Robotic Paradigms and Control Architectures

Jan Faigl

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

Lecture 02

B4M36UIR – Artificial Intelligence in Robotics

Overview of the Lecture

- Part 1 – Robotic Paradigms and Control Architectures
  - Robotics Paradigms
  - Hierarchical Paradigm
  - Reactive Paradigm
  - Hybrid Paradigm
  - Example of Collision Avoidance
  - Robot Control

Part I

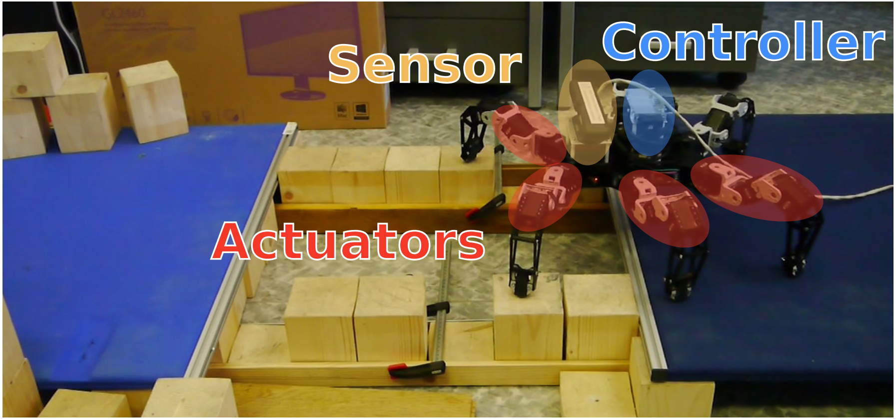
Part 1 – Robotic Paradigms and Control Architectures

Outline

- Robotics Paradigms
- Hierarchical Paradigm
- Reactive Paradigm
- Hybrid Paradigm
- Example of Collision Avoidance
- Robot Control

Robot

- A robot perceives an environment using **sensors** to **control** its **actuators**.



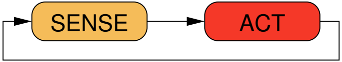
- The main parts of the robot corresponding to the primitives of robotics: **Sense**, **Plan**, and **Act**.
- The primitives form a **control architecture** that is called **robotic paradigm**.

Robotic Paradigms

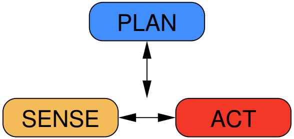
- **Robotic paradigms** define relationship between the robotics primitives: **Sense**, **Plan**, and **Act**.
- Three fundamental paradigms have been proposed.

1. **Hierarchical paradigm** is a purely deliberative system.

2. **Reactive paradigm** represents reactive control.



3. **Hybrid paradigm** combines reactive and deliberative.

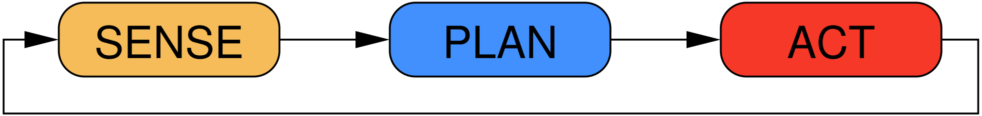


Outline

Hierarchical Paradigm

- The robot senses the environment and creates the "world model".

  A "world model" can also be an a priori available, e.g., prior map.
- Then, the robot plans its action and executes it.



- The advantage is in ordering the relationship between the primitives.
- It is a direct "implementation" of the first AI approach to robotics.
  - Introduced in Shakey, the first AI robot (1967-70).
- It is **deliberative architecture**.
  - It uses a generalized algorithm for planning.
  - *General Problem Solver* – STRIPS

    Stanford Research Institute Problem Solver
- It works under the **closed world assumption**.
  - The world model contains everything the robot needs to know.

Disadvantages of the Hierarchical Model

- Disadvantages are related to planning and its **computational requirements**.
- Planning can be very slow and the "global world" representation has to contain further all information needed for planning. Sensing and acting are always disconnected.
- The "global world" representation has to be up-to-date.
  - The world model used by the planner has to be frequently updated to achieve a **sufficient accuracy** for the particular task.
- A general problem solver needs many facts about the world to search for a solution.
- Searching for a solution in a huge search space is quickly computationally intractable, and the problem is related to the so-called **frame problem**.
  - Even simple actions need to reason over all (irrelevant) details.
- **Frame problem** is a problem of representing real-world situations to be computationally tractable. *Decomposition of the world model into parts that best fit the type of actions.*

Examples of Hierarchical Models

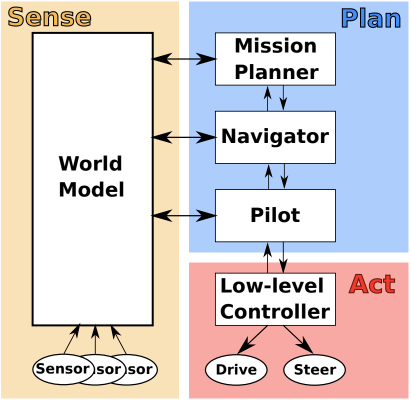
- Despite drawbacks of the hierarchical paradigm, it has been deployed in various systems, e.g., *Nested Hierarchical Controller* and *NIST Realtime Control System*.

