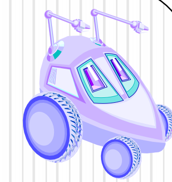
Lecture 1 - 2

The Hierarchical Paradigm

Introduction to AI Robotics (Ch. 2)



Quote of the Week

“Don't tell people how to do things. Tell them what to do and let them surprise you with their results.”

George Patton

Announcements

- Lab 1 - Locomotion and Odometry is due on Thursday, 3/11/10
- The lab memo and code is due on Angel by midnight on Thursday, 3/11/10
- Quiz 2 on Ch. 2, Lecture 1 – 2 on Monday, 3/15/10

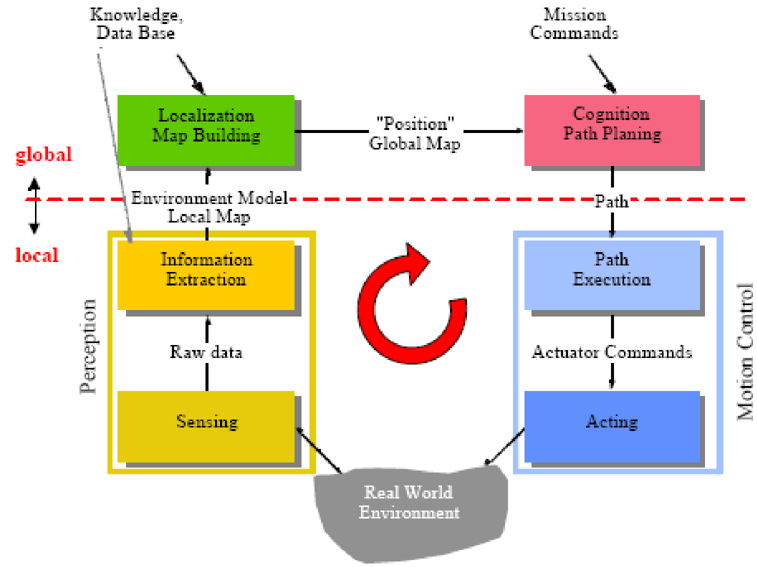
Objectives

Upon completion of this lecture the student should be able to:

- Describe the hierarchical paradigm in terms of the 3 robot primitives
- Understand the meaning of the terms: precondition, closed world assumption, open world, frame problem
- List two advantages and disadvantages of the Hierarchical Paradigm

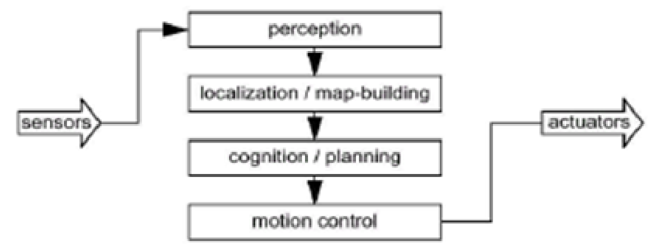
Control of Mobile Robots

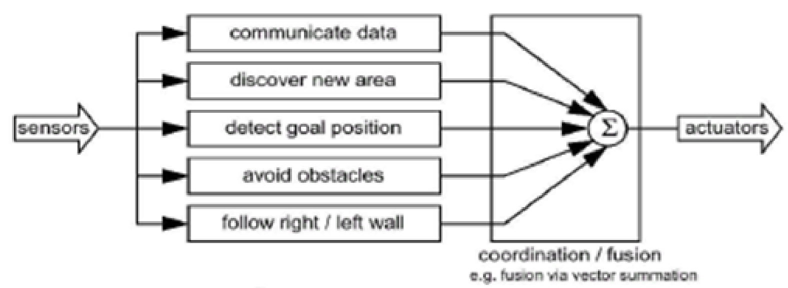
- Controls
  - Dynamically changing
  - No compact model available
  - Multiple sources of error
- Most functions are ‘local’ and do not involve localization or cognition
- Localization and global path planning are slower and should be performed only when needed
- This is similar to what human beings do

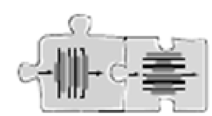


Control Approaches

- Classical AI Control
  - Complete modeling
  - Function based
  - Horizontal decomposition
- New AI Control
  - Sparse or no modeling
  - Behavior-based
  - Vertical decomposition (bottom up)
- Possible solution
  - Combine approaches







Control Architectures

- The controller is the brains of the robot. Robot control is the means by which sensing and action of a robot are coordinated.
- Feedback control is good for low level control of actuators (wall following, obstacle avoidance)
- More complex tasks require a control architecture
  - This may involve using multiple controllers
  - This may involve arbitration of controllers for complex tasks
- A robot ***control architecture*** provides the guiding principles and constraints for organizing a robot's control system (i.e. brain)

