上海科技大学

ShanghaiTech University

CS283: **Robotics Spring 2026**

Sören Schwertfeger

ShanghaiTech University

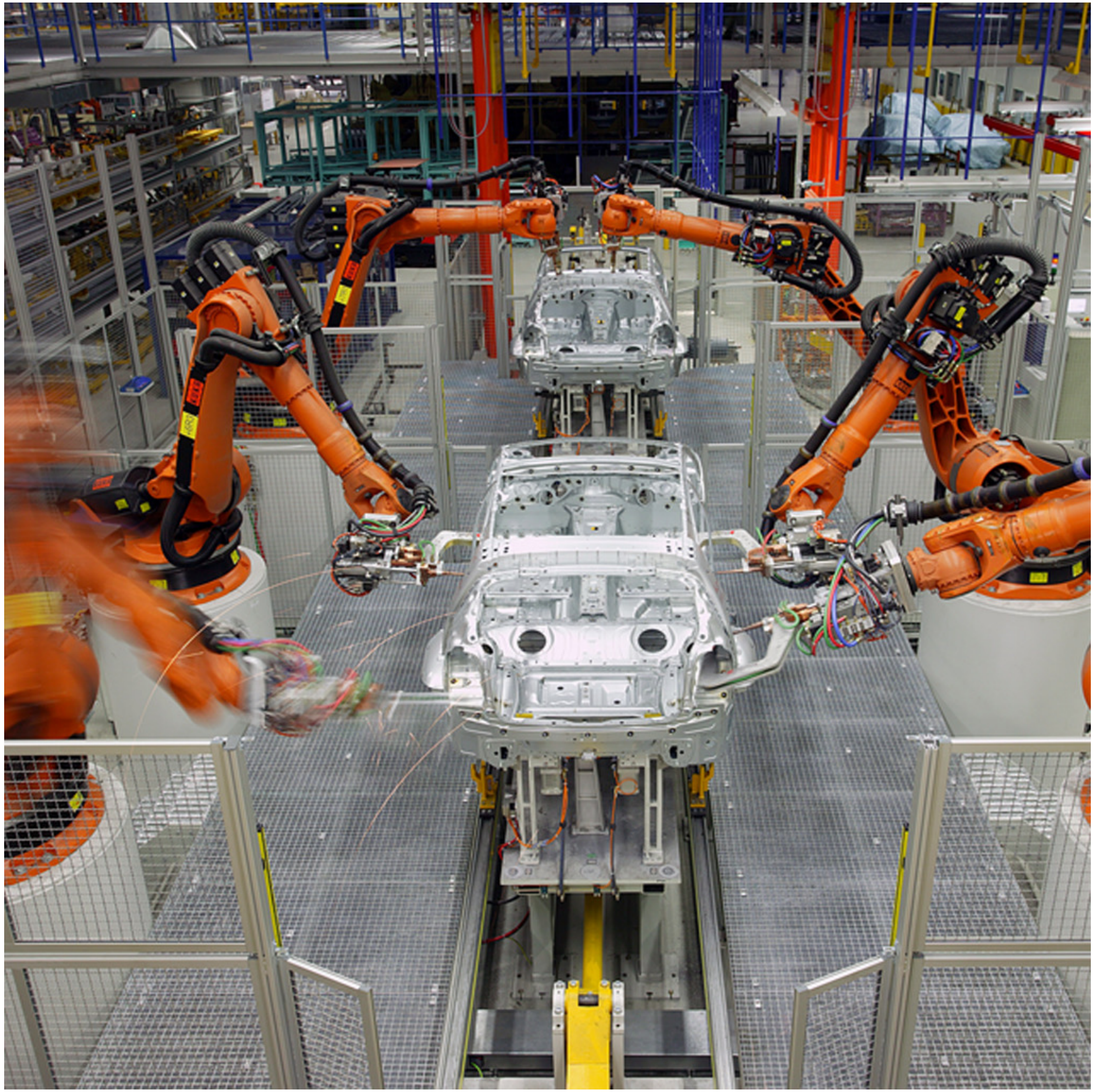
STAR CENTER

Outline

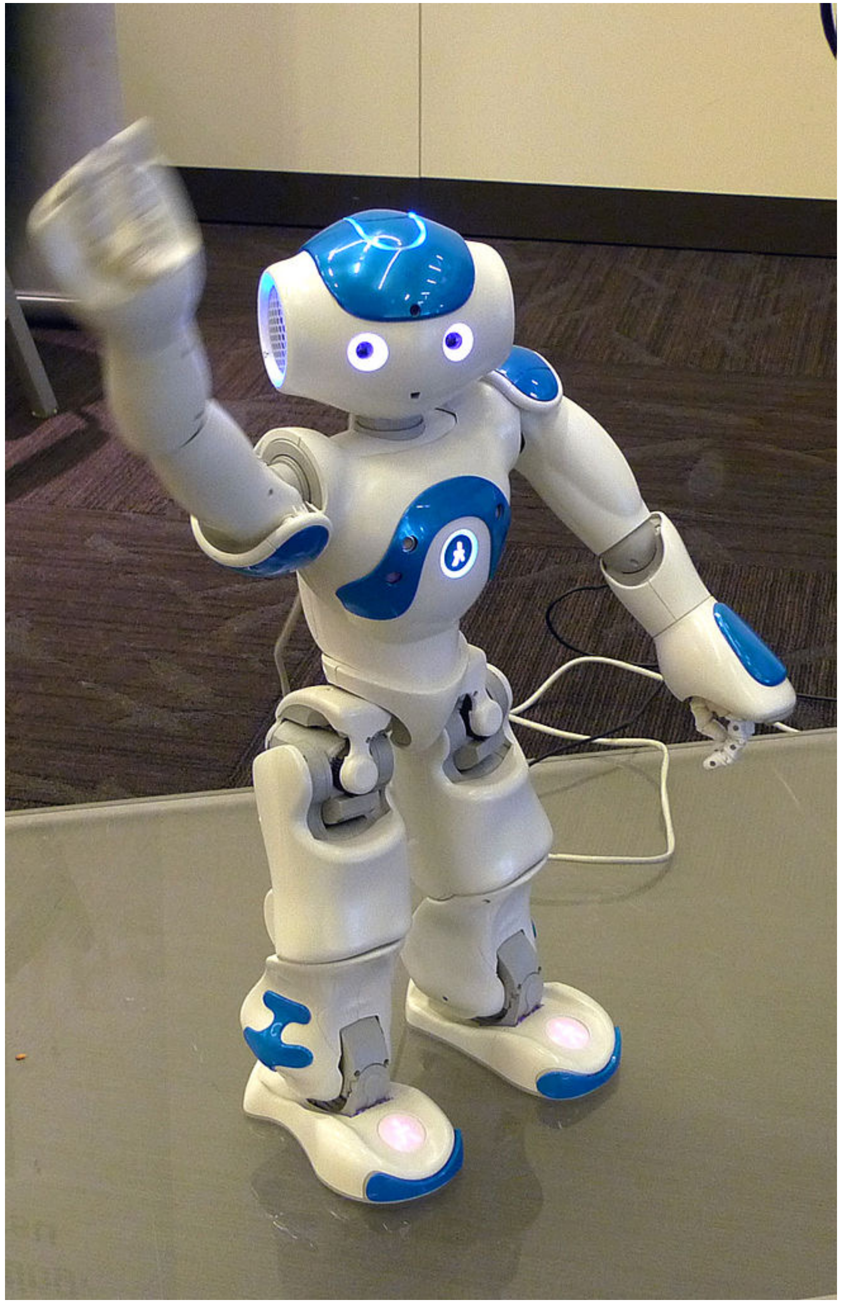
- What is a Robot?
- Why Mobile Robotics?
- Why Autonomous Mobile Robotics?
- Course Overview
- Brief History
- Software

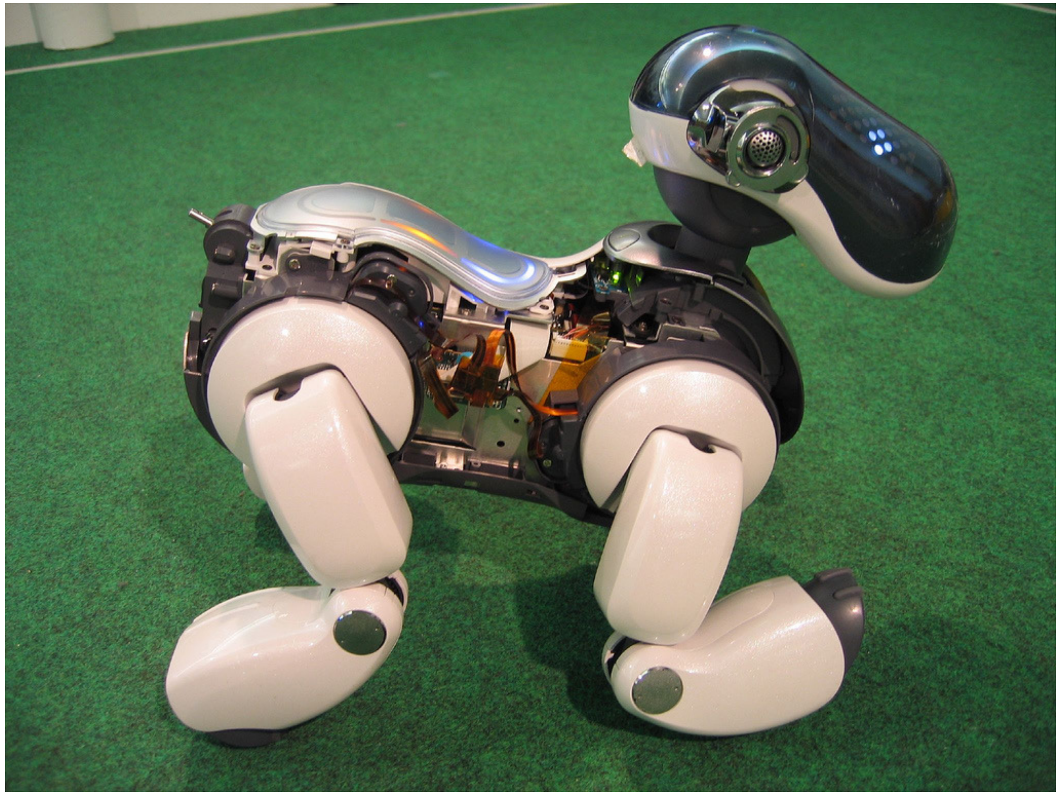
What is a Robot?

