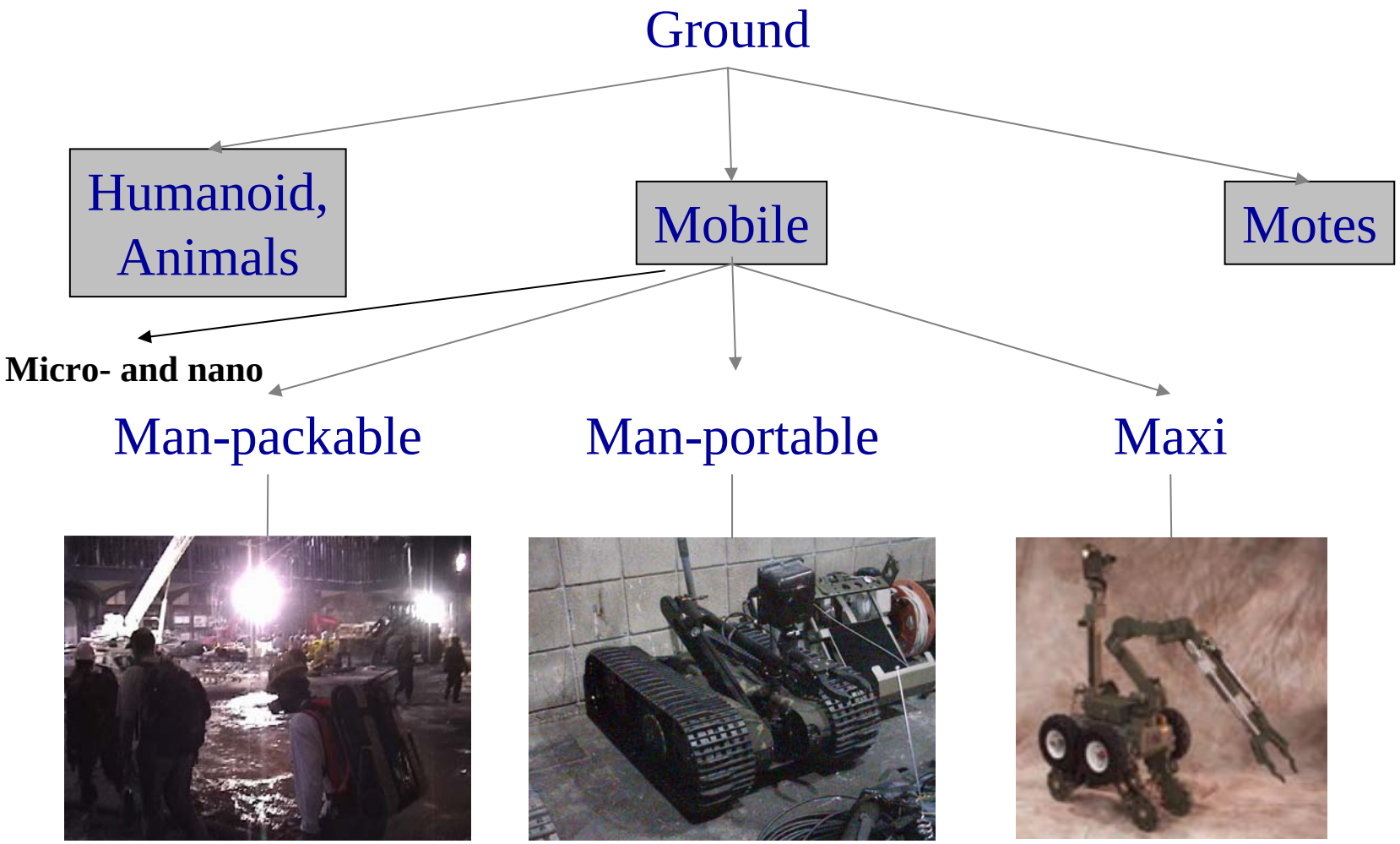
Hybrid Intelligent Systems

Lecture 15

Neural Networks in Robotics

Taxonomy of Mobile Robots



Robin Murphy "Introduction to AI Robotics"(2004, MIT)

Control Approaches

- *Reactive Control*
  - *Don't think, (re)act.*
- *Deliberative Control*
  - *Think hard, act later.*
- *Hybrid Control*
  - *Think and act independently, in parallel.*
- *Behavior-Based Control*
  - *Think the way you act.*

Reactive Systems

- Collections of *sense-act (stimulus-response) rules*
- Inherently *concurrent (parallel)*
- No/minimal state
- No memory
- Very fast and reactive
- Unable to plan ahead
- Unable to learn

Deliberative Systems

- Based on the *sense->plan->act* (SPA) model
- Inherently sequential
- Planning requires search, which is slow
- Search requires a world model
- World models become outdated
- Search and planning takes too long

Hybrid Systems

- Combine the two extremes
  - reactive system on the bottom
  - deliberative system on the top
  - connected by some intermediate layer
- Often called 3-layer systems
- Layers must operate *concurrently*
- Different *representations and time-scales* between the layers
- The best or worst of both worlds?

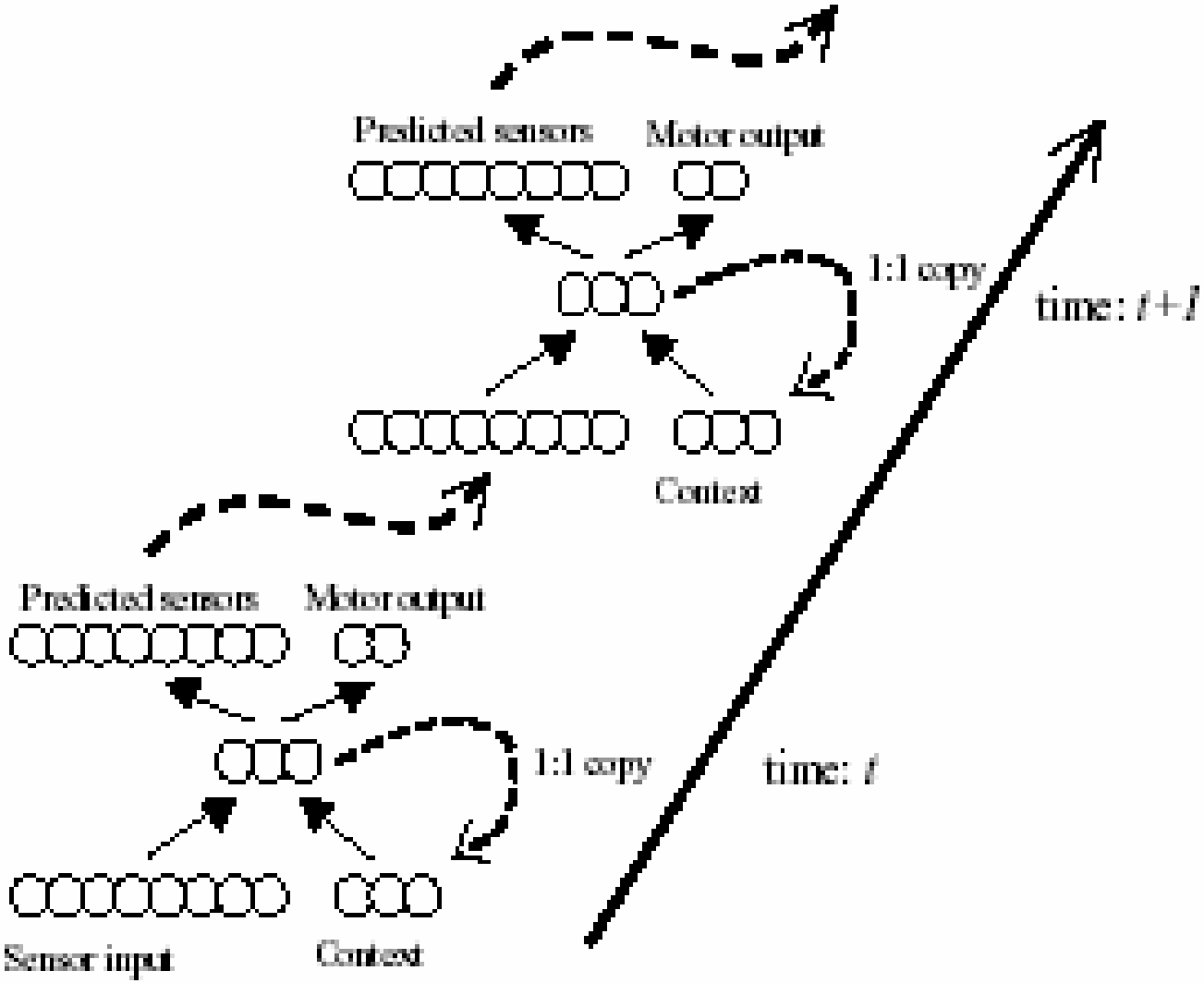
Behavior-Based Systems

- An alternative to hybrid systems
- Have the same capabilities
  - the ability to act reactively
  - the ability to act deliberatively
- There is no intermediate layer
- A unified, consistent representation is used in the whole system=> *concurrent behaviors*
- That resolves issues of time-scale

Tasks for neural networks in robotics

- Recognition of releaser in behavior based robots
- Recognition of objects in vision systems
  - Obstacles
  - Objects for manipulations
  - Gestures
  - Faces and emotions
- Control of manipulator
- Control of gain of legged robots
- In feedback control loop for adaptive control of motors
- Vision-guide motions
- Learning in path planning without mapping
  - Based on prediction
  - Map is coding inside NN as weights of learned NN
- Localization of robot
- In perspective all levels of perception, decision making and control of motions

Robot sensory prediction with SRN (Simple Recurrent Network)



Short Term Memory (STM) & Simple T-maze problems

- Delayed response tasks are a standard way of investigating short-term memory (STM). The agent is typically assumed to '**remember**' **in some way** the necessary information about the stimulus (for example the side on which a stimulus appeared) during the delay period.

