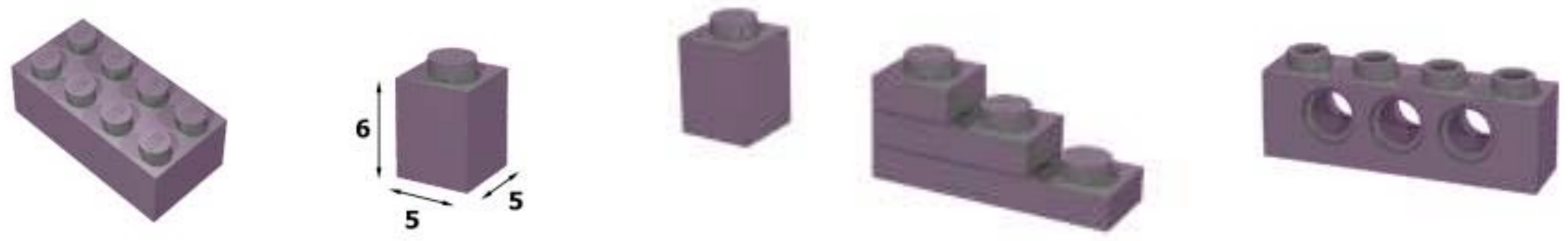
Learning Robots

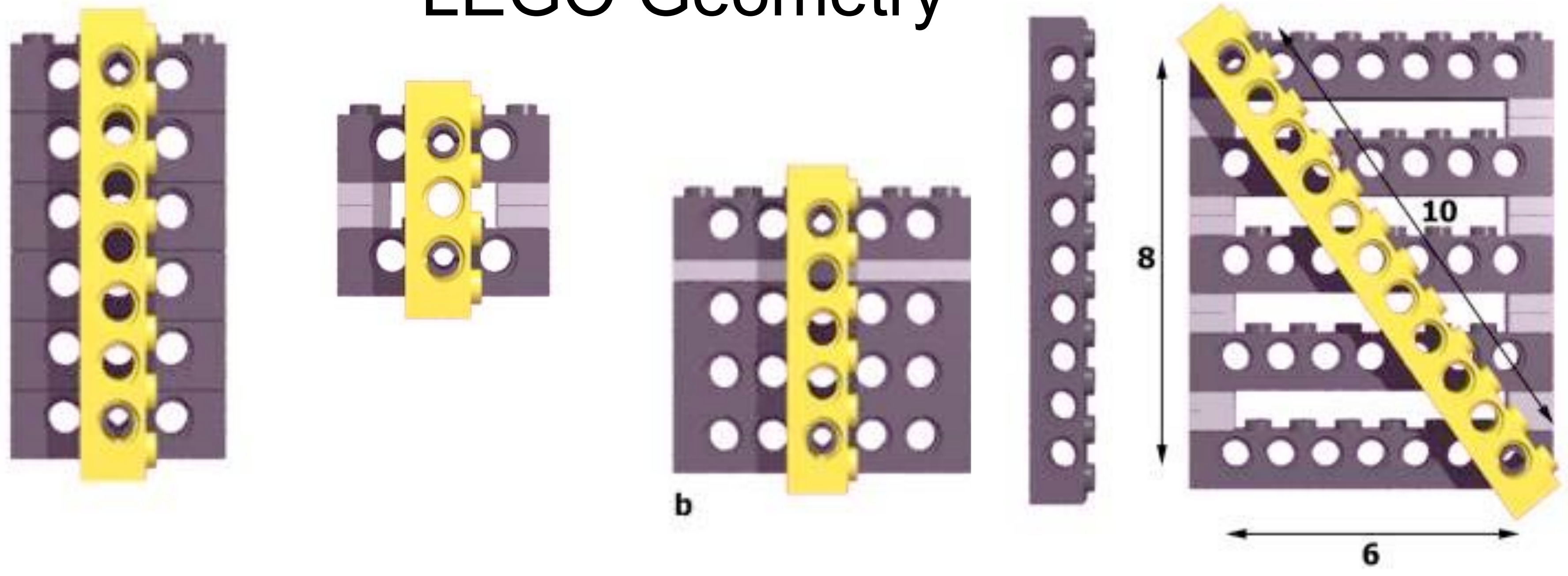
Pavel Petrovič
Department of Applied Informatics,
Faculty of Mathematics, Physics and Informatics
ppetrovic@acm.org

August 2009

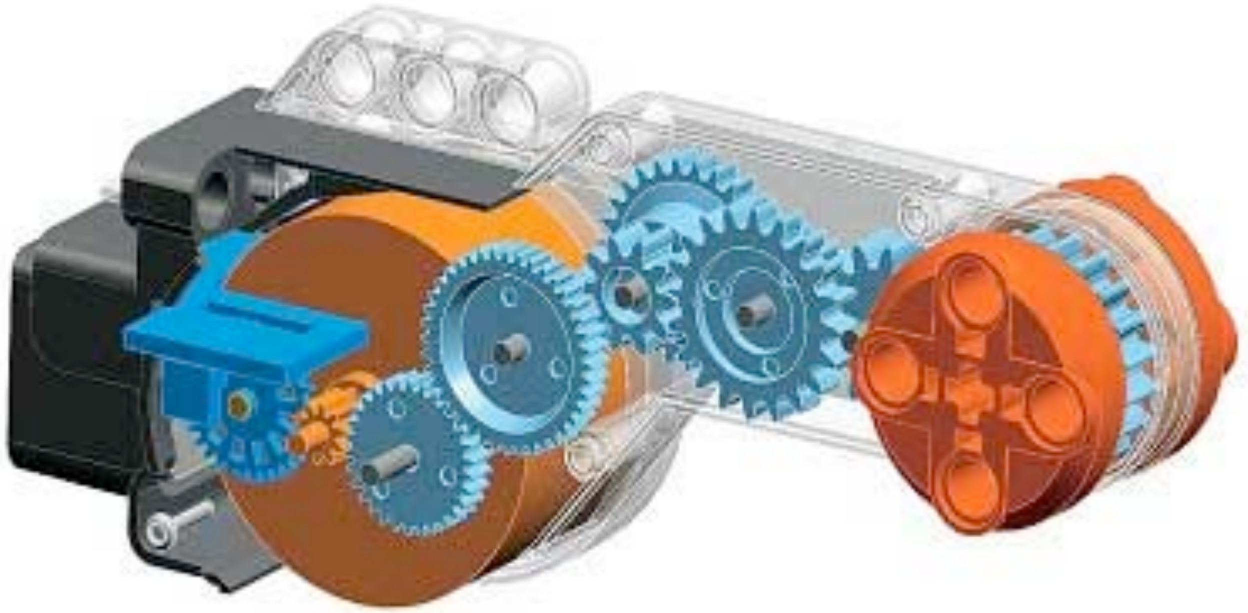
Life is learning... :-)



LEGO Geometry

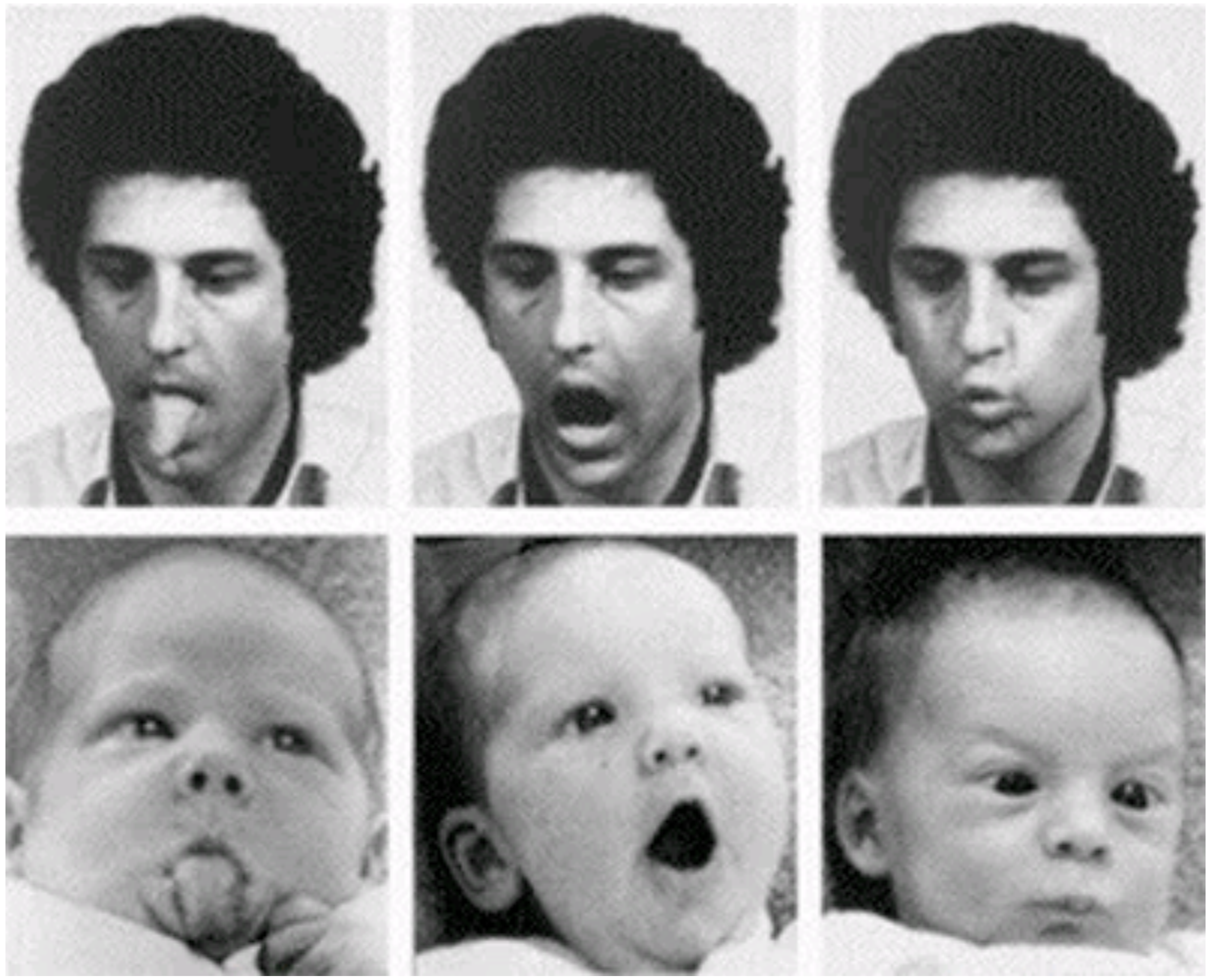


But how does the learning work inside?

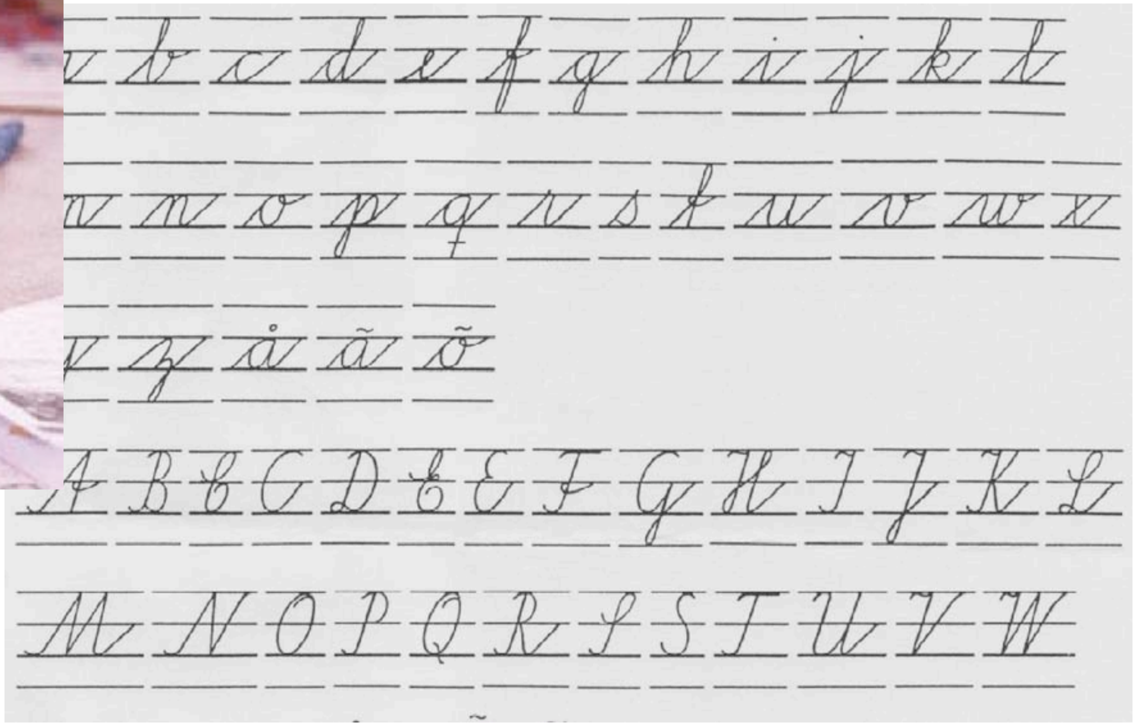


What is Learning?

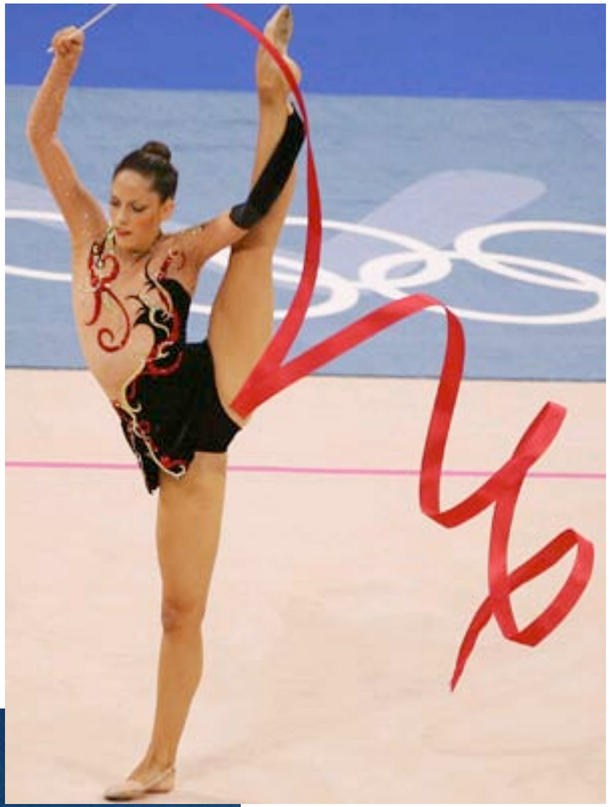
What is Learning?



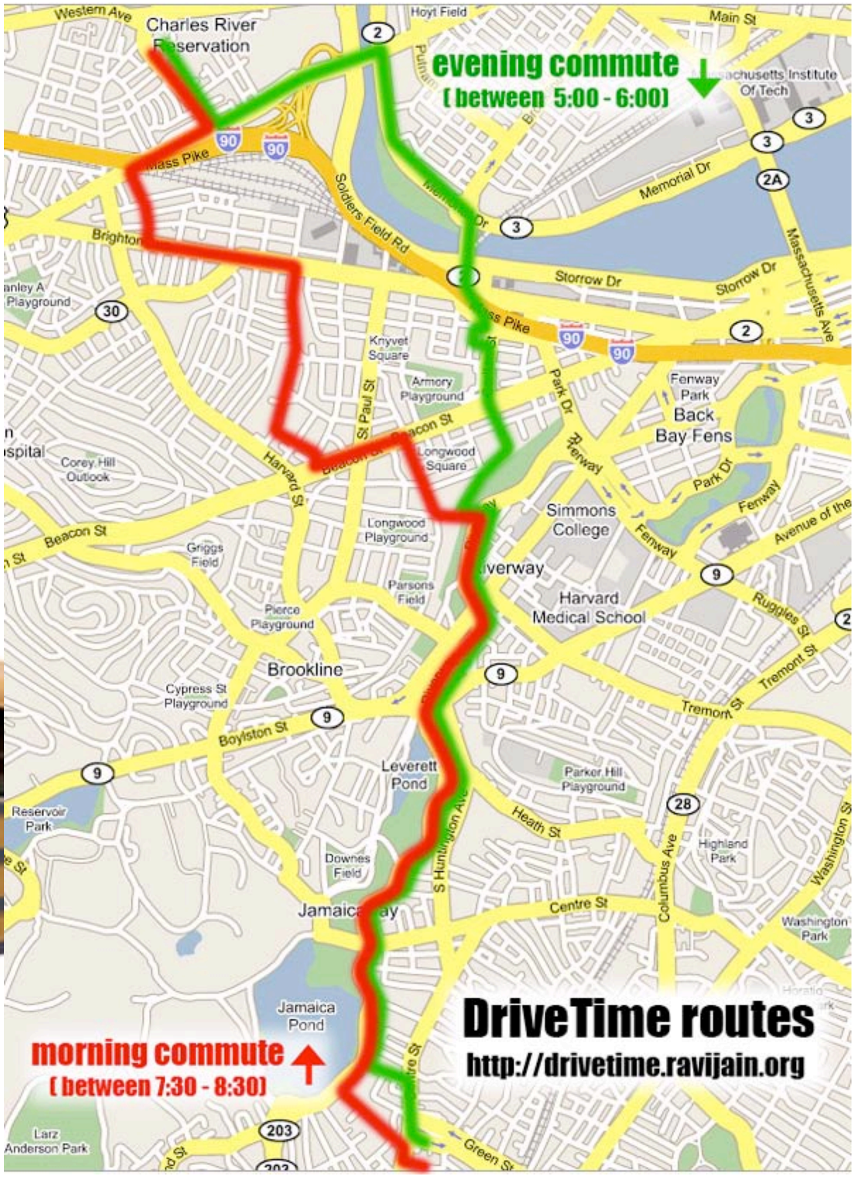
What is Learning?



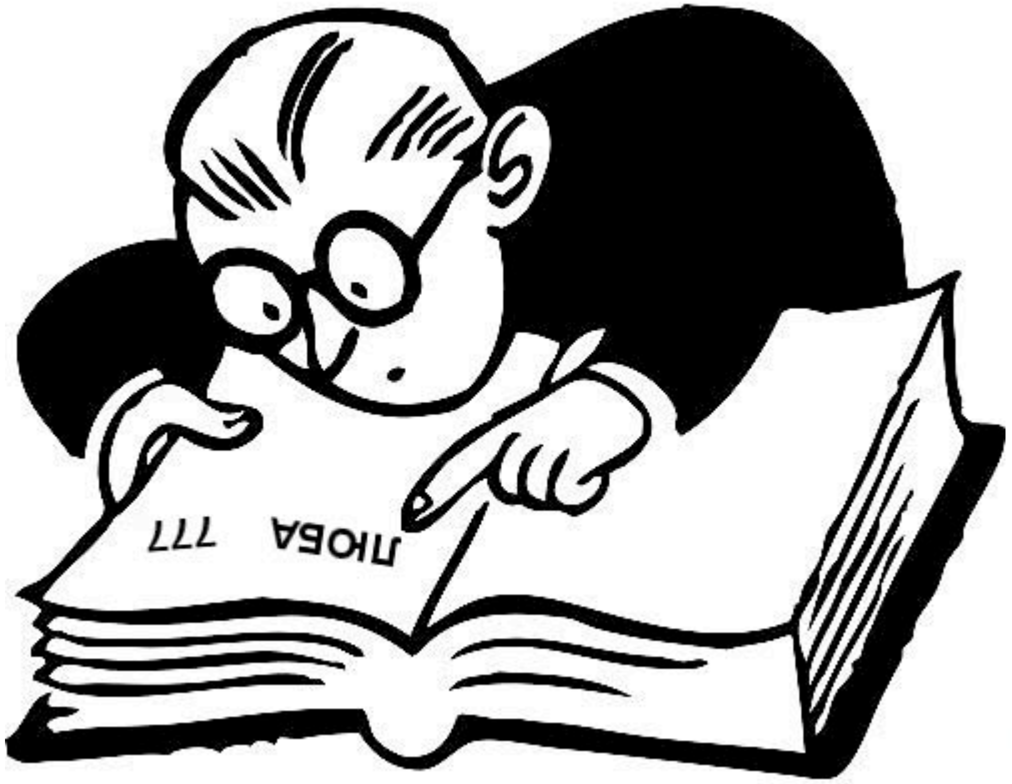
What is Learning?



