# Intelligent Robotics notes

Anita Giacomin
academic year: 2024/2025

# INTRODUCTION

## ARTIFICIAL INTELLIGENCE

Make machines act intelligently
- Knowledge representation
- Understanding Natural Language
- Learning
- Planning and problem solving
- Inference
- Search
- Vision

## ROBOT

- ISO 8372:2012 : an actuated mechanism, programmable in two or more axes, with a certain degree of autonomy, moving in its environment, to perform intended tasks.
- It comprises { sensors, information processing, actuators }

## INTELLIGENT ROBOT

Physically situated intelligent agent (maximize chance of success).
A core characteristic: autonomy.

## CLOSED WORLD VS OPEN WORLD ASSUMPTION

1. Everything relevant is known a priori. Everything that is not explicitly stated is assumed to be false. We can make accurate models, stable control loops, sensing can be minimized.
2. Models are partial and unpredictably correct, sensors are required to dynamically adapt. Real world is an open world.

## AUTOMATION VS AUTONOMY

1. Physically situated tools execute highly-repetitive, pre-planned actions for well-modeled tasks, under closed world assumption.
   Focus on formal, stable control loops.
2. Physically-situated agent generates new plan and adapt dynamically in an open world.
   Focus on AI.

| | Automation | Autonomy |
|---|---|---|
| Plans | Execution (perhaps plan once, then repeat the plan) | Generation (of new plans) |
| Actions | Deterministic (can model the system) | Non-Deterministic (system is too complex) |
| Models | Closed-world | Open-world (partial models) |
| Knowledge Representation | Signals (controls/decision making is at the signal level) | Symbols (controls/decision making is with symbols) |

Automation VS Autonomy ; impacts:
- hardware (do we need sensors?)
- software (controls vs AI)
- failure modes

# LOCOMOTION

## KINEMATICS VS DYNAMICS

1. Study of motion of objects without regard to the forces that cause the motion.
2. Study of how forces affect the motion of objects.

## LOCOMOTION

Fundamental ability of a robot to move from place to place, relying on the physical interaction between vehicle and environment.
It is concerned with the interaction forces, along with the mechanisms and actuators that generate them.
The main objectives:
- speed/acceleration
- precision, repeatability
- flexibility and robustness
- efficiency

### LOCOMOTION ASPECTS

1. STABILITY : numer of contact points, center of gravity
   static vs dynamic, inclination of terrain.
2. CONTACT : contact point/area, angle of contact, friction.
3. ENVIRONMENT : structure, medium.

## CHALLENGES IN LOCOMOTION

1. STABILITY
2. MANEUVERABILITY : mobility + steerability
3. CONTROLLABILITY : positions achievable

tradeoffs are required.

## HOLONOMIC VS NON-HOLONOMIC

Holonomy is the property of moving instantaneously in any direction (it can control all its DoF instantaneously and independently).
In other words, controllable DoF = workspace DoF (holonomic robot has zero non-holonomic constraints).

## TYPES OF LOCOMOTION

- Many are inspired by nature, but difficult to imitate technically.
  { Wheeled, Legged, Snake, Free-floating, Swimming }
- we have to consider maneuverability, motion models, and control strategies.

## LEGGED LOCOMOTION

Walking of a biped is close to rolling of a polygon (stride = length of sides).
Passive dynamic walker : forward falling + passive leg swings on a ramp.
Can overcome many obstacles but require controlling several DoFs.
At least 3 legs for static stability, 6 for dynamic stability.

## WHEELED LOCOMOTION

3 wheels for stability (2 with center of gravity below wheel axis), more require suspension system.

### WHEEL TYPE

- Standard : 2 DoF
- Castor : 2 DoF
- Sweedish : 3 DoF
- Spherical : omnidirectional

### COMMON WHEEL CONFIGURATIONS

- Rack and Pinion
- Differential Drive
- Skid-Steer
- Synchro-drive

All non-holonomic, two motors but different kinematics.