Pictures on the following slides all from http://commons.wikimedia.org



ASIMO
HONDA

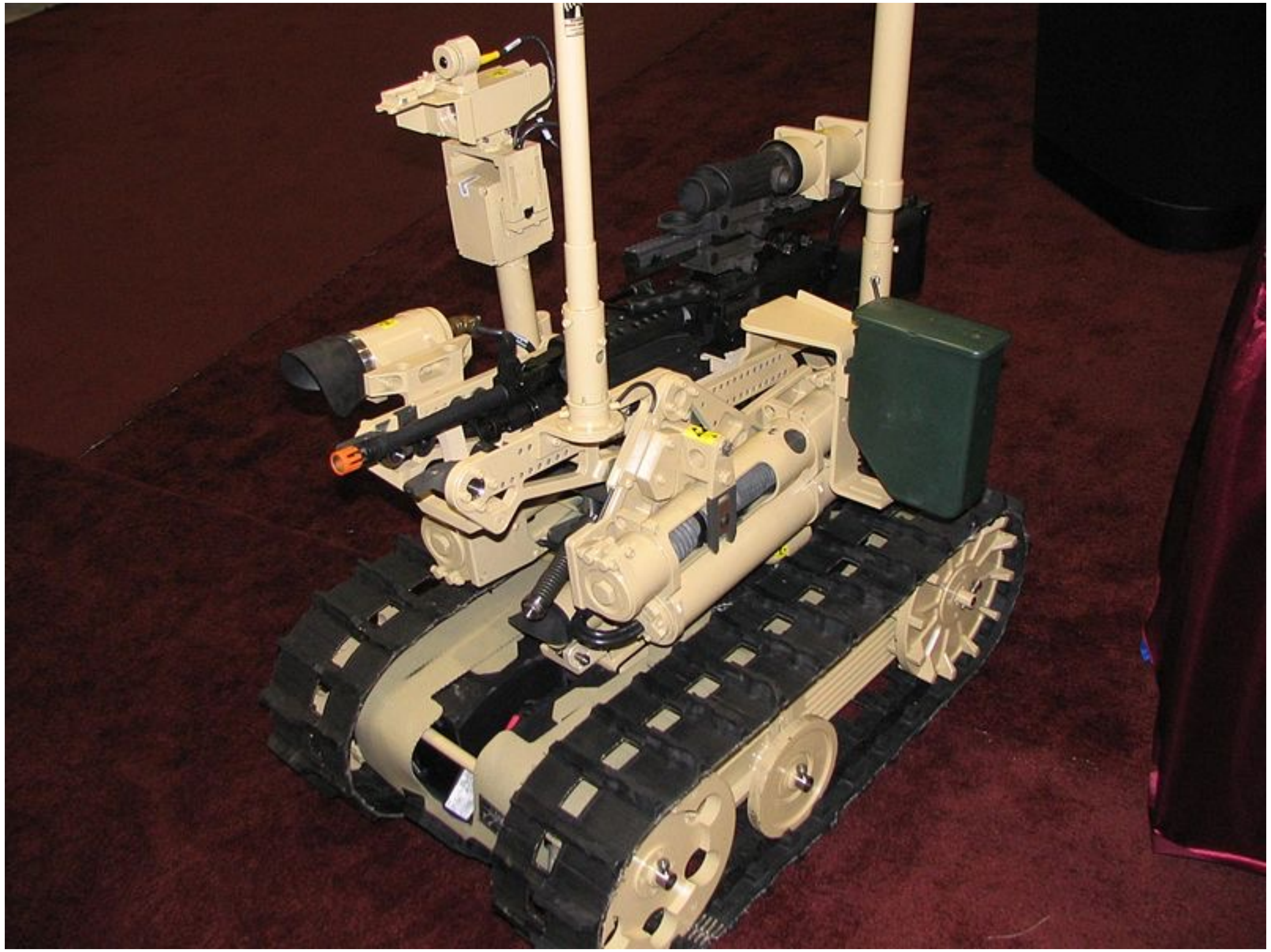




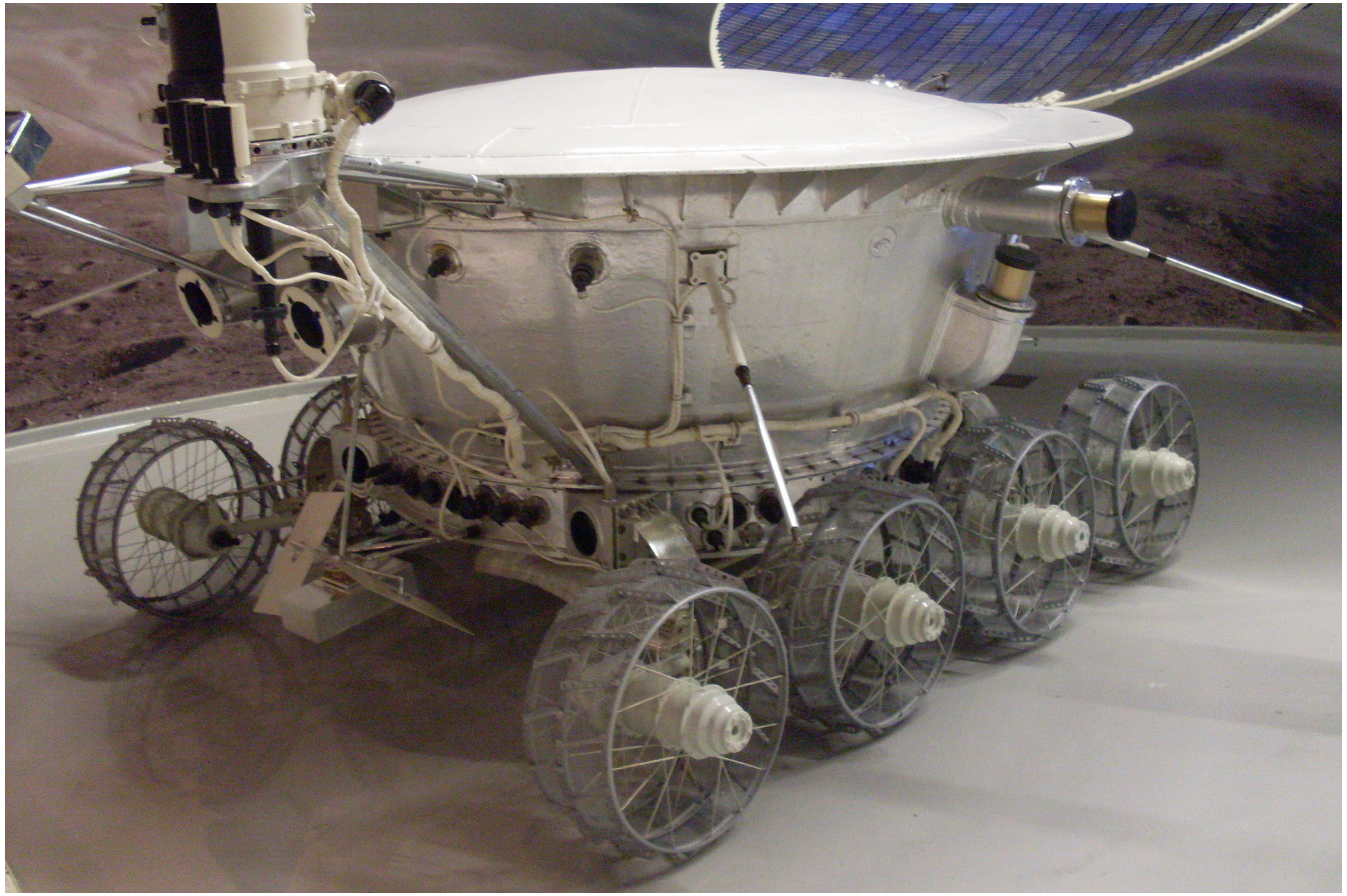
KEEP CLEAR

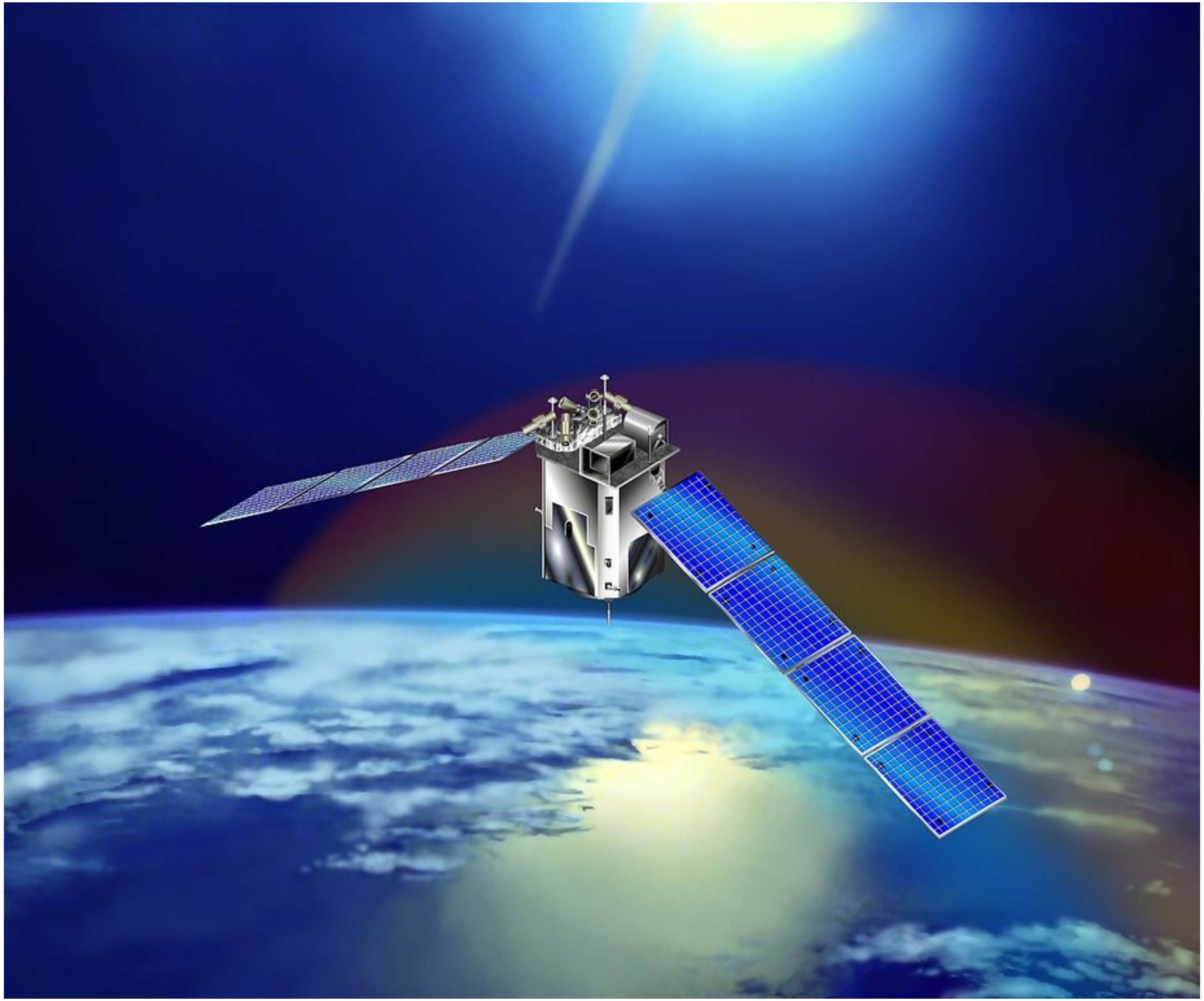
iRobot
Roomba™
Robotic FloorVac
S
M
L
Select Room Size to Start / Stop
Always empty both sides of Particle Bin after each use

Robotics Technology
Development and Commercialization
Unmanned Vehicle
Systems Engineering
Autonomous
Navigation Systems
NREC



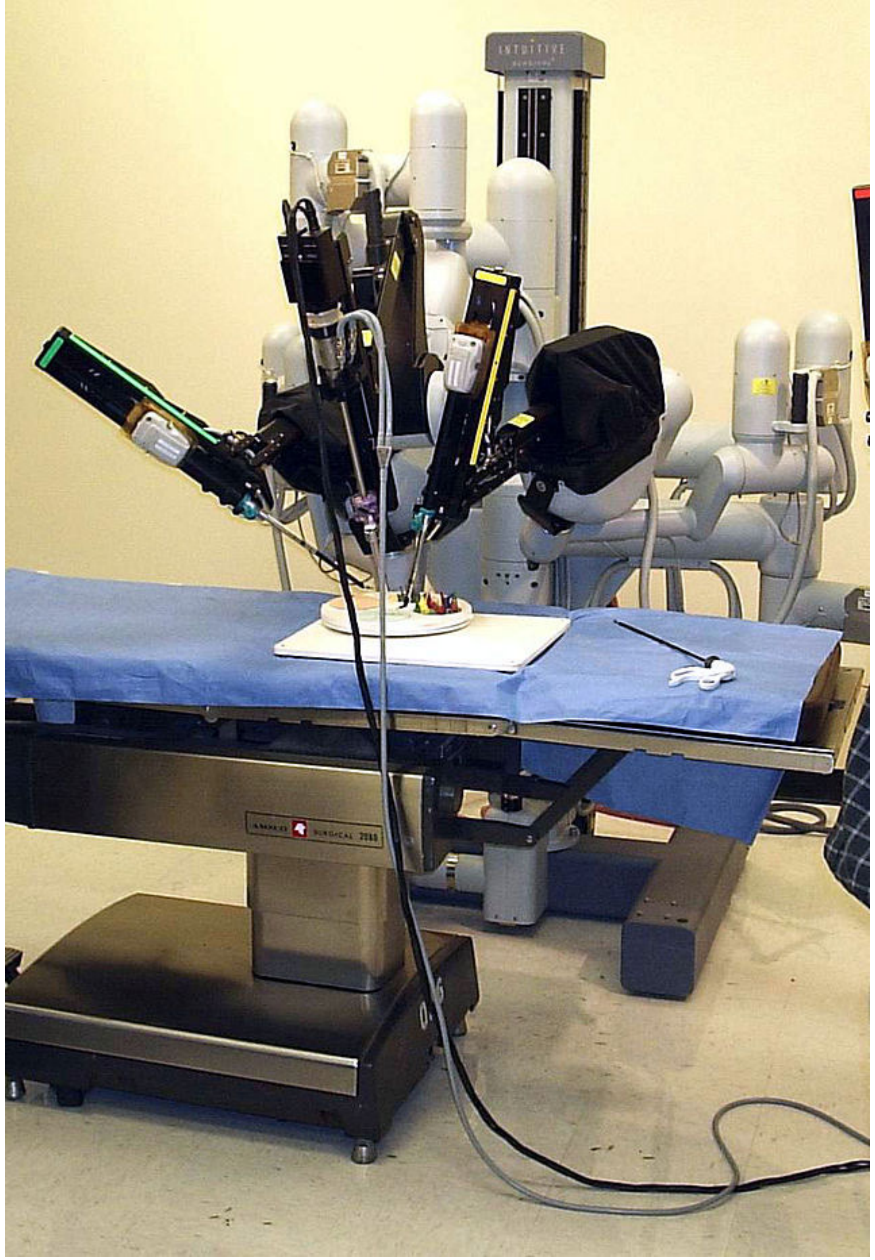
U.S. AIR FORCE
GLOBAL HAWK
HIGH ALTITUDE LONG ENDURANCE
UNMANNED AERIAL VEHICLE SYSTEM
Full Scale Mock-up
NORTHROP GRUMMAN