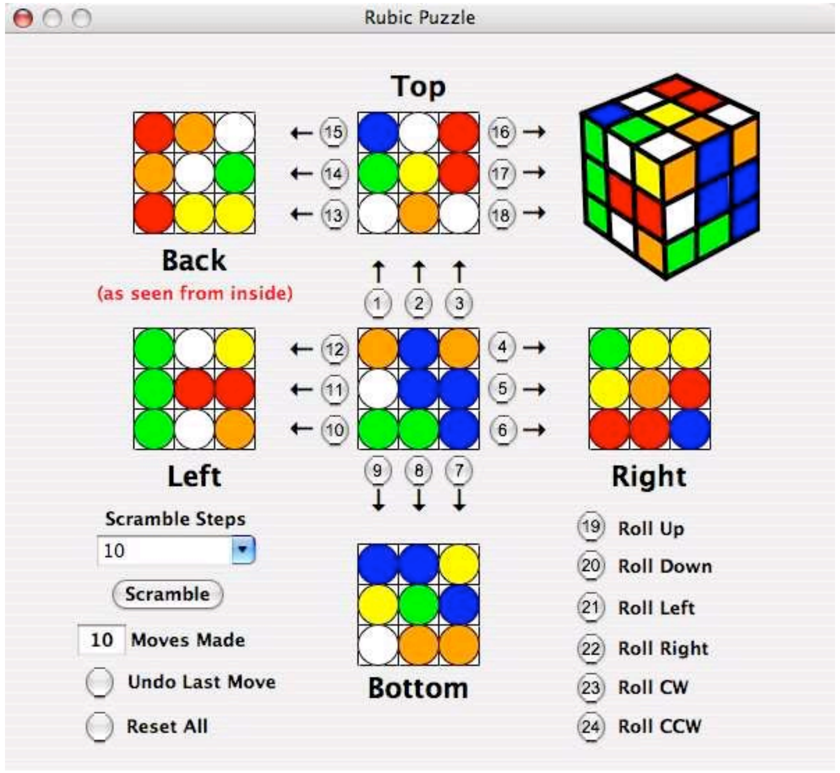
What is Learning?



What is Learning?



What is Learning?



What is Learning?

Webster: *learning*

1 : the act or experience of one that learns
2 : knowledge or skill acquired by instruction or study
3 : modification of a behavioral tendency by experience
(as exposure to conditioning)

Encyclopedia Britannica: *learning*

the alteration of behaviour as a result of individual experience. When an organism can perceive and change its behaviour, it is said to learn.

Wikipedia: *learning*

acquiring new knowledge, behaviors, skills, values, preferences or understanding, and may involve synthesizing different types of information. The ability to learn is possessed by humans, animals and some machines.

Learning x Adaptation?

Adaptation – „small“ learning, usually related to physics of the world

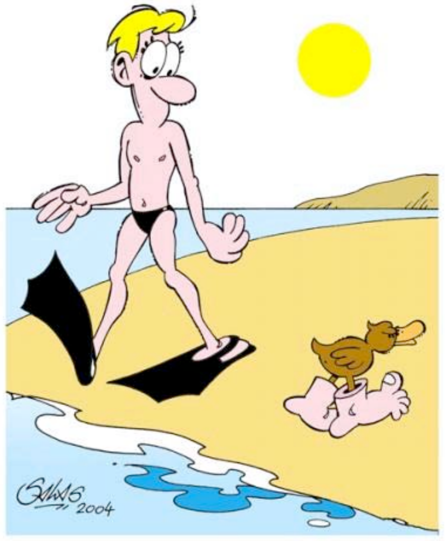
Adaptation of species

Adaptation – changing body shape, behavior, foraging, life style

Evolutionary adaptation

Learning of individuals

Learning usually relates to cognitive processes, physiological changes are only in the brain



Robot Learning

Why do robots need to learn?

Standard robots used in controlled factory conditions usually do not learn. They may adapt to different material properties, and be programmable – to perform different action sequences.

Robots that share the real environment with us can learn to perform tasks better.

Environment properties:

Unknown	= do not know what to expect ahead
Dynamic	= changes may occur
Unpredictable	= do not know when and how it changes

Robot Learning

What can the robots learn?

- Map of their environment
- Properties of their environment
- Recognize objects, faces, people
- Manipulation tasks
- Navigational tasks
- Coordinate and cooperate with other robots
- Effective communication with humans
- Understand situations and take apropriate actions
- Complex tasks

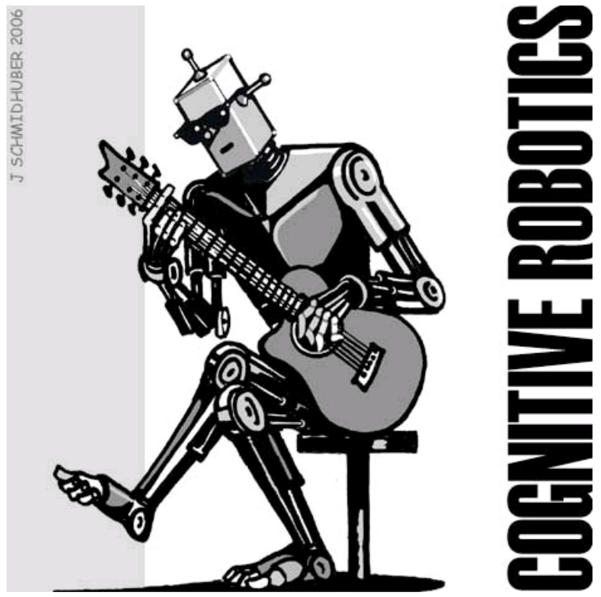
Robot Learning

How can the robots learn?

Pattern Recognition & **Machine Learning**

(in general: Artificial Intelligence)

Let's take a closer look...



Machine Learning – simple example

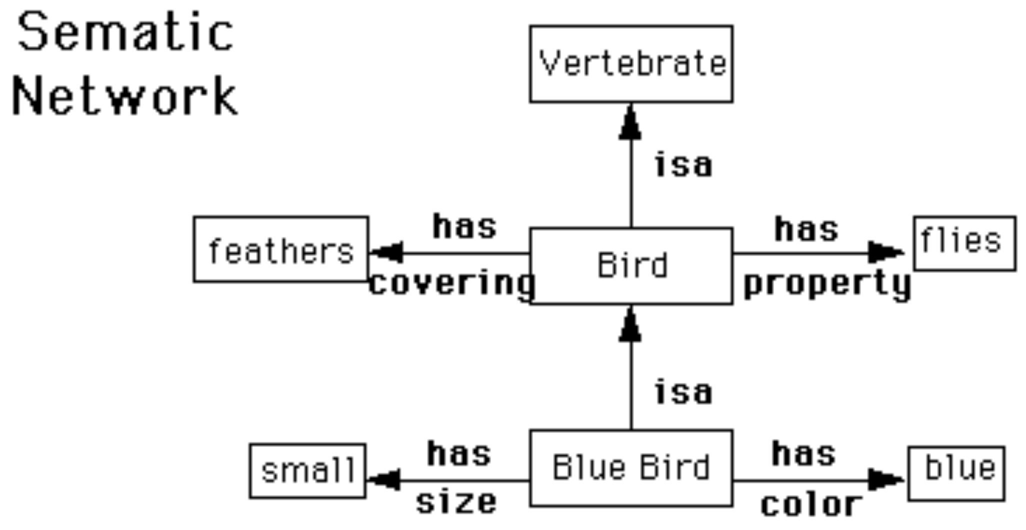
Animal game

Computer	Human
Is it a mammal?	yes
Does it live in water?	no
Is it a carnivore?	yes
Does it have stripes?	yes
Is it a tiger?	yes
I won!	

Computer	Human
Is it a mammal?	yes
Does it live in water?	yes
Is it a whale?	no
I give up. What is it?	dolphin
Please enter a question distinguishing between a whale and a dolphin:	Is it very large?
For a dolphin the answer to this question is:	no

ML: Knowledge representation + learning rule/algorithm

Knowledge Representation - Symbolic



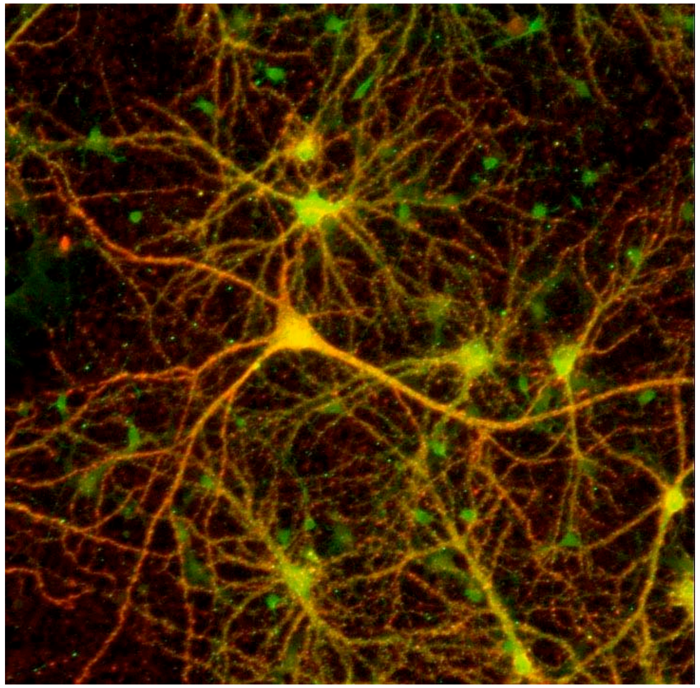
Knowledge representation: LISP expressions
Learning algorithm: predicate logic

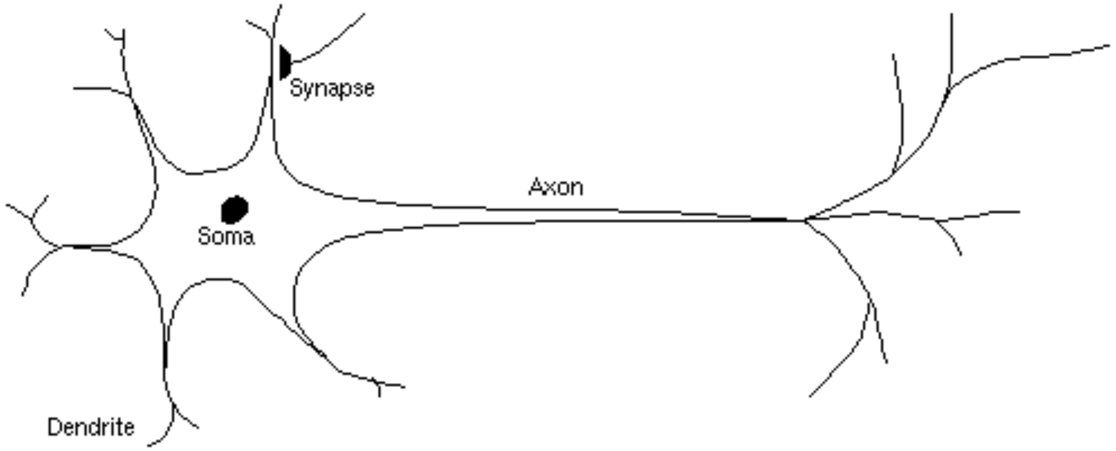
Knowledge Representation - Symbolic

GENERAL TRIANGLE {
(Generalized_by: closed planar geometric object)
(Generalization_of: acute-angled triangle, obtuse-angled triangle, right-angled triangle,
equilateral triangle, isoscales-triangle)
(parameters: x-side, y-side, z-side, φ-angle, χ-angle, ψ-angle, x-altitude, y-altitude, z-altitude,
x-median, y-median, z-median, r_inner_circle_radius, R_outer_circle_
radius, P_perimeter=x+y+z, V_volume=(x*x-altitude)/2)
(number of sides *[<cardinality:1> <data type:INT>]* value: 3)
(number of angles *[<cardinality:1> <data type:INT>]* value: 3)
(x-side *[<cardinality:1> <data type:REAL> <if-needed: ask, measure, infer> <ifchanged:
check consistency (x<y+z)>]* length value: UNKNOWN)
(y-side, z-side similarly)
(φ-angle *[<cardinality:1> <data_type:REAL > <data_template: .**, 0< φ<180 >
<if-needed: ask, measure, infer><if-changed: check_consistency (φ+χ +ψ=180)>]*
value: UNKNOWN)
(χ-angle, ψ-angle similarly)
(x-altitude *[<cardinality:1> <data_type:REAL > <data_template: .**> <if-needed:
ask, measure, infer> <if-changed: check_consistency)>]* value: UNKNOWN)
(y-altitude, z-altitude similarly)
(x-meridian *[<cardinality:1> <data_type:REAL > <data_template: .**> <if-needed:
ask, measure, infer> <if-changed: check_consistency)>]* value: UNKNOWN)
(y-meridian, z-median similarly)
(P_perimeter *[<cardinality:1> <data_type:REAL > <if-needed: ask, measure, infer _by:
P = x+y+z>]* value: UNKNOWN)
(V_volume *[<cardinality:1> <data_type: REAL> <data_template: .**>
<if-needed: ask, infer>]* value: √ [(p/2)(p-x)(p-y)(p-z)]|(x*x-altitude)/2) }

Knowledge Representation: Sub-symbolic

Nature's way:

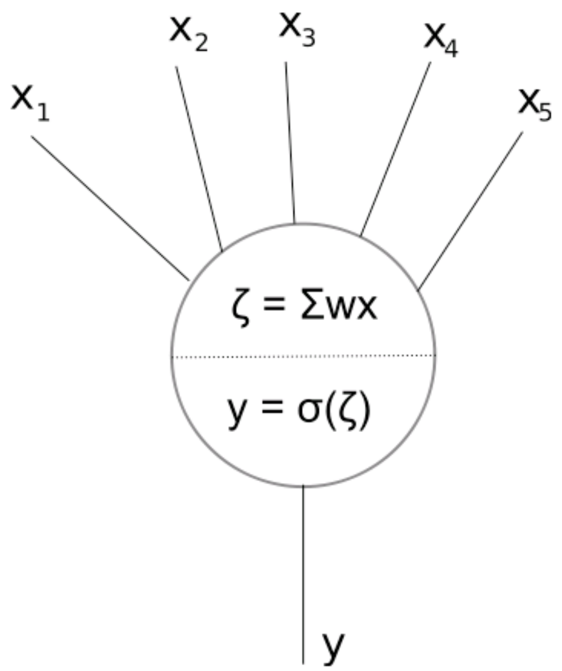




*Information is **distributed**, represented by millions of numerical values that serve multiple purpose/meanings...*

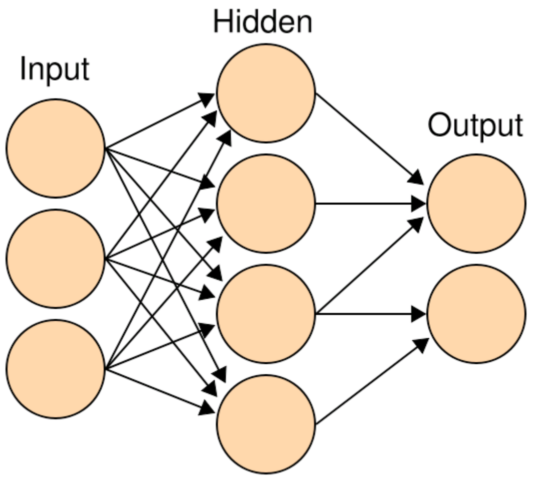
Knowledge Representation: Sub-symbolic

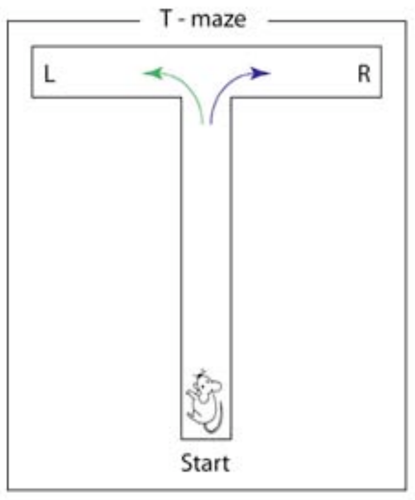
Connectionists: Artificial Neural Network (ANN) can represent the knowledge, can learn, do reasoning, generate actions



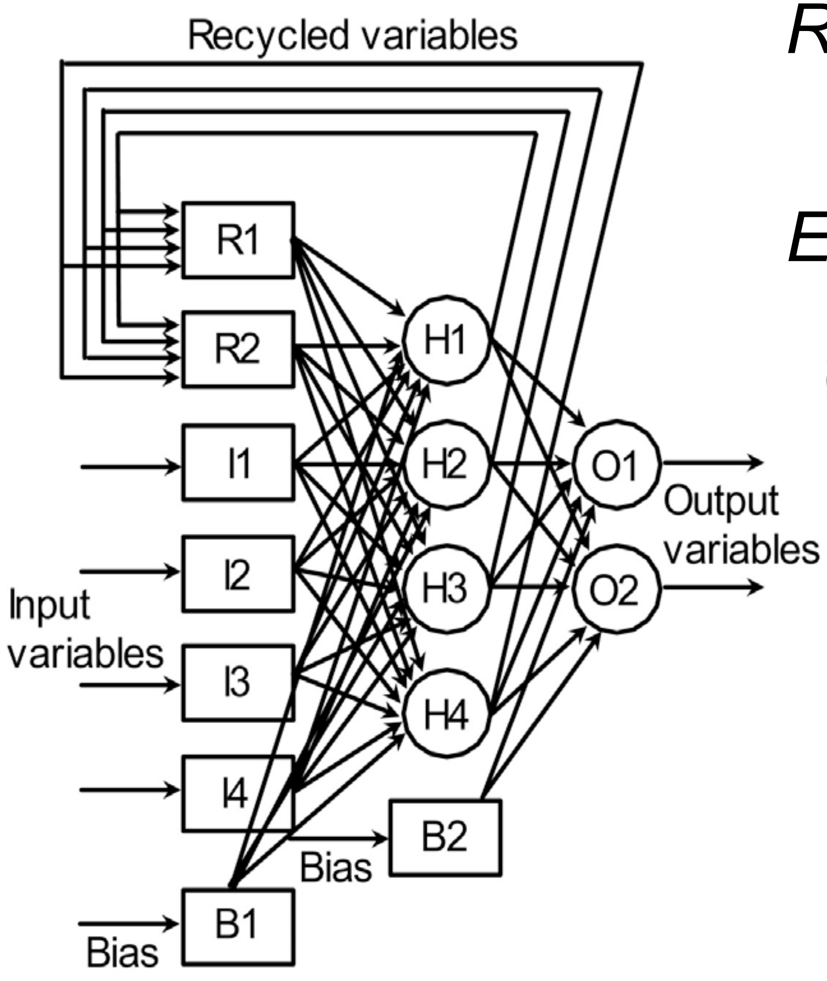
In robotics: Sensory-motor systems

Reactive systems vs. Internal state



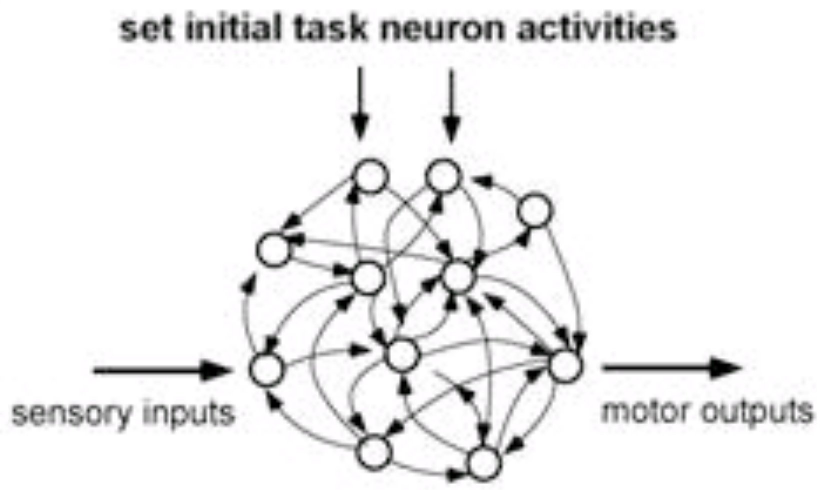


Knowledge Representation: Sub-symbolic



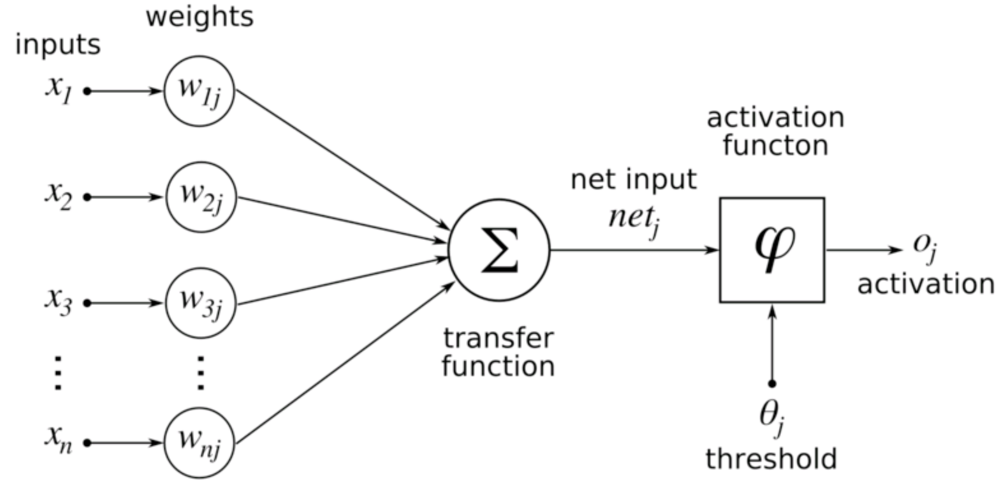
RNNs can compute any computable function

