
Formal Methods for Robot Programming

reading group - Fall 2003

Week 3 - Behavior-Based Systems

Speakers:

Uluç Saranlı
Clark Haynes

Organizers:

Uluç Saranlı
Alfred Rizzi
Frank Pfenning

► Today's agenda

- Subsumption architecture
 - Brooks, R. *A Robust Layered Control System for a Mobile Robot*. Technical report 864, MIT 1985
- Motor schemas and behavior based robotics
 - Arkin, R. *Motor Schema-Based Mobile Robot Navigation*, in International Journal of Robotics Research 8(4):92-112, 1989.

► Motivating principles

- Multiple goals
 - Usually conflicting and prioritized
- Multiple sensors
 - Must make decisions under unreliable, conflicting and redundant sensory information
- Robustness
 - Must be able to deal with internal failures and environmental changes
- Additivity
 - extensibility with respect to sensors and capabilities

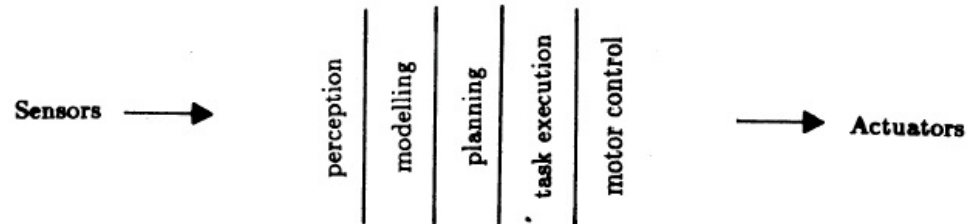
▶ Starting assumptions

- Complex behavior need not be a product of an extremely complex control system
- Keep things simple : interfaces and components
- Map making is crucial
- Use 3D representation of the environment
- Avoid using absolute coordinate systems
- Do not rely on an engineered environment
- Vision is better than sonars
- Quick recovery from failures is essential

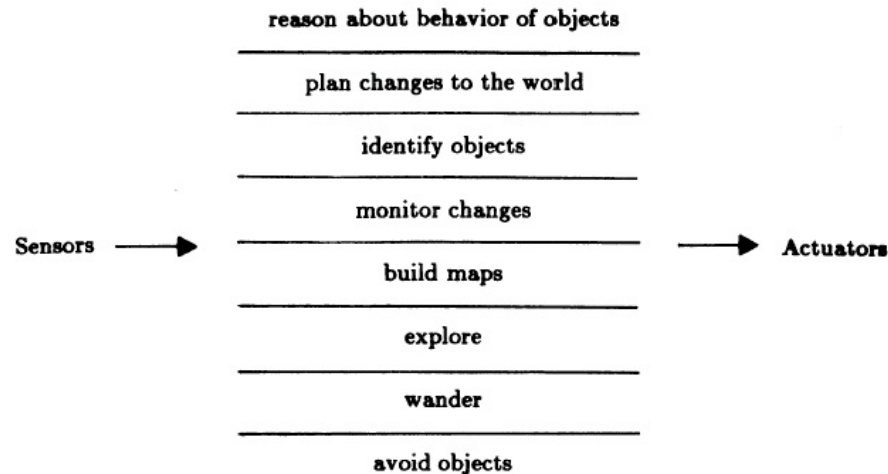
► Experimental platform

- Wheeled mobile platform
- 12 sonars and 2 tilt head cameras
- Small onboard CPU for local control, offboard machines for LISP processing and vision
- Communications through a radio link and video transmitter

► Levels of competence



traditional decompositions



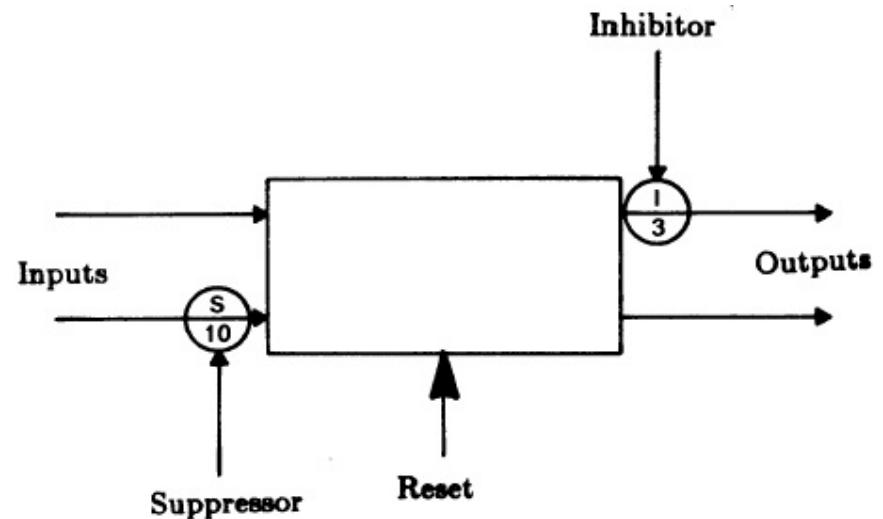
proposed decomposition

► The subsumption architecture

- Layers of control are built to achieve levels of competence
- Each layer is able to examine data from lower levels and inject data into their internal interfaces
- Additional layers never modify the structure of lower levels