  It was used until 1980, when the focus was changed to the reactive paradigm.
- The development of hierarchical models further exhibited additional advancements such as a potential to address the **frame problem**.
- They also provide a way to organize the particular blocks of the control architecture.
- Finally, the hierarchical model represents an architecture that **supports evolution and learning systems** towards fully autonomous control.

Nested Hierarchical Controller

- Decomposition of the planner into three different subsystems: **Mission Planner**, **Navigation**, **Pilot**.
- **Navigation** is planning a path as a sequence of waypoints.
- **Pilot** generates an action to follow the path.

  It can response to sudden objects in the navigation course. The plan exists, and it is not necessary to perform complete planning.



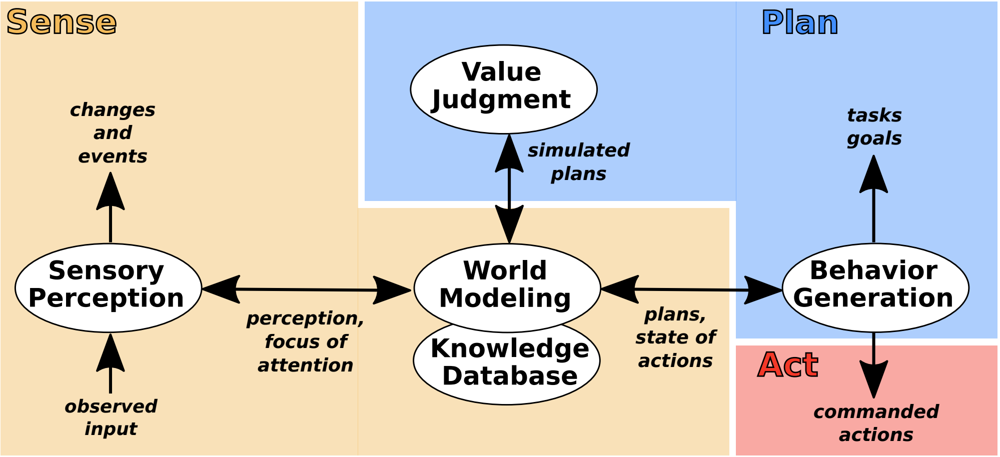
NIST Real-time Control System (RCS)

- Motivated to create a guide for manufacturers to add intelligence to their robots.
- It is based on the NHC, and the main feature it introduces is a set of models for sensory perception.
- It introduces preprocessing step between the sensory perception and a world model.
- The sensor preprocessing is called **feature extraction** such as following.
  - An extraction of the relevant information for creating a model of the environment, such as salient objects utilized for localization.
- It also introduced the so-called Value Judgment module.
  - After planning, it simulates the plan to ensure its feasibility.
- Then, the plan is passed to **Behavior Generation** module to convert the plans into actions that are performed (**Act**).

The "behavior" is further utilized in reactive and hybrid architectures.

Overview of the Real-time Control System (RCS)

- Key features are sensor preprocessing, plan simulator for evaluation, and behavior generator.



Hierarchical Paradigm – Summary

- Hierarchical paradigm represents **deliberative architecture** also called **sense-plan-act**.
- The robot control is decomposed into functional modules that are sequentially executed.

  The output of the sense module is the input of the plan module, etc.
- It has centralized representation and reasoning.
- May need extensive and computationally demanding reasoning.
- Encourage open loop execution of the generated plans.
- Several architectures have been proposed, e.g., using STRIP planner in Shakey, Nested Hierarchical Controller (NHC), NIST Real-time Control System (RCS).

  NIST – National Institute of Standards and Technology

Despite the drawbacks, hierarchical architectures tend to support the evolution of intelligence from semi-autonomous control to fully autonomous control.



Navlab Testbed 1986 – https://youtu.be/ntIczNQKfjQ

Navlab vehicles 1–5

Navlab (1996) uses 90% of autonomous steering from Washington DC to Los Angeles.

