Adaptive real-time learning of robot controllers

Rod Goodman - Gaea Corporation

Owen Holland – University of Essex, UK

In collaboration with:

Igor Aleksander -

Imperial College, London, UK

Alan Winfield -

University of the West of England, UK

Alcherio Martinoli–

EPFL, Lausanne, Switzerland

Michel Lauria –

Université de Sherbrooke, Canada

Frederik Linaker –

University of Sheffield, UK

Autonomous Robots

How to learn controllers – as opposed to hand crafted

- Learning – Adaptation- Reinforcement
- Explicit internal representations
- Environment models - Self models
- Model based predictive control
- Novelty detection
- Attention - Awareness
- Neural Networks - Genetic Algorithms
- Sensory processing
- Collective robotics - Swarm intelligence

Sony Dream Robot

EPFL “Shrimp” Robot

Goodman Lab - Caltech

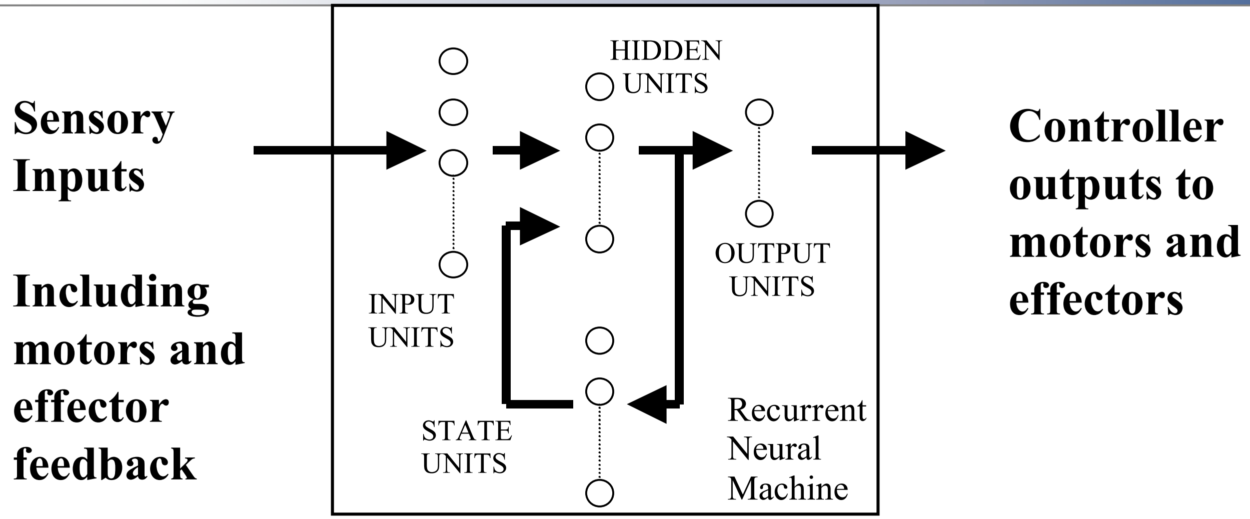
Robot Controllers

- Animal and human brains evolved to *control behavior* in a changeable and partially knowable environment.
- The goal of the controller is to produce the agent's next *action*.
- The agent uses sensory input, memory, goals, drives, to produce the correct *action* given the *current state* of the *environment*.
- There is only *one* action at a time.
- Incorrect or multiple actions are very obvious and can damage the robot quickly. (Parkinson's, Huntington's, Tourette's)
- The action may *change* the environment.
- Good control requires the ability both to *predict* events, and to exploit those predictions.
- Controllers are *layered* in increasing *levels* of abstraction.
- The best such control systems known to engineers are adaptive model-based predictive controllers.

Controllers should be able to:

- Learn models of the environment, the *self*, and of the interaction of the self with the environment.
- Adapt models *automatically* based on experience.
- Deal with *novel* situations automatically, and assimilate the new experience.
- Manipulate models *internally* to plan actions and goals.
- Make their internal models and reasoning *visible* in human terms.
- Be able to interact, model, and collaborate on tasks with other similar agents.