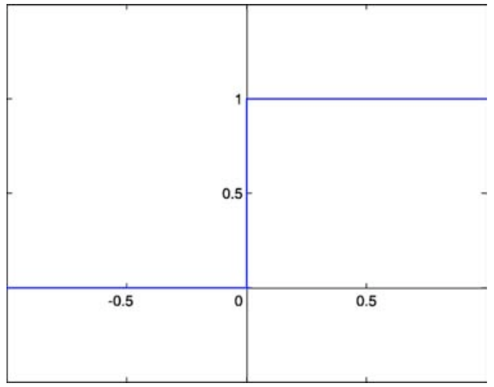
Elman-type or fully connected



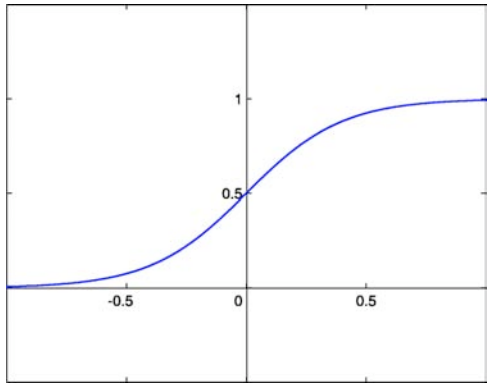
Knowledge Representation: Sub-symbolic

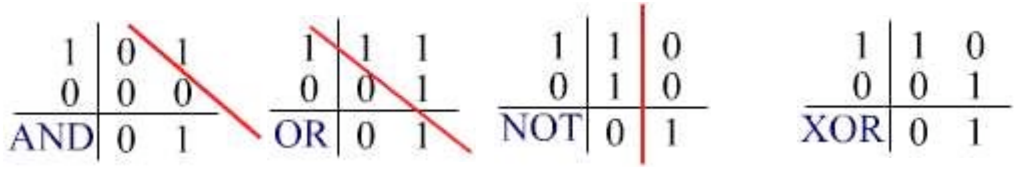
What can a simple perceptron represent?





$$\sigma(t) = \frac{1}{1 + e^{-\beta t}}$$





1	0	1
0	0	0
AND	0	1

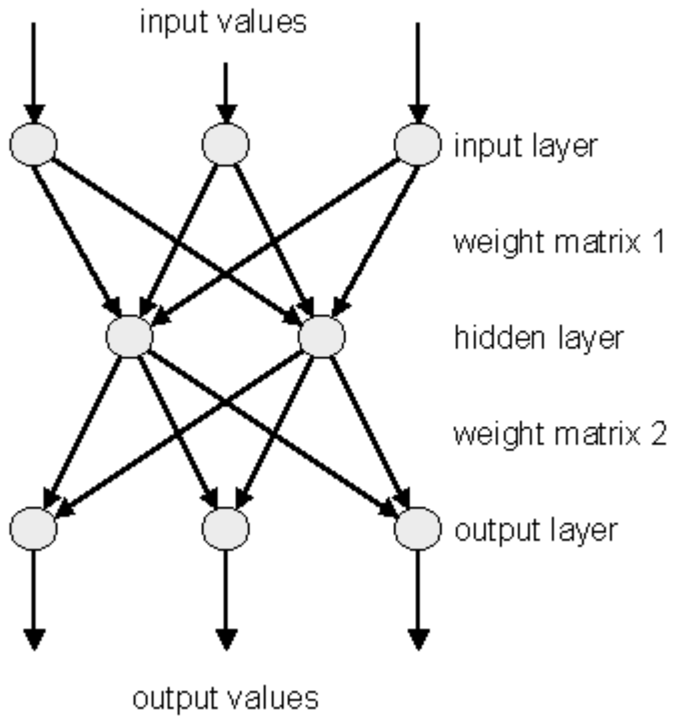
1	1	1
0	0	1
OR	0	1

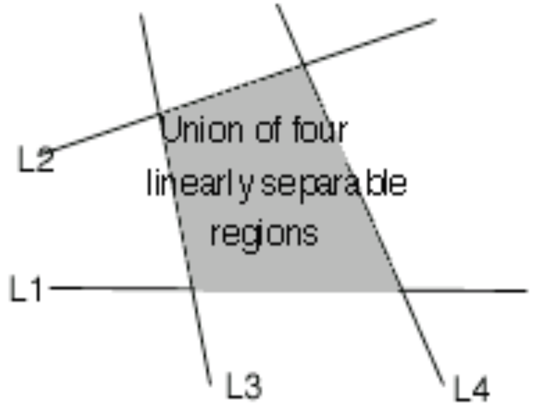
1	1	0
0	1	0
NOT	0	1

1	1	0
0	0	1
XOR	0	1

Knowledge Representation: Sub-symbolic

Solution – multilayer perceptron





classification

Knowledge Representation: Sub-symbolic

How to learn?
Example: Backpropagation algorithm

1. Network propagates inputs forward in the usual way, i.e.

- All outputs are computed using sigmoid thresholding of the inner product of the corresponding weight and input vectors.
- All outputs at stage n are connected to all the inputs at stage n+1

2. Propagates the errors backwards by apportioning them to each unit according to the amount of this error the unit is responsible for.

Knowledge Representation: Sub-symbolic

$\vec{x_j}$ = input vector for unit *j* (x_{ji} = *i* th input to the *j* th unit)
$\vec{w_j}$ = weight vector for unit *j* (w_{ji} = weight on x_{ji})
$z_j = \vec{w_j} \cdot \vec{x_j}$ = the weighted sum of inputs for unit *j*
o_j = output of unit *j*
t_j = target for unit *j*

We want to calculate $\frac{\partial E}{\partial w_{ji}}$ for each input weight w_{ji} for each output unit *j*. Note first that since z_j is a function of w_{ji} regardless of where in the network unit *j* is located

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} \\ &= \frac{\partial E}{\partial z_j} x_{ji} \end{aligned}$$

$$\frac{\partial E}{\partial z_j} = \delta_j$$

Knowledge Representation: Sub-symbolic

Output units:

$$E = \frac{1}{2} \sum_{k \in Outputs} (t_k - \sigma(z_k))^2$$

$$\sigma(t) = \frac{1}{1 + e^{-\beta t}}$$

$$\begin{aligned}
\delta_j = \frac{\partial E}{\partial z_j} &= \frac{\partial}{\partial z_j} \frac{1}{2} (t_j - o_j))^2 \\
&= -(t_j - o_j) \frac{\partial o_j}{\partial z_j} \\
&= -(t_j - o_j) \frac{\partial}{\partial z_j} \sigma(z_j) \\
&= -(t_j - o_j)(1 - \sigma(z_j))\sigma(z_j) \\
&= -(t_j - o_j)(1 - o_j)o_j
\end{aligned}$$

Weight update rule:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_i j} = \eta \delta_j x_{ji}$$

Knowledge Representation: Sub-symbolic

Hidden units:

$$\begin{aligned}\frac{\partial E}{\partial w_{ji}} &= \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} \\ &= \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \cdot x_{ji}\end{aligned}$$

$$\begin{aligned}\delta_j &= \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \\ &= \sum_{k \in Downstream(j)} \delta_k w_{kj} o_j (1 - o_j)\end{aligned}$$

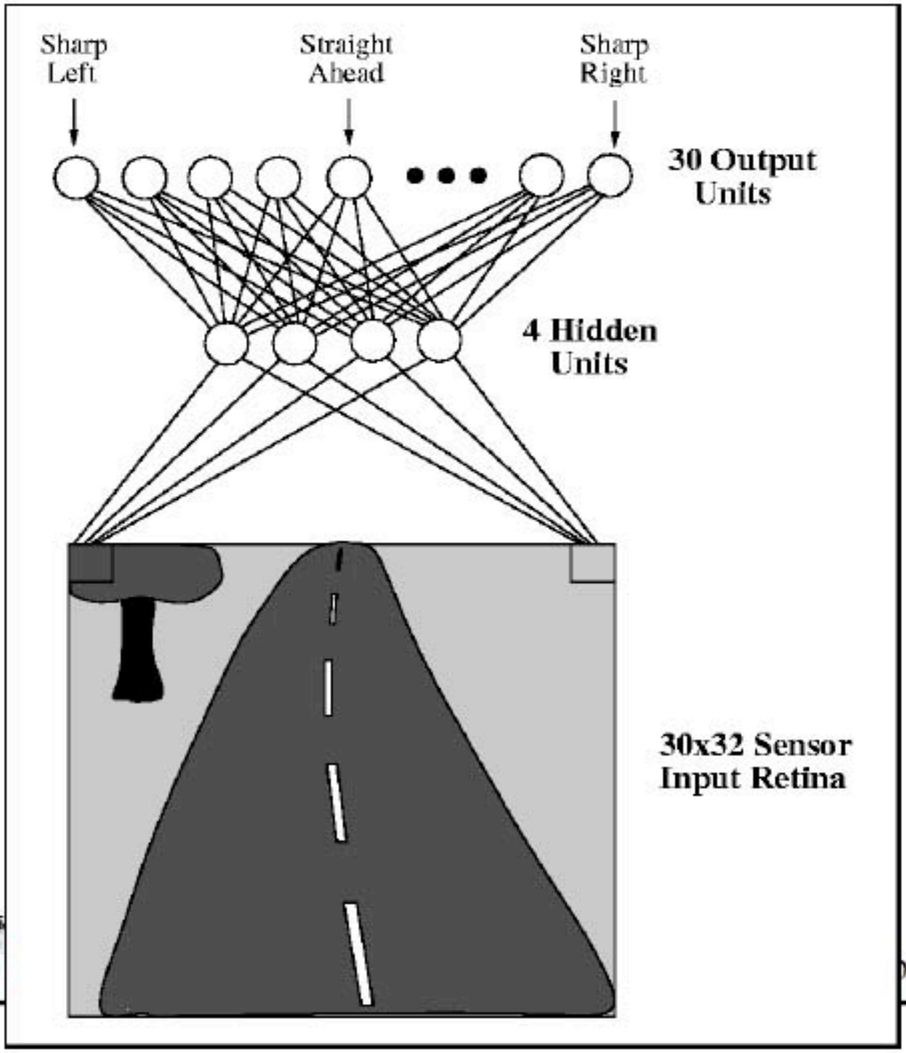
$$\delta_j = o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}$$

ALVINN: Autonomous Land Vehicle In a Neural Network

- Dean Pomerleau's Ph.D. thesis (1992).
- How ALVINN Works
  - Architecture
  - Training Procedure
  - Performance
- Why ALVINN Works
  - Hidden Unit Analysis
- Integrating Multiple Networks
- Other Applications

Interactive Systems Labs

ALVINN Network Architecture

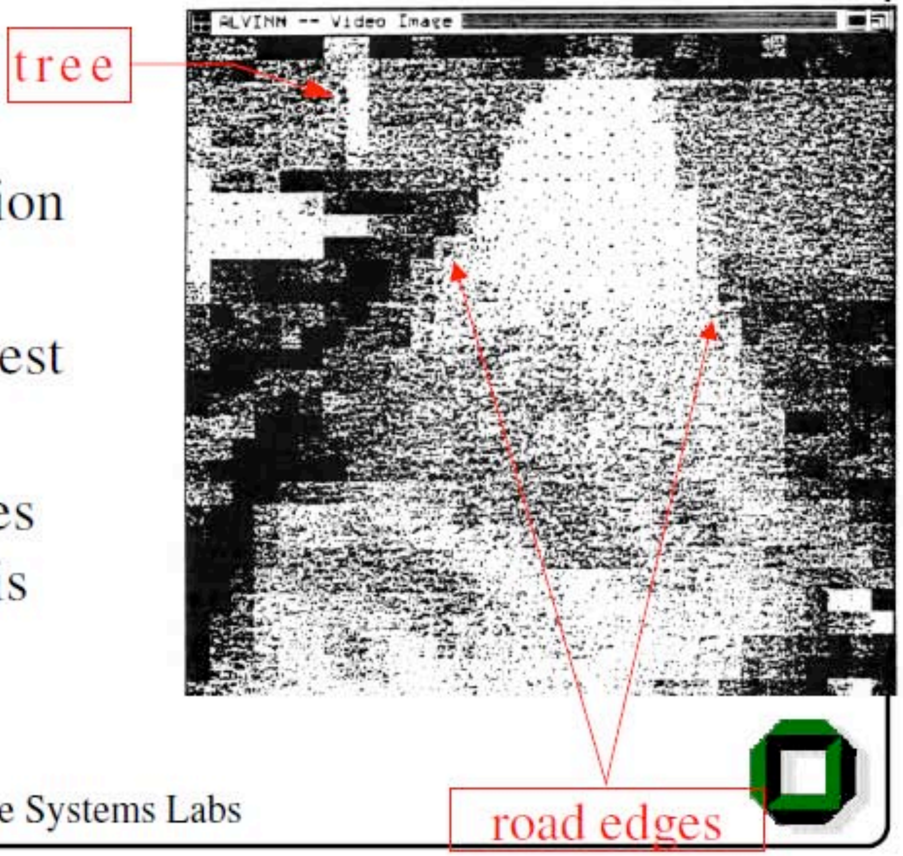


How many inputs?
$30 \times 32 = 960$

How many weights?
$961 \times 4 + 5 \times 30 = 3994$

Original Training Scheme

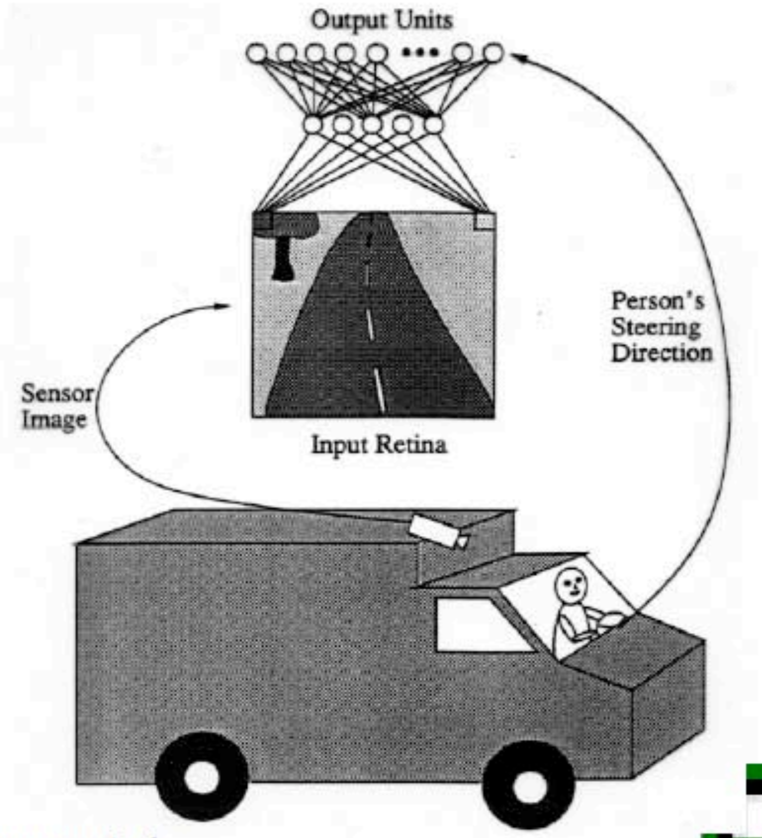
- Generate artificial road images mimicing situations the network is expected to encounter, including noise.
- Calculate correct steering direction for each image.
- Train on artificial images, then test on real roads.
- Problem: realistic training images are difficult to produce: training is expensive.



Interactive Systems Labs

Training on the Fly

- Digitize the steering wheel position.
- Train the network by having it observe live sensor data as a human drives the vehicle.
- The human "teaches" the network how to drive.
- Can this really work?
    - It's not so simple...



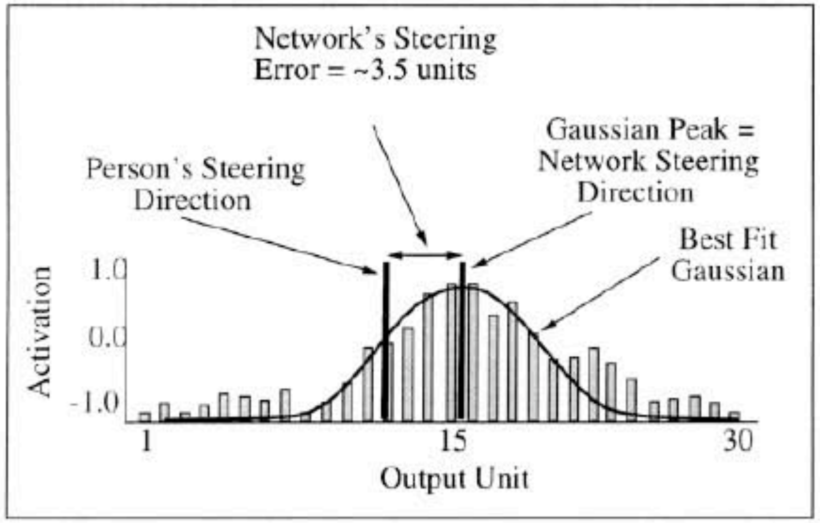
Interactive Systems Labs

Measuring Steering Error

•Train with a Gaussian bump centered over the desired steering direction.

•To test: fit a Gaussian to the network's output vector.

•Measure distance between Gaussian's peak and human steering direction.



Why use a Gaussian for the output pattern?

Interactive Systems Labs

Learning to Correct Steering Errors

- If the human drives perfectly, the network never learns to make corrections when it drifts off the desired track.
- Crude solution:
  - –Turn learning off temporarily, and drive off course.
  - –Turn learning back on, and let the network observe the human making the necessary corrections.
  - – Repeat.

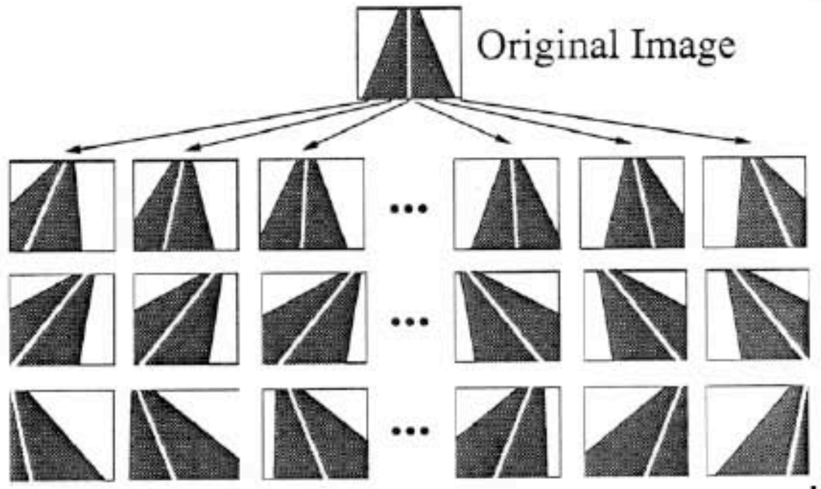
- Relies on the human driver to generate a rich set of steering errors: time consuming and unreliable.

Can be dangerous if training in traffic.

Interactive Systems Labs

Simulating the Steering Errors

- Let humans drive as best they can.
- Increase training set variety by *artificially* shifting and rotating the video images, so that the vehicle appears at different orientations relative to the road.



