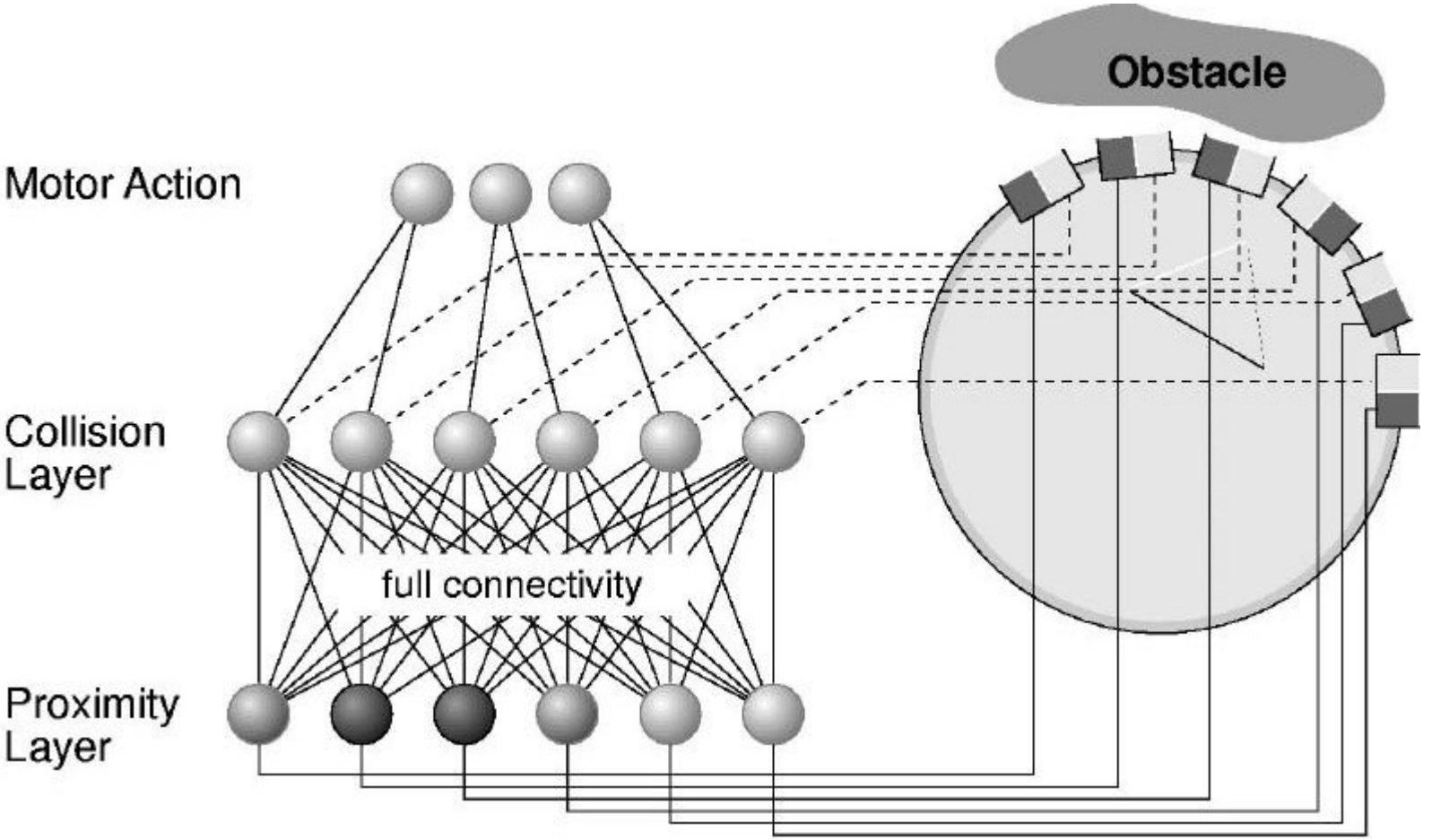
# Robot Behaviour - Lernen

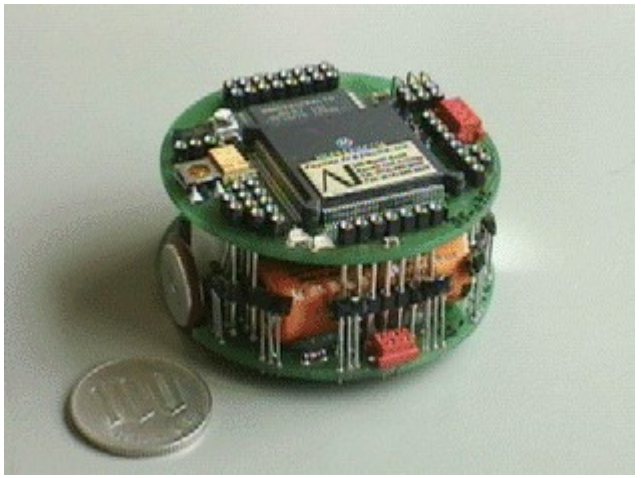
# DAC - Distributed Adaptive Control

[eine genaue Beschreibung ist im Buch „Understanding Intelligence“ zu finden]




[Pfeifer, R. et al. (1992)]

# DAC



The original version was implemented on a real Khepera robot.

The DAC Control Architecture is a neural network with three layers:

- proximity layer,
- collision layer
- motor action layer

Each proximity sensor is connected to a node in the proximity layer. Each collision sensor is connected to a node in the Collision layer. The nodes of the proximity layer have continuous sigmoid activation functions and are connected uni-directionally to the nodes of the collision layer. The nodes of the collision layer have binary activation and have **hardwired** connections to the motor layer.

Using **Hebbian learning** as the robot runs, the network will strengthen the connections between the collision layer nodes and proximity layer nodes which are active at the same time. As the network trains, the robot will learn to avoid obstacles as the proximity sensors will activate the collision layer nodes and in effect "**predict**" collisions before they can occur, so the robot can turn away.

# Features

- hard-wired reflexes
- forgetting
- development of robot behaviour over time (draw)
- target detectors (light sensor)
- no obstacles, no light -> move forward

# Pseudo Code

**Main Control Loop:**

1. For each node in PROXIMITY layer:

   Proximity-Node-Input:
   IR-SENSOR value

   Proximity-Node-Output:
   If Proximity-Node-Input > 1, output 1
   If Proximity-Node_input < 0, output 0
   else output Proximity-Node-Input

2. For each node in COLLISION layer:

   Collision-Node-Input:
   Proximity-node-output * Connection Weight
   to collision node (for all proximity-nodes)
   + COLLISION-SENSOR value

   Collision-Node-Output:
   if Collision-Node-Input > 0.3, output 1
   else output 0

3. For each node in MOTOR ACTION layer:

   Motor-Node-Input:
   Collision-Node-Output * Connection weight to motor node (for all collision-nodes)
   + DEFAULT-MOTOR value

   Motor-Node-Output:
   output Normalized Motor-Node-Input --> set MOTOR SPEEDs

4. HEBBIAN LEARNING:

   For each connection from PROXIMITY to COLLISION layer:

      Learning:
      If the connected Proximity-Node and Collision-Node are activated at the same time:
      Increase connection weight (proportional to their levels of activation)

      Forgetting:
      Decrease connection weight (proportional to average activation in collision layer)

# Pavlovsches Lernen