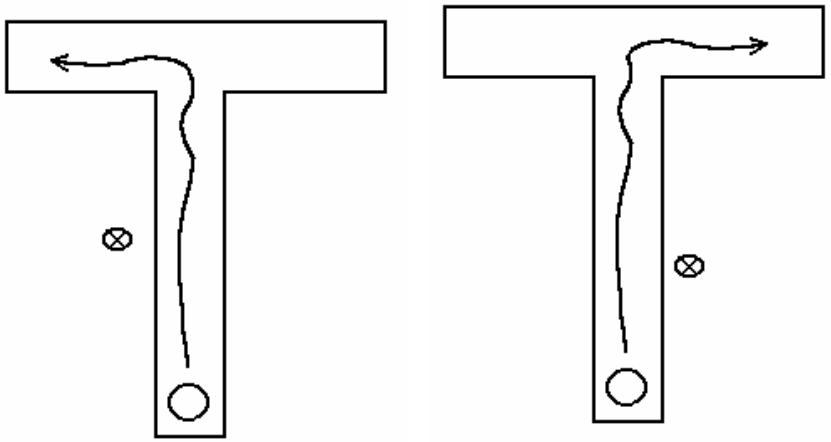
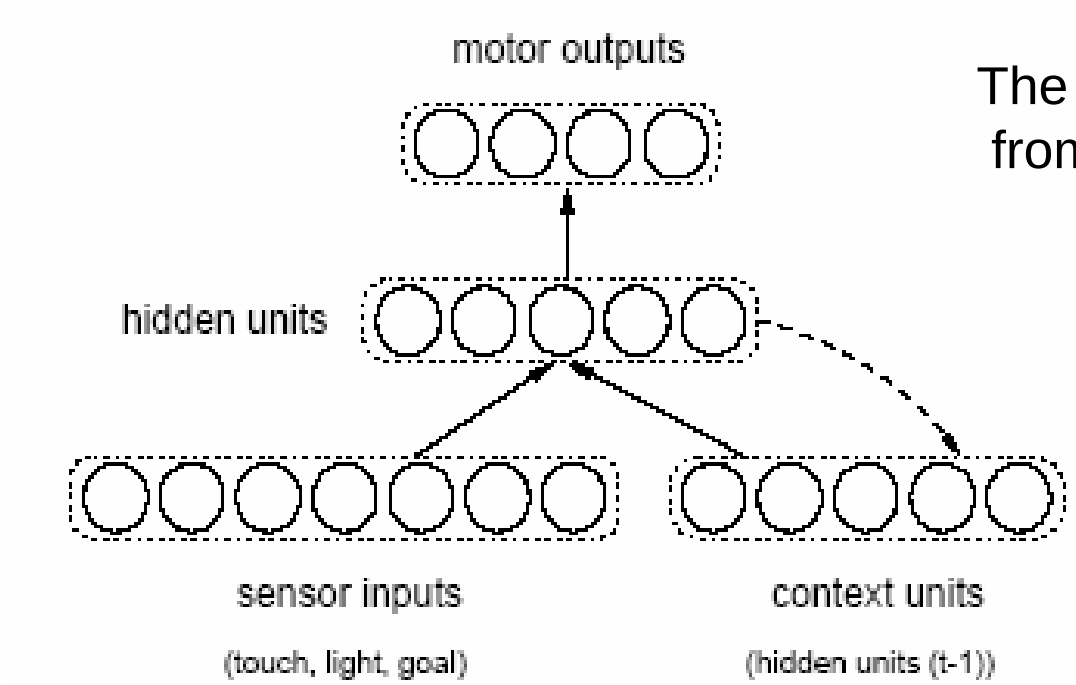


Figure The two situations in the simple T-maze environment. Adapted from Ulbricht (1996).

Meeden's (1996) recurrent robot control architecture.



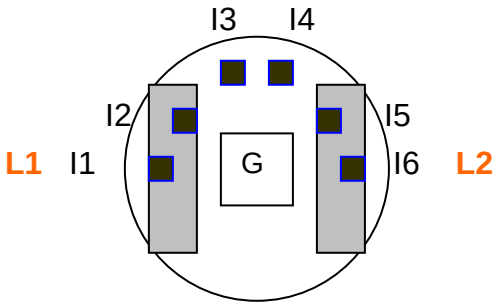
The network's inputs came from light and touch sensors

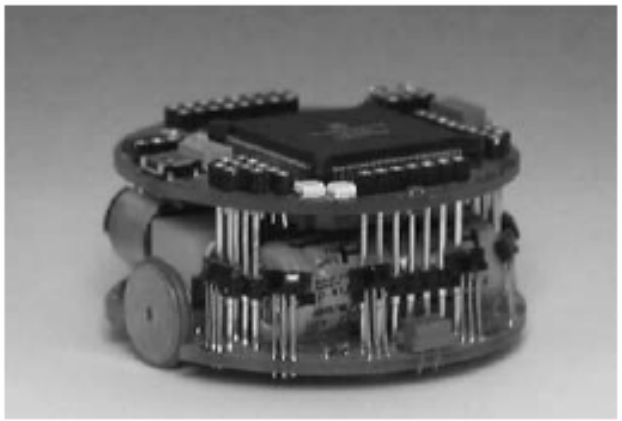
Solid arrows represent fully connected layer of weights between two layers of units (indicated by surrounding dotted lines).
Hidden unit values are fed back via a 1:1 copy connection (dashed arrow) and used as extra inputs in the next time step.

Where is memory what to do in RNN?

- RNNs utilize their internal state (i.e. the hidden unit activation values) to carry out behavioral sequences corresponding to particular motion strategies instead of merely reacting to the current input
- For example, to avoid the light the robot would (starting from a position facing the light) first move backwards for a couple of time steps (into the center of the environment) and then carry out a series of alternating forward right and backward left movements until it faces away from the light
- Thus, it executes a multi-turn strategy in the center of the environment, which overcomes the problem that the environment's smallest dimension is smaller than its own turning radius.

Sensors of the Khepera Robot as Artificial Neural Network Inputs



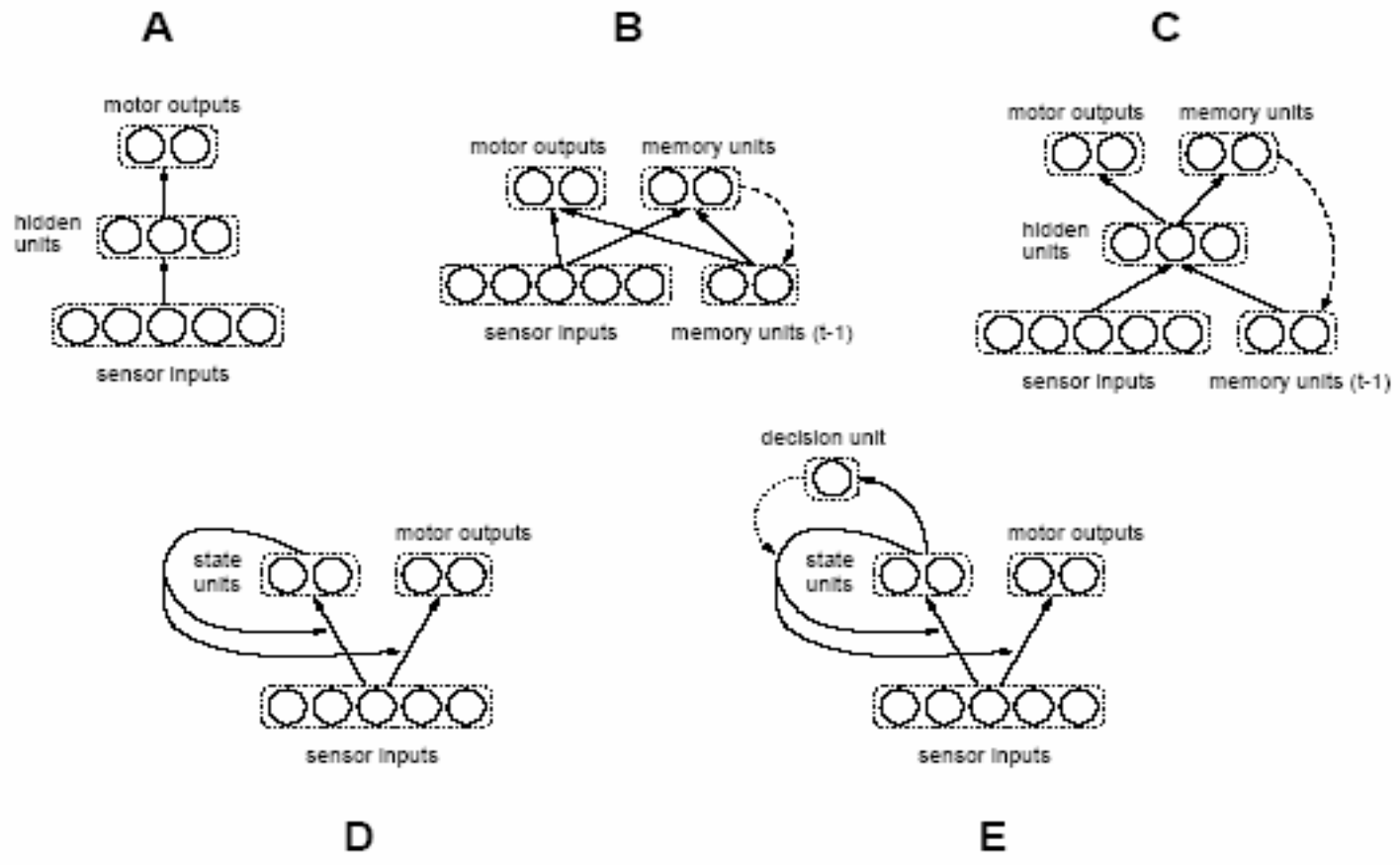


Diameter 5.5 sm

- Sensors of the Khepara robot used in this study:

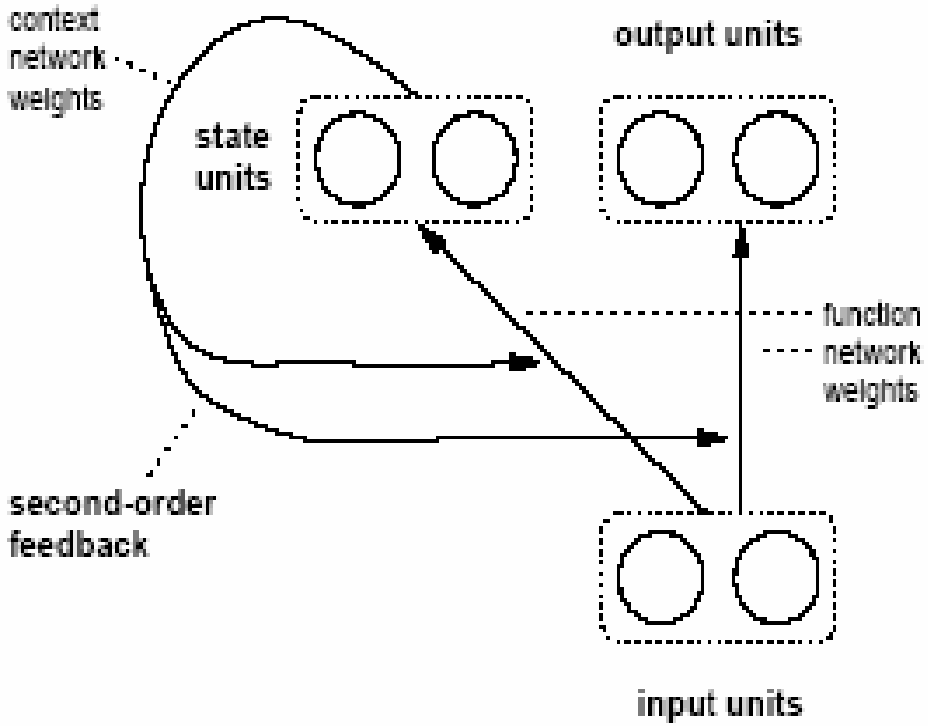
- **I1, I2....I6**: infrared sensors;
- **G**: ground (zone) sensor.
- **L1,L2**: Light sensors.

Kinds of neural-based control system of robot used in experiments (Ziemke, 2000)



All networks receive input from the four proximity sensors and the ground sensor, and produce two outputs directly controlling the left and right wheels' motors

Sequential Cascaded Network (SCN)



A second-order recurrent architecture consisting of (a) a function network mapping input to output and internal state, and (b) a context network mapping the internal state to the next time step's function network weights. The solid arrows represent a fully connected layer of weights between two layers of units.

Environment of experiments

- The robot is placed in a rectangular environment of 1000 mm * 600 mm, surrounded by walls (the straight lines), which contains a zone (the large circle) outside of which it should keep moving while avoiding collisions, whereas once it has entered the zone it should simply not leave it anymore
- The robot is initially placed with a random orientation in a random position outside the zone, and during learning it is punished for collisions and rewarded strongly for every time step it spends in the zone
- Moreover, while outside the zone, it is rewarded for moving as quickly and straight as possible and keeping away from walls
- It uses four infrared proximity sensors at the front and a ground sensor, which is only fully active in the time step when the robot passes the black line marking the zone border
- These five sensors provide input to the controller networks, and the network's two output units directly control the speeds of the two wheels.

Environment of experiments (2)

The large circle indicates the zone the robot should enter and stay in. The small circle represents the robot, and the lines inside the robot indicate position and direction of the infrared proximity sensors used in experiments 1 and 2.

Training of RNN in experiments

- In these experiments all control networks have been trained using an evolutionary algorithm, a genetic algorithm evolving an initially randomized population of 100 individuals over 5000 generations
- For architectures A, B and C the artificial genotype of each individual encodes all the connection weights (including biases) of a complete control network as a single bitstring. Each real-valued weight (between –10.0 and +10.0) is represented by a string of 8 bits.
- For architectures D and E, in which function network weights change dynamically, the genotype encodes the connection weights in the context network plus initial state unit activation values, such that the initial function network weights can be derived by propagating the initial state through the context network.

Training of RNN in experiments (2)

- To evaluate their fitness each individual of every generation is used to control the robot during a trial period of 400 time steps, starting from a random position outside the zone and with a random orientation. While outside the zone individuals score between 0.0 and 1.0 fitness points per time step, being rewarded for moving as fast and as straight as possible while minimizing encounters with walls. While inside the zone, they simply receive 100 fitness points for every time step they remain inside.

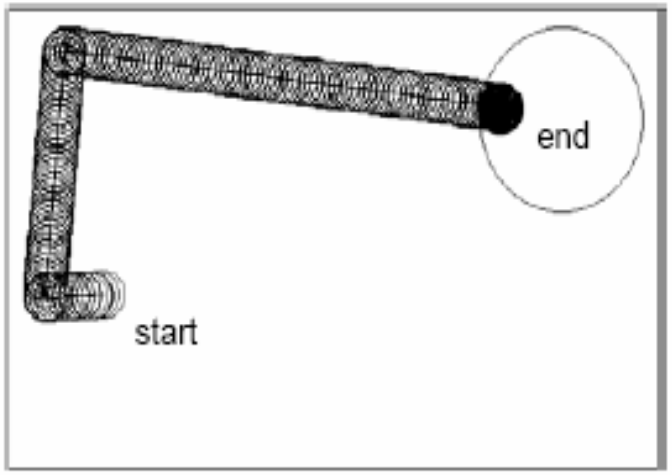
Results

- Networks of architectures B, C, D, and E quickly evolved to robustly solve the task. The evolutionary process was nevertheless continued for 5000 generations to ensure that evolution had converged. Networks of architecture E exhibit best overall performance, although they are not able/allowed to score points while using feedback

Results (2)

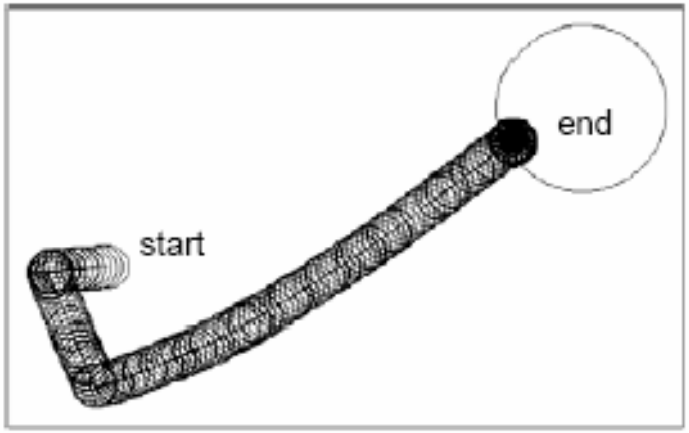
	Best individuals	mean fitness [20 selected]	Mean fitness [whole population]
A	-40.8% (-40.1%)	-51.2% (-49.5%)	-44.5% (-42.0%)
B	-5.5% (-6.6%)	-1.6% (-2.7%)	+6.6% (+3.8%)
C	-7.0% (-10.4%)	-4.4% (-8.9%)	-10.1% (-14.4%)
D	-2.2% (-3.4%)	-2.1% (-4.7%)	-0.7% (-3.2%)

Performance differences between architecture E and other architectures in experiment 1. All differences are stated in percent of the performance of architecture E



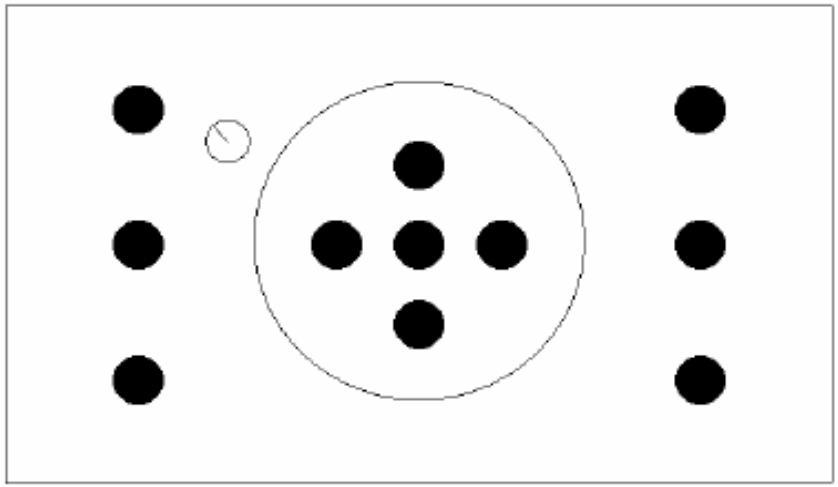
Example trajectory for a robot controller (architecture B) in experiment 1.

Results (3)



Example trajectory for a robot controller (architecture D) in experiment 1.
The robot correctly avoids the walls twice by turning left, then enters the zone and stays there, spinning in place.