History Corner

- Where to? A history of autonomous vehicles.

  https://computerhistory.org/blog/where-to-a-history-of-autonomous-vehicles/

  - Stanford Artificial Intelligence Laboratory Cart, 1964-71.
  - Ernst Dickmanns' VaMoRs Mercedes van, Bundeswehr University Munich, 1986-2003.
  - Navlab 1 – Navlab 5, 1984–1990. https://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/navlab5_details.html

    Driverless Car Technology Overview at Carnegie Mellon University – https://www.youtube.com/watch?v=2KMAAmkz9go
  - DARPA Grand Challenge – 2004 (no winner) and 2005 in Desert Southwest (6 h 53 min).
  - DARPA Urban Challenge 2007.

Navlab 1 (1986)

http://youtu.be/ntIczNQKfjQ

Navlab 5 (1997)

http://youtu.be/xkJVV1_4l8E

VaMoRs (1986–2003)

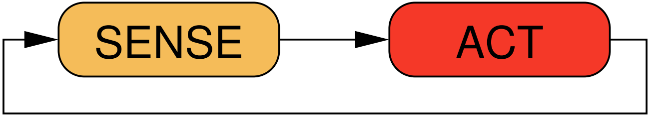
http://youtu.be/I39sxwYKlEE

Outline

- Robotics Paradigms
- Hierarchical Paradigm
- Reactive Paradigm
- Hybrid Paradigm
- Example of Collision Avoidance
- Robot Control

Reactive Paradigm

- The **reactive paradigm** is a connection of sensing with acting.



- It is biologically inspired as humans and animals provide evidence of intelligent behavior in an **open world**, and thus it may be possible to overcome the **close world assumption**.
- Insects, fish, and other "simple" animals exhibit intelligent behavior without virtually no brain.
- There must be the same mechanism that avoids the **frame problem**.
- For further discussion, we need some terms to discuss the properties of "intelligence" of various entities.

Agent and Computational-Level Theory

- **Agent** is a self-contained and independent entity.
  - It can interact with the world to make changes and sense the world.
  - It has **self-awareness**.
- The reactive paradigm is influenced by Computational-Level Theories.

  D. Marr, a neurophysiologist who worked on computer vision techniques inspired by biological vision processes.

  - **Computational Level** – **What?** and **Why?**

    What is the goal of the computation, and why is it relevant?
  - **Algorithmic level** – **How?** *Focus on the process rather than the implementation.*

    How to implement the computational theory? What is the representation of input and output? What is the algorithm for the transformation of input to output?
  - **Physical level** – How to implement the process?

    How to physically realize the representation and algorithm?

Behaviors

- **Behavior** is the mapping of sensory inputs to the pattern of motor action.

Sensory-Motor Pattern

- Behaviors can be divided into three categories.
  - **Reflexive behaviors** are "hardwired" stimulus-response (S-R).

    Stimulus is directly connected to the response – fastest response time.
  - **Reactive behaviors** are learned, and they are then executed without conscious thought.

    E.g., Behaviors based on "muscle memory" such as biking and skiing are reactive behaviors.
  - **Conscious behaviors** are deliberative as a sequence of previously developed behaviors.

    Notice, in ethology, the reactive behavior is the learned behavior, while in robotics, it connotes a reflexive behavior.

Reflexive Behaviors

- Reflexive behaviors are fast "hardwired" – if there is a sense, they produce the action.
- It can be categorized into three types.
  1. **Reflexes** – the response lasts only as long as the stimulus.
     - The response is proportional to the intensity of the stimulus.
  2. **Taxes** – the response to stimulus results in a movement towards or away from the stimulus,
     - e.g., moving to light, warm, etc.
  3. **Fixed-Action Patterns** – the response continues for a longer duration than the stimulus.
- The categories are not mutually exclusive.
  - An animal may keep its orientation to the last sensed location of the food source (*taxis*) even when it loses the "sight" of it (*fixed-action patterns*).

"Tactile-based" (e.g., model-based) triggering of obstacle avoidance or staircaise locomotion.

Four Ways to Acquire a Behavior

- Ethology provides insights into how animals might acquire and organize behaviors.

Konrad Lorenz and Niko Tinbergen

1. **Innate** – be born with a behavior, e.g., be pre-programmed.
2. **Sequence of innate behaviors** – be born with the sequence.
   - The sequence is logical but important.
   - Each step is triggered by the combination of the internal state and the environment.

   *It is similar to the **Finite State Machine**.*
3. **Innate with memory** – be born with behaviors that need initialization.

   E.g., a bee does not bear with the known location of the hive. It has to perform some initialization steps to learn what the hive looks like.
   - Notice, S-R (stimulus-response) types of behaviors are simple to pre-program, but it certainly should not exclude usage of memory.
4. **Learn** – to learn a set of behaviors.