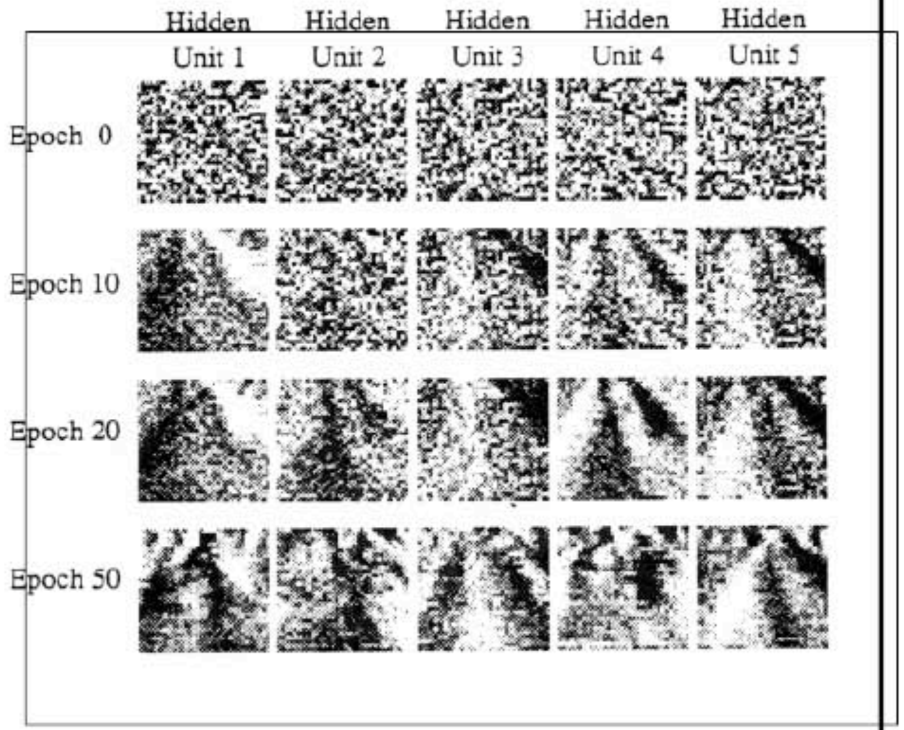
- Generate 14 random shift/rotations for each image.
- A simple steering model is used to predict how a human driver would react to each transformation.

Interactive Systems Labs

Network Weights Evolving

- •Initial random weights look like "salt and pepper" noise.

- •During training, the hidden units evolve into a set of complementary feature detectors.

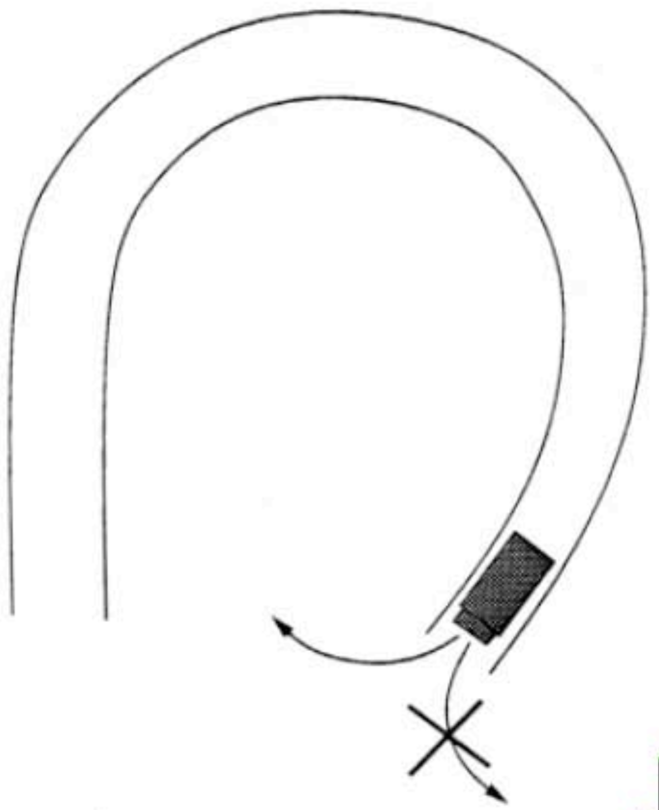


Interactive Systems Labs

Problem with Online Learning: Network Can "Forget"

- The network tends to overlearn recently encountered examples and forget how to drive in situations encountered earlier in training.

- After a long right turn, the network will be biased toward turning right, since recent training data focused on right turns.



Interactive Systems Labs

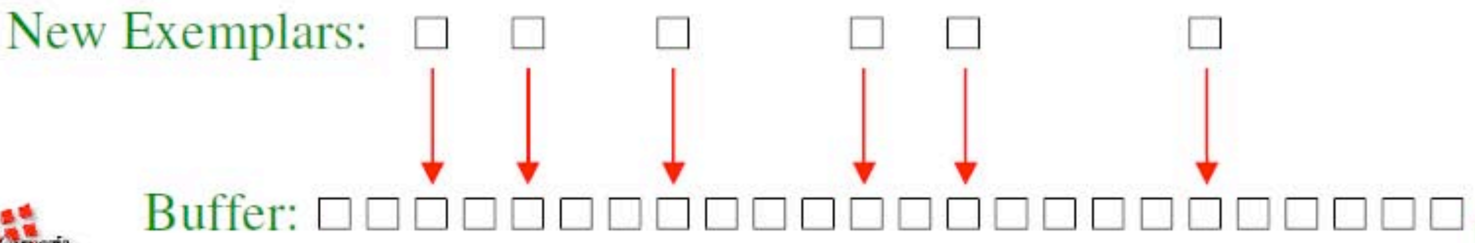
Solution: Maintain a Buffer of Balanced Training Images

This is a semi-batch learning approach. Keep a buffer of 200 training images.

Replace 15 old exemplars with new ones derived from the current camera image. Replacement strategies:

(1) Replace the image with the lowest error

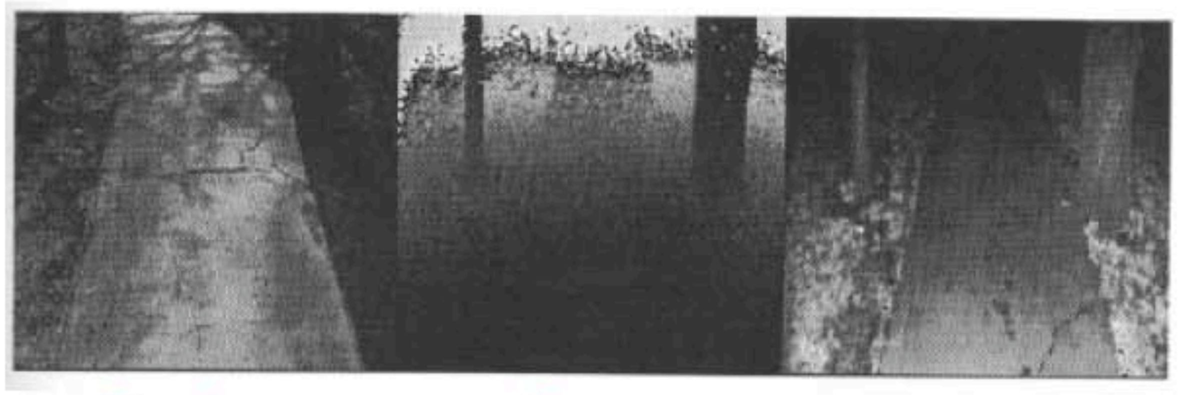
(2) Replace the image with the closest steering direction



Interactive Systems Labs

Multi-Modal Inputs

•ALVINN can avoid obstacles using a laser rangefinder. It can drive at night using laser reflectance imaging.

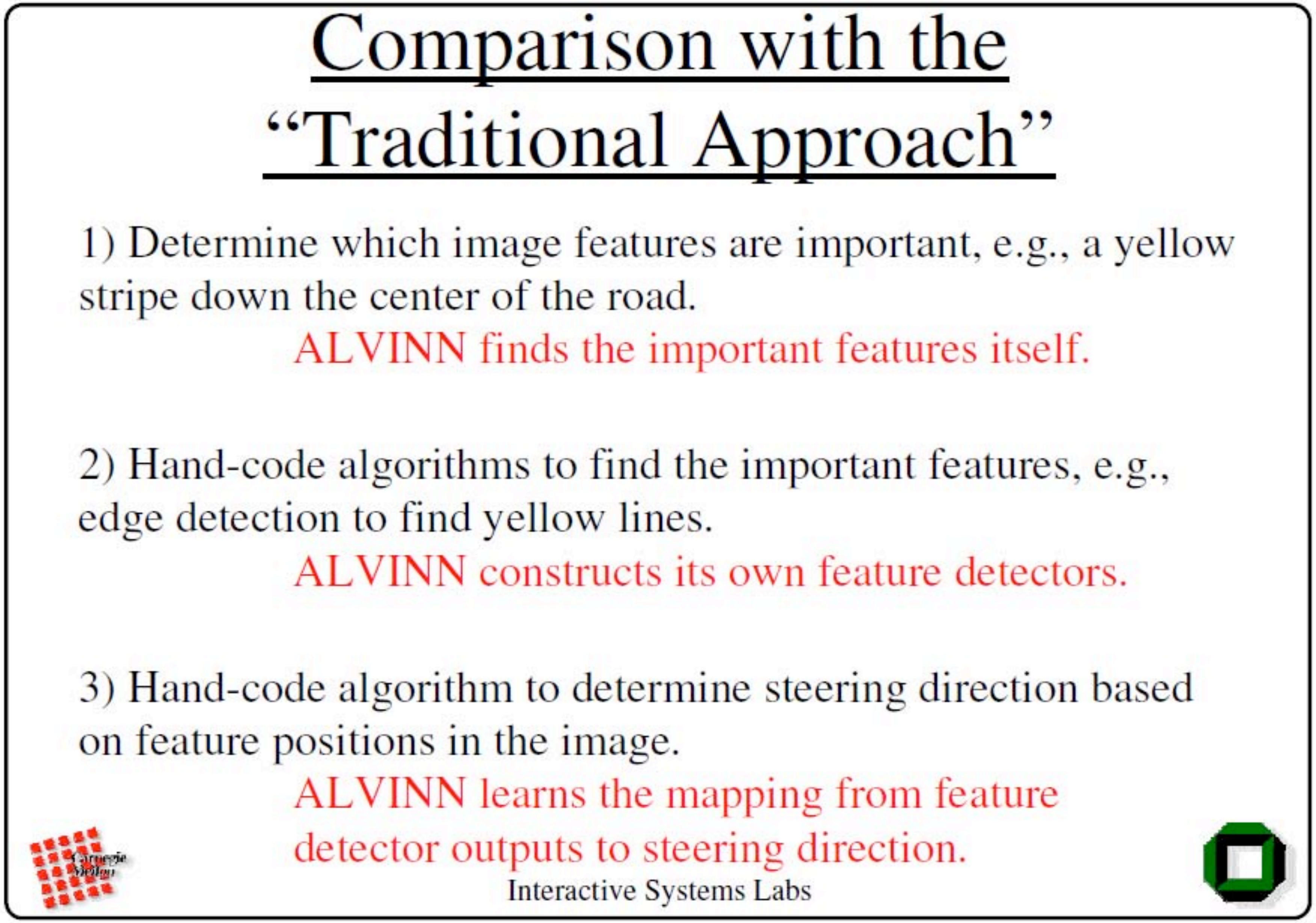


Regular Video

Laser Rangefinder

Laser Reflectance

Interactive Systems Labs



Comparison with the "Traditional Approach"

1) Determine which image features are important, e.g., a yellow stripe down the center of the road.

ALVINN finds the important features itself.

2) Hand-code algorithms to find the important features, e.g., edge detection to find yellow lines.

ALVINN constructs its own feature detectors.

3) Hand-code algorithm to determine steering direction based on feature positions in the image.

ALVINN learns the mapping from feature detector outputs to steering direction.

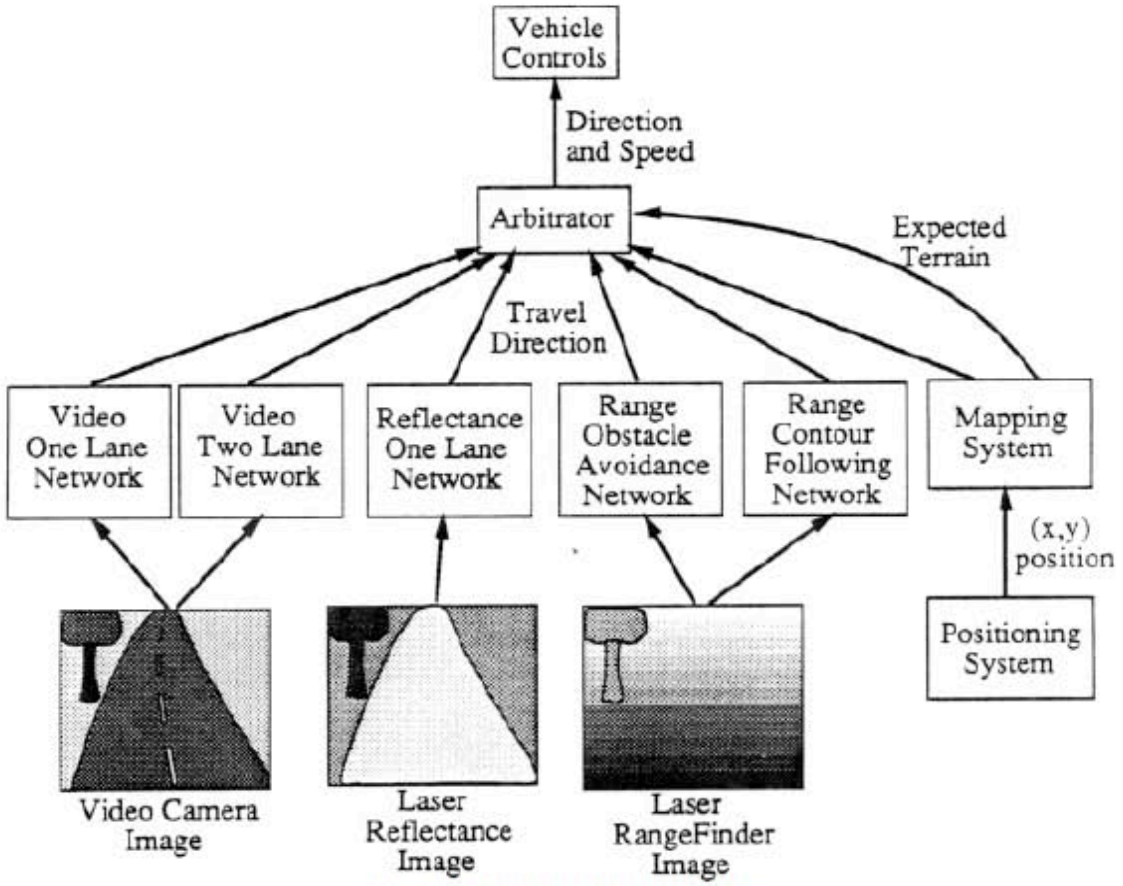
Interactive Systems Labs

ALVINN's Shortcomings

- The single-network ALVINN architecture can only drive on one type of road (unpaved, single-lane, double-lane, lane-striped, etc.)
- Can't transition from one road type to another.
- Can't follow a route.
- Solution: rule-based multi-network integration.

Interactive Systems Labs

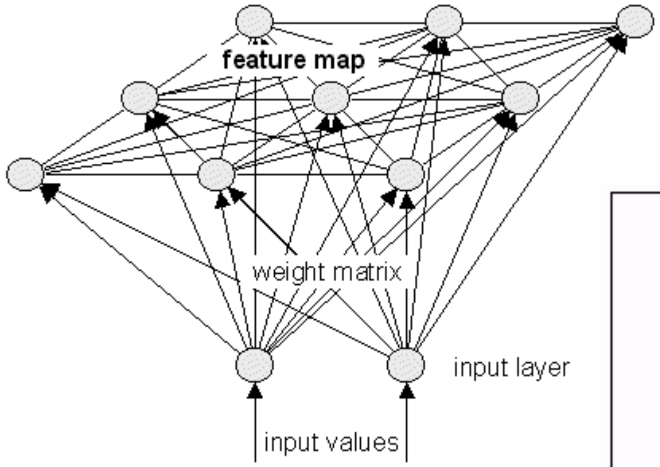
Hybrid ALVINN Architecture

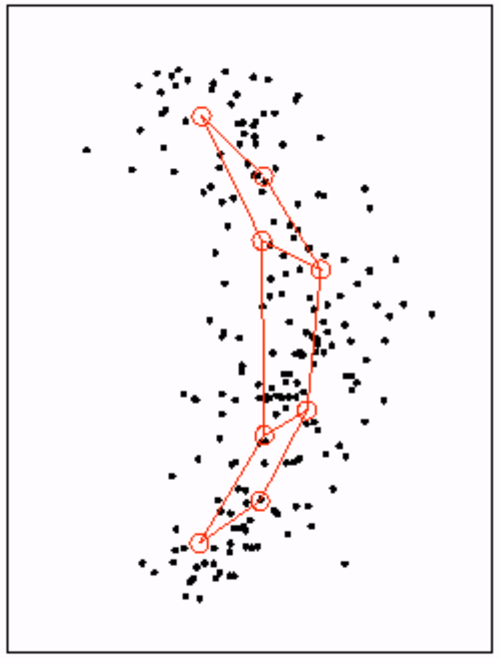


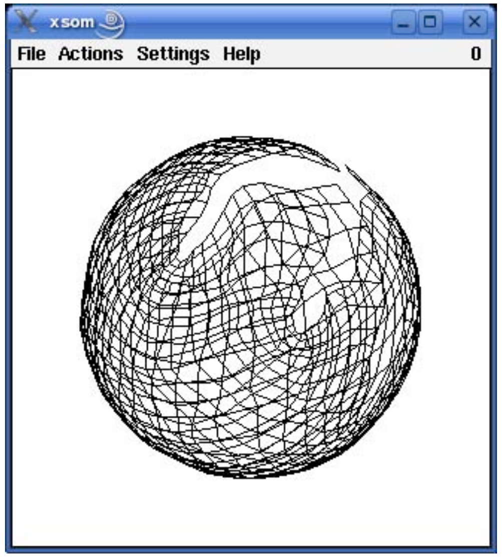
Interactive Systems Labs

Other types of ANN

Clustering, topological mapping...







Homunculus

Are ANN good for everything?

Types of learning:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Nature of data – sensors

- Information obtained from real world has completely different nature than the discrete data stored in the computer: sensors provide noisy data and algorithms must cope with that!
- Sensors never provide a complete information about the state of the environment – only measure some physical variables / phenomena with a bounded precision and certainty
- Information from the sensors is not available at any time, obtaining the data costs time and resources

Why Probabilities

- Real environments imply uncertainty in accuracy of
- robot actions
- sensor measurements
- Robot accuracy and correct models are vital for successful operations
- All available data must be used
- A lot of data is available in the form of probabilities

What Probabilities

- Sensor parameters
- Sensor accuracy
- Robot wheels slipping
- Motor resolution limited
- Wheel precision limited
- Performance alternates based on temperature, etc.