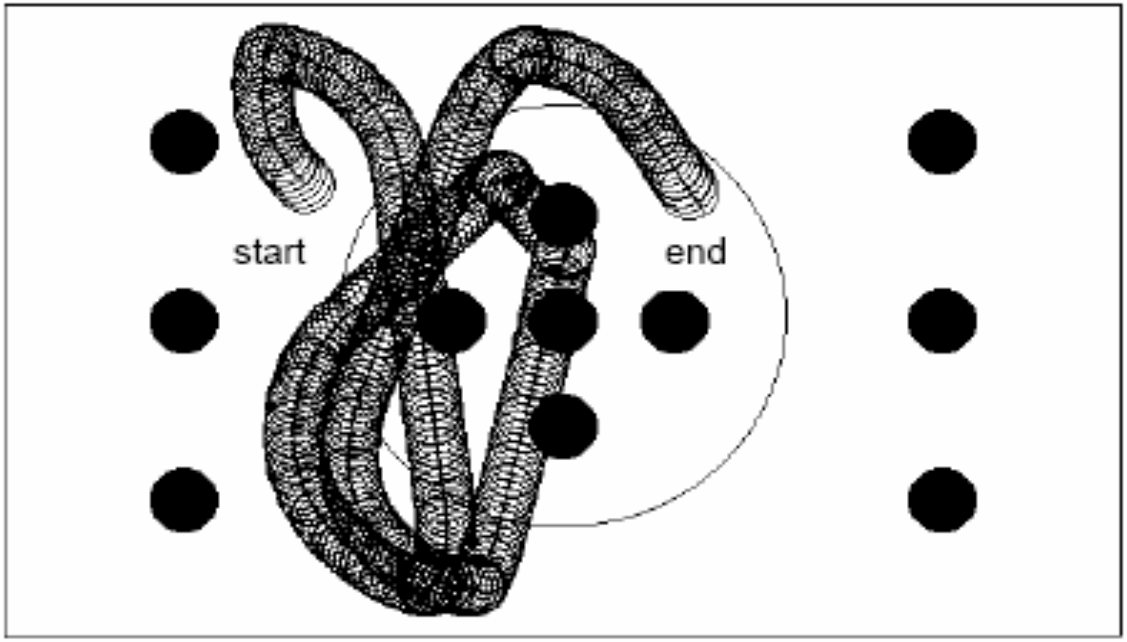
Environment in experiment 2



Simulated Khepera robot in environment 2.
The black circles represent objects, which are to be avoided outside the zone (indicated by the large circle), but to be ‘collected’ inside.

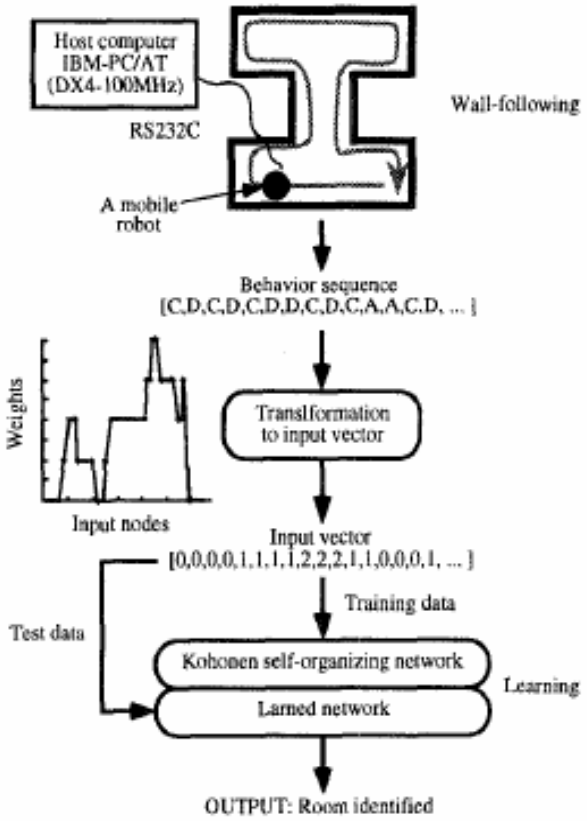
In this experiment the robot receives –500 fitness points for collisions outside the zone (possibly multiple times if persisting to bump into the obstacle) and +500 points for ‘collisions’ inside, but in this case always only once per ‘collected’ object since it disappears immediately. The robot is not punished for leaving the zone, but inside it receives 0.0 to 4.0 fitness points per time step for moving as fast and as straight as possible and approaching objects. Outside the zone, as in experiment 1, the only positive reward the robot can get is 0.0 to 1.0 fitness points per time step for moving as fast and as straight as possible and staying away from objects.

Results



Example trajectory for a robot controller (architecture B) in experiment 2. The robot successfully collects four objects in the zone, and correctly avoids objects and walls outside the zone.

Recognition of room by Kohonen self-organizing map (Seiji Yamada and Morimichi Murota, 1996)



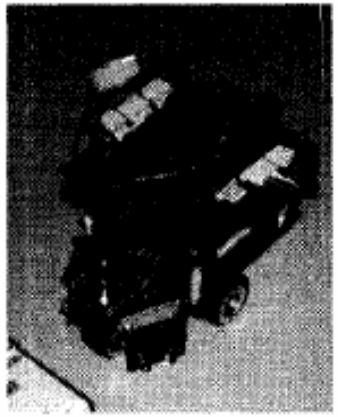


Fig. 2: A mobile robot

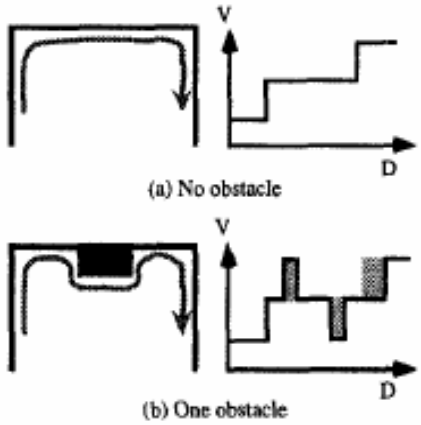


Fig. 3: Influence of an obstacle

Behaviors

- **Behavior** A *(turning in the concave corner): **if*** any obstacle within 10 cm in the front and within 10 cm
- **Behavior** B *(turning in the convex corner): if* no obstacle within ***5*** cm in the left and the right, and within
- **Behavior** C *(wall-following-1): **if*** any obstacle within ***5*** cm in the left *then* steering 13.5"right.
- **Behavior** D *(wall-following-2): **if*** no obstacle within ***5*** cm in the left *then* steering 13.5"left.

Transformation of sequence of behavior to input vector

BI-transformation

(1) If r_i = A then $v_i = v_{i-1} + 1$.

(2) If r_i = B then $v_i = v_{i-1} - 1$.

(3) If r_i = C or D then $v_i = v_{i-1}$.

(4) otherwise $v_i = 0$ $(i > n)$.

Rooms

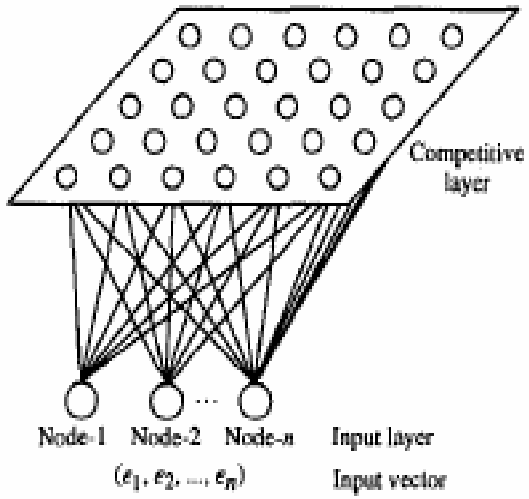
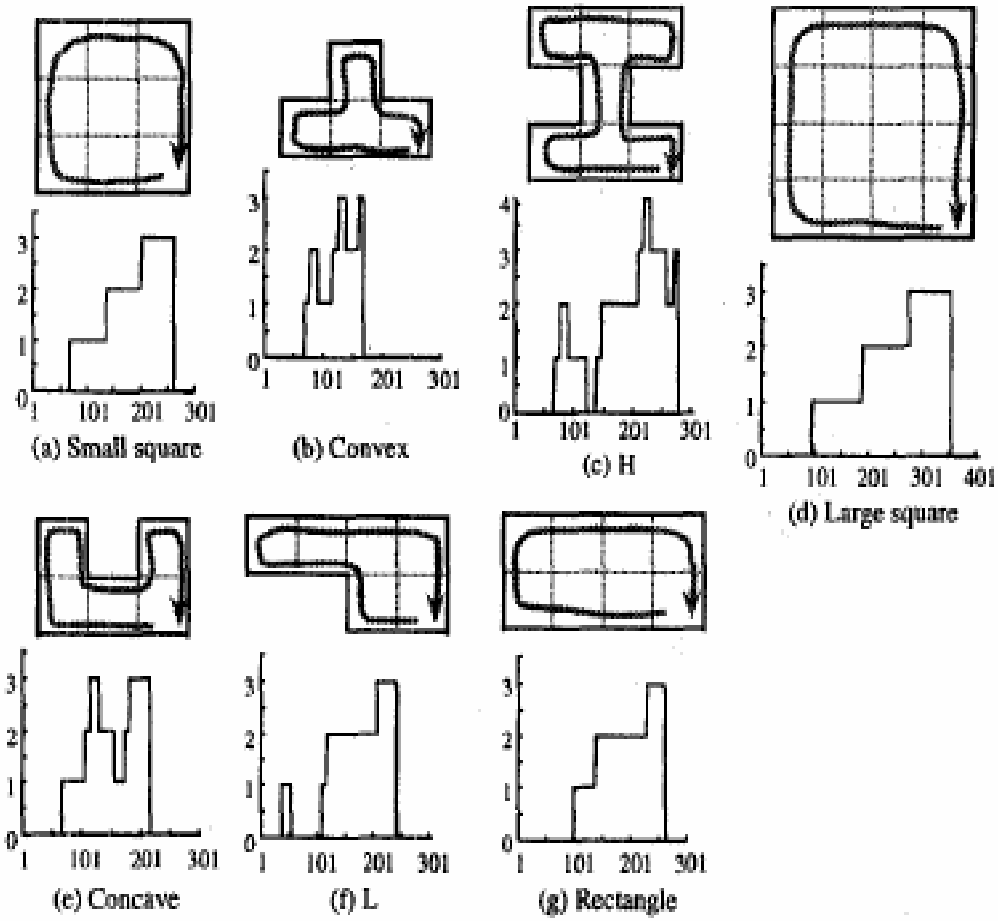