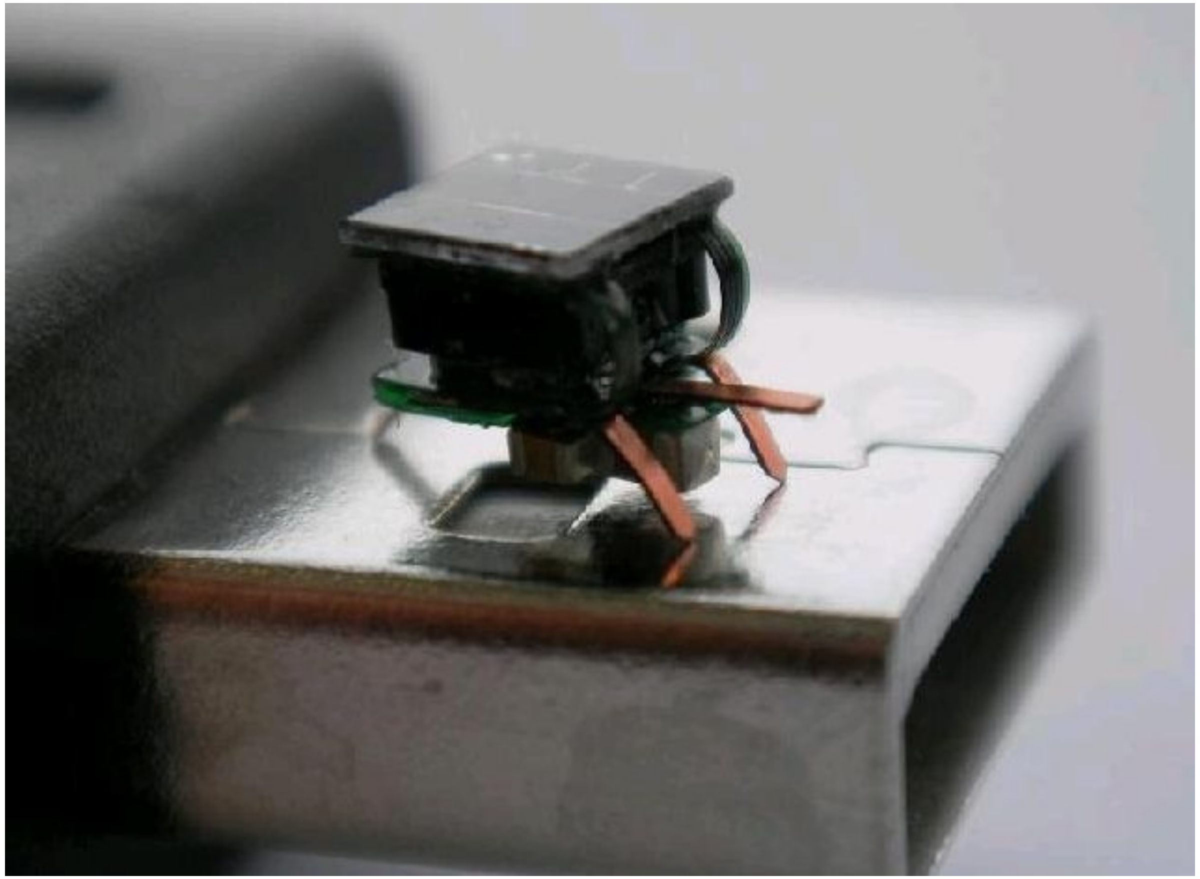


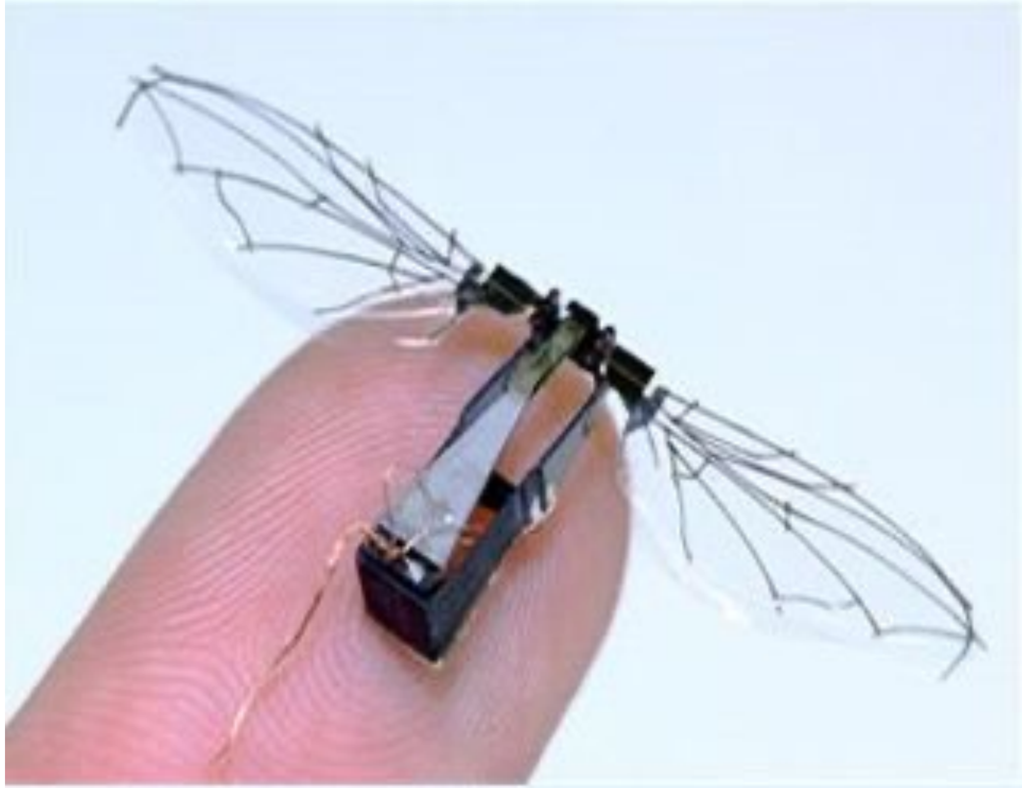
EXPLOSIVE

JASON
WOODS HOLE
OCEANOGRAPHIC INSTITUTION


INTUITIVE

P
Electric vehicle recharging point only
At any time
TESLA
TESLA
Electric vehicle recharging point only
At any time
TESLA
Electric vehicle recharging point only
At any time
TESLA
Electric vehicle recharging point only
At any time
GB





Recent prototype of the Harvard Microrobotic Fly, a three-centimeter wingspan flapping-wing robot.

Image Credit: Ben Finio, The Harvard Microrobotics Lab

Prof‘s Definition: A Robot is ...

A machine
capable of performing complex tasks
in the physical world,
that is using sensors to perceive the environment
and acts tele-operated or autonomous.

International Organization for Standardization: ISO 8373 Definition

- https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en
- Robot
  - actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks
- Industrial Robot
  - automatically controlled,
  - reprogrammable,
  - multipurpose
  - manipulator,
  - programmable in three or more axes
  - which can be either fixed in place or mobile for use in industrial automation applications
- Service Robot
  - robot that performs useful tasks for humans or equipment excluding industrial automation applications

What is a robot – and what is not?

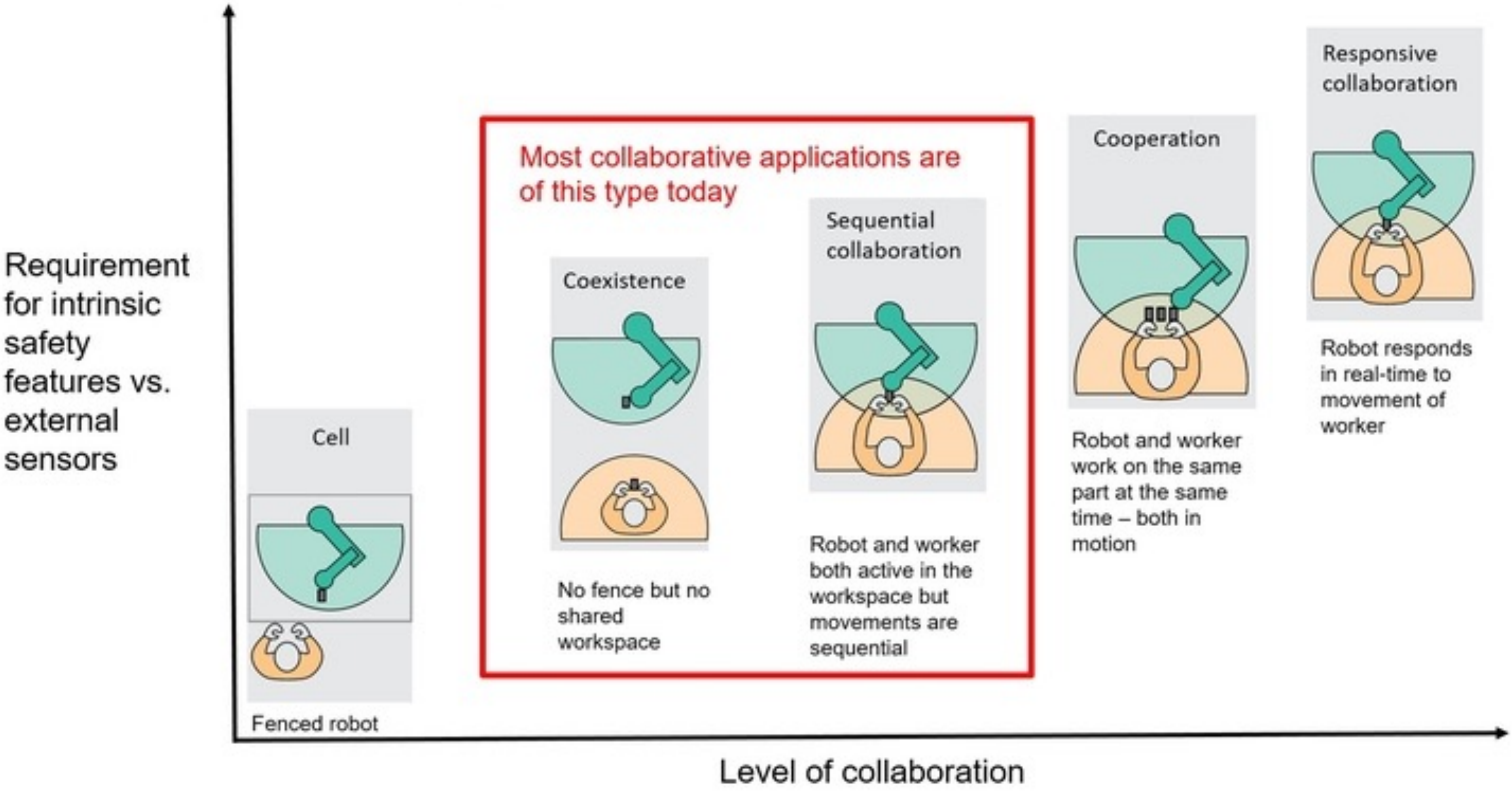
IFR International Federation of Robotics

Image: Yaskawa

- “**Robot**” defined by **International Standards Organization** ISO

- **No robots:**
  - ⊗ software (“bots”, AI, Robotic Process Automation-RPA)
  - ⊗ remote-controlled drones, UAV, UGV, UUV
  - ⊗ autonomous cars
  - ⊗ ATMs, smart washing machines, etc.
  - ⊗ voice assistants

Types of collaboration with industrial robots



Requirement for intrinsic safety features vs. external sensors
Level of collaboration

Cell

Fenced robot

Most collaborative applications are of this type today

Coexistence

No fence but no shared workspace

Sequential collaboration

Robot and worker both active in the workspace but movements are sequential

Cooperation

Robot and worker work on the same part at the same time – both in motion

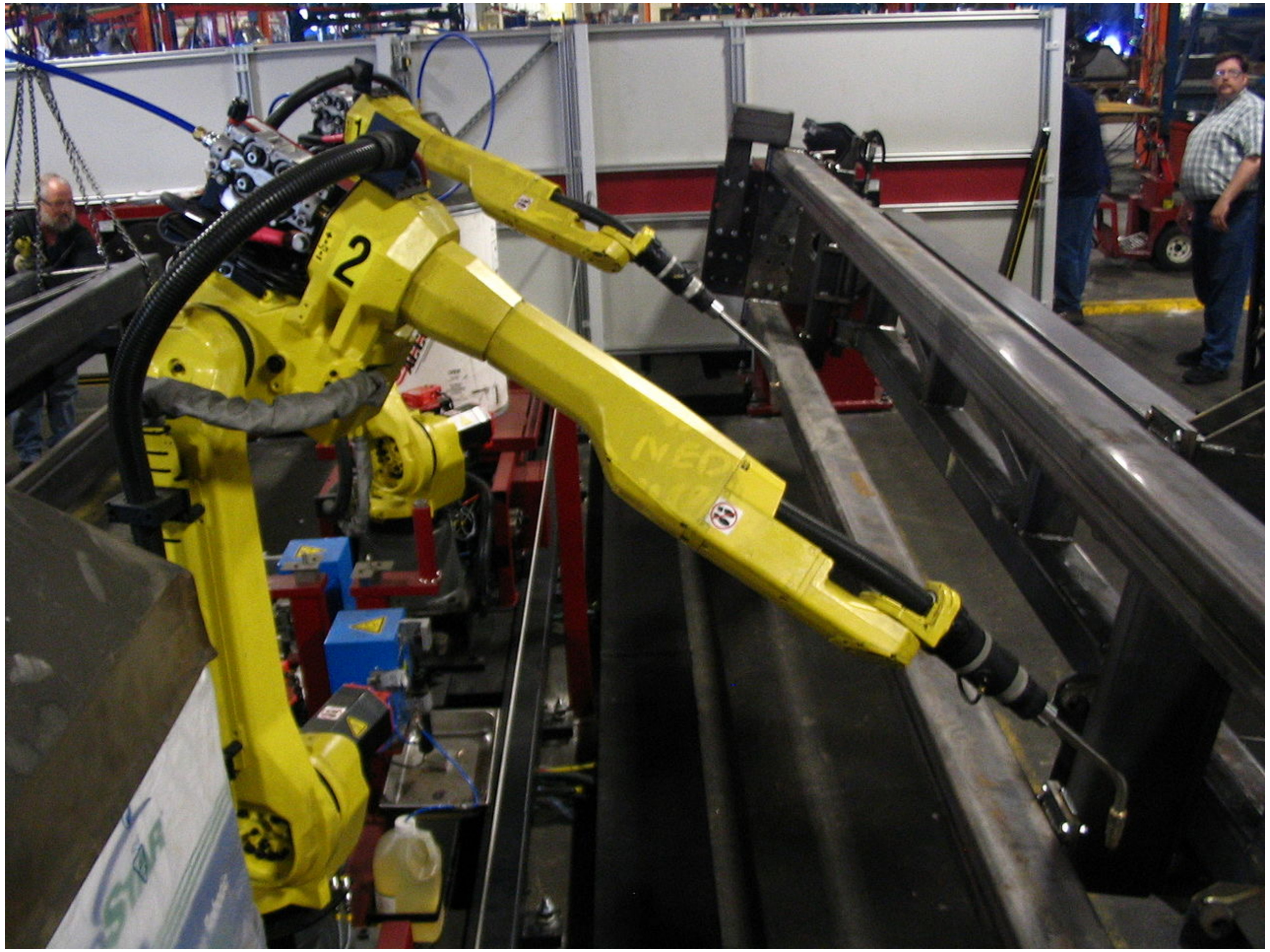
Responsive collaboration

Robot responds in real-time to movement of worker

Green area: robot's workspace; yellow area: worker's workspace

Source: IFR (classification), adapted and modified from Bauer et al. (2016).