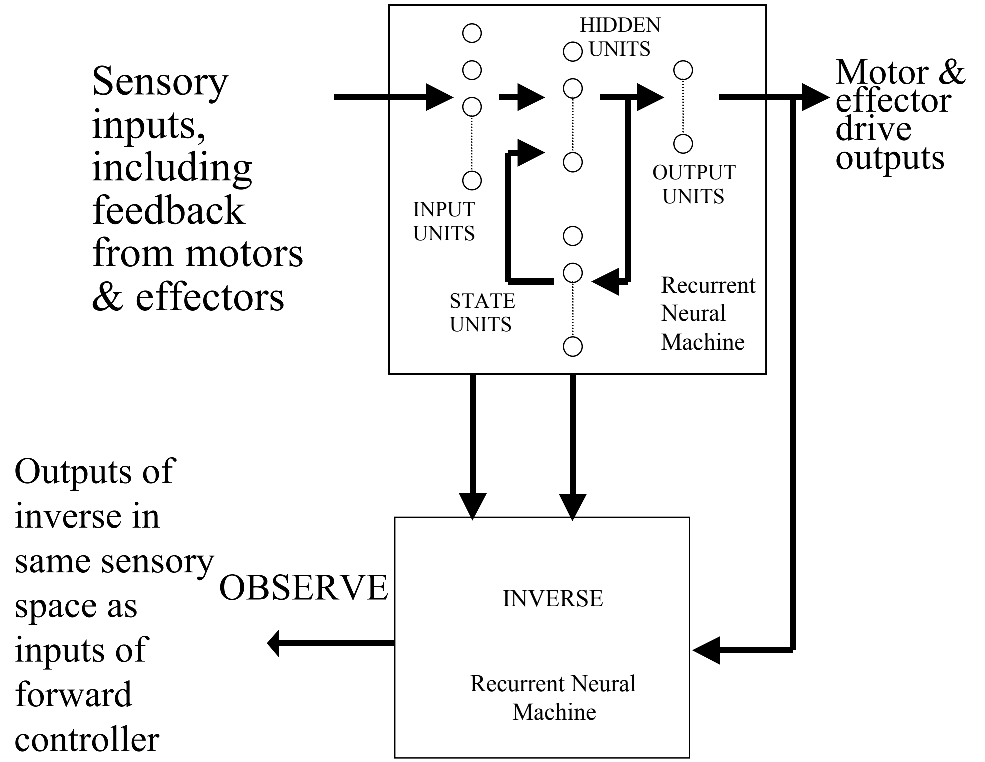
Generic Controller Architecture



- The controller of the robot is a neural network with recurrent feedback, capable of forming internal representations of sensory information in the form of a neural state machine.
- Sensory inputs (vision, sound, smell, etc) are fed into the controller, including *feedback* signals from the motors and effectors.
- Controller outputs drive the locomotion and manipulators of the robot.
- The neural controller learns to perform a task, using NN and GA techniques.
- Novel inputs that are unrecognized must be adaptively learned by the model.
- The model learns continuously over sequences of actions in time via reinforcement learning, supervised learning, or mimicing a human controller.
- The model continuously refines itself to improve its prediction accuracy.
- But - the internal model of the controller is *implicit* and therefore *hidden* from us.