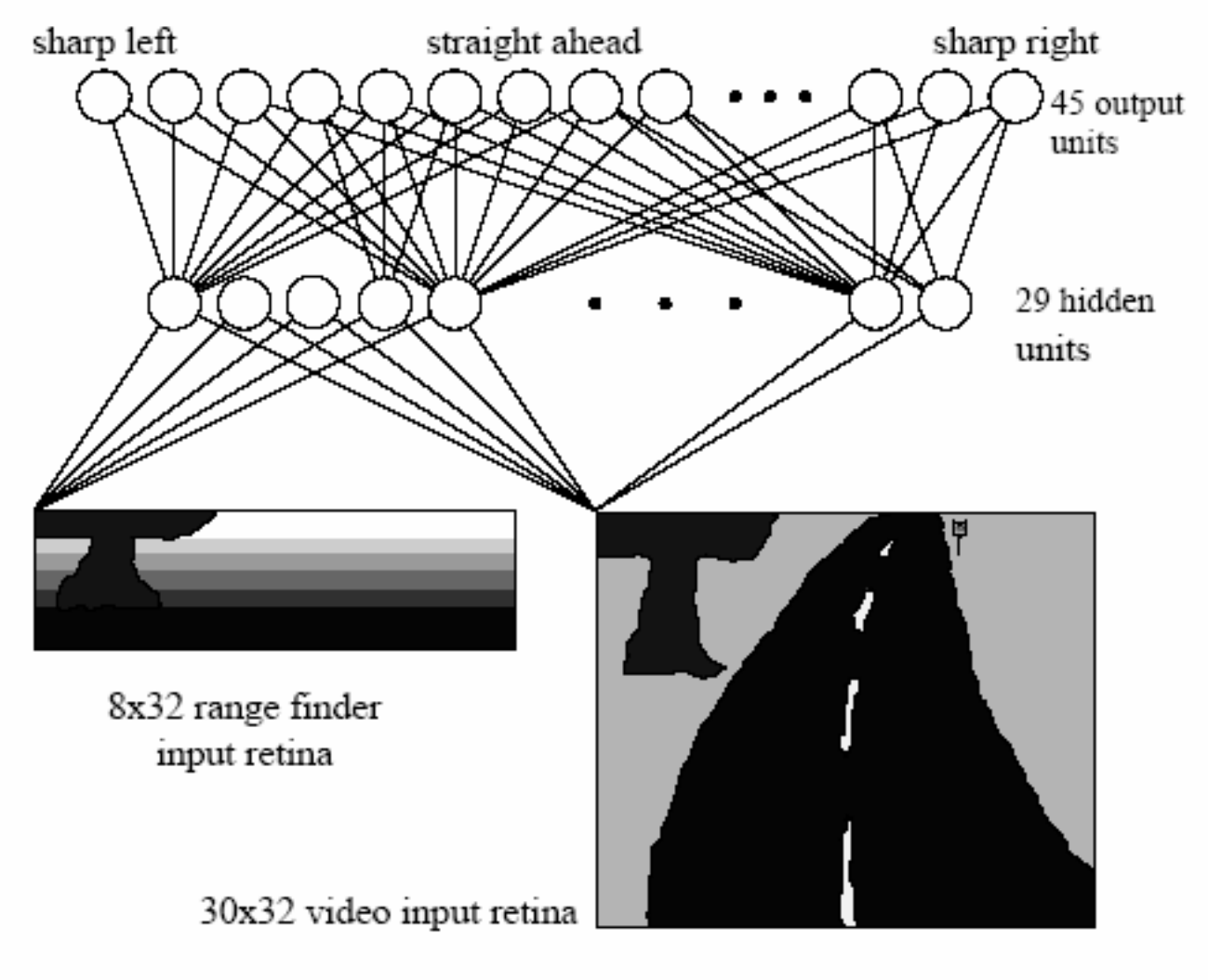
Fig. 4: A self-organizing network



Experiments with real robot

- **Exp-1: (identifying the rooms)** After learning with training data, the test data **(63** input vectors) were given to the learned self-organizing network. As a result, all the test data were correctly identified, and verified the utility of our approach. Note that the 10 input vectors for the identical room were somewhat different mutually because the wall-following included noise like failure of executing behaviors.
- **Exp-2: (the rooms with obstacles)** Though the test data used in Exp-1 included noise, it was not so much. In this experiment, they dealt with more noise like obstacles. They located some obstacles in the rooms, and the mobile robot did wall-following in the rooms. The nine behavior sequences were obtained and transformed into input vectors. The input vectors (test data) were given to the self-organizing network which was trained in Exp-1. As a result, five data were correctly identified. The robot failed to recognize a room in which several obstacles scattered. However the obstacles made the room the different shape from the original one, thus we consider the failure of identification is natural.

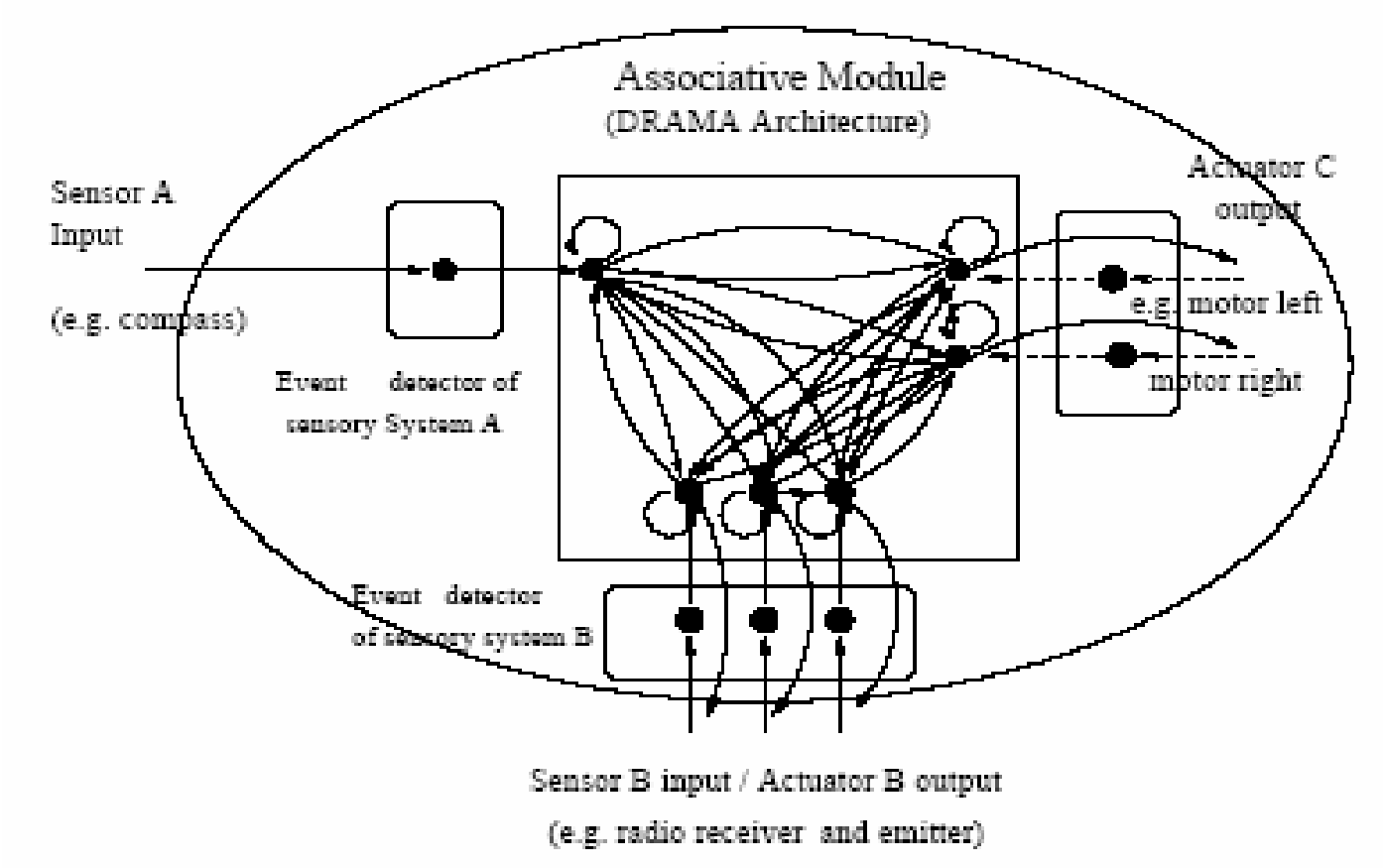
The structure of the network for the autonomous land vehicle



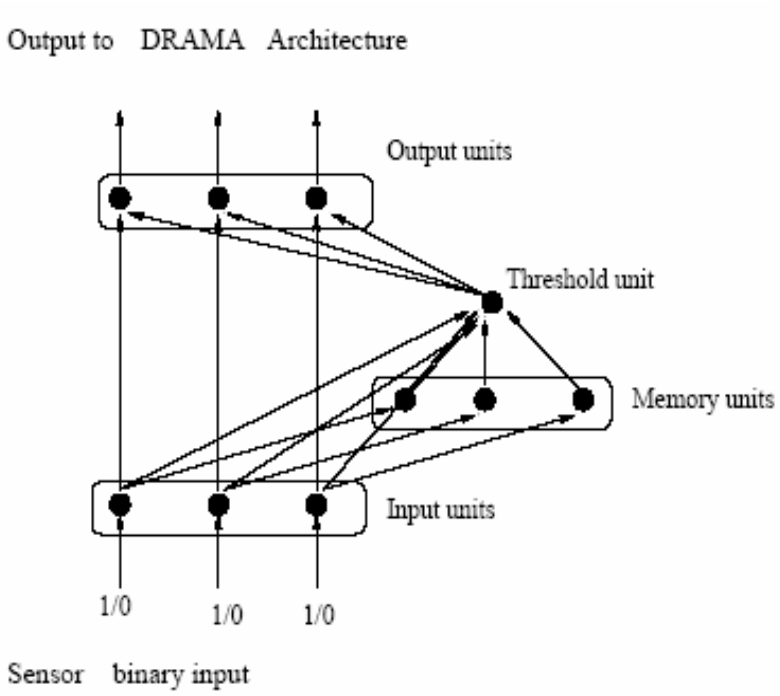
Experiments

The network was trained by presenting it samples with as inputs a wide variety of road images taken under different viewing angles and lighting conditions. 1200 Images were presented, 40 times each while the weights were adjusted using the backpropagation principle The authors claim that once the network is trained the vehicle can accurately drive at about km/hour along '… a path though a wooded area adjoining the Carnegie Mellon campus under a variety of weather and lighting conditions.' The speed is nearly twice as high as a non-neural algorithm running on the same vehicle.

