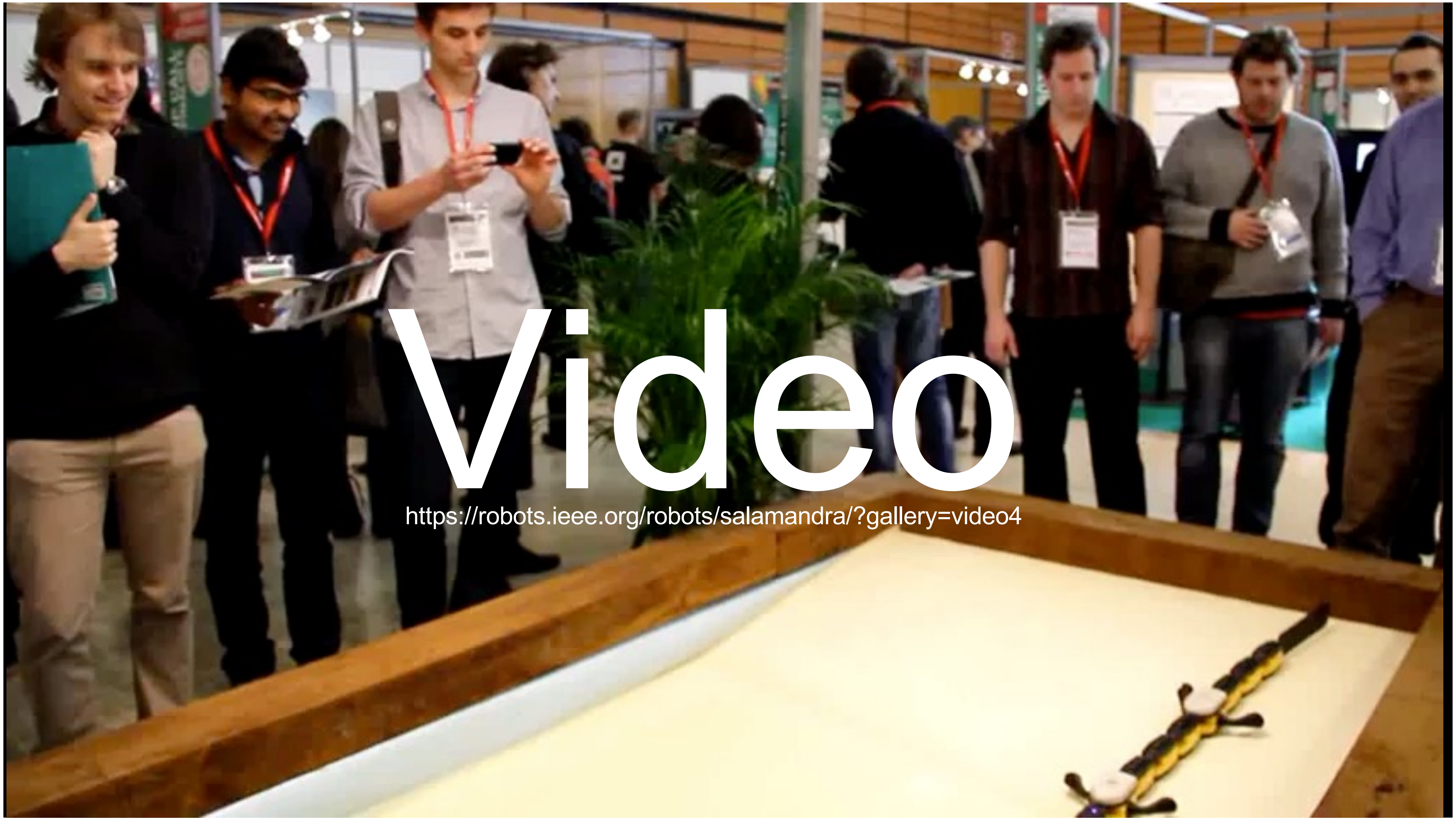
CREATOR
Biorobotics Laboratory at EPFL

COUNTRY
Switzerland

YEAR
2012

TYPE
Research

Source: https://robots.ieee.org/robots/salamandra/

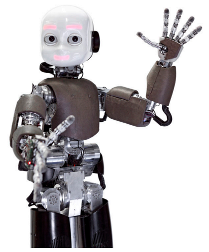


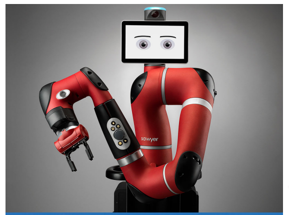
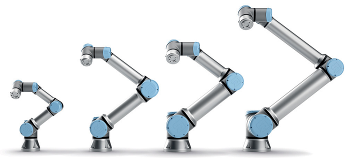
Video

https://robots.ieee.org/robots/salamandra/?gallery=video4

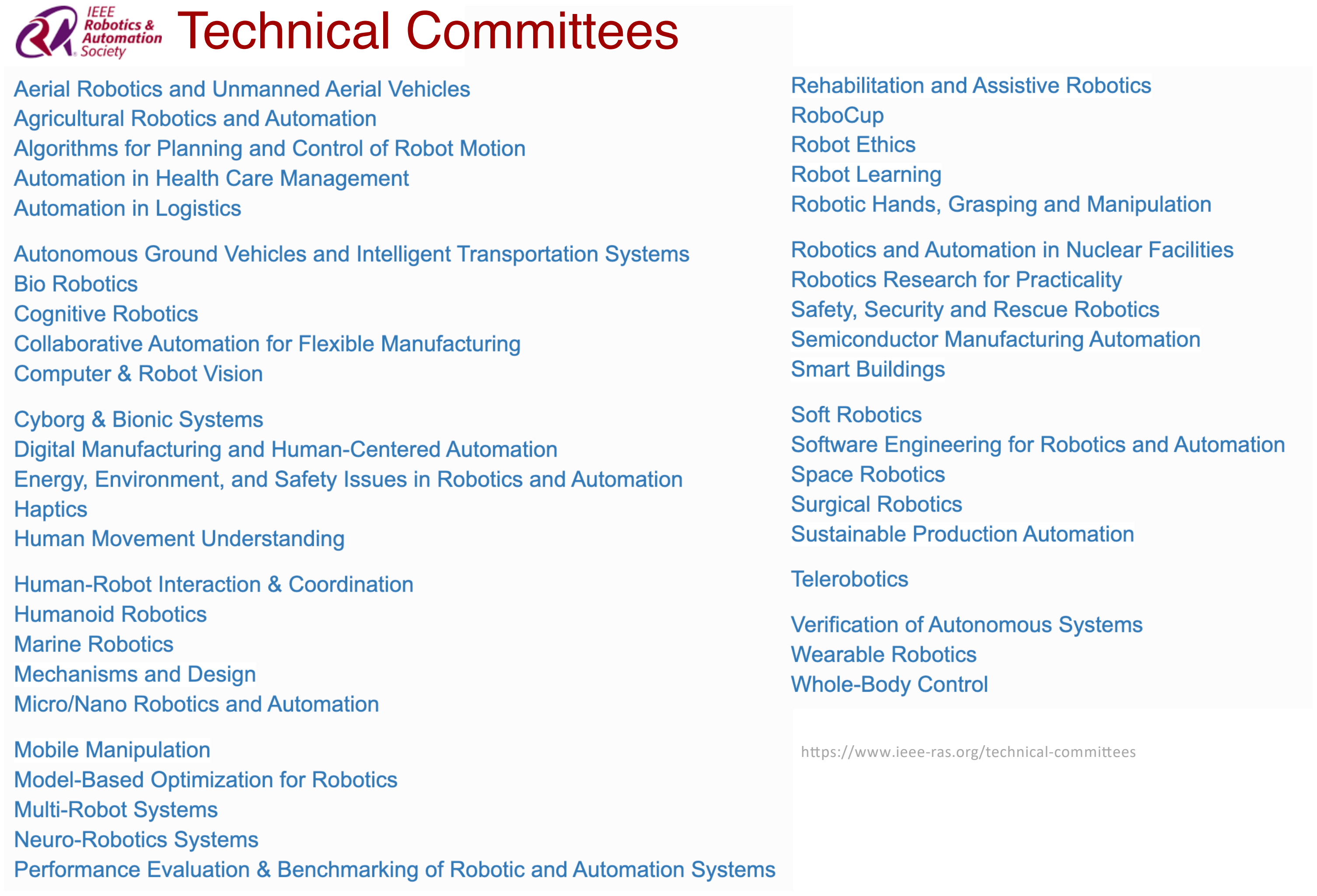
Physical Embodiment

- Humanoid vs non-humanoid
- Manipulator arms
- Mobile robots
- Mobile manipulators



The Many Areas of Robotics



Technical Committees

- Aerial Robotics and Unmanned Aerial Vehicles
- Agricultural Robotics and Automation
- Algorithms for Planning and Control of Robot Motion
- Automation in Health Care Management
- Automation in Logistics
- Autonomous Ground Vehicles and Intelligent Transportation Systems
- Bio Robotics
- Cognitive Robotics
- Collaborative Automation for Flexible Manufacturing
- Computer & Robot Vision
- Cyborg & Bionic Systems
- Digital Manufacturing and Human-Centered Automation
- Energy, Environment, and Safety Issues in Robotics and Automation
- Haptics
- Human Movement Understanding
- Human-Robot Interaction & Coordination
- Humanoid Robotics
- Marine Robotics
- Mechanisms and Design
- Micro/Nano Robotics and Automation
- Mobile Manipulation
- Model-Based Optimization for Robotics
- Multi-Robot Systems
- Neuro-Robotics Systems
- Performance Evaluation & Benchmarking of Robotic and Automation Systems
- Rehabilitation and Assistive Robotics
- RoboCup
- Robot Ethics
- Robot Learning
- Robotic Hands, Grasping and Manipulation
- Robotics and Automation in Nuclear Facilities
- Robotics Research for Practicality
- Safety, Security and Rescue Robotics
- Semiconductor Manufacturing Automation
- Smart Buildings
- Soft Robotics
- Software Engineering for Robotics and Automation
- Space Robotics
- Surgical Robotics
- Sustainable Production Automation
- Telerobotics
- Verification of Autonomous Systems
- Wearable Robotics
- Whole-Body Control

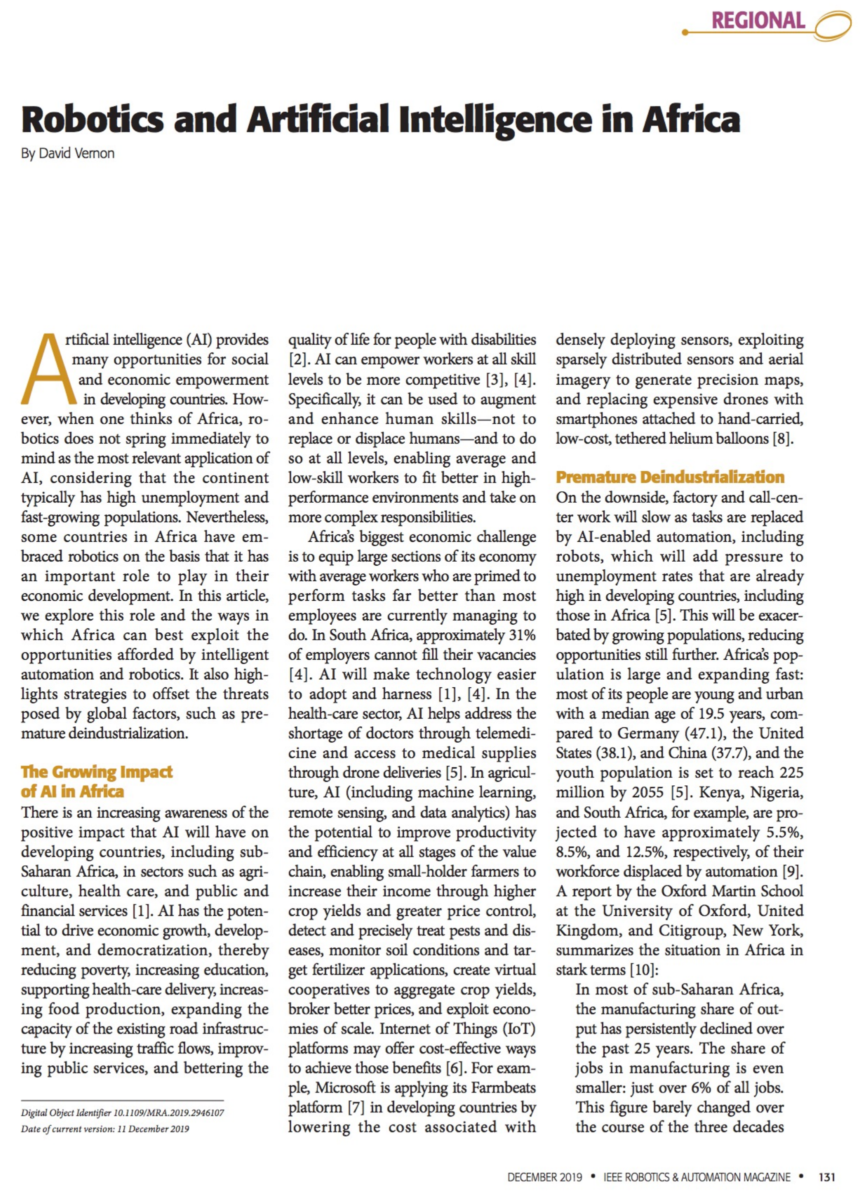
https://www.ieee-ras.org/technical-committees

Reading

D. Vernon, "Robotics and Artificial Intelligence in Africa", IEEE Robotics & Automation Magazine, Vol. 26, No. 4, pp. 131-135, December 2019.

http://vernon.eu/publications/19_Vernon_RAM.pdf

M. Mataric, The Robotics Primer, MIT Press, 2007. Chapter 1.


REGIONAL

Robotics and Artificial Intelligence in Africa

By David Vernon

Artificial intelligence (AI) provides many opportunities for social and economic empowerment in developing countries. However, when one thinks of Africa, robotics does not spring immediately to mind as the most relevant application of AI, considering that the continent typically has high unemployment and fast-growing populations. Nevertheless, some countries in Africa have embraced robotics on the basis that it has an important role to play in their economic development. In this article, we explore this role and the ways in which Africa can best exploit the opportunities afforded by intelligent automation and robotics. It also highlights strategies to offset the threats posed by global factors, such as premature deindustrialization.

The Growing Impact of AI in Africa

There is an increasing awareness of the positive impact that AI will have on developing countries, including sub-Saharan Africa, in sectors such as agriculture, health care, and public and financial services [1]. AI has the potential to drive economic growth, development, and democratization, thereby reducing poverty, increasing education, supporting health-care delivery, increasing food production, expanding the capacity of the existing road infrastructure by increasing traffic flows, improving public services, and bettering the quality of life for people with disabilities [2]. AI can empower workers at all skill levels to be more competitive [3], [4]. Specifically, it can be used to augment and enhance human skills—not to replace or displace humans—and to do so at all levels, enabling average and low-skill workers to fit better in high-performance environments and take on more complex responsibilities.

Africa's biggest economic challenge is to equip large sections of its economy with average workers who are primed to perform tasks far better than most employees are currently managing to do. In South Africa, approximately 31% of employers cannot fill their vacancies [4]. AI will make technology easier to adopt and harness [1], [4]. In the health-care sector, AI helps address the shortage of doctors through telemedicine and access to medical supplies through drone deliveries [5]. In agriculture, AI (including machine learning, remote sensing, and data analytics) has the potential to improve productivity and efficiency at all stages of the value chain, enabling small-holder farmers to increase their income through higher crop yields and greater price control, detect and precisely treat pests and diseases, monitor soil conditions and target fertilizer applications, create virtual cooperatives to aggregate crop yields, broker better prices, and exploit economies of scale. Internet of Things (IoT) platforms may offer cost-effective ways to achieve those benefits [6]. For example, Microsoft is applying its Farmbeats platform [7] in developing countries by lowering the cost associated with densely deploying sensors, exploiting sparsely distributed sensors and aerial imagery to generate precision maps, and replacing expensive drones with smartphones attached to hand-carried, low-cost, tethered helium balloons [8].

Premature Deindustrialization

On the downside, factory and call-center work will slow as tasks are replaced by AI-enabled automation, including robots, which will add pressure to unemployment rates that are already high in developing countries, including those in Africa [5]. This will be exacerbated by growing populations, reducing opportunities still further. Africa's population is large and expanding fast: most of its people are young and urban with a median age of 19.5 years, compared to Germany (47.1), the United States (38.1), and China (37.7), and the youth population is set to reach 225 million by 2055 [5]. Kenya, Nigeria, and South Africa, for example, are projected to have approximately 5.5%, 8.5%, and 12.5%, respectively, of their workforce displaced by automation [9]. A report by the Oxford Martin School at the University of Oxford, United Kingdom, and Citigroup, New York, summarizes the situation in Africa in stark terms [10]:

> In most of sub-Saharan Africa, the manufacturing share of output has persistently declined over the past 25 years. The share of jobs in manufacturing is even smaller: just over 6% of all jobs. This figure barely changed over the course of the three decades

Digital Object Identifier 10.1109/MRA.2019.2946107
Date of current version: 11 December 2019

https://ieeexplore.ieee.org/document/8931075
http://vernon.eu/publications/19_Vernon_RAM.pdf

Videos

Atlas (0:30):	https://robots.ieee.org/robots/atlas2016/?gallery=video5
iCub (2:40):	https://robots.ieee.org/robots/icub/?gallery=video1
Roomba (1:30):	https://robots.ieee.org/robots/roomba/?gallery=video2
Turtlebot (1:30):	https://robots.ieee.org/robots/turtlebot/?gallery=video1
Zipline (0:06):	http://www.vernon.eu/videos/Zipline_hero.mp4
Zipline (1:09):	https://www.youtube.com/watch?v=QWglZKVP26c
Zipline (0:15):	http://www.vernon.eu/videos/Zipline_drop.mp4
Zipline (11:44):	https://www.youtube.com/watch?v=jEbRVNxL44c
Picker Robots (0:15):	https://robots.ieee.org/robots/invia/?gallery=video5
Sawyer (0:30):	https://robots.ieee.org/robots/sawyer/?gallery=video1
Meca (1:15):	https://robots.ieee.org/robots/meca500/?gallery=video1
Shadow Hand (3:00):	https://robots.ieee.org/robots/shadow/?gallery=video4
Spot (2:00):	https://robots.ieee.org/robots/spotmini/?gallery=video1
Salamandra (0:43):	https://robots.ieee.org/robots/salamandra/?gallery=video4

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

$\Bigg\}$ Principles

Robot Components

- Sensors — To perceive the environment
- Actuators
- Effectors

  } To take action

- Controllers — For autonomy

Sensors

- Differentiate between
  - proprioceptive sensors that sense the state of the robot (proprioception)
    - Internal state, as the robot perceives it
  - exteroceptive sensors that sense the state of the environment (exteroception)
    - External state, as the robot perceives it
- The set of all possible states is referred to as the state space (discrete or continuous)

Sensors

Internal state can be used to remember information about the environment

- Representation
- Also known as internal model

Sensors

Different modalities

- Visual
- Auditory
- Olfactory (smell)
- Tactile (touch)
- Proximity (distance)

Sensors

- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors

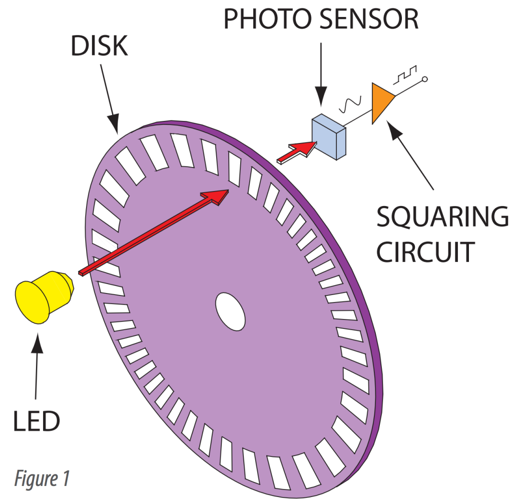
$\}$ Proprioceptive sensors

- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors

$\}$ Exteroceptive sensors

Sensors

- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors

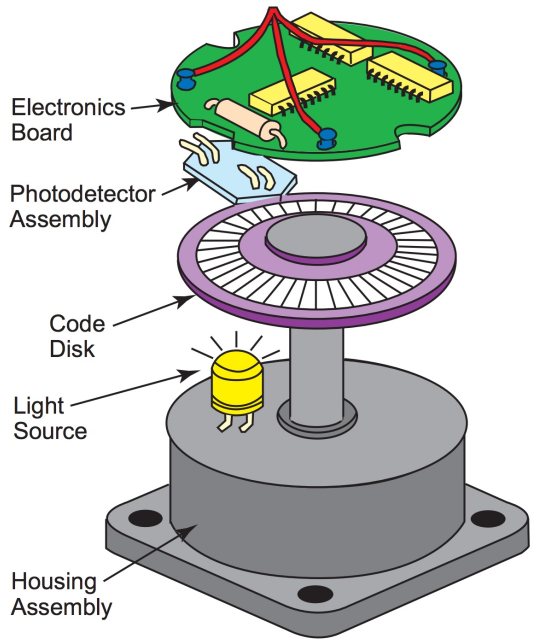


Source: http://encoder.com/core/files/encoder/uploads/files/WP-2011.pdf

Go to http://encoder.com/videos/
to watch a video explaining the operation of encoders

Sensors

- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors



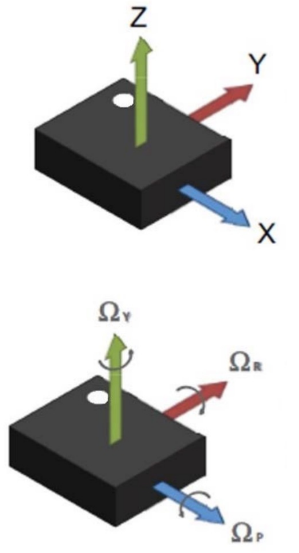
Source: http://encoder.com/core/files/encoder/uploads/files/WP-2011.pdf

Go to http://encoder.com/videos/
to watch a video explaining the operation of encoders

Sensors

- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors

Accelerometers sense change in position
Gyroscopes sense change in orientation

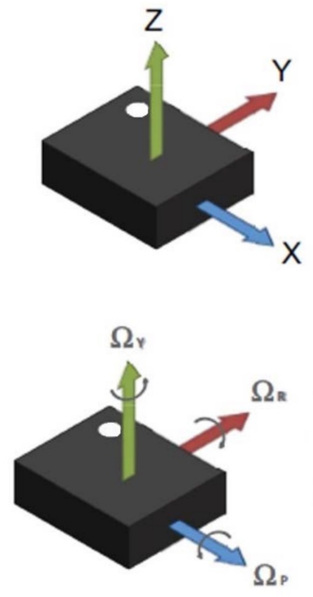


Source: https://www.st.com/resource/en/datasheet/asm330lhh.pdf

Sensors

Inertial Measurement Unit IMU

- Combines three accelerometers and three gyroscopes
- In three orthogonal (x, y, z) directions
- To sense change in position and orientation



Source: https://www.st.com/resource/en/datasheet/asm330lhh.pdf

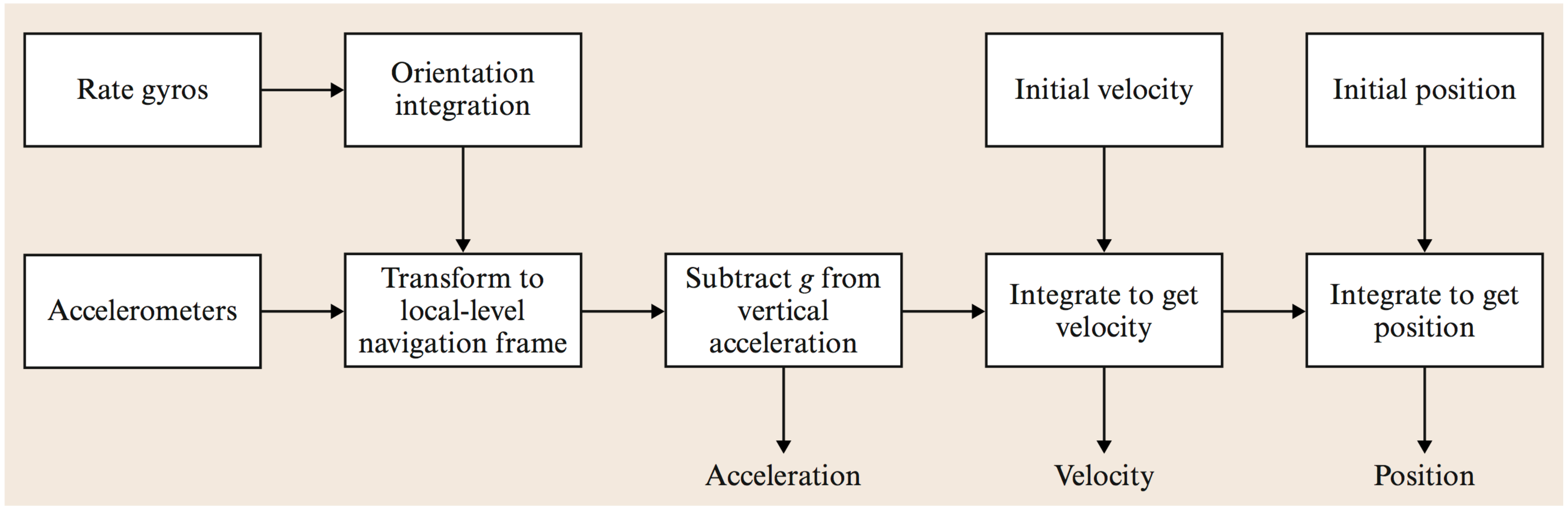
Sensors

Options for detecting change in relative position:

- Accelerometers sense acceleration … we want change in position
- Gyroscopes sense rate of change of orientation … we want change in orientation
- We get what we want by integrating the sensed data with respect to time

Sensors

Block diagram for estimating position with an IMU


Rate gyros
Orientation integration
Initial velocity
Initial position
Accelerometers
Transform to local-level navigation frame
Subtract *g* from vertical acceleration
Integrate to get velocity
Integrate to get position
Acceleration
Velocity
Position

Source: B. Siciliano and O. Khatib (eds.), Springer Handbook of Robotics, Springer, 2008.

Sensors

Double integration of acceleration to determine position

Ideally, the position x of a body at any time t can be determined from the time-dependent acceleration of that body

$$x(t) = \int_0^t \int_0^t a(t)dtdt + \int_0^t v_0 dt + x_0$$

Position

Acceleration of the object: measured by the accelerometer

Initial velocity of the object

Initial position of the object

Source: https://d10bqar0tuhard.cloudfront.net/en/document/AN013-Position-determination-using-Accelerometers.pdf

Sensors

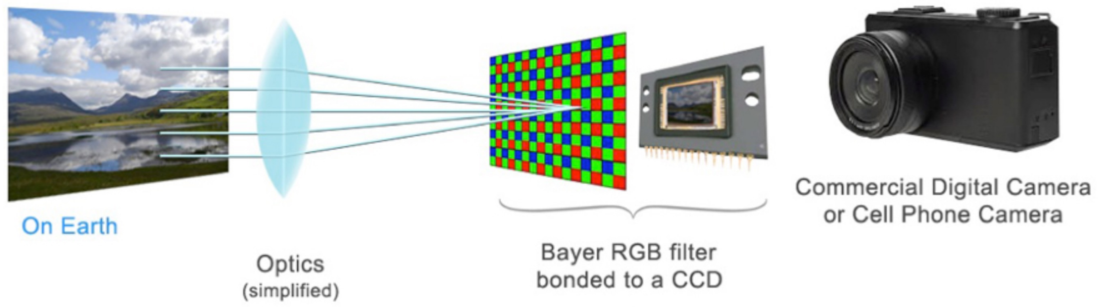
- Precise estimate if
  - initial estimate of v_0 and s_0 are precise
  - measurement of $a(t)$ is precise

For more information on how to minimize errors with a MEMS accelerometer, see the technical note here: https://d10bqar0tuhard.cloudfront.net/en/document/AN013-Position-determination-using-Accelerometers.pdf

- However, sensors are not perfect: errors arise
- Errors accumulate without bounds
  - Double integration means that the errors grow quadratically
  - Need to reset the position from time to time, e.g., using absolute position estimation

Sensors

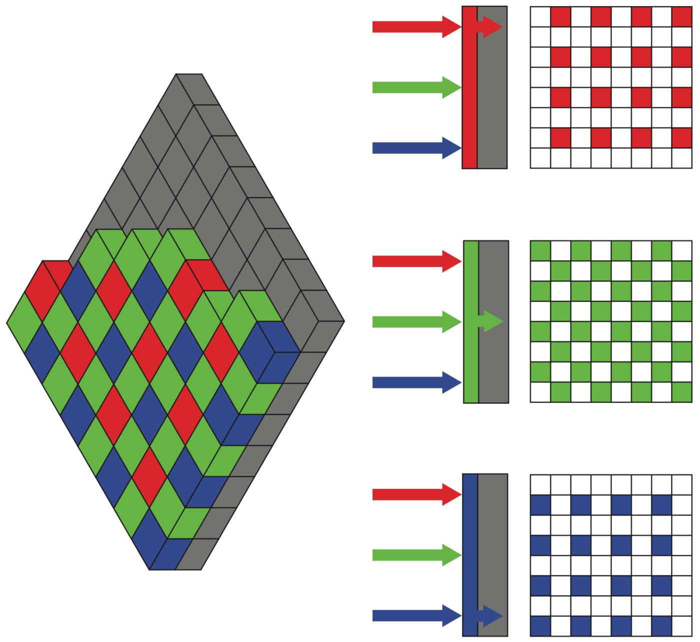
- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors



Chttps://www.nasa.gov/mission_pages/msl/multimedia/pia16799.html

Sensors

- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors



C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

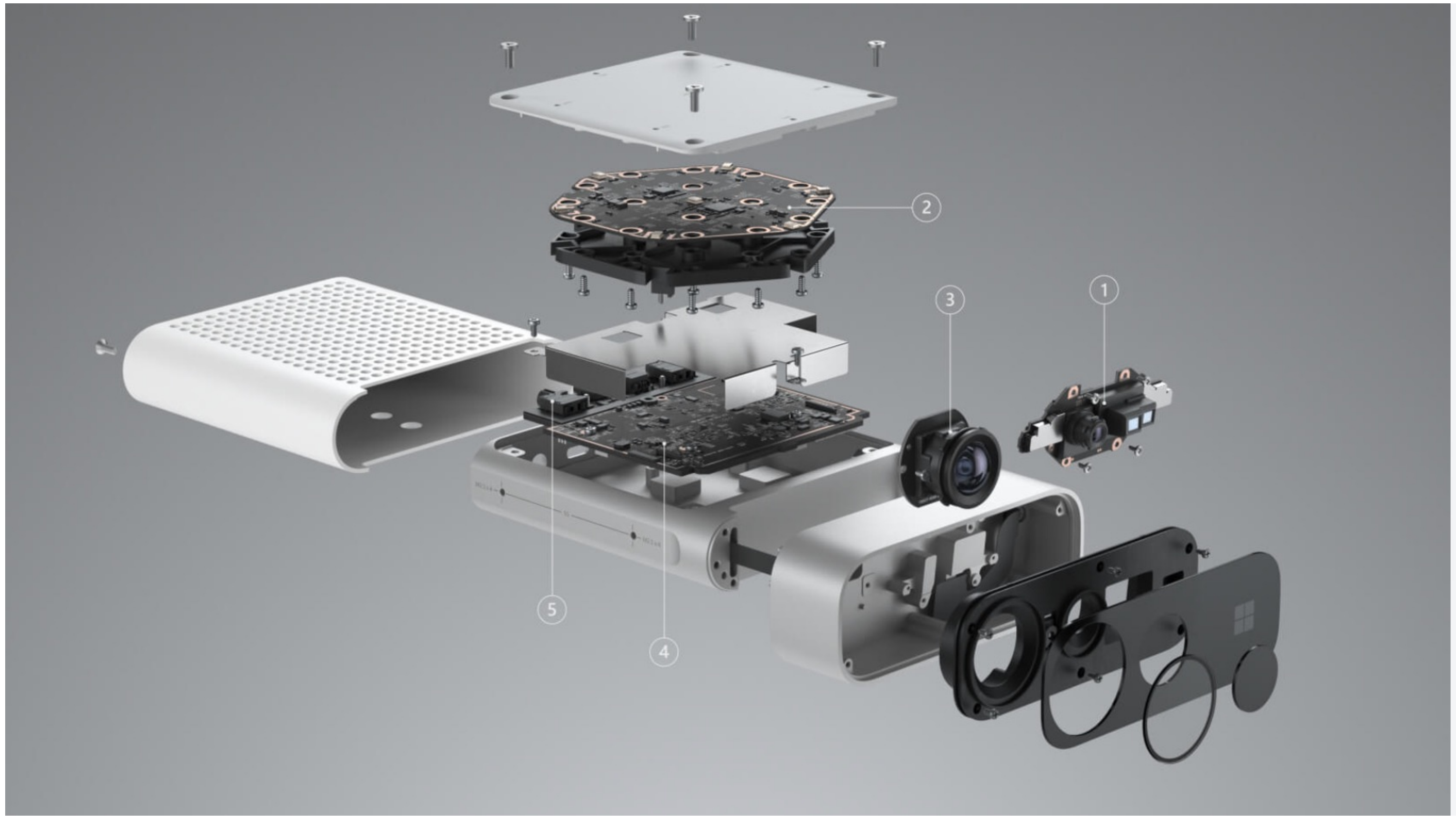
Sensors

- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors

The Microsoft Azure Kinect DK sensor

https://azure.microsoft.com/en-us/topic/mixed-reality/#demystifying

Sensors

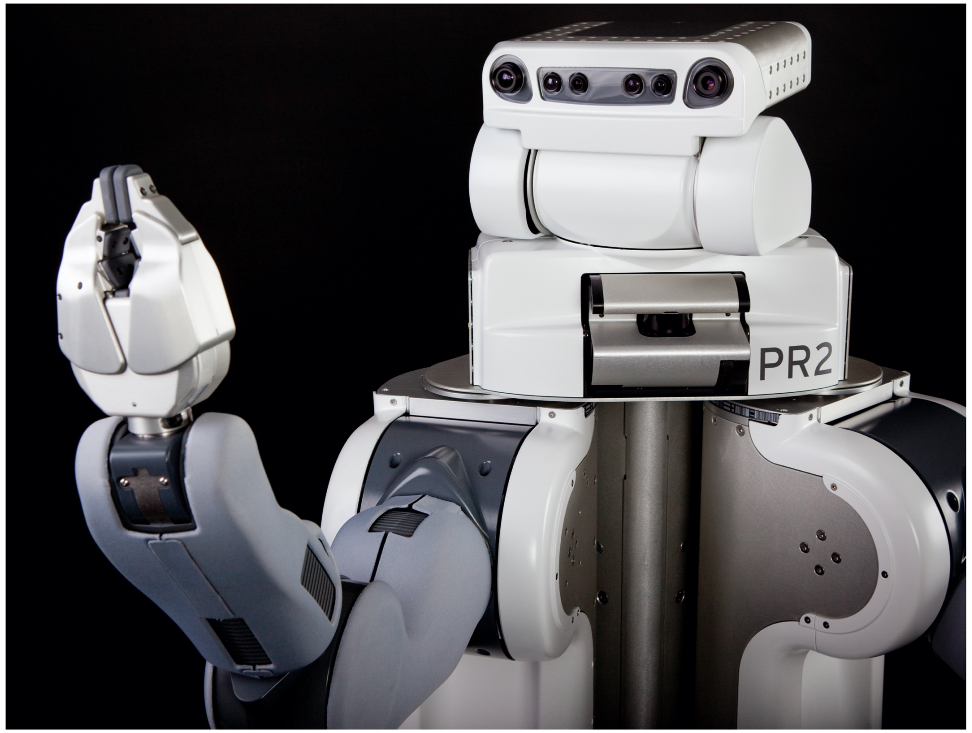


1. 1-MP depth sensor with wide and narrow field-of-view (FOV) options that help you optimize for your application
2. 7-microphone array for far-field speech and sound capture
3. 12-MP RGB video camera for an additional color stream that's aligned to the depth stream
4. Accelerometer and gyroscope (IMU) for sensor orientation and spatial tracking
5. External sync pins to easily synchronize sensor streams from multiple Kinect devices

https://azure.microsoft.com/en-us/services/kinect-dk/#industries

Sensors

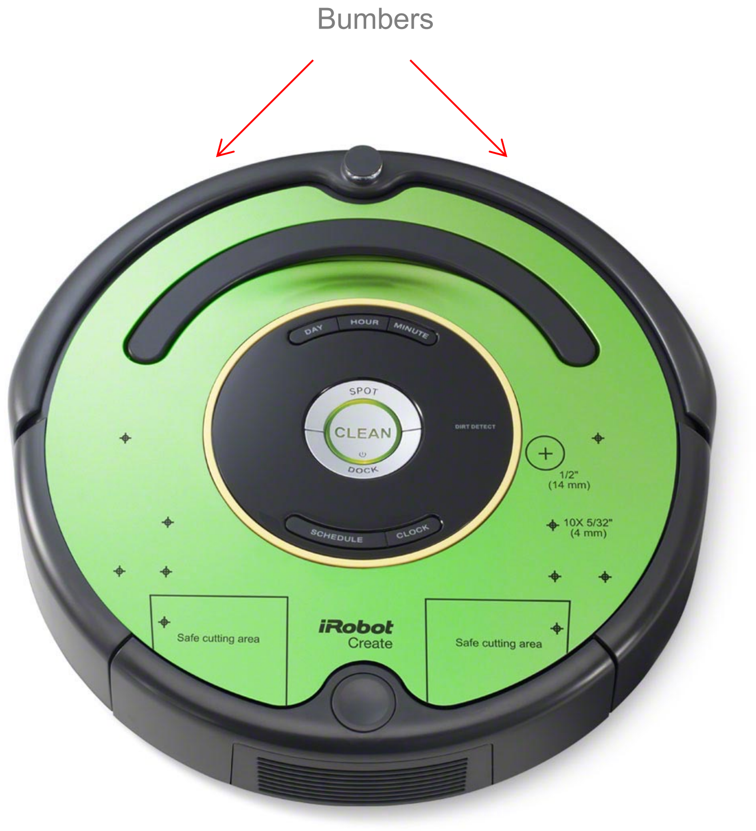
- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors



C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Sensors

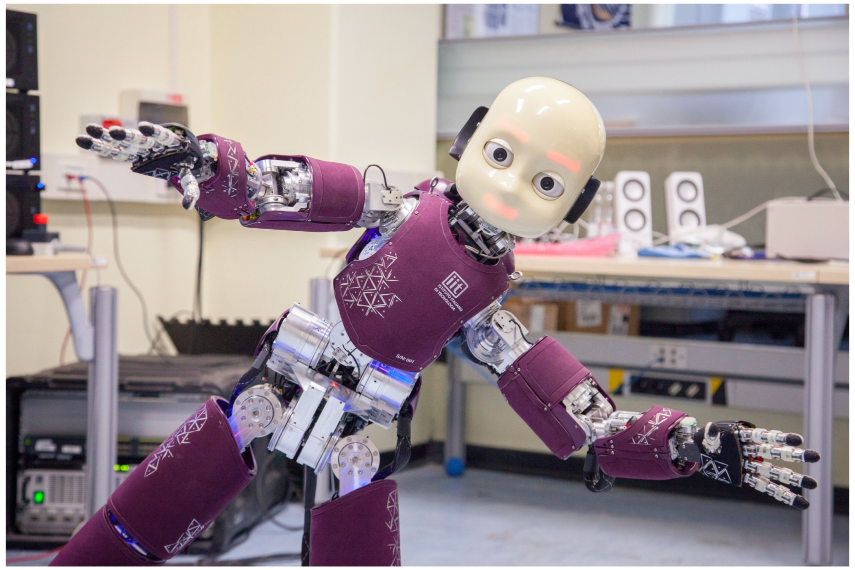
- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors



C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Sensors

- Joint angle & angular velocity encoders
- Joint torque sensor
- Inertial Measurement Unit (IMU) accelerometer and gyroscope sensors
- RGB video cameras
- Depth cameras
- RGB-D cameras
- Microphone audio sensors
- Capacitive touch sensors
- Laser distance sensors
- Ultrasonic distance sensors
- Bumper touch sensors



C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Reading

C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020. Chapter 3: How a Robot Works.

https://www.human-robot-interaction.org/download/170/

M. Mataric, The Robotics Primer, MIT Press, 2007. Chapter 3.

Videos

Encoders (5:14):	http://encoder.com/videos/

Robot Components

• Sensors	To perceive the environment
• Actuators	To take action
• Effectors	To take action
• Controllers	For autonomy

Robot Components

- Sensors
- Actuators
- Effectors
- Controllers

Effectors are the mechanisms that the robot uses to interact physically with its environment

Effectors for manipulation,
i.e., moving objects in the environment

Effectors for locomotion:
moving the robot around the environment

Robot Components

- Sensors
- Actuators
- Effectors
- Controllers

Effectors are the mechanisms that the robot uses to interact physically with its environment

Effectors for manipulation,
i.e., moving objects in the environment

Effectors for locomotion:
moving the robot around the environment

Manipulator robotics

Mobile robotics

Two major subfields of robotics

Robot Components

- Sensors
- Actuators Mechanisms that physically move the effectors: i.e. actuate wheels, legs, arms, fingers, …

$\downarrow$

- Effectors
- Controllers

Definition

An actuator is an electromechanical device which converts energy into mechanical work

- Work is an activity involving a force and movement in the direction of the force

  work = force x distance

- Energy is the capacity to do work

- Power is the rate at which work is done

Linear vs. Rotary Actuators

- Linear Actuators

  The shaft of the linear actuators moves along its axis

- Rotary actuators

  The shaft of the rotary actuator rotates about its axis

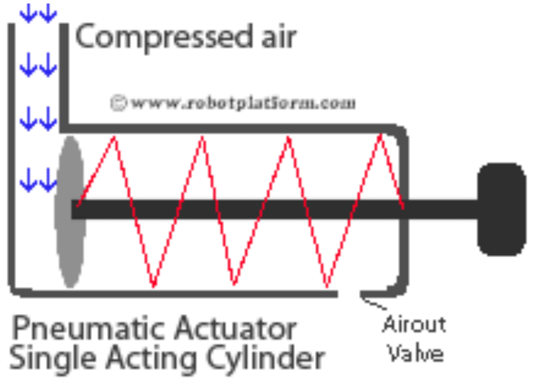
Types of Actuator

- Direct current (DC) motors
- Pneumatic actuators
- Hydraulic actuators
- Materials that are sensitive to light, heat, or chemicals can also be used as actuators

C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Pneumatic Actuator

- Compressed air creates a force that moves
  - Diaphragm
  - Piston
- Large & powerful
- Potentially dangerous
- Need to prevent leaks

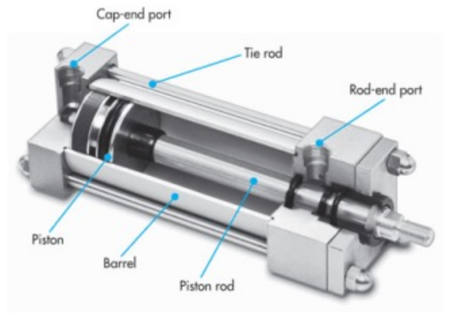


http://www.robotplatform.com/knowledge/actuators/types_of_actuators.html

Pneumatic Actuator

- Compressed air creates a force that moves
  - Diaphragm
  - Piston
- Large & powerful
- Potentially dangerous
- Need to prevent leaks

Pneumatic actuator

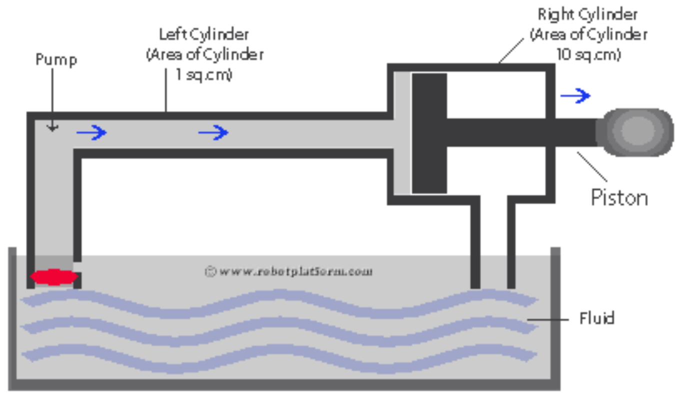


Section of pneumatic actuator

https://robocademy.com/2020/04/13/how-to-choose-an-actuator-for-your-robot/

Hydraulic Actuator

- Fluid pressure moves the actuator
- Large
- Powerful
- Precise
- Potentially dangerous
- Need to prevent leaks


Pump
Left Cylinder (Area of Cylinder 1 sq.cm)
Right Cylinder (Area of Cylinder 10 sq.cm)
Piston
Fluid
© www.robotplatform.com

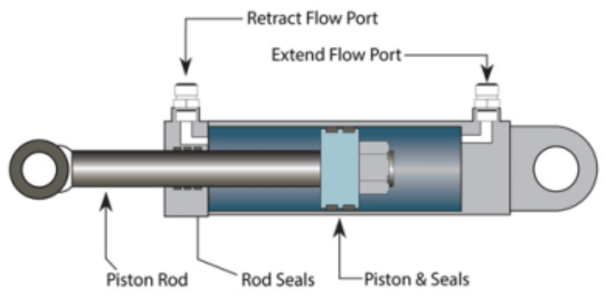
Hydraulic Actuator

http://www.robotplatform.com/knowledge/actuators/types_of_actuators.html

Hydraulic Actuator

- Fluid pressure moves the actuator
- Large
- Powerful
- Precise
- Potentially dangerous
- Need to prevent leaks

Hydraulic Actuator



Section of Hydraulic Actuator

https://robocademy.com/2020/04/13/how-to-choose-an-actuator-for-your-robot/

Video

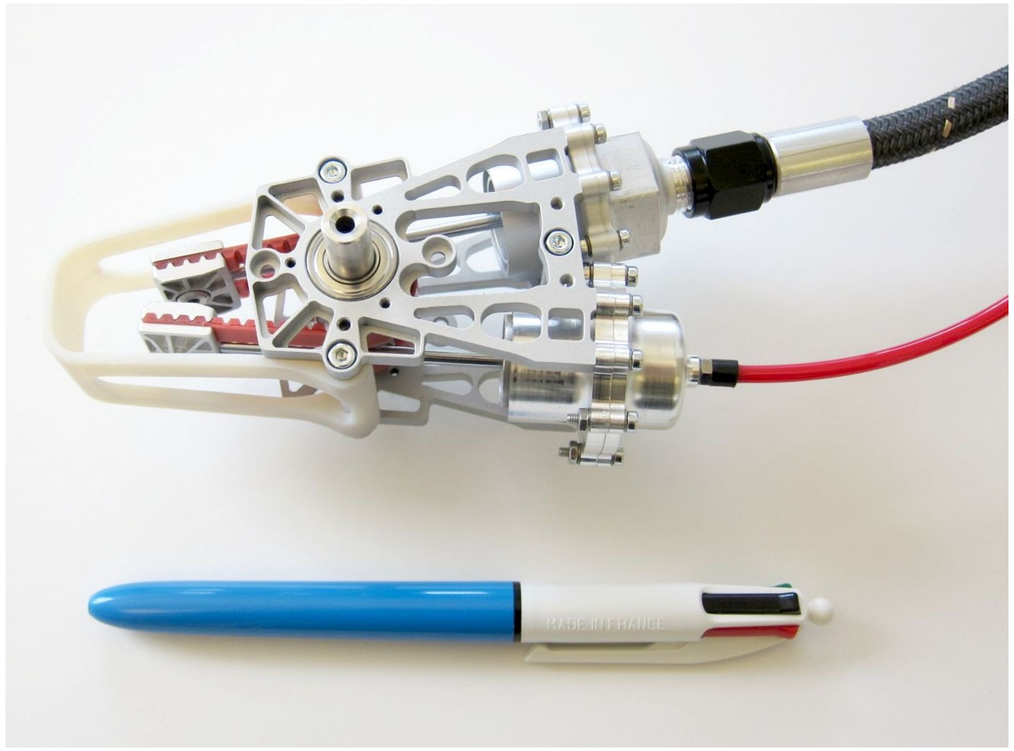
https://robots.ieee.org/robots/alphadog/

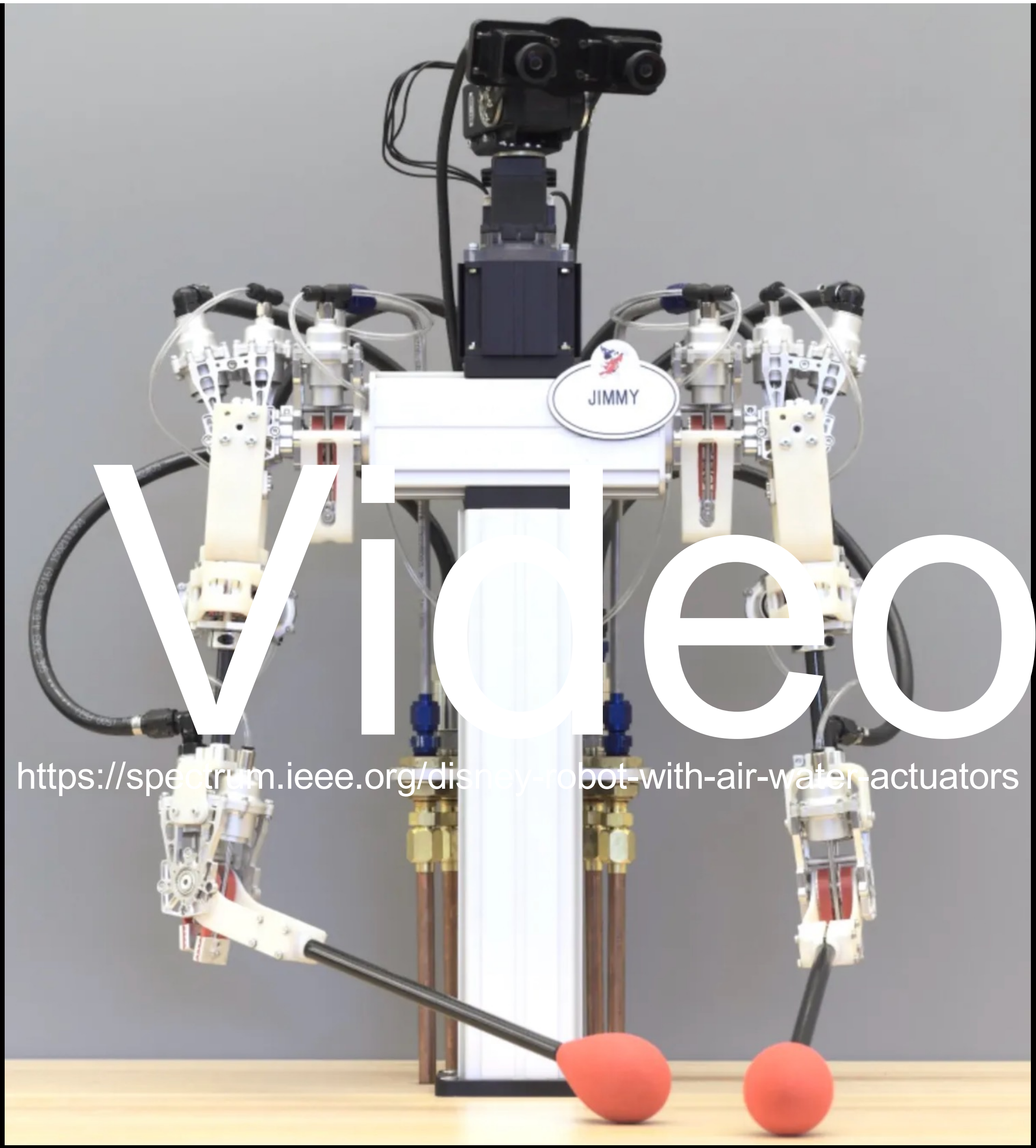
AlphaDog

Hybrid Actuator

"The actuator uses two fluid transmission lines, one with air and one with water. The air transmission (thin red tube in the picture above) works as a preloaded spring and damping system, while the water transmission (thicker black tube) actuates the joint. Each transmission is attached to cylinders containing a rolling diaphragm. The diaphragm moves back and forth, pushing a rod. A gear system (red geared pieces) converts the rod's linear motion into rotation. The actuator weighs only 120 grams and can deliver up to 4.5 newton meters of continuous torque with a 135° range of motion."

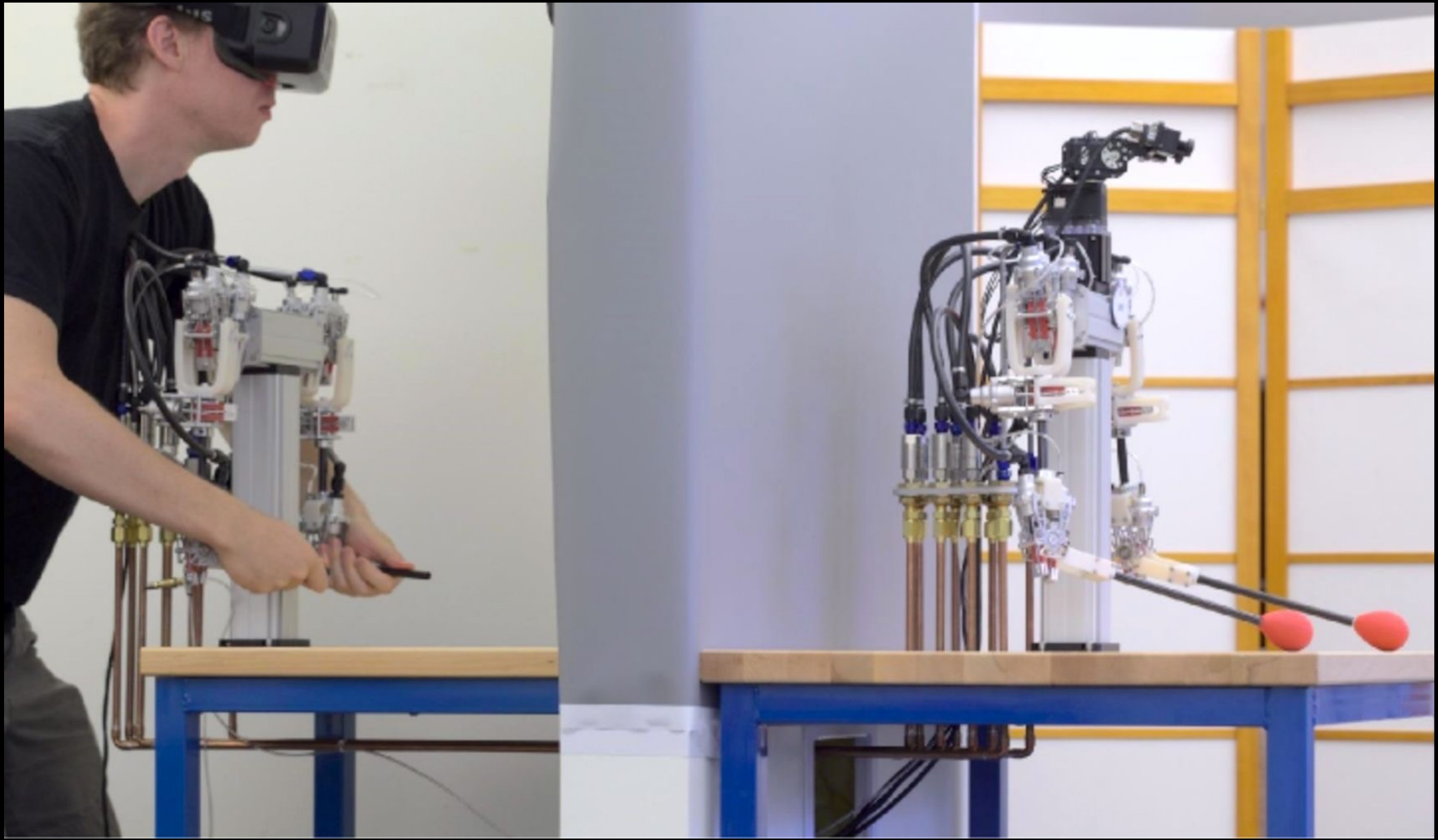
E. Guizzo, Disney Robot With Air-Water Actuators Shows Off "Very Fluid" Motions Meet Jimmy, a robot puppet powered by fluid actuators, IEEE Spectrum, 2016.


https://spectrum.ieee.org/disney-robot-with-air-water-actuators



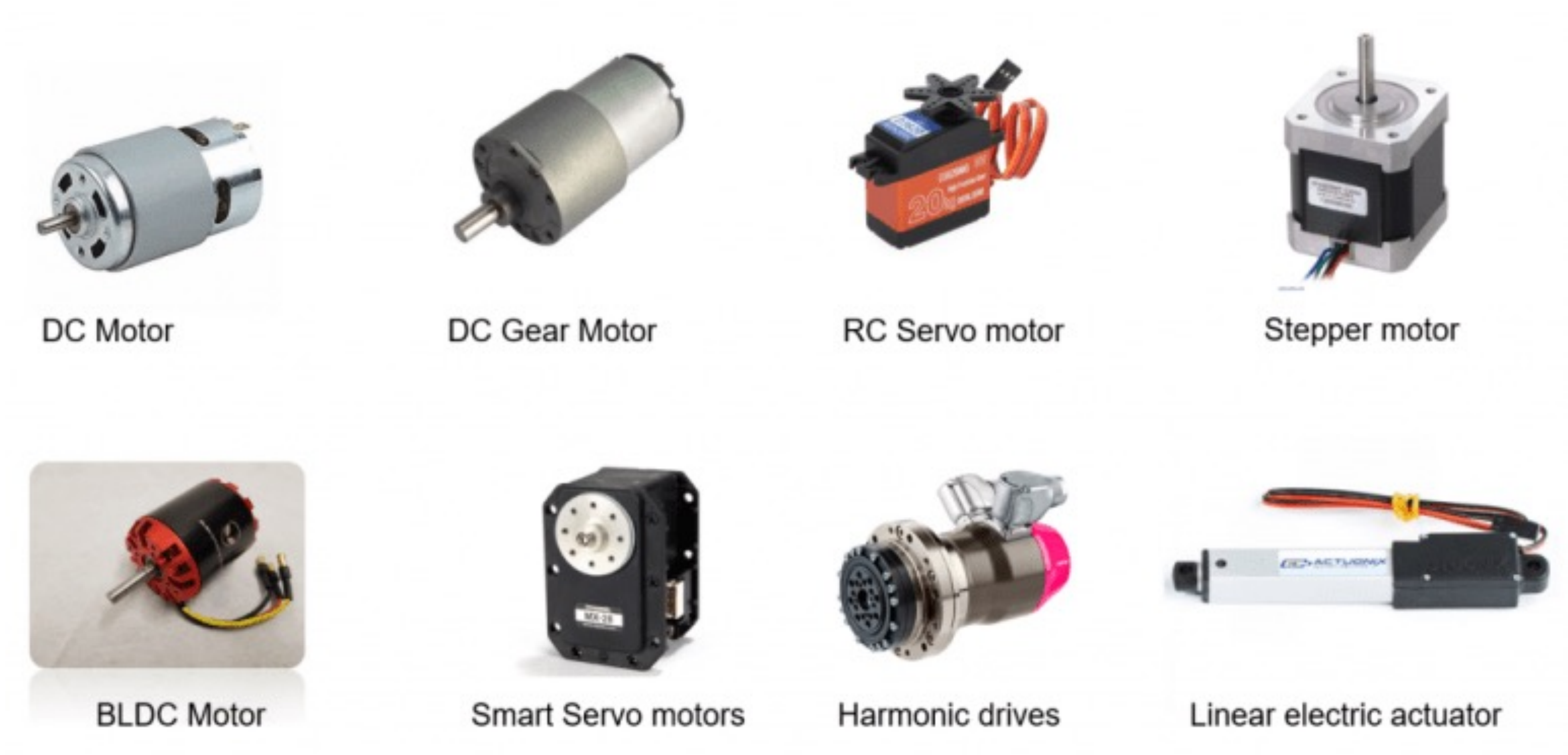
Video

https://spectrum.ieee.org/disney-robot-with-air-water-actuators



https://spectrum.ieee.org/disney-robot-with-air-water-actuators

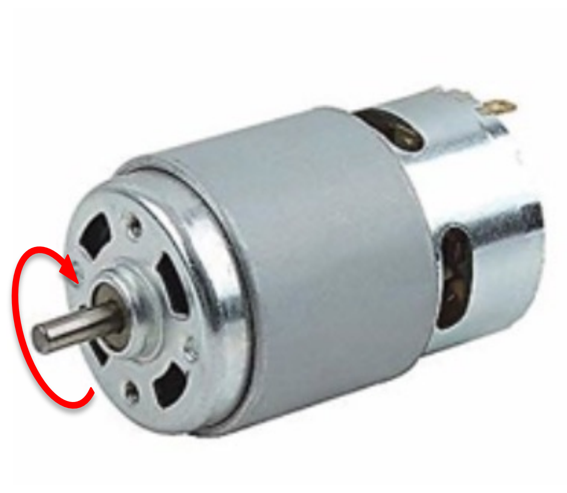
Direct Current (DC) Motors



https://robocademy.com/2020/04/13/how-to-choose-an-actuator-for-your-robot/

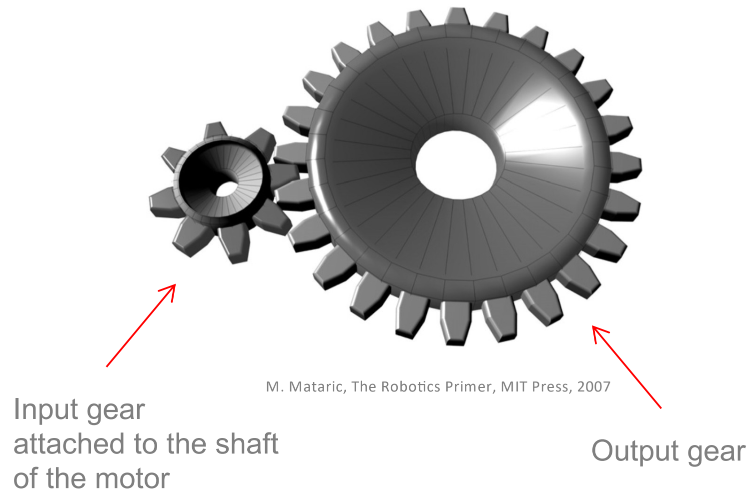
Direct Current (DC) Motors

- Simple, inexpensive, easy to use, and easy to source
- Wide variety of types and specifications
- Electrical energy is converted into mechanical (kinetic) energy resulting in rotation of the shaft



Direct Current (DC) Motors

- To make standard DC motors useful for robotics, we use gears to reduce the rotational velocity and increase torque
- The force generated at the edge of a gear is the ratio of the torque to the radius of the gear
- By combining gears with different radii, we can change the amount of force and torque that is generated.


M. Mataric, The Robotics Primer, MIT Press, 2007
Input gear attached to the shaft of the motor
Output gear

3-to-1 (3:1) gear reduction

8-tooth cog rotates three times to rotate the 24-tooth cog once

Direct Current (DC) Motors

- Gears can be organized in series or "ganged"
  - Two 3:1 gears in series results in a 9:1 reduction
  - Three 3:1 gears in series results in a 27:1 reduction
  - …

https://www.robotshop.com/ca/en/6v-2981-micro-metal-gearmotor-hp-100rpm.html

Direct Current (DC) Motors

If the gear teeth do not mesh properly, we get backlash

- The gear mechanism can move back and forth without turning to output gear
- This can lead to error in the positioning of the output gear
- Reducing backlash requires tight meshing between the gear teeth
    - To avoid increase in friction and decrease in efficiency
    - Requires high-precision manufacturing and increase in cost
    - High-precision gearboxes are expensive

https://www.robotshop.com/ca/en/6v-2981-micro-metal-gearmotor-hp-100rpm.html

Direct Current (DC) Servo Motors

- Direct current (DC) servo motors
- Pneumatic actuators
- Hydraulic actuators
- Materials that are sensitive to light, heat, or chemicals can also be used as actuators

C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Direct Current (DC) Servo Motors

- DC motors rotate continuously in one direction
- Often, we need a motor that can move an effector to a particular position
- Motors that turn the shaft to a specific position are called servo motors (or servos, for short)

Used in the shoulder joint
of the Lynxmotion AL5D robot arm

https://www.robotshop.com/en/hs-805bb-giant-scale-servo-motor.html

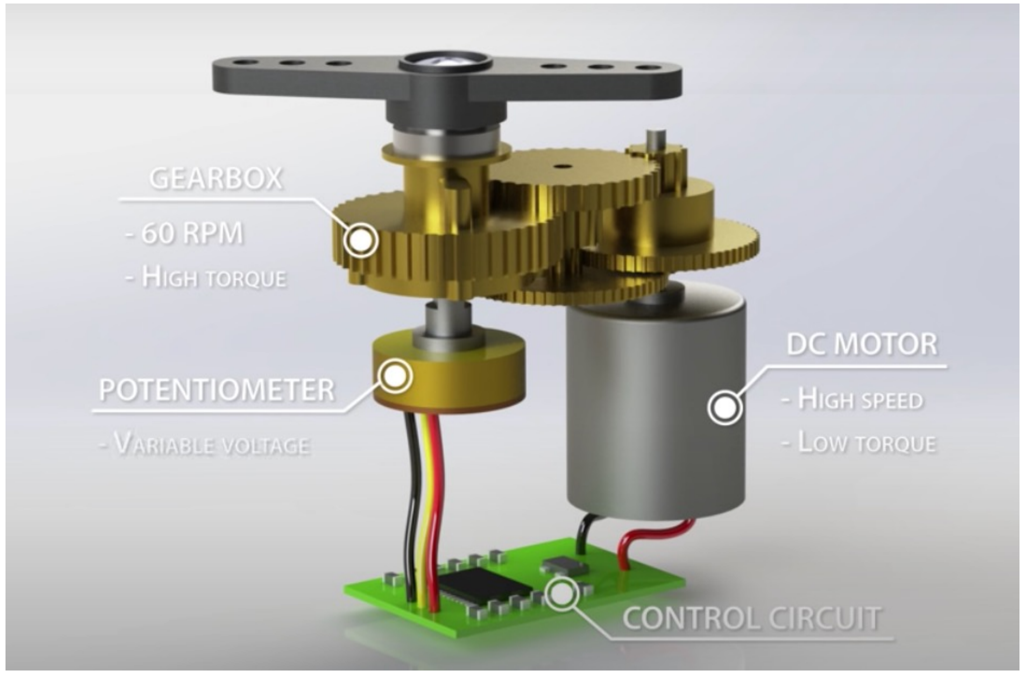
Direct Current (DC) Servo Motors

- DC motors rotate continuously in one direction
- Often, we need a motor that can move an effector to a particular position
- Motors that turn the shaft to a specific position are called servo motors (or servos, for short)

Direct Current (DC) Servo Motors

Direct current (DC) servo motor:

- DC motor
- Gearbox for gear reduction
- Position sensor for the output shaft
- Control circuit
  - Direction of rotation
  - Angle of rotation (+/- 180 degrees)



https://robocademy.com/2020/04/13/how-to-choose-an-actuator-for-your-robot/

Direct Current (DC) Servo Motors

Position sensor

- Potentiometer
- Encoder

which outputs the absolute or relative position of the motor's output shaft

Direct Current (DC) Servo Motors

- For motors used in a robot’s arms, legs, and head
  - the controller typically performs position control to rotate the motor toward a given joint angle
  - Also performs velocity control and torque (force x distance) control
- For motors used in wheels on a mobile base
  - the controller typically performs velocity control to rotate the motor at the required joint velocity

Reading

C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020. Chapter 3: How a Robot Works.

https://www.human-robot-interaction.org/download/170/

M. Mataric, The Robotics Primer, MIT Press, 2007. Chapters 3 and 4.

Videos

Encoders (1:29):	https://robots.ieee.org/robots/alphadog/
Jimmy (0:59):	https://spectrum.ieee.org/disney-robot-with-air-water-actuators

Robot Components

- Sensors
- Actuators
- Effectors
- Controllers

Effectors

Effectors for locomotion

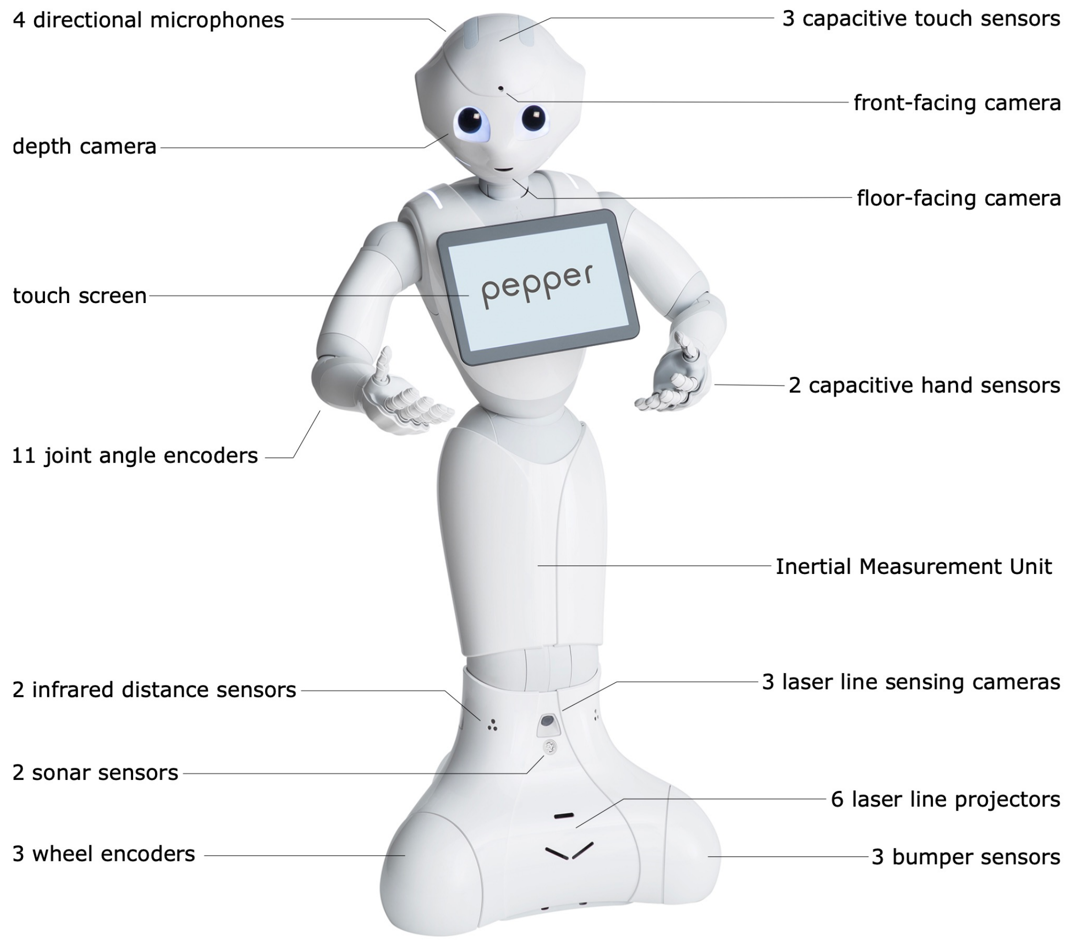
- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors much be matched to the task the robot has to do and the environment in which it has to work

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

End-effectors



C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Degrees of Freedom (DOF)

- The minimum number of coordinates required to completely specify the motion of a mechanical system

- Determines what poses (positions and orientations) the robot can achieve

- Determines how it can move

Degrees of Freedom (DOF)

- It requires six degrees of freedom to position and orient a body in space
  - Three translational degrees of freedom
  - Three rotational degrees of freedom
- The position and orientation of a body is referred to as its pose
- Much more on pose specification later

Effectors

Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

End-effectors

Atlas

Atlas is the most agile humanoid in existence. It uses whole-body skills to move quickly and balance dynamically. It can lift and carry objects like boxes and crates, but its favorite tricks are running, jumping, and doing backflips.