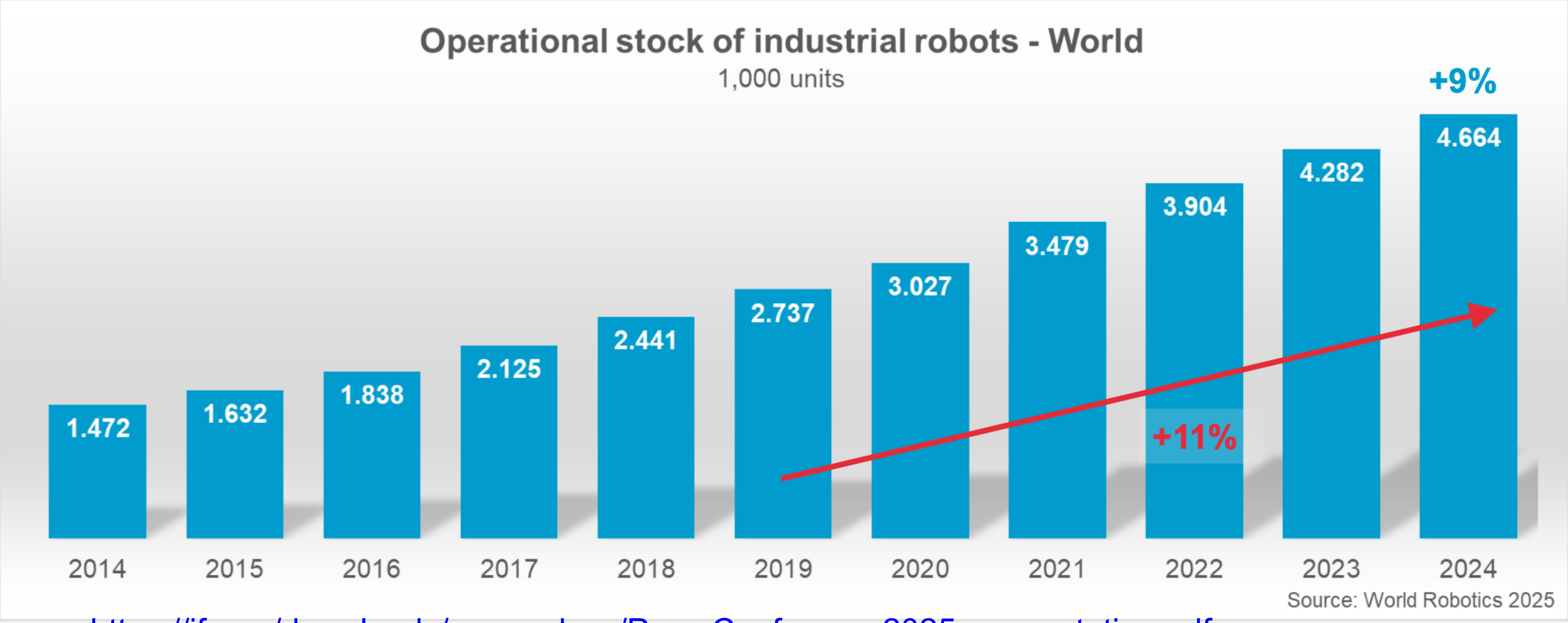
Industry vs Mobile Robots



- Industrial Robots rule:
    - 2021: 3.5 million industrial robots installed
    - Over 0.5 million new robots that year!
    - China biggest robot market regarding annual sales - also fasted growing market worldwide
- Industrial Robots stay at one place!
- Almost all other robots move => **Mobile Robotics**

Continuous increase in operational stock

IFR International Federation of Robotics



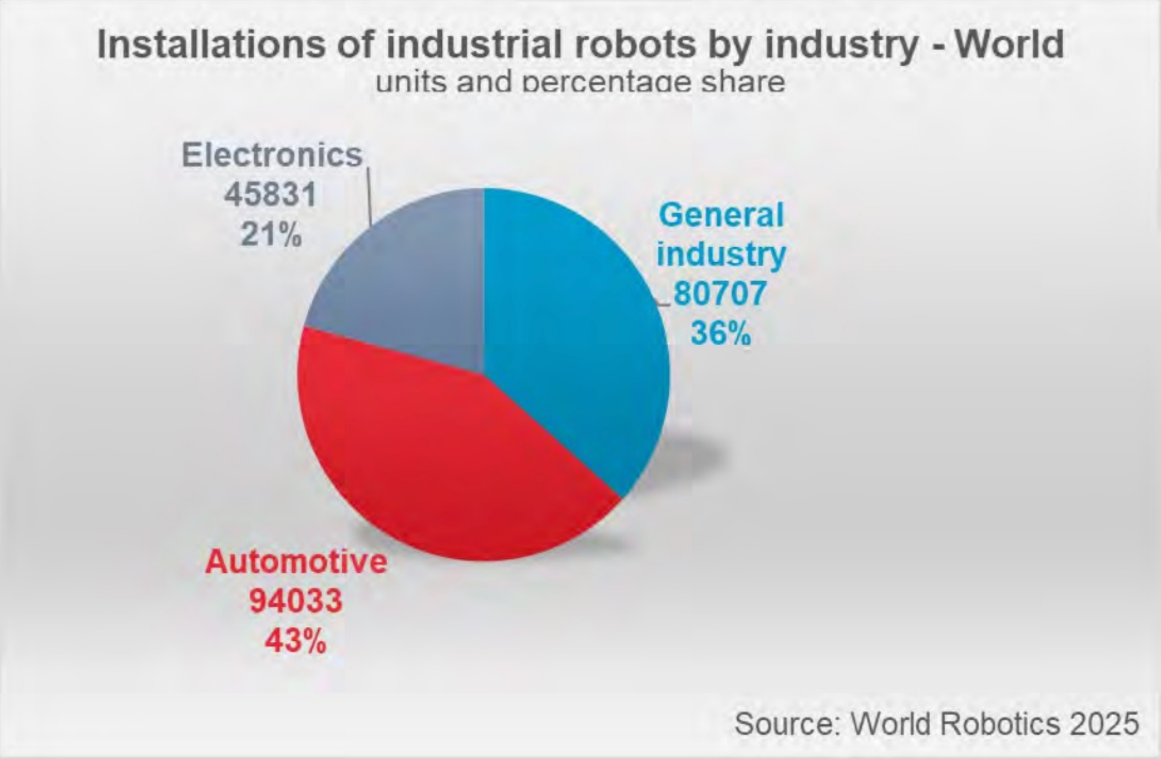
Source: World Robotics 2025

https://ifr.org/downloads/press_docs/PressConference2025_presentation.pdf

General industries are growing their market share

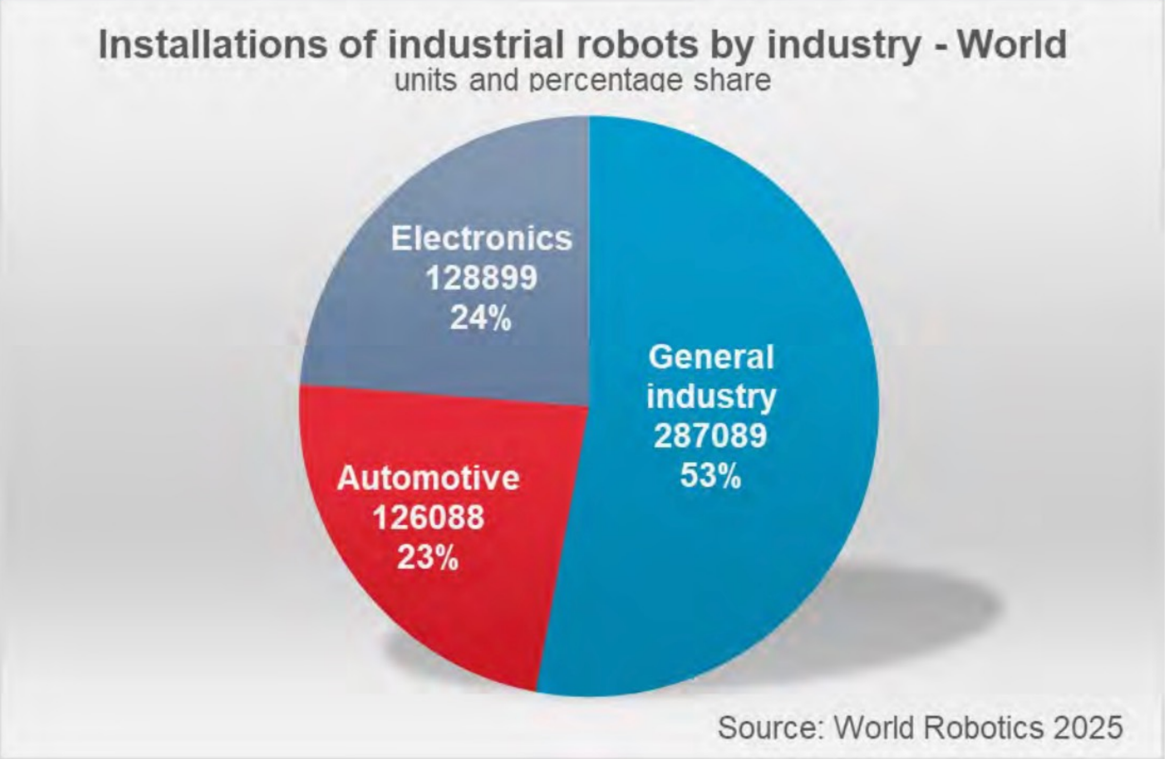
IFR
International
Federation of
Robotics