Drama (A.Billard, J.Hayes, 1999)



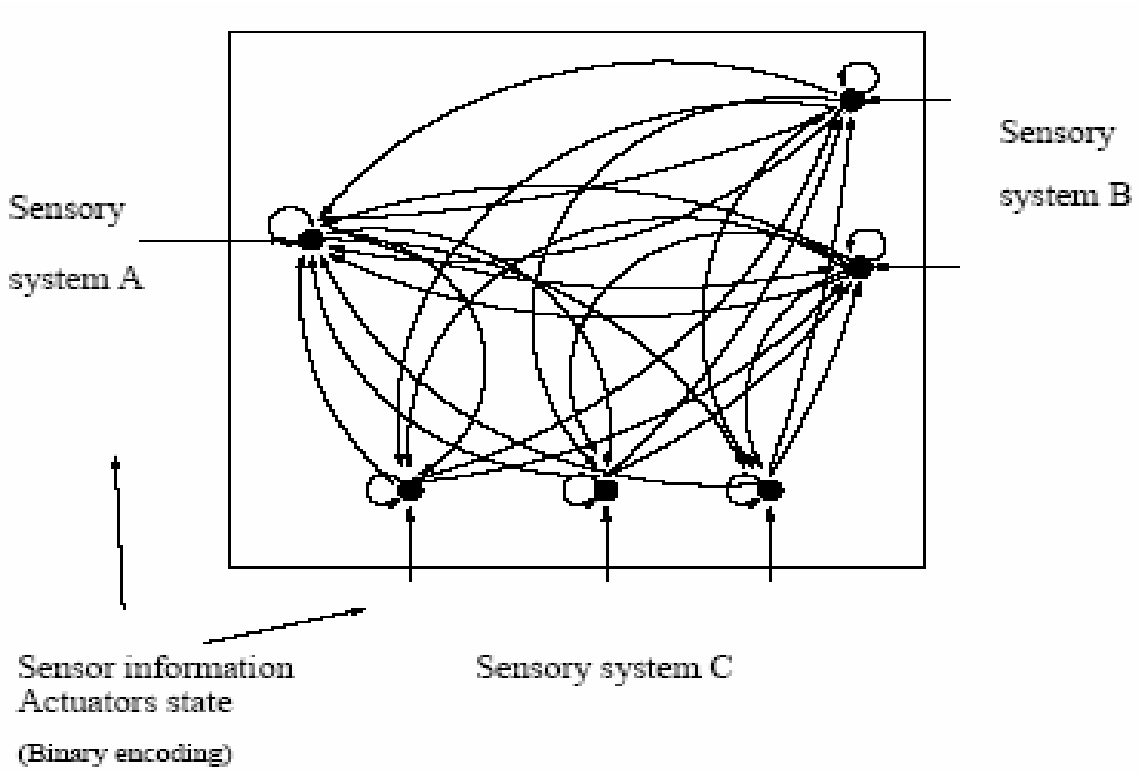
DRAMA (2)



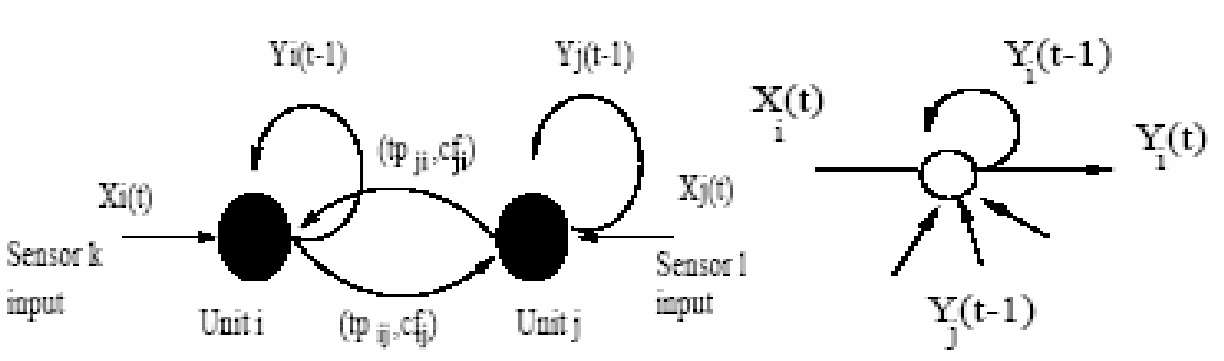
There is one *event detector* module per sensor. Each module receives n input units and outputs to n associated units in the DRAMA architecture, where n is the number of units of the particular sensor (see Figure 2). The neuronal representation of the internal structure of the module is given in Figure 2. Each input unit is connected to one *memory* unit, one *output* unit and to a *threshold* unit. Output $y_i^m(t)$ of the memory unit i at time t is simply the value x_i of the input unit i at time t-1, $y_i^m(t) = x_i(t-1)$. Output y_i^{th} of the threshold unit is the result of the function $\theta(x, H)$ applied onto the difference between the input units and memory units outputs: $y_i^{th}(t) = \theta\left(\sum_{i=1}^{n} \left| x_i(t) - y_i^m(t) \right|, H\right)$, where the function $\theta(x, H)$ is a threshold function that outputs 1 when x >= H. Finally, the state of the output unit $y_i(t)$ is calculated as follows:

$$y_i = \theta\left(x_i(t) + y_i^{th}(t), 2\right) = \theta\left(x_i(t) + \theta\left(\sum_{i=1}^{n} \left| x_i(t) - x_i(t-1) \right|, H\right), 2\right) \quad (1)$$

DRAMA (3). Associative module



DRAMA (4)



$$y_i(t) = F\left(x_i(t) + tp_{ii} \cdot y_i(t-1) + \sum_{j \neq 1} G(tp_{ji}, cf_{ji}, y_j(t-1)) \right) \quad (2)$$

where F, the transfer function, is the identity function for input value less than 1 and saturates to 1 for value greater than 1, $F(x) = x$ if $x \leq 1$, otherwise $F(x) = 1$,

and G is the retrieving function whose equation is given below in Equation 3 and explained in the following paragraph. The indices notation used in the equations should be interpreted as follows: cf_{ji} is the confidence factor of the connection leading from unit j to unit i.

DRAMA (5)

$$G(tp_{ji}, cf_{ji}, y_j(t-1)) = A(tp_{ji}) \cdot B(cf_{ji})$$

$$A(tp_{ji}) = 1 - \theta(|y_j(t-1) - tp_{ji}|, e)$$

$$B(cf_{ji}) = \theta\left(cf_{ji}, \frac{\max_{yi>0}(cf_{yi})}{T}\right)$$

where $\max_k(cf_{ji})$ is the maximal value of confidence factor of all the connections between activated units j and unit i, which satisfy the temporal condition encoded in $A(tp_{ji})$. The function $\theta(x, H)$ is a threshold function that outputs 1 when $x >= H$. The output of function G is equal to 1 when both A and B terms are equal to 1, otherwise it is zero. The temporal and spatial conditions represented by the A and B terms can be paraphrased as follows: 1) $A(tp_{ji}) = 1$ if the time delay for which the activation of unit j has been memorized before being correlated to the activation of unit i (this time delay is encoded in the value of $y_j(t)$, which decreases linearly with time when no new activation occurs, see *short term memory* paragraph) is equal to the time encoded in the time parameter tp_{ji} within an interval error e. 2) $B(cf_{ji}) = 1$ if the confidence factors cf_{ji} associated with the connection between one activated unit j and unit i, which satisfies the condition $A(tp_{ji}) = 1$, is greater than or equal to 1/T times the maximum confidence factor of other activated connections, $\max_{yi>0}(cf_{yi})$. The effect of the two terms $A(tp_{ji})$ and $B(cf_{ji})$, and in particular of the threshold T and e, on the memory capacity will be discussed further in Sections 3.1 and 5.3 and an algorithm for calculating the parameters T and e on-line will be presented in Section 3.1.