CREATOR
Boston Dynamics

COUNTRY
United States 🇺🇸

YEAR
2016

TYPE
Humanoids, Industrial

Source: https://robots.ieee.org/robots/atlas2016/

Effectors

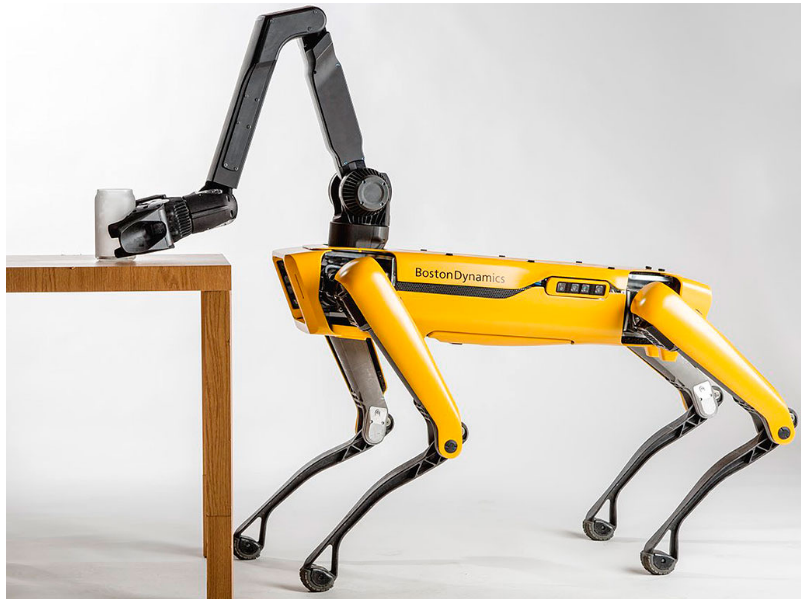
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\ \text{Grippers}\\ \text{Tools}\end{array}\right\}$ End-effectors



Spot

Spot is a compact, nimble four-legged robot that can trot around your office, home, or outdoors. It can map its environment, sense and avoid obstacles, climb stairs, and open doors. It can also fetch you a drink.

CREATOR
Boston Dynamics

COUNTRY
United States 🇺🇸

YEAR
2016

TYPE
Industrial, Research

Source: https://robots.ieee.org/robots/spotmini/

Effectors

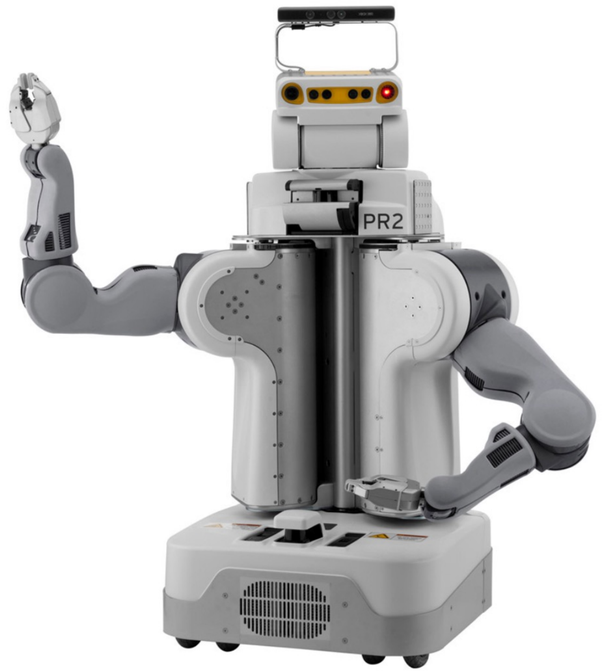
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\ \text{Grippers}\\ \text{Tools}\end{array}\right\}$ End-effectors



PR2

The PR2 is one of the most advanced research robots ever built. Its powerful hardware and software systems let it do things like clean up tables, fold towels, and fetch you drinks from the fridge.

CREATOR
Willow Garage

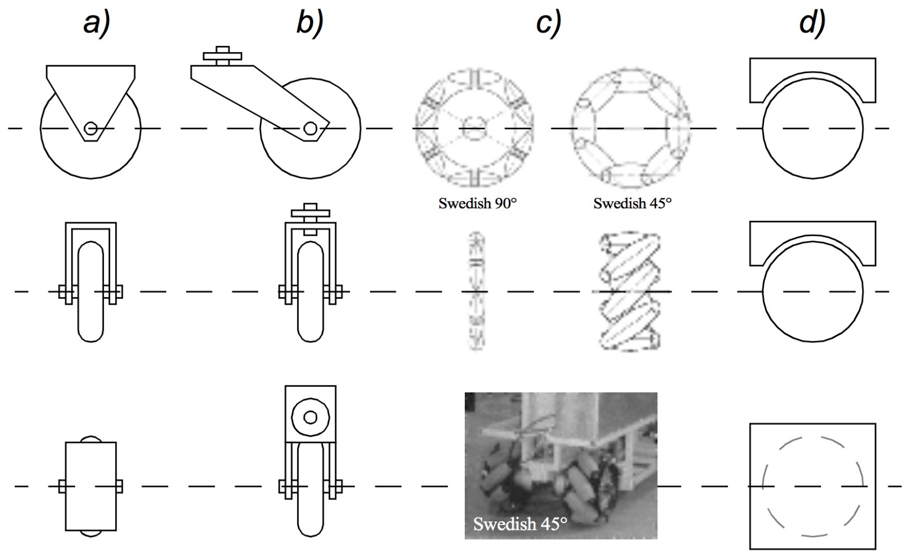
COUNTRY
United States 🇺🇸

YEAR
2010

TYPE
Research, Humanoids

Source: https://robots.ieee.org/robots/pr2/

Wheels



Source: R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, MIT Press, 2004

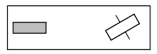
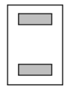
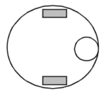
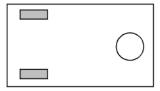
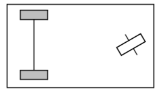
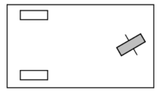
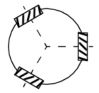
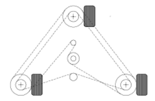
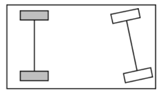
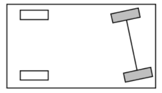
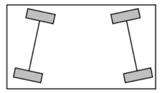
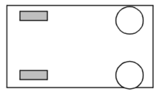
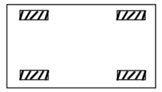
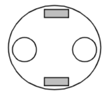
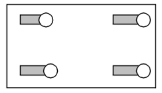
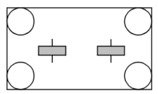
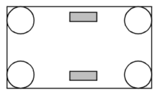
(a) Standard wheel ← Rotation about axle for movement and about contact point for steering

(b) Castor wheel ← Rotation about axle for movement and about vertical axis for steering; imparting a force on the robot body when steering

(c) Swedish wheel ← Rotation about axle for movement but also about rollers allowing movement is any direction

(d) Ball or spherical wheel ← Omnidirectional wheel: can spin in any direction

Wheels

# of wheels	Arrangement	Description	Typical examples
2		One steering wheel in the front, one traction wheel in the rear	Bicycle, motorcycle
		Two-wheel differential drive with the center of mass (COM) below the axle	Cye personal robot
3		Two-wheel centered differential drive with a third point of contact	Nomad Scout, smartRob EPFL
		Two independently driven wheels in the rear/front, 1 unpowered omnidirectional wheel in the front/rear	Many indoor robots, including the EPFL robots Pygmalion and Alice
		Two connected traction wheels (differential) in rear, 1 steered free wheel in front	Piaggio minitrucks
		Two free wheels in rear, 1 steered traction wheel in front	Neptune (Carnegie Mellon University), Hero-1
		Three motorized Swedish or spherical wheels arranged in a triangle; omnidirectional movement is possible	Stanford wheel Tribolo EPFL, Palm Pilot Robot Kit (CMU)
		Three synchronously motorized and steered wheels; the orientation is not controllable	"Synchro drive" Denning MRV-2, Georgia Institute of Technology, I-Robot B24, Nomad 200
4		Two motorized wheels in the rear, 2 steered wheels in the front; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with rear-wheel drive
		Two motorized and steered wheels in the front, 2 free wheels in the rear; steering has to be different for the 2 wheels to avoid slipping/skidding.	Car with front-wheel drive
		Four steered and motorized wheels	Four-wheel drive, four-wheel steering Hyperion (CMU)
		Two traction wheels (differential) in rear/front, 2 omnidirectional wheels in the front/rear	Charlie (DMT-EPFL)
		Four omnidirectional wheels	Carnegie Mellon Uranus
		Two-wheel differential drive with 2 additional points of contact	EPFL Khepera, Hyperbot Chip
		Four motorized and steered castor wheels	Nomad XR4000
6		Two motorized and steered wheels aligned in center, 1 omnidirectional wheel at each corner	First
		Two traction wheels (differential) in center, 1 omnidirectional wheel at each corner	Terregator (Carnegie Mellon University)

Icons for the each wheel type are as follows:

○	unpowered omnidirectional wheel (spherical, castor, Swedish);
▨	motorized Swedish wheel (Stanford wheel);
▭	unpowered standard wheel;
▬	motorized standard wheel;
▬○	motorized and steered castor wheel;
╪	steered standard wheel;
⌶	connected wheels.

Source: R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, MIT Press, 2004

Effectors

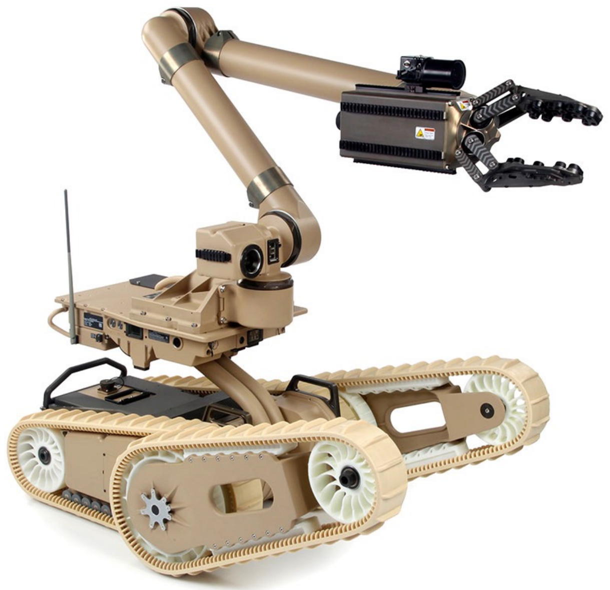
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\\text{Grippers}\\\text{Tools}\end{array}\right\}$ End-effectors



Kobra

Kobra is a rugged, remote control robot designed to search for explosives and carry out reconnaissance missions. It rolls on tank-like treads, and its manipulator arm can lift heavy payloads.

CREATOR
Endeavor Robotics
(Originally created by iRobot)

COUNTRY
United States 🇺🇸

YEAR
2011

TYPE
Military & Security, Disaster Response

Source: https://robots.ieee.org/robots/kobra/

Effectors

Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\\text{Grippers}\\\text{Tools}\end{array}\right\}$ End-effectors

Zipline

Zipline is an autonomous fixed-wing aircraft drone used to carry blood and medicine from a distribution center to wherever it's needed. It can launch within minutes, and travel in any weather.

CREATOR
Zipline

COUNTRY
United States 🇺🇸

YEAR
2016

TYPE
Drones, Medical

Source: https://robots.ieee.org/robots/zipline/

Effectors

Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\\text{Grippers}\\\text{Tools}\end{array}\right\}$ End-effectors

Salamandra robotica II

Salamandra robotica II is an amphibious robot inspired by the salamander's anatomy and nervous system. It's used to study robot locomotion and test neurobiological models in real environments.

CREATOR
Biorobotics Laboratory at EPFL

COUNTRY
Switzerland

YEAR
2012

TYPE
Research

Source: https://robots.ieee.org/robots/salamandra/

Effectors

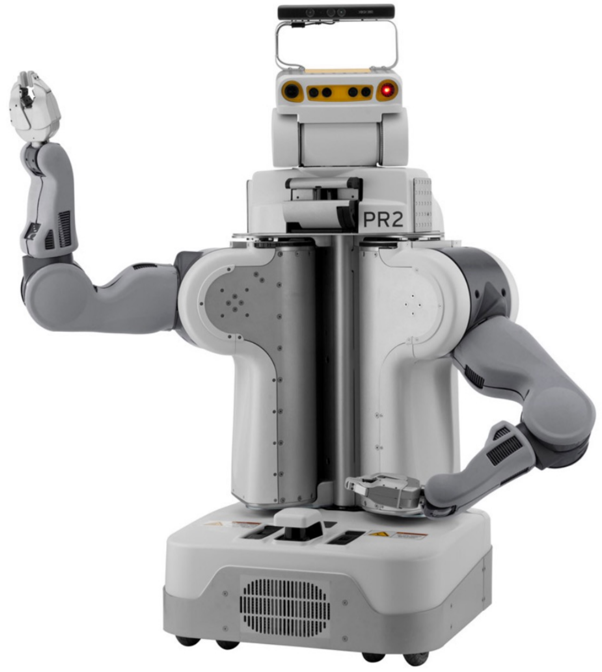
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\ \text{Grippers}\\ \text{Tools}\end{array}\right\}$ End-effectors



PR2

The PR2 is one of the most advanced research robots ever built. Its powerful hardware and software systems let it do things like clean up tables, fold towels, and fetch you drinks from the fridge.

CREATOR
Willow Garage

COUNTRY
United States 🇺🇸

YEAR
2010

TYPE
Research, Humanoids

Source: https://robots.ieee.org/robots/pr2/

Effectors

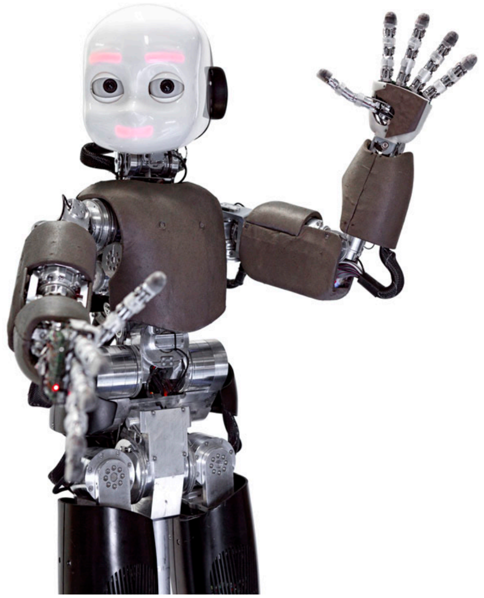
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\ \text{Grippers}\\ \text{Tools}\end{array}\right\}$ End-effectors



iCub

iCub is a child-size humanoid robot capable of crawling, grasping objects, and interacting with people. It's designed as an open source platform for research in robotics, AI, and cognitive science.

CREATOR
RoboCub Consortium and IIT

COUNTRY
Italy 🇮🇹

YEAR
2004

TYPE
Humanoids, Research

Source: https://robots.ieee.org/robots/icub/

Effectors

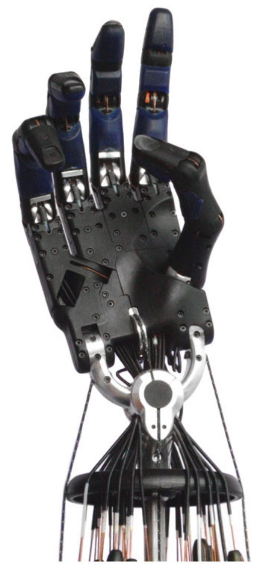
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\\text{Grippers}\\\text{Tools}\end{array}\right\}$ End-effectors



Shadow Hand

The Shadow Dexterous Hand is one of the most advanced robot hands in the world. It's designed to replicate as much of the functionality, dimensions, and range of motion of the human hand as possible.

CREATOR
Shadow Robot Company

COUNTRY
United Kingdom 🇬🇧

YEAR
2004

TYPE
Industrial, Telepresence, Research

Source: https://robots.ieee.org/robots/davinci/

Effectors

Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\ \text{Grippers}\\ \text{Tools}\end{array}\right\}$ End-effectors

C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Effectors

Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\ \text{Grippers}\\ \text{Tools}\end{array}\right\}$ End-effectors

Sawyer

Sawyer is an industrial collaborative robot designed to help out with manufacturing tasks and work alongside humans. You can teach it new tasks by demonstrating what to do using the robot's own arm.

CREATOR
Rethink Robotics

COUNTRY
United States 🇺🇸

YEAR
2015

TYPE
Industrial

Source: https://robots.ieee.org/robots/sawyer/

Effectors

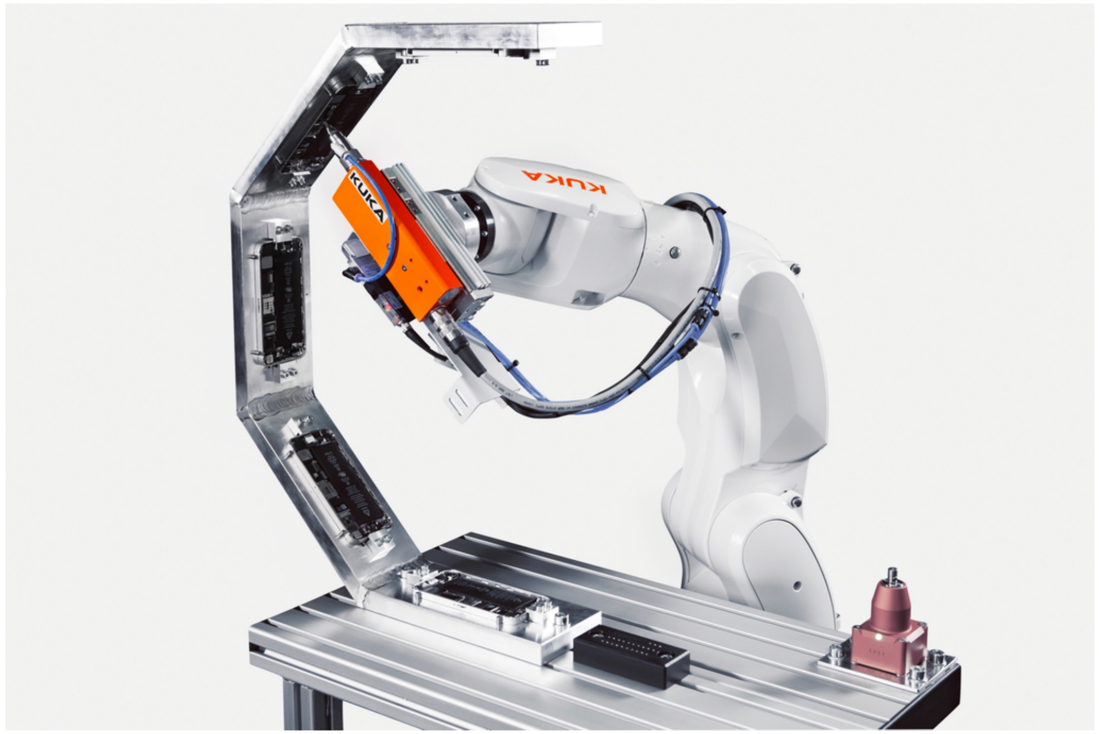
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

$\left.\begin{array}{l}\text{Hands}\\ \text{Grippers}\\ \text{Tools}\end{array}\right\}$ End-effectors



C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020

Effectors

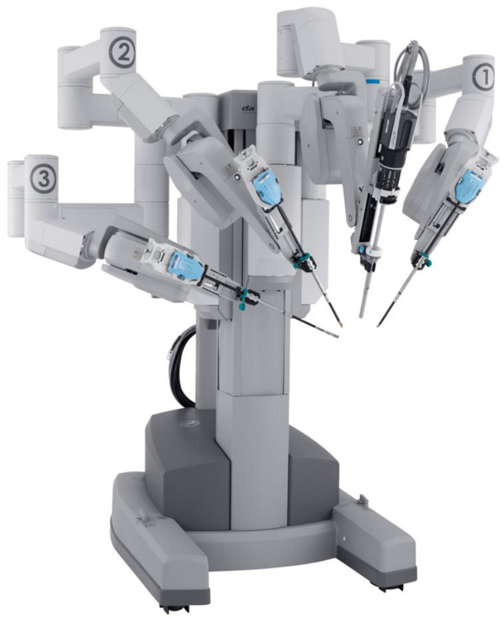
Effectors for locomotion

- Legs
- Wheels
- Tracks
- Wings
- Flippers

Effectors for manipulation

- Arms
- Hands
- Grippers
- Tools

End-effectors



Da Vinci

The da Vinci is a surgical robot designed for minimally invasive procedures. It has four arms equipped with surgical instruments and cameras that a physician controls remotely from a console.

CREATOR
Intuitive Surgical

COUNTRY
United States 🇺🇸

YEAR
1999

TYPE
Medical

Source: https://robots.ieee.org/robots/davinci/

Reading

C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020. Chapter 3: How a Robot Works.

https://www.human-robot-interaction.org/download/170/

M. Mataric, The Robotics Primer, MIT Press, 2007. Chapters 5 and 6.

Robot Components

- Sensors
- Actuators
- Effectors
- Controllers

Control Systems

Controllers

Plural: there may be different controllers for different sub-systems in the robot

- Enable the robot to be autonomous
- Autonomy is the ability to make one's own decisions and act on them, based on
  - Sensor inputs
  - Stored knowledge
- Autonomy can be complete or partial
- There are different approaches to the organization of controllers and sub-systems

See later

Control Theory Terminology

Goal

- Get some process or plant to a desired state
- Maintain that state

"plant" is a term used to refer to the system being controlled
e.g., water tank, pipeline, mobile robot

Example states:

Water level in a tank
Temperature of water in a tank
Flow rate of a pipeline
Speed of a mobile robot
Position of a mobile robot
Orientation of a mobile robot

These are referred to as "process variables"

Control Theory Terminology

Strictly speaking, "a plant in control theory is the combination of process and actuator"
https://en.wikipedia.org/wiki/Plant_(control_theory)

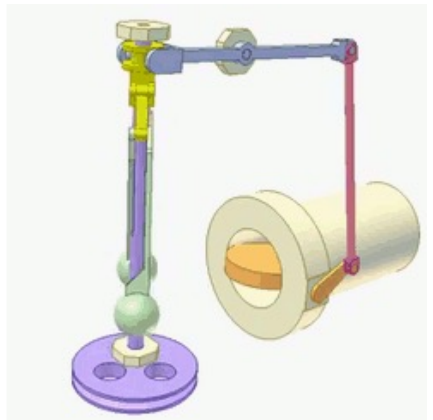
- Plant: the process or device to be controlled
- Process variable (PV): the actual state of the process or plant
- Set point (SP): the desired state of the process or plant
- Error: the difference between PV and SP

Control Theory Terminology

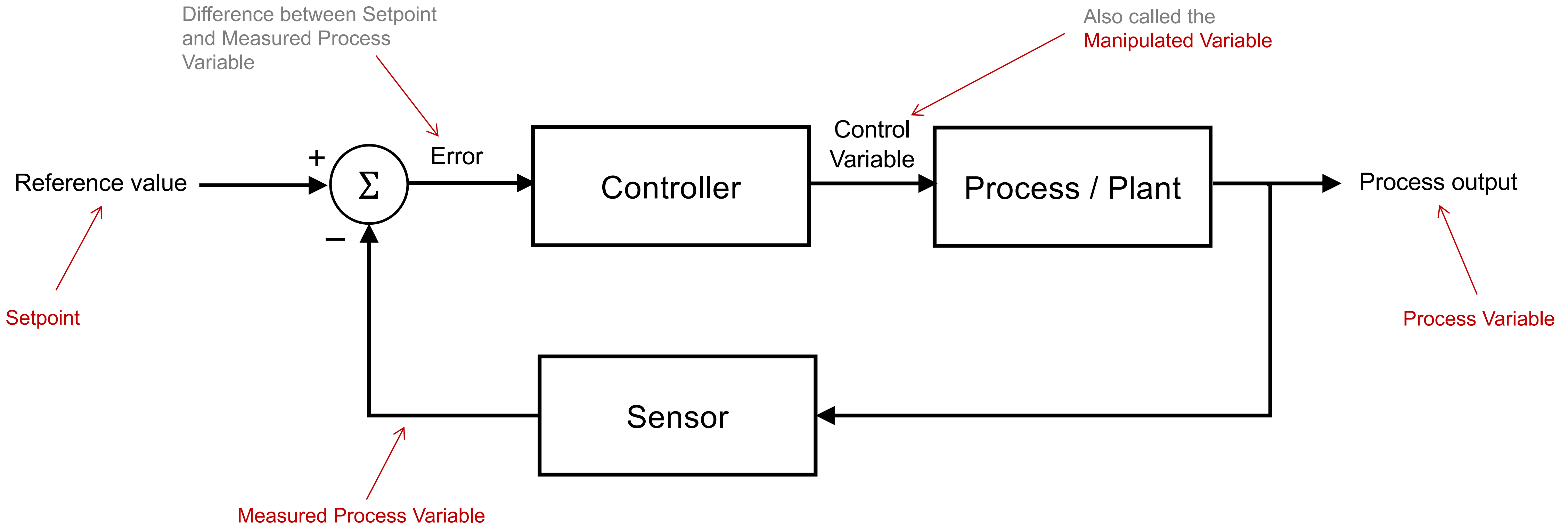
- Effector: a mechanism that changes the state of the process or plant (i.e. control action)
- Sensor: a mechanism that measures the state of the process or plant
- Control variable: the input to the effector
- Controller:

Can be a physical device (e.g., a mechanical governor) or software implementing a control algorithm (see later)

A mechanism to identify the value of the control signal that reduces the error to zero as quickly as possible, without overshoot, in a stable manner

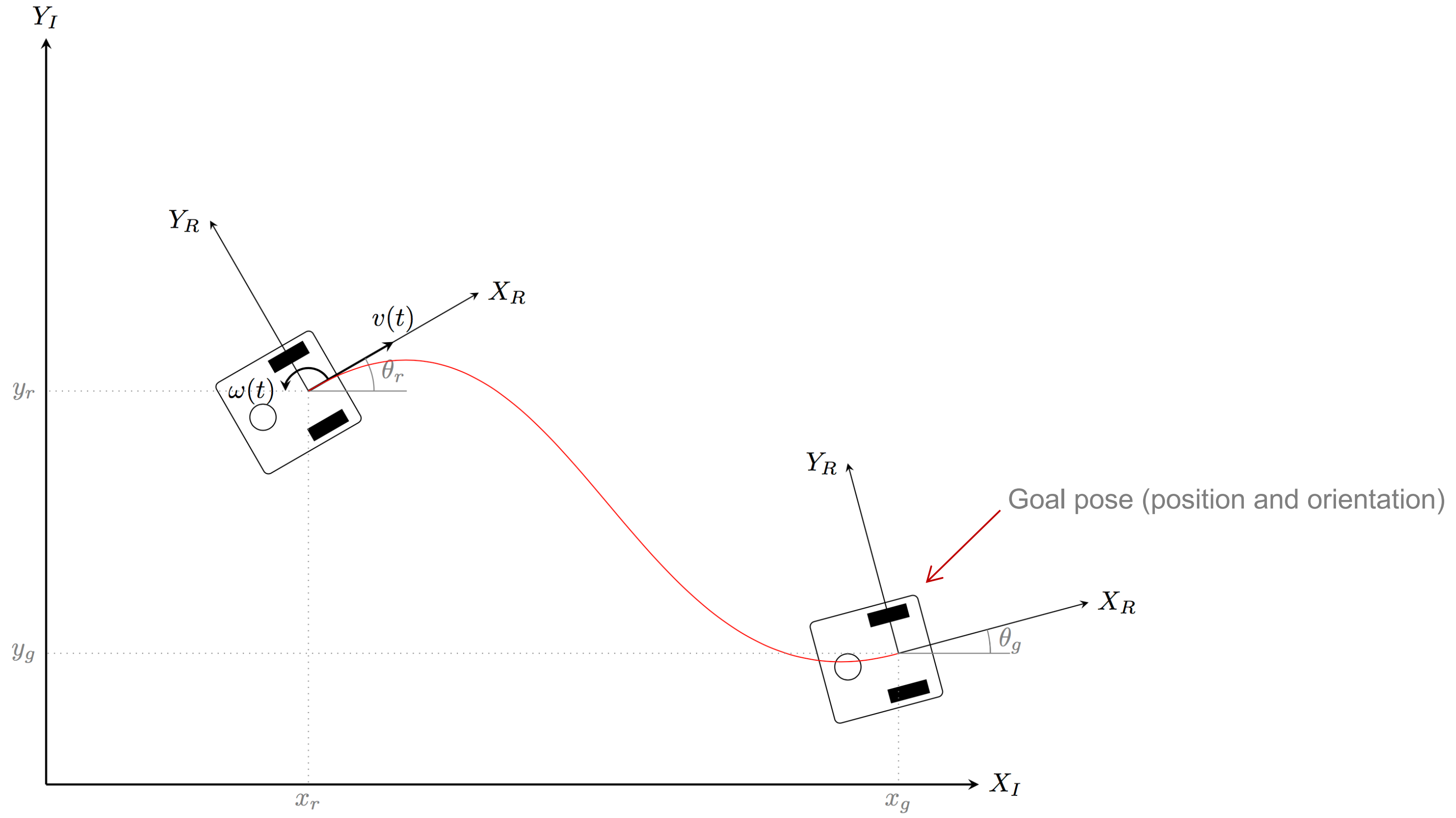

https://www.mech.kuleuven.be/en/tme/thermotechnisch-instituut/basisprincipes/Watt-regulator

Closed-loop Feedback Control


Difference between Setpoint and Measured Process Variable
Also called the Manipulated Variable
+
Σ
Error
Controller
Control Variable
Process / Plant
Reference value
Process output
−
Setpoint
Process Variable
Sensor
Measured Process Variable

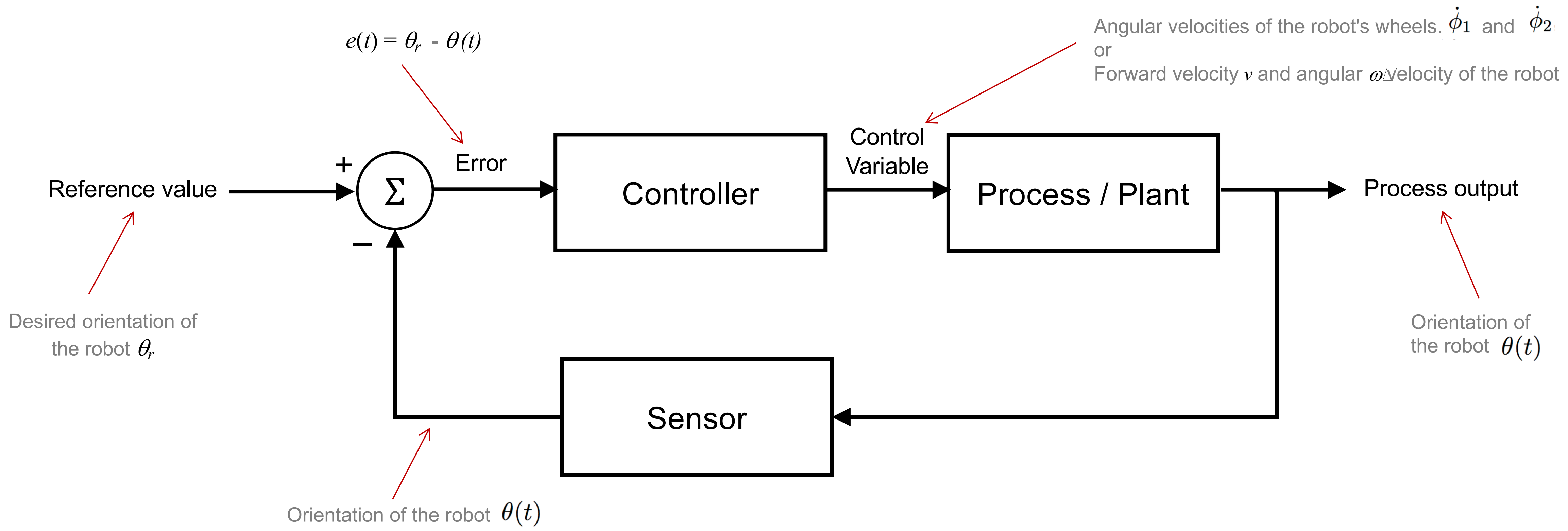
Closed-loop Feedback Control

For example, controlling the orientation of a mobile robot



Closed-loop Feedback Control

For example, controlling the orientation of a mobile robot



$e(t) = \theta_r - \theta(t)$
Angular velocities of the robot's wheels. $\dot{\phi}_1$ and $\dot{\phi}_2$
or
Forward velocity v and angular ω velocity of the robot
+
Σ
–
Error
Reference value
Controller
Control Variable
Process / Plant
Process output
Desired orientation of the robot θ_r
Orientation of the robot $\theta(t)$
Sensor
Orientation of the robot $\theta(t)$

Closed-loop Feedback Control

Control variable is a function of the error: $f(e)$

e = error between

desired value (i.e. the setpoint)

and

the actual value (i.e. the measured process value)

PID Controller

Which function?

f = "proportional to e"

f = "proportional to the accumulation of e" ← Integral

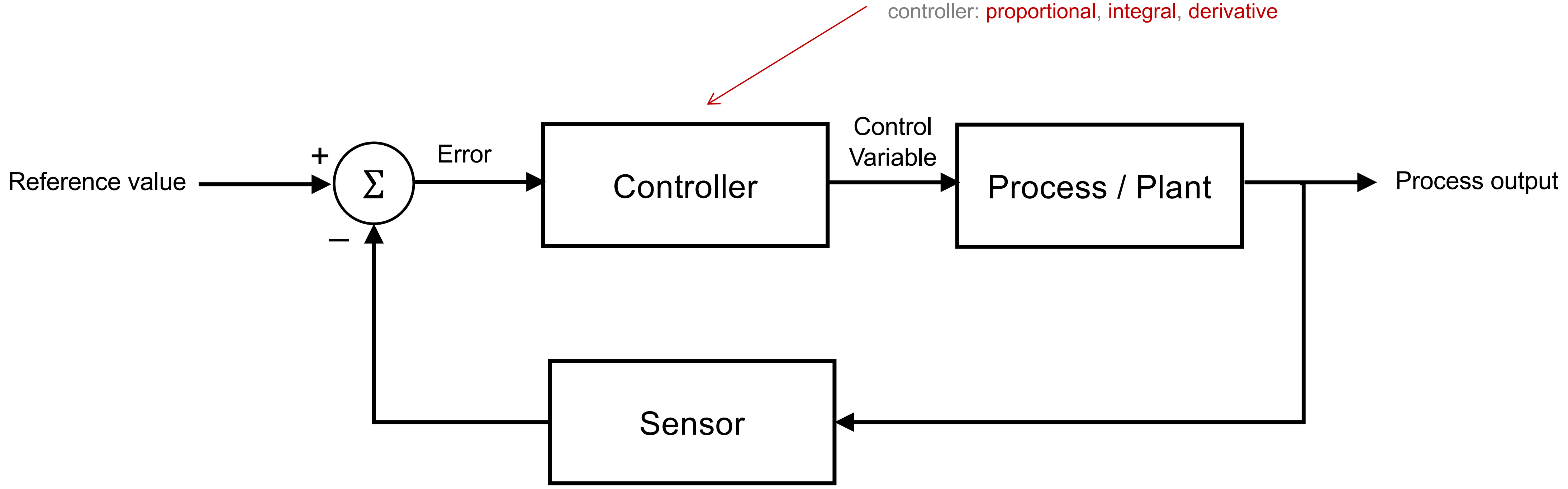
f = "proportional to the rate of change of e" ← Derivative

. . . or a combination of these

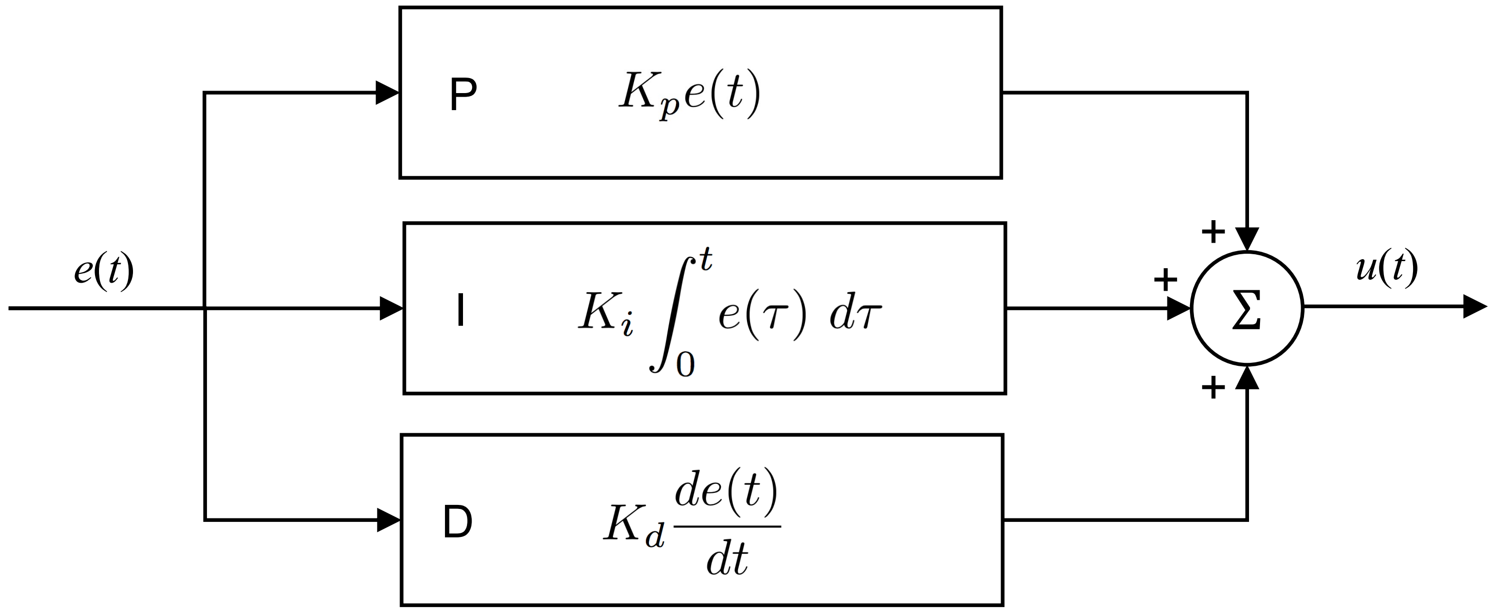
Each component is modulated by a respective gain: K_p, K_i, K_d

PID Controller

Classical control theory uses a PID controller: proportional, integral, derivative



PID Controller



$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

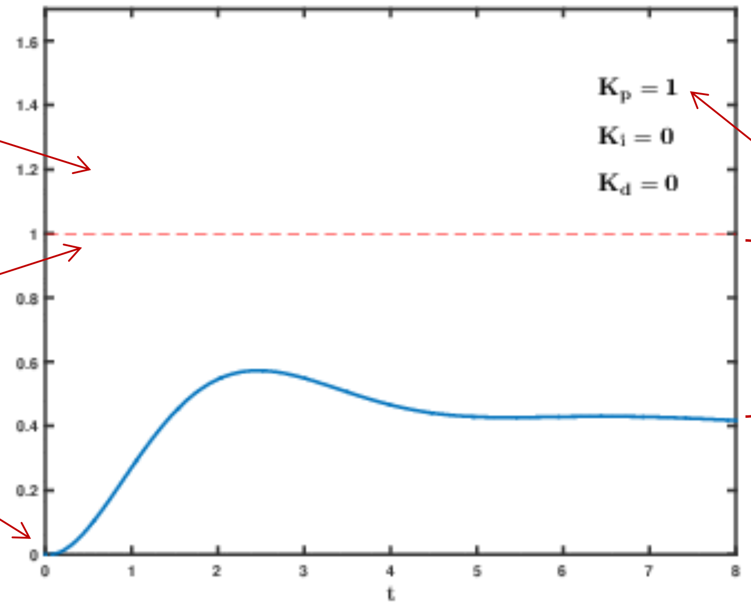
PID Controller

Effect of varying the three gains

The overshoot can be reduced by lowering the K_p but this will increase the time it takes to reduce the error

Setpoint

Initial state of the process variable



https://en.wikipedia.org/wiki/PID_controller

Note that there is a steady-state error for pure proportional control,

(i.e. when the gains of the integral and derivative terms are zero).