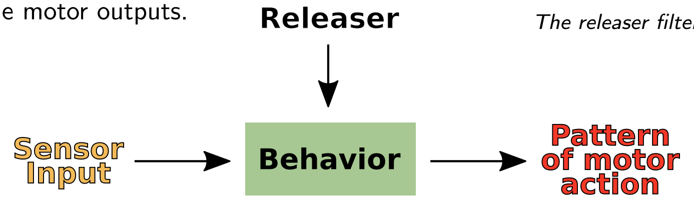
Releasing Behavior – When to Stop/Suppress the Behavior

- The **internal state** and/or **motivation** may release the behavior.

Being hungry results in looking for food.

- Behaviors can be sequenced into complex behavior.
- **Innate releasing mechanism** is a way to specify when behavior gets turned on/off.
- The releaser acts as a control signal to activate behavior.
  - If the behavior is not released, it does not respond to sensory inputs, and it does not produce the motor outputs.

The releaser filters the perception.



- The releasers can be compound – multiple conditions have to be satisfied to release the behavior.

Concurrent Behaviors

- Behaviors can execute concurrently and independently, resulting in different interactions.
  - **Equilibrium** – the behaviors seem to balance each other out.
    E.g., an undecided behavior of squirrel whether to go for food or rather run avoiding human.
  - **Dominance of one** – winner takes all as only one behavior can execute and not both simultaneously.
  - **Cancellation** – the behaviors cancel each other out.
    E.g., one behavior going to light and the second behavior going out of the light.
- It is (might) not (be) known how different mechanisms for conflicting behaviors are employed.
- However, it is important to be aware how the behaviors will interact in a robotic system.

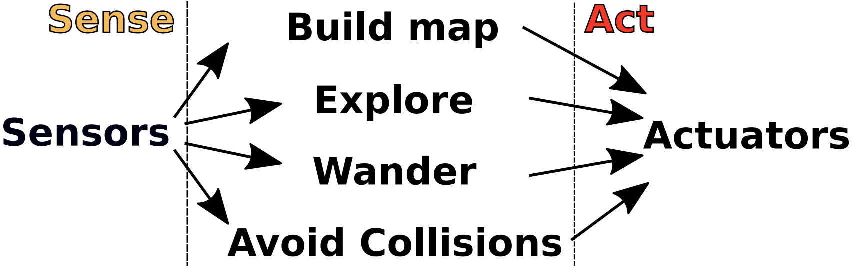
Behaviors Summary

- Behavior is a fundamental element in biological intelligence and a fundamental component of intelligence in robotic systems.
- Complex actions can be decomposed into independent behaviors that couple sensing and acting.
- Behaviors are inherently parallel and distributed.
- Straightforward activation mechanisms (such as boolean variables) may be used to simplify the control and coordination of behaviors.
- Perception filters may be used to sense what is relevant to the behavior (action-oriented perception).
- Direct perception reduces the computational complexity of sense.

  Allows actions without memory, inference, or interpretation.
- Behaviors are independent, but the output from one behavior:
  - Can be combined with another to produce the output;
  - May serve to inhibit another behavior.

Reactive Paradigm

- Reactive paradigm originates from dissatisfaction with the hierarchical paradigm (S-P-A), which is influenced by ethology.

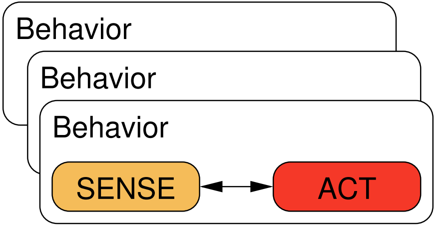


- Contrary to the S-P-A, which exhibits horizontal decomposition, the reactive paradigm (S-A) provides *vertical decomposition*.
  - Behaviors are layered, where lower layers are "survival" behaviors.
  - Upper layers may reuse the lower, inhibit them, or create parallel tracks of more advanced behaviors.

If an upper layer fails, the bottom layers would still operate.

Multiple, Concurrent Behaviors

- Strictly speaking, one behavior does not know what another behavior is doing or perceiving.



- Mechanisms for handling simultaneously active multiple behaviors are needed for complex reactive architectures.
- Two main representative methods have been proposed in the literature.
  - **Subsumption architecture** proposed by Rodney Brooks.
  - **Potential fields** methodology studied by Ronald Arkin, David Payton, et al.

Characteristics of Reactive Behaviors

1. Robots are **situated agents** operating in an ecological niche.
   - Robot has its intentions and goals; it changes the world by its actions, and what it senses influences its goals.
2. *Behaviors serve as the building blocks for robotic actions, and the overall behavior of the robot is **emergent**.*
3. *Only local, behavior-specific sensing is permitted* – usage of explicit abstract representation is avoided – **ego-centric** representation.