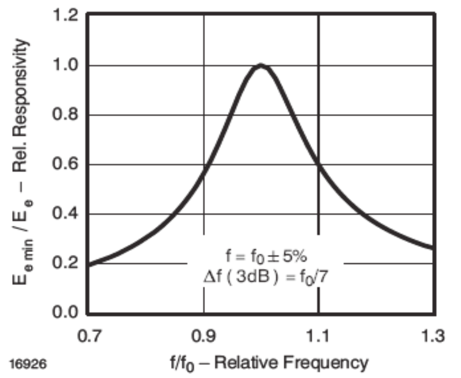


Figure 5. Frequency Dependence of Responsivity

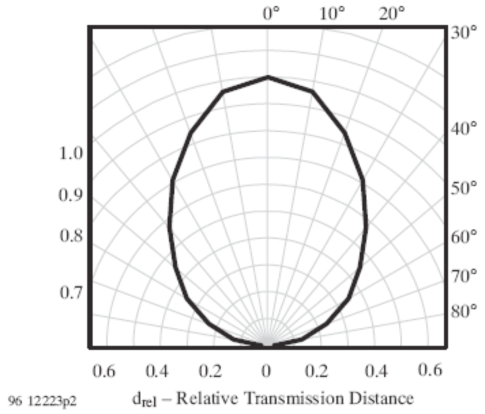
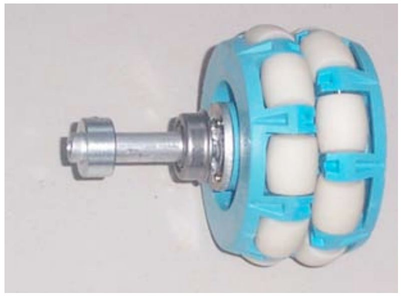


Figure 12. Directivity



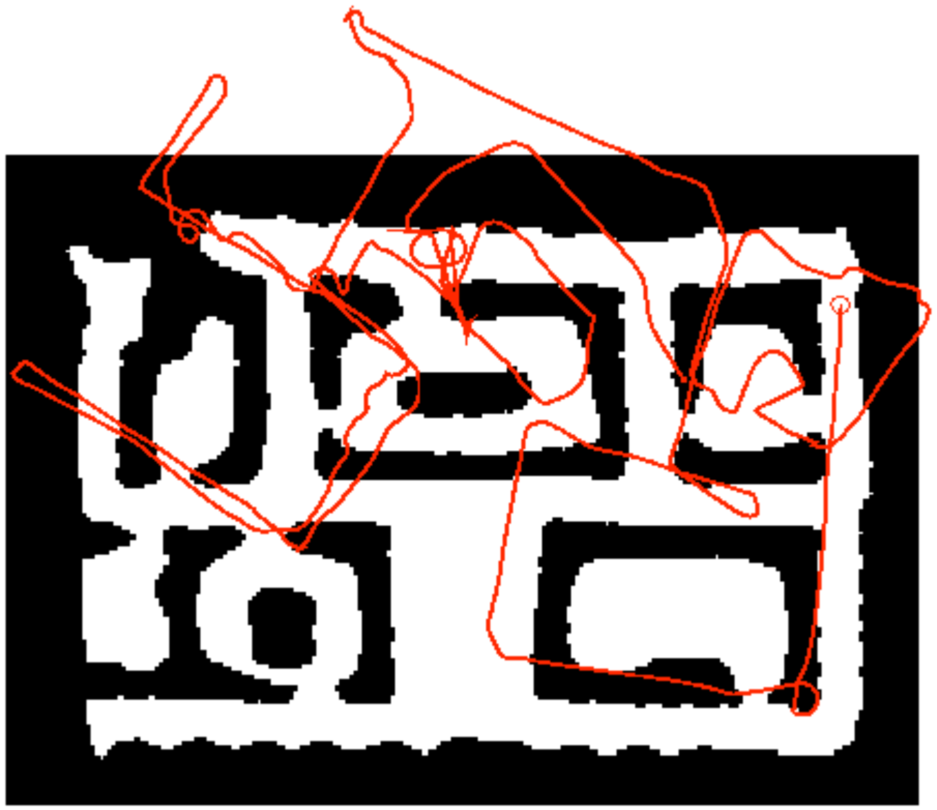
What Probabilities

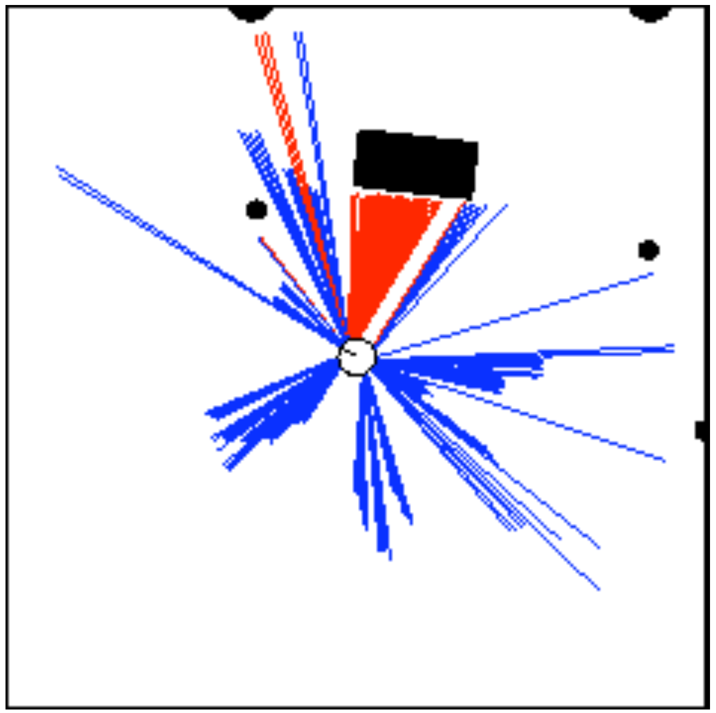
- These inaccuracies can be measured and modelled with random distributions
- Single reading of a sensor contains more information given the **prior** probability distribution of sensor behavior than its actual value
- Robot cannot afford throwing away this additional information!

What Probabilities

- More advanced concepts:
- Robot
  po
  sition and orientation (*robot pose*)
- Map of the environment
- Planning and control
- Action selection
- Reasoning...

Nature of Data


Odometry Data


Range Data

Simple Example of State Estimation

- Suppose a robot obtains measurement z
- What is *P(open|z)?*

Causal vs. Diagnostic Reasoning

- *P(open|z)* is diagnostic
- *P(z|open)* is causal
- Often causal knowledge is easier to obtain.
- Bayes rule allows us to use causal knowledge:

count frequencies!

Example

- $P(z|open) = 0.6$ $\qquad$ $P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$

- z raises the probability that the door is open

Combining Evidence

- Suppose our robot obtains another observation z_2
- How can we integrate this new information?
- More generally, how can we estimate $P(x|\ z_1 \ldots z_n)$?

Recursive Bayesian Updating

Markov assumption: z_n is independent of $z_1,\ldots,z_{n-1}$ if we know x.

$$P(x \mid z_1,\ldots,z_n) = \frac{P(z_n \mid x)\, P(x \mid z_1,\ldots,z_{n-1})}{P(z_n \mid z_1,\ldots,z_{n-1})}$$

$$= \eta\, P(z_n \mid x)\, P(x \mid z_1,\ldots,z_{n-1})$$

$$= \eta_{1\ldots n} \prod_{i=1\ldots n} P(z_i \mid x)\, P(x)$$

Example: Second Measurement

- $P(z_2|open) = 0.5$ $P(z_2|\neg open) = 0.6$
- $P(open|z_1)=2/3$

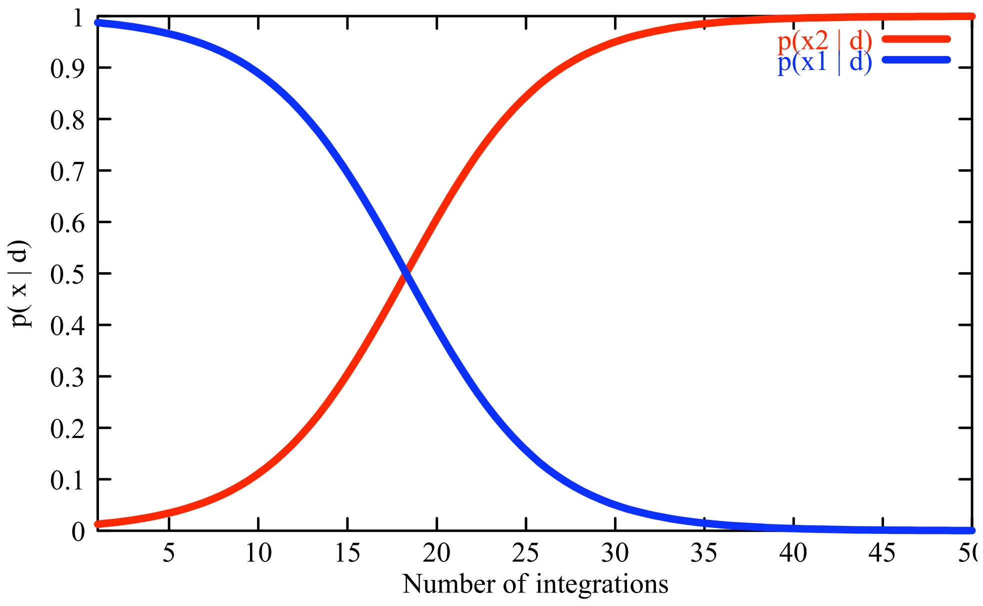
$$P(open \mid z_2, z_1) = \frac{P(z_2 \mid open)\, P(open \mid z_1)}{P(z_2 \mid open)\, P(open \mid z_1) + P(z_2 \mid \neg open)\, P(\neg open \mid z_1)}$$

$$= \frac{\frac{1}{2} \cdot \frac{2}{3}}{\frac{1}{2} \cdot \frac{2}{3} + \frac{3}{5} \cdot \frac{1}{3}} = \frac{5}{8} = 0.625$$

- z_2 lowers the probability that the door is open

A Typical Pitfall

- Two possible locations x_1 and x_2
- $P(x_1)=0.99$
- $P(z|x_2)=0.09$ $P(z|x_1)=0.07$



Actions

- Often the world is **dynamic** since
  - **actions carried out by the robot**,
  - **actions carried out by other agents**,
  - or just the **time** passing by

  change the world.

- How can we **incorporate** such **actions**?

Typical Actions

- The robot **turns its wheels** to move
- The robot **uses its manipulator** to grasp an object
- Plants grow over **time**…

- Actions are **never carried out with absolute certainty**.
- In contrast to measurements, **actions generally increase the uncertainty**.

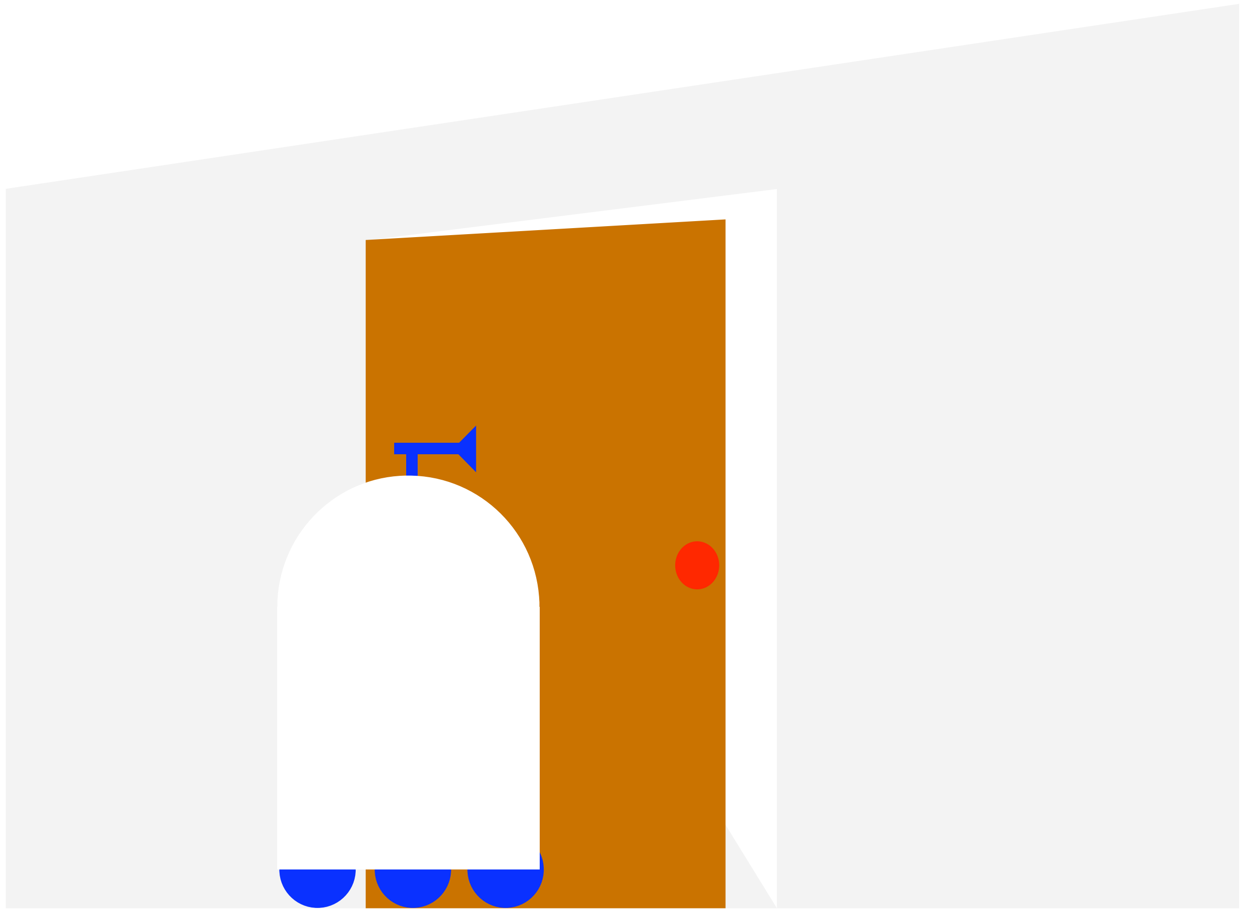
Modeling Actions

- To incorporate the outcome of an action *u* into th
e current “belief”, we use the conditional pdf

$$P(x|u,x')$$

- This term specifies the pdf that
**e
xecuting *u* changes the state from *x' to x***

Example: Closing the door



State Transitions

$P(x|u,x')$ for u = “close door”:

If the door is open, the action “close door” succeeds in 90% of all cases

Integrating the Outcome of Actions

Continuous case:

Discrete case:

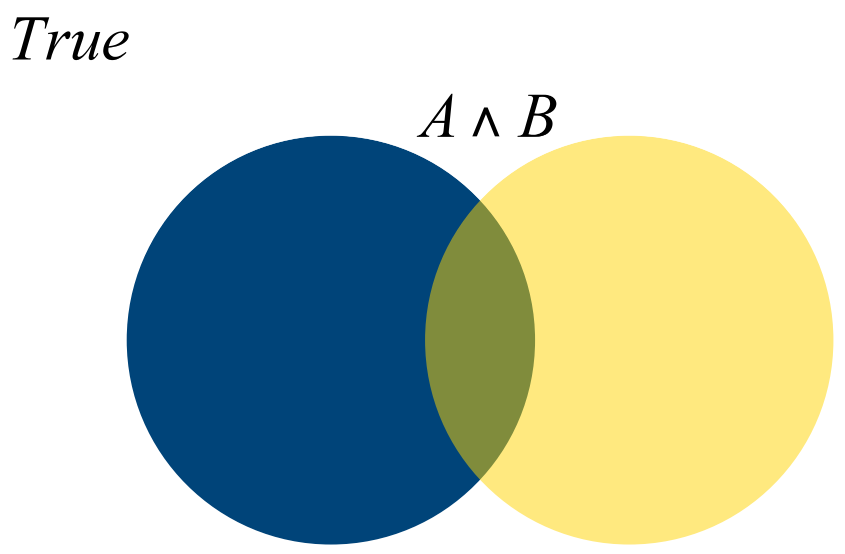
Example: The Resulting Belief

Axioms of Probability Theory

Pr*(A)* denotes probability that proposition *A* is true.

- $$0 \le \Pr(A) \le 1$$
- $$\Pr(True) = 1$$
-

A Closer Look at Axiom 3



Using the Axioms

Discrete Random Variables

- *X* denotes a random variable.
- *X* can take on a countable number of values in $\{x_1, x_2, \ldots, x_n\}$.
- $P(X=x_i)$, or $P(x_i)$, is the probability that the random variable *X* takes on value x_i.
- $P(X)$ is called probability mass function.
- E.g.

Continuous Random Variables

- *X* takes on values in the continuum.
- *p(X=x)*, or *p(x)*, is a probability density function.

$$\Pr(x \in (a,b)) = \int_a^b p(x)\,dx$$

- E.g.

Joint and Conditional Probability

- $P(X{=}x \text{ and } Y{=}y) = P(x,y)$
- If X and Y are independent then

  $P(x,y) = P(x)\,P(y)$
- $P(x \mid y)$ is the probability of x given y

  $P(x \mid y) = P(x,y) / P(y)$

  $P(x,y) \quad = P(x \mid y)\,P(y)$
- If X and Y are independent then

  $P(x \mid y) = P(x)$

Law of Total Probability, Marginals

Discrete case

$$\sum_x P(x) = 1$$

$$P(x) = \sum_y P(x, y)$$

$$P(x) = \sum_y P(x \mid y) P(y)$$

Continuous case

$$\int p(x)\, dx = 1$$

Bayes Formula

$$P(x, y) = P(x \mid y)P(y) = P(y \mid x)P(x)$$

$$\Rightarrow$$

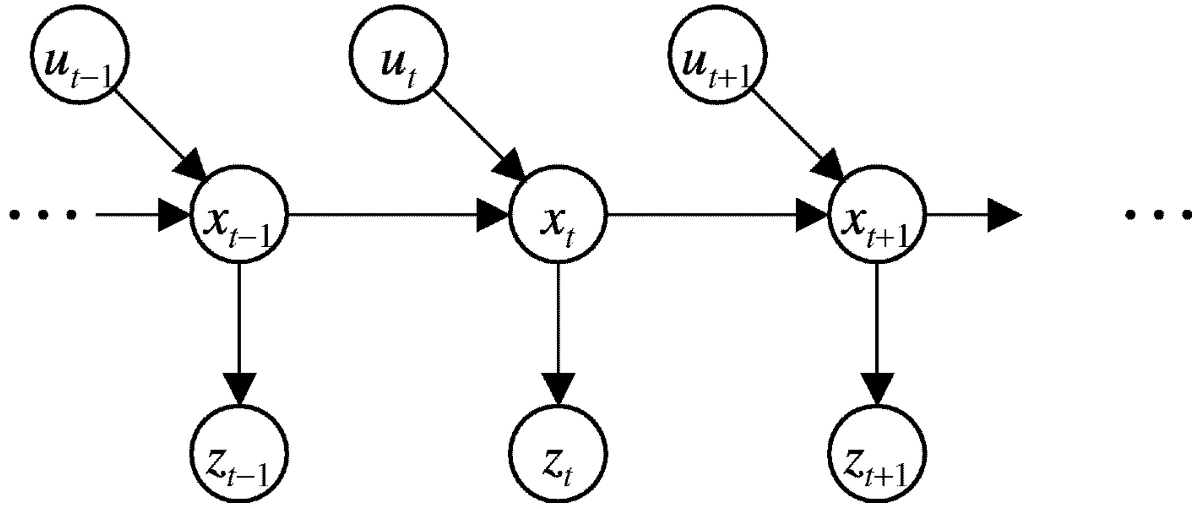
$$P(x \mid y) = \frac{P(y \mid x)\ P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

Bayes Filters: Framework

- **Given:**
  - Stream of observations *z* and action data *u:*

  - Sensor model *P(z|x).*
  - Action model *P(x|u,x')*.
  - Prior probability of the system state *P(x).*
- **Wanted:**
  - Estimate of the state *X* of a dynamical system.
  - The posterior of the state is also called **Belief**:

Markov Assumption


$$\begin{tikzcd}
 & u_{t-1} \arrow[dr] & & u_t \arrow[dr] & & u_{t+1} \arrow[dr] & & \\
\cdots \arrow[rr] & & x_{t-1} \arrow[rr] \arrow[d] & & x_t \arrow[rr] \arrow[d] & & x_{t+1} \arrow[r] \arrow[d] & \cdots \\
 & & z_{t-1} & & z_t & & z_{t+1} &
\end{tikzcd}$$

Underlying Assumptions

- Static world
- Independent noise
- Perfect model, no approximation errors

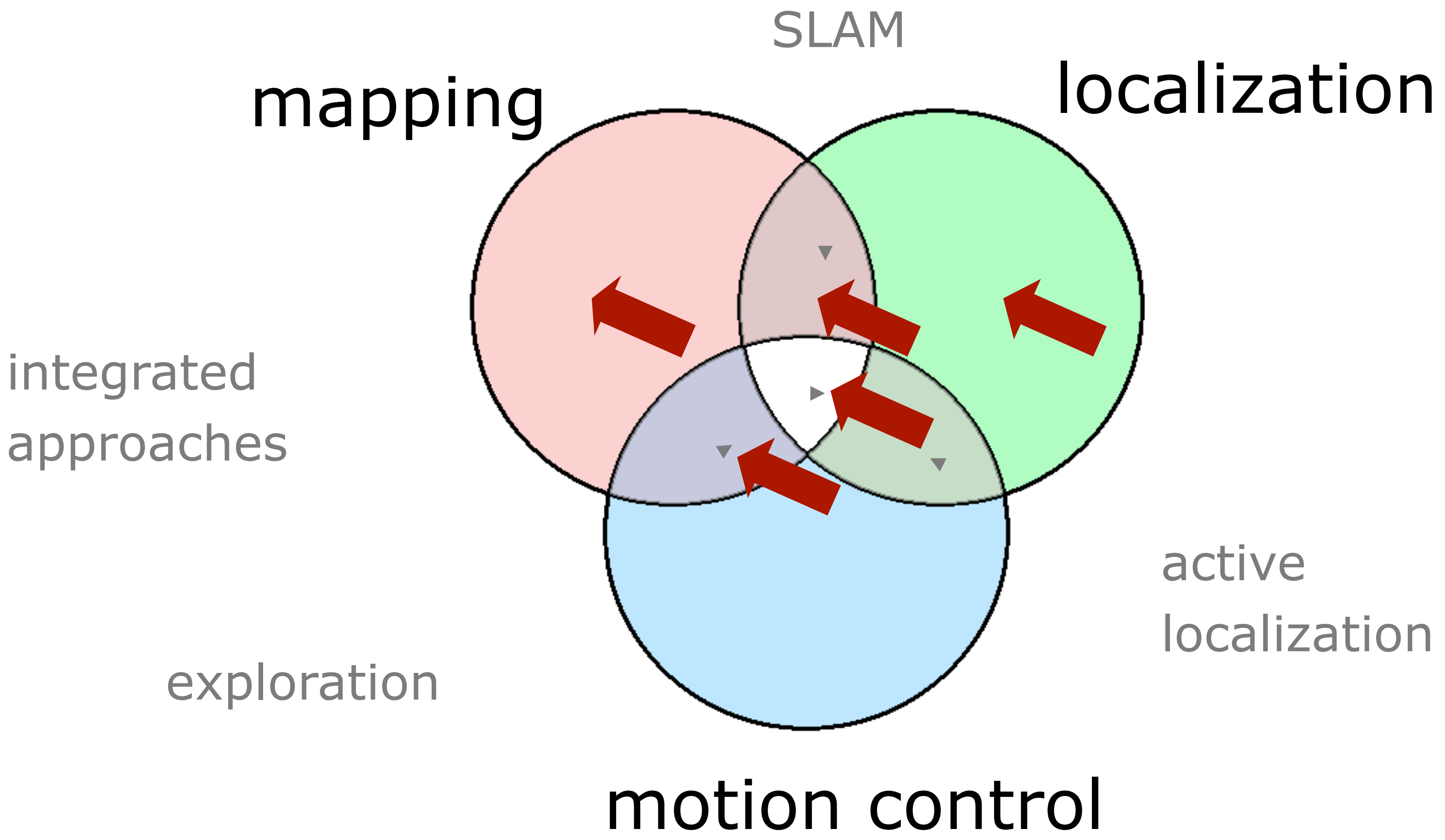
Bayes Filters are Familiar!