Shares of majors industries in 2014



Source: World Robotics 2025

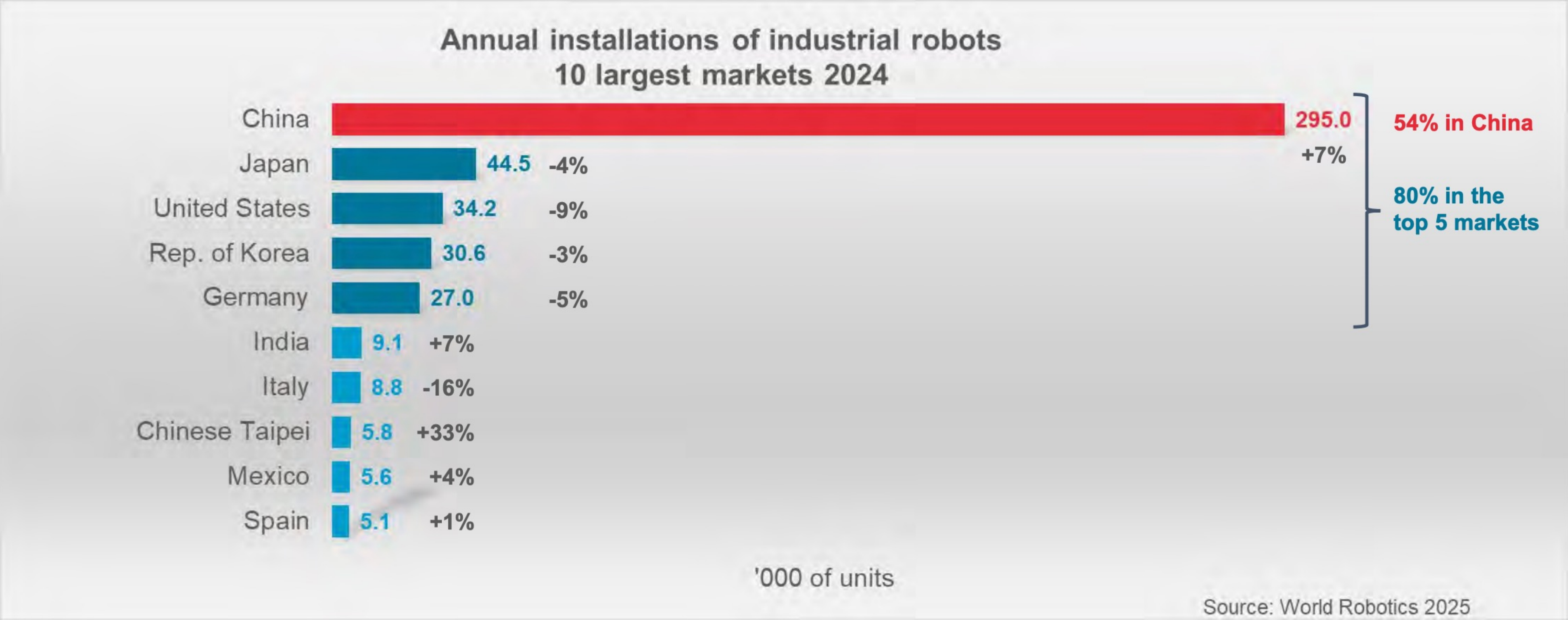
Shares of majors industries in 2024



Source: World Robotics 2025

Top 10 markets - new installations

IFR International Federation of Robotics

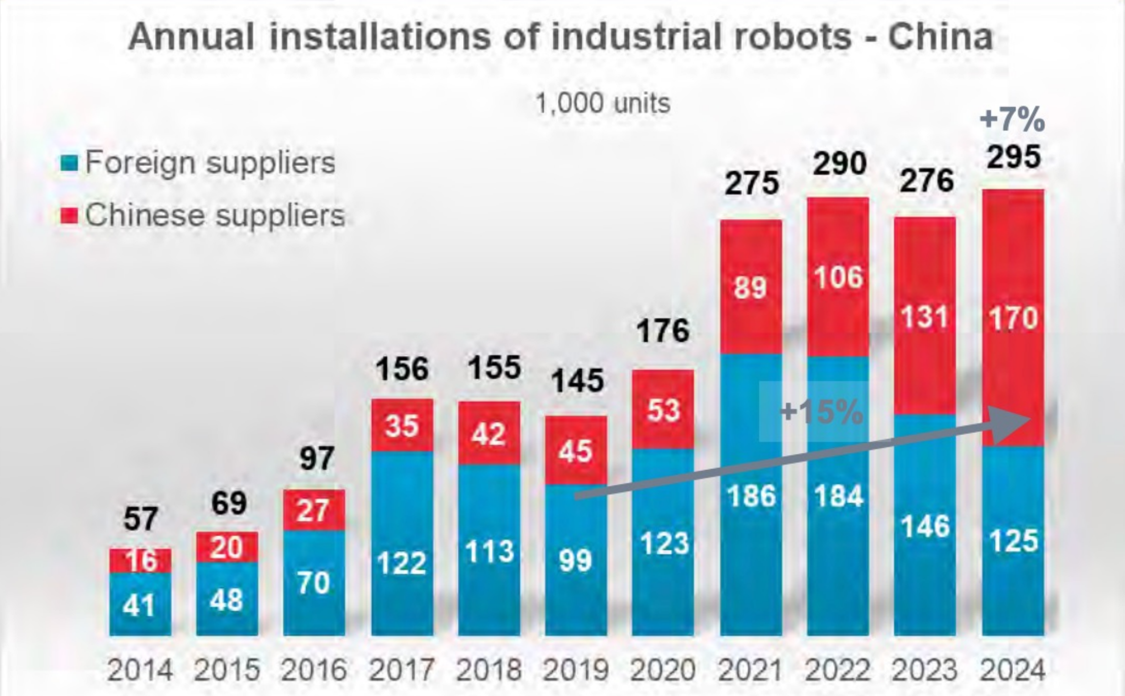


Source: World Robotics 2025

Growth in China comparable to global growth

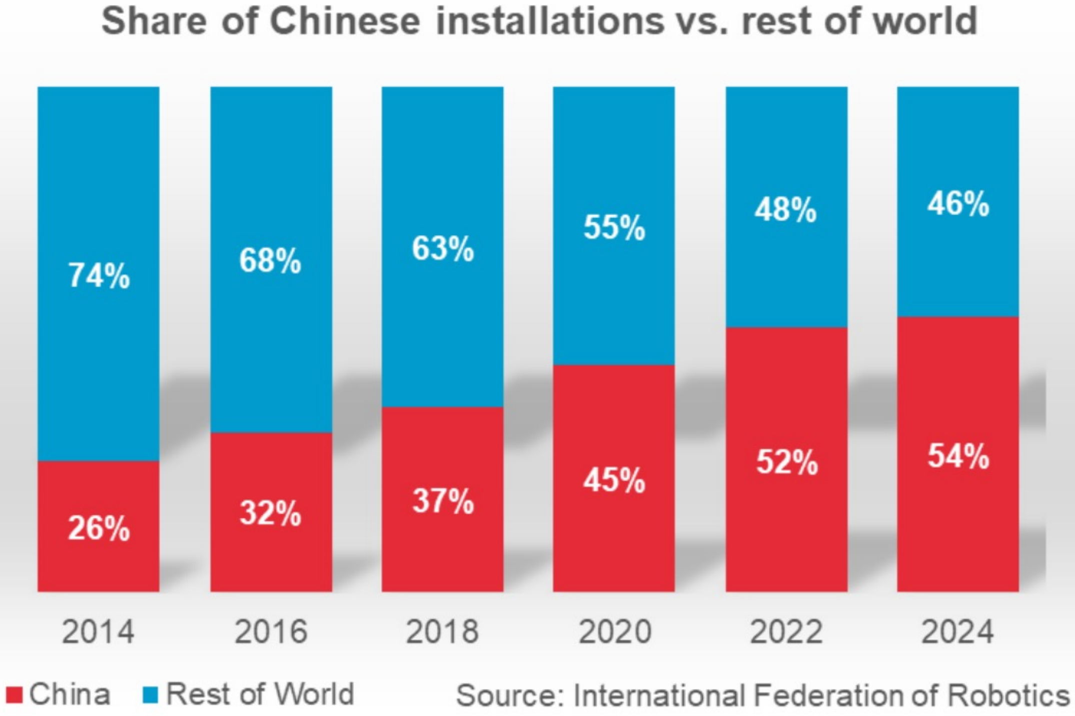
IFR International Federation of Robotics

Share of Chinese suppliers at 57%

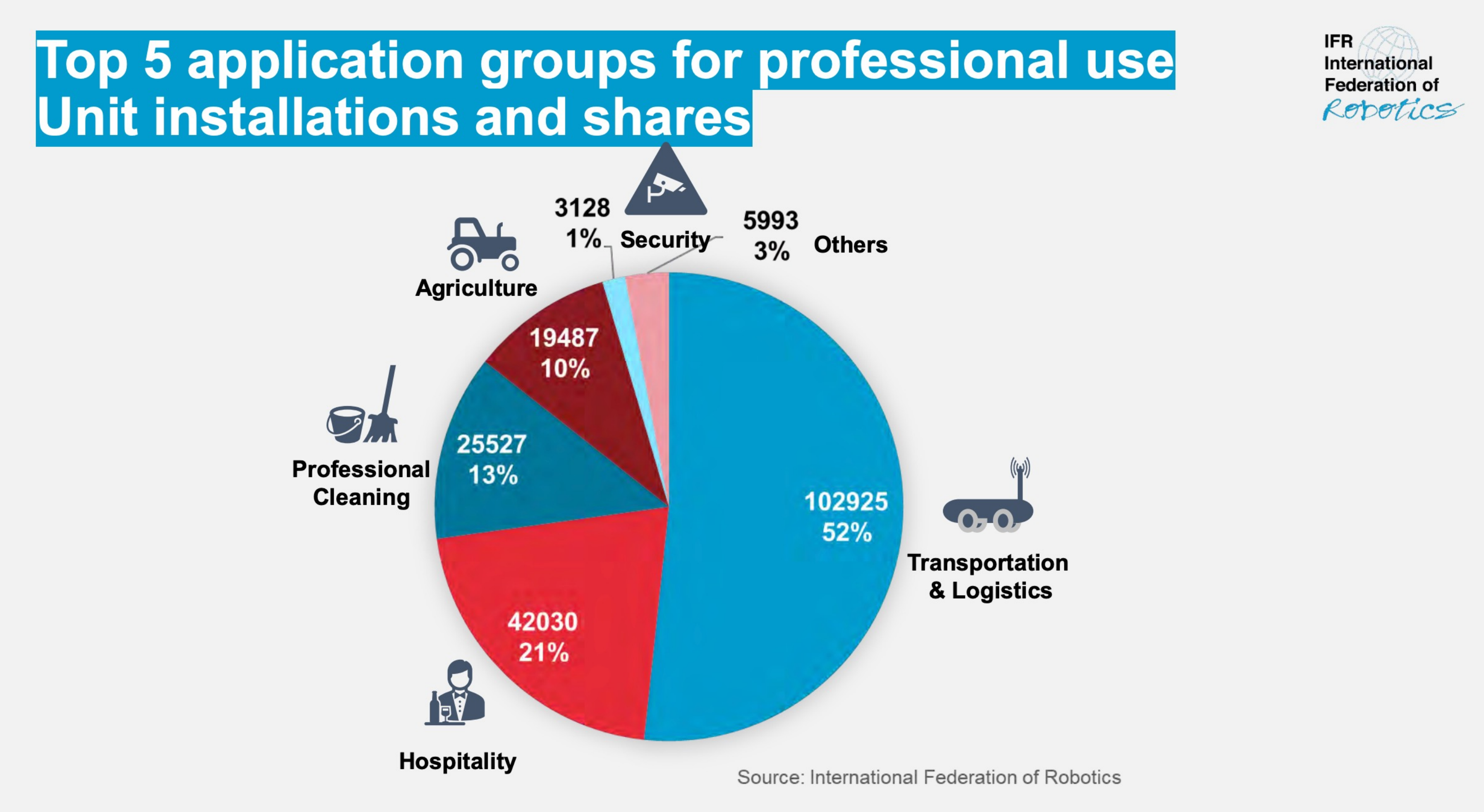


Source: World Robotics 2025

Increasing market shares of China



Source: International Federation of Robotics



Top 5 application groups for professional use
Unit installations and shares

IFR International Federation of Robotics

Source: International Federation of Robotics

Medical robots growing strongly: 16,700 new units - +91%

IFR International Federation of Robotics

Surgery

Image: Kawasaki

+41%

Rehabilitation & non-invasive therapy

Image: Life Science Robotics

+106%

Diagnostics & Medical laboratory analysis

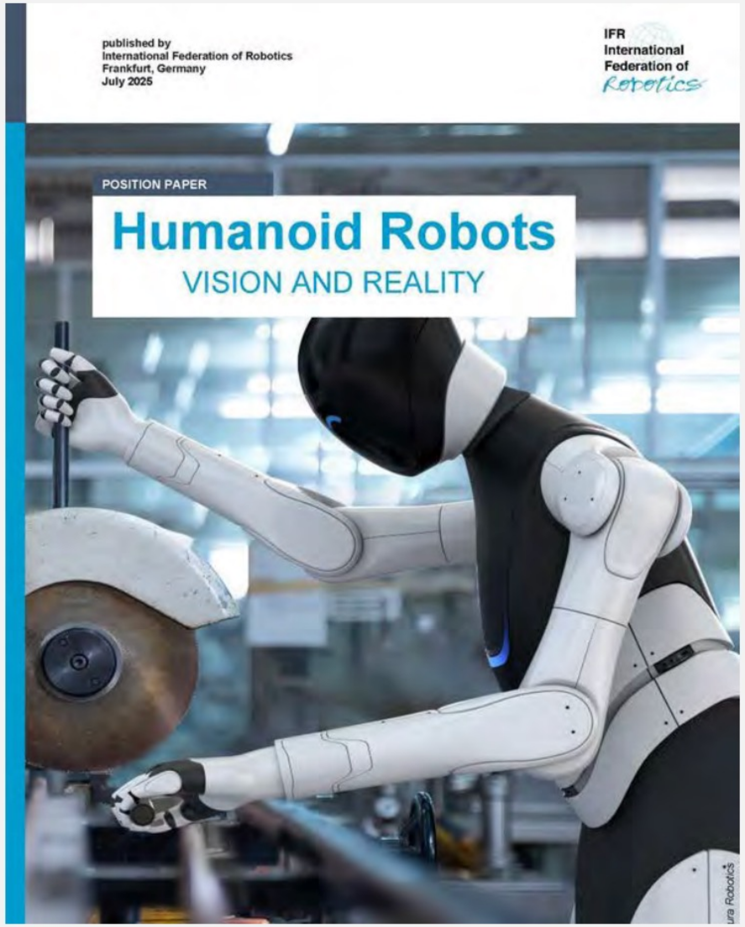
Image: ABB Robotics

+610%

Humanoid robots

IFR International Federation of Robotics

- There is no massive use today but **serial production in preparation**
- **Manufacturers** are building **humanoid robots** for **R&D purposes** for several years and producing **on demand**
- **New companies** in the market do build **humanoids** at a **demonstrator** or **prototype stadium** for first trial applications
- **Application fields** of humanoids still have to be determined and proven in practice



published by
International Federation of Robotics
Frankfurt, Germany
July 2025

IFR International Federation of Robotics

POSITION PAPER

Humanoid Robots

VISION AND REALITY

Why **Autonomous** Mobile Robotics?

- Tele-operating robots: boring and inefficient
- Autonomous robots: Robots that act by their own reasoning
    - Human operator might be present: Gives high level tasks
- Why autonomy?
    - Autonomous behaviors might be **better** than remote control by humans
    - Remote control might be **boring** or **stressful** and **tiresome**
    - Human operators might be a **scarce** resource or **expensive**
    - Multi robot approaches: One operator for many robots
- Semi-autonomy:
    - Autonomous behaviors that help the operator, for example:
    - Way-point navigation, autonomous stair climbing, assisted manipulation
    - Gradual development from tele-operation to full autonomy possible

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- Where am I?
  - Global Positioning System: outdoor, error measured in meters
  - Guiding system: (painted lines, inductive guides), markers, iBeacon
  - Model of the environment:
    - Map, Localize yourself in this model
    - Mapping: Build the map while driving

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- Where is my goal?
- Two part problem:
  - What is the goal?
    - Expressed using the world model (map)
    - Using object recognition
    - No specific goal (random)
  - Where is that goal?
    - Coordinates in the map
    - Localization step at the end of the object recognition process
    - User input

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - <u>They need to know **how** to get there.</u>

- Different levels:
  - Control:
    - How much power to the motors to move in that direction, reach desired speed
  - Navigation:
    - Avoid obstacles
    - Classify the terrain in front of you
    - Follow a path
  - Planning:
    - Long distance path planning
    - What is the way, optimize for certain parameters

Most important capability

(for autonomous mobile robots)

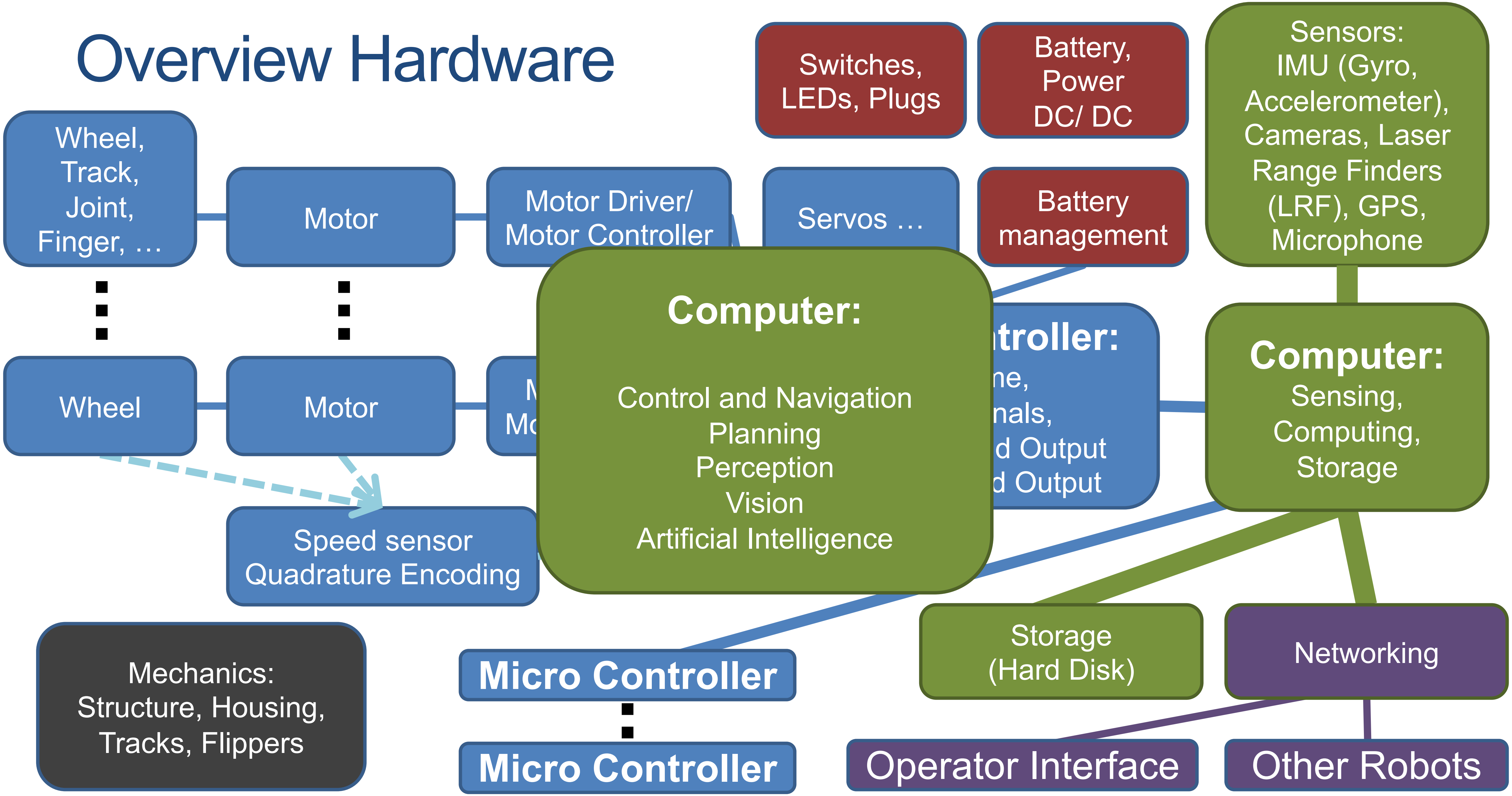
How to get from place A to place B?