The integral term eliminates this.

PID Controller

Take-home message

The key to effective PID control is to use the right gain values

but

identifying them is difficult

Reading

C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, S. Šabanović, Human-Robot Interaction – An Introduction, Cambridge University Press, 2020. Chapter 3: How a Robot Works.

https://www.human-robot-interaction.org/download/170/

M. Mataric, The Robotics Primer, MIT Press, 2007. Chapters 3 and 10.

R. Murphy, Introduction to AI Robotics, MIT Press, 2000. Part I Robotic Paradigms: Overview.

Lecture Topics

$\left.\begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix}\right\}$ Principles

ROS

ROS is an open-source, meta-operating system for robots

It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management

It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers

http://wiki.ros.org/ROS/Introduction

Features

- Distributed computation
  - Divide software into small stand-alone parts that, together, achieve the overall goal
  - Communication between multiple concurrent processes that may or may not be running on the same computer
  - Based on component-based software engineering
- Software reuse
  - ROS's standard packages provide stable implementations of many important algorithms

Features

Community support

- Hardware drivers
- Libraries: PCL, OpenCV, TF, …
- Capabilities: navigation, manipulation, control, .
- Applications: fetching beer, making popcorn, …

ROS is not …

- A programming language
  - It supports C++, Lisp, Python, Java, among others
- Just a library
  - also include a central server, command-line tools, graphical tools, build systems.
- An integrated development environment (IDE)

ROS Distributions

- Major versions of ROS are called distributions
- Distributions are named using adjectives that start with successive letters of the alphabet
  - …, Groovy, Hydro, Indigo, Jade, Kinetic, Lunar, Melodic, Noetic, … (see http://wiki.ros.org/Distributions)
- Referred to in the ROS documentation by the term distro
- Different distributions use different build systems
  - Melodic uses catkin

Packages

- All ROS software is organized into packages

- A ROS package is a coherent collection of files
  - Serves a specific purpose.
  - Includes executables and supporting files

- All ROS software is part of one package or another

- rospack list provides a list of all installed ROS packages

ROS Master

- ROS software comprises a collection of small, independent, loosely-coupled programs called nodes that all run at the same time

Coupling is effected by sending messages on topics (see below)

- These nodes must be able to communicate with one another

- The part of ROS that facilitates this communication is called the ROS master

- To start the master, use the roscore command

Topics and Messages

- ROS nodes communicate by sending messages

- Messages are organized into named topics

  - A node can publish messages on a topic

  - Another node that wants to receive the topic messages can subscribe to that topic

- The ROS master takes care of linking publishers and subscribers,
  but the messages are sent directly from publisher to subscriber

Services

Service calls: an alternative way of communicating with nodes

- Bi-directional
  - One node sends information to another node (e.g. requesting information)
  - The other node responds (e.g. with the required information)
  - In contrast, when a message is published, there is no concept of a response, and no guarantee that there is even a node subscribing to topic and receiving the messages
- One-to-one
  - Each service call is initiated by one node and the response goes back to it
  - In contrast, topics and message may have many publishers and many subscribers

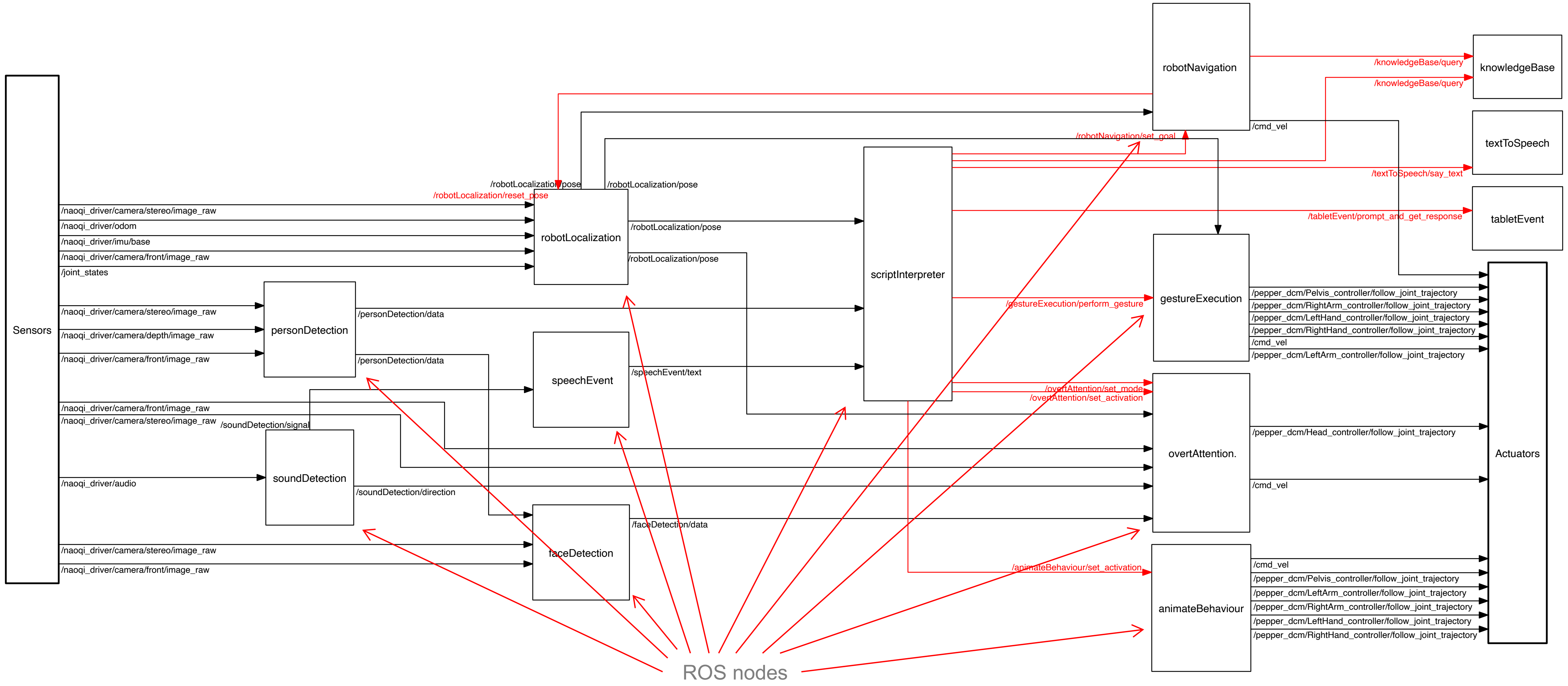
Services

Terminology

- Client node sends some data called a request to a server node
  - Waits for a reply
- Server node receives the request
  - Takes some action
  - Sends some data called a response back to the client
- The content of the request and the response is determined by the service data type
  - Similar to the message type associated with a topic
  - Two parts (and possibly two different types): request and response

The CSSR4Africa System Architecture

http://www.cssr4africa.org/



Sensors
robotNavigation
knowledgeBase
textToSpeech
tabletEvent
robotLocalization
scriptInterpreter
gestureExecution
personDetection
speechEvent
soundDetection
overtAttention.
faceDetection
animateBehaviour
Actuators
ROS nodes
/knowledgeBase/query
/knowledgeBase/query
/cmd_vel
/robotNavigation/set_goal
/textToSpeech/say_text
/tabletEvent/prompt_and_get_response
/robotLocalization/pose
/robotLocalization/reset_pose
/naoqi_driver/camera/stereo/image_raw
/naoqi_driver/odom
/naoqi_driver/imu/base
/naoqi_driver/camera/front/image_raw
/joint_states
/naoqi_driver/camera/depth/image_raw
/personDetection/data
/speechEvent/text
/soundDetection/signal
/naoqi_driver/audio
/soundDetection/direction
/faceDetection/data
/gestureExecution/perform_gesture
/overtAttention/set_mode
/overtAttention/set_activation
/animateBehaviour/set_activation
/pepper_dcm/Pelvis_controller/follow_joint_trajectory
/pepper_dcm/RightArm_controller/follow_joint_trajectory
/pepper_dcm/LeftHand_controller/follow_joint_trajectory
/pepper_dcm/RightHand_controller/follow_joint_trajectory
/pepper_dcm/LeftArm_controller/follow_joint_trajectory
/pepper_dcm/Head_controller/follow_joint_trajectory

ROS Resources

Wiki	http://wiki.ros.org/
Installation	http://wiki.ros.org/ROS/Installation
Tutorials	http://wiki.ros.org/ROS/Tutorials
Tutorial Videos	http://www.youtube.com/playlist?list=PLDC89965A56E6A8D6
ROS Cheat Sheet	http://www.vernon.eu/RPP/ROS_Cheatsheet.pdf

Recommended Reading

http://wiki.ros.org/catkin/Tutorials/create_a_workspace

http://wiki.ros.org/ROS/Tutorials/CreatingPackage

http://wiki.ros.org/roscpp/Overview/InitializationandShutdown

http://wiki.ros.org/roscpp/Overview/NodeHandles

http://wiki.ros.org/ROS/Tutorials/BuildingPackages

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(c++)

J. M. O'Kane, A Gentle Introduction to ROS, 2014.
https://cse.sc.edu/~jokane/agitr/

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

$\bigg\}$ Principles

A Brief Review of Robot Programming

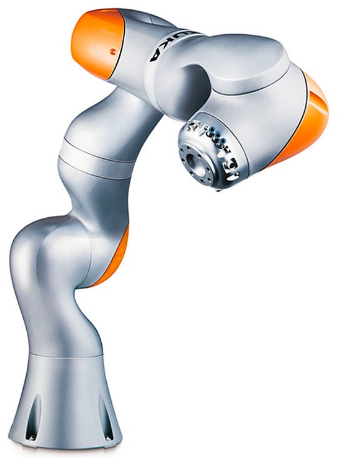
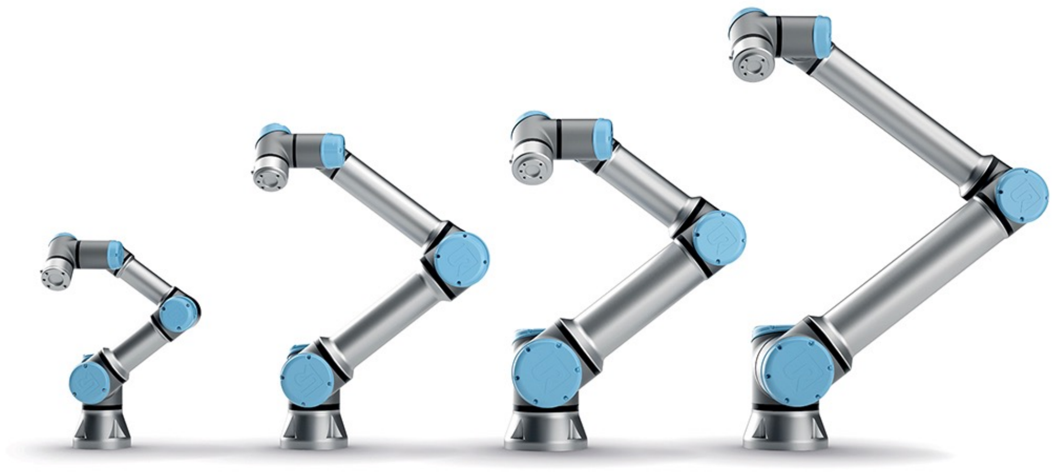
There are, broadly speaking, three main categories of robot programming system which are, in order of the level of sophistication

- Guiding Systems
- Robot-Level or Explicit-Level Systems and
- Task Level Systems

- Guiding systems are typified by the manual lead-through approach in which the manipulator is trained by guiding the arm through the appropriate positions using, for example, a teach-pendant and recording the individual joint positions

- Task execution is effected by driving the joints to these recorded positions

- This type of manual teaching is the most common of all programming systems

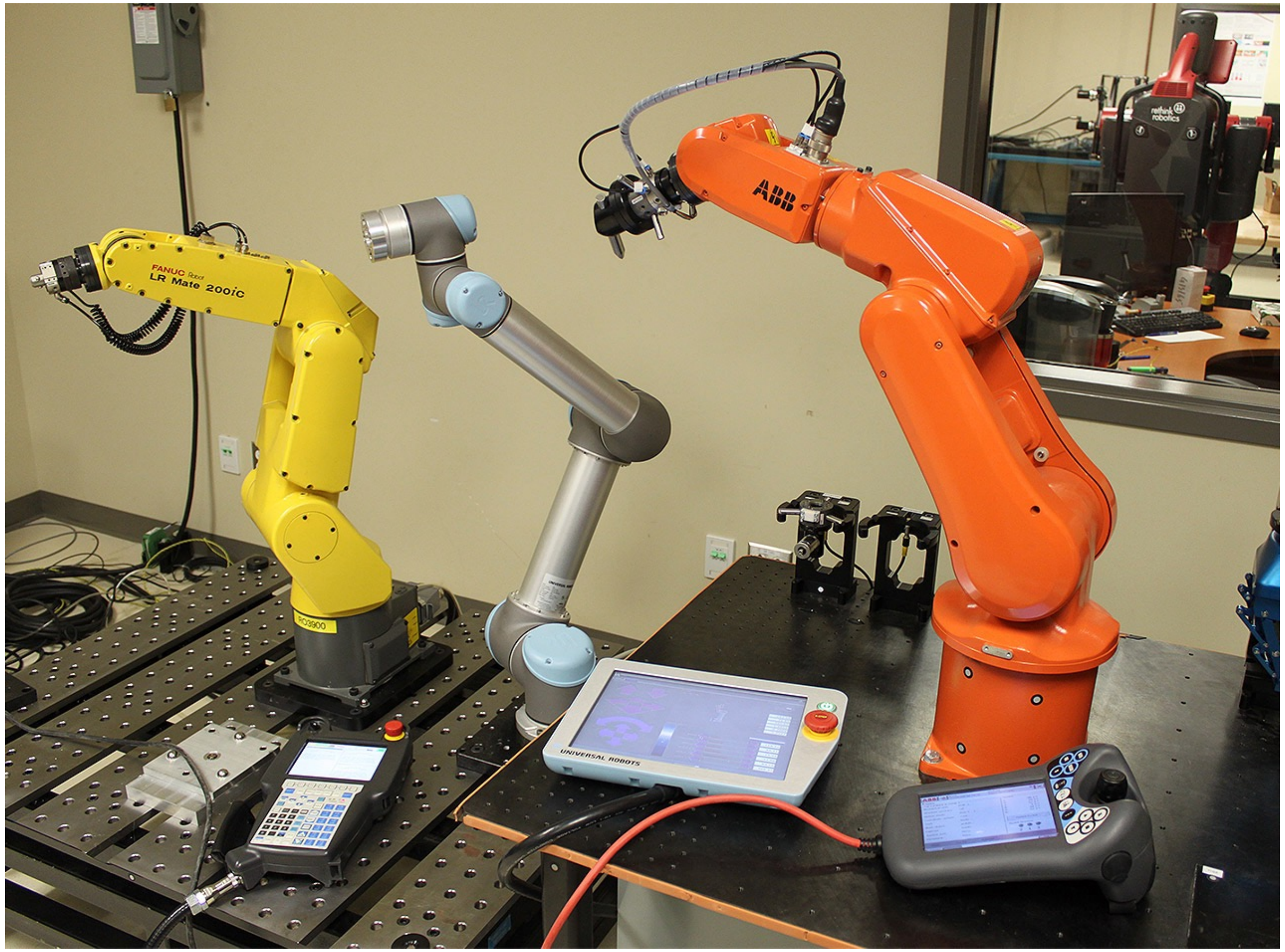



KUKA LBR iiwa

https://robots.ieee.org/robots/lbriiwa

Universal Arms

https://robots.ieee.org/robots/lbriiwa



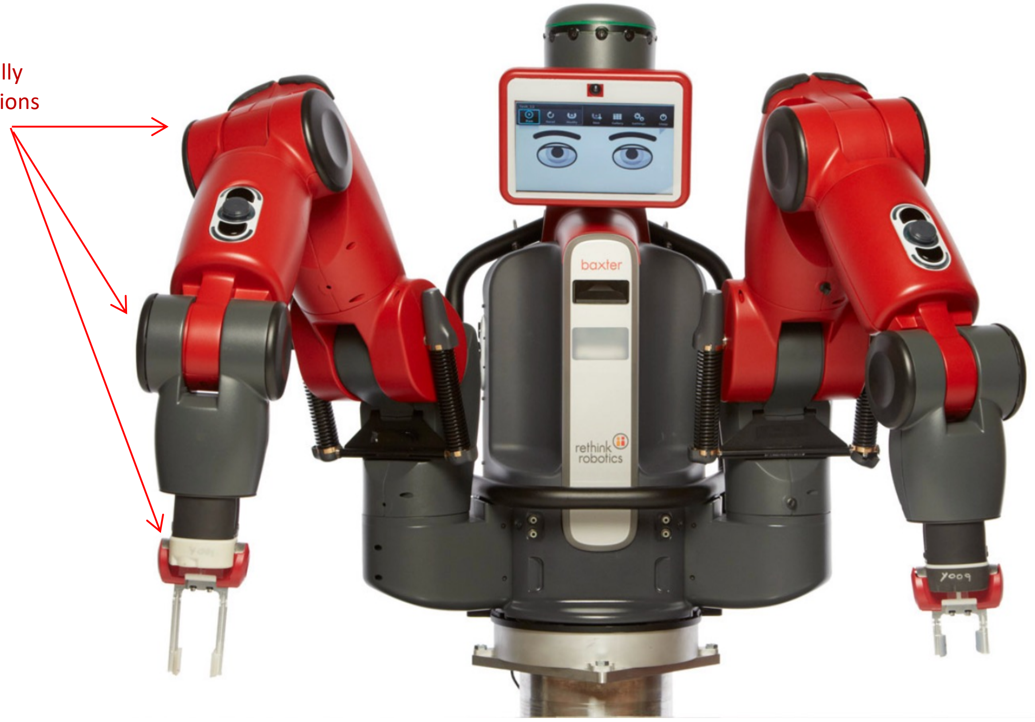
FANUC LR Mate 200iC, Universal Robots UR5, ABB IRB 120

https://robohub.org/what-is-so-special-about-the-robot-arms-of-universal-robots/

Actively-compliant arms:
They move in response to an externally applied force

This allows the operator to guide the robot by physically placing the arm at the required positions and orientations

Sometimes referred to as a co-bot:
a robot that is safe to work with in close proximity



Baxter

Baxter is a versatile manufacturing robot. Its cameras and force-sensing actuators let it adapt to changes in the environment, and a user can program a new task simply by moving its arms around.

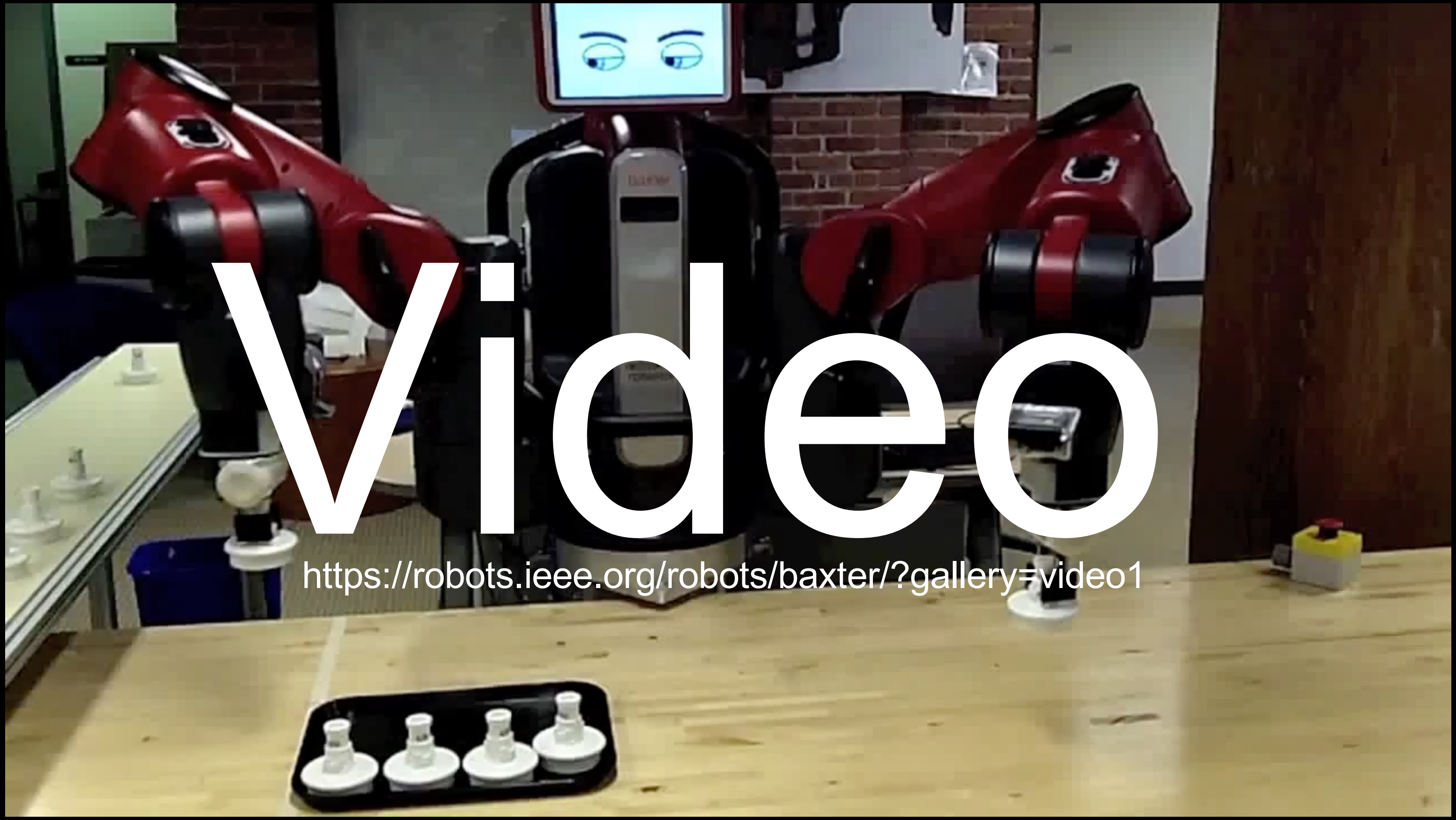
CREATOR
Rethink Robotics

COUNTRY
United States 🇺🇸

YEAR
2012

TYPE
Industrial

Source: https://robots.ieee.org/robots/baxter/


Video

https://robots.ieee.org/robots/baxter/?gallery=video1

Source: https://robots.ieee.org/robots/baxter/

Robot-level programming systems, for the most part, simply replace the teach pendant with a robot programming language

Manipulator movements are still programmed by explicitly specifying joint positions

However, several languages also facilitate robot control in a three-dimensional Cartesian space, rather than in the joint space

- (Forward) Kinematic Solution of the manipulator arm

  Allows you to compute the pose (position and orientation) of the end-effector in a 3D Cartesian frame of reference, given the manipulator joint positions

- Inverse Kinematic Solution

  Allows you to compute the joint positions for a given position and orientation of the end-effector

- The more advanced of these languages incorporate structured programming control constructs.

- They make extensive use of coordinate transformations and coordinate frames

With this approach

- The robot control is defined in terms of transformations on a coordinate frame (a set of XYZ axes) associated with, and embedded in, the robot hand

- Off-line programming is more feasible as long as the transformations representing the relationships between the frames describing the objects in the robot environment are accurate

Task-level robot programming languages attempt to describe assembly tasks as sequences of goal spatial relationships between objects

- they focus on the objects rather than on the manipulator joints
- the robot is merely a mechanism to achieve these goals
- they typically require the use of task planning, path planning, collision avoidance and world-modelling

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

$\Big\}$ Principles

- Robot manipulation is concerned, in essence, with the spatial relationships between several objects, between objects and manipulators, and with the reorganization of these relationships
- We use homogeneous transformations and vectors & quaternions to represent these spatial relationships
- We begin by introducing homogeneous transformations showing how they can be used to represent coordinate frames of reference

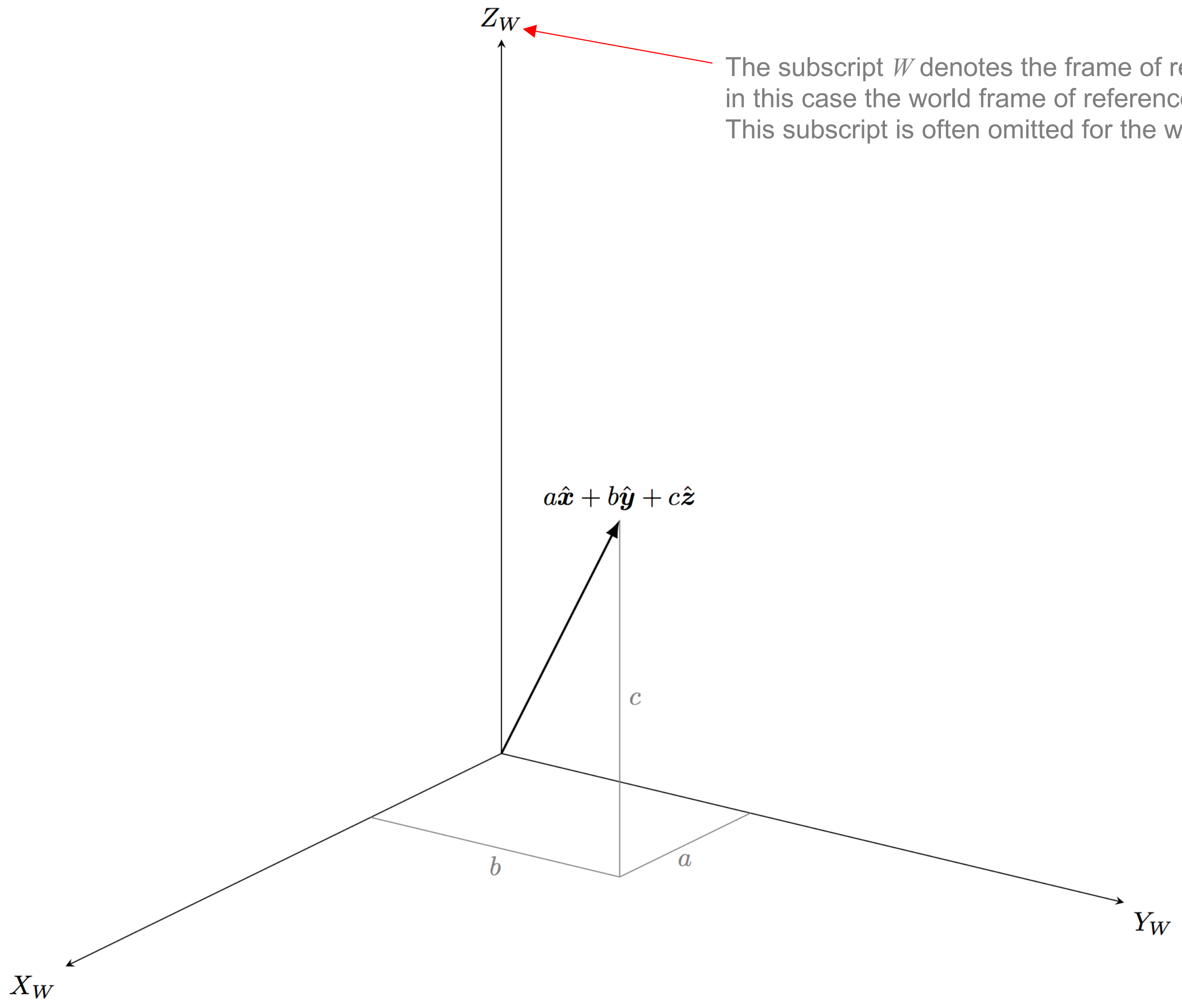
A 3D vector $\boldsymbol{v} = a\hat{\boldsymbol{x}} + b\hat{\boldsymbol{y}} + c\hat{\boldsymbol{z}}$, where $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$ are unit vectors along the X, Y, and Z axes are represented in homogeneous coordinates as

Note the use of the tilde to denote a homogeneous representation of a vector

$$\tilde{\boldsymbol{v}} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Scaling factor, thus a single 3D vector can be represented by several homogeneous coordinates

where $a = \frac{x}{w}$, $b = \frac{y}{w}$, and $c = \frac{z}{w}$


Z_W
$a\hat{\boldsymbol{x}} + b\hat{\boldsymbol{y}} + c\hat{\boldsymbol{z}}$
c
b
a
X_W
Y_W

The subscript W denotes the frame of reference of which this is an axis, in this case the world frame of reference.
This subscript is often omitted for the world frame of reference.

For example, $\boldsymbol{v} = 3\hat{\boldsymbol{x}} + 4\hat{\boldsymbol{y}} + 5\hat{\boldsymbol{z}}$ can be represented by $\begin{bmatrix} 3 \\ 4 \\ 5 \\ 1 \end{bmatrix}$ or $\begin{bmatrix} 6 \\ 8 \\ 10 \\ 2 \end{bmatrix}$

Since division by zero is indeterminate, the vector $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ is undefined

A general transformation $\boldsymbol{H}$, in 3D space, representing translation, rotation, stretching and perspective distortions, is a 4×4 matrix in homogeneous formulation

Given a point represented by the vector $\tilde{\boldsymbol{u}}$, its transformation $\tilde{\boldsymbol{v}}$ is represented by the matrix product

$$\tilde{\boldsymbol{v}} = \boldsymbol{H}\tilde{\boldsymbol{u}}$$

The transformation H corresponding to a translation by a vector $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ is

$$\boldsymbol{H} = \boldsymbol{Trans}(a, b, c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For example : to transform $\tilde{\boldsymbol{u}} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$ by $\boldsymbol{H}$

$$\tilde{\boldsymbol{v}} = \boldsymbol{H}\tilde{\boldsymbol{u}} = \boldsymbol{Trans}(a,b,c)\tilde{\boldsymbol{u}} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x + aw \\ y + bw \\ z + cw \\ w \end{bmatrix} = \begin{bmatrix} x/w + a \\ y/w + b \\ z/w + c \\ 1 \end{bmatrix}$$

The transformations corresponding to rotations about X, Y and Z axes by an angle θ are:

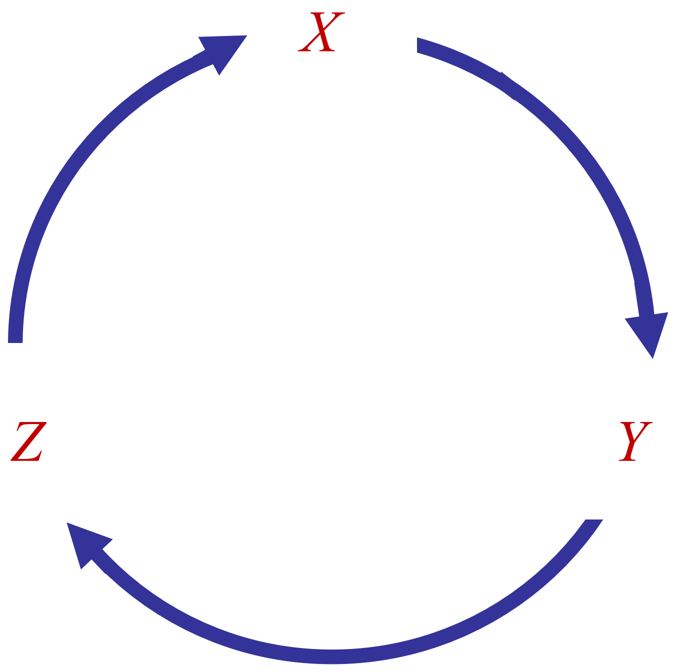
$$\boldsymbol{Rot}(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Rot}(Y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Rot}(Z,\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Remember when deciding in which sense to make a rotation that:

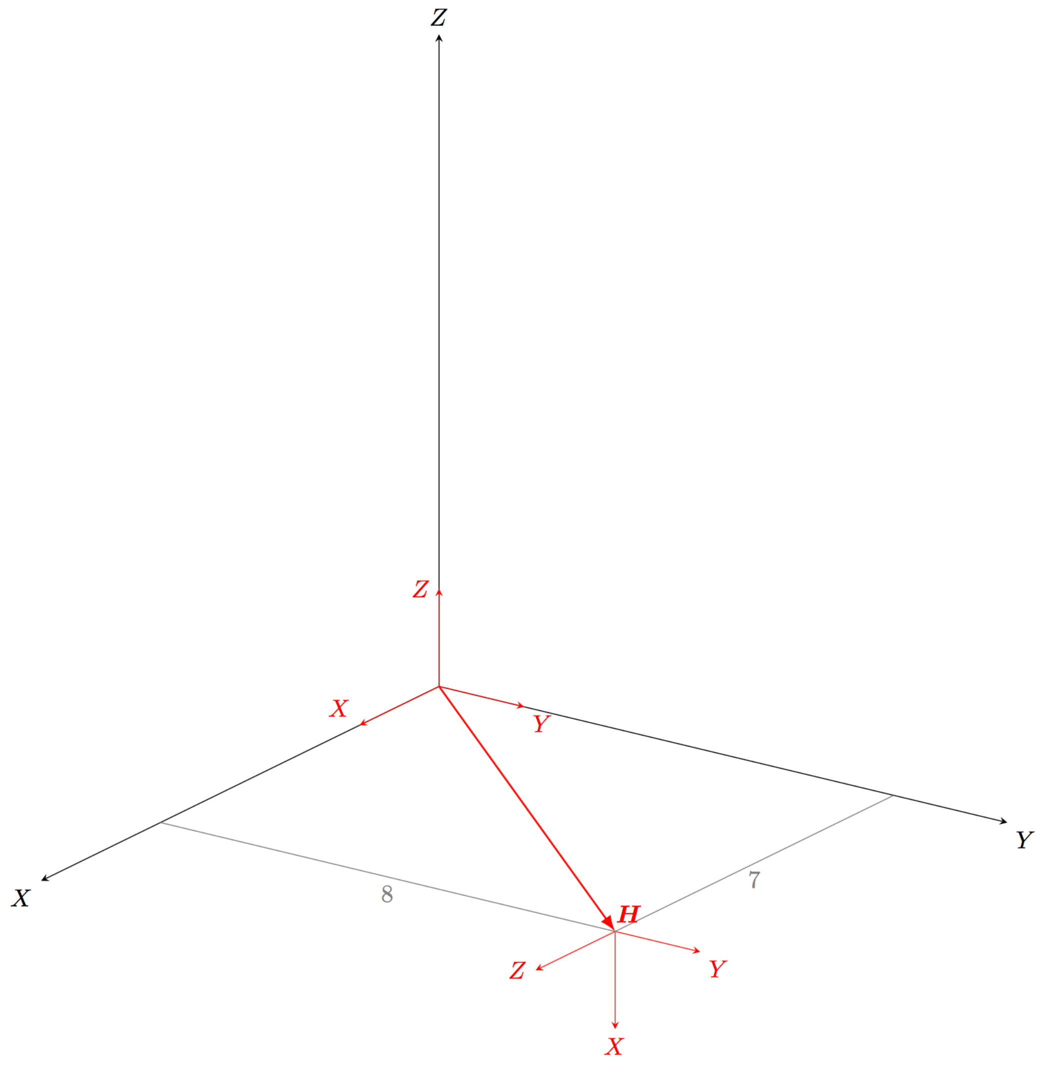
- a positive rotation about the X axis takes the Y axis toward the Z axis
- a positive rotation about the Y axis takes the Z axis toward the X axis
- a positive rotation about the Z axis takes the X axis toward the Y axis



We can interpret the homogeneous transformation as a coordinate reference frame

In particular, a homogeneous transformation describes the position and orientation of a coordinate frame with respect to another previously defined coordinate frame

Thus, the homogeneous transformation represents, not only transformations of vectors (points), but also positions and orientations



$$\boldsymbol{H} = \boldsymbol{Trans}(7, 8, 0)\boldsymbol{Rot}(Y, 90)$$

Interpreting a homogeneous transformation as a coordinate frame

Specifically, a coordinate frame is defined by four things: the position of its origin and the direction of its X, Y and Z axes.