DRAMA (6)

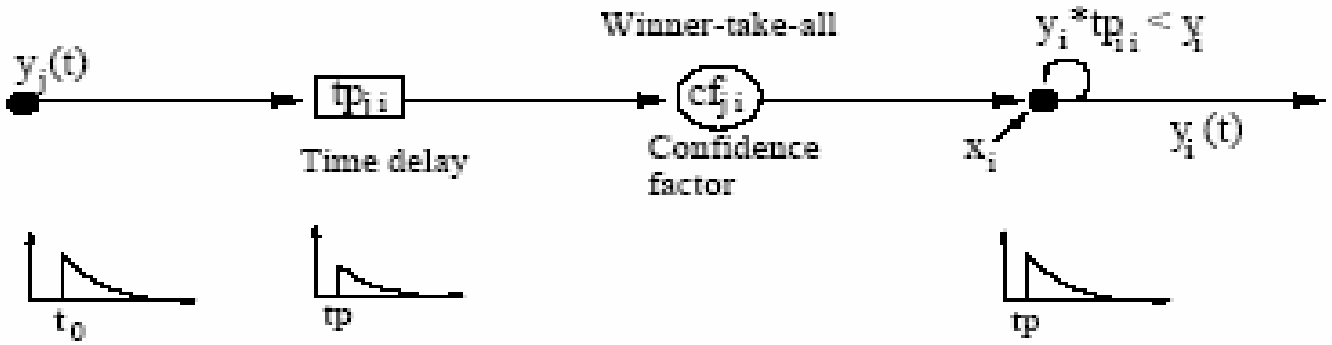


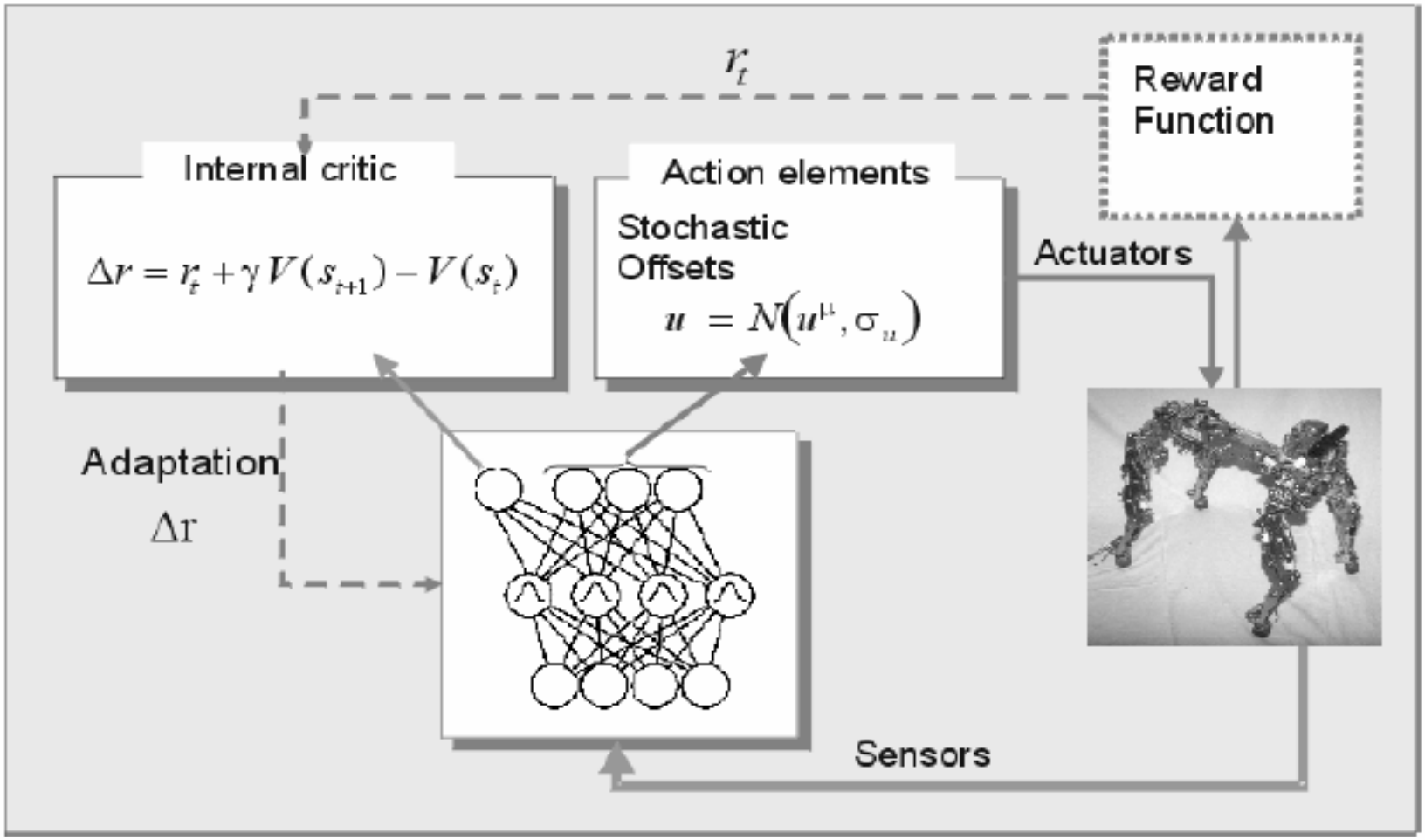
Figure 5. Propagation of the unit activity along the network connections.

$$tp_{ji}(t) = \frac{tp_{ji}(t-1) \cdot \frac{cf_{ji}}{a} + \frac{y_j(t)}{y_i(t)}}{\frac{cf_{ji}}{a} + 1}$$

Learning

$$cf_{ji}(t) = cf_{ji}(t-1) + a$$

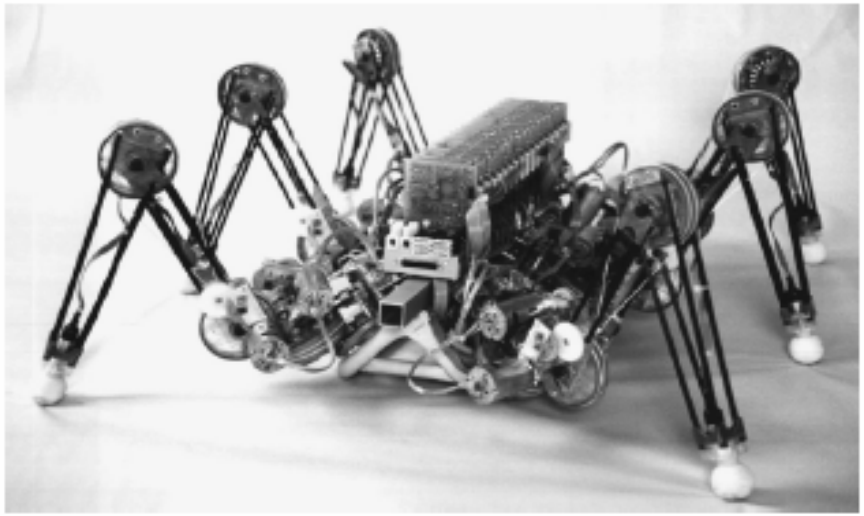
Adaptive movement control of a four-legged walking machine

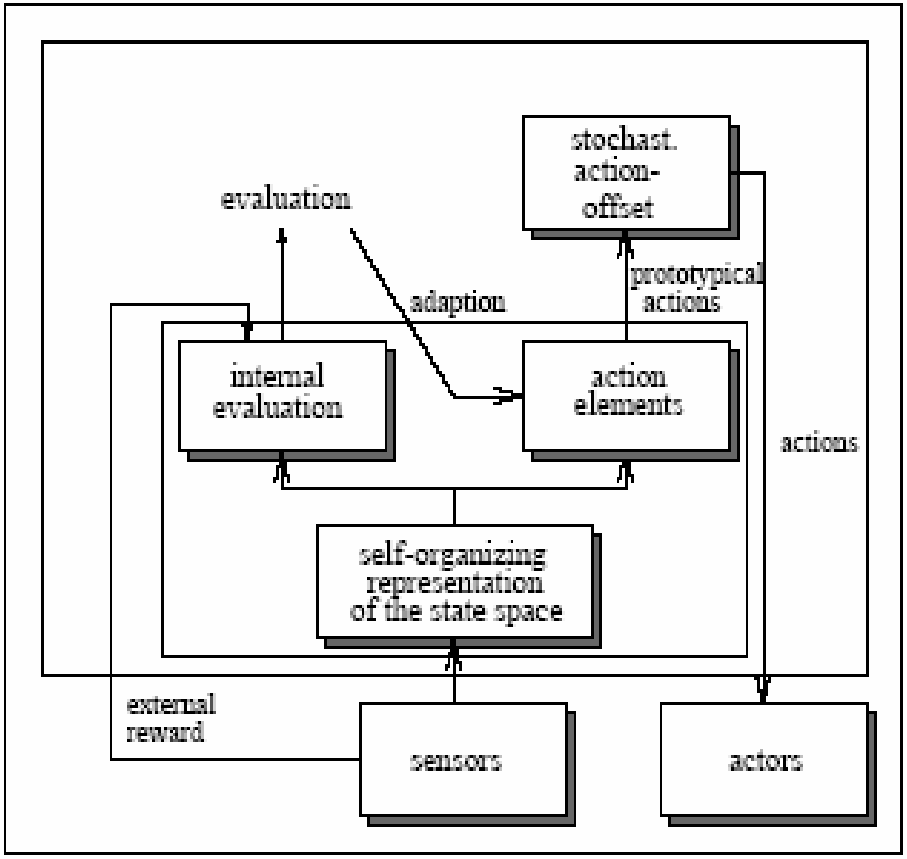


Adaptive movement control of a four-legged walking machine (2)

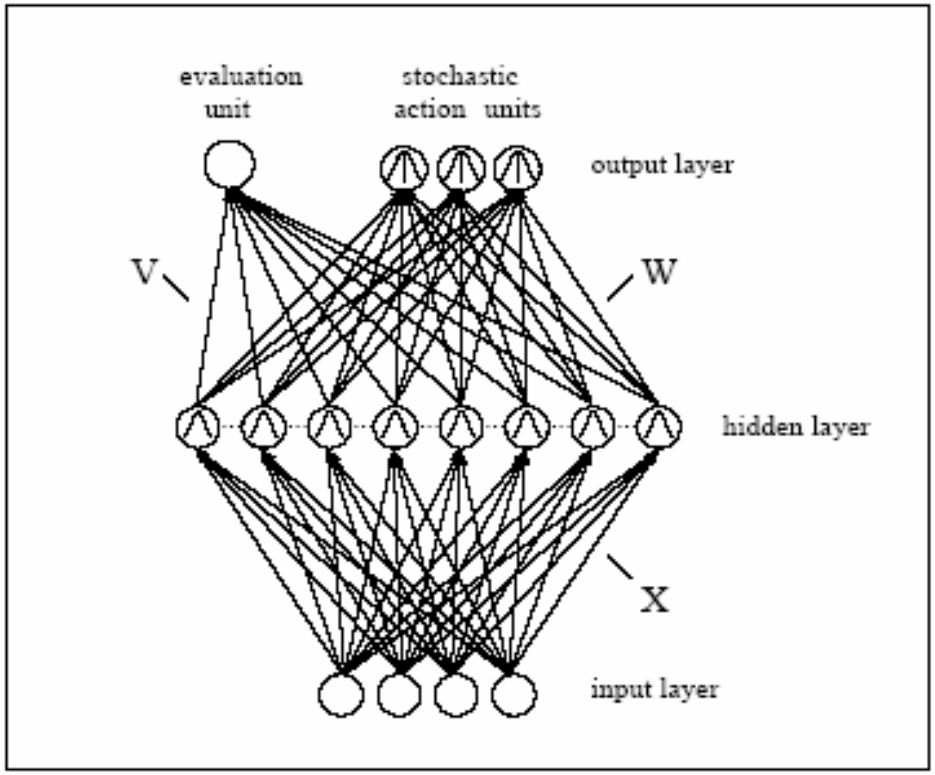
- The network generates the internal evaluation and prototypical actions.
- For exploration purposes, stochastic offsets are added to these actions. The stochastic offsets are generated using a normal distribution. The variance of this distribution is determined by the current performance of the net.
- The executed action sequence causes an external reward. The adaptation of the internal evaluation and the action units are based on the successive external and internal evaluations.