Software versus Hardware

- Robot control can take place in hardware or software
- More complex controllers are typically implemented in software
- Hardware is good for fast and specialized uses
- Software is good for flexible, more general programs
- Brains use programs to solve problems and to achieve goals
- Solving a problem using a finite step by step procedure is called an ***algorithm***

Types of Control Architectures

- Hierarchical (Deliberative) Control
- Reactive (Behavior-based) Control
- Hybrid Deliberative/Reactive Control
- These architectures differ in the way they handle
  - Time
  - Modularity
  - Representation

Time

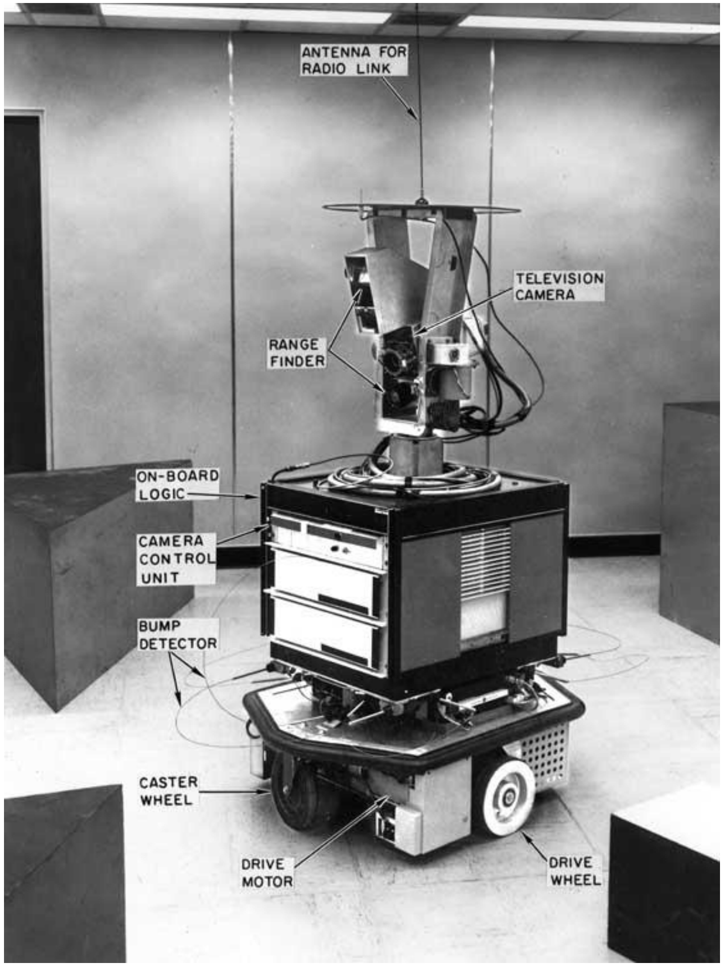
- How fast do things happen?
- Do all components of the controller run at the same speed?
- ***Time*** refers to how fast the robot responds to the environment compared with how quickly it can sense and think
- ***Deliberative control*** looks into the future so it works on a long time-scale
- ***Reactive control*** responds to the immediate, real- time demands of the environment without looking into the past or the future
- ***Hybrid control*** combines the long time scale of deliberative control and the short time scale of reactive control
- ***Behavior-based control*** works to bring the time-scales together in a different way than hybrid control

Modularity

- What are the components of the control system?
- What can talk to what?
- ***Modularity*** refers to the way the control system is broken into pieces, components, or modules.
- ***Modularity*** also refers to how the modules interact with each other to product the robot's overall behavior
- ***Deliberative control*** has a control system with sensing, planning and acting modules that work in sequence
- In ***reactive control***, multiple modules are all active in parallel and can send messages to each other in various ways
- In ***hybrid control***, the three main modules are the deliberative, reactive and the connection in between
- In ***behavior-based control***, there are more than 3 modules and they work in parallel and talk to each other

Shakey (AI-Inspired robot)

- Shakey was built at the Stanford Research Institute in 1970
- First mobile robot to use AI techniques
- It was named Shakey because it shook when it executed plans to move in the world
- It was controlled by a large computer
- It used spatial data from camera and laser range measurements to recognize objects
- It created a path to the object
- Pushed the objects (blocks) over when found



Other AI-Inspired Robots

- The Stanford CART developed in 1977 by Hans Moravec used vision-based navigation
- This robot was a cart on bicycle wheels
- It moved slowly because of the difficulty of processing data from vision and computer processors
- The CMU Rover developed in 1983 by Hans Moravec used a camera and ultrasound sensing for navigation