- the first three columns of the homogeneous transformation represent the direction of the X, Y and Z axes of the coordinate frame with respect to the base coordinate reference frame

- the fourth column represents the position of the origin

- A homogeneous transformation, which can be a combination of many simpler homogeneous transformations, applies equally to other homogeneous transformations as it does to vectors

- Thus, we can take a coordinate reference frame and move it elsewhere by applying an appropriate homogeneous transformation

If the coordinate frame to be “moved” is originally aligned with the so-called base coordinate reference frame

the homogeneous transformation is

- a description of how to transform the base coordinate frame to the new coordinate frame and …
- a description of this new coordinate frame with respect to the base coordinate reference frame.

The rotations and translations we have been describing have all been made relative to the fixed base frame of reference

Thus, in the transformation given by

$$\boldsymbol{H} = \boldsymbol{Trans}(7, 8, 0)\boldsymbol{Rot}(Y, 90)$$

1. First, the frame is first rotated by 90° around the Y axis of the base frame of reference

2. Then translated by $7\hat{\boldsymbol{x}} + 8\hat{\boldsymbol{y}} + 0\hat{\boldsymbol{z}}$ in the base frame of reference

This operation may also be interpreted in reverse order, from left to right, *viz.*

1. First, the object (frame) is first translated by $7\hat{\boldsymbol{x}} + 8\hat{\boldsymbol{y}} + 0\hat{\boldsymbol{z}}$

2. Then rotated by 90° around the station frame Y axis (i.e. the translated frame of reference)

- This second interpretation is more intuitive since we can forget about the base reference frame and just remember "where we are": our current station coordinate reference frame

- We then just need to decide what transformations are necessary to get us to where we want to be based on the orientation of the station axes

In this way, we can get from pose to pose by incrementally identifying the appropriate station transformations,

$$\boldsymbol{H_1}, \boldsymbol{H_2}, \boldsymbol{H_3}, \ldots \boldsymbol{H_n}$$

which we apply sequentially, as we go, and the final pose is defined with respect to the base simply as

$$\boldsymbol{H} = \boldsymbol{H_1} \boldsymbol{H_2} \boldsymbol{H_3} \ldots \boldsymbol{H_n}$$

Sometimes we include an explicit composition operator, e.g. Corke (2017

$$\boldsymbol{H} = \boldsymbol{H_1} \oplus \boldsymbol{H_2} \oplus \boldsymbol{H_3} \oplus \ldots \boldsymbol{H_n}$$

In order to clarify the relative nature of these transformations

- Each of these frames/transformations is normally written with a leading superscript
- This superscript identifies the coordinate frame with respect to which the (new) frame/transformation is defined
- If the leading superscript is omitted, it is assumed to be the base (or world) frame

$$H = H_1\, {}^{H_1}H_2\, {}^{H_2}H_3 \;\ldots\; {}^{H_{n-1}}H_n$$

$$\boldsymbol{H = H_1\, {}^{H_1}H_2\, {}^{H_2}H_3 \ldots {}^{H_{n-1}}H_n}$$

or $\boldsymbol{H = H_1 \oplus {}^{H_1}H_2 \oplus {}^{H_2}H_3 \oplus \ldots {}^{H_{n-1}}H_n}$

As a general rule:

- If we post-multiply a transform representing a frame by a second transformation describing a rotation and/or translation we make that rotation/transformation **with respect to the frame axis described by the first transformation**

- If we pre-multiply the frame transformation representing a rotation/transformation then the rotation/transformation is made **with respect to the base reference coordinate frame**

At this stage, we have developed a system where we can

specify the position and orientation of coordinate reference frames anywhere

with respect to each other ← w.r.t. station frame of reference

or

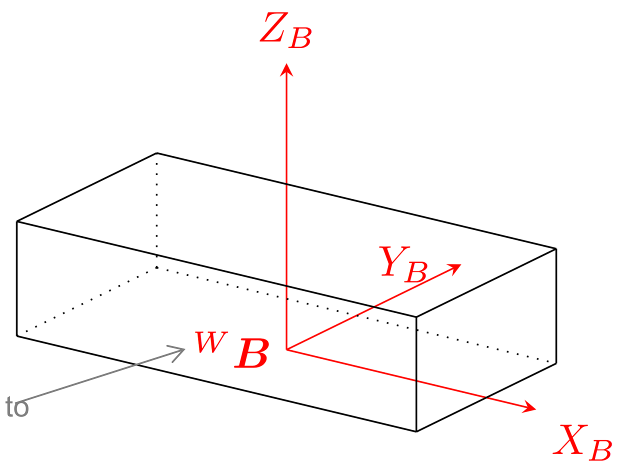
with respect to a given base frame ← w.r.t. fixed world frame of reference

This, in itself, is not much use since the world you and I know does not have too many coordinate reference frames in it

What we really require is a way of identifying the pose of objects

Position and Orientation:
Six degrees of freedom

The trick, and it is no more than a trick, is to attach a coordinate frame to an object, *i.e.* symbolically glue an XYZ frame into an object simply by defining it to be there

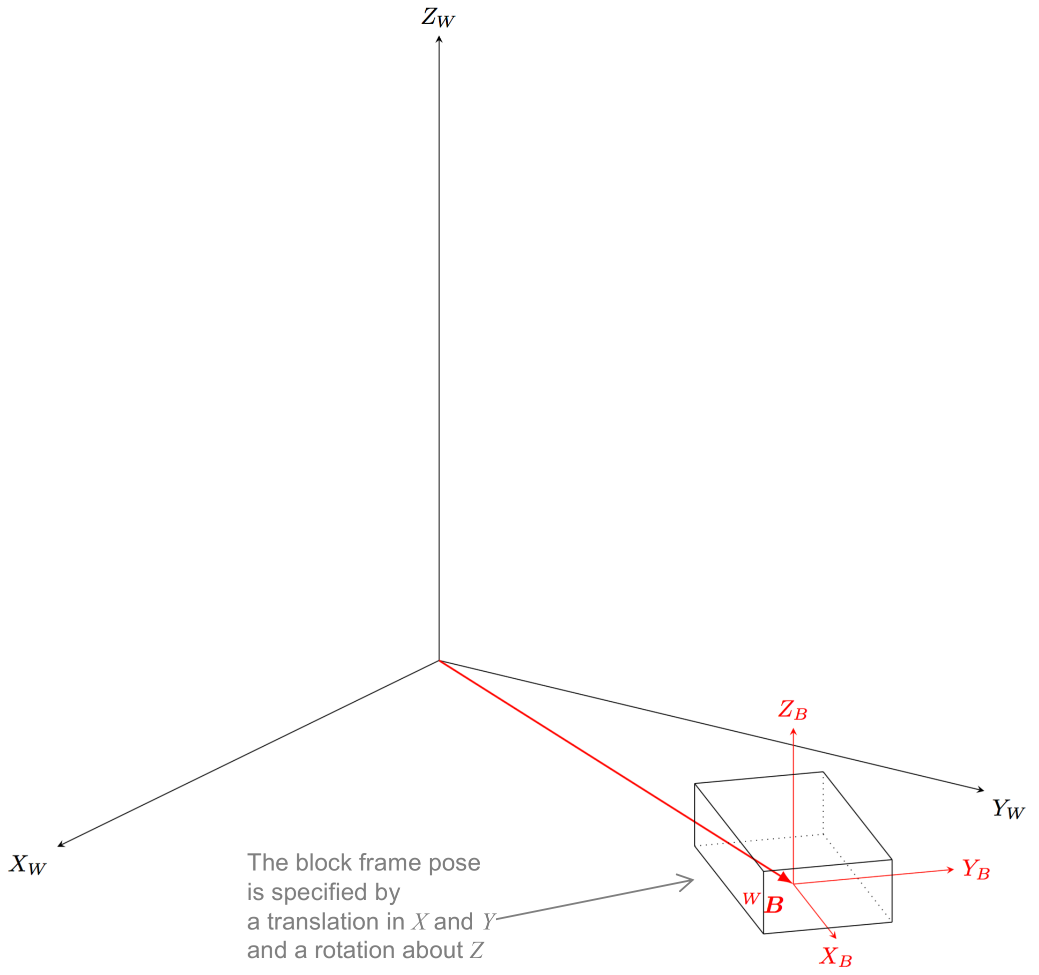


The block frame is defined with respect to the world frame of reference

- As we rotate and translate the coordinate frame, so we rotate and translate objects

- We can arbitrarily position and orient a coordinate frame – and an object – by specifying the required translations and rotations

- Thus, we specify the pose of an object by specifying its associated coordinate frame (homogeneous transformation)

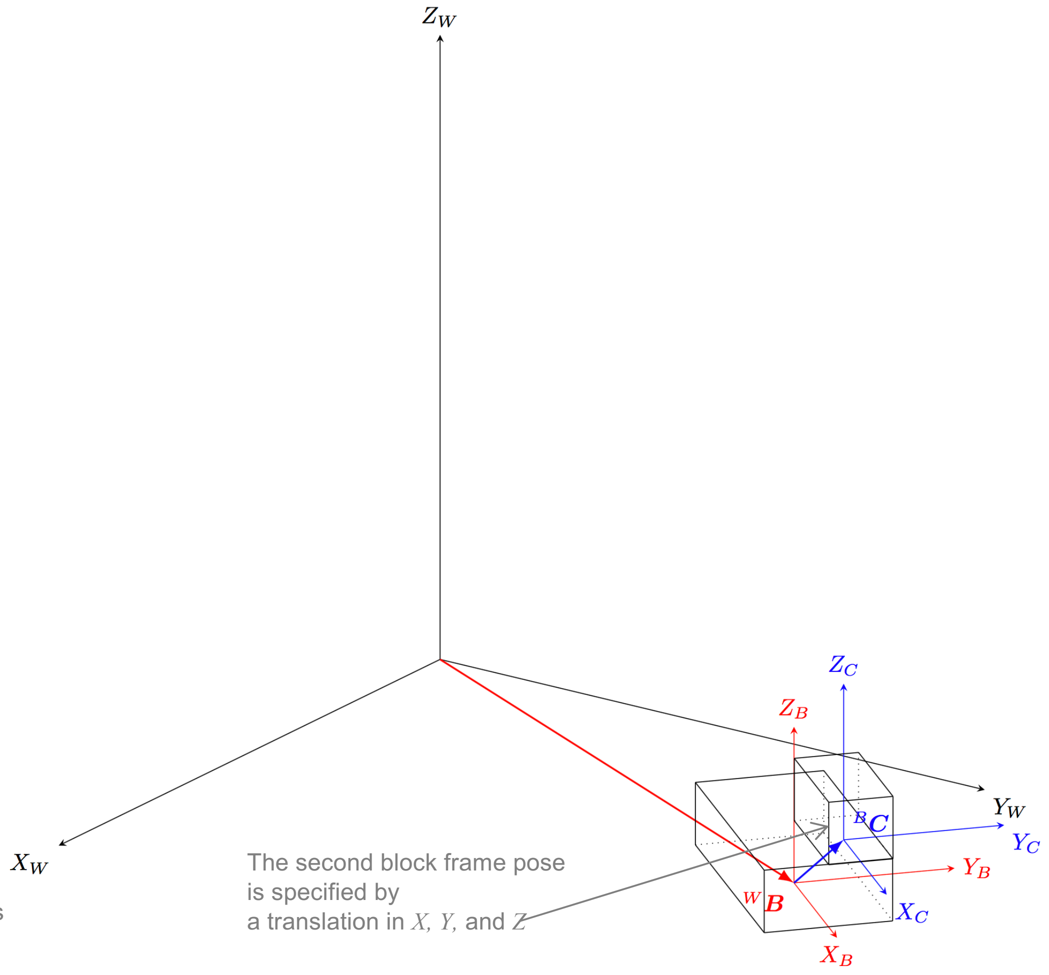

Z_W
Y_W
X_W
Z_B
Y_B
X_B
${}^{W}\boldsymbol{B}$
The block frame pose is specified by a translation in X and Y and a rotation about Z

- We can arbitrarily position and orient one object, i.e. its pose, with respect to another object

- How? By specifying the required translations and rotations of its associated coordinate frame (homogeneous transformation)

  e.g., ${}^{B}C = \mathrm{Trans}(x, y, z)$

These values represent translations along the X_B, Y_B, Z_B axes; the values of the translations depend on the dimensions of the objects



The second block frame pose is specified by a translation in X, Y, and Z

Specifying Pose in ROS

- We will use homogeneous transformations to specify a frame of reference, for end-effector and object pose

- ROS uses a different (but entirely equivalent) approach

  - Specify the origin of the frame as a 3-D vector

  - Specify the orientation of the frame as a quaternion: a single rotation about some (appropriate) Euler axis

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

$\left.\right\}$ Principles

Robot Programming by Task Specification

By defining a series of manipulator end-effector positions *Mn,* a task can be described as a sequence of manipulator movements to these defined positions

For example, a task to pick and place an object might be formulated as follows

$M0$:	Move out of the field of view of the camera
	Determine the pose of a object and a suitable grasp point (possibly using a camera)
$M1$:	Move to an approach position close to the grasp point
$M2$:	Move to the grasp position
	Grasp the object
$M3$:	Move to the depart position above the grasp point
$M4$:	Move to the approach position in above the destination position
$M5$:	Move to the destination position
	Release the object
$M6$:	Move to the depart position away from the destination position

- We are specifying the task in terms of movements of the robot but the object are what we are really interested in

- The object movements are implicit in the fact that the manipulator has grasped it

- We make up for this when we describe the structure of the task by considering the structure of the task's component objects:

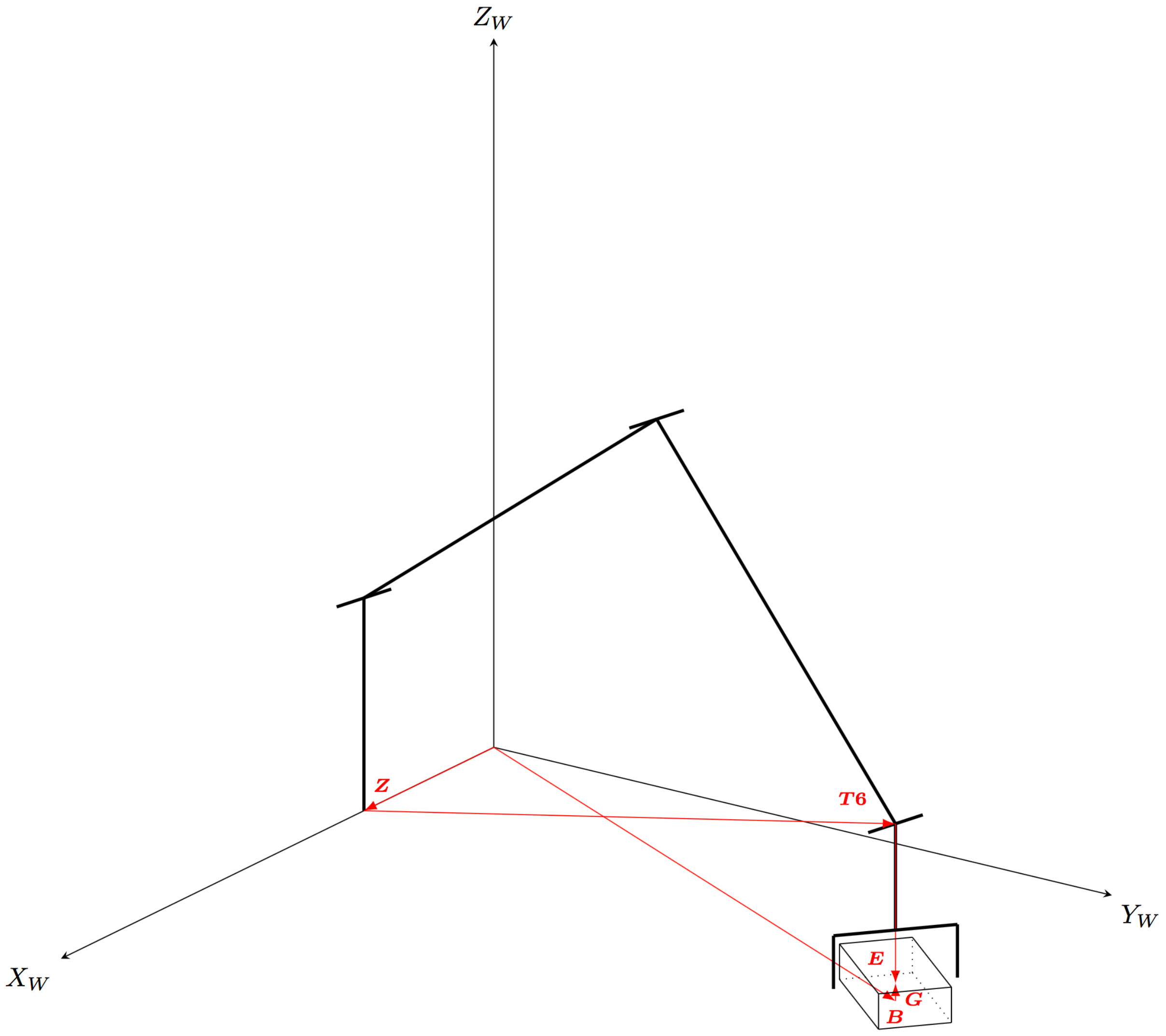
  - the manipulator
  - the end-effector
  - the object being manipulated
  - the object grasp pose

- We will use the explicit positional relationships between these objects to describe the task structure

Since coordinate frames can be used to describe object position and orientation …

And since we may need to describe a coordinate frame in two or more ways (there is more than one way to reach any given position and orientation) …

We use transform equations to relate the two descriptions



A manipulator grasping a block

$\boldsymbol{Z}$	is the transformation (frame) which describes the position of manipulator with respect to the base co-ordinate reference frame
$^{Z}\boldsymbol{T6}$	describes the end of the manipulator (i.e., the wrist) with respect to the base of manipulator, i.e., with respect to $\boldsymbol{Z}$
$^{T6}\boldsymbol{E}$	describes the end-effector with respect to the end of the manipulator, i.e., with respect to $\boldsymbol{T6}$
$\boldsymbol{B}$	describes a block's position with respect to the base coordinate reference frame
$^{B}\boldsymbol{G}$	describes the manipulator end-effector with respect to the block, i.e., with respect to $\boldsymbol{B}$.

In this example, the end-effector is described in two ways, by the transformations leading from the base to the wrist to the end-effector:

$$\boldsymbol{Z} * {}^{Z}\boldsymbol{T6} * {}^{T6}\boldsymbol{E}$$

and by the transformations leading from the block to the end-effector grip position:

$$\boldsymbol{B} * {}^{B}\boldsymbol{G}$$

Equating these descriptions, we get the following transformation equation:

$$\boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ {}^{T6}\boldsymbol{E} = \boldsymbol{B}\ {}^{B}\boldsymbol{G}$$

Alternatively, including the explicit composition operator in Corke (2016)

$$\boldsymbol{Z} \oplus {}^{Z}\boldsymbol{T6} \oplus {}^{T6}\boldsymbol{E} = \boldsymbol{B} \oplus {}^{B}\boldsymbol{G}$$

- Solving for $\boldsymbol{T6}$ by multiplying across by the inverse of $\boldsymbol{Z}$ and ${}^{T6}\boldsymbol{E}$

$$ {}^{Z}\mathbf{T6} = \boldsymbol{Z}^{-1} \;\; \boldsymbol{B} \;\; {}^{B}\boldsymbol{G} \;\; {}^{\mathbf{T6}}\boldsymbol{E}^{-1} $$

- $\boldsymbol{T6}$ is a function of the joint variables of the manipulator and, if known, then the appropriate joint variables can be computed using the inverse kinematic solution

- $\boldsymbol{T6}$ then is the coordinate frame which we wish to program in order to effect the manipulation task
- An arm position and orientation specified by $\boldsymbol{T6}$ is, thus, equivalent to our previous informal movement $\boldsymbol{Mn}$

$$\text{Move } \boldsymbol{Mn} = \text{Move } {}^{Z}\boldsymbol{T6}$$

  - since we can compute $\boldsymbol{T6}$ in terms of our known frame we now have an arm movement which is specified in terms of the frames which describe the task structure

- Assigning the appropriate value to $\boldsymbol{T6}$ and moving to that position, implicitly using the inverse kinematic solution

$$^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{B}\ {}^{B}\boldsymbol{G}\ {}^{T6}\boldsymbol{E}^{-1}$$

$$\text{Move}\ {}^{Z}\boldsymbol{T6}$$

- What we have not yet done is to fully specify each of these frames by embedding them in the appropriate objects and specifying the transformations which define them

- Note that the position of the end-effector with respect to the base reference system is represented by

$$\boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ {}^{T6}\boldsymbol{E}$$

- This allows you to generate general-purpose and reusable robot programs

- In particular, the calibration of the manipulator to the workstation is represented by $\boldsymbol{Z}$, while if the task is to be performed with a change of tool, only $\boldsymbol{E}$ need be altered

- As we have seen, we specify the orientation of ***T6*** by solving for it in terms of other frames/transformations in the task specification …

- We do this by

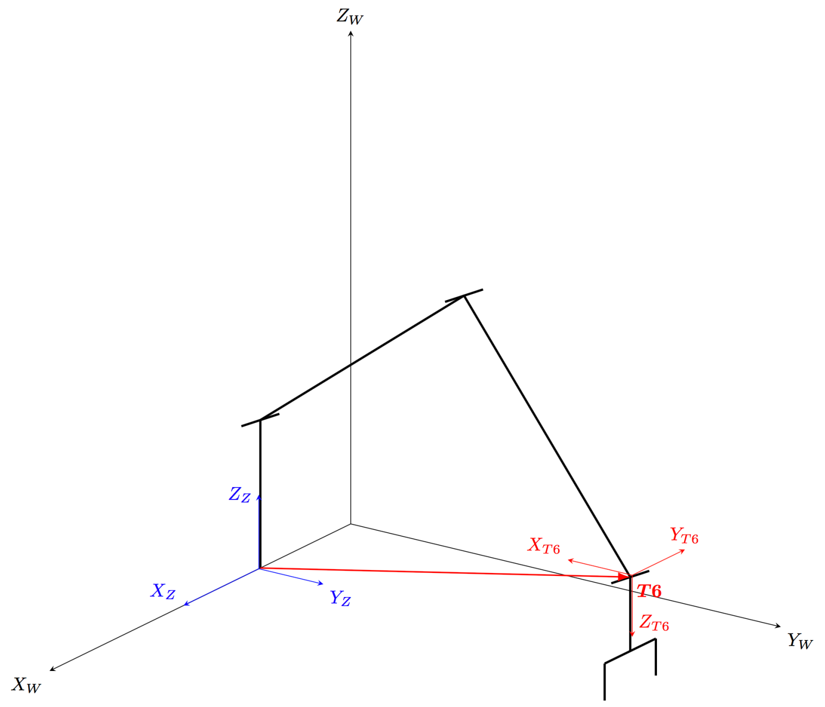
1. Embedding a frame in an object (or a desired point in space)

2. Specifying the position of the origin of the frame by applying a translation

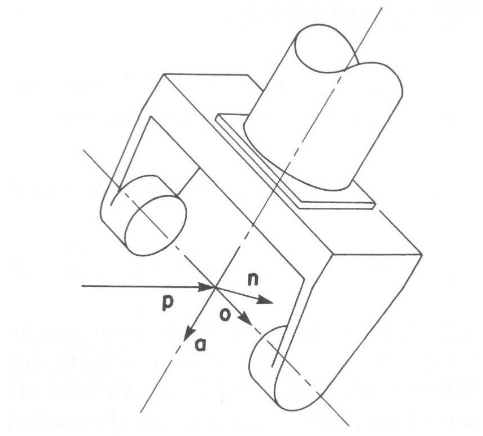
3. Specifying the orientation of the frame by applying one or more rotations

There is a convention that the ***T6*** frame should be embedded in the manipulator

- with the origin at the wrist
- with the Z axis directed outward from the wrist to the gripper
- with the Y axis directed in the plane of movement of the gripper when it is opening and closing
- with the X axis making up a right-hand system



The same convention applies to the $\boldsymbol{E}$ frame that is embedded in a two-finger gripper (end-effector … hence $\boldsymbol{E}$)

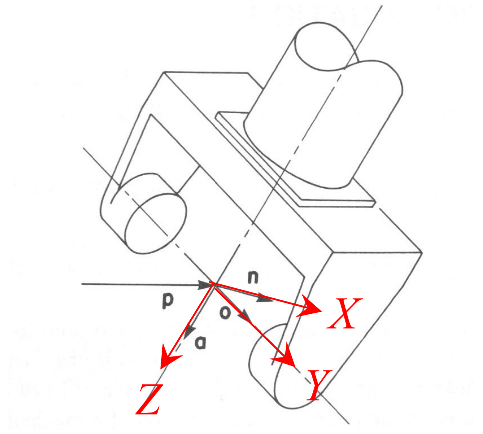


(Paul, 1981)

$$\boldsymbol{E} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

n	Normal
o	Orientation
a	Approach

The same convention applies to the $\boldsymbol{E}$ frame that is embedded in a two-finger gripper (end-effector … hence $\boldsymbol{E}$)



(Paul, 1981)

Direction of X axis

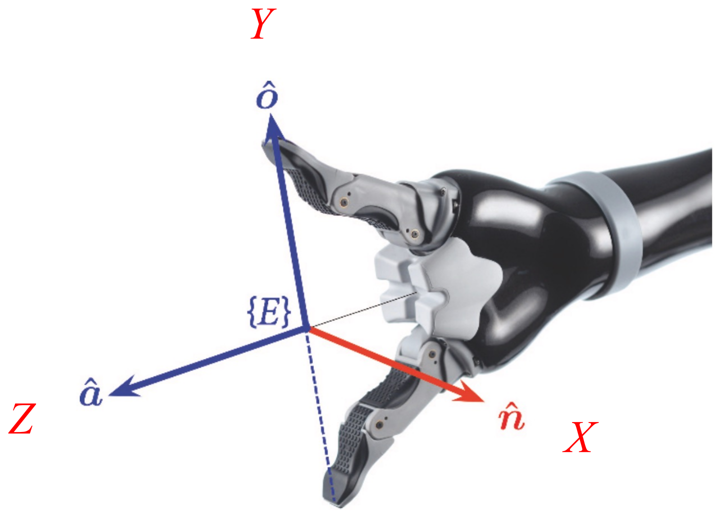
Direction of Y axis

Direction of Z axis

$$E = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

n	Normal
o	Orientation
a	Approach

The same convention applies to the ***E*** frame that is embedded in a two-finger gripper (end-effector … hence ***E***)



(Corke, 2017), p. 41

$$E = \begin{array}{c} \text{Direction of } X \text{ axis} \quad \text{Direction of } Y \text{ axis} \quad \text{Direction of } Z \text{ axis} \\ \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

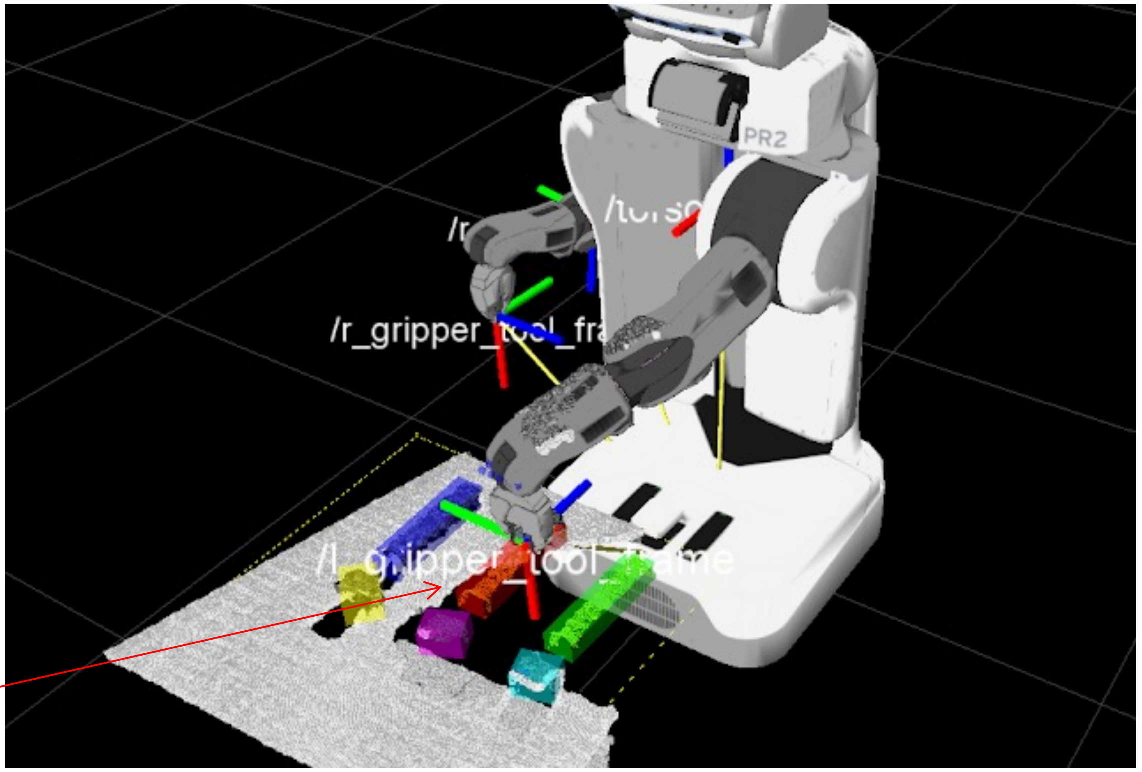
n	Normal
o	Orientation
a	Approach

ROS uses a different convention

"If the end effector is a grasping device, the frame should be located at the recommended object grasping location. The frame orientation is defined as X the axis going 'toward' the object. Y the main dimension in which the grasping device moves and Z orthogonal to X and Y axes."

https://www.ros.org/reps/rep-0120.html#l-gripper-and-r-gripper

This approach is consistent with the convention of embedding a frame in a vehicle, with the X axis aligned with the direction of travel; see conventions on specifying orientation using roll, pitch, and yaw in the following slides.

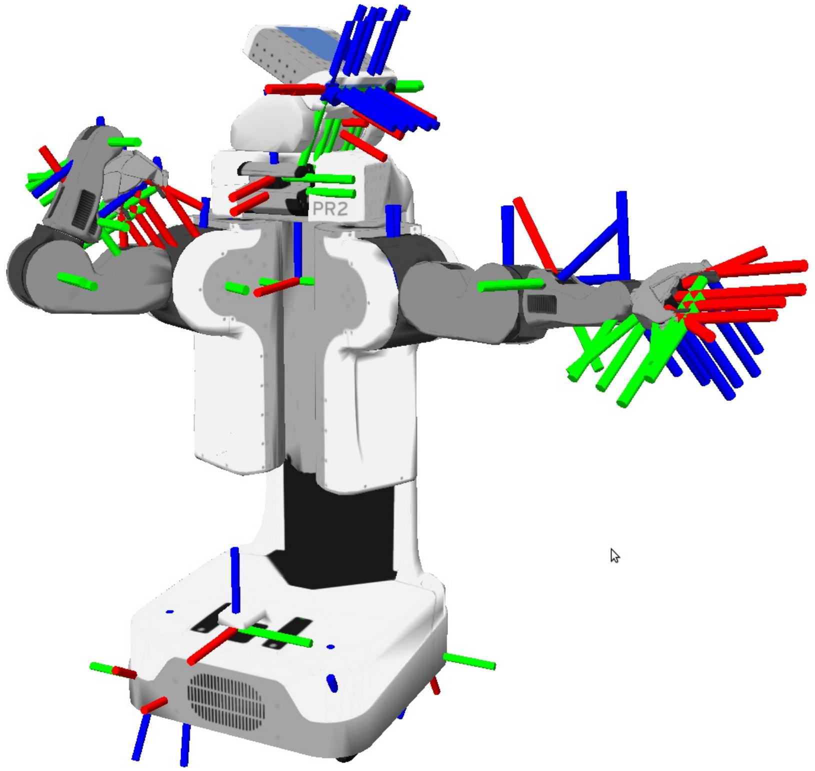


https://alliance.seas.upenn.edu/~meam620/wiki/index.php?n=IanMcMahon2011.Final

ROS uses a different convention

"If the end effector is a grasping device, the frame should be located at the recommended object grasping location. The frame orientation is defined as X the axis going 'toward' the object. Y the main dimension in which the grasping device moves and Z orthogonal to X and Y axes."

https://www.ros.org/reps/rep-0120.html#l-gripper-and-r-gripper

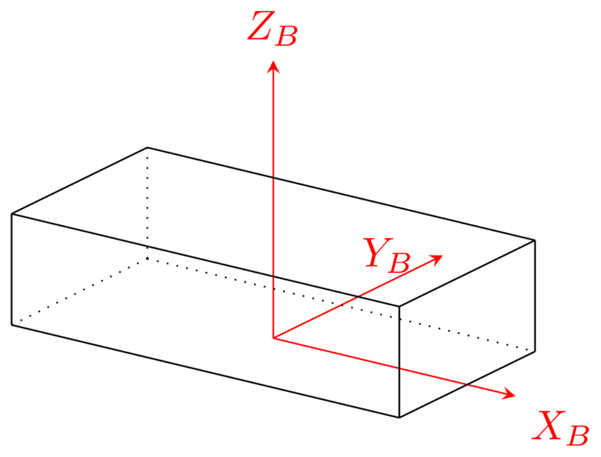


http://library.isr.ist.utl.pt/docs/roswiki/tf2.html

Specifying Pose

We have seen that the pose of an object can be specified by embedding a frame in the object in some appropriate manner … for example:

- Placing the origin at the centre of the object
- Aligning the axes with the major and minor axes of the object

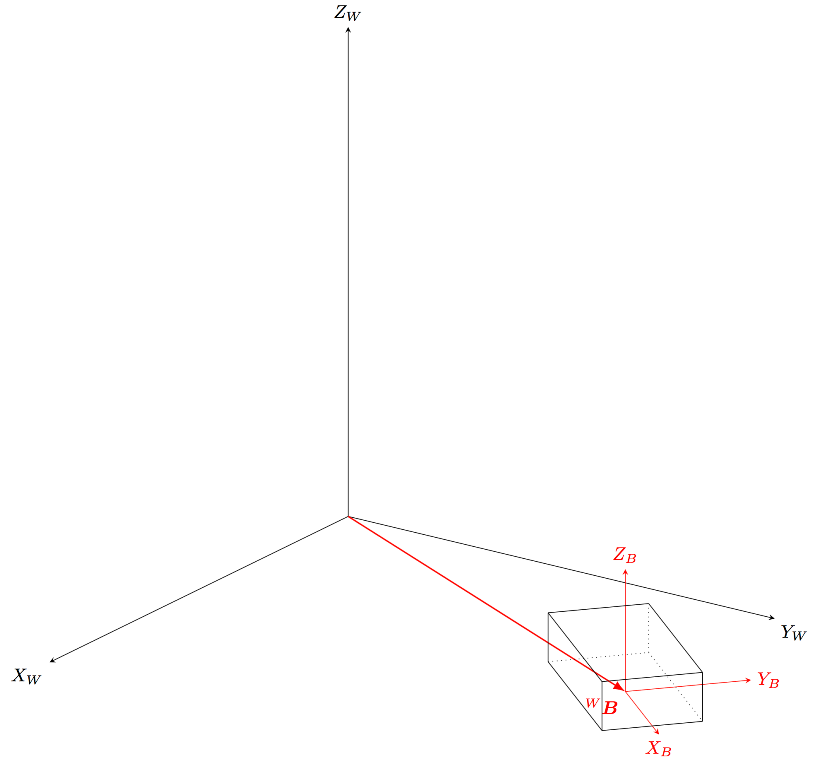


Specifying Pose

Then applying a homogenous transformation, e.g., $\boldsymbol{B} = \boldsymbol{Trans}\ (10, 20, 0)\ \boldsymbol{Rot}\ (Z, 50)$

- Translation part
  - Possibly several translations, applied in turn
- Rotation part
  - Possibly several rotations, applied in turn

You can specify them in whatever order you like, yielding a valid transform equation such as
$\boldsymbol{B} = \boldsymbol{Trans}\ (10, 20, 0)\ \boldsymbol{Rot}\ (Z, 50)\ \boldsymbol{Rot}\ (X, 10)\ \boldsymbol{Rot}\ (Z, 30)$



Specifying Orientation

That said, there are several conventions for the way these rotations are specified

One is Roll-Pitch-Yaw (RPY) … sometimes referred to as Cardan angles

RPY can be confusing. There are two reasons.