► Structure of layers

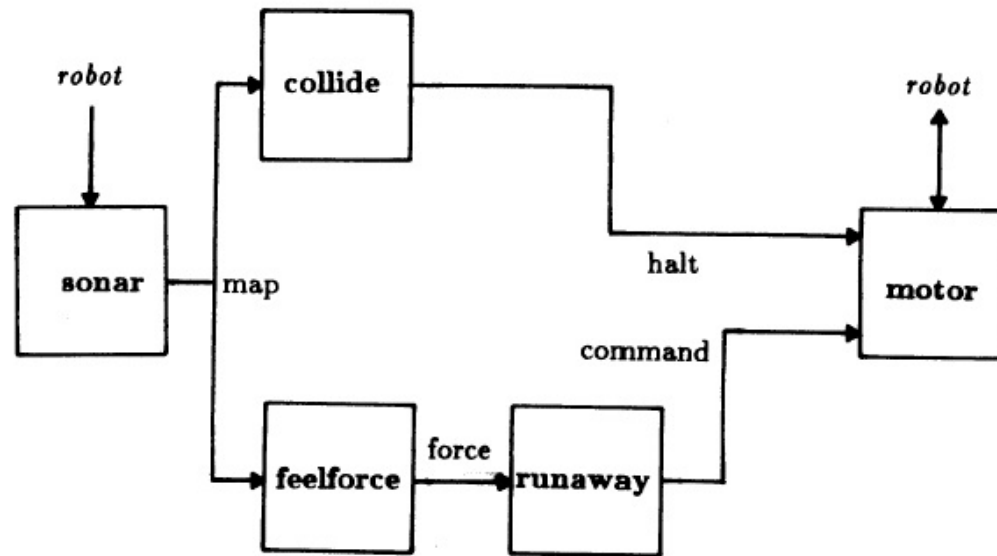
- Composed of “processors”: state machines with internal data and timers.
- Message passing
- 4 types of states:
 - Output
 - Side-effect
 - Conditional dispatch
 - Event dispatch
- Inhibitors and suppressors for higher levels to mess with