## LOCALIZATION AND MAPPING

**BELIEF**: robot's current estimate of its state given all available sensor measurement and prior knowledge (probabilistic representation).
**ROBOT**: device that moves through the environment and modifies it.
**STATE**: collection of all aspects of the robot and the environment that may have some impact on the behavior of the robot.
**MAPPING**: build a map given location and observations.
**EXPLORATION**: autonomous/semi autonomous exploration of unknown environments.

### SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)

localize and map at the same time.
Bayes filter algorithm: recursively update OBSERVATION MODEL and MOTION MODEL based on new controls and sensed info.

- FULL SLAM: process batch data, compute the posterior probability.
- ONLINE SLAM: estimate only current pose, process data sequentially.

Types of SLAM
- Volumetric vs Feature-Based
- Topological vs Metric
- Known vs Unknown Correspondance
- Static vs Dynamic
- Small vs Large Uncertainty
- Active vs Passive
- Single Robot vs Multi-Robot
- Anytime and Any-Place

VISUAL SLAM
- Visual Odometry
- Visual graph SLAM

### LOCALIZATION

estimate pose (position + orientation) relative to a known map based on a sequence of sensor readings and controls.

1. Feature-based
   - Markov localization: compute probability of each possible pose.
   - Extended Kalman filter: recursive state estimation, Gaussian noise, linearizes system dynamics around the current estimate.
2. Iconic Localization (use raw sensor data, not features extracted)
   - Grid-based: sensed world is discretized, compute probability of each cell.
   - Monte Carlo Localization: particles are sampled through space according to the probability of the poses/particles in the previous iteration.

The uncertainty is modeled by a covariance matrix

### 3 SLAM PARADIGMS

#### 1. EXTENDED KALMAN FILTER

use Jacobian matrix to linearize the system, parametric and online method, assumes Gaussian noise.
- Prediction Step (predict the next state and linearize)
- Correction step (update observation model, linearize, compute gain and correct the state to minimize uncertainty.

#### 2. PARTICLE FILTER

Known as Monte Carlo Localization
- prediction step: propagate particles
- update step: evaluate the likelihood of each particle given observation.
- resampling step: select particles with high likelihood.

(for SLAM: particles are full trajectories hypotheses)

#### 3. GRAPH-BASED SLAM

Slam as graph optimization problem. Robot's poses and landmarks are nodes, edges are temporal-spatial constraints.
- Graph construction
- Loop closure detection
- Optimization

## MOTION PLANNING

**MOTION PLANNING**
Ability to compute their own motion in order to achieve certain goals.
MP Goal: find a path from 'start' to 'goal' without collisions.

**CONFIGURATION SPACE**
- Workspace: static environment populated by obstacles
- Obstacle space
- Robot Configuration
- Configuration space
- Collision space
- Free space

**PATH PLANNING VS MOTION PLANNING**
1. Purely geometric problem of computing a collision-free path among static obstacles.
2. used for problems involving time, dynamic constraints, object coordination,...
   - PATH: geometric concept
   - TIMING LAW: time dependence
   - TRAJECTORY: path over which a timing law has been assigned (actual output of MP)

### CLASSICAL PLANNING APPROACHES

1. COMBINATORIAL PLANNING
   - Generalized Voronoi Diagram
   - Visibility Graph
   - Exact cell Decomposition (convex cells ⟶ connectivity graph, A*)

   Elegant and complete, but intractable for high-dimensionality
2. SAMPLING-BASED PLANNING
   - Probabilistic Roadmap (PRM) (learning - connect random sample, SEARCH)
   - Rapidly-Exploring Random Tree (RRT) (random sample → new point < ε)
   - RRT-Connect (2 trees alternately expanding)

   As time increases, the probability of finding a free path tends to 1.
3. ARTIFICIAL POTENTIAL FIELDS

   Forces are orthogonal to equipotential contours, forces increase approaching the boundary.

   Those approaches are not complete but workarounds exist for online motion planning:
   - Best-first
   - Navigation function
   - Numerical navigation function

## ROBOT PERCEPTION

**SENSOR**: device that converts a physical parameter or environmental characteristic into a signal that can be digitally measured and processed to perform specific tasks (provide raw data).
**PERCEPTION**: ability to acquire, process, and interpret sensory data from its environment.
**SENSING**: combination of sensors and algorithms that transform this data into meaningful percepts.
**SENSOR FUSION**: integration of data from multiple sensors to generate higher-level percepts and world models.