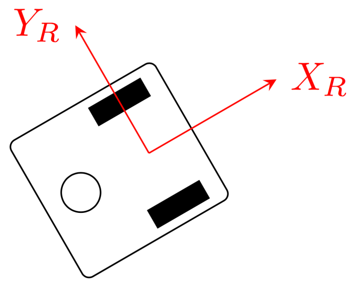
1. There are two conventions, each specifying a different sequence of axes about which to rotate:
    - $Z\,Y\,X$ normally used with vehicles
    - $X\,Y\,Z$ normally used with end-effectors
2. The angles are specified in the order yaw, pitch, roll (despite the name roll-pitch-yaw)

Specifying Orientation

That said, there are several conventions for the way these rotations are specified

Roll-Pitch-Yaw (RPY) with vehicles $Z\ Y\ X$

- The frame embedded in a vehicle normally has
  - X axis in the direction of travel
  - Z axis directly up
  - Y axis specified a right-hand system



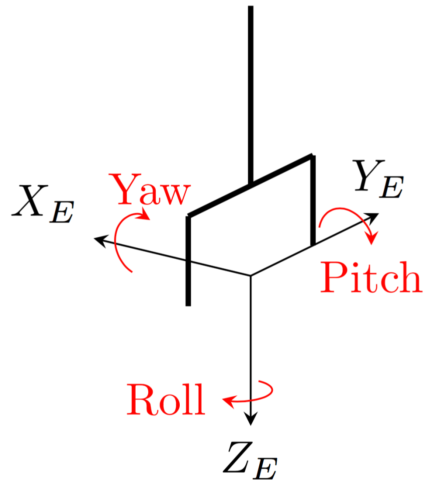
- The orientation is specified by $\boldsymbol{RPY}\,(\theta_y, \theta_p, \theta_r) = \boldsymbol{Rot}\,(Z, \theta_y)\ \boldsymbol{Rot}\,(Y, \theta_p)\ \boldsymbol{Rot}\,(X, \theta_r)$
  - First, rotate the yaw angle θ_y about the Z axis (i.e. about the vertical, thus specifying the direction of travel)
  - Second, rotate the pitch angle θ_p about the Y axis (thus specifying the angle of ascent or descent)
  - Third, rotate the roll angle θ_r about the X axis (thus specifying the banking angle)

Specifying Orientation

That said, there are several conventions for the way these rotations are specified

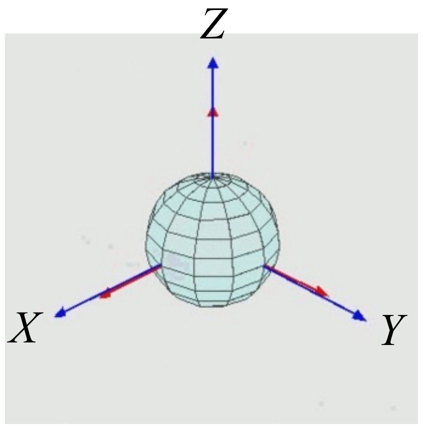
Roll-Pitch-Yaw (RPY) with end-effectors $X\ Y\ Z$

- The frame embedded in an end-effector or two-finger gripper normally has
  - X axis in the normal direction (i.e. normal to the movement of the fingers)
  - Z axis directed in the approach direction
  - Y axis direction in the orientation direction (i.e. parallel to the movement of the fingers)
- The orientation is specified by $\boldsymbol{RPY}(\theta_y, \theta_p, \theta_r) = \boldsymbol{Rot}(X, \theta_y)\, \boldsymbol{Rot}(Y, \theta_p)\, \boldsymbol{Rot}(Z, \theta_r)$
  - First, rotate the yaw angle θ_y about the X axis (i.e. about the normal)
  - Second, rotate the pitch angle θ_p about the Y axis (about the orientation)
  - Third, rotate the roll angle θ_r about the Z axis (about the approach)



Specifying Orientation
Euler Angles

- There are other commonly-used conventions for specifying the orientation of objects/frames
  - For example: Euler angles (e.g. rotation about Z, X, Z axes, in that order)



https://en.wikipedia.org/wiki/Euler_angles

  - Note that there are twelve Euler angle conventions; this is just one of them
- We also use quaternions, especially in ROS

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

$\left.\right\}$ Principles

A Simple Pick-and-Place Task Specification

$M0$:	Move out of the field of view of the camera
	Determine the pose of a block and a suitable grasp point (possibly using a camera)
$M1$:	Move to an approach position above the grasp point
$M2$:	Move to the grasp position
	Grasp the block
$M3$:	Move to the depart position above the grasp point
$M4$:	Move to the approach position in above the destination position
$M5$:	Move to the destination position
	Release the block
$M6$:	Move to the depart position above the block

- Again, we are specifying the task in terms of movements of the robot but the objects are what we are really interested in

- The object movements are implicit in the fact that the manipulator has grasped it

- We describe the structure of the task by considering the structure of the task's objects and related end-effector poses

  - the manipulator
  - the end-effector
  - the block
  - the block grasp position
  - the destination
  - the approach and depart positions

- We use the explicit positional relationships between these objects to describe the task structure

As before:

$\boldsymbol{Z}$	is the transformation which describes the position of manipulator with respect to the world coordinate reference frame.
${}^{Z}\boldsymbol{T6}$	describes the end of the manipulator (*i.e.* the wrist) with respect to the base of manipulator, *i.e.* with respect to $\boldsymbol{Z}$
${}^{T6}\boldsymbol{E}$	describes the end-effector with respect to the end of the manipulator, *i.e.,* with respect to $\boldsymbol{T6}$

We now define:

$\boldsymbol{B}$	the pose of the block, defined with respect to the base co-ordinate reference system
${}^{B}\boldsymbol{G}$	the pose of end-effector grasping the block, defined with respect to the block
${}^{G}\boldsymbol{A}$	the pose of end-effector approaching/departing grasp position, defined with respect to the grasp position
$\boldsymbol{D}$	the pose of the block destination, defined with respect to the base co-ordinate reference system
${}^{D}\boldsymbol{G}$	the pose of end-effector grasping the block, defined with respect to the block destination

If we were using a camera to identify the block pose,
we might also define

$\boldsymbol{OOV}$	the pose of the end-effector out of the field of view of the camera with respect to the base co-ordinate reference system

The manipulator movements $M0$ through $M6$ can now be expressed as combinations of these transformations

$M0 \quad \boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ \boldsymbol{E} = \boldsymbol{OOV}$

$M1 \quad \boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ \boldsymbol{E} = \boldsymbol{B}\ {}^{B}\boldsymbol{G}\ {}^{G}\boldsymbol{A}$

$M2 \quad \boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ \boldsymbol{E} = \boldsymbol{B}\ {}^{B}\boldsymbol{G}$

Grasp the block

$M3 \quad \boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ \boldsymbol{E} = \boldsymbol{B}\ {}^{B}\boldsymbol{G}\ {}^{G}\boldsymbol{A}$

$M4 \quad \boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ \boldsymbol{E} = \boldsymbol{D}\ {}^{D}\boldsymbol{G}\ {}^{G}\boldsymbol{A}$

$M5 \quad \boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ \boldsymbol{E} = \boldsymbol{D}\ {}^{D}\boldsymbol{G}$

Release the block

$M6 \quad \boldsymbol{Z}\ {}^{Z}\boldsymbol{T6}\ \boldsymbol{E} = \boldsymbol{D}\ {}^{D}\boldsymbol{G}\ {}^{G}\boldsymbol{A}$

We express these equations in terms of ${}^{Z}\boldsymbol{T6}$ because ${}^{Z}\boldsymbol{T6}$ specifies the robot pose and we pass ${}^{Z}\boldsymbol{T6}$ as an argument to the **move** function in the robot programming language

$M0$	${}^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{OOV}\ \boldsymbol{E}^{-1}$
$M1$	${}^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{B}\ {}^{B}\boldsymbol{G}\ {}^{G}\boldsymbol{A}\ \boldsymbol{E}^{-1}$
$M2$	${}^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{B}\ {}^{B}\boldsymbol{G}\ \boldsymbol{E}^{-1}$

Grasp the block

$M3$	${}^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{B}\ {}^{B}\boldsymbol{G}\ {}^{G}\boldsymbol{A}\ \boldsymbol{E}^{-1}$
$M4$	${}^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{B}\ {}^{D}\boldsymbol{G}\ {}^{G}\boldsymbol{A}\ \boldsymbol{E}^{-1}$
$M5$	${}^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{B}\ {}^{D}\boldsymbol{G}\ \boldsymbol{E}^{-1}$

Release the block

$M6$	${}^{Z}\boldsymbol{T6} = \boldsymbol{Z}^{-1}\ \boldsymbol{B}\ {}^{D}\boldsymbol{G}\ {}^{G}\boldsymbol{A}\ \boldsymbol{E}^{-1}$

Note that ${}^{G}\boldsymbol{A}$ is a translation transformation concerned with approaching and departing a particular object

Sometimes, in order to allow smooth approach and departure trajectories, these translation distances are iterated from zero to some maximum value or from some maximum value to zero (in integer intervals) depending on whether the effector is approaching or departing

For example:

${}^{G}\boldsymbol{A}$ is the approach position of the end-effector before grasping the block and is
(to be) defined as a translation, in the negative Z direction of the ${}^{B}\boldsymbol{G}$ frame, of the approach distance
$z\_approach$, say

Thus,

$$ {}^{G}\boldsymbol{A} = \boldsymbol{Trans}\ (0,\ 0,\ -(z\_approach)) $$

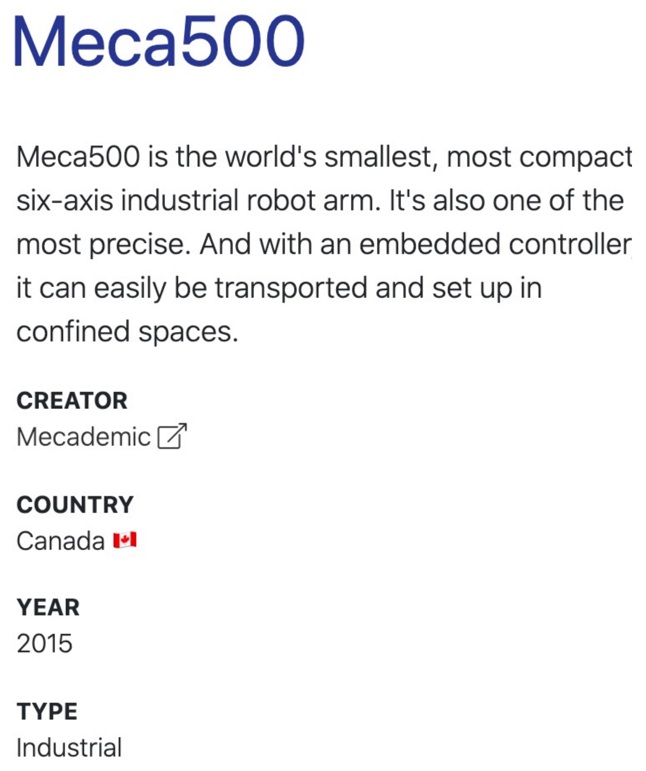
where:

$z\_approach = z\_approach\_initial$
$z\_approach\_initial - delta$
$z\_approach\_initial - 2*delta$
$:$
0

It should be noted well that this type of explicit point-to-point approximation of continuous path control would not normally be necessary with a commercial industrial robot programming language since they usually provide facilities for specifying the end-effector trajectory

Types of Robot

Industrial



Meca500

Meca500 is the world's smallest, most compact six-axis industrial robot arm. It's also one of the most precise. And with an embedded controller it can easily be transported and set up in confined spaces.

CREATOR
Mecademic

COUNTRY
Canada 🇨🇦

YEAR
2015

TYPE
Industrial

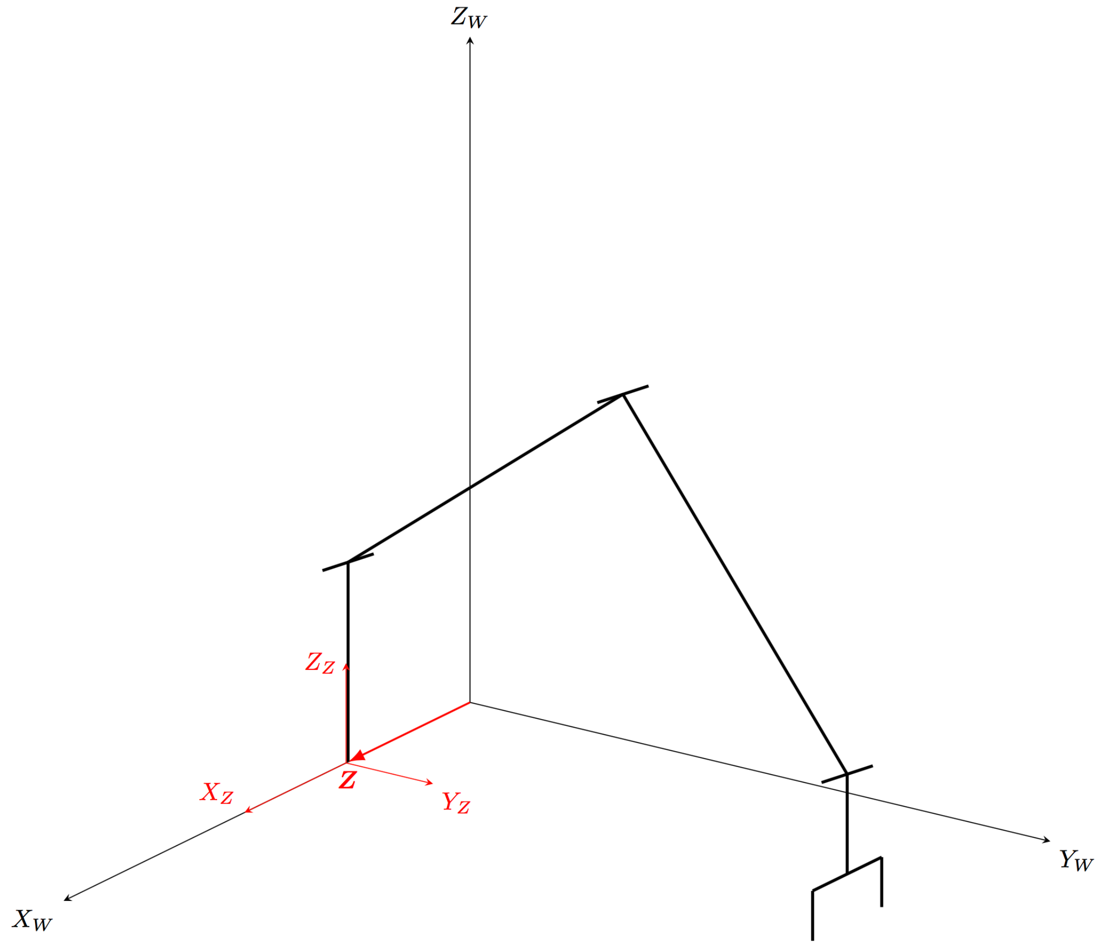
Source: https://robots.ieee.org/robots/meca/

- To complete the task specification, we now have to define the rotations and translations associated with these transformations/frames

- Some, e.g., $\boldsymbol{E}$, can be determined by empirical methods, embedding a frame in an object and measuring the object position and orientation

Others, $\boldsymbol{B}$ in particular, are defined here

but their components might be determined at run time, e.g., using a camera



Z The base of the manipulator

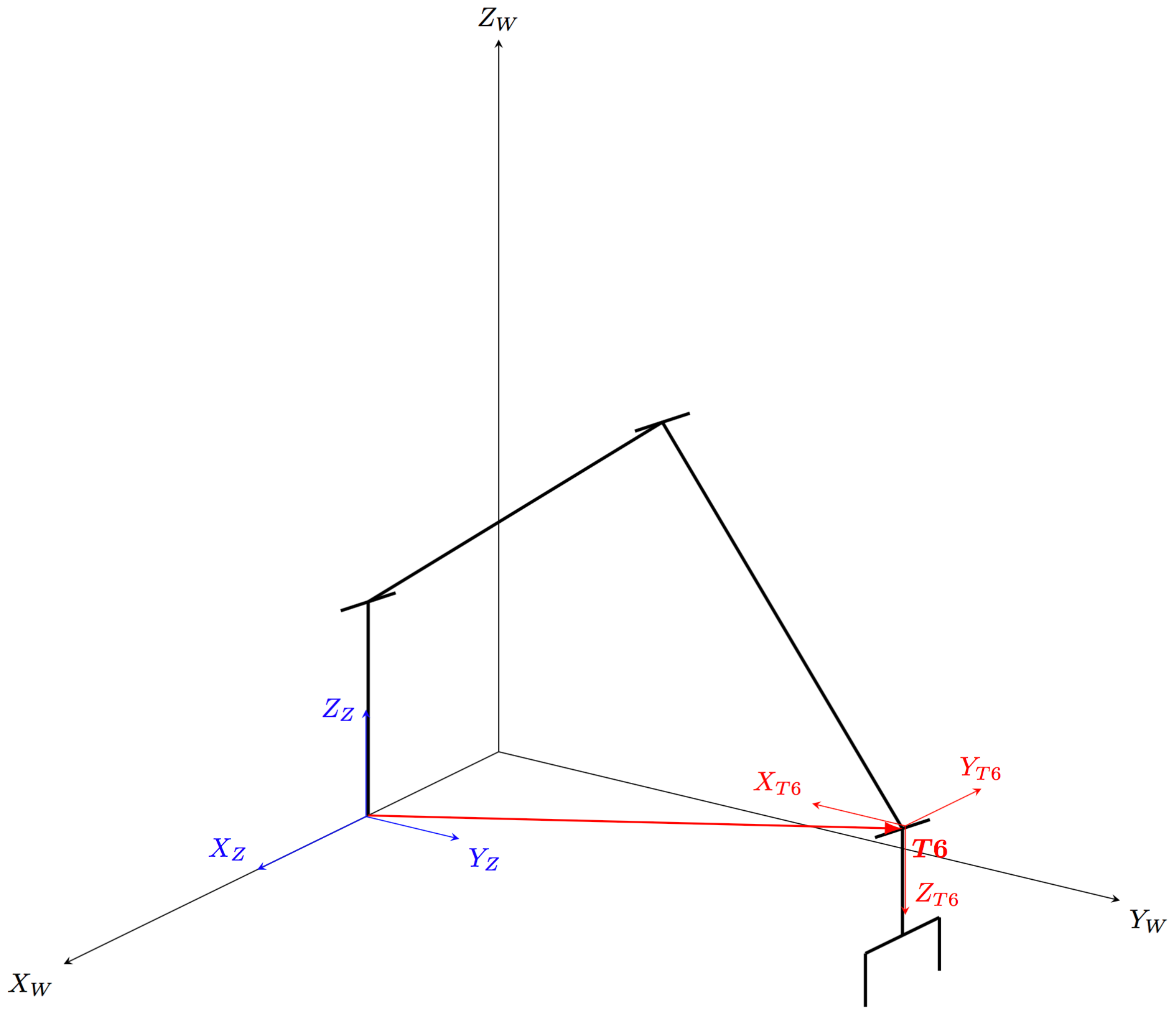
$\boldsymbol{Z}$ The pose of the position of manipulator with respect to the base co-ordinate reference frame

Later, we will assume that the base co-ordinate system is aligned with the frame embedded in the manipulator base

In that case,

$$\boldsymbol{Z} = \boldsymbol{I} = \textit{Identity Transformation}$$

Note that the frame defining the manipulator base is dependent on the kinematic model of the robot manipulator



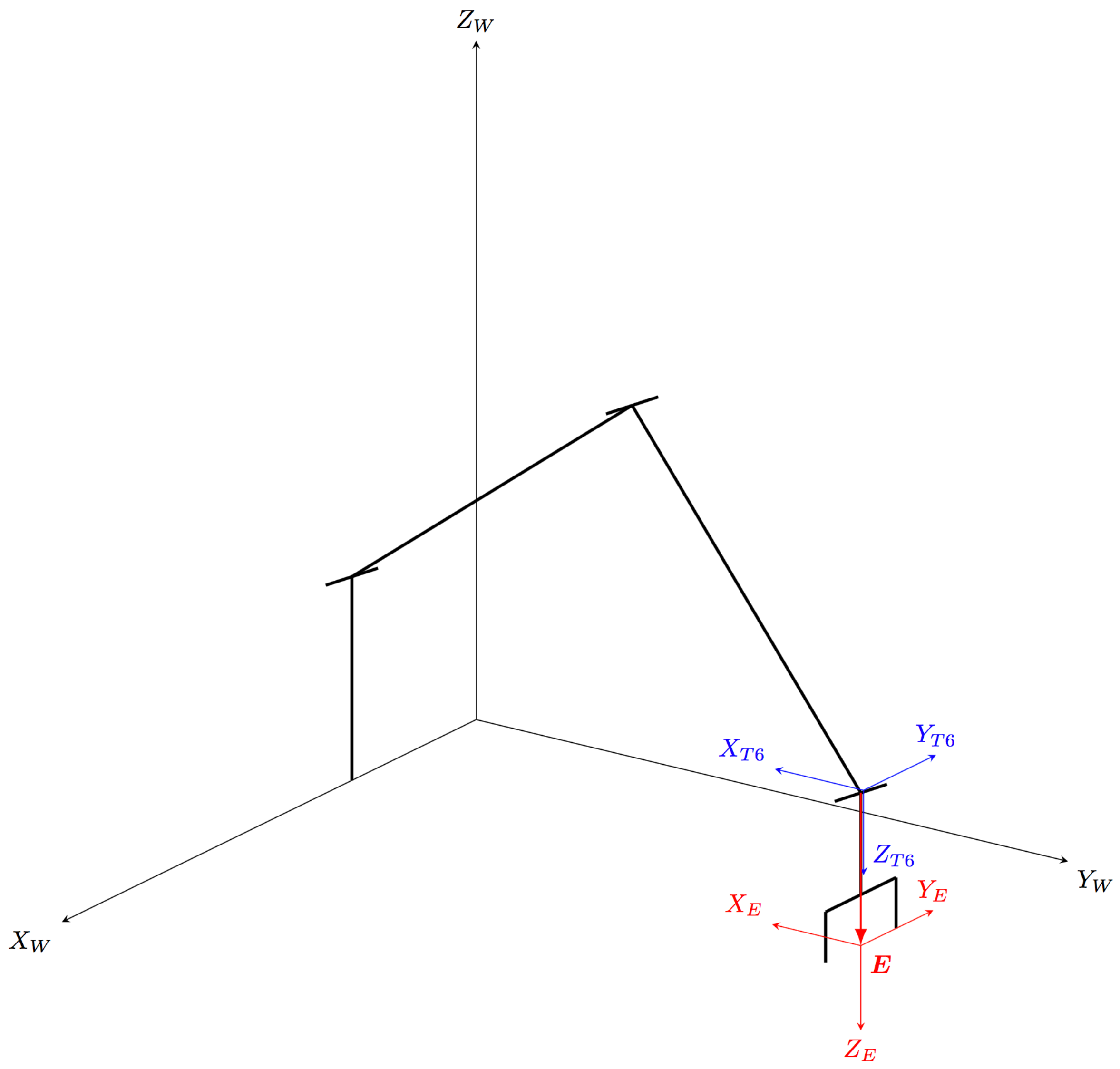
T6 The manipulator wrist

T6 The pose of the manipulator wrist

with respect to its base at ***Z***

The ***T6*** frame is a computable function of the other frames

Once we have computed the action-specific ***T6***,
we can then determine joint variables that correspond to this pose
using the inverse kinematic solution



$\boldsymbol{E}$ The manipulator wrist

E The pose of the end-effector with respect to the wrist, i.e. with respect to $T6$

The frame E representing is embedded in the tip of the effector

and hence is defined by a translation of 100 mm along the Z axis of the $T6$ frame

This will vary from end-effector to end-effector

$${}^{T6}\boldsymbol{E} = \boldsymbol{Trans}\ (0,\ 0,\ 100)$$

- As we have seen, we specify the orientation of ***T6*** by solving for it in terms of other frames/transformations in the task specification …

- So, let's now define each of these other frames

- We do this by

  1. Embedding a frame in an object (or a desired point in space)

  2. Specifying the position of the origin of the frame by applying a translation

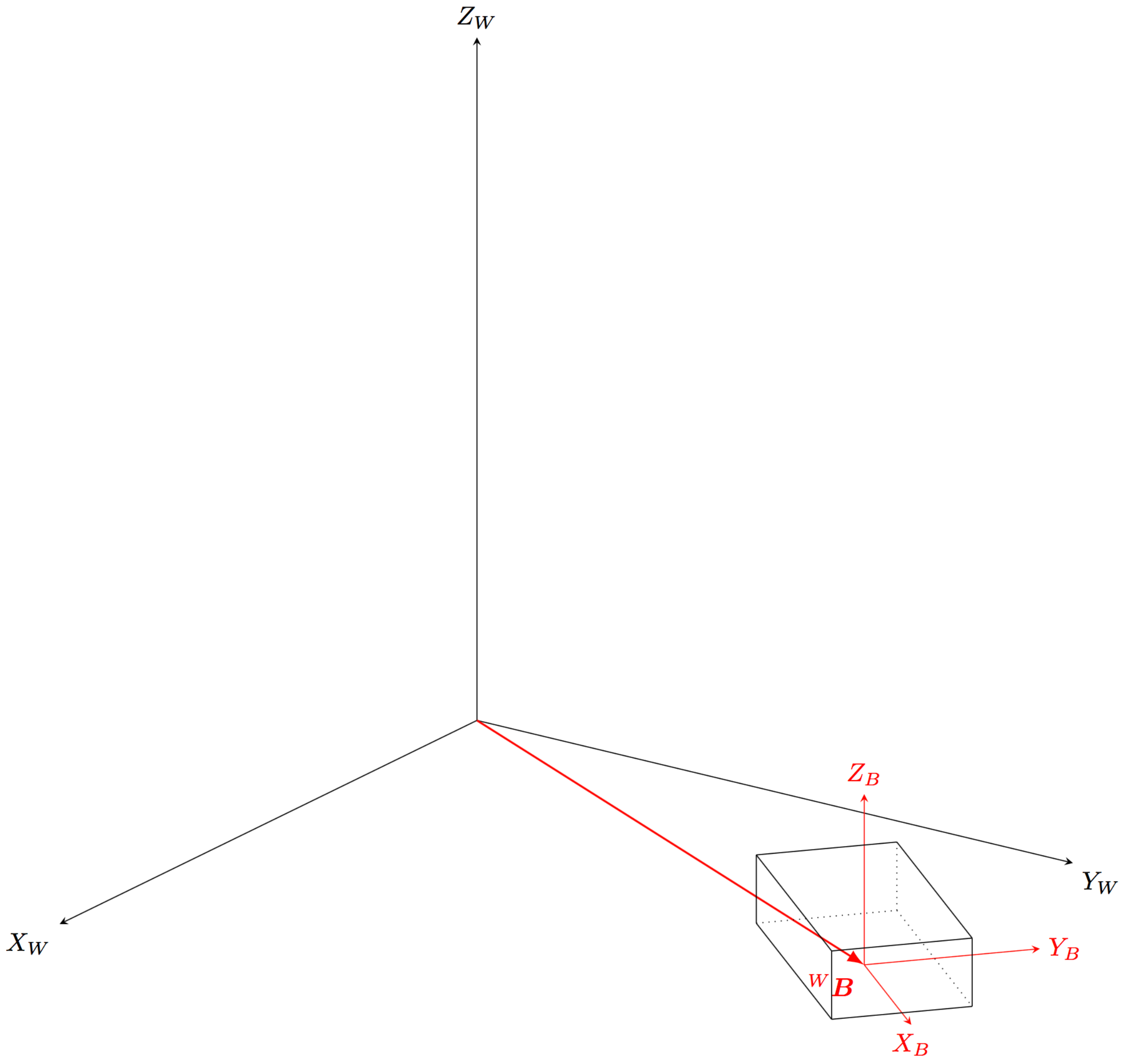
  3. Specifying the orientation of the frame by applying one or more rotations

- As noted previously, there are several commonly-used conventions for specifying the orientation of objects

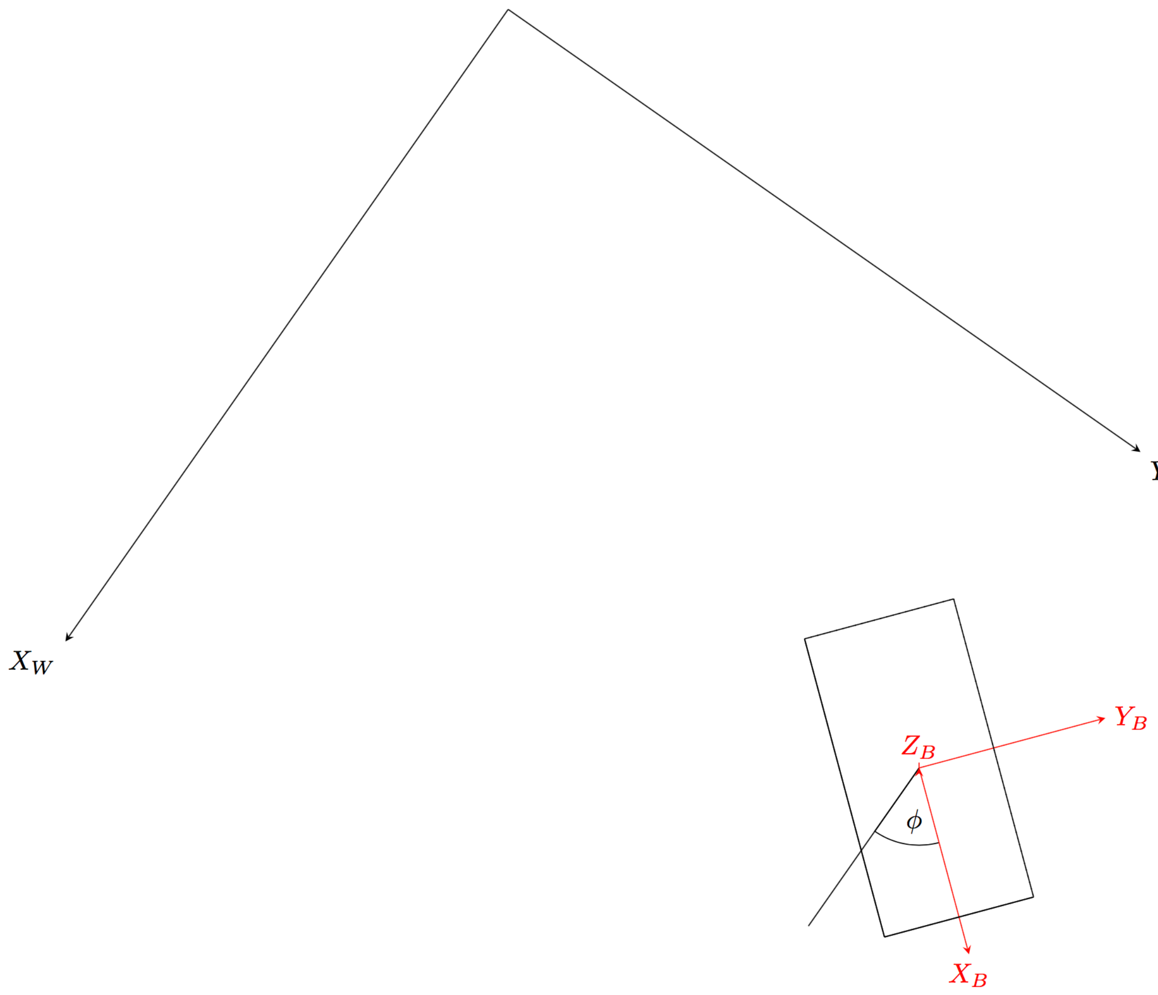
- One convention is roll-pitch-yaw

- This convention identifies three rotations about the station (local) co-ordinate frame embedded in the object which are applied in turn and in a specified order

  – a yaw of θ_y degrees about the station X axis
  – a pitch of θ_p degrees about the station Y axis
  – a roll of θ_r degrees about the station Z axis
  – … in that order



$\boldsymbol{B}$ The pose of the block



$\boldsymbol{B}$ The pose of the block

B We assume here that the only degree of freedom in the orientation of the block is its
rotation