(safely and efficiently)

How to get from A to B?

What are the components of a ROBOT?

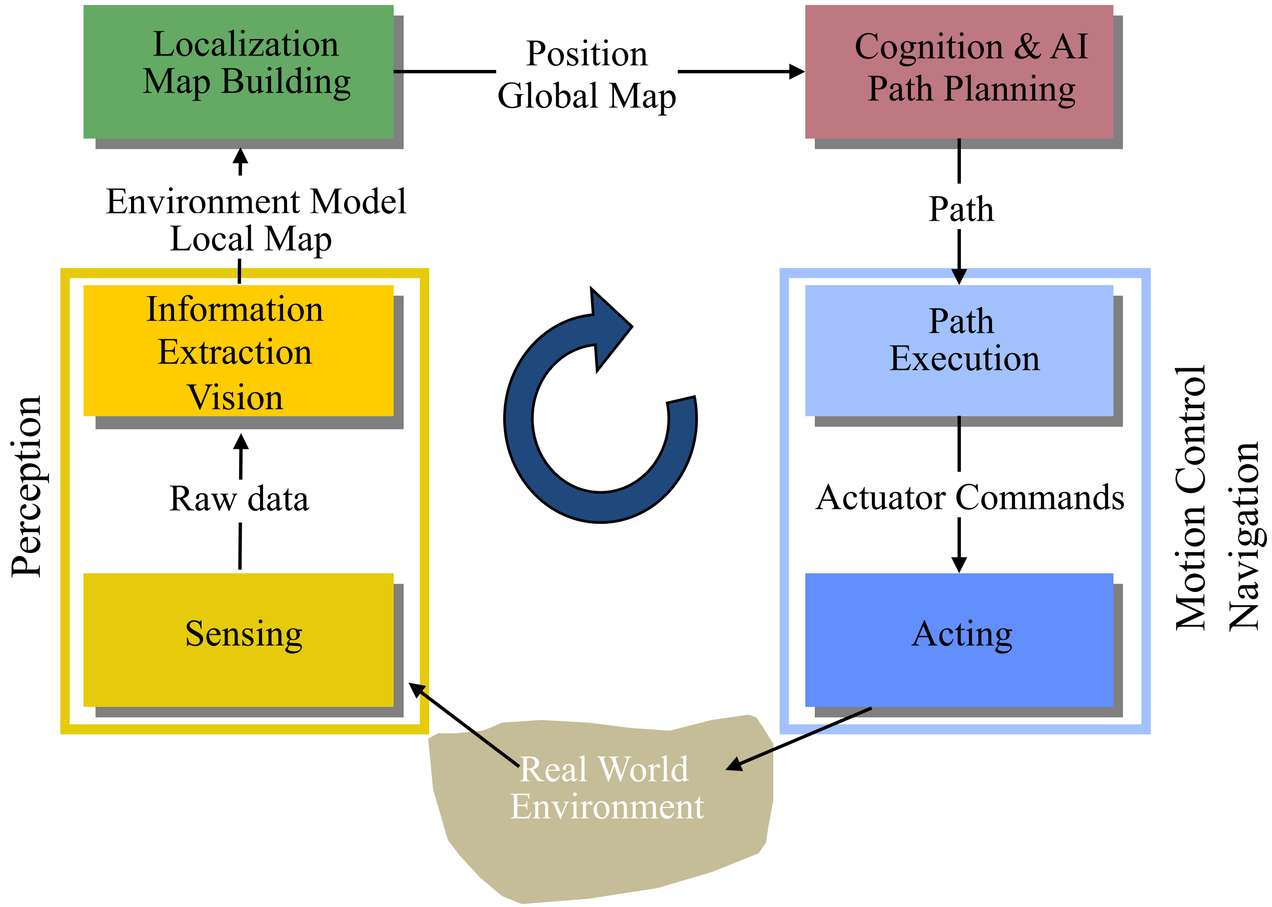
Overview Hardware



How to get from A to B?

How to program an intelligent ROBOT to go from A to B?

General Control Scheme for Mobile Robot Systems



With material from Roland Siegwart and Davide Scaramuzza, ETH Zurich

ADMINISTRIVIA

Teaching Plan

- Lectures

- Homework
  - Including one real Hardware Group Project

- Presentation about robotics paper (related to your project)

- Midterm and Final Exams

- Project…

Course Contents

- Robotics CS283 is about
  - Algorithms and software for
  - Mobile robots
  - Especially w.r.t:
  - Perception
  - Mapping/ SLAM
  - Navigation/ Planning/ Control
  - Autonomy

- Mobile Manipulation CS289:
  - Emphasis on manipulation
  - CS283 has double the credit points for project – higher emphasis – more work – better results expected!

Project

- 2 credit points!
- Work in groups, min 2 students, max 3 students!
- Next lecture: Topics will be proposed…
  - You can also do your own topic, but only after approval of Prof. Schwertfeger
    - Prepare a short, written proposal till next Tuesday!

- Topic selection: in 1-2 weeks!
  - One member writes an email for the whole group to Bichi: zhangbch2025 (at) shanghaitech.edu.cn ; Put the other group members on CC
  - Subject: [Robotics] Group Selection

- One graduate student from my group will co-supervise your project
- Weekly project meetings!

- Oral ”exams” to evaluate the contributions of each member
- No work on project => bad grade of fail

Grading

- Grading scheme is not 100% fixed
- Approximately:

Lecture:	50%	
Homework:		20%
Midterm:		10%
Final:		20%
Project:	50%	
Paper Presentation:		5%
Project Proposal:		5%
Intermediate Report:		5%
Weekly project meetings:		10%
Final Report:		10%
Final Demo:		10%
Final Webpage:		5%

Getting Help

- Piazza:
  - For discussions and announcements
  - https://piazza.com/shanghaitech.edu.cn/spring2025/cs283/home
  - Ask questions regarding your reading assignments and homework
  - You are not allowed to give the solutions – just guidance
- Ask questions during the lecture!
- Upon request we can organize a tutorial session
- Only if everything else fails: write e-mails
- Office Hours Prof. Schwertfeger: Tuesday afternoon
- Office Hours TA Bichi Zhang: make appointment via email

Policy on Plagiarism

- The homework are individual tasks!
- You may discuss the ideas and algorithms of homework with others but:
  - At no time should you read the source code or possess the source files of any other person, including people outside this course.
  - We will **detect plagiarism** using automated tools and will **prosecute** all violations to the fullest extent of the university regulations, including failing this course, academic probation, and expulsion from the university.

- Homework, project submissions, etc. will be submitted through git – using gitlab. You will get accounts on:
  - https://robotics.shanghaitech.edu.cn/gitlab

Mobile Robotics

- Topic Robots and how to program them:
  - Applications of robotics, software design, locomotion, hardware, sensing, localization, motion planning, autonomy for mobile robots, manipulation

- **Literature:**
- Mobile Robotics Mathematics, Models, and Methods
  - Alonzo Kelly
  - ISBN 978-1-107-03115-9
- Introduction to Autonomous Mobile Robots
  - Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza
  - ISBN: 978-0-262-01535-6

Material

- Webpage
  - https://robotics.shanghaitech.edu.cn/teaching/robotics2026 (currently down)
  - Slides will be available on the webpage
- Forum
- Where to find us:
  Office: SIST 1D 201.A
  Lab: SIST 1D 203
- E-Mail:
  - soerensch@ShanghaiTech.edu.cn
- Wechat group: >>>>>>>>

Group: Robotics 2026

Prerequisite: Robot Operating System 2 !

- Program in C++ (or python) and ROS 2 (https://docs.ros.org/)

- Recommended: Operating System Ubuntu Linux (www.ubuntu.com)
  - **Recommended option:** Dual boot on your own Laptop/ Computer – needs min. 40 GB from HD
  - Virtual Machine will perform poorly for some HW requiring to run a robot simulator

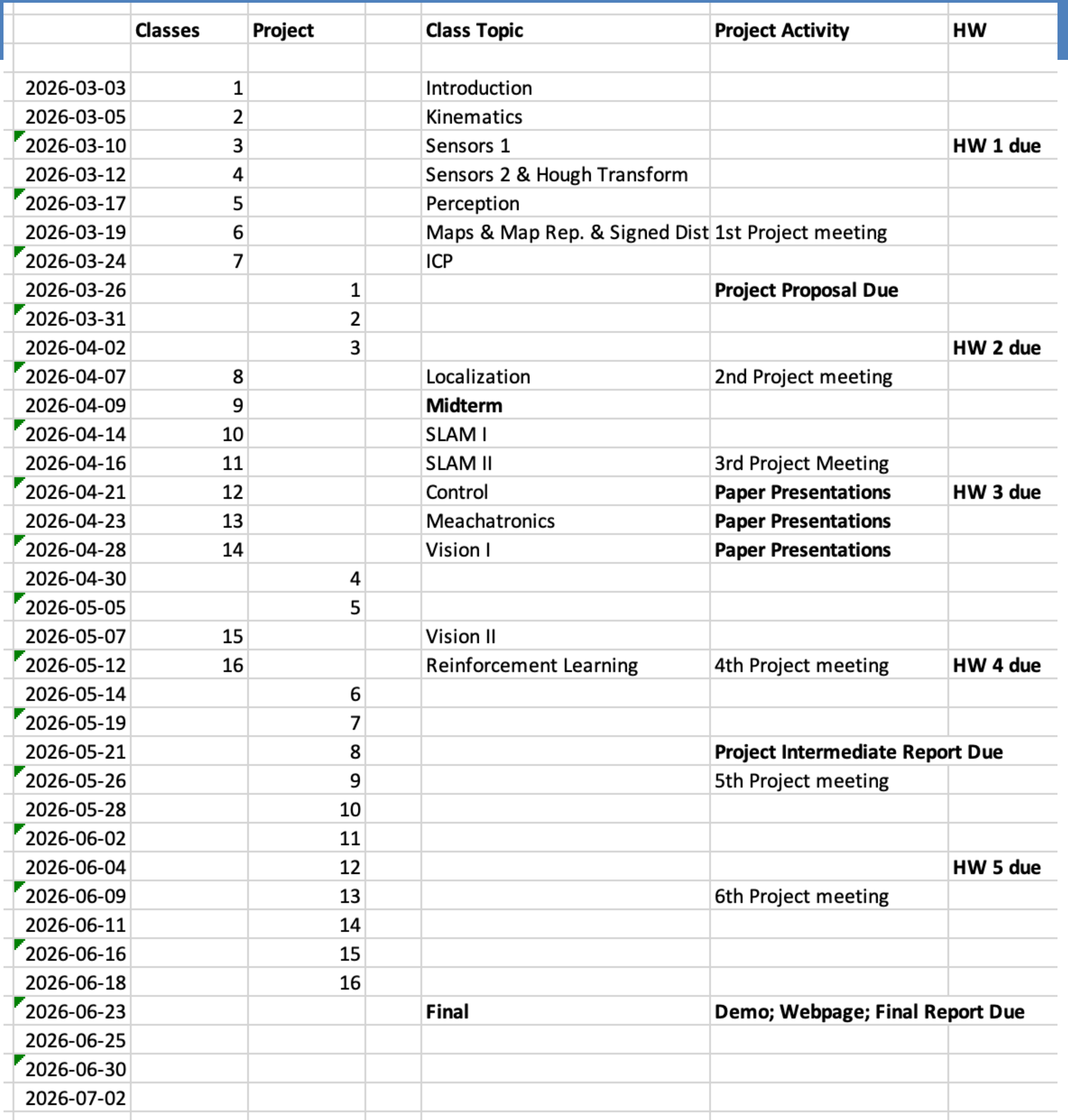
- ROS 2 version: We strongly suggest Humble – with Ubuntu 24.04 or 22.04

- ROS 2 also supports Windows and ~~MacOS~~ – but we will not offer any help/ user support for these – use at your own risk – Ubuntu/ Linux is suggested!
  - Certain libraries (e.g. pcl) are Linux only on ROS 2

- Other tools: git, LaTeX, …

Schedule

- May change – always check on webpage for most recent version!