Understanding the Controller

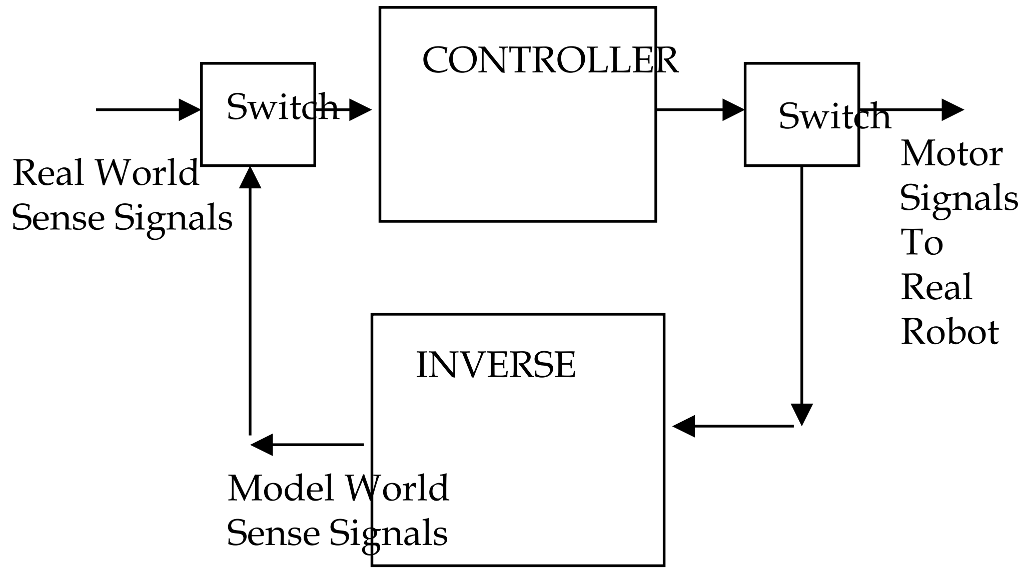
Introduce a second recurrent neural network, separate from the first system, which learns the *inverse* relationship between the internal activity of the controller and the sensory input space.



- This mechanism will allow us to represent the hidden internal state of the controller in terms of the sensory inputs that correspond to that state.
- Thus we may claim to know something of what the robot is *thinking*.
- We assume that the controller is learned first, and that, once this is learned and reasonably stable, the inverse can be learned.

Inverse-Predictor Controller

- We now allow the inverse to be fed back into the controller via the switch.
- The controller then has an image of its internal hidden state or <u>*self*</u> in the same feature space as its real sensory inputs.
- It can *see* what it <u>*itself*</u> is thinking.
- As before <u>*we*</u> can also observe what the machine is <u>*thinking*</u>.



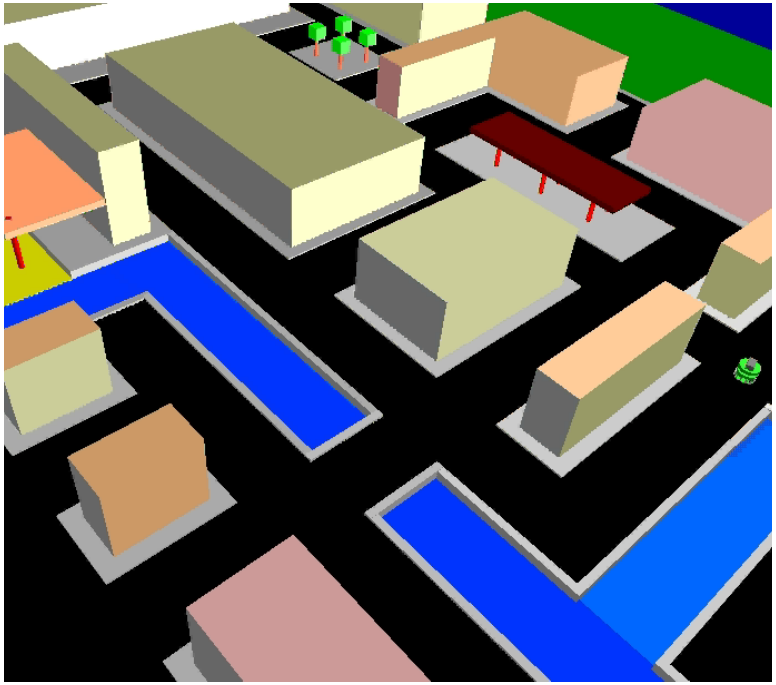
Normal Mode – controller produces motor signals, inverse detects mismatch or *novelty*.

