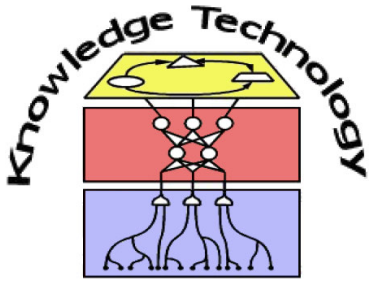
Bio-Inspired Artificial Intelligence

Lecture 7: Behaviour Based Robotics



http://www.informatik.uni-hamburg.de/WTM/

Outline

- Classic AI Approach to robotics, functional decomposition
- Braitenberg Vehicles
- Subsumption Architecture
- Motor Schemas

Shakey

“Classic” Robotics

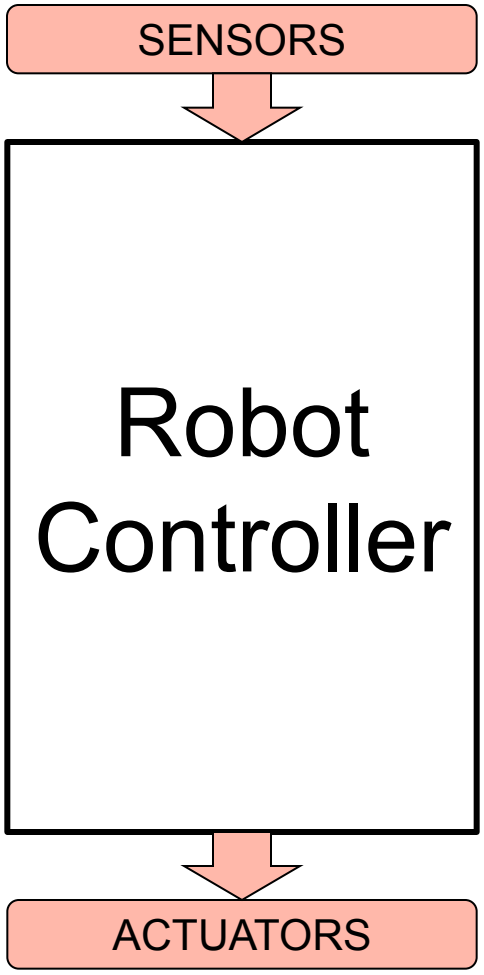
- Shakey, the robot (1966-1972)
  - Operated in world defined by rooms, corridors and objects
  - Tasks: Route finding, planning, rearranging objects
  - Command was e.g. “push the block off the platform”
  - Top down AI approach
    - STRIPS planning system, A* Search
- Move → Think → Next Move

Functional Decomposition

- **Classical Approach:**
  Top-Down Decomposition into logical units

  1. Processing sensory data
  2. Creation of world model
  3. Planning
  4. Plan execution

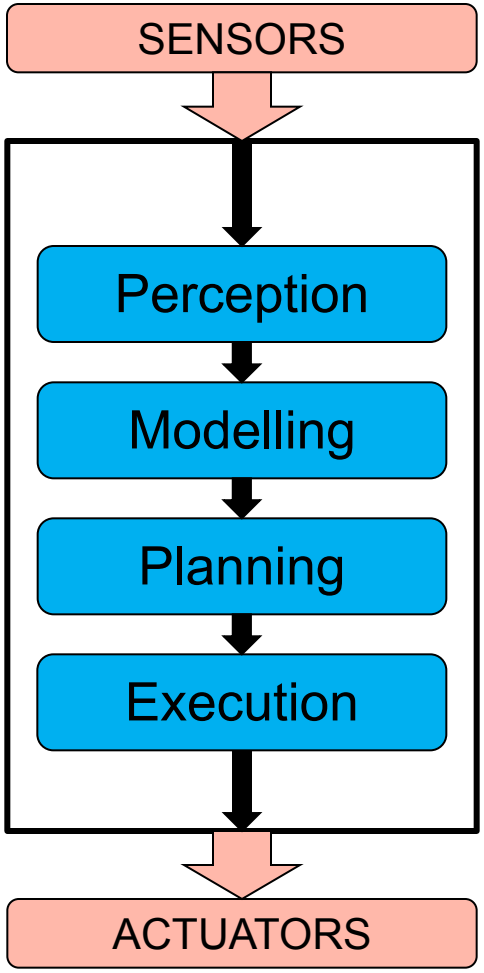
- Multiple variations of this schema (e.g. partial planning, heuristics, etc)