### SENSOR CLASSIFICATION

ACTIVE VS PASSIVE
1. Emit energy and measure the reaction (higher performance, could influence the environment).
2. Measure naturally occurring energy.

BY MEASUREMENT DOMAIN
- PROPRIOCEPTIVE: sense stimuli generated within the robot.
- EXTEROCEPTIVE: " " from external sources.
- EXPROPRIOCEPTIVE: " the position of external objects relative to parts of the body.

BY OUTPUT
- Image vs non-image
- Matrix vs non-matrix

### CHARACTERIZING SENSOR PERFORMANCE

- Linearity: consistency of output variation to input variation.
- Measurement Range: [smallest val, largest val] measurable by the sensor.
- Dynamic Range: ratio between lower and upper limit (in decibels: $20 * \log_{10}$)
- Resolution: smallest measurable difference.
- Perceptive Horizon: Range + Occlusion.
- Frequency: speed at which the sensor provides readings.
- Bandwidth: low bandwidth sensors cannot measure high-frequency changes.
- Response time: time between the reception of input signal change and output change.

### IN-SITU PERFORMANCE:

- Sensitivity: ratio of output change to input change.
- Cross-sensitivity: influence of other environmental parameters.
- Precision: ability to reproduce the same results under the same conditions.
- Accuracy: correctness of output compared to input.
- Systematic error: predictable error.
- Random error: unpredictable error described probabilistically.

### PROPERTIES OF MEASUREMENT SYSTEMS

- Accuracy
- Repeatability
- Stability
- Linearity Error
- Offset Error
- Resolution Error

### PROXIMITY AND CONTACT SENSORS

- Photoelectric proximity sensors
- Bumpers

### ENCODERS

- Regular
- Optical
- Quadrature
- Reference Pulse
- Hall-effect Encoders

### GLOBAL POSITIONING SYSTEMS

- Outdoor (GNSS)
  - Differential Global Positioning System (DGPS)
  - Real-time Kinematic (RTK)
- Indoor (UWB)
  - Time Difference of Arrival (TDOA)
  - Time of Flight (TOF)

### INERTIAL AND HEADING SENSORS

- Accelerometers
  - MEMS
  - Piezoelectric
  - Capacitive
- Gyroscopes
  - Mechanical
  - Optical
- Compasses
  - Hall Effect
  - Fluxgate
- Inertial Measurement Unit (IMU)

### DIGITAL CAMERAS

Imaging Sensor (CCS/CMOS), optical system (lens) + ADC
- Perspective
- Omnidirectional/Panoramic
- Event

### RANGING

- Sonar/Ultrasonic
- LiDAR
  - Continuous-wave
  - Pulse-based
  - Solid-state
- Radar
- Time-of-flight cameras
- Stereo Cameras
- Structured-Light cameras
  - spatial encoding
  - temporal encoding
- RGB-Depth Cameras
- Depth from X
- Infrared Sensors

### SENSOR FUSION

- Fission/Redundant
- Fusion/Complementary
- Fashion/Cooperative

# SOFTWARE ARCHITECTURES

## ARCHITECTURE

- Overall style of design or organization. Describes the components and how they interact, provides the structure but also the constraints.
- Why is it important for good software engineering:
  - Abstraction: ignore details, focus on general organization.
  - Modularity: low coupling, support unit testing and debugging, high cohesion.
  - Anticipation of change, incrementality: how to adapt, support evolution.
  - Generality: not re-inventing the wheel each time.
- 3 levels of design and organization:

1. **OPERATIONAL**: what the system does at a high level, not how it does it. (Focus on theoretical and conceptual aspects)

   Canonical organization:
   - Deliberative layer (reasoning over symbols/info about goal)
   - Interfacing layer (converting sensor data into symbols)
   - Reactive/Behavioral layer (skills and responses)

   We can extend it with an Interfacing layer (interface for collaboration)

   Challenges:
   - From Reactive to Deliberative: direct vs recognition perception, different time horizons (present vs ppf), need a central structure (world model)
   - From Reactive/Deliberative to Interaction: additional "theory of mind" (beliefs, desires, intentions) of oter agents.