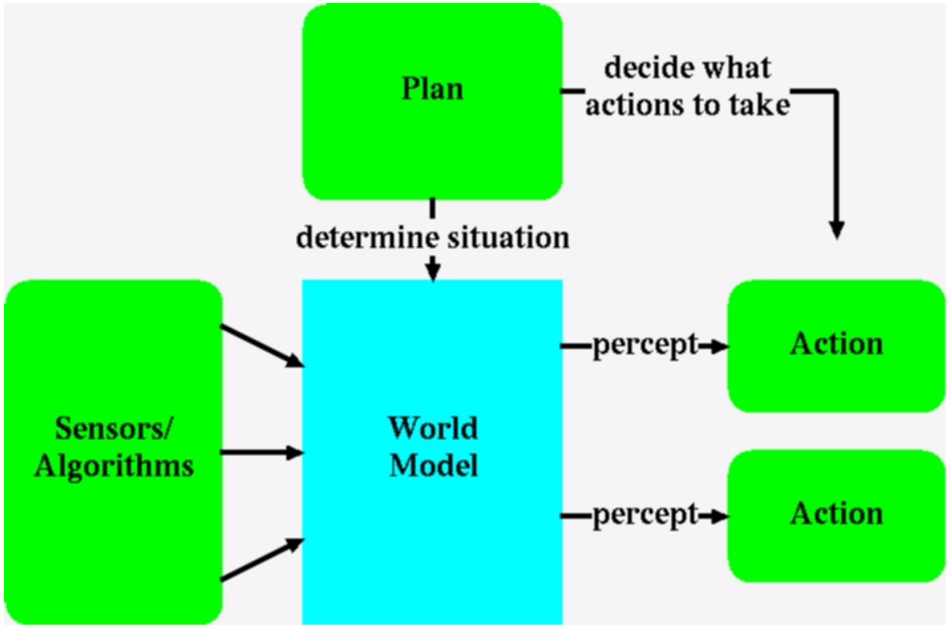
Attributes of the Hierarchical Paradigm

- It is sequential and orderly
- The robot **senses** the world and constructs a global map
- The robot **plans** all directives to reach a goal based upon the map
- The robot **acts** to carry out the directives

Robot Primitives	Input	Output
SENSE	Sensor Data	Sensed information
PLAN	Information (sensed/ cognitive)	Directives
ACT	Directives	Actuator commands

World Model

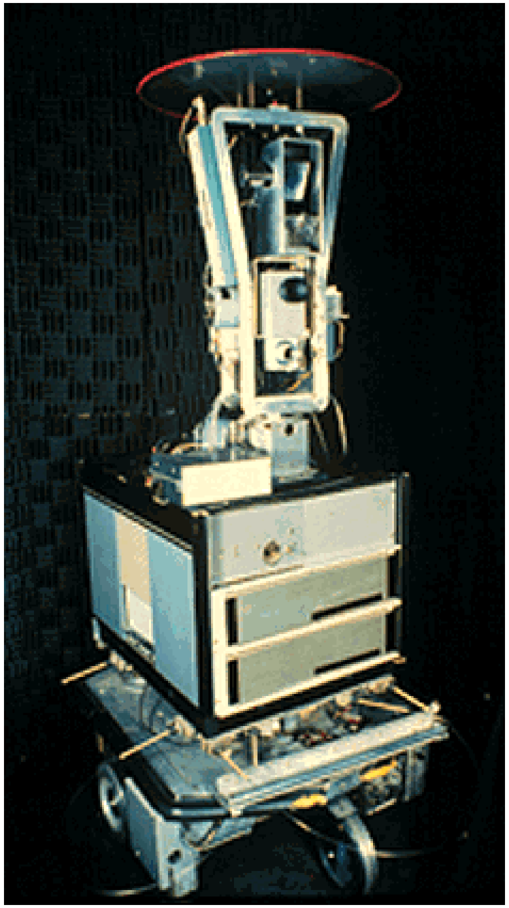
All sensor observations are fused into one global data structure, the **world model**.