	Classes	Project	Class Topic	Project Activity	HW
2026-03-03	1		Introduction		
2026-03-05	2		Kinematics		
2026-03-10	3		Sensors 1		**HW 1 due**
2026-03-12	4		Sensors 2 & Hough Transform		
2026-03-17	5		Perception		
2026-03-19	6		Maps & Map Rep. & Signed Dist	1st Project meeting	
2026-03-24	7		ICP		
2026-03-26		1		**Project Proposal Due**	
2026-03-31		2			
2026-04-02		3			**HW 2 due**
2026-04-07	8		Localization	2nd Project meeting	
2026-04-09	9		**Midterm**		
2026-04-14	10		SLAM I		
2026-04-16	11		SLAM II	3rd Project Meeting	
2026-04-21	12		Control	**Paper Presentations**	**HW 3 due**
2026-04-23	13		Meachatronics	**Paper Presentations**	
2026-04-28	14		Vision I	**Paper Presentations**	
2026-04-30		4			
2026-05-05		5			
2026-05-07	15		Vision II		
2026-05-12	16		Reinforcement Learning	4th Project meeting	**HW 4 due**
2026-05-14		6			
2026-05-19		7			
2026-05-21		8		**Project Intermediate Report Due**	
2026-05-26		9		5th Project meeting	
2026-05-28		10			
2026-06-02		11			
2026-06-04		12			**HW 5 due**
2026-06-09		13		6th Project meeting	
2026-06-11		14			
2026-06-16		15			
2026-06-18		16			
2026-06-23			**Final**	**Demo; Webpage; Final Report Due**	
2026-06-25					
2026-06-30					
2026-07-02					

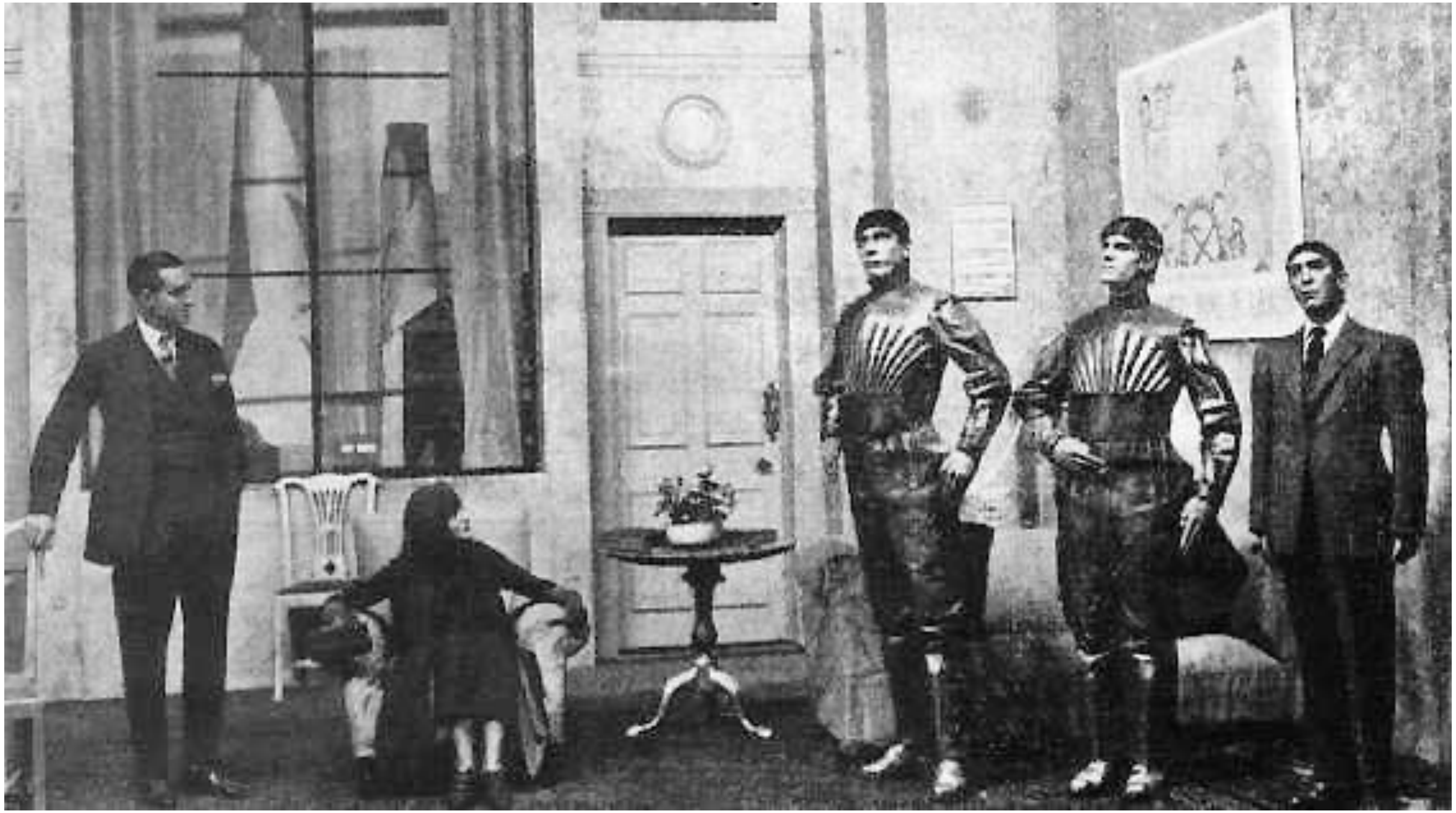
For this week

- Organize access to the two text books
- Do HW1: due Friday, March 10, 23:59
  - For the dual-boot installation of Ubuntu:
    - Backup your all your data
    - Free enough space (40 GB)
    - Download Ubuntu

BRIEF HISTORY

Brief History

Robota “forced labor”: Czech, Karel Čapek R.U.R. 'Rossum's Universal Robots' (1920).



Isaac Asimov - Three Laws of Robotics (1942)

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

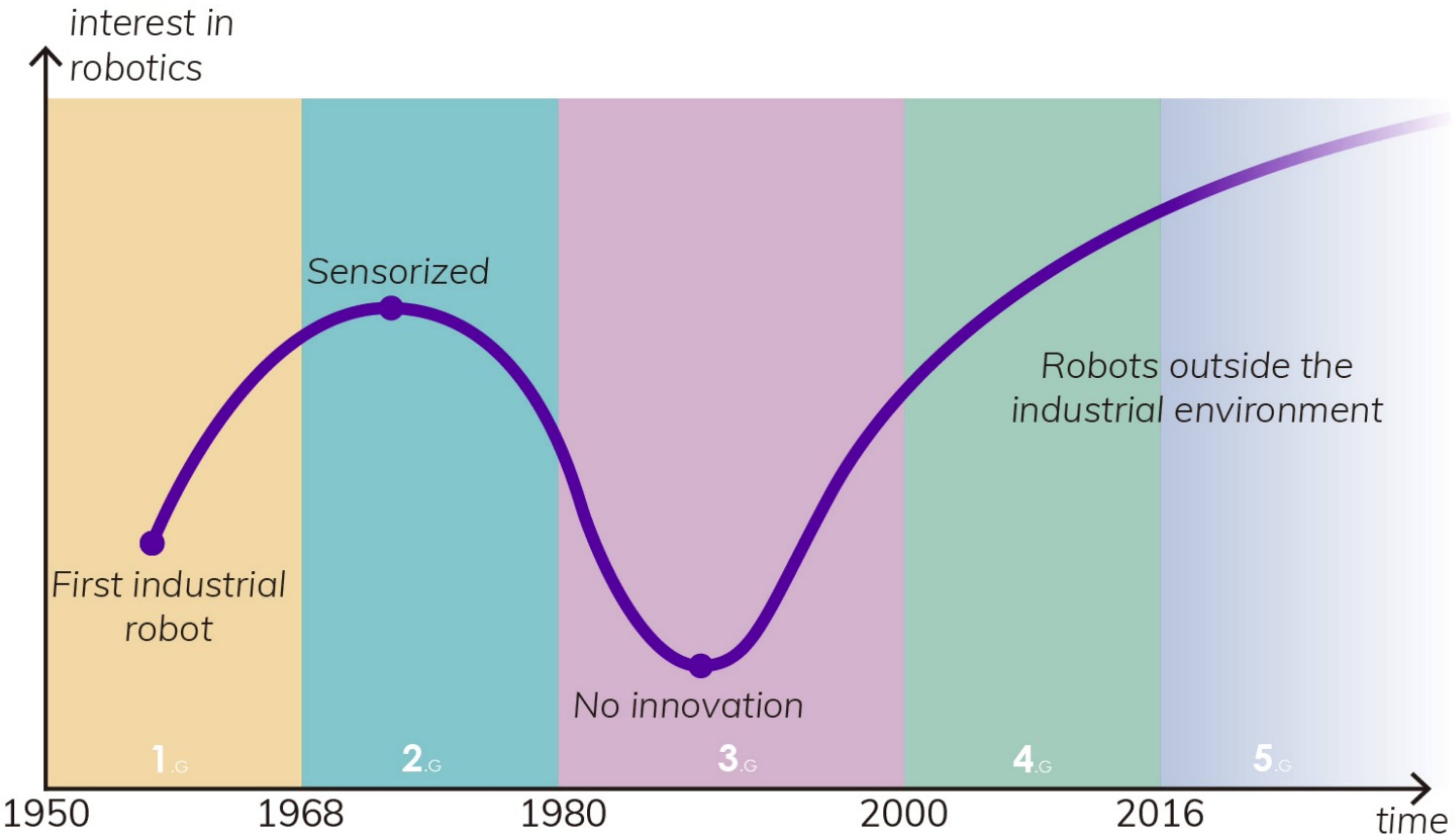
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

0. A robot may not harm humanity, or, by inaction, allow humanity to come to harm.

History

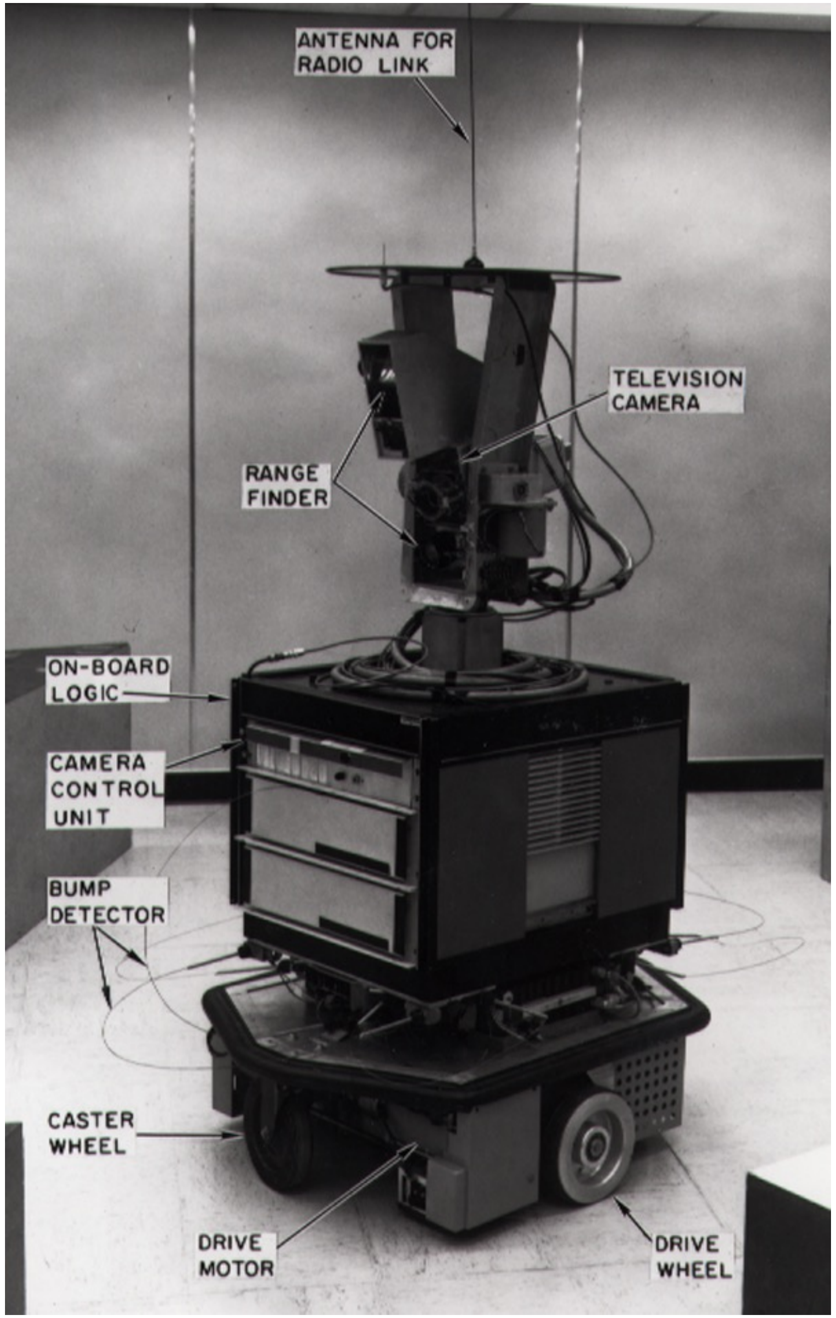
- First electronic autonomous robots 1949 in England (William Grey Walter, Burden Neurological Institute at Bristol)
  - three-wheeled robots: drive to recharging station using light source (phototaxis)
- Turing Test: 1950 (British mathematician Alan Turing)
- Unimate: 1961 lift hot pieces of metal from a die casting machine and stack them. First industry robot. Inventor: George Devol, user: General Motors.
- Lunokhod 1: 1970, lunar vehicle on the moon (Soviet Union)
- Shakey the robot: 1970
- 1989: Chess programs from Carnegie Mellon University defeat chess masters
- Aibo: 1999 Sony Robot Dog
- ASIMO: 2000 Honda (humanoid robot)



https://arxiv.org/pdf/1704.08617.pdf

Shakey the robot (1970)

- First general-purpose mobile robot to be able to reason about its own actions
- Advanced hardware:
  - radio communication
  - sonar range finders
  - television camera
  - on-board processors
  - bump detectors
- Advanced software:
  - Sensing and reasoning
- Very big impact
- https://robotics.shanghaitech.edu.cn/static/videos/Shakey.mp4



SOFTWARE

Robot Software: Tasks/ Modules/ Programs (ROS: node)

Support

- Communication with Micro controller
- Sensor drivers
- Networking
  - With other PCs, other Robots, Operators
- Data storage
  - Store all data for offline processing and simulation and testing
- Monitoring/ Watchdog

Robotics

- Control
- Navigation
- Planning
- Sensor data processing
  - e.g. Stereo processing, Image rectification
- Mapping
- Localization
- Object Recognition
- Mission Execution
- Task specific computing, e.g.:
  - View planning, Victim search, Planning for robot arm, …

Software Design

- Modularization:
  - Keep different software components separated
  - ☺ Keep complexity low
  - ☺ Easily exchange a component (with a different, better algorithm)
  - ☺ Easily exchange multiple components with simulation
  - ☺ Easily exchange dada from components with replay from hard disk instead of live sensor data
  - ☺ Multiple programming teams working on different components easier
  - Need: Clean definition of interfaces or exchange messages!
  - Allows: Multi-Process (vs. Single-Process, Multi-Thread) robot software system
  - Allows: Distributing computation over multiple computers

ROS 1 vs ROS 2

- ROS 1 developed since 2007 – but end of life scheduled for 2025
- ROS 2 developed since 2015
- Advantages/ changes ROS 2:
  - Supports more OS’s (Windows, MacOS)
  - Core written in C, bindings (e.g. C++, python) more consistent
  - Modern C++ interface (C++11, C++17)
  - OOP kind of enforced (code quality)
  - ROS2 Components (similar to ROS1 nodelets) enforced -> more performant intra-process communication
  - Communication via DDS (Data Distribution Service) – no more ROS master – better, following industry standards
  - Quality of Service (QoS) for communication (e.g. accept lost messages)
  - More modern build system (Ament instead of catkin)
  - Ros1 bridge: have a system run ROS1 and ROS2 with communication between nodes … (as a “hack” for the transition)