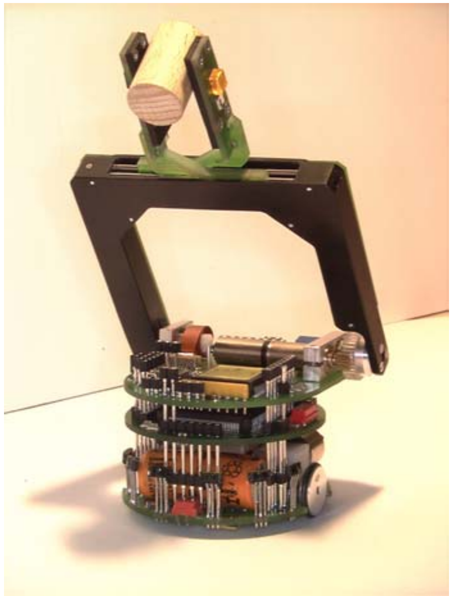
Thinking or Planning– inverse drives input, sequences of action to a goal can be manipulated *mentally*, and then switched on for action.

Sleeping – noise is input, producing imagined mental images or *dreams*. The noise vectors can be used to update (learn) the inverse.

An Experiment

Using the Khepera Robot & Webots – Khepera Embodied Simulator





- Simulator complexity is OK for a simple robot like the Khepera, but for more complex robots, the simulator may be too complex or not simulate the real word accurately.
- Simulators allow faster operation than real robots – particularly if learning involved.

A much simplified I-P controller

- Fixed static empty environment.
- Simple Robot: Khepera with 8 IR sensor signals plus 2 motor drive signals.
- Simple adaptive *unsupervised* VQ modeling system:
  - ✓ Learns model features *directly* in the input sensory space.
  - ✓ Hence no inverse to learn - the internal representation learned by the robot is directly visible as an input space vector.
  - ✓ We can directly *spy* on the internal model.
  - ✓ (based on Linaker and Niklasson 2000 ARAVQ algorithm).
- A 10-dimensional feature space is formed from the 8 Khepera IR sensor signals plus the 2 motor drive signals.
- Clusters novel feature-vectors, to form prototype feature vector *models*.
- Adds new models based on two criteria:
  - ✓ Novelty: Large distance from existing models.
  - ✓ Stability: Low variance in buffered history of features.
- *Continuously* learning new models and adapting existing models over time.

Learning in Action

- First we learn or program the forward model or robot controller:
- In this simple experiment we program in a simple reactive wall-following behavior, rather than learn a complex behavior.
- The robot starts with no internal model, and adaptively learns its internal representation in an unsupervised manner as it performs its wall following behavior.

•Colors show learned concepts:

Black – right wall

Blue – ahead wall

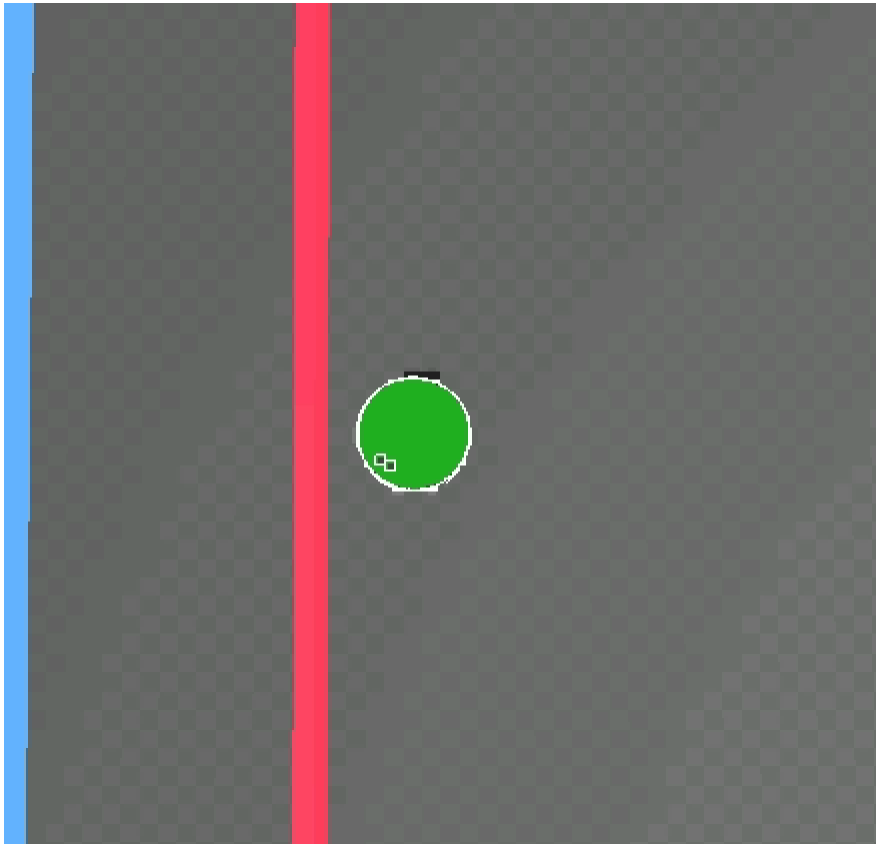
Green – 45 degree right wall

Red – corridor

Light Blue – outside corner

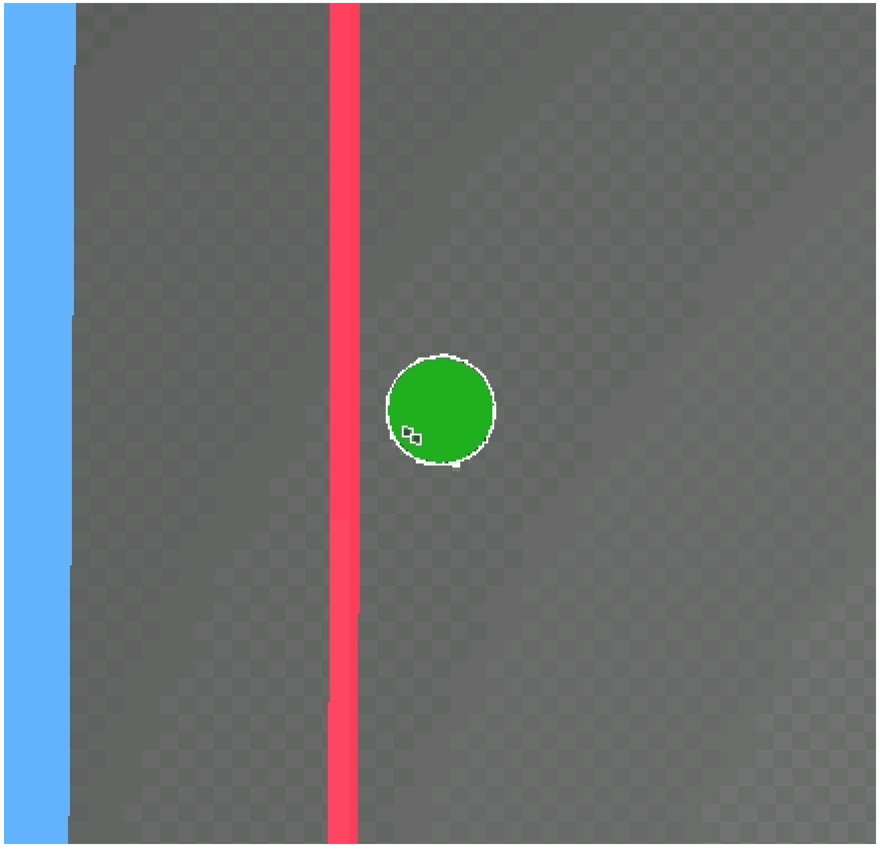
•Only changes in features are shown.

•The robot is continuously outputting a string of recognized or newly learned features.



Running with the model

- Switch off the wall follower.
- The robot *sees* features as it moves.
- Choose the closest learned model vector at each step.
- Use the model vector motor drive values to actually drive the motors.

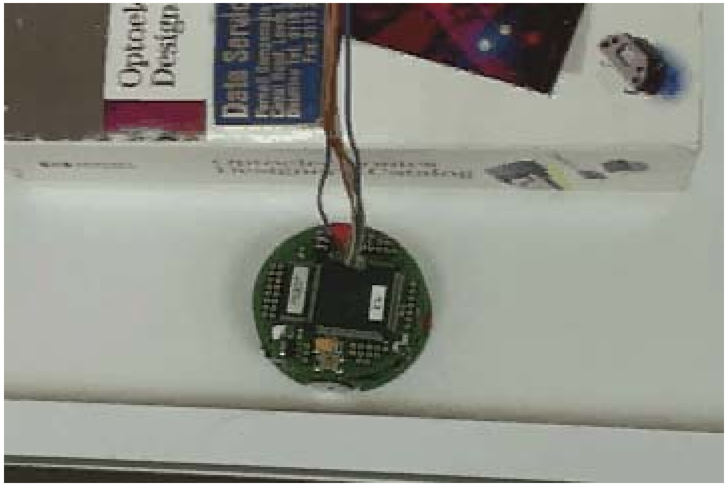


Color indicates which is the current "best" fit model feature.