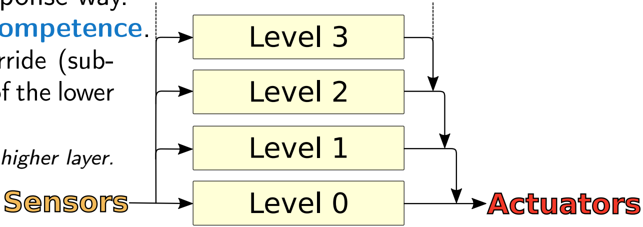
   E.g., robot-centric coordinates of an obstacle are relative and not in the world coordinates.
4. Reactive-based systems follow good software design principles – modularity of behaviors supports decomposition of a task into particular behaviors.
   - Behaviors can be tested independently.
   - Behaviors can be created from other (primitive) behaviors.
5. Reactive-based systems or behaviors are often biologically inspired.

   Under reactive paradigm, it is acceptable to mimic biological intelligence.

An Overview of Subsumption Architecture

- Subsumption architecture has been deployed in many robots that exhibit walk, collision avoidance, etc. without the "move-think-move-think" pauses of Shakey.
- Behaviors are released in a stimulus-response way.
- Modules are organized into **layers of competence**.
  1. Modules at the higher layer can override (subsume) the output from the behaviors of the lower layer.

     Winner-take-all – the winner is the higher layer.

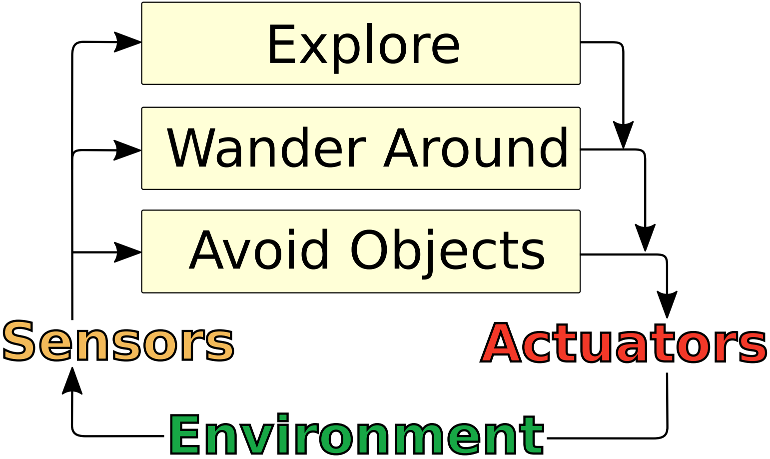


  2. Internal states are avoided.

     A good behavioral design minimizes the internal states that can be, e.g., used in releasing behavior.
  3. A task is accomplished by activating the appropriate layer that activates a lower layer and so on.
- In practice, the subsumption-based system is not easily taskable.

  It needs to be reprogrammed for a different task; however, it can serve well for the defined task.

An Example of Subsumption Architecture

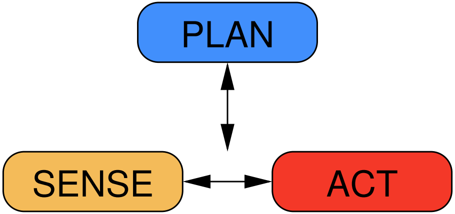


Further reading: R. Murphy, Introduction to AI Robotics.

Outline

Hybrid Paradigm

- The main drawback of reactive-based architectures is a lack of planning and reasoning about the world.
  - An example is a robot that cannot plan an optimal trajectory.
- Hybrid architecture combines the hierarchical (deliberative) paradigm with the reactive paradigm. *Beginning of the 1990's*



- Hybrid architecture can be described as **Plan**, then **Sense-Act**.
  - Planning covers a relatively long time horizon, and it uses a global world model.
  - Sense-Act covers the reactive (real-time) part of the control.

Characteristics of Reactive Paradigm in Hybrid Paradigm

- **Hybrid paradigm** is an extension of the **Reactive paradigm**.
- The term behavior in the hybrid paradigm includes reflexive, innate, and learned behaviors.

  In the reactive paradigm, it connotes purely reflexive behaviors.
- Behaviors are also sequenced over time, and more complex emergent behaviors can occur.
- **Behavioural management** – planning which behavior to use requires information outside the particular model (a global knowledge).

  Reactive behavior works without any outside knowledge.
- **Performance monitor** evaluates if the robot is making progress toward its goal.

  For example, whether the robot is moving or stuck.
  - In order to monitor the progress, the program has to know the behavior the robot is trying to accomplish.