Functional Decomposition

- **Advantages**:
  - system is precise
  - controllable
  - predictable

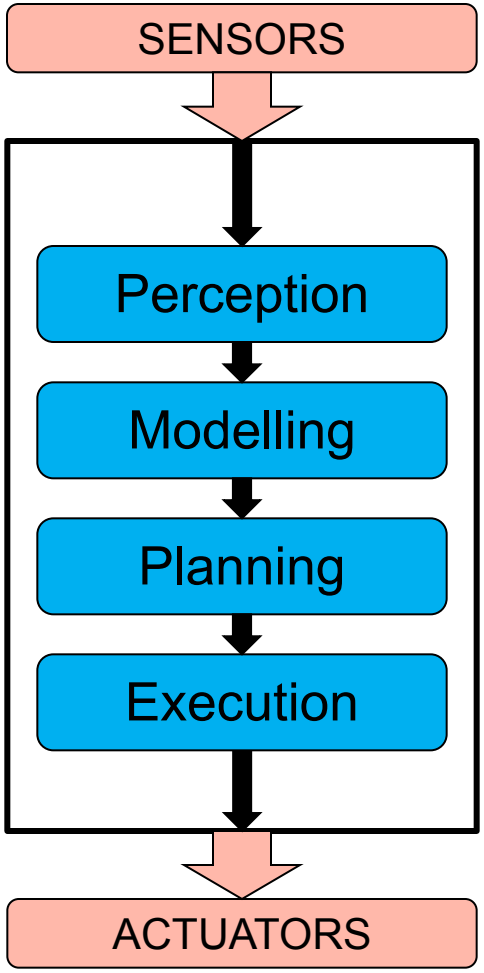
- **Disadvantages**:
  - difficult to handle noise and uncertainty
  - unit failure $\Rightarrow$ system failure
  - computationally expensive
  - granularity problem (model)



Behaviour-Based Architecture

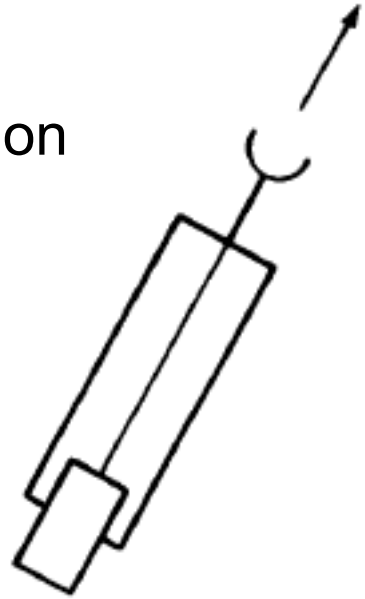
- Requirements for intelligent autonomous robot platform:
  - Achieving multiple, conflicting goals
  - Handle multiple, noisy sensors
  - Robustness
  - Additivity, without impairing previous function

➔ Classic approach not optimal when dealing with noise and dynamic environments.



Braitenberg Vehicles

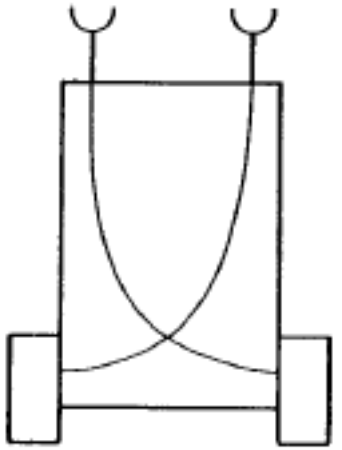
- Thought-experiment by Valentino Braitenberg (1984)
- Braitenberg vehicle is simple agent with
  - Sensor(s): measure stimulus at given point
  - Actuator(s): Transform input value into motion



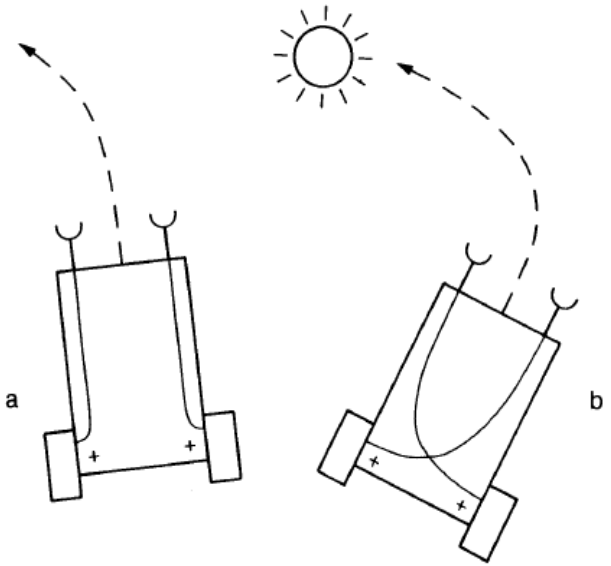
- Simplest vehicle:
  - Light sensor connected to one actuator
  - Produces forward motion
  - Faster when light is brighter
  - Stops in darkness

[Braitenberg, V. (1984). “Vehicles. Experiments in Synthetic Psychology”. MIT Press, MA.]

