Introduction to Robotics

Minimal architecture of an autonomous robot: perception $\rightarrow$ decision $\rightarrow$ action

---

DIROBOTS — Université de Montréal

1) Robotics: what is it, concretely?

Robotics: the discipline that builds **physical systems** capable of interacting with the real world.

A robot is neither "just software" nor "just hardware":

- **Sensors**: observe (vision, distance, orientation, etc.).
- **Computation / software**: interpret, decide, plan.
- **Actuators**: move, manipulate, apply forces.

Key idea: robotics is fundamentally a problem of **integration** (hardware + software + physical constraints). Much of the difficulty comes from making everything work **together**, in real-world conditions.

2) Robots and applications: why this field is “everywhere”

Examples of robots:

- autonomous drones and underwater robots;
- mobile platforms (warehouses, delivery), robotaxis;
- quadrupeds, humanoids, exoskeletons;
- industrial manipulators, service robots, medical robots.

Application domains:

- industry, healthcare, agriculture;
- logistics, transportation, exploration;
- home, education, public safety, entertainment.

Common denominator: imperfect data, uncertainty, real-time constraints, energy limits, and safety requirements.

3) A robot = a complete system (not a single component)

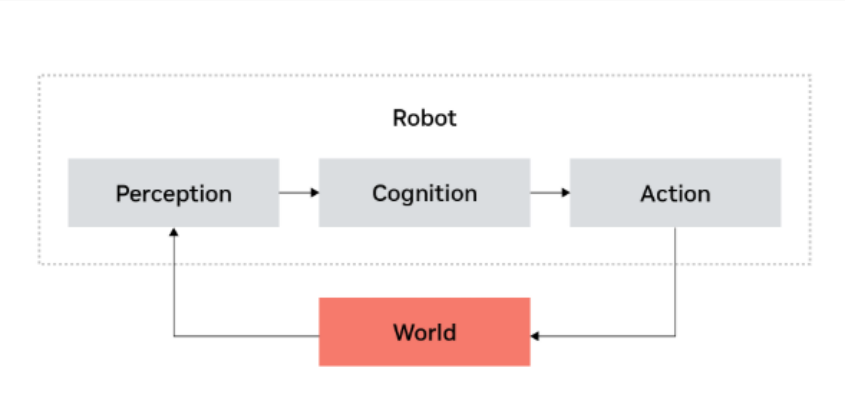
System view: a robot operates as a closed loop with the real world.

- The world produces signals (images, contacts, obstacles, motion).
- Sensors convert these signals into data.
- Software produces a decision (where to go, what to do, how).
- Actuators execute, and the world changes in return.

Why it is hard: latency, noise, calibration, physical limits, synchronization, and communication.

4) Basic architecture: Embodied AI (perception → cognition → action)

Reference model: the perception–decision–action loop.



Simple reading:

- **Perception:** "What is happening around me?"
- **Cognition / decision:** "What should I do now?"
- **Action:** "I move / act," which modifies the world.

Analogy: eyes → brain → muscles.

5) Hardware: sensors, actuators, controllers

Sensors (perception): convert the world into measurements.

- **Cameras:** visual information (objects, landmarks).
- **LiDAR:** distances / geometry (obstacles, 3D maps).
- **IMU:** orientation and acceleration (stability, navigation).
- **GPS:** global position (mainly outdoors).

Actuators (action): apply motion or force.

- motors (wheels, joints), servos (precise positioning),
- pneumatic / hydraulic systems (high force applications).

Controllers (embedded computing): execute control loops and manage communication.

- microcontrollers, embedded computers (e.g. SBCs), dedicated platforms.

6) Software: building blocks found everywhere

Even though each robot is different, the same modules appear repeatedly.

- **Perception:** extract useful information from sensors (obstacles, objects, scene understanding).
- **Localization / state estimation:** estimate pose and dynamics (position, orientation, velocity).
- **Planning:** choose goals and produce feasible trajectories (global route + local avoidance).
- **Control:** track trajectories despite noise, latency, disturbances.
- **Supervision / safety:** limits, emergency stop, degraded modes, logging.

Key idea: autonomy emerges from the reliable cooperation of these blocks, not from any single module.

7) Autonomy: a typical pipeline (from goal to execution)

An autonomous architecture is naturally organized in layers.

- **Perception + localization:** understand the environment and the robot state.
- **Global planning:** define a mission (goal, route, waypoints).
- **Behavior planning:** decide actions (priorities, interactions, subtasks).
- **Local planning:** generate short-horizon, reactive trajectories.
- **Control / action:** execute and continuously correct (closed loop).

Feedback loop: sensors continuously feed state estimation and re-planning.

8) Navigation and manipulation: two core capabilities

Autonomous navigation:

- move without human intervention;
- avoid obstacles, handle sensor uncertainty, remain stable and safe.

Object manipulation:

- **grasping:** robustly grasp diverse shapes;
- more complex manipulation: precise placement, assembly, fine tasks;
- handling contacts and forces.

Key point: these tasks become hard as soon as the environment is not perfectly controlled.

9) Integration: making hardware and software cooperate

In real robots, many failures come from **integration** (interfaces, timing, consistency).

What must work together:

- sensor data (rates, latency, calibration, reference frames);
- actuator commands (saturation, precision, safety);
- inter-module communication (messages, synchronization, logs).

Why middleware (e.g. ROS) is useful:

- structure the system into replaceable modules;
- standardize communication and diagnostics;
- facilitate debugging and reproducibility via logging.

10) Simulation and testing: before the real world

Why simulate?

- test quickly and at scale;
- reduce hardware and human risk;
- validate the perception–decision–action pipeline before deployment.

Simulation tools (non-exhaustive):

- Isaac Sim / Isaac Lab (NVIDIA ecosystem);
- Gazebo;
- MuJoCo (widely used in control and robot learning).

Testing strategies:

- **SIL** (Software-in-the-loop): full system in simulation;
- **HIL** (Hardware-in-the-loop): real hardware inside the loop;
- unit tests (modules) and system tests (full pipeline).

References

- Kevin M. Lynch, Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.
(ISBN 9781107156302)

- NVIDIA Deep Learning Institute. *A Beginner's Guide to Autonomous Robots*. Online course.
`https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-35+V1`