- Kalman filters
- Discrete filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially
  O
  b
  servable Markov Decision Processes (POMDPs)

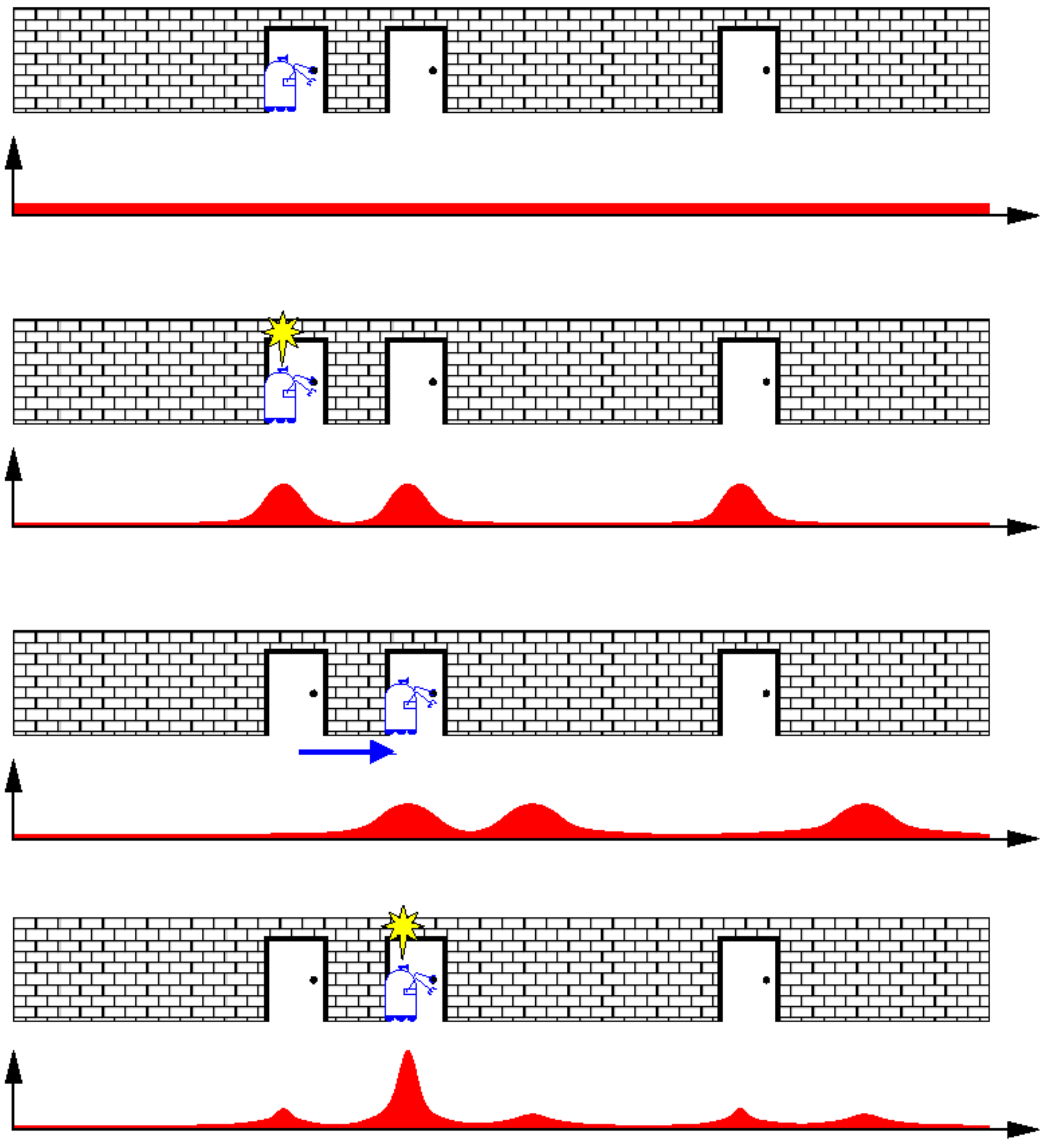
Summary

- Bayes rule allows us to compute probabilities that are hard to assess otherwise
- Under the Markov assumption, recursive Bayesian updating can be used to efficiently combine evidence
- Bayes filters are a probabilistic tool for estimating the state of dynamic systems.

Dimensions of Mobile Robot Navigation



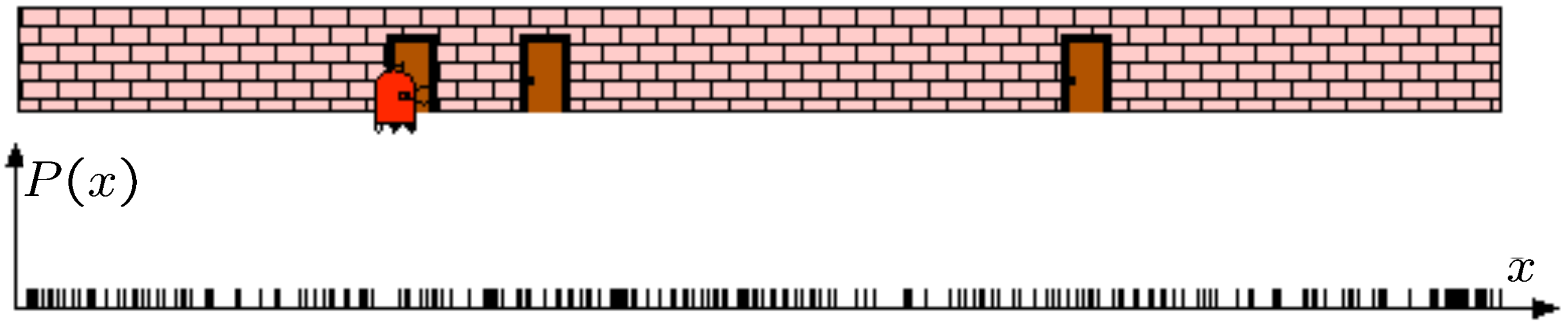
Probabilistic Localization



What is the Right Representation?

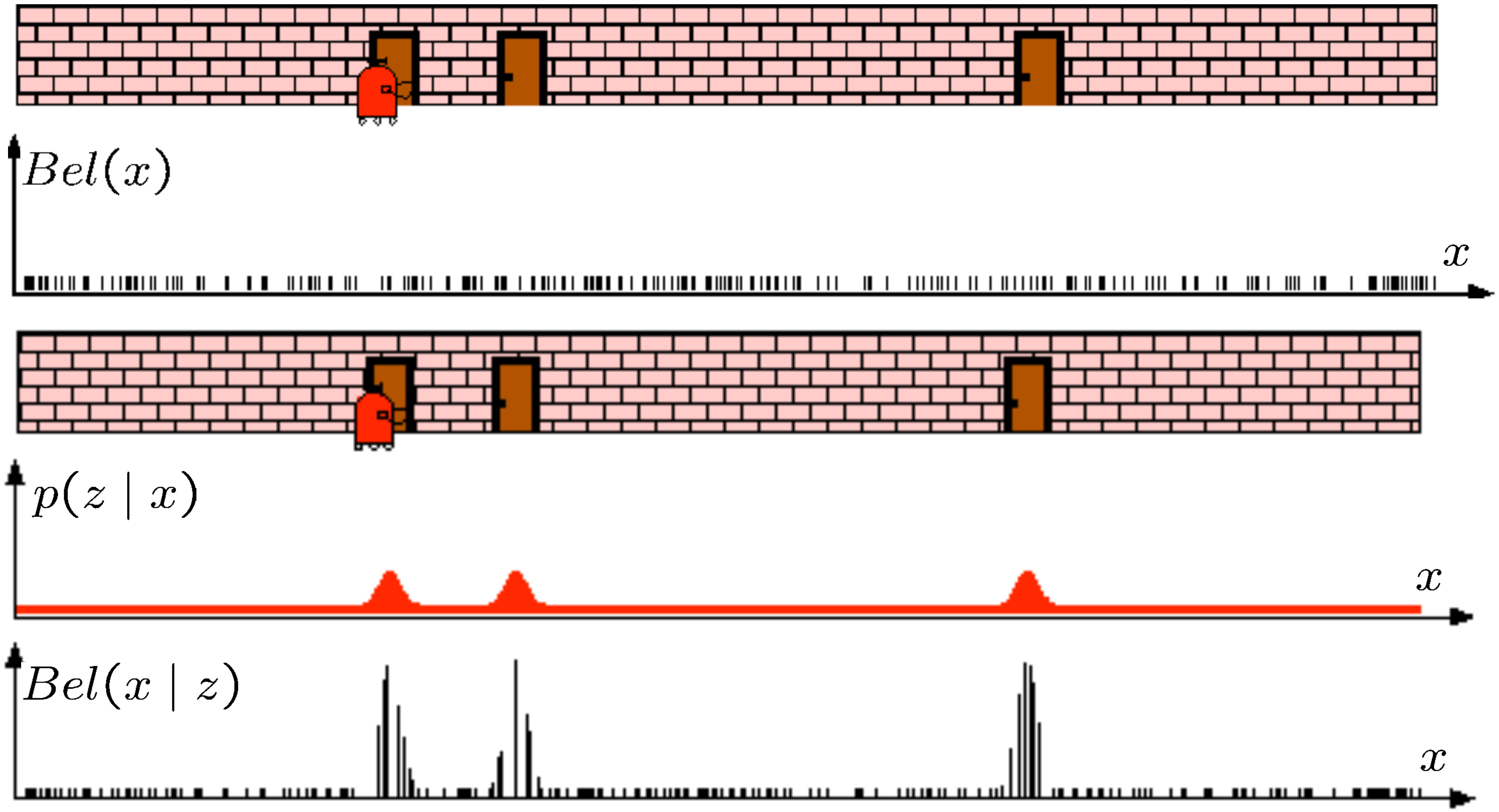
- Kalman filters
- Multi-hypothesis tracking
- Grid-based representations
- Topological approaches
- Particle filters

Mobile Robot Localization with Particle Filters



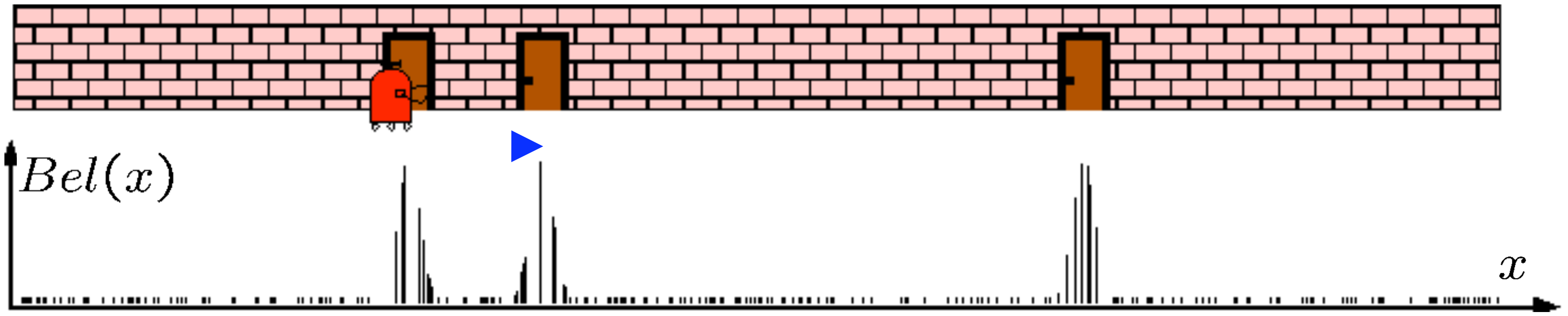
MCL: Sensor Update

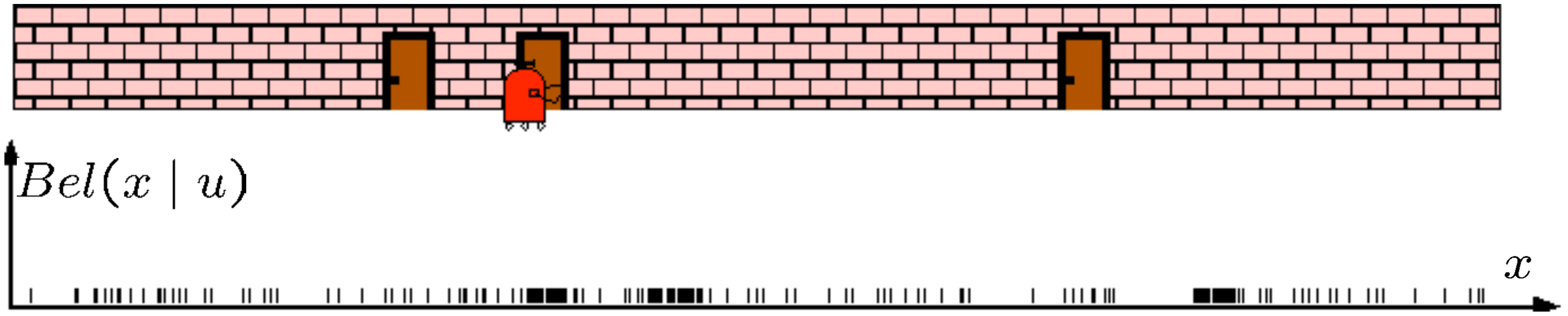
$$Bel(x \mid z) = \alpha\, p(z \mid x)\, Bel(x)$$



PF: Robot Motion

$$Bel(x \mid u) = \int_{x'} p(x \mid u, x')\, Bel(x')$$

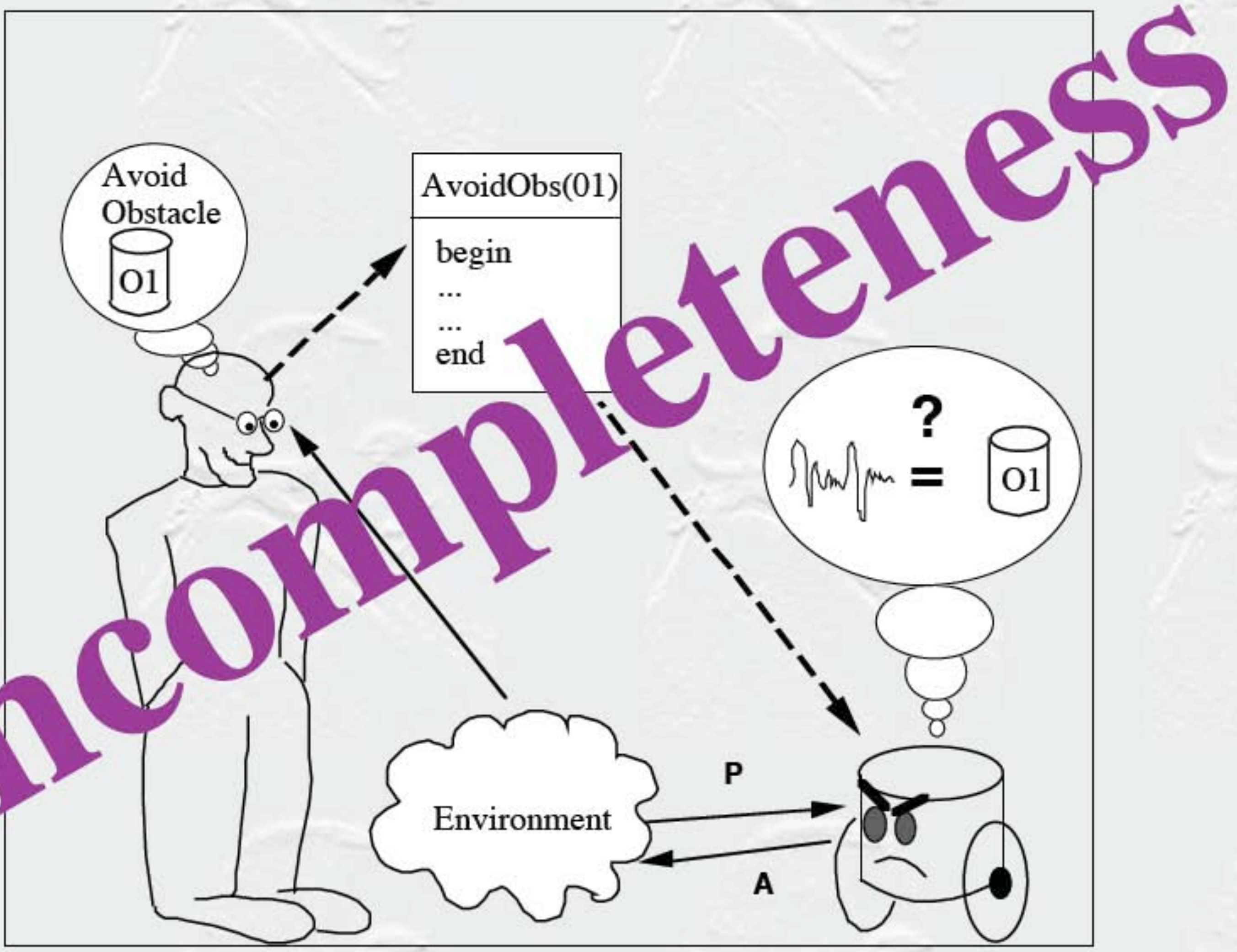




Bayesian Robot Programming

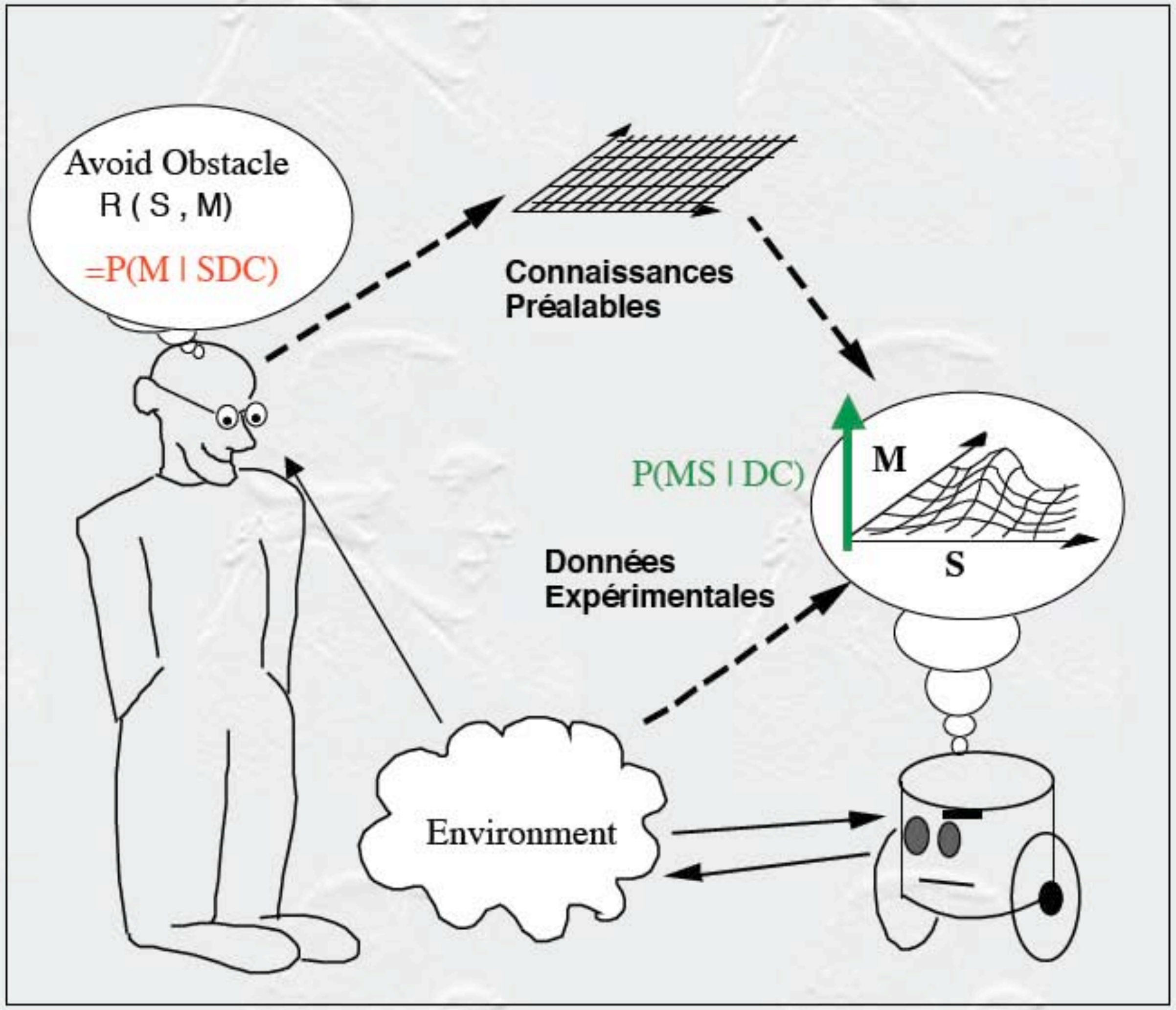
- Integrated approach where parts of the robot interacti
  on with the world are modelled by probabilities
- Example: training a Khepera robot
- (video)