Braitenberg Vehicles

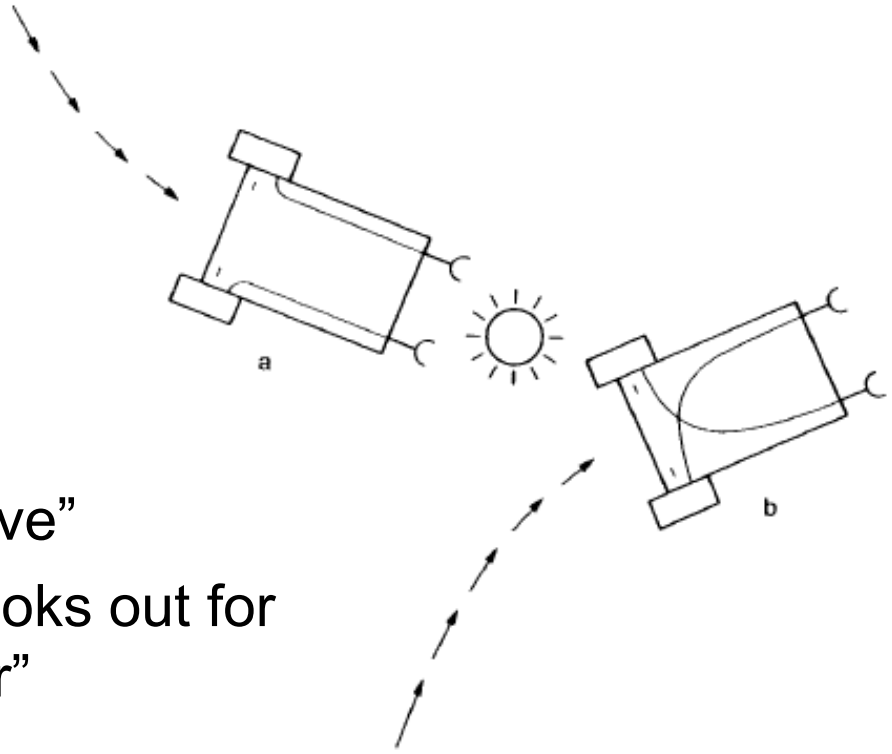


- One agent can have many sensors and actuators with multiple connections
- Agent “behaves” depending on connection scheme
- Behaviours of agents can be observed and labelled
- “Fear”: Agent a turns wheel on side of light faster, therefore turning away. Slows down with increasing distance
- “Aggression”: b turns towards light and speeds up with decreasing distance, hitting the light full speed



Braitenberg Vehicles

- Connection can also be inhibitory, i.e. the stronger the sensor value, the smaller the input to the actuator (note the "-" or "+")



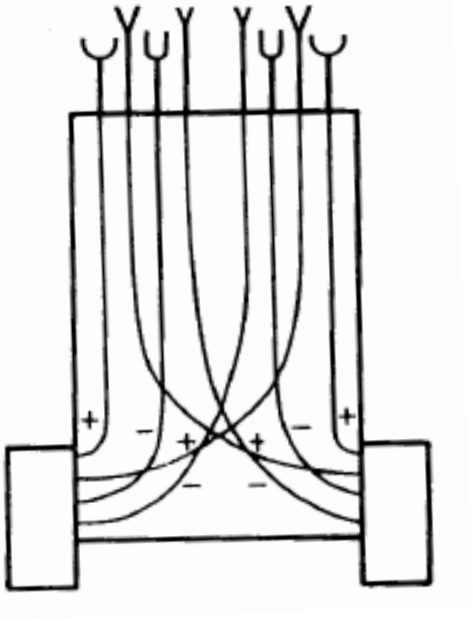
- Example: How would you describe the behaviour?

  Both "like" the source

  a) "Admires" the source: "Love"
  b) Wants to stay close but looks out for stronger source: "Explorer"