θ about its Z axis

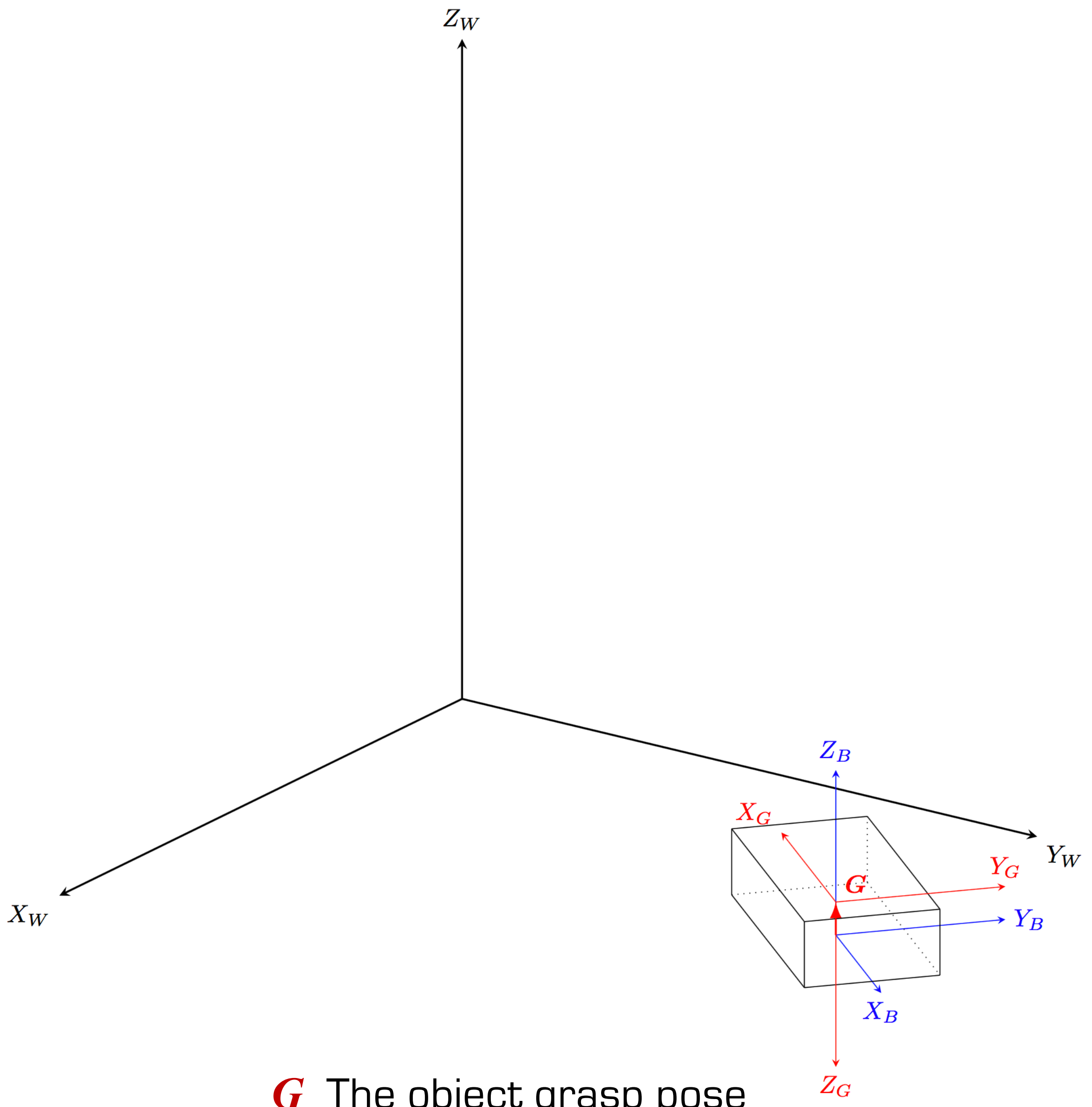
Furthermore, we assume that the surface on which the block is lying is in the x-y
plane

in the world frame of reference

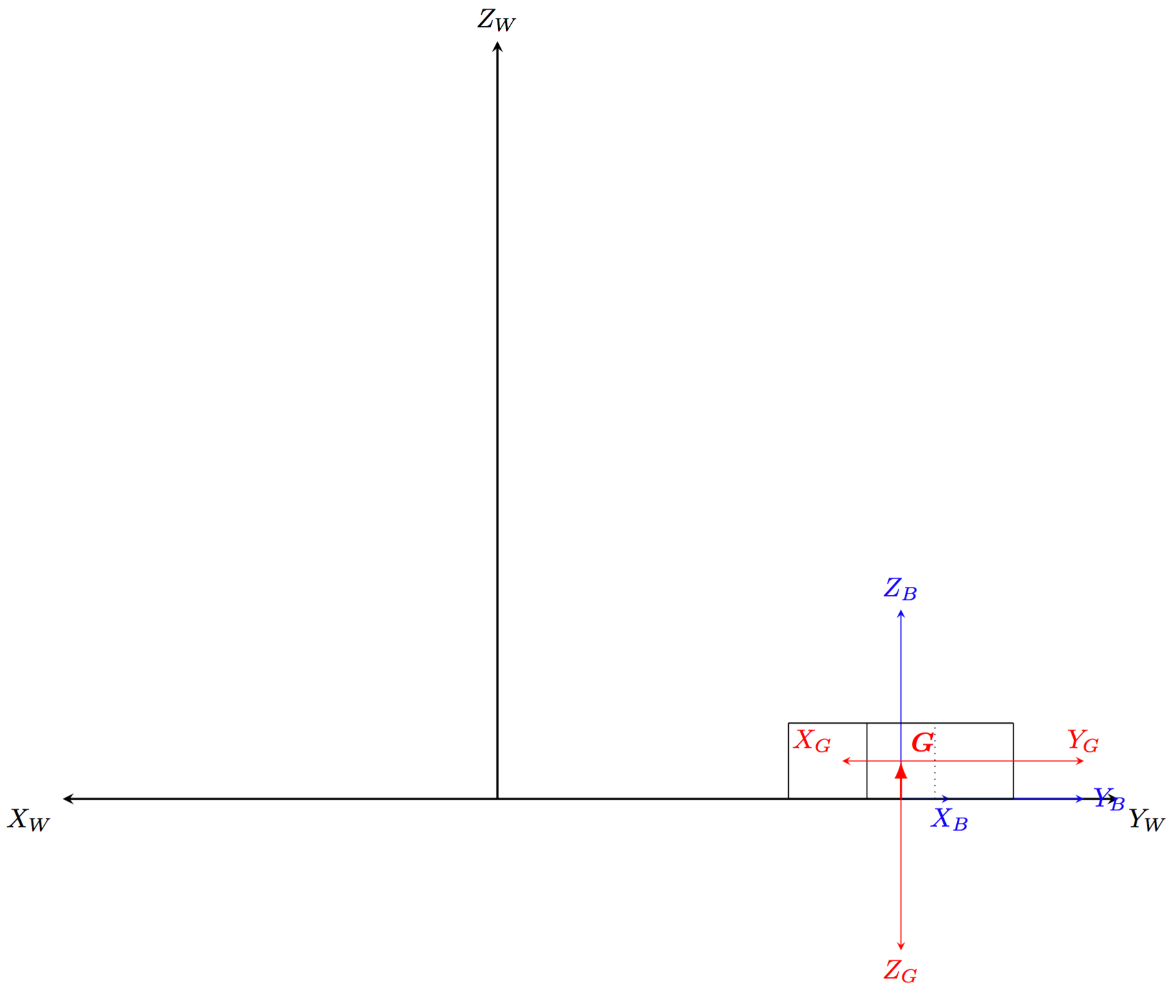
Hence, the z coordinate of its position is zero

If we are using a camera, the vision system computes x, y and ϕ

For the purposes of this example, we will specify x, y and ϕ explicitly
(see example later)



$\boldsymbol{G}$ The object grasp pose



$\boldsymbol{G}$ The object grasp pose

$^{B}\boldsymbol{G}$	the position of the end-effector holding the block, defined with respect to the block

The origin of the gripper frame ^{B}G is defined to be located a a distance half the height of the block from the origin of B along the block's Z axis

To accomplish this, we perform a translation $\boldsymbol{Trans}\ (0,\ 0,\ h/2)$

${}^{B}\boldsymbol{G}$	the position of the end-effector holding the block, defined with respect to the block

The Z axis is defined to be normal to the block's x-y plane,
but directed downwards

The X axis is defined to be aligned along the major axis of the block

The Y axis makes up a right-hand system and, hence, the gripper grasps the block along is minor axis

To accomplish this, we perform a rotation of 180 degrees about the station Y axis (i.e. w.r.t. the translated frame):
$\boldsymbol{Rot}(Y,\ 180)$

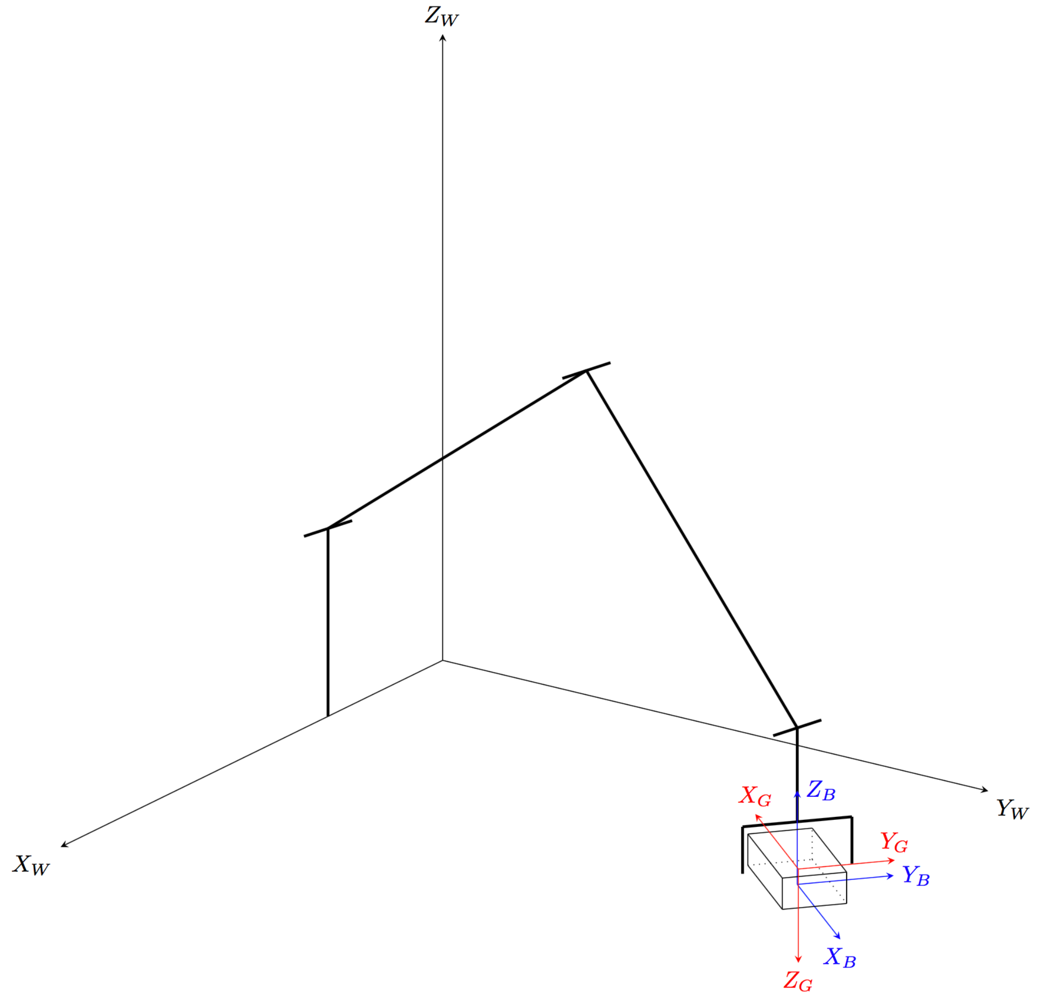
$^{B}\boldsymbol{G}$ the position of the end-effector holding the block, defined with respect to the block

Thus,

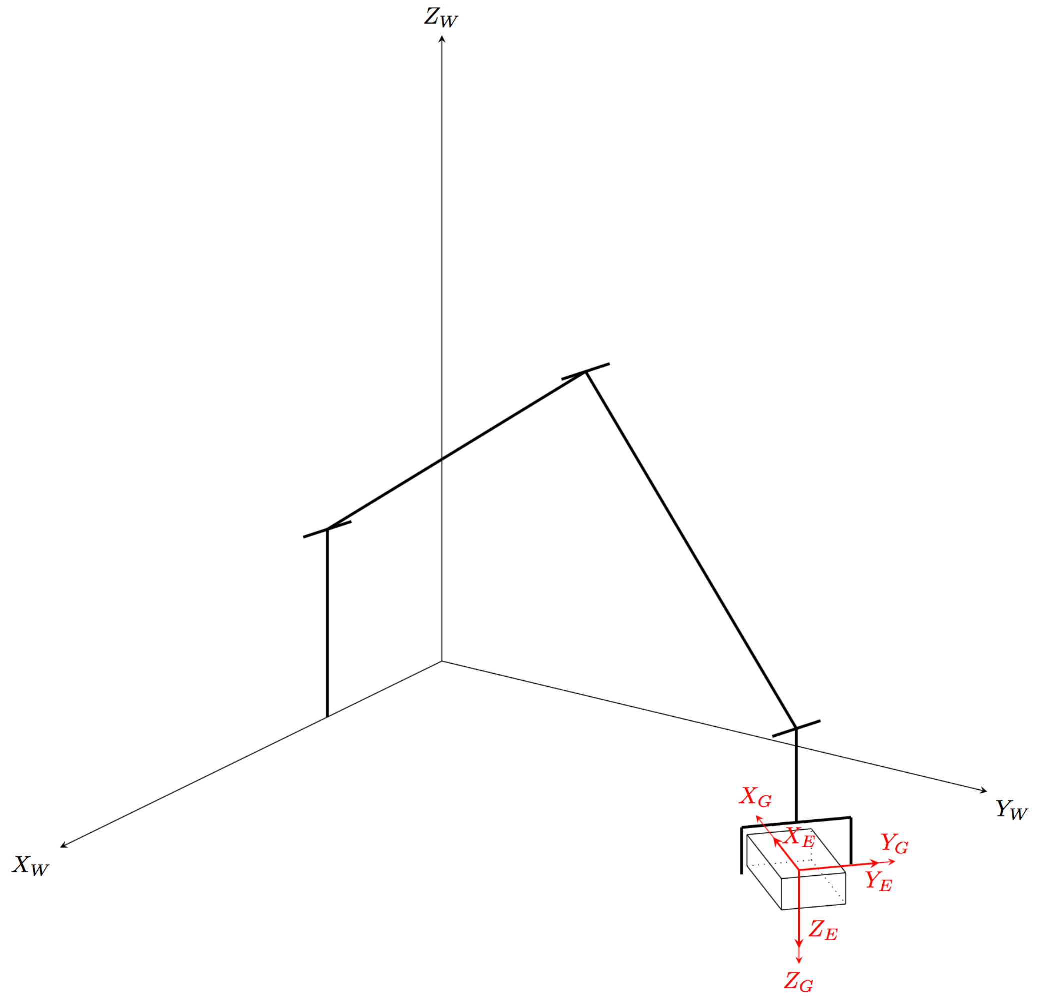
$$^{B}\boldsymbol{G} = \boldsymbol{Trans}\,(0, 0, h/2)\ \ \boldsymbol{Rot}(Y, 180)$$

It is important to note that we define the ${}^{B}\boldsymbol{G}$ frame in this manner because this is how the end-effector $\boldsymbol{E}$ will be oriented when grasping the block …

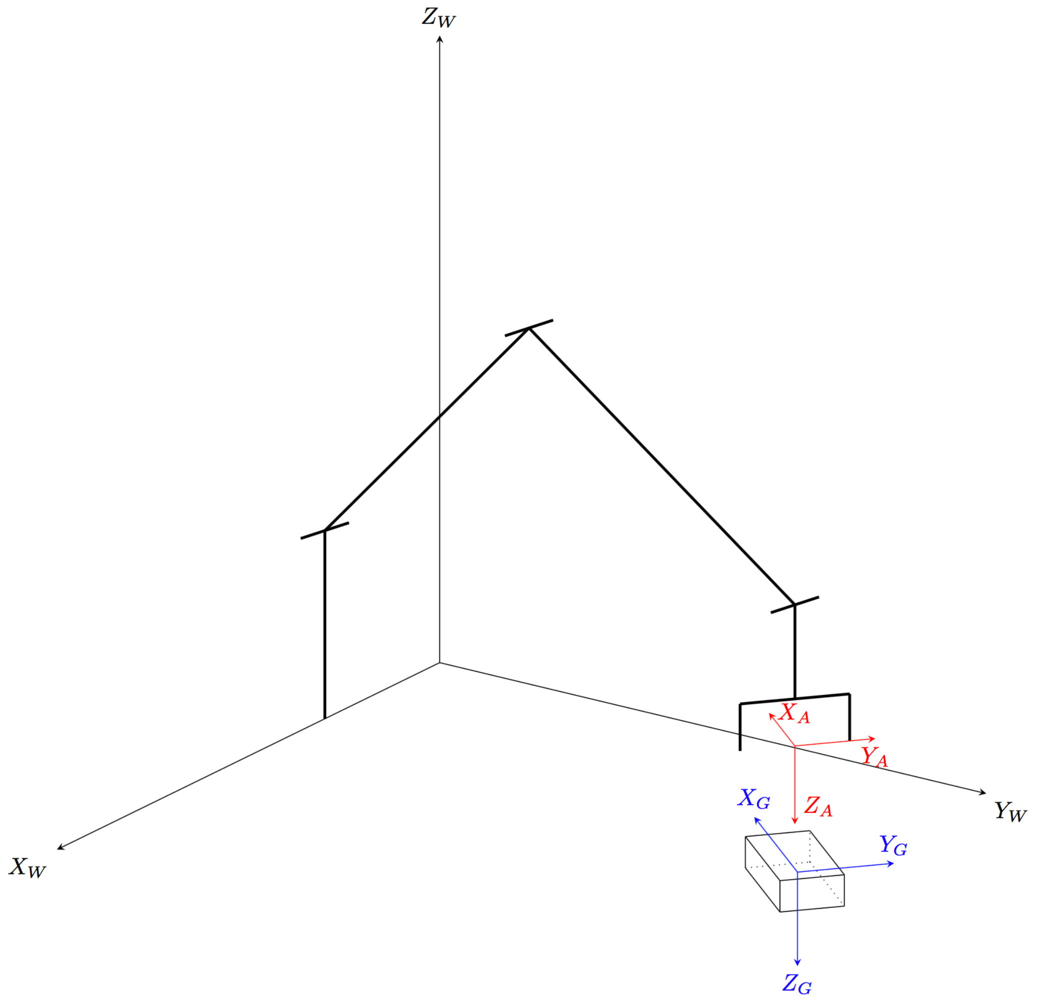
with the Z axis pointing vertically downward and the Y axis at right angles to the major axis of the block



We define the object grasp pose so that ...



… the end-effector pose aligned with the object grasp pose



A The position of the end-effector approaching the grasp position

$^{G}\boldsymbol{A}$ the pose of the end-effector approaching the grasp position, defined with respect to the grasp position

This is defined to be a position directly above the grasp point

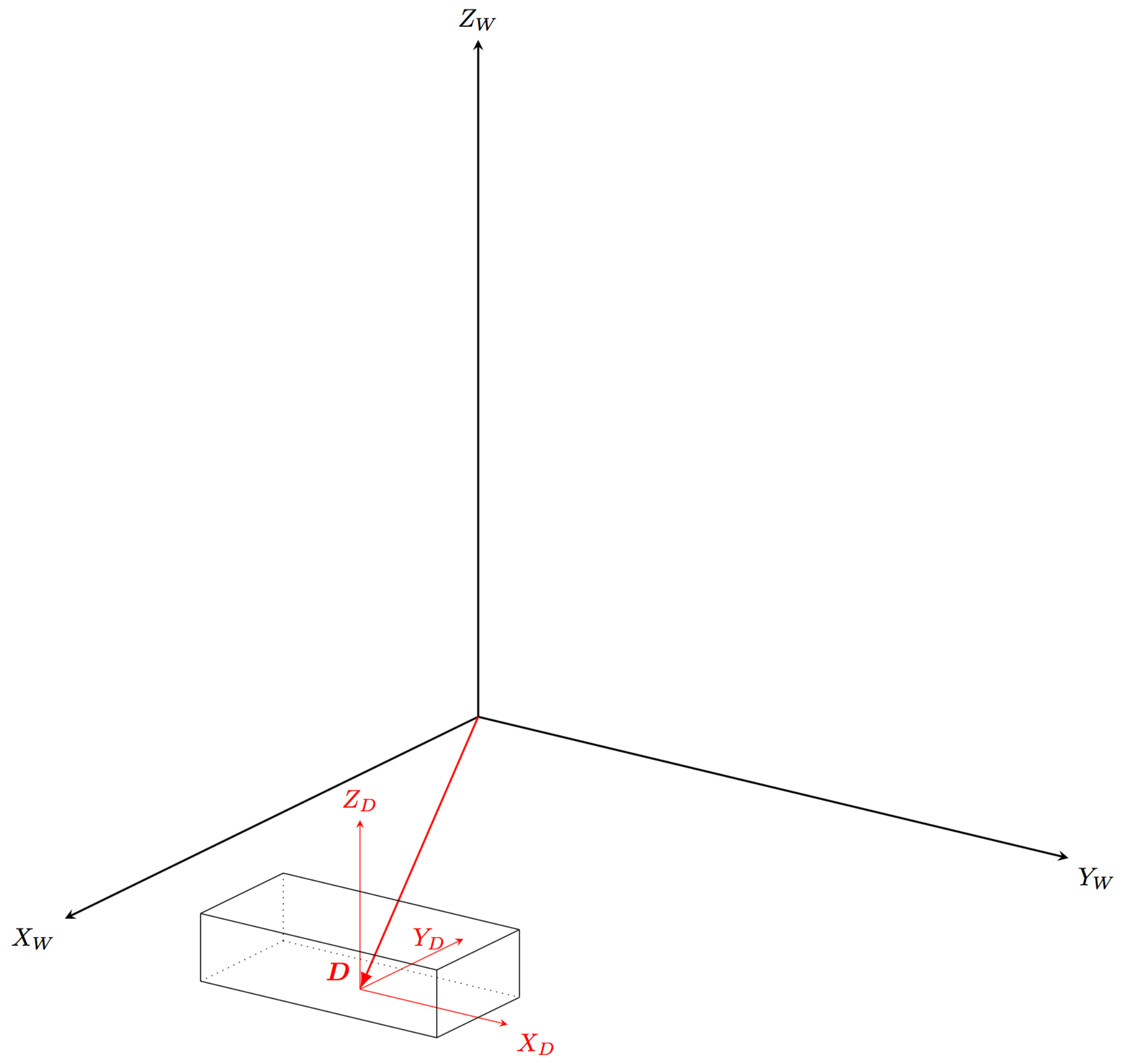
As such, it simply involves a translation in the negative direction of the Z axis of the $^{B}\boldsymbol{G}$ frame

Foe convenience, we use the same frame to define the pose of the end-effector departing the grasp position, after having grasped the block

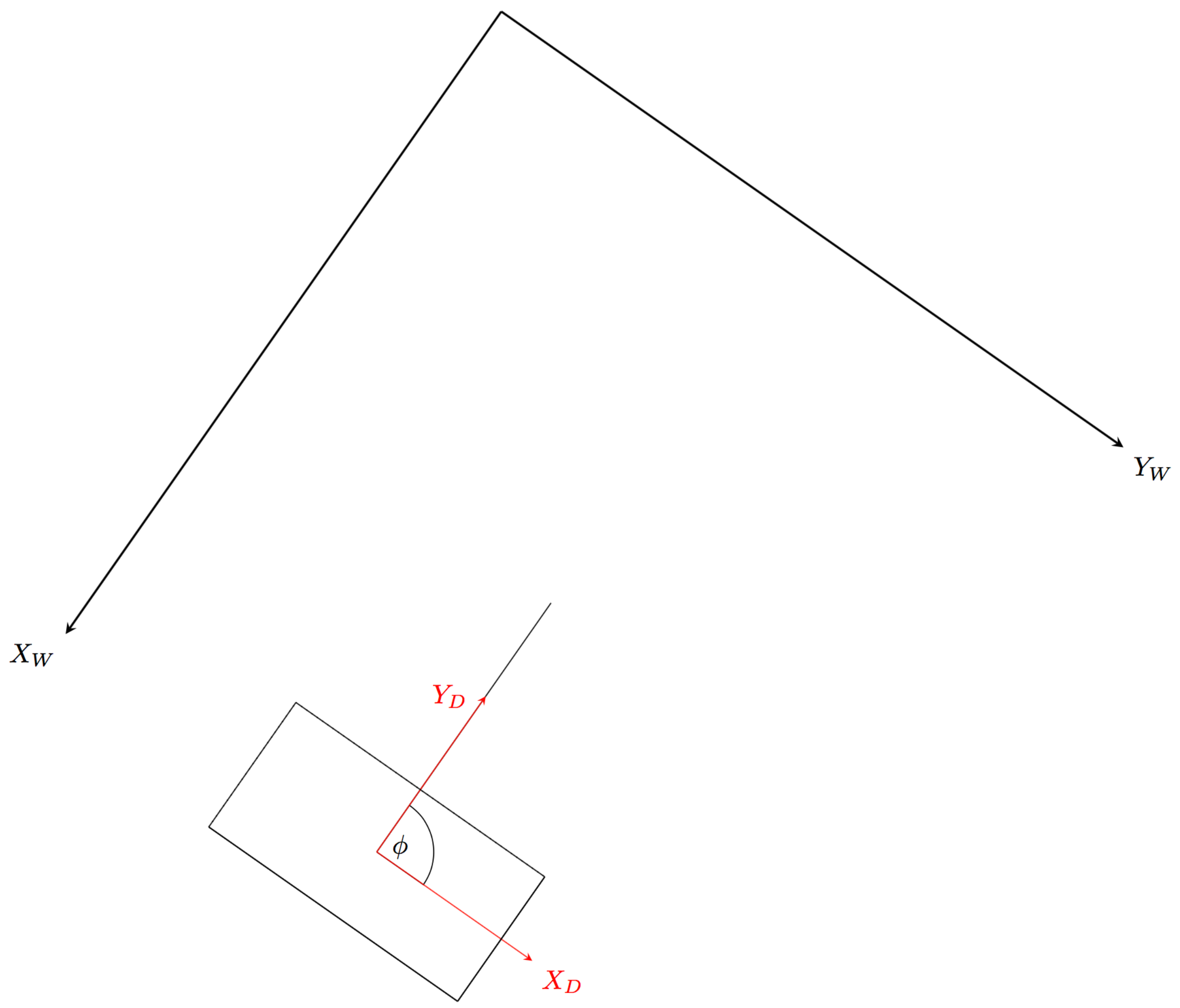
Thus,

$$^{G}\boldsymbol{A} = \boldsymbol{Trans}(0, 0, -d)$$

D the pose of the destination of the block, defined with respect to the base co-ordinate reference system



D The pose of the block at the destination



D The pose of the block at the destination

A Simple Pick-and-Place Task Specification

$M0$:	Move out of the field of view of the camera
	Determine the pose of a block and a suitable grasp point (possibly using a camera)
$M1$:	Move to an approach position above the grasp point
$M2$:	Move to the grasp position
	Grasp the block
$M3$:	Move to the depart position above the grasp point
$M4$:	Move to the approach position in above the destination position
$M5$:	Move to the destination position
	Release the block
$M6$:	Move to the depart position above the block

A Simple Robot Programming Language

- This task-level approach to robot programming is typical of many commercial manipulators and they typically provide their own frame-based programming language

- In the following, we show how it can be implemented in C++ by defining a Frame class

  - The assignment operator is overloaded to allow assignment of Frame objects

  - The multiplication operator overloaded so that it effects the concatenation of Frame objects, i.e. homogeneous transformation

Thus, assuming the frames `T6`, `Z`, `B`, `G`, and `E` have been declared, and the pose values `x`, `y`, `theta`, and `blockHeight` have valid values, the transformation equation

$$T6 = Z^{-1}\ B\ \ G\ \ E^{-1}$$

can be implemented as

```
Z = trans(0, 0, 0);
B = trans(x, y, 0) * rotz(phi);
G = trans(0, 0, blockHeight/2) * roty(180);
E = trans(0, 0, 100);
T6 = inv(Z) * B  * G * inv(E);
move(T6);
```

- Note: for the sake of clarity, we are adopting the convention that the frame variables are written in upper case

- Normally, in C++, the first character of an object is written in lower case and the first character of a class name in upper case

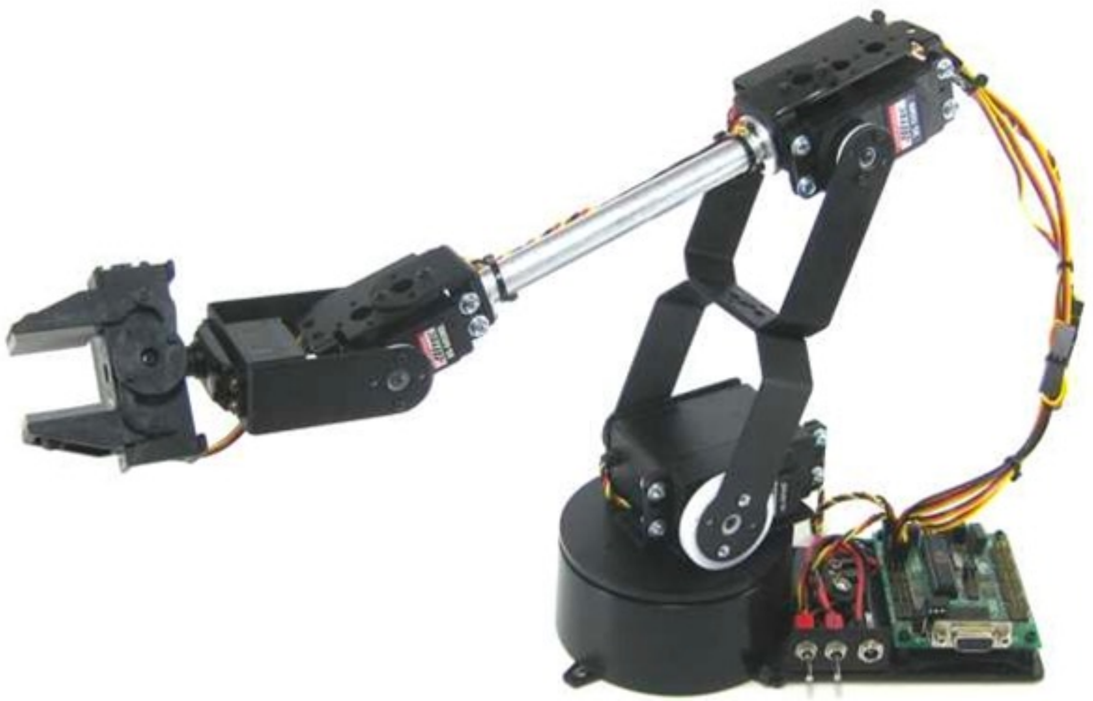
```
/*******************************************************************************************************************
*   Example pick-and-place program for a LynxMotion AL5D robot arm
*   ---------------------------------------------------------------
*
*   This application implements a simple robot program to grasp a simple object (a block),
*   lift it up, and place it somewhere else.
*
*   The position and orientation (pose) of the object and the goal position are specified in the input file.
*   (The pickAndPlaceVision application uses a camera to determine the object pose.)
*
*   The program uses task-level programming using frames to specify the object, robot, and gripper poses.
*
*   This application reads three lines from an input file pickAndPlace.txt.
*
*   The first line contains a filename of the file with the robot calibration data, i.e. for the inverse kinematic solution.
*   This allows the program to be used with different robots (by specifying the corresponding calibration data file).
*
*   The second line contains the object pose, i.e. the x, y, and z coordinates and the phi angle of the object (i.e. rotation about z).
*
*   The third line contains the destination pose, i.e. the x, y, and z coordinates and the phi angle of the destination (i.e. rotation about z).
*
*   It is assumed that the input file is located in a data directory given by the path ../data/
*   defined relative to the location of executable for this application.
*
*
*   David Vernon, Carnegie Mellon University Africa
*   4 February 2020
*
*   Audit Trail
*   -----------
*   No changes yet
*
*******************************************************************************************************************/
```

```
#include "pickAndPlace.h"

int main(int argc, char ** argv) {

   extern robotConfigurationDataType robotConfigurationData;
   bool debug = true;
   FILE *fp_in;                    // pickAndPlace input file
   int  end_of_file;
   char robot_configuration_filename[MAX_FILENAME_LENGTH];

   /* Frame objects */

   Frame E;
   Frame Z;
   Frame T6;
   Frame block;
   Frame grasp;
   Frame approach;
   Frame destination;

   /* data variables */

   float effector_length;           // this is initialized from robot configuration file

   float object_x          = -40;   // default values; actual values are read from the input file
   float object_y          = 150;   //
   float object_z          =   0;   //
   float object_phi        = -90;   // rotation in degrees about the z (vertical) axis

   float destination_x     =  40;   // default values; actual values are read from the input file
   float destination_y     = 150;
   float destination_z     =   0;
   float destination_phi   = -90;   // rotation in degrees about the z (vertical) axis

   float grasp_x           =   0;   // grasp pose relative to object and destination poses
   float grasp_y           =   0;
   float grasp_z           =  10;
   float grasp_theta       = 180;   // rotation in degrees about the y axis

   float approach_distance = 100;   // approach and departure distance from grasp pose in -z direction
```

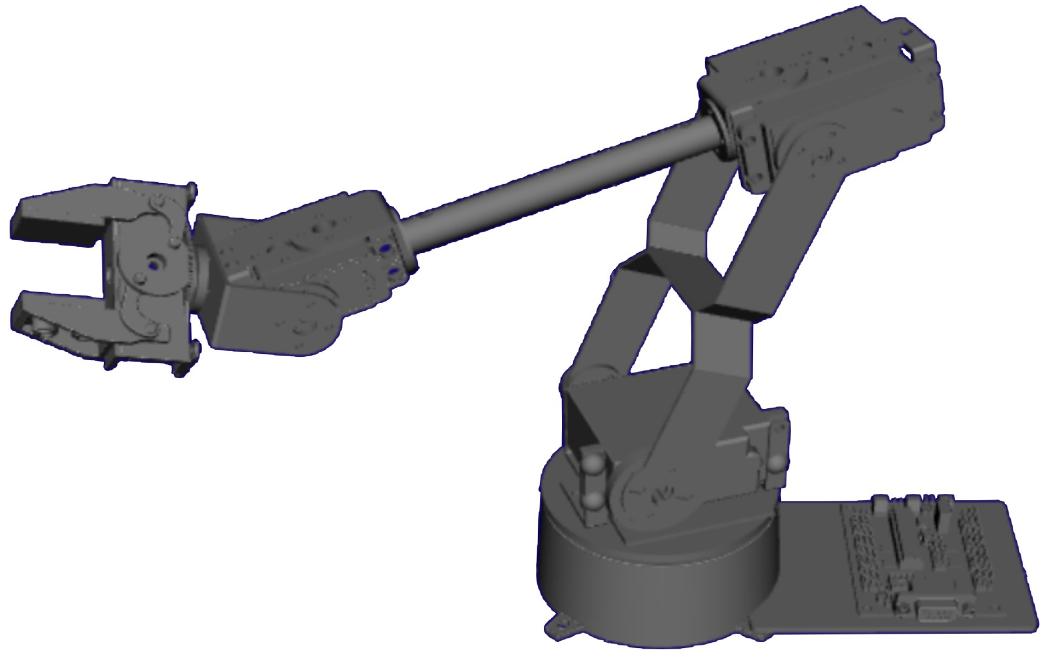
```
/* open the input file */
/* ------------------- */

if ((fp_in = fopen("../data/pickAndPlaceInput.txt","r")) == 0) {
   printf("Error can't open input pickAndPlaceInput.txt\n");
   prompt_and_exit(0);
}


/* get the robot configuration data */
/* -------------------------------- */

end_of_file = fscanf(fp_in, "%s", robot_configuration_filename); // read the configuration filename
if (end_of_file == EOF) {
   printf("Fatal error: unable to read the robot configuration filename\n");
   prompt_and_exit(1);
}

readRobotConfigurationData(robot_configuration_filename);


/* get the object pose data */
/* ------------------------ */

end_of_file = fscanf(fp_in, "%f %f %f %f", &object_x, &object_y, &object_z, &object_phi);
if (end_of_file == EOF) {
   printf("Fatal error: unable to read the object position and orientation\n");
   prompt_and_exit(1);
}


/* get the destination pose data */
/* ----------------------------- */

end_of_file = fscanf(fp_in, "%f %f %f %f", &destination_x, &destination_y, &destination_z, &destination_phi);
if (end_of_file == EOF) {
   printf("Fatal error: unable to read the destination position and orientation\n");
   prompt_and_exit(1);
}
```

```
/* now start the pick and place task */
/* --------------------------------- */

effector_length = (float) robotConfigurationData.effector_z; // initialized from robot configuration data

E            = trans(0.0, 0.0, effector_length);                                          // end-effector (gripper) frame
Z            = trans(0.0 ,0.0, 0.0);                                                      // robot base frame
object       = trans(object_x,      object_y,      object_z)      * rotz(object_phi);      // object pose
destination  = trans(destination_x, destination_y, destination_z) * rotz(destination_phi); // destination pose
grasp        = trans(grasp_x,       grasp_y,       grasp_z)       * roty(grasp_theta);     // grasp frame w.r.t. object & destination frames
approach     = trans(0,0,-approach_distance);                                             // frame defined w.r.t. grasp frame

/* close the gripper */
/* ----------------- */

setGripper(GRIPPER_OPEN);
wait(1000);  //  1 second

/* move to initial approach pose */
/* ----------------------------- */

T6 = inv(Z) * object * grasp * approach * inv(E);

if (move(T6) == false)
   display_error_and_exit("move error ... quitting\n");;
wait(4000);  // 2 seconds

/* move to the grasp pose */
/* ---------------------- */

T6 = inv(Z) * object * grasp * inv(E);
if (move(T6) == false)
   display_error_and_exit("move error ... quitting\n");
wait(2000);   // 2 seconds

/* close the gripper */
/* ----------------- */

setGripper(GRIPPER_CLOSED);
wait(2000);
```

```
   /* move back to initial approach pose */
   /* ---------------------------------- */

   T6 = inv(Z) * object * grasp * approach * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");
   wait(3000);  // 3 seconds

   /* move to destination approach pose */
   /* --------------------------------- */

   T6 = inv(Z) * destination * grasp * approach * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");
   wait(3000); // 2 seconds

   /* move to the destination pose */
   /* ---------------------------- */

   T6 = inv(Z) * destination * grasp * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");;
   wait(2000);  // 2 seconds

   /* open the gripper */
   /* ---------------- */

   setGripper(GRIPPER_OPEN);
   wait(2000);  // 2 seconds

   /* move back to initial approach pose */
   /* ---------------------------------- */

   T6 = inv(Z) * destination * grasp * approach * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");;
   wait(3000);  // 2 seconds

   goHome(); // this returns the robot to the home position; could also do this with a move() as shown above
   return 0;
}
```



Lynxmotion AL5D Robotic Arm with serial interface



Robot Arms
Lynxmotion AL5D Simulator

Video

https://youtu.be/wprb2AhSLUE

Recommended Reading

D. Vernon, Machine Vision – Automated Visual Inspection and Robot Vision, Prentice Hall International, 1991. Chapter 8.

http://vernon.eu/publications/91_Vernon_Machine_Vision.pdf

Similar material to that presented in this lecture.