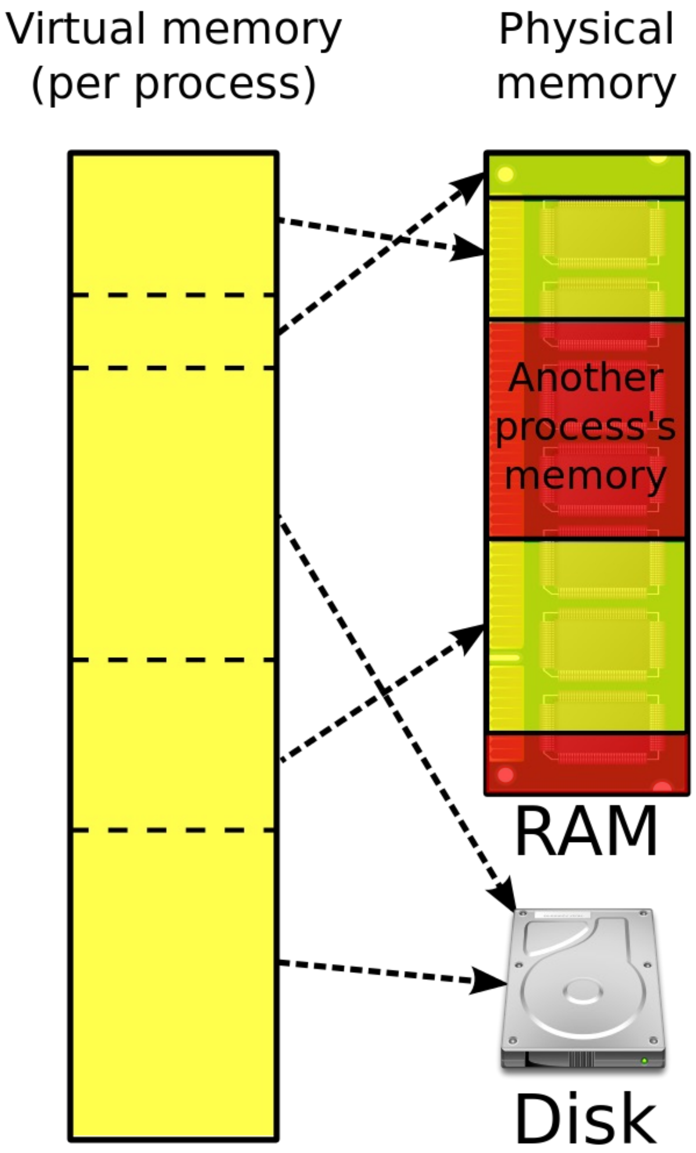
Programming review

- Process vs. Thread
- C++ Object Orientation
- Constant Variables
  - const-correctness
- C++ Templates
- Shared Pointer

- Objective:
  - Prerequisites for understanding ROS.
  - Understand how we can efficiently retrieve and transfer data in ROS.

Process

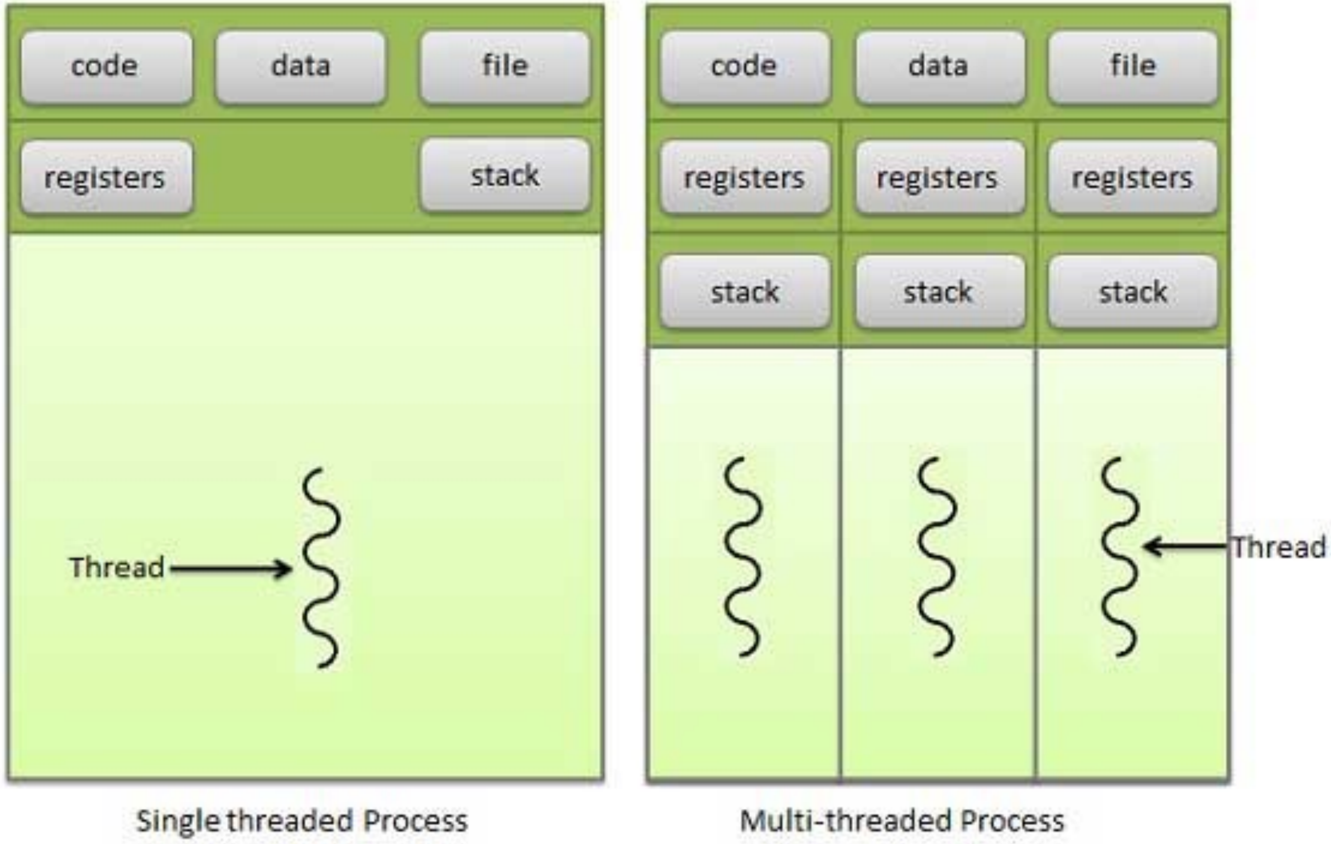
- Execution of one instance of a computer program
- Virtual memory:
  - Contains only code and data from this program, the libraries and the operating system
  - Other processes (programs) can not access this memory (shared memory access is possible but complicated)
- Operating system gives each process equal amount of processing time (scheduling) – if the processes need it
  - Good support from the operating system to give certain processes higher or lower priority
  - Linux console program to see processes: **top**



(From Wikipedia)

Multi-Threading

- In one process, multiple threads => parallel execution
- ☺ Code and Memory is shared => easy exchange of data, save mem.
- 😐 Synchronization can be tricky (mutex, dead lock, race condition)
- ☹ If one thread crashes, the whole process (all threads) die



(from http://www.tutorialspoint.com)

Processes and Threads in Robotics - Messages

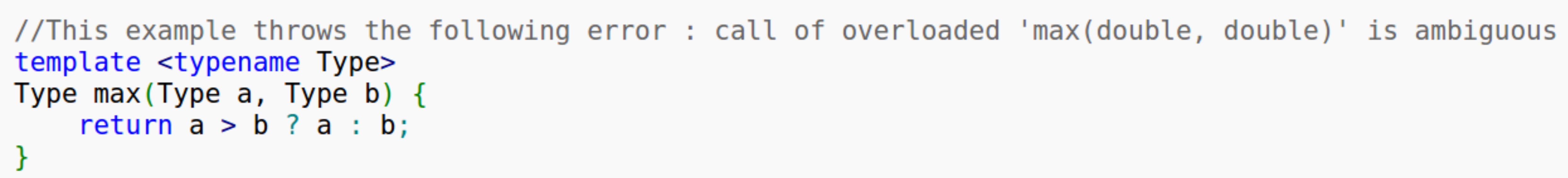
- Robot Operating System (ROS): Multiple Processes:
  - Each component runs in its own process: called **node**
  - A node can have multiple threads => faster computation
  - Nodes communicate using **messages**
  - A node can send (**publish**) **messages** under different names called **topic**
  - Nodes can listen to (**subscribe**) **messages** under different **topics**
  - **messages** have a type (e.g. sensor_msgs/Image) – a **topic** can only have one **type**!
  - The messages are transferred over the network (TCP/IP) => multiple computers work together transparently
  - ☹ Messages are serialized, copied and de-serialized even if both nodes on the same computer => slow (compared to pointer passing)
    - Optimization: ROS 2 uses Component architecture: run different nodes in the SAME process => fast communication

C++ Templates

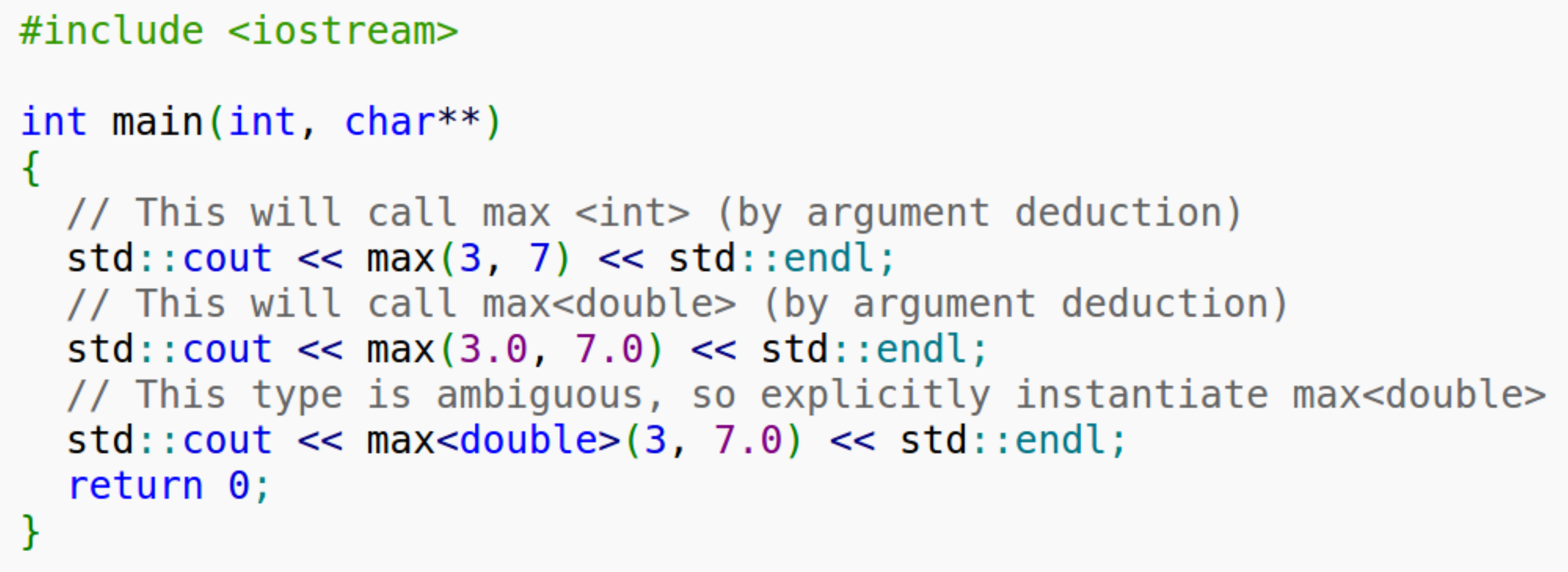
- Functions and classes that operate with generic types
- Function or class works on many different data types without rewrite
  - `template <typename T> int compare( T v1, T v2);`
  - Type of T is determined during compile time => errors during compilation (and not run-time)
  - Any type (type == class) that offers the needed methods & variables can be used
  - Usage: `compare<string>( string(“string number one”), “hello world” );`
    - Explicit declaration: typename T = string
    - typename T can (most often) deducted by the compiler from the argument types
- Class template:

  -
    ```
    template <typename T> class myStuff{
        T v1, v2;
        myStuff(T var1, T var2){  v1 = var2; v2 = var2; }
    };
    ```

Template example



```
//This example throws the following error : call of overloaded 'max(double, double)' is ambiguous
template <typename Type>
Type max(Type a, Type b) {
    return a > b ? a : b;
}
```



```
#include <iostream>

int main(int, char**)
{
  // This will call max <int> (by argument deduction)
  std::cout << max(3, 7) << std::endl;
  // This will call max<double> (by argument deduction)
  std::cout << max(3.0, 7.0) << std::endl;
  // This type is ambiguous, so explicitly instantiate max<double>
  std::cout << max<double>(3, 7.0) << std::endl;
  return 0;
}
```

Constant Variables

- Declare variables that do not change (anymore) in the code: `const`
- Works for variables and objects
- Const Objects:
  - Only methods that do not change any variable of the object may be called =>
  - Those methods have to be declared const
- Used for program-correctness
- Especially for multi-threading:
  - Share the data (e.g. image)
  - Make it read only via const
  - => no side-effects between different threads

1. const int x = 5; // x may not be changed
2. int * someValue = &x; // pointer – compilation error!!
3. const int * pointy = &x; // good
4. *pointy = 8; // error – pointing to const!
5. int y = 4;
6. pointy = &y; // from non const to const is always possible!
7. const int * p2 const = &y; // pointing to const variable and p2 is also const
8. p2 =&x; // error – p2 is const

Shared Pointer

- C++ Standard Library (std): heavily templated part of C++ Standard (many parts used to be in boost library)
- Pointer: address of some data in the heap – in the virtual address space
- Space for data has to be allocated (reserved) with: `new`
- After usage of data it has to be destroyed to free the memory: `delete`
- Problem: Data (e.g.) image is shared among different modules/ components/ threads. Who is the last user – who has to delete the data?
  - Shared pointer: counts the number of users (smart pointers); upon destruction of last user (smart pointer) the object gets destroyed : called “Reference counting”
  - Problem: Shared pointer needs to know the destructor method for the pointer =>
  - Shared pointer is a templated class: Template argument: class type of the object pointed to
  - Shared pointer can also point to const object!