► Module example

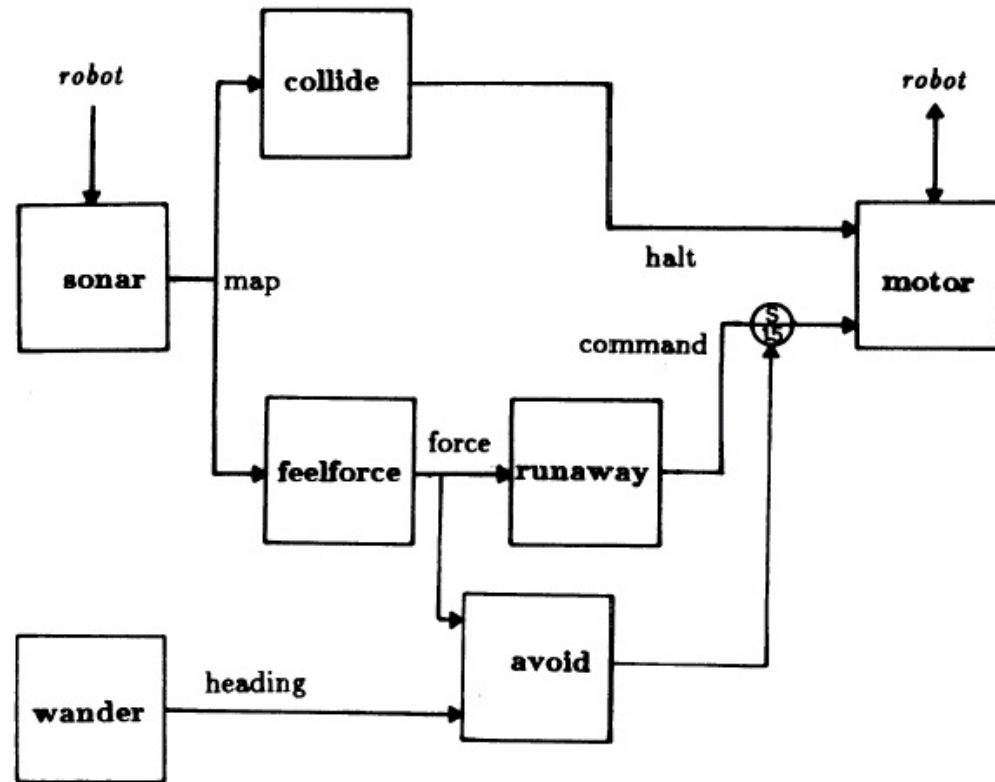
```
(defmodule avoid
  :inputs (force heading)
  :outputs (command)
  :instance-vars (resultforce)
  :states
    ((nil (event-dispatch (and force heading) plan))
     (plan (setf resultforce (select-direction force heading))
            go)
     (go (conditional-dispatch (significant-force-p resultforce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

► Example system: Level 0



- **motor**: commands robot actuators
- **sonar**: reads and filters sonars, builds robot centered local map
- **collide**: monitors sonar readings, halts robot to avoid collisions
- **feelforce**: computes repulsive force vector away from obstacles
- **runaway**: reads output from feelforce and commands the motors

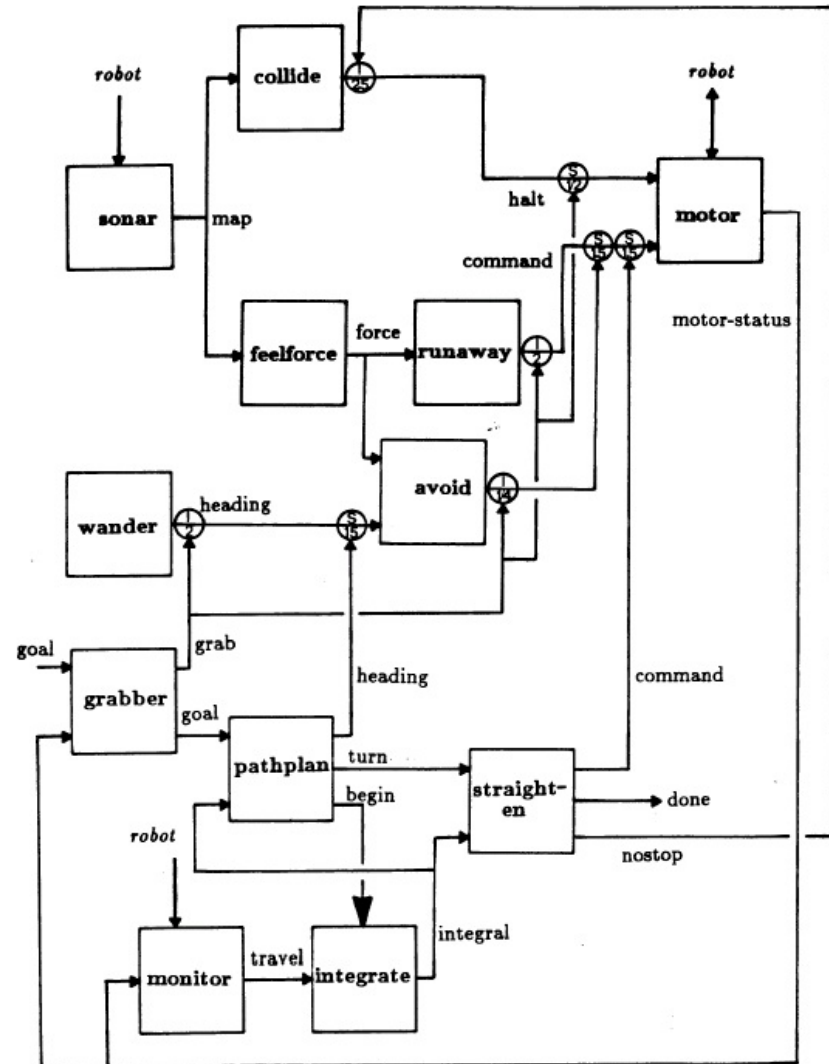
► Example system: Level 1



- **wander**: generates new headings every 10 seconds
- **avoid**: combines repulsive force and heading to produce motor command

► Example system: Level 2

- **grabber**: suppresses lower level motor commands
- **monitor**: watches motor execution and estimates displacement
- **integrate**: computes position
- **pathplan**: takes a goal and chooses headings
- **straighten**: perform orientational commands from pathplan by direct commands to the motors



► Does it work?

- Implemented in simulation for the mobile robot
 - Seems to accomplish the desired task
 - Few words about performance, not much data
- Decentralized control of hexapod locomotion
 - Implements basic behaviors such as walking, maintaining balance, target following
- Others...

► Discussion

- Pros...

- Cons...