R. P. Paul, Robot Manipulators – Mathematics, Programming, and Control, MIT Press, 1981. Chapter 1.

https://books.google.rw/books?id=UzZ3LAYqvRkC&printsec=frontcover&source=gbs_ViewAPI&redir_esc=y#v=onepage&q&f=false

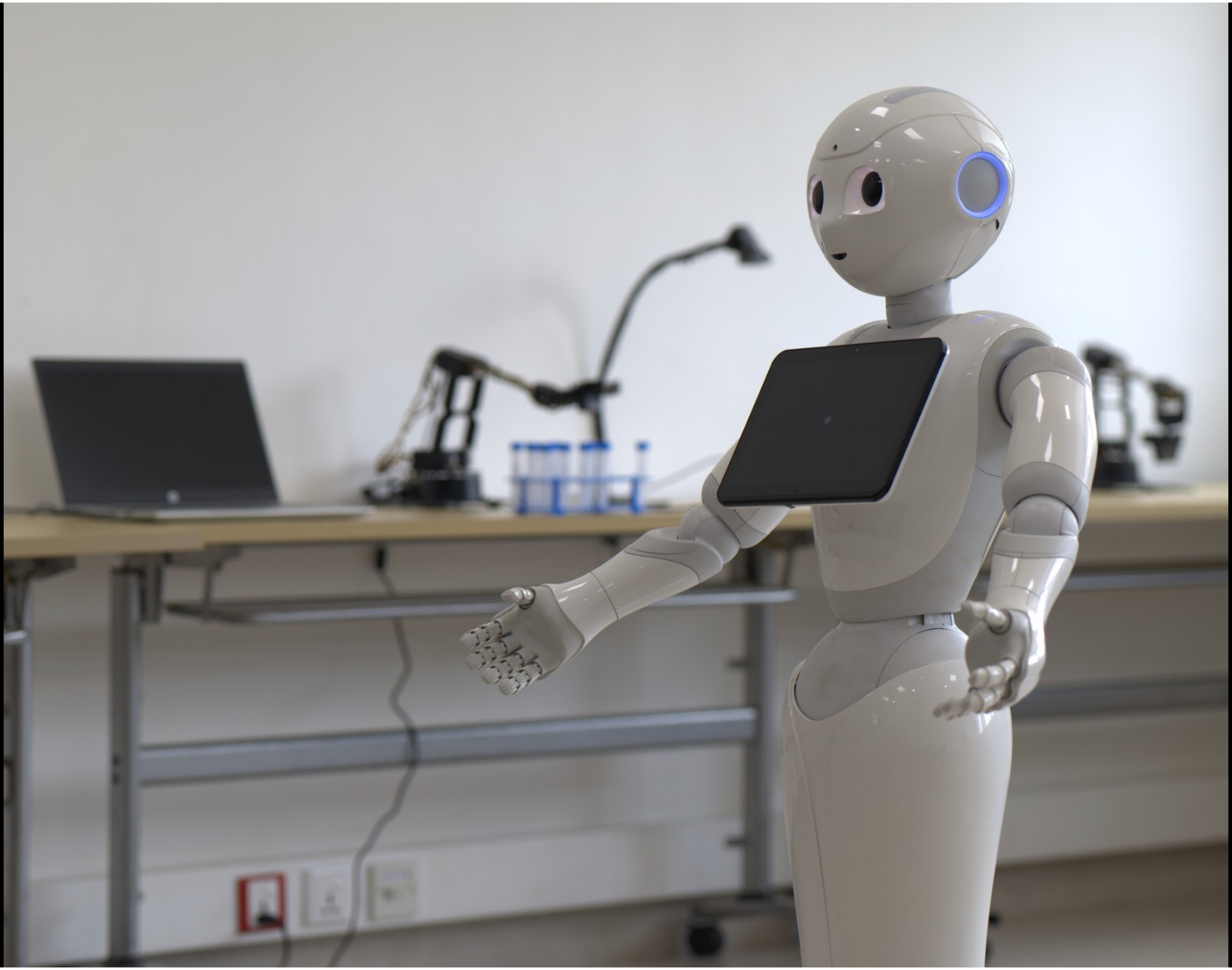
Similar material to that presented in this lecture but complete comprehensive treatment.

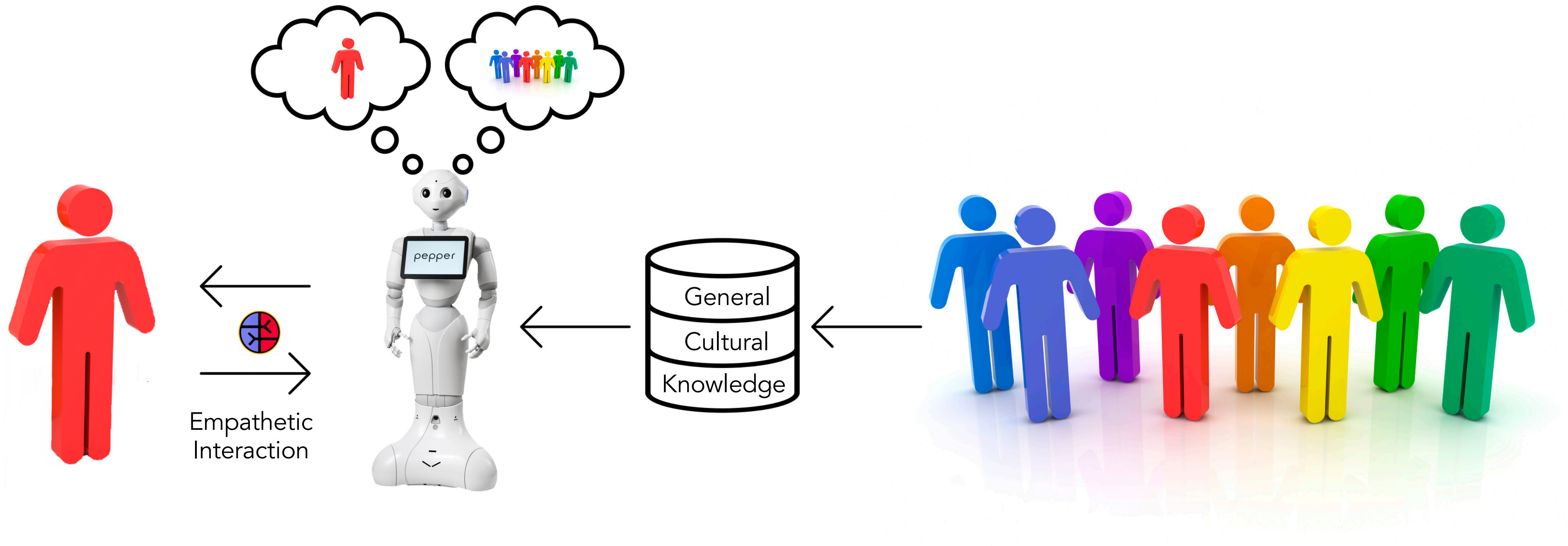
P. Corke, Robotics, Vision and Control, 2nd Edition, Springer, 2017.
Comprehensive contemporary treatment; highly recommended.

Lecture Topics

1. What is a robot?
2. Types of robot
3. Sensors
4. Actuators
5. Effectors
6. Control systems
7. The Robot Operating System (ROS)
8. Programming robot manipulators
9. Object pose specification
10. Fame-based task specification
11. Pick-and-place example of task-level robot programming
12. Inclusive social robotics

$\Big\}$ Principles



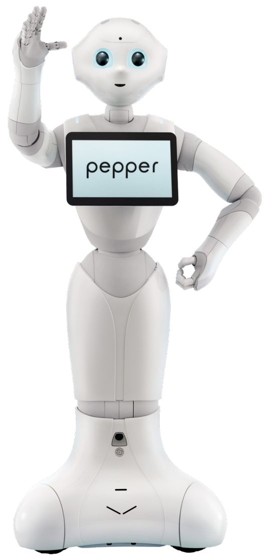


Graphic based based on work by Bruno et al. (2017)

A Sample of African Culture-specific Knowledge

No.	Socio-cultural Norm or Trait
1	All interactions should begin with a courteous greeting.
2	The younger interaction partner should enable a greeting to be initiated by an older person.
3	The younger interaction partner should bow when greeting an older person or when rendering a service.
4	One should not wave at someone from a distance; one should move towards them to greet them.
5	To show respect, one should bow slightly and lower gaze when greeting someone older.
6	To show respect, one should raise both hands and lower gaze a little when greeting.
7	One should suspend work or movements and pay attention when addressed.
8	One should use an open palm of the hand to point to people and objects.
9	One should not point an upward facing palm of the hand at someone.
10	One should not use the left hand to point to anything.
11	One should not use the left hand to hand something to someone.
12	To show respect, one should hand over and accept gifts with two hands and do so from the front, facing the recipient.
13	It is respectful to use local languages and they should be used for verbal interaction when possible.
14	One should use formal titles when addressing someone.
15	One should engage in a preamble before getting to the point, as being too forward may be regarded as disrespectful.
16	One should not interrupt or talk over someone when they are speaking.
17	One should not interrupt or talk over someone when they are speaking.
18	One should keep intermittent eye contact; lack of eye contact depicts disrespect as it shows divided attention during the interaction.
19	One should not make persistent eye contact with an older person.
20	One should not make eye contact when being corrected.
21	To show respect, one should shake hands with the right hand and use the left arm to support the right forearm when doing so.
22	One should not walk far ahead of an older person, unless leading the person (in which case, one should walk slightly to the side).
23	One should not walk between two or more people who are conversing; it is considered rude to do so.
24	An appreciation of rhythmic sound and movement is valued.
25	Behaviours should focus on fostering social connections and relationships; they should not be purely functional.

After (Bruno et al, 2019)

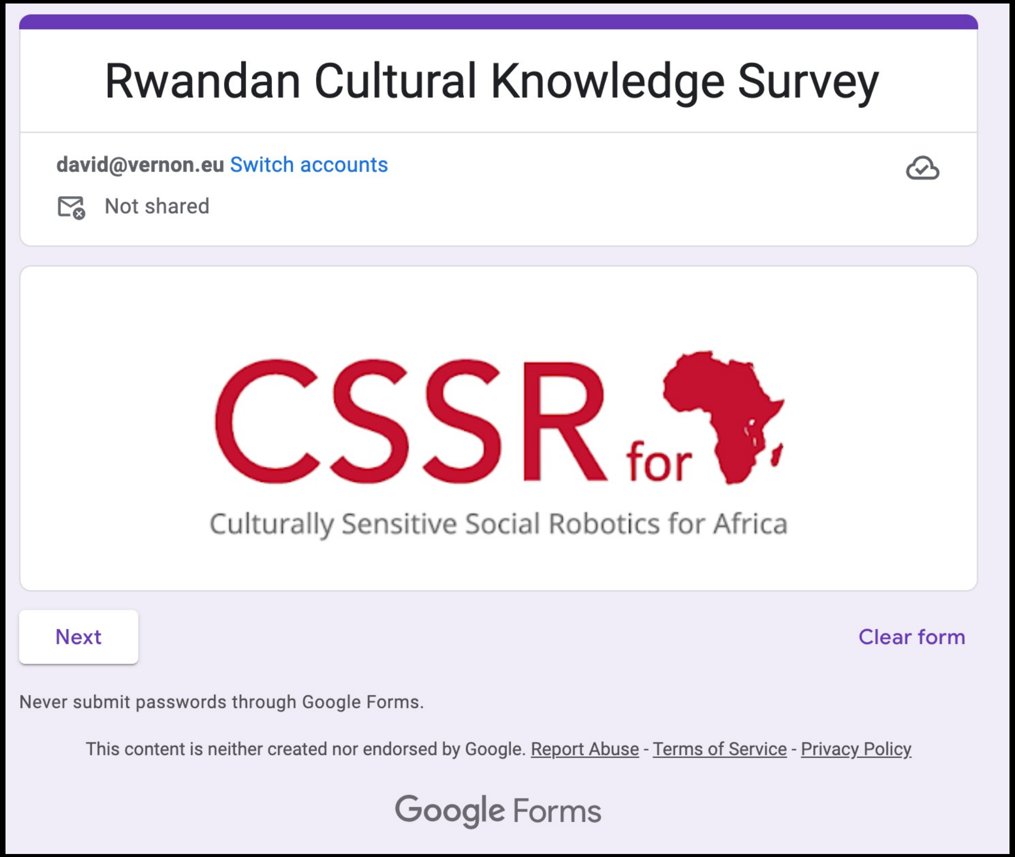


Spatial, Non-verbal, Verbal Interaction

Design Pattern	Culturally Competent Behavior
Initial Introduction	The robot should acknowledge the presence of the person. The robot should initiate an interaction with a slight bow. The robot should greet first and should use a formal greeting. The robot should respect personal and intimate distances during interaction.
Reciprocal Turn Taking	The robot should respectfully give the initial turn to the human interaction partner. The robot should give priority to older people; it should not interrupt and it should let the other person finish their turn.
Didactic Communication	Pointing a hand directly at someone is disrespectful. For deictic gestures, the robot should use its left hand. The robot should gesture with an open palm rather than pointing a finger.
Personal Interests and History	The robot should avoid trying to share personal history since it will be perceived to be inauthentic. The robot should focus on and highlight its functional usefulness.
In Motion Together	The robot should explicitly say "Please come along" to remove any ambiguity of intention. The robot should not walk too far ahead when showing the way.
Recovering from Mistakes	The robot should apologize profusely. The robot should slightly bow when introducing itself and after it makes a mistake.
Physical Intimacy	Personal space should be entered only with prior consent. The robot should not pass in between two people that are interacting.
Claiming Unfair Treatment or Wrongful Harm	To enhance the perception that the robot is being respectful, the robot should not be aggressive by claiming unfair treatment.

A Sample of Africa-centric Design Patterns for Social Robots

After (Kahn et al, 2008)



Rwandan Cultural Knowledge Survey

david@vernon.eu Switch accounts

Not shared

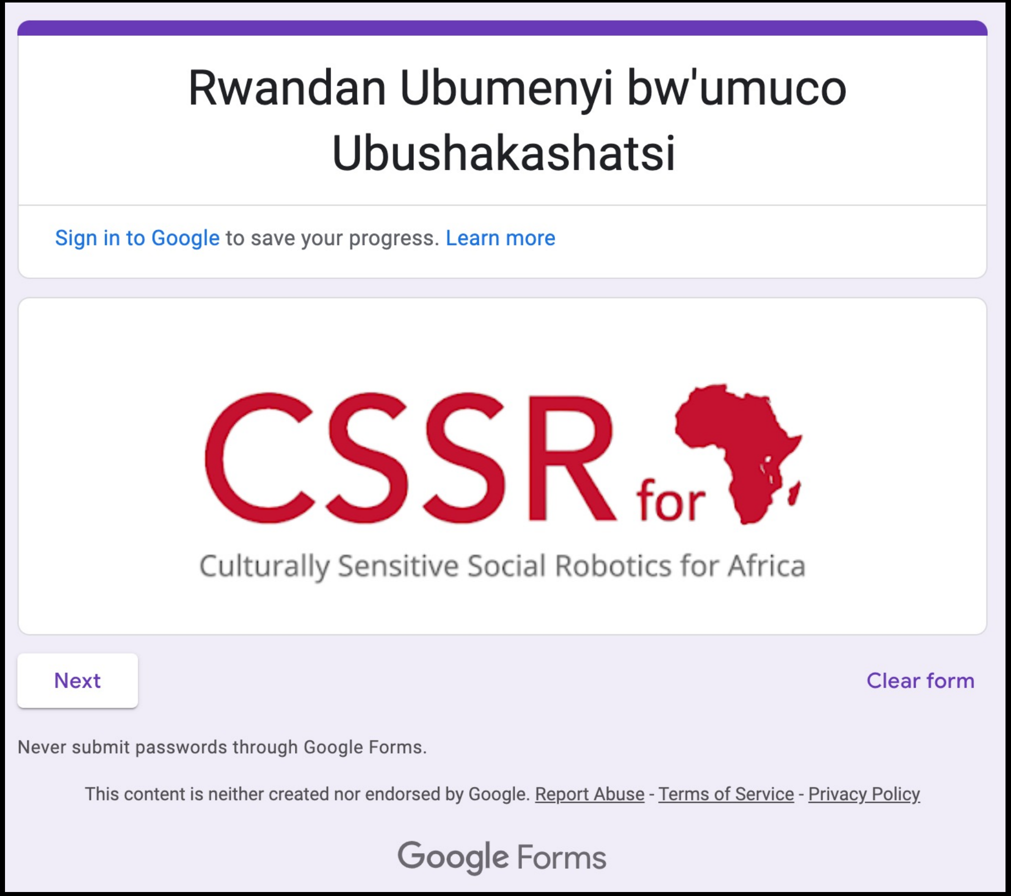
Next

Clear form

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. Report Abuse - Terms of Service - Privacy Policy

Google Forms


Rwandan Ubumenyi bw'umuco Ubushakashatsi

Sign in to Google to save your progress. Learn more

CSSR for Africa
Culturally Sensitive Social Robotics for Africa

Never submit passwords through Google Forms.

Culturally Competent Social Robot

{Bruno et al, 2017}

1

Cultural knowledge representation

Culturally Competent Social Robot

{Bruno et al, 2017}

1. **Cultural knowledge representation**
2. **Culturally sensitive planning and action execution**

Culturally Competent Social Robot

{Bruno et al, 2017}

Cultural knowledge representation

Culturally sensitive planning and action execution

Culturally aware multimodal human-robot interaction

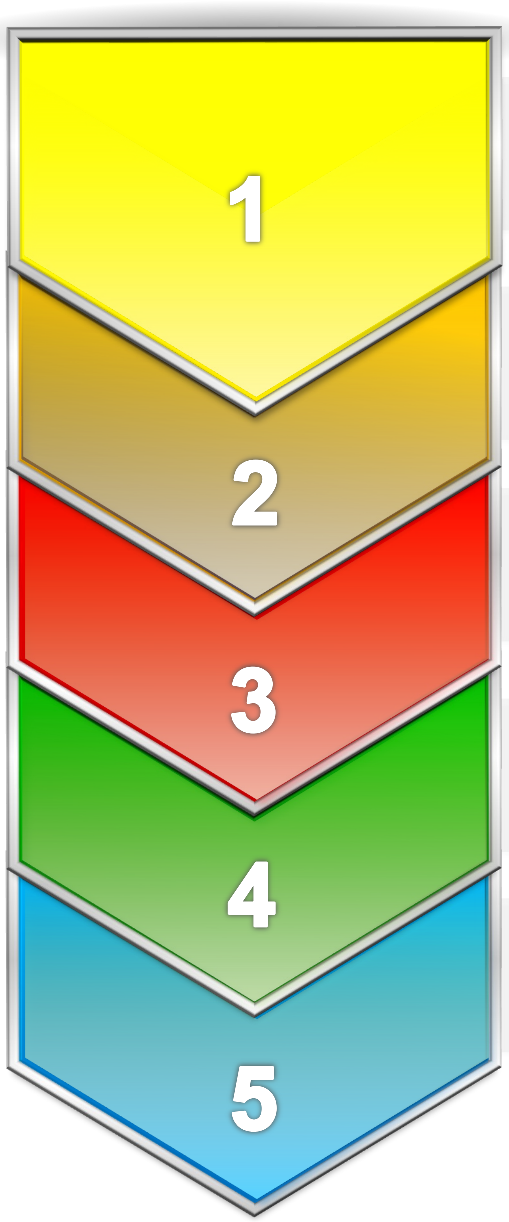
Culturally Competent Social Robot

{Bruno et al, 2017}

1. Cultural knowledge representation
2. Culturally sensitive planning and action execution
3. Culturally aware multimodal human-robot interaction
4. Culture-aware human emotion recognition

Culturally Competent Social Robot

{Bruno et al, 2017}



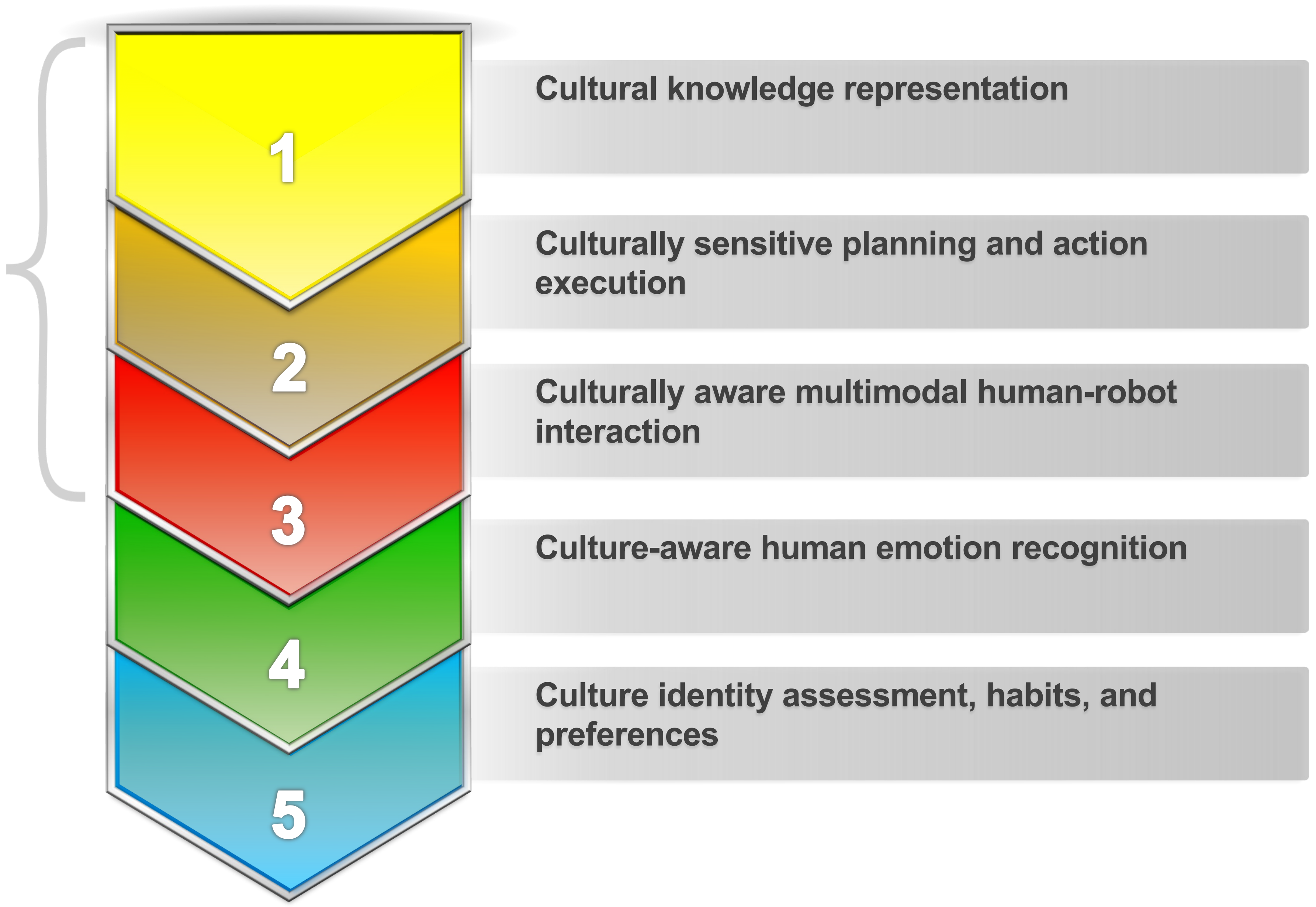
Cultural knowledge representation

Culturally sensitive planning and action execution

Culturally aware multimodal human-robot interaction

Culture-aware human emotion recognition

Culture identity assessment, habits, and preferences



Cultural knowledge representation

Culturally sensitive planning and action execution

Culturally aware multimodal human-robot interaction

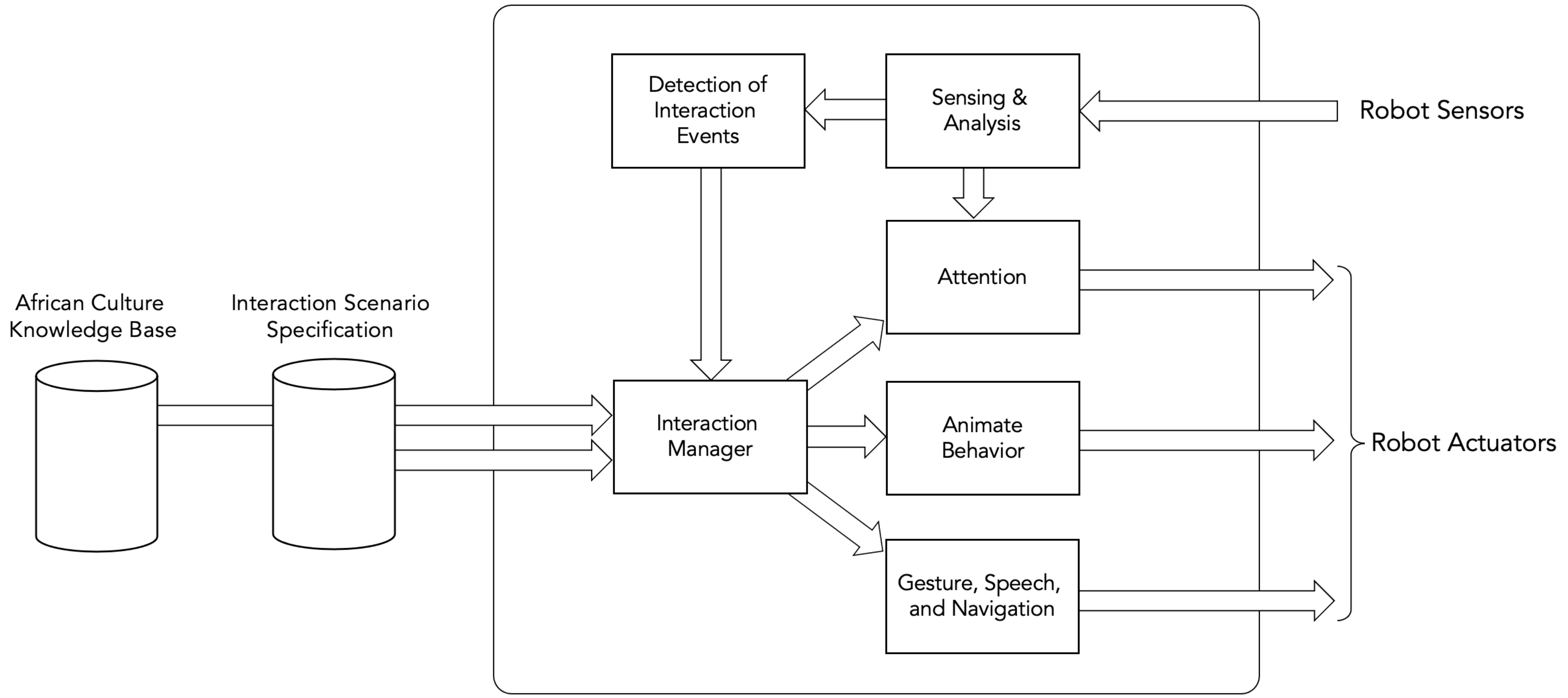
Culture-aware human emotion recognition

Culture identity assessment, habits, and preferences

CSSR for

Culturally Sensitive Social Robotics for Africa

www.cssr4africa.org

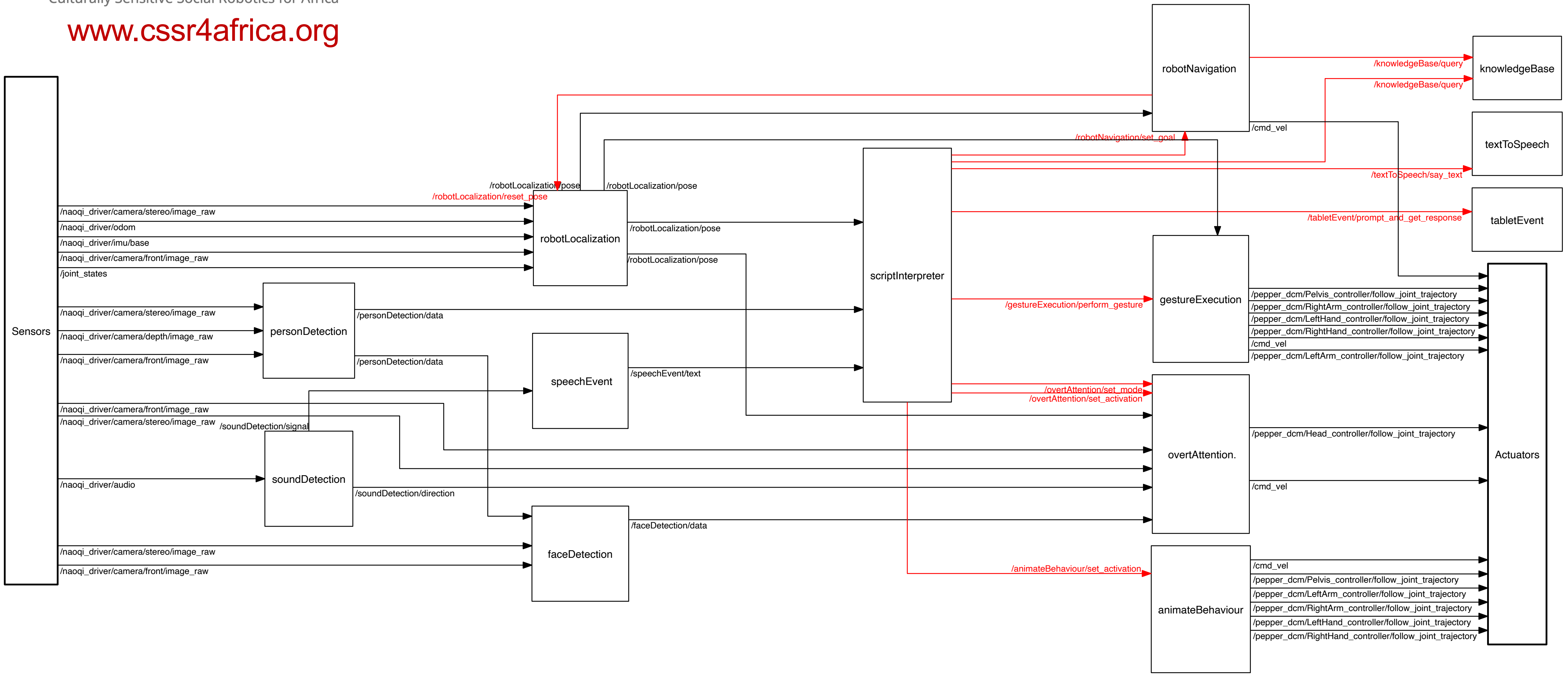


Detection of Interaction Events
Sensing & Analysis
Robot Sensors
Attention
African Culture Knowledge Base
Interaction Scenario Specification
Interaction Manager
Animate Behavior
Robot Actuators
Gesture, Speech, and Navigation

CSSR for Africa

Culturally Sensitive Social Robotics for Africa

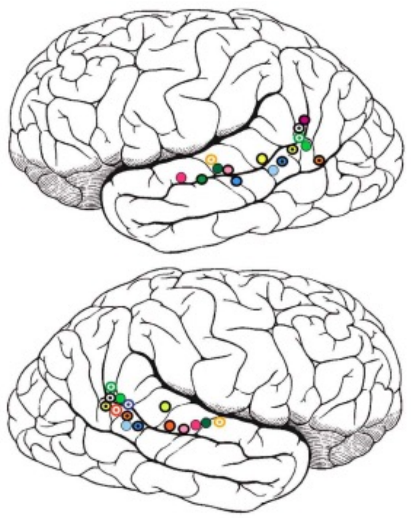
www.cssr4africa.org



Sensors
robotLocalization
personDetection
speechEvent
soundDetection
faceDetection
scriptInterpreter
robotNavigation
gestureExecution
overtAttention.
animateBehaviour
knowledgeBase
textToSpeech
tabletEvent
Actuators
/naoqi_driver/camera/stereo/image_raw
/naoqi_driver/odom
/naoqi_driver/imu/base
/naoqi_driver/camera/front/image_raw
/joint_states
/naoqi_driver/camera/stereo/image_raw
/naoqi_driver/camera/depth/image_raw
/naoqi_driver/camera/front/image_raw
/naoqi_driver/camera/front/image_raw
/naoqi_driver/camera/stereo/image_raw
/soundDetection/signal
/naoqi_driver/audio
/naoqi_driver/camera/stereo/image_raw
/naoqi_driver/camera/front/image_raw
/robotLocalization/pose
/robotLocalization/reset_pose
/robotLocalization/pose
/robotLocalization/pose
/robotLocalization/pose
/personDetection/data
/personDetection/data
/speechEvent/text
/soundDetection/direction
/faceDetection/data
/robotNavigation/set_goal
/knowledgeBase/query
/knowledgeBase/query
/textToSpeech/say_text
/tabletEvent/prompt_and_get_response
/gestureExecution/perform_gesture
/overtAttention/set_mode
/overtAttention/set_activation
/animateBehaviour/set_activation
/cmd_vel
/pepper_dcm/Pelvis_controller/follow_joint_trajectory
/pepper_dcm/RightArm_controller/follow_joint_trajectory
/pepper_dcm/LeftHand_controller/follow_joint_trajectory
/pepper_dcm/RightHand_controller/follow_joint_trajectory
/cmd_vel
/pepper_dcm/LeftArm_controller/follow_joint_trajectory
/pepper_dcm/Head_controller/follow_joint_trajectory
/cmd_vel
/cmd_vel
/pepper_dcm/Pelvis_controller/follow_joint_trajectory
/pepper_dcm/LeftArm_controller/follow_joint_trajectory
/pepper_dcm/RightArm_controller/follow_joint_trajectory
/pepper_dcm/LeftHand_controller/follow_joint_trajectory
/pepper_dcm/RightHand_controller/follow_joint_trajectory

CSSR for
Culturally Sensitive Social Robotics for Africa
www.cssr4africa.org

Significance of Biological Motion in Gestural Communication



Biological motion activates the superior temporal sulcus (STS) of the human brain, promoting engagement
(Puce and Perret, 2003)

CSSR for Africa
Culturally Sensitive Social Robotics for Africa
www.cssr4africa.org

Models of Biological Motion

Minimum Jerk
(Chan et al., 2021)

$$C = 1/2 \underbrace{\int_{t_1}^{t_2} \left[\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2 \right]}_{\text{Cost function being minimized}} dt$$

Two-thirds Power Law
(Viviani and Flash, 1995)

$$\underset{\text{Tangential Velocity}}{V(t)} = \underset{\text{Velocity Gain Factor } (>0)}{K(t)} \left(\frac{R(t)}{1 + \alpha \underset{\text{Radius of Curvature}}{R(t)}} \right)^{\overset{\text{Empirical value } \frac{2}{3}}{\beta}}$$

Decoupled Minimum-Jerk
(Huber et al., 2009)

$$\underset{\text{Trajectory in z-direction}}{r_z(t)} = \sum_{k=0}^{5} a_{kz} t^k$$

$$\underset{\text{Trajectory in xy-direction}}{r_{xy}(t)} = \sum_{k=0}^{5} a_{kxy} t^k$$

Recommended Reading

A. Akinade, Y. Haile, N. Mutangana C. Tucker, and D. Vernon, "Culturally Competent Social Robots Target Inclusion in Africa", Science Robotics, 2023.

http://vernon.eu/publications/2023_Akinade_et_al.pdf

D. Vernon, "An African Perspective on Culturally Competent Social Robotics: Why DEI Matters in HRI", IEEE Robotics and Automation Magazine, accepted for publication.

http://vernon.eu/publications/2024_Vernon.pdf