Keeping it real

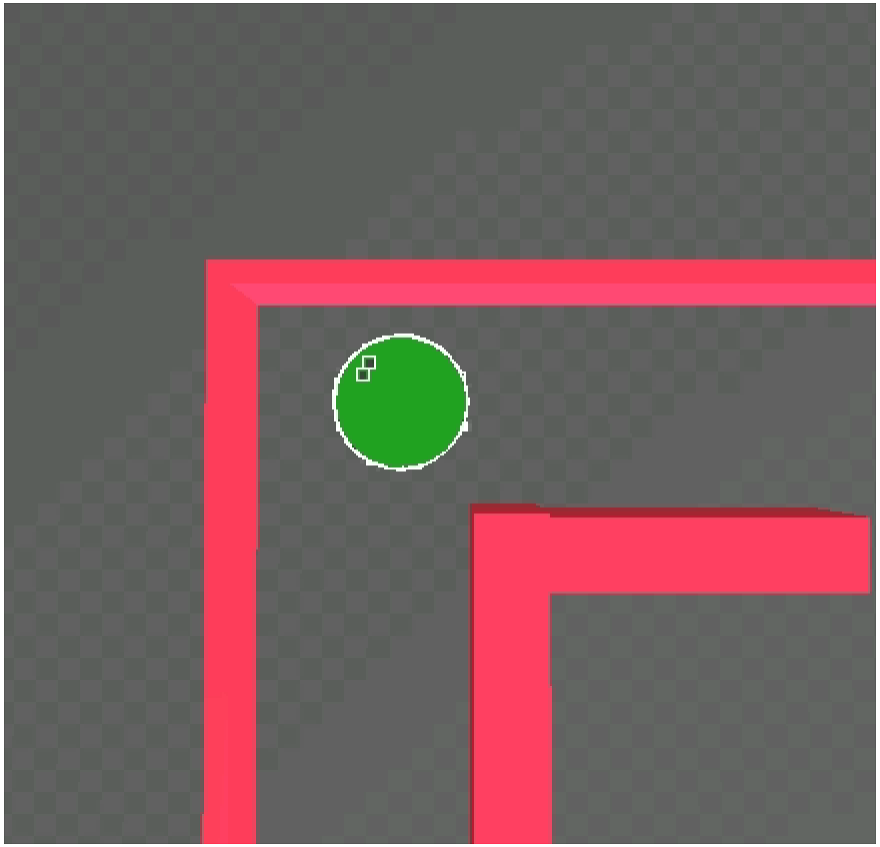
Run the model learned in Webots in the real robot.

Run the model learned in the real robot, in the real robot



Manipulating the model "mentally"

- Take the sequence of learned model feature vectors and cluster sub –sequences into higher-level concepts.
- For example:
  - Blue-Green-black = Left Corner
  - Red = Corridor
  - Black = right wall
- At any instant ask the robot to go to "home".
- Run the model forwards mentally to decide if it is shorter to go ahead or to go back.
- Signal appropriate action.