2. **SYSTEMS**: how a system works in terms of major subsystems (Focus on both theoretical and implementation aspects)
3. **TECHNICAL**: how a system works in terms of implementation details. (Focus on implementation and software aspects)

## BEHAVIORS

- Behavior: mapping of sensory inputs to a pattern of motor actions that are used to achieve a task.
  - Reflexive: stimulus response, hard-wired for fast response.
  - Reactive: learned, executed without conscious thought.
  - Conscious: require deliberative thought
- Braitenberg Vehicles: 14 agents of increasing complexity and different types of connection (coward vs aggressive)

BEHAVIOR: a piece of code executed as a thread that defines a control law for achieving/maintaining a given goal.

ARBITER: behaviour module that selects the most suitable behaviors to be activated based on sensor data.

ARBITRATION: spatial or temporal ordering of robot behaviors.

ROBOT DESIGN: specification of sensors and effectors, definition of basic behavior modules and arbiters.

## ARBITRATION OF BEHAVIORS

If behaviors don't follow a fixed sequence, they have to interact:
- Equilibrium
- Dominance
- Cancellation

Main Strategies:
- COOPERATIVE: blends the output of different behaviors so to be consistent with goal. (fusion via vector sum)
- COMPETITIVE: coordinate responses in case of conflicts.
  - Fixed prioritization (take the one with higher priority)
  - Action selection (take the one with highest activation level)
  - Arbitrage with voting

## ATTRIBUTES FOR DESCRIBING LAYERS

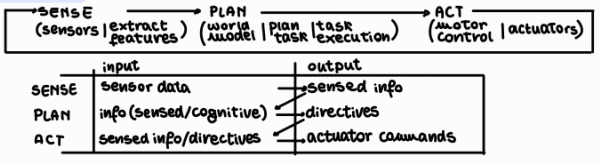
- PRIMITIVES: sense - extract info from sensor data
  plan - produce directives from sensed/cognitive info
  act - produce actuator commands from sensed info/directives.
  learn - maximize chances of success.
- PERCEPTUAL ABILITIES: can it work with raw sensor data or does it need conversion into symbols?
- PLANNING HORIZON: Past, Past and Present, Past Present and Future.
- TIME SCALE: fast (reflex), slow (reasoning).
- WORLD MODEL: how does it store information.

## ROBOTICS' PARADIGMS

Paradigm: set of hypotheses/techniques that characterize an approach to a class of problems.

2 ways to describe them:
- how SENSE, PLAN, ACT are connected.
- how sensory data is managed

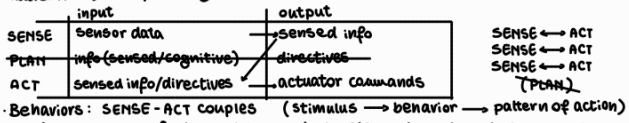
### 1. HIERARCHICAL:




- The world model: a priori representation + sensed info + cognitive understanding.
- Disadvantages: relying on world model can create bottlenecks, relies on extensive representation (closed-world assumption, frame problem), implementations are planning centric. This makes it not reliable in the real world.

### 2. REACTIVE:

- Arises from dissatisfaction with hierarchical paradigm and is influenced by the theory of cybernetics (principles of controls in both animals and machines). No planning involved.

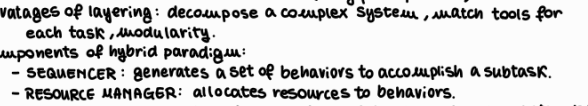



- Behaviors: SENSE-ACT couples (stimulus ⟶ behavior ⟶ pattern of action)
- Limitations: very fast, modular and simple, but no awareness or mission monitoring, missing world model.

⟶ SUBSUMPION ARCHITECTURE (first reactive architecture)
Multiple layers of behavior that 'fire' simultaneously.
Each behavior: Augmented Finite State Machine (AFSM), with inhibitor, suppressor, reset. Competitive coordination (priority of higher layers).
Higher layers subsume lower ones.