Components of Hybrid Deliberative/Reactive Paradigm

- **Sequencer** – generates a set of behaviors to accomplish a subtask.
- **Resource Manager** – allocates resources to behaviors, which can include a selection of suitable sensors. *In reactive architectures, resources for behaviors are usually hardcoded.*
- **Cartographer** – creates, stores, and maintains a map or spatial information, a global world model, and knowledge representation. *It can be a map but not necessarily.*
- **Mission Planner** – interacts with the operator and transforms the commands into the robot term.
  - Construct a mission plan. For a mobile robot, it can consist of navigation to some place where further action is taken.
- **Performance Monitoring and Problem Solving** – it is a sort of **self-awareness** allowing the robot to monitor its progress.

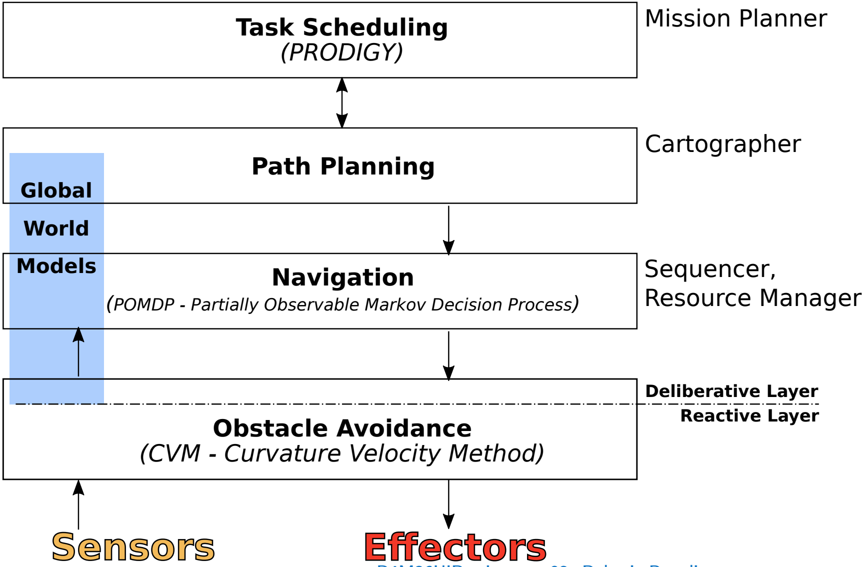
Existing Hybrid Architectures

- Managerial architectures use agents for high-level planning at the top; then there are agents for plan refinement to the reactive behaviors at the lowest level.

  E.g., Autonomous Robot Architecture, and Sensor Fusion Effects.
- State-Hierarchy architectures organize activity by the scope of the time knowledge

  E.g., 3-Tiered architectures.
- Model-Oriented architectures concentrate on symbolic manipulation around the global world. *E.g., Saphira.*
- **Task Control Architecture** (TCA) – layered architecture:
  - Sequencer Agent, Resource Manager – Navigation Layer;
  - Cartographer – Path-Planning Layer;
  - Mission Planner – Task Scheduling Layer;
  - Performance Monitoring Agent – Navigation, Path-Planning, Task-Scheduling;
  - Emergent Behavior – Filtering.

Task Architecture



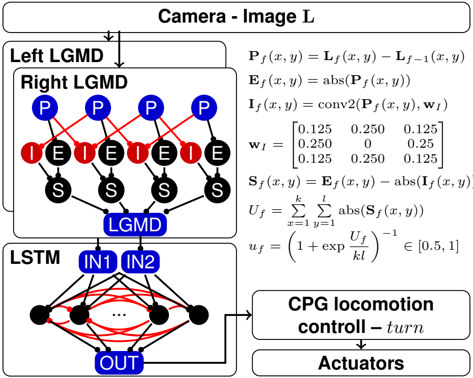
Outline

Example of Reactive Collision Avoidance

- Biologically inspired reactive architecture with vision sensor and CPG.

Notice all are hardwired into the program, and the robot goes "just" ahead with avoiding intercepting obstacles.

  - CPG-based locomotion control can be parametrized to steer the robot motion to the left or right.
  - **Avoiding collisions** with obstacles and intercepting objects is based on the visual perception inspired by the Lobula Giant Movement Detector (LGMD), which is a neural network detecting approaching objects.



$$\mathbf{P}_f(x,y) = \mathbf{L}_f(x,y) - \mathbf{L}_{f-1}(x,y)$$

$$\mathbf{E}_f(x,y) = \mathrm{abs}(\mathbf{P}_f(x,y))$$

$$\mathbf{I}_f(x,y) = \mathrm{conv2}(\mathbf{P}_f(x,y), \mathbf{w}_I)$$

$$\mathbf{w}_I = \begin{bmatrix} 0.125 & 0.250 & 0.125 \\ 0.250 & 0 & 0.25 \\ 0.125 & 0.250 & 0.125 \end{bmatrix}$$