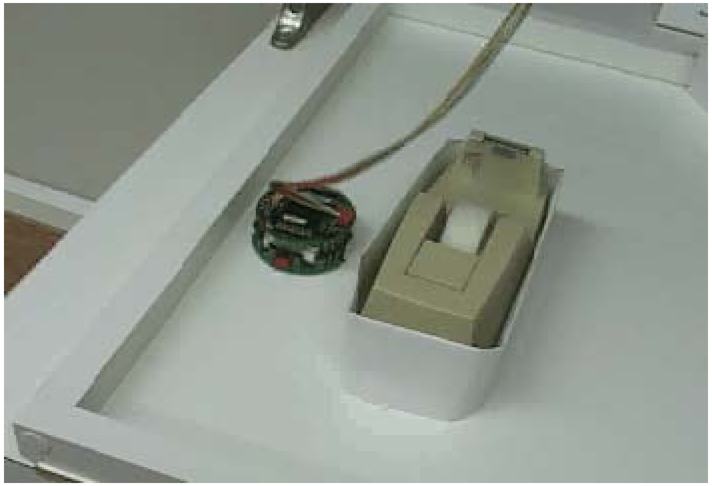
•Corridor corner is home.

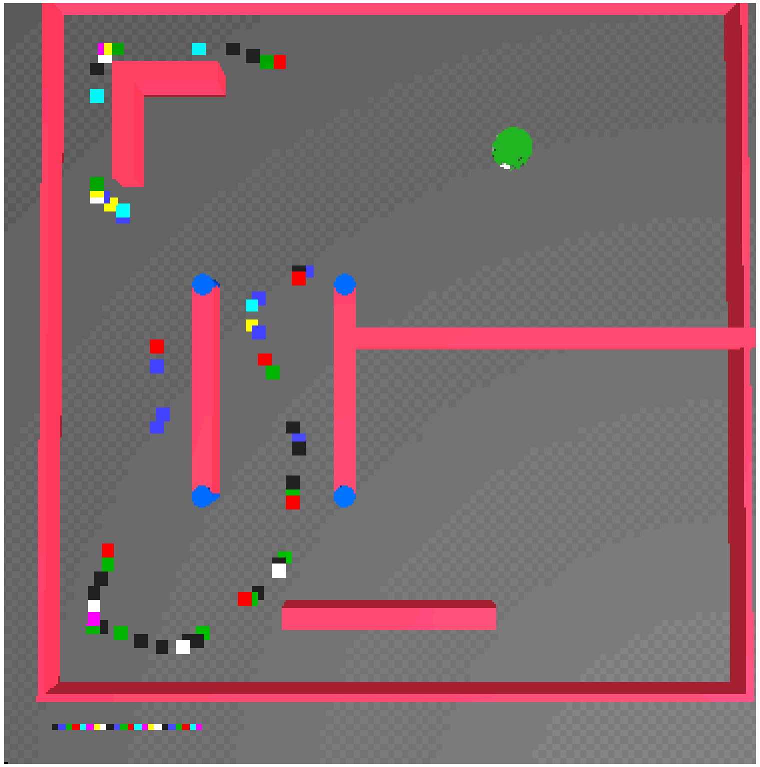
•Rotate = Home is behind me.

•Flash LED's = Home is ahead of me.

More Complex Controller & Environment

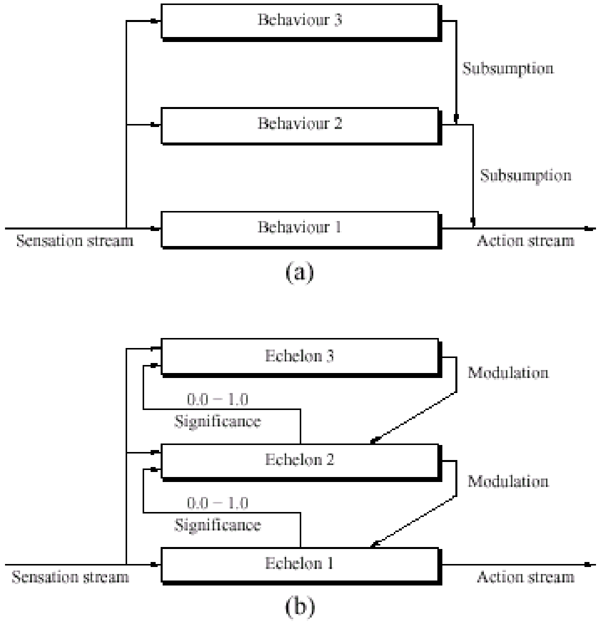
- Braitenberg Obstacle Avoider.
- Model learned from simulation.
- More (22) model features learned.
  - but complexity still very low for the more complex behavior & environment.