Logical Paradigm



CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Bayesian Paradigm



Principle

Incompleteness

Bayesian Learning

Preliminary Knowledge
+
Experimental Data
=
Probabilistic Representation

Uncertainty

Bayesian Inference

$$P(a) + P(\neg a) = 1$$
$$P(a \wedge b) = P(a)P(b \mid a) = P(b)P(a \mid b)$$

Decision

Bayesian Program

- Program
  - Description
    - *Specification*
      - Variables
      - Decomposition
      - Parametrical Forms or Recursive Question

      Preliminary Knowledge π
    - *Identification*

      Experimental Data δ
  - Question
    - Utilization

Pushing Objects

Program

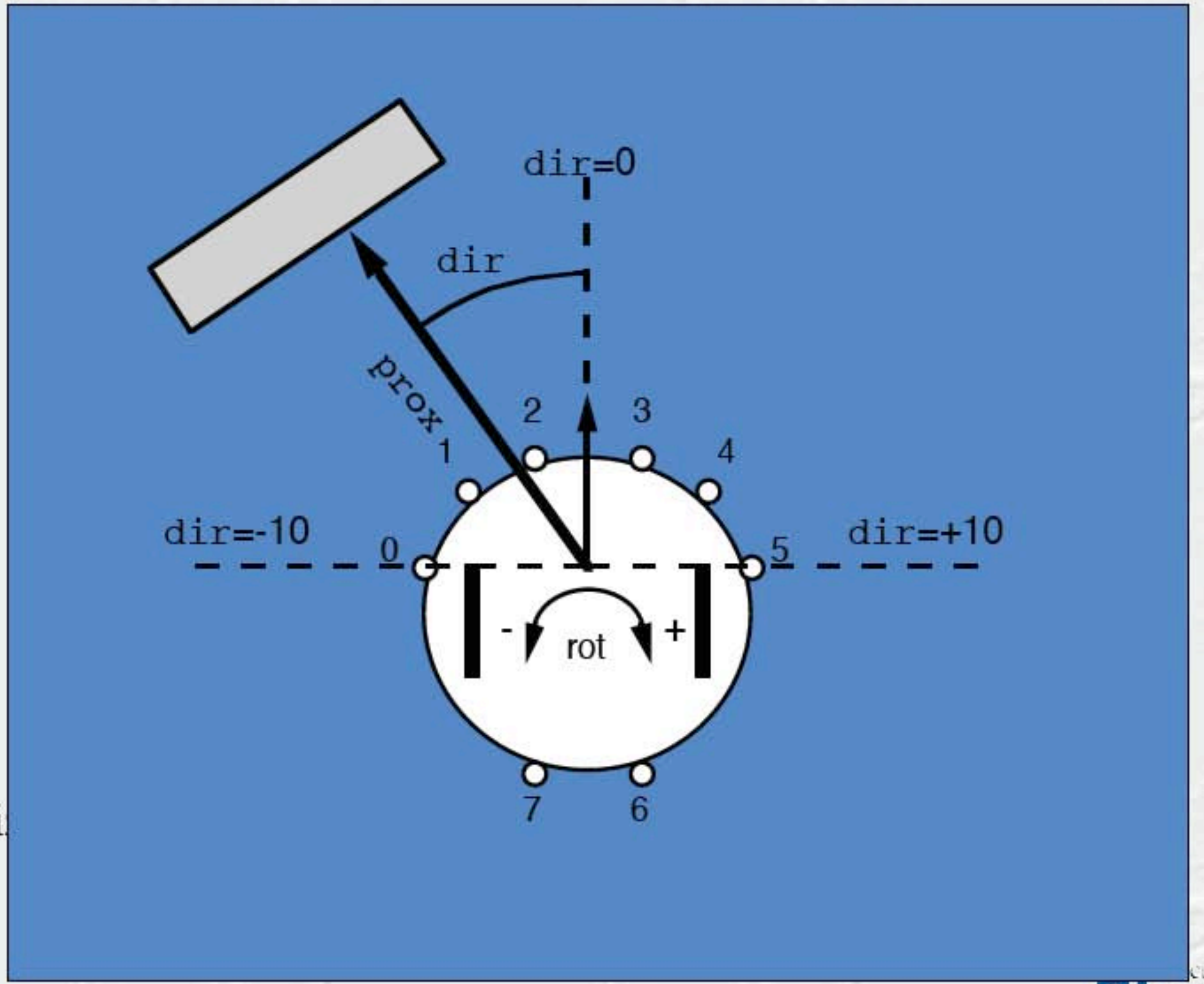
Description

Question

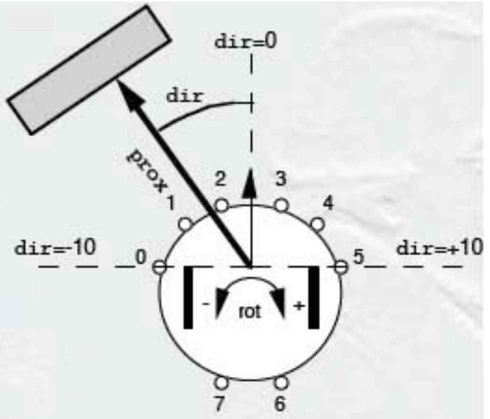
Specification

- Variables

Utili



Pushing Objects



Program

Description

Specification

- Variables

$$Dir \wedge Prox \wedge Vrot$$

- Decomposition

$$P(Dir \wedge Prox \wedge Vrot \mid \delta \wedge \pi)$$

$$= P(Dir \mid \delta \wedge \pi) \times P(Prox \mid \delta \wedge \pi) \times P(Vrot \mid Dir \wedge Prox \wedge \delta \wedge \pi)$$

- Parametrical Forms

$$P(Dir \wedge Prox \mid \delta \wedge \pi) \leftarrow \text{Uniform}$$

$$P(Vrot \mid Dir \wedge Prox \wedge \delta \wedge \pi) \leftarrow \text{Gaussians}$$

➔ Preliminary Knowledge π

Identification

- Joystick Remote Control ➔ Experimental Data δ1

$$P(Dir \wedge Prox \wedge Vrot \mid \delta 1 \wedge \pi)$$

Question

Utilization

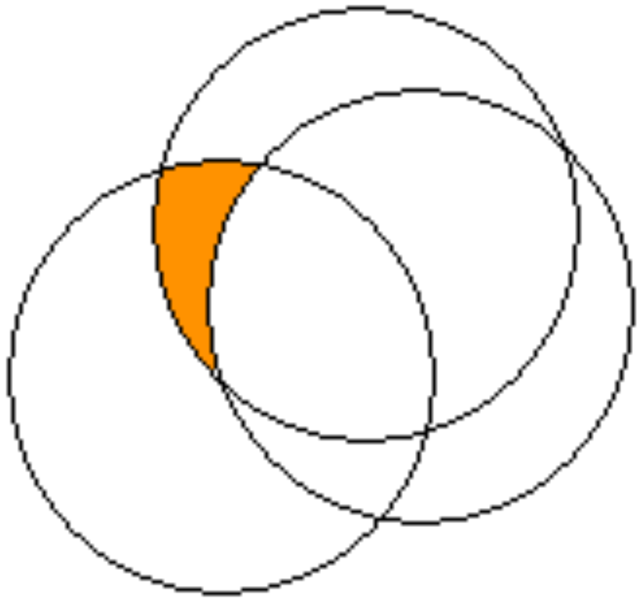
$$P(Vrot \mid [Dir = d] \wedge [Prox = p] \wedge \delta 1 \wedge \pi)$$

Further Information

- Recently published book: *Pierre Bessière, Juan-Manuel Ahuactzin, Kamel Mekhnacha, Emmanuel Mazer*: Bayesian Programming
- MIT Press Book (Intelligent Robotics and Autonomous Agents Series): *Sebastian Thrun, Wolfram Burgard, Dieter Fox*: Probabilistic Robotics
- ProBT library for Bayesian reasoning
- bayesian-cognition.org

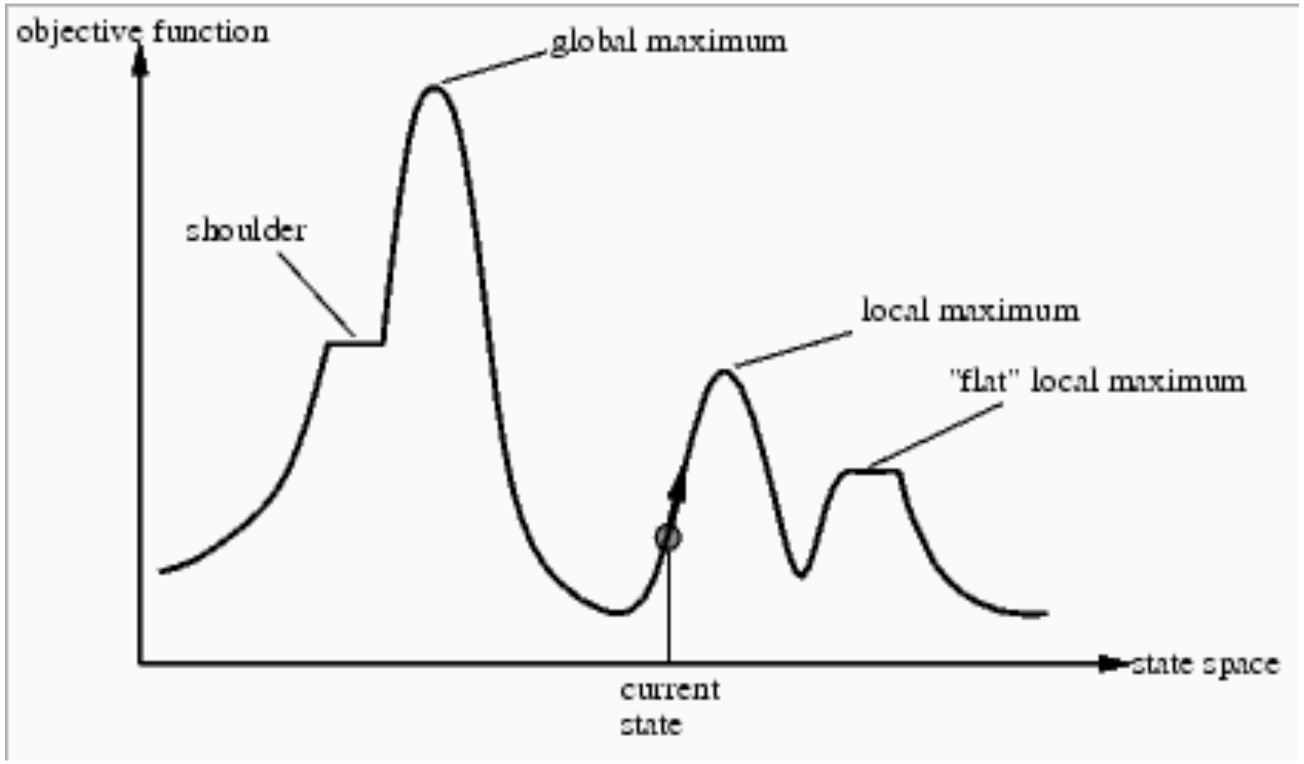
Stochastic methods: Monte Carlo

- Determine the area of a particular shape:



Stochastic methods: Simulated Annealing

- Navigating in the search space using local neighborhood:

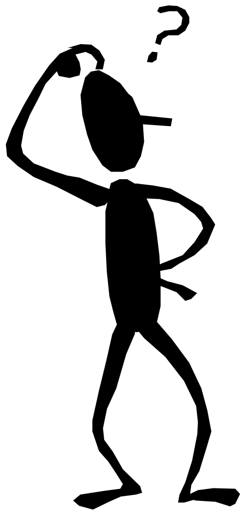


Principles of Natural Evolution

- Individuals have information encoded in genotypes that consist of genes, alleles
- The more successful individuals have higher chance of survival and therefore also higher chance of having descendants
- The overall population of individuals adapts to the changing conditions so that the more fit individuals prevail in the population
- Changes in the genotype are introduced through mutations and recombination

Evolutionary Computation

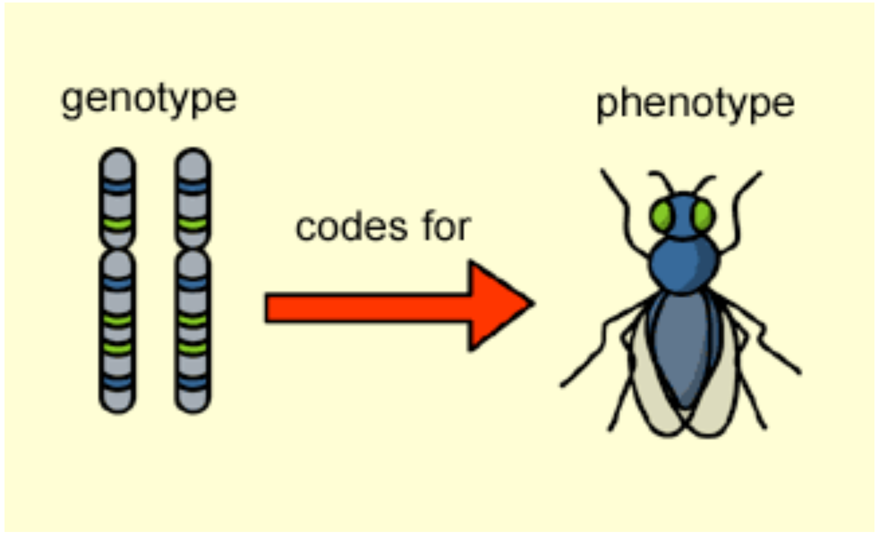
- Search for solutions to a problem
- Solutions uniformly encoded
- Fitness: objective quantitative measure
- Population: set of randomly generated solutions
- Principles of natural evolution:
  - selection, recombination, mutation
- Run for many generations



EA Concepts

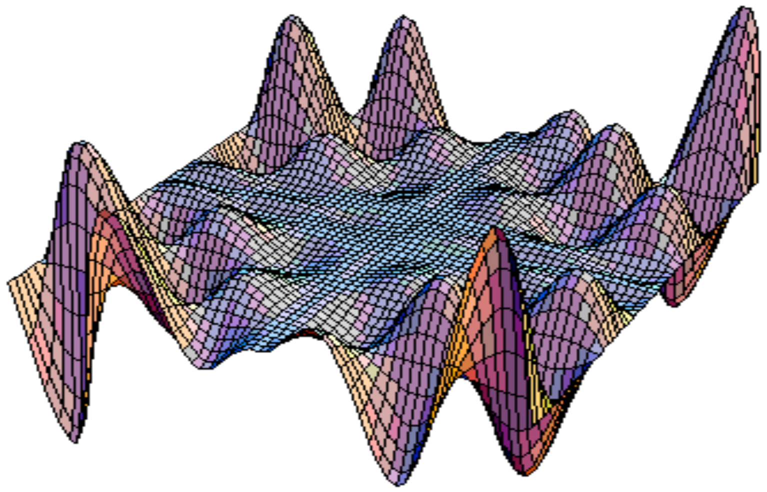
- genotype and phenotype
- fitness landscape
- diversity, genetic drift
- premature convergence
- exploration vs. exploitation
- selection methods: roulette wheel (fit.prop.), tournament, truncation, rank, elitist
- selection pressure
- direct vs. indirect representations
- fitness space

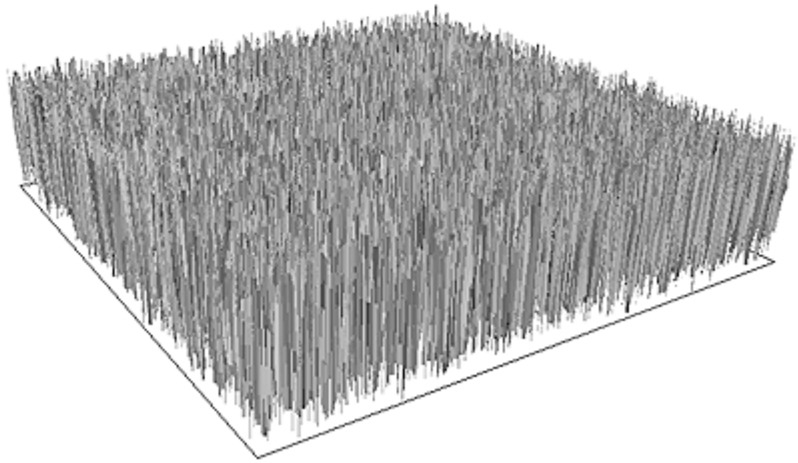
Genotype and Phenotype



- *Genotype* – all ge n etic material of a particular individual (genes)
- *P henotype* – the real features of that individual

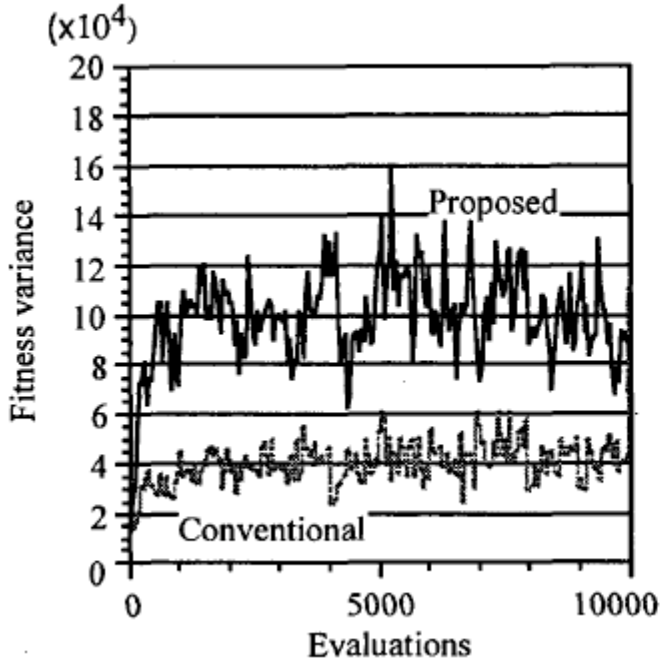
Fitness landscape

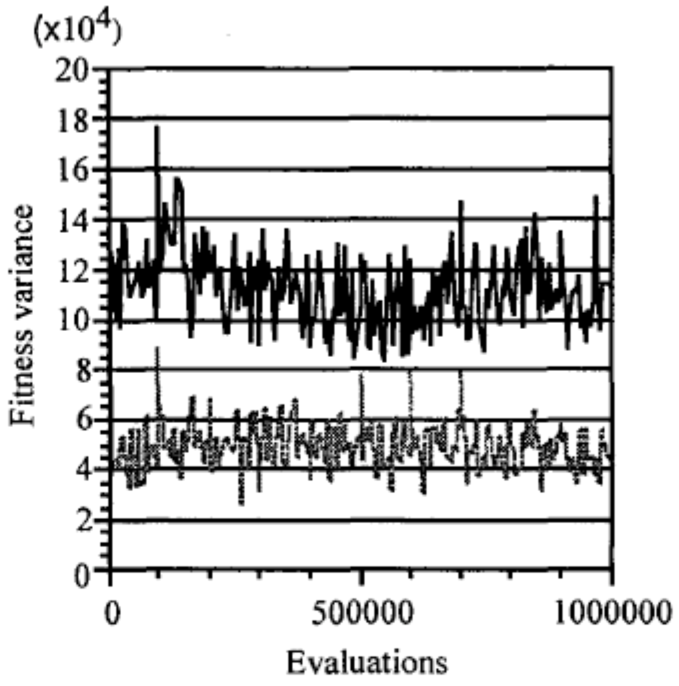




- Genotype space – difficulty of the problem – shape of fitness landscape, neighborhood function