### 3. HYBRID:

- Neither deliberative nor reactive approaches are suitable for building agents.
- Combine deliberative (symbolic world model, planning) and reactive subsystems, organizing them in layers. How can we balance them?
- PLAN then SENSE-ACT:




- Advatages of layering: decompose a complex system, match tools for each task, modularity.
- Components of hybrid paradigm:
  - SEQUENCER: generates a set of behaviors to accomplish a subtask.
  - RESOURCE MANAGER: allocates resources to behaviors.
  - CARTOGRAPHER: creates, stores and maintains a map/world model/knowledge.
  - MISSION PLANNER: interacts with operator, from commands to robot terms.
  - PERFORMANCE MONITORING AND PROBLEM SOLVING

# CAUSAL ROBOTICS

## CAUSAL INFERENCE

Can help us answer 'why did that happen?'
Correlation does not imply causation
We need causal models to explain how data was generated

## EXAMLE: Human-Robot Interaction

Use causal inference to understand cause and effect, so that the robot can predict human intentions and cooperate, build a causal model to anticipate how robot's interventions can affect human behavior.

# ROS: ROBOT OPERATING SYSTEM

## ROS IS NOT

- Programming language
- (Only) a library
- A (traditional) OS

## ROS IS

Open Source, meta-operating system for your robot. It provides 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 different hardware platforms.

## WHY USE ROS

1. Modular Design
2. Distributed Computation
3. Easy Shareability and Reusability
4. Rapid Debugging and testing
5. Vibrant Community

## FILESYSTEM

Workspace
Metapackage
Packages
- Package Manifest
- Messages
- Services
- Code
- Other

## NODES

Process where computation is done

## ROS MASTER

Stores info about the network, connecting nodes (peer 2 peer)

## MESSAGES

Data that gets sent from one node to the others

## TOPIC

Channel of communication (publisher ⟶ subscriber, decoupling)

## SERVICE

Request-Response communication

## ACTION

Like a Service but the node requesting it can stop the process or get periodic feedbacks.

## ROS BAG

Store ROS message data, data logging, data analysis, visualize and plot data

## TAGS

Automatic detection of Fiducial Markers

# HUMAN-ROBOT INTERACTION: TELEOPERATION

## TELEOPERATION

Human and robot are physically separated and must interact to accomplish tasks.
Telesystems are the end state for remote presence applications.

## TASKABLE AGENTS VS REMOTE PRESENCE

1. Given a task, executes without supervision, returns (with information)
2. Human and robot share the task and the role, task execution is blended (joint cognitive system)

The two are not mutually exclusive. Shifting between them: TRADED CONTROL

## COMPONENTS OF A TELESYSTEM

Local
- Display
- Control

⟷ Communication

Remote
- Sensor
- Control
- Effector
- Power

## PROBLEMS

- Communication issues (dropout, bandwidth limitations, ...)
- Cognitive fatigue, lack of situational awareness (poor interfaces)
- Increased manpower due to human inefficiencies.
- HUMAN OUT-OF-THE-LOOP: humans may not seamlessly take over in traded control ⟶ smooth transfer has to be explicitly desired.

## IMPORTANT DEFINITIONS

SUPERVISORY CONTROL: human provides high-level goals that are seen more like change in the process than control signals.

TRADED CONTROL: human and robot are active at different times. The human decides high-level goals and exits the loop (no interaction).

SHARED CONTROL: human remains in the loop (controlling some variables). Two control signals are mixed. Needed when neither could perform the task alone.

SHARED AUTONOMY: robot as an intelligent tool able to act autonomously to a certain degree (user freed from many details of the action).

SHARED INTELLIGENCE: planning tasks are shared between operator and robot, and the latter can take over in case of human error. Mutual collaboration between the two agents in making safe plans for the robot.

level of detail ↕ role of robot in decision-making

| Concept | Who has Control? | Simultaneous or Alternating? | Key Features |
|---|---|---|---|
| 1 | Human oversees | A | Human monitors, system acts |
| 2 | Either human or system | A | Control shifts explicitely |
| 3 | Both | S | Inputs blend together |
| 4 | Both | S | System understands and enhances intent |
| 5 | Both | S | Human and AI collaborate in decision-making |

## HUMAN SUPERVISORY CONTROL

One or more human operators are intermittently giving directives and continually receiving information from a computer that itself closes an autonomous control loop through artificial effectors and sensors to the controlled process or task environment

The human is always involved at some level.

| | | Supervisory Control: Autonomy | |
|---|---|---|---|
| Robot Primary Controller | Social Interactions | Shared Control / Traded Control | Teleoperation |
| Operator Primary Controller | Remote Controlled | Manual Control | Teleoperation |
| | Operator can see robot | Operator can't see robot | |

Shared control helps to reduce user's workload and minimize noise from incorrect user commands.