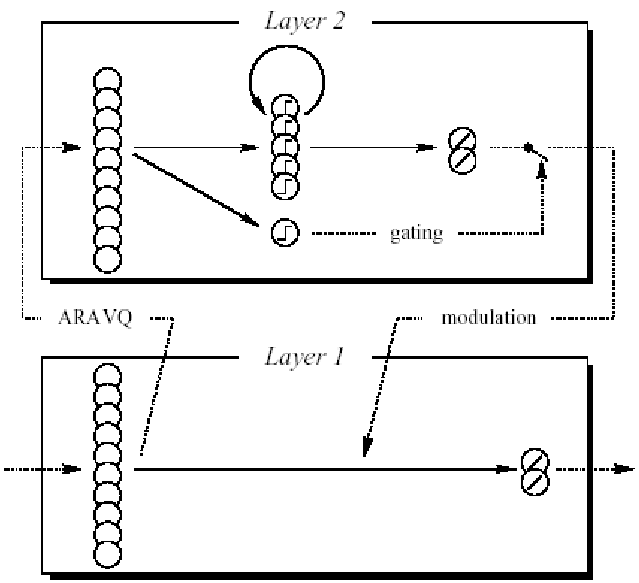


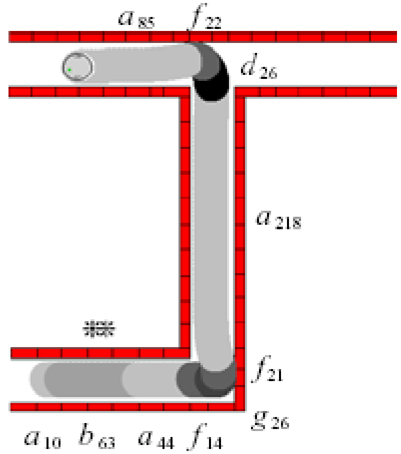


Higher level behavioral controllers

- *layer* controllers -higher and higher levels of abstraction (Linaker 2002)
- The lowest level operates at the ms timescale of sensors and actuator control. The highest levels operate at symbolic levels and much longer goal-driven timescales.
- Adjacent layers *modulate* the predictions of higher and lower layers, as opposed to subsumption (Brooks 1990).
- The controller is capable of solving much more difficult tasks such as delayed response tasks – e.g. the road sign problem.
- Learned using delayed reinforcement learning.

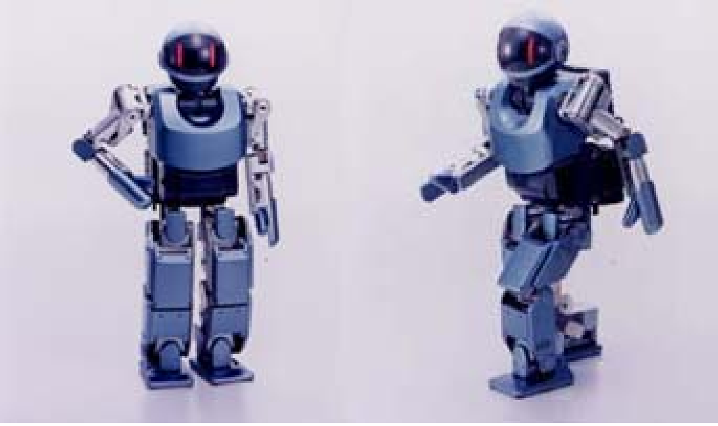






Challenge – Increase Complexity

- More complex robots
- More complex environments
- More complex architecture



Sony Dream Robot

Head:2 degrees of freedom
Body:2 degrees of freedom
Arms:4 degrees of freedom (x2)
Legs:6 degrees of freedom (x2)
(Total of 24 degrees of freedom)

Environment

Fixed environment
Moving objects
Movable objects
Objects with different values
Other agents – prey
Other agents – predators
Other agents – competitors
Other agents – collaborators
Other agents – mates
Etc

Agent

Movable body
More sensors
Effectors
Articulated body
Metabolic state
Acquired skills
Tools
Imitative learning
Language
Etc