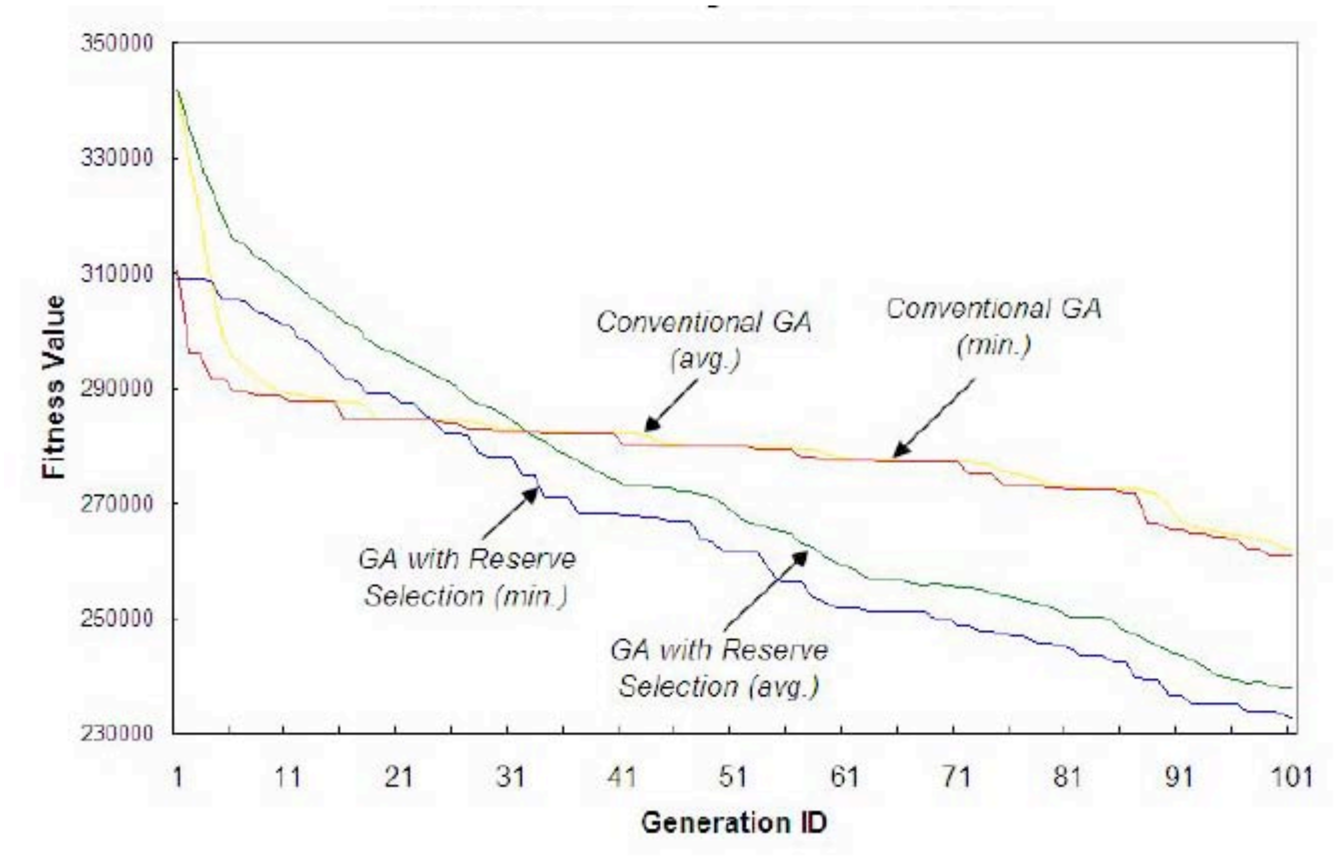
Population diversity





- Must be kept high for the evolution to advance

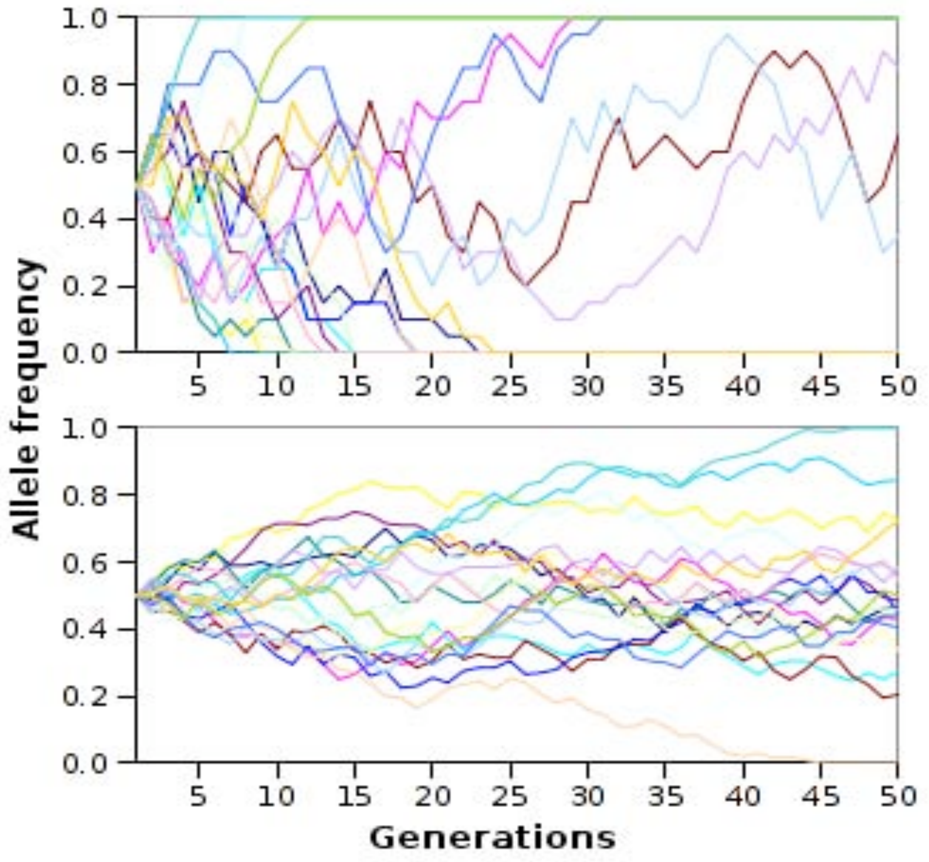
Premature convergence



- important building blocks are lost early in the evolutionary run

Genetic drift

- Loosing the population distribution due to the sampling error



Exploration vs. Exploitation

- Exploration phase: localize promising areas
- Exploitation phase: fine-tune the solution

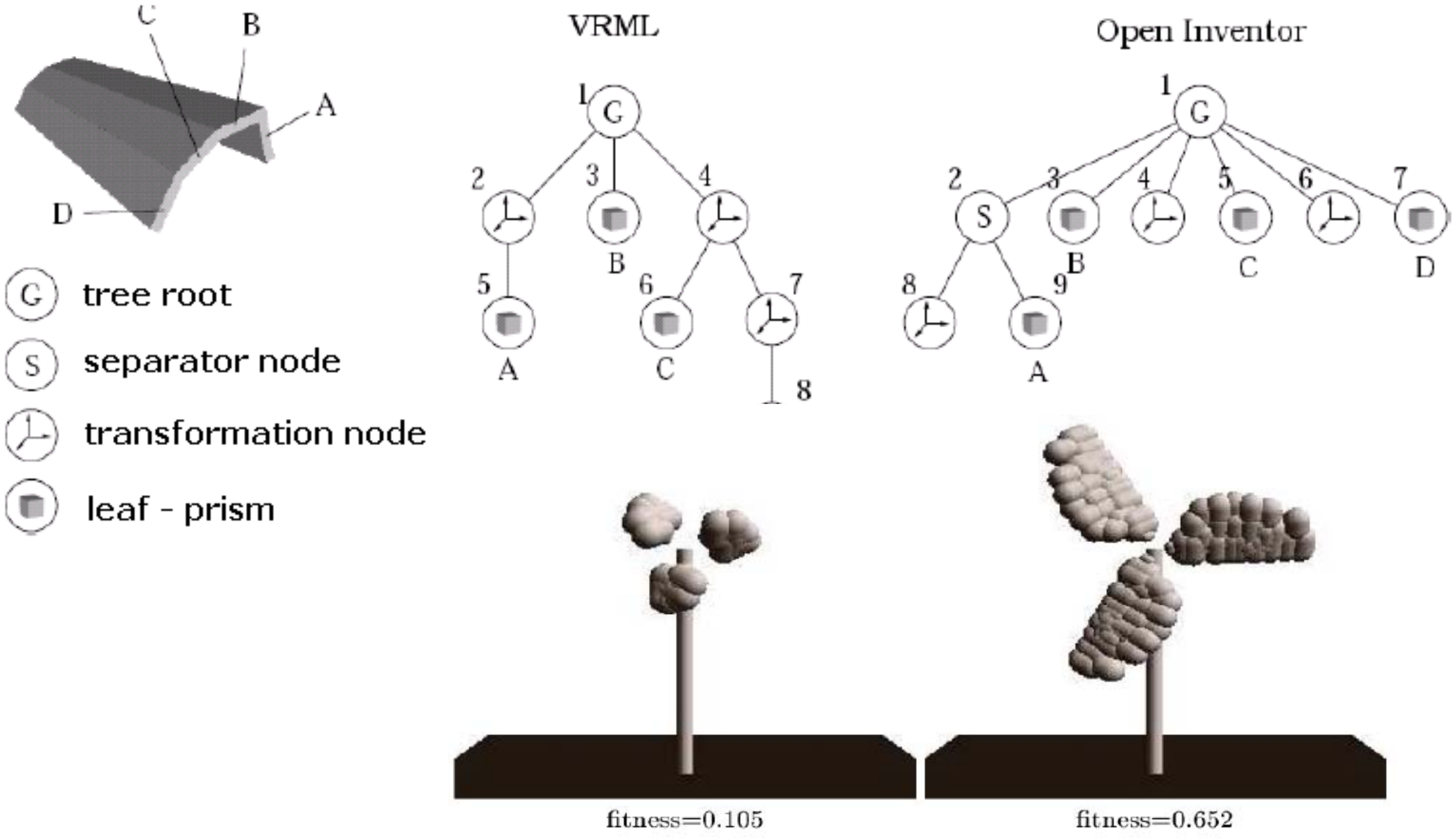
Selection methods

- roulette wheel (fitness proportionate selection),
- tournament selection
- truncation selection
- rank selection
- elitist strategies

Selection pressure

- Influenced by the problem
- Relates to evolutionary operators

Direct vs. Indirect Representations

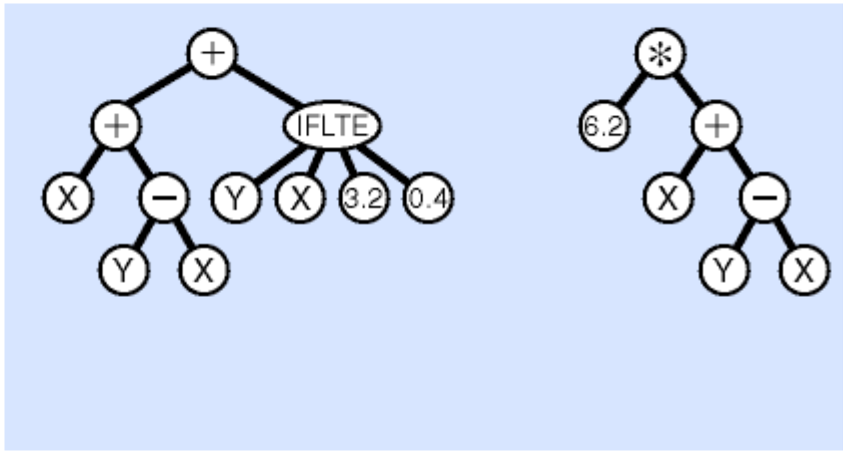


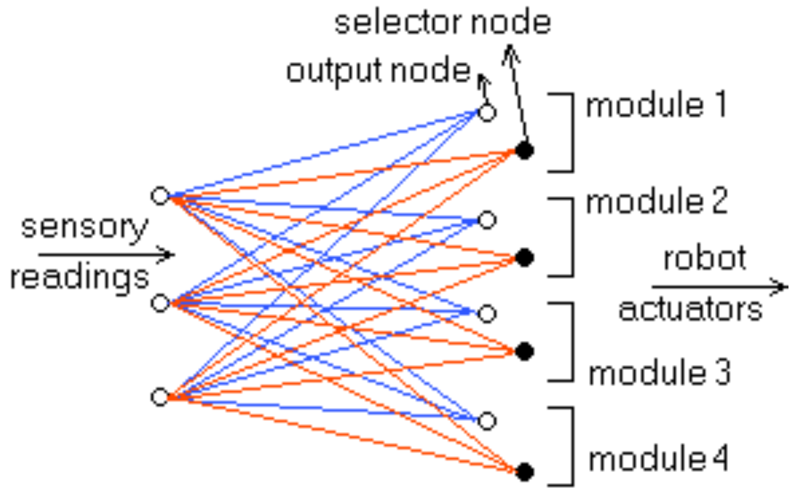
Fitness Space (Floreano)

- Functional vs. behavioral
- Explicit vs. implicit
- External vs. internal

Evolutionary Robotics

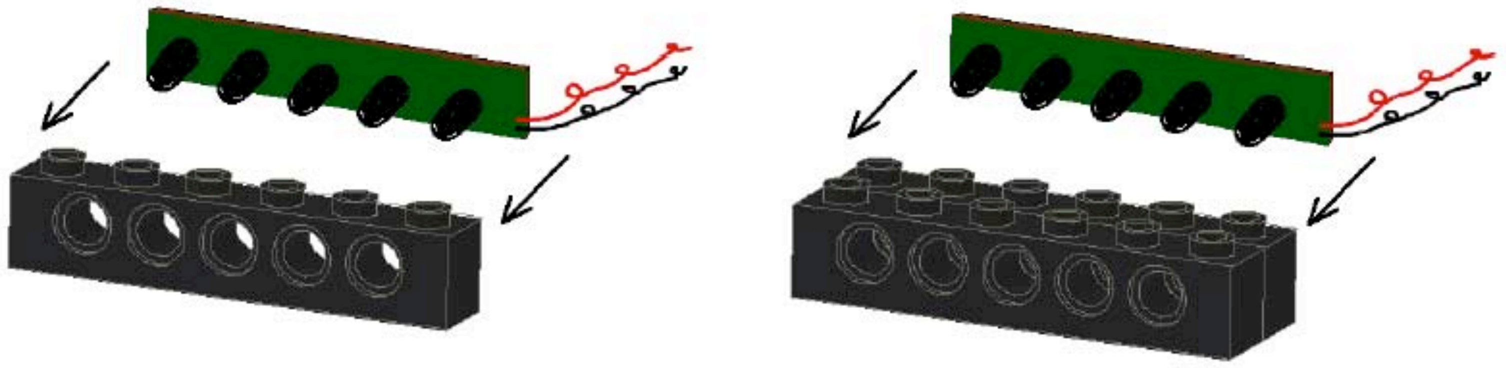
- Solution: Robot’s controller



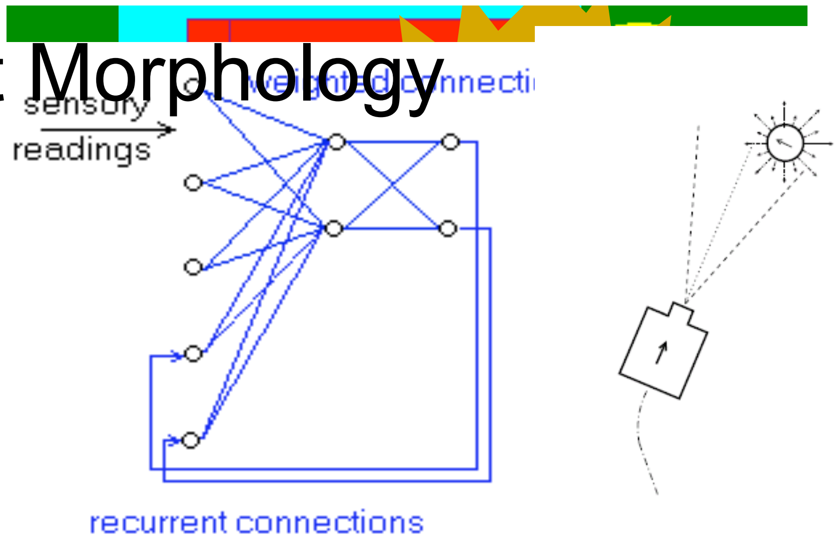


- Fitness: how well the robot performs
- Simulation or real robot

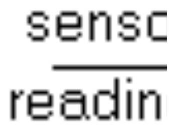
Fitness Influenced by

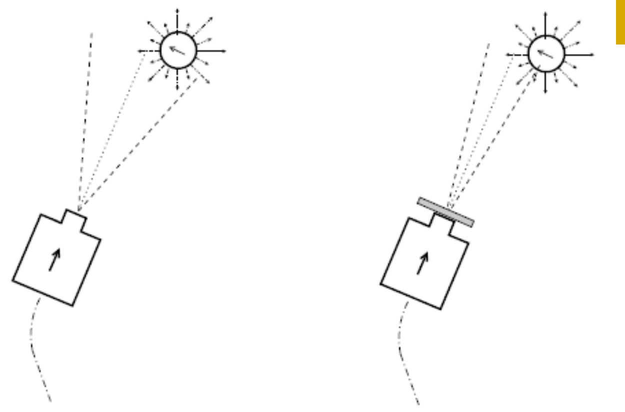


- Robot Morphology



Inc





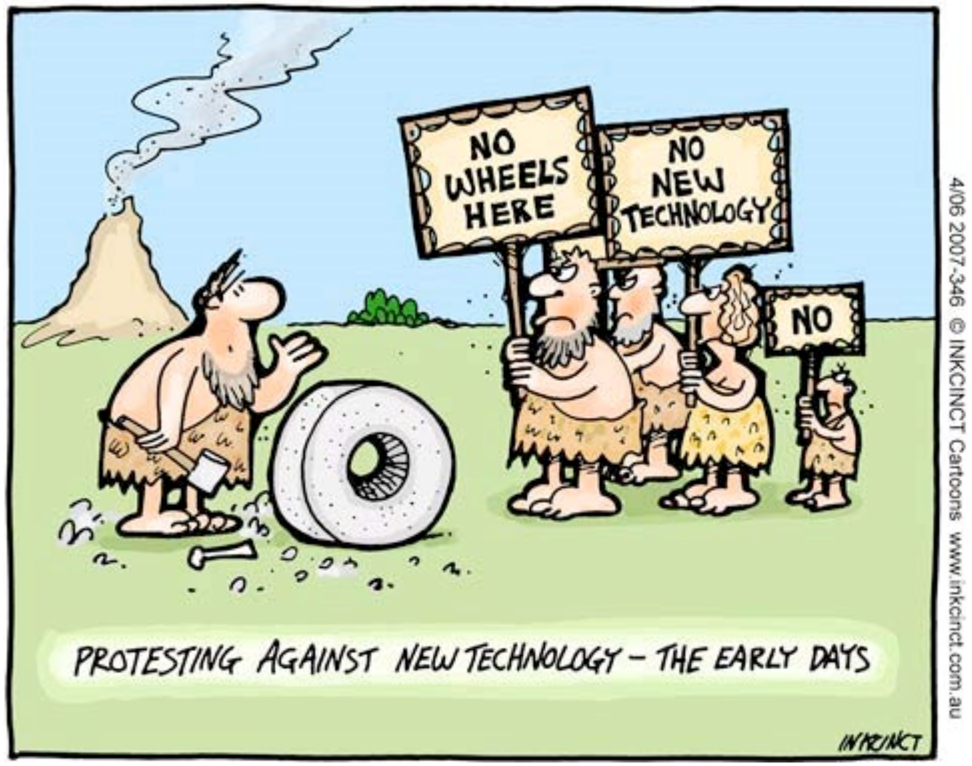
Evolvable Tasks

- Wall following
- Obstacle avoidance
- Docking and recharging
- Artificial ant following
- Box pushing
- Lawn mowing
- Legged walking
- T-maze navigation
- Foraging strategies
- Trash collection
- Vision discrimination and classification tasks
- Target tracking and navigation
- Pursuit-evasion behaviors
- Soccer playing
- Navigation tasks

Neuroevolution through augmenting topologies

- The most successful method for evolution of artificial neural networks
- Sharing fitness
- Starting with simple solutions
- Global counter
- i.e. Topological crossover – very important for preserving evolved structures

What is Learning?


NO WHEELS HERE
NO NEW TECHNOLOGY
NO
PROTESTING AGAINST NEW TECHNOLOGY – THE EARLY DAYS
INKCINCT
4/06 2007-346 © INKCINCT Cartoons www.inkcinct.com.au