### 1. MANUAL CONTROL

The operator needs (proprio-, extero-, exproprio-ception) sensors.

### 2. SHARED CONTROL

Operator: deliberative control, Robot: reactive control.

### 3. TRADED CONTROL

The control is mostly in the robot, but transfers to the operator when needed

### 4. AUTONOMY

The operator only provides directives.

## 3 FORMS OF COOPERATION

- human has the control, robot acts
- human and robot both trigger the loop
- human and robot both plan, and the mixed input starts the loop.

# DEEP LEARNING FOR ROBOTICS

## ATTENTION MECHANISM

Input sentence ⟶ Embedding ⟶ Queries, Keys and Values ⟶ Attention Matrix

- Attention Matrix: correlation between queries and Keys
- Queries: "questions"
- Keys: possible "answers"
- Attention scores: measure of correctness

We can use attention for different inputs.

### SELF-ATTENTION VS CROSS ATTENTION

1. Relate different parts of the same input sequence.
2. Get new representation of input 2 that depends on the correlation between input 1 and 2. This allows to mix different inputs.

### SINGLE-HEAD VS MULTI-HEAD

1. Single attention mechanism
2. Multiple attention mechanisms (with different weights) operate in parallel

## TRANSFORMERS

Initially developed for NLP, the idea is to transform input data into contextual representation, using a self-attention mechanisms to capture long-range dependencies.

### TRANSFORMER BLOCK

Input ⟶ Multi-head self-attention ⟶ layer normalization ⟶ feedforward NN

1. ENCODER: Raw input tokens ⟶ Embeddings. Used for prediction, knowledge storage, ...
2. DECODER: Embeddings ⟶ Output tokens. Used for language generation, ...
3. ENCODER-DECODER: Raw input tokens ⟶ Output tokens. Used for end-to-end tasks (translation, ...)

Transformers are good in learning long-term dependencies in (even multi-modal) data, they scale well but need a lot of data

## DEEP LEARNING AND ROBOTICS

- Perception
- Planning
- Sense-Plan-Act All-in-One (AiO)

### 1. PERCEPTION

- Scene Understanding: Object detection, segmentation, object tracking, ...
- Robotic Grasping: Scene description ⟶ extract candidate grasping poses ⟶ score candidates

### 2. PLANNING

Generate suitable (feasible, no collisions, and successful) trajectories for the robot.

- Neural Motion Planning: Train a deep NN to predict trajectories on synthetic data.
- Task Planning with LLM: perform previously defined (atomic) operations given directives.
- Planning with diffusion models

### 3. SENSE-PLAN-ACT All-in-One (AiO)

End-to-End 'real-time' robot controller

Raw sensor readings + task description ⟶ Control Signals

- Use NLP prompts + sensor readings
- tokenize (one for each different type of input)
- Use LLM to predict output tokens in an autoregressive way
- Fine-tune LLM to output tokens that represent actions
- Detokenize actions.

Problem: LLM are very heavy

⟶ Light Robotic Transformers
- Map task description and observations into tokens using L.R.T.
- Train encoder to predict action embeddings
- Decode with Light Action Head

More efficient, modular, less reasoning and zero-shot capabilities

- Navigation with Vision Language Action (VLA)

Generate navigation subgoals, given instructions

Vision transformer encoder ⟶ MLP gives prompt ⟶ LLM generates high-level instructions ⟶ low-level commands

Advantages
- High-level approach
- Zero/Few-shot capabilities
- Multi-modal, Multi-task

Limitations
- Large Dataset
- Extreme computations
- Out-of-Domain Generalization