The world model includes:

- a priori representation
- sensed information
- cognitive knowledge

Strips



- Shakey used the Strips algorithm for planning how to accomplish goals
- This was a *means-end analysis* that chose actions to reduce the difference between the initial state and goal state
- Inspired by cognitive behavior in humans

Strips Difference Table

Difference	Operator	Pre-conditions	Add-list	Delete-list
$d \leq 200$	fly		at Y at airport	at X
$100 < d < 200$	train		at Y at station	at X
$d \leq 100$	drive_rental drive_personal	at airport at home		
$d < 1$	walk			

- Designer must set up the
  - World model representation, difference table and difference evaluator
- Strips assumes ***closed world model***
- Strips suffers from a ***frame problem***

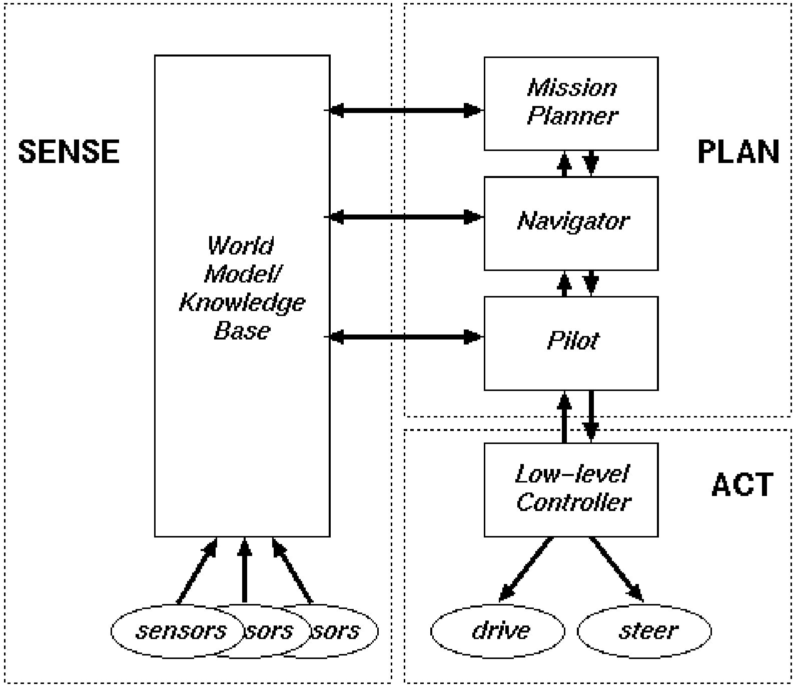
Hierarchical Representative Architectures

- There are 3 hierarchical paradigms
  - Top down (plan)
  - Control (measure error)
  - Planning (world models)

- Nested Hierarchical Controller (NHC) by Meystel
- NIST Realtime Control System (RCS) by Albus

NHC

- Easily identified sense, plan, act
- Key contribution is the decomposition of the planner into 3 components
- Different from Strips because it interleaves planning and acting
- One disadvantage is that the planning is only appropriate for navigation tasks



NIST RCS

- Designed by Albus for intelligent industrial manipulators
- RCS used NHC in the design
- The main difference was the sensory perception module used preprocessing or ***feature extraction*** between the sensor and fusion into the world model
- Well-suited for semi-autonomous control
- The human operator provides the world model, decides the mission, and decomposes it into a plan and actions
- The robot carries out the actions
- As robotics advances, the robot moves up the autonomy hierarchy

Advantages and Disadvantages

- Provided an ordering of the relationship between sense, plan, act
- The disadvantage was planning and updating a global world model was slow
- Sensing and acting were disconnected
- The dependence on the world model had a frame problem
- Uncertainty such as sensor noise and actuator error were not addressed

Time Scale Drawback

- It takes a long time to search a large state space
- Combinations of analog and digital sensory data creates a large sensor state space
- When the sensor state space combines with internal models or representations the state space is very large and hard to search
- If the planning is slow compared to the robot’s motion it has to stop and wait for the plan to finish
- To make progress, the robot must plan rarely and move as much as possible which is open loop control and bad for dynamic environments

Space Drawback

- It takes a great deal of space or memory storage to represent the robot’s state space representation
- Computer memory is cheap but all memory is finite and some algorithms may run out of it

Information Drawback

- The planner assumes that the state space representation is accurate and up to date
- The plan will not be useful is this is not a valid assumption
- The representation used must be updated and check as often as necessary to keep it accurate for the task
- Updating the plan for a real environment requires taking time to update the world model

Plan Usefulness

- The plan is only useful if
  - the environment does not change during execution in a way that affects the plan
  - the robot knows the state of the world and of the plan it is in at all times
  - the robot’s effectors are accurate enough to execute each step of the plan

Fate of Deliberative Systems

- Due to these challenges, purely deliberative architectures are no longer used for the majority of physical robots
- Robot surgery is one system that can use deliberative systems because it requires a great deal of advance planning and no time pressure and the environment (patient's body) is kept perfectly static
- Deliberative control is also used for AI in chess systems