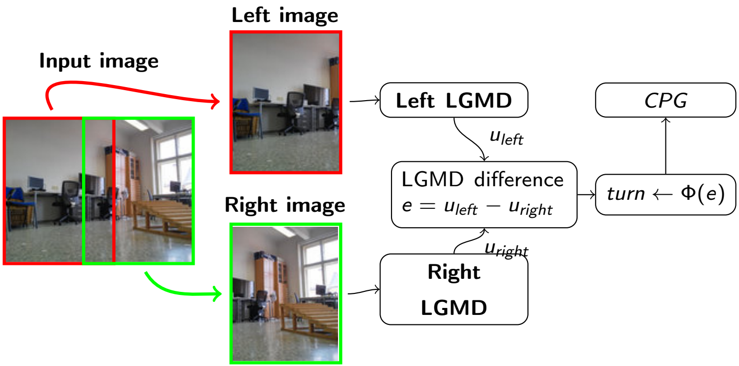
$$\mathbf{S}_f(x,y) = \mathbf{E}_f(x,y) - \mathrm{abs}(\mathbf{I}_f(x,y))$$

$$U_f = \sum_{x=1}^{k} \sum_{y=1}^{l} \mathrm{abs}(\mathbf{S}_f(x,y))$$

$$u_f = \left(1 + \exp \frac{U_f}{kl}\right)^{-1} \in [0.5, 1]$$

Čížek, Milička, Faigl (IJCNN 2017)

LGMD-based Collision Avoidance – Control Rule

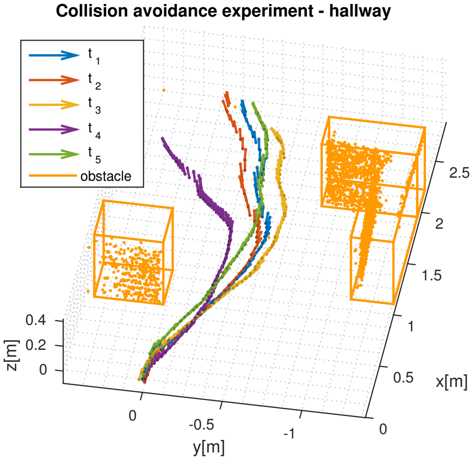


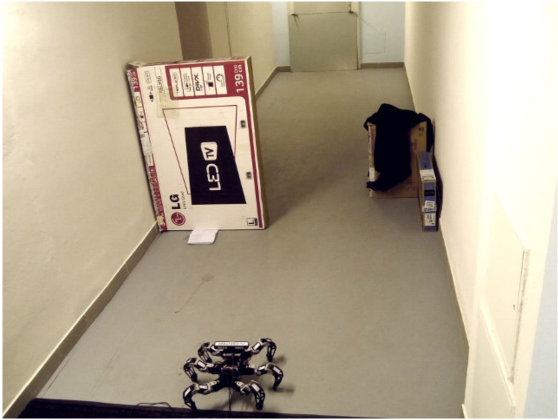
A mapping function: Φ from the output of the LGMD vision system to the *turn* parameter of the CPG

$$\Phi(e) = \begin{cases} 100/e & \text{for } \text{abs}(e) \geq 0.2 \\ 10000 \cdot \text{sgn}(e) & \text{for } \text{abs}(e) < 0.2 \end{cases} .$$

Čížek, Milička, Faigl (IJCNN 2017)

Example of LGMD-based Collision Avoidance





- LGMD output together with the proposed mapping function provide a smooth motion of the robot.

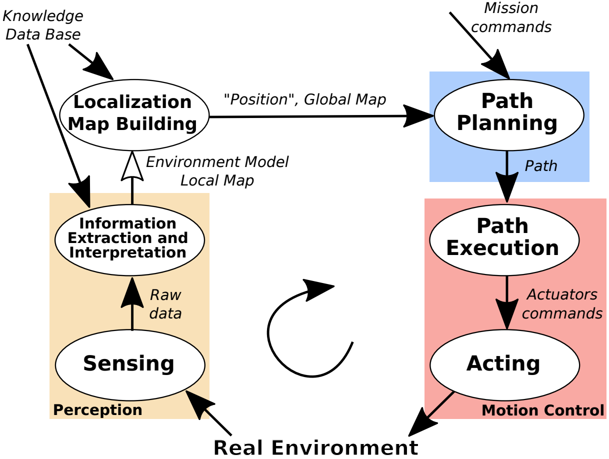
Čížek, Faigl (Bioinspiration & Biomimetics, 2019)

Outline

- Robotics Paradigms
- Hierarchical Paradigm
- Reactive Paradigm
- Hybrid Paradigm
- Example of Collision Avoidance
- Robot Control

A Control Schema for a Mobile Robot

- A general control schema for a mobile robot consists of **Perception Module**, **Localization and Mapping Module**, **Path Planning Module**, and **Motion Control Module**.