Adaptive control of 6-legged walking machine

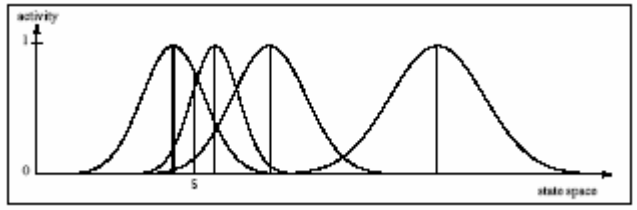




Adaptive control of 6-legged walking machine (2)



A RBF-network for realization of the self-organizing state representation in the hidden layer and the internal evaluation and the action elements of the output layer.
Vectors X is bases of Gaussian activation functions.
W – matrix of connections between “exemplars” and action stochastic units.
Matrix V for connections of exemplars with evaluation unit.



Receptive fields

Adaptive control of 6-legged walking machine (3)

- Inputs – 3 Euclidean coordinates of foot, ground contact sensor.
- Generated outputs – angle offsets for 3 leg joints.
- External reward is calculated as

$$r_{extern} = r_{max} - \| \vec{pos}^{end} - \vec{pos}^{target} \|$$

- Maximal award is returned if the target position is exactly reached

Localization (A.Dubrawsky, 1996)

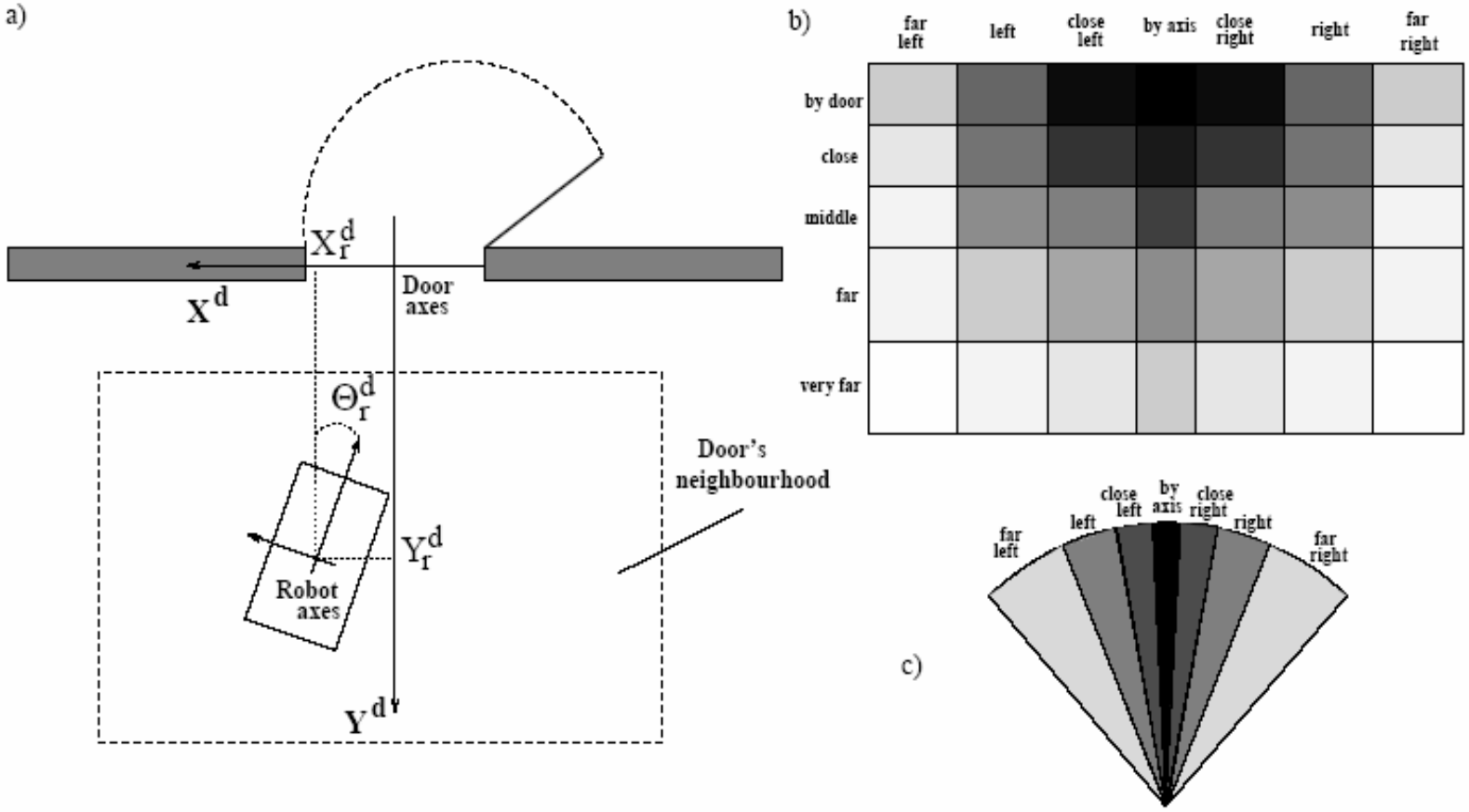


Figure 3. Robot in front of the door: a) metric description; b) topological categorization of the position coordinates; c) topological categorization of the orientation angle.

Localization (2)

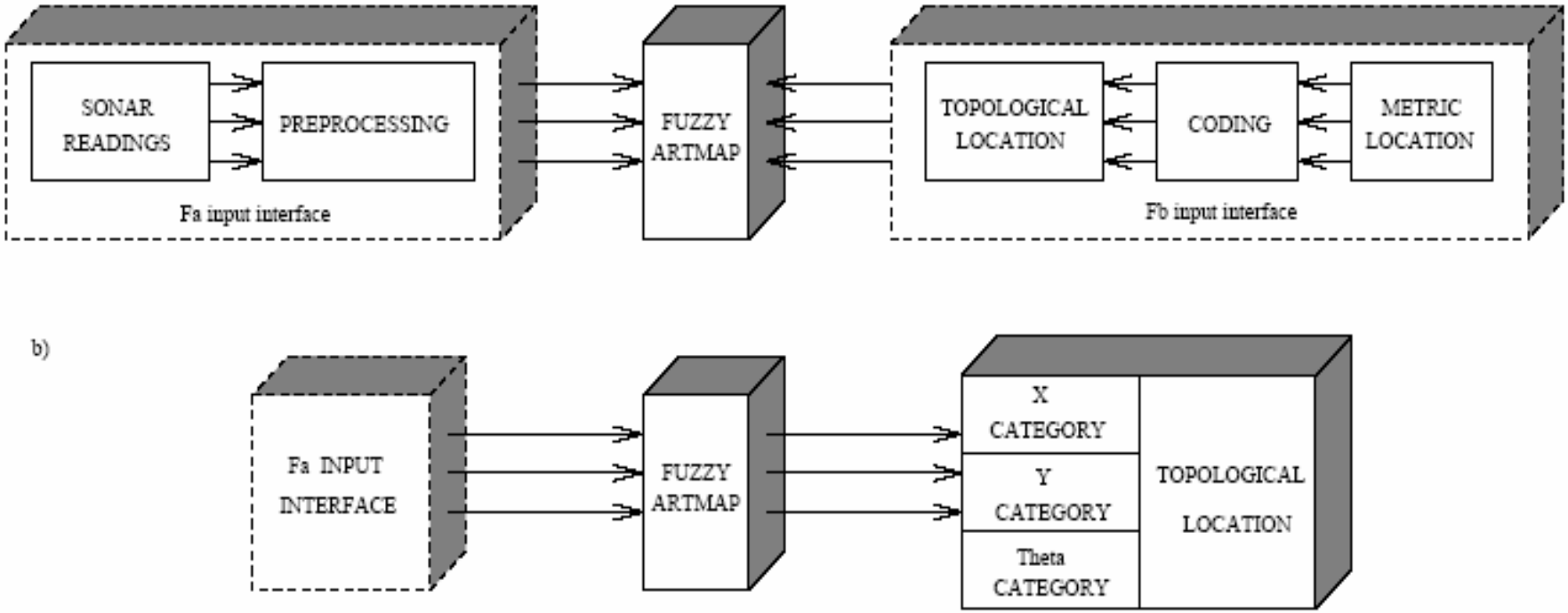


Figure 4. Localization association process: a) supervised learning of the mapping between the environment space and the topological location space; b) performance mode: the system predicts relative location, processing a set of the selected ultrasonic range sensors readouts.

Localization (3)

Then, during the performance mode, the network is able to retrieve the location of the robot within a frame of the door neighborhood region, as long as the robot is actually placed somewhere in this predefined region. The retrieved data has rather a qualitative meaning, due to the topological categorization of the position and orientation coordinates. It bears however enough information to be useful for a door passing locomotion task.