Intelligent Robotics

MAP-I Doctoral Programme in Computer Science

Luís Paulo Reis (University of Porto)

Nuno Lau (University of Aveiro)

http://paginas.fe.up.pt/~lpreis/ir2007_08/

Questions?

- Questions about Introduction?
- Questions about Sensors and Perception?
- Questions about Ciber-Mouse Simulator?
- Any questions about Assignment #1?

“A robot in every home”

- In the January 2007 issue of *Scientific American* Bill Gates wrote:
  - As I look at the trends that are now starting to converge, I can envision a future in which robotic devices will become a nearly ubiquitous part of our day-to-day lives
  - The challenges facing the robotics industry are similar to those we tackled in computing three decades ago
  - Robotics companies have no standard operating software that could allow popular application programs to run in a variety of devices
  - The standardization of robotic processors and other hardware is limited, and very little of the programming code used in one machine can be applied to another
  - The goal of Microsoft Robotic Studio is to deliver a set of programming tools that would provide the essential plumbing so that anybody interested in robots with even the most basic understanding of computer programming could easily write robotic applications that would work with different kinds of hardware

Some Robotic Issues

- Agent/Robot control architectures
- Behavior-based systems
- Sensors and Perception
- Representation Issues
- Adaptation and Learning
- Path planning and Navigation
- Localization and Mapping
- Intelligent Planning
- Multi-robot systems

Some Robotic Issues

- How do I interpret my sensor feedback to determine my current state and surroundings? [sensor processing/perception]
- Where am I? [localization]
- How do I make sense of noisy sensor readings? [uncertainty management]
- How do I fuse information from multiple sensors to improve my estimate of the current situation? [sensor fusion]
- What assumptions should I make about my surroundings? [structured/unstructured environments]
- How do I know what to pay attention to? [focus-of-attention]

Some Robotic Issues

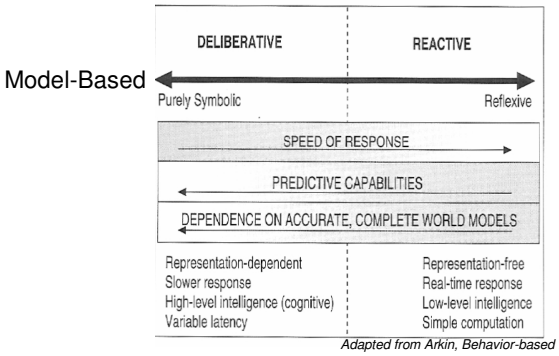
- What should my control strategy be to ensure that I respond quickly enough? [control architecture]
- How should I make decisions? [reasoning, task arbitration]
- Where do I want to be, and how do I get there? [path planning, navigation]
- I have lots of choices of actions to take --what should I do in my current situation? [action selection]
- How should I change over time to respond to a dynamic environment? [learning, adaptation]
- Why doesn't the same action that worked in this situation beforenot work now? [hidden state]
- How can I change my state from A to B [planning]
- How should I work with other robots? [multi-robot cooperation, communication]

Robotic Architecture - definition(s)

- An architecture provides a principled way of **organizing a control system**. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved [Mataric, 1992]
- Robotic architecture usually refers to **software**, rather than hardware [Arkin, 1998]
- How the job of generating actions from percepts is organized [Russel and Norvig, 2002]

Spectrum of Robot Control Architectures

- Deliberative control: “think hard, then act”
- Reactive control: “don't think, (re)act”
- Hybrid control: “think and act in parallel”



Adapted from Arkin, Behavior-based Robotics (MIT Press, 1998)

Typical Organizations

- Typical organizations:
  - Hierarchical
  - Behavior-based / Reactive
  - Hybrid
- Reactive
  - Cheap low memory processing
  - No world model
- Deliberative
  - Making maps
  - Selecting behaviors
  - Monitor performance
  - Planning
- Hybrid deliberative/reactive paradigm

Deliberative Architectures

- In **Deliberative control**, the robot takes all of the available sensory information, and all of the internally stored knowledge it has, and it thinks ("reasons") about it in order to create a plan of action.
- To do so, the robot must search through potentially all possible plans until it finds one that will do the job. This requires the robot to look ahead, and think in terms of: "if I do this next, and then this happens, then what if I this next, then this happens,..." and so on.
- This can take a long time, which is why if the robot must react quickly, it may not be practical. However, if there is time, this allows the robot to act strategically.

Reactive Architectures

- **Reactive Control** is a technique for tightly coupling sensory inputs and effector outputs, to allow the robot to respond very quickly to changing and unstructured environments.
- Think of reactive control as as "stimulus-response". This is a powerful control method: many animals are largely reactive.
- Limitations to this approach are that such robots, because they only look up actions for any sensory input, do not usually keep much information around, have no memory, no internal representations of the world around them, and no ability to learn over time.

Hybrid Architectures

- In **Hybrid Control**, the goal is to combine the best of both Reactive and Deliberative control. In it, one part of the robot's "brain" plans, while another deals with immediate reaction, such as avoiding obstacles and staying on the road.
- The challenge of this approach is bringing the two parts of the brain together, and allowing them to talk to each other, and resolve conflicts between the two.
- This requires a "third" part of the robot brain, and as a result these systems are often called "three-layer systems“

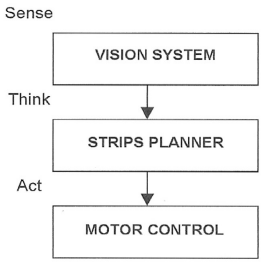
Adapted from http://www-robotics.usc.edu/~maja/robot-control.html

Deliberative vs Reactive

- No single approach is "the best" for all robots; each has its strengths and weaknesses
- Control requires some unavoidable trade-offs because:
  - Thinking is slow
  - Reaction must be fast
  - Thinking allows looking ahead (planning) to avoid bad actions
  - Thinking too long can be dangerous (e.g., falling off a cliff)
  - To think, the robot needs (a lot of) accurate information
  - The world keeps changing as the robot is thinking, so the slower it thinks, the more inaccurate its solutions
- As a result of these trade-offs, some robots don't think at all, while others mostly think and act very little.
  - **It all depends on the robot's task and its environment!**

Model-based - Deliberative

- Sense-plan-act paradigm: dominant view in the AI community was that a control system for an autonomous mobile robot should be decomposed into three functional elements [Nilsson, 1980]:
  - a sensing system (translate raw sensor input into a world model,
  - a planning system (take the world model and a goal and generate a plan to achieve the goal)
  - and an execution system (take the plan and generate the actions it prescribes)
- Perception is the establishment and maintenance of correspondence between the internal world model and the external real world [Albus 1991].
- Action results from reasoning over the world model.
- Perception is not tied directly to action.



Reactive

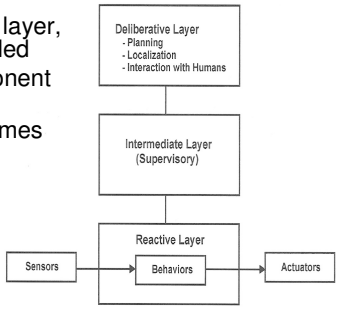
- General assumptions:
  - The environment lacks temporal consistency and stability
  - The robot's immediate sensing is adequate for the task at hand
  - It is difficult to localize a robot relative to a world model
  - Symbolic representational world knowledge is of little or no value

Behavior-Based - Reactive

- Common features:
  - Emphasis on the importance of coupling sensing and action tightly.
  - Avoidance of representational symbolic knowledge (because the world can change over time and uncertainty is hard to model).
  - Decomposition into contextually meaningful units (behaviors or situation-action pairs).
- Distinctions:
  - Granularity of behavioral decomposition
  - Basis for behavior specification (ethological, situated activity, or experimental)
  - Response encoding (e.g., discrete or continuous)
  - Coordination methods (e.g., competitive vs cooperative)
  - Programming methods, language, reusability

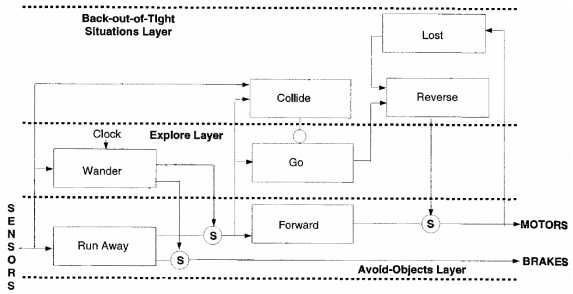
Hybrid Architectures

- Combine the responsiveness, robustness, and flexibility of purely reactive systems with more traditional symbolic/deliberative methods
- Reason: purely reactive systems lack the ability to take into account a priori knowledge (e.g. about the world) and to keep track of the history (memory)
- Typical three-layer hybrid architecture
  - Bottom layer is the reactive/behavior-based layer, in which sensors/actuators are closely coupled
  - Upper layer provides the deliberative component (e.g., planning, localization)
  - The intermediate between the two is sometimes called supervisory layer
- Examples of coupling between planning and reactive layers:
  - Planning to guide reaction: planning sets reactive system parameters.
  - Coupled: planning and reacting are concurrent activities, each guiding the other



Subsumption Architecture [Brooks 1986]

- Behaviors are augmented finite state machines (AFSM)
- Stimulus or response signals can be suppressed or inhibited by other active behaviors; a reset input returns the behavior to its start conditions
- Each behavior is responsible for its own perception of the world
- Arrangement in layers: lower layers have no awareness of higher layers



Brooks – Behavior languages

Brooks has put forward three theses:

1. Intelligent behavior can be generated *without* explicit representations of the kind that symbolic AI proposes
2. Intelligent behavior can be generated *without* explicit abstract reasoning of the kind that symbolic AI proposes
3. Intelligence is an *emergent* property of certain complex systems

Brooks – Behavior languages

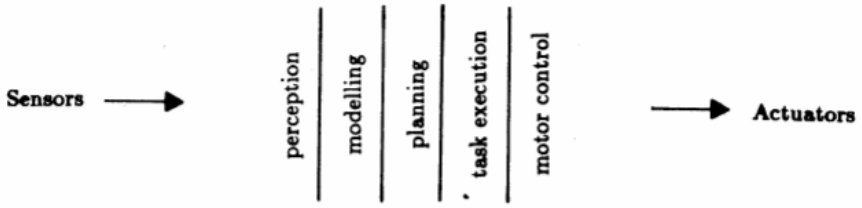
He identifies two key ideas that have informed his research:

1. Situatedness and embodiment: 'Real' intelligence is situated in the world, not in disembodied systems such as theorem provers or expert systems
2. Intelligence and emergence: 'Intelligent' behavior arises as a result of an agent's interaction with its environment. Also, intelligence is 'in the eye of the beholder'; it is not an innate, isolated property

Brooks – behavior languages

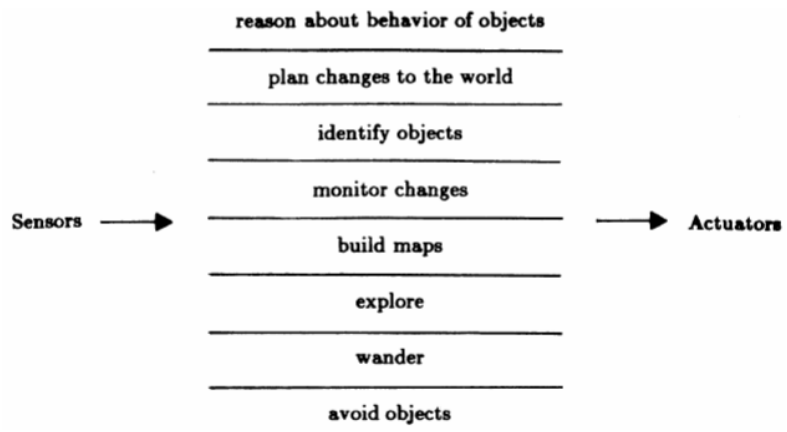
- To illustrate his ideas, Brooks built some based on his *subsumption architecture*
- A subsumption architecture is a hierarchy of task-accomplishing *behaviors*
- Each behavior is a rather simple rule-like structure
- Each behavior 'competes' with others to exercise control over the agent
- Lower layers represent more primitive kinds of behavior (such as avoiding obstacles), and have precedence over layers further up the hierarchy
- The resulting systems are, in terms of the amount of computation they do, *extremely* simple
- Some of the robots do tasks that would be impressive if they were accomplished by symbolic AI systems

A Traditional Decomposition of a Mobile Robot Control System into Functional Modules



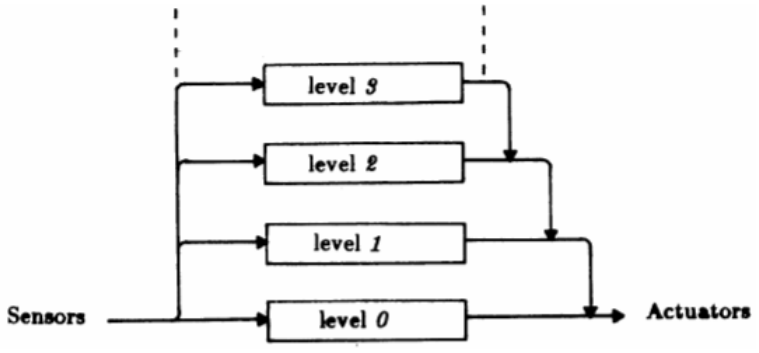
From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

A Decomposition of a Mobile Robot Control System Based on Task Achieving Behaviors



From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

Layered Control in the Subsumption Architecture



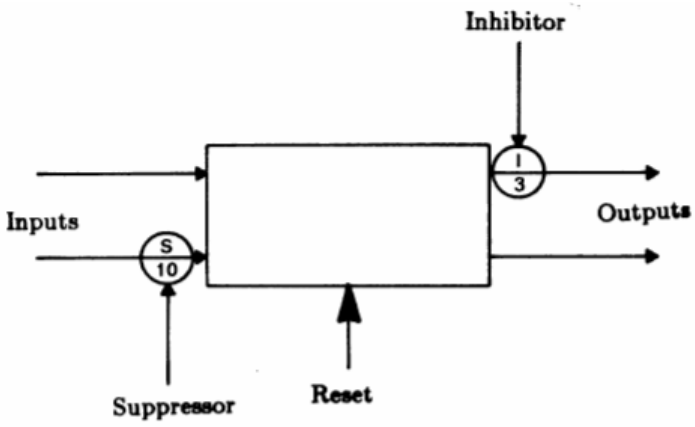
From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

Example of a Module – Avoid

```
(defmodule avoid
  :inputs (force heading)
  :outputs (command)
  :instance-vars (resultforce)
  :states
    ((nil (event-dispatch (and force heading) plan))
     (plan (setf resultforce (select-direction force heading))
           go)
     (go (conditional-dispatch (significant-force-p resultforce 1.0)
                               start
                               nil))
     (start (output command (follow-force resultforce))
            nil)))
```

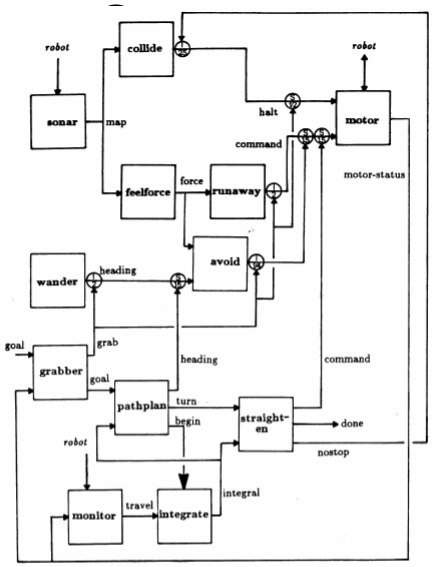
From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

Schematic of a Module



From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

Levels 0, 1, and 2 Control



From Brooks, "A Robust Layered Control System for a Mobile Robot", 1985

Steels' Mars Explorer

- Steels' Mars explorer system, using the subsumption architecture, achieves near-optimal cooperative performance in simulated 'rock gathering on Mars' domain:
  - *The objective is to explore a distant planet, and in particular, to collect sample of a precious rock. The location of the samples is not known in advance, but it* is *known that they tend to be clustered.*

Steels' Mars Explorer Rules

- For individual (non-cooperative) agents, the lowest-level behavior, (and hence the behavior with the highest "priority") is obstacle avoidance:
  - *if* detect an obstacle *then* change direction (1)
- Any samples carried by agents are dropped back at the mother-ship:
  - *if* carrying samples *and* at the base *then* drop samples (2)
- Agents carrying samples will return to the mother-ship:
  - *if* carrying samples and *not* at the base *then* travel up gradient (3)
- Agents will collect samples they find:
  - *if* detect a sample *then* pick sample up (4)
- An agent with "nothing better to do" will explore randomly:
  - *if* true *then* move randomly (5)

Assignment #2

- Develop a simple Subsumption Architecture for a Ciber-Mouse Agent:
  - *if* detect an obstacle *then* change direction (1)
  - *if at the target* signal end (2)
  - *if not* at the target *then rotate to* find the target (3)
  - *if* true *then* moves in front/randomly (4)