*In B4M36UIR, we focus on **Path Planning Module**.*

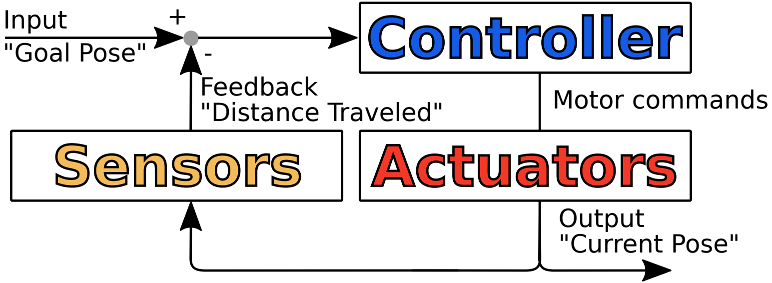
Motion Control

- An important part of navigation is the execution of the planned path.
- Motion control module is responsible for the path realization.
  - **Position control** aims to navigate the robot to the desired location.
  - **Path-Following** is a controller that aims to navigate the robot along the given path.
  - **Trajectory-Tracking** differs from the path-following in that the controller forces the robot to reach and follow a time parametrized reference (path).

  E.g., a geometric path with an associated timing law.
- The controller can be realized as one of two types:
  - **Feedback controller**;
  - **Feedforward controller**.

FeedBack Controller

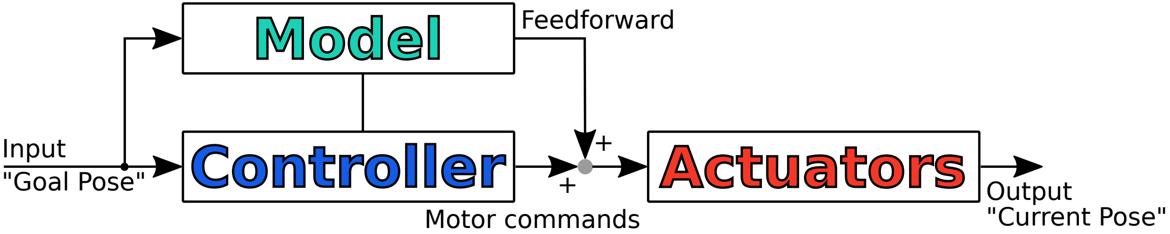
- The difference between the goal pose and the distance traveled so far is the error used to control the motors.
- The controller commands the motors (actuators), which change the real robot pose.
- Sensors, such as encoders for a wheeled robot, provide information about the traveled distance.



Notice, the robot may stuck, but it is not necessarily detected by the encoders.

Feed-Forward Controller

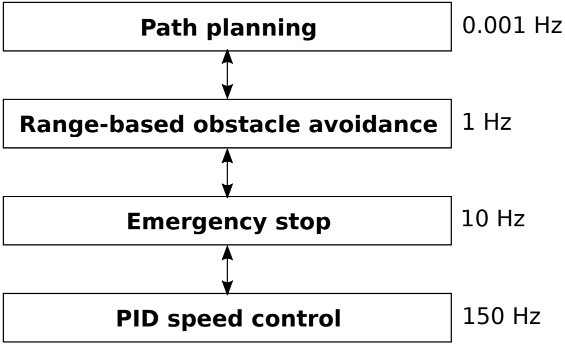
- In the feed-forward controller, there is no feedback from the real-world execution of the performed actions.
- Instead of that, a model of the robot is employed in the calculation of the expected effect of the performed action.



In this case, we fully rely on the assumption that the actuators will be performed as expected.

Temporal Decomposition of Control Layers

- The robot control architecture typically consists of several modules (behaviors) that may run at different frequencies.
- Low-level control is usually the fastest, while path planning is slower as the robot needs some time to reach the desired location.
- An example of possible control frequencies of different control layers.



Adapted from Introduction to Autonomous Mobile Robots, R. Siegwart et al.

Summary of the Lecture

Topics Discussed

- Robotic Paradigms:
  1. Hiearchical paradigm;
  2. Reactive paradigm;
  3. Hybrid Hiearchical/Reactive paradigm.
- Example of Reactive architecture – collision avoidance.
- Robot Control.

- Next: Path and Motion Planning.

Topics Discussed

- Robotic Paradigms:
    1. Hiearchical paradigm;
    2. Reactive paradigm;
    3. Hybrid Hiearchical/Reactive paradigm.
- Example of Reactive architecture – collision avoidance.
- Robot Control.

- Next: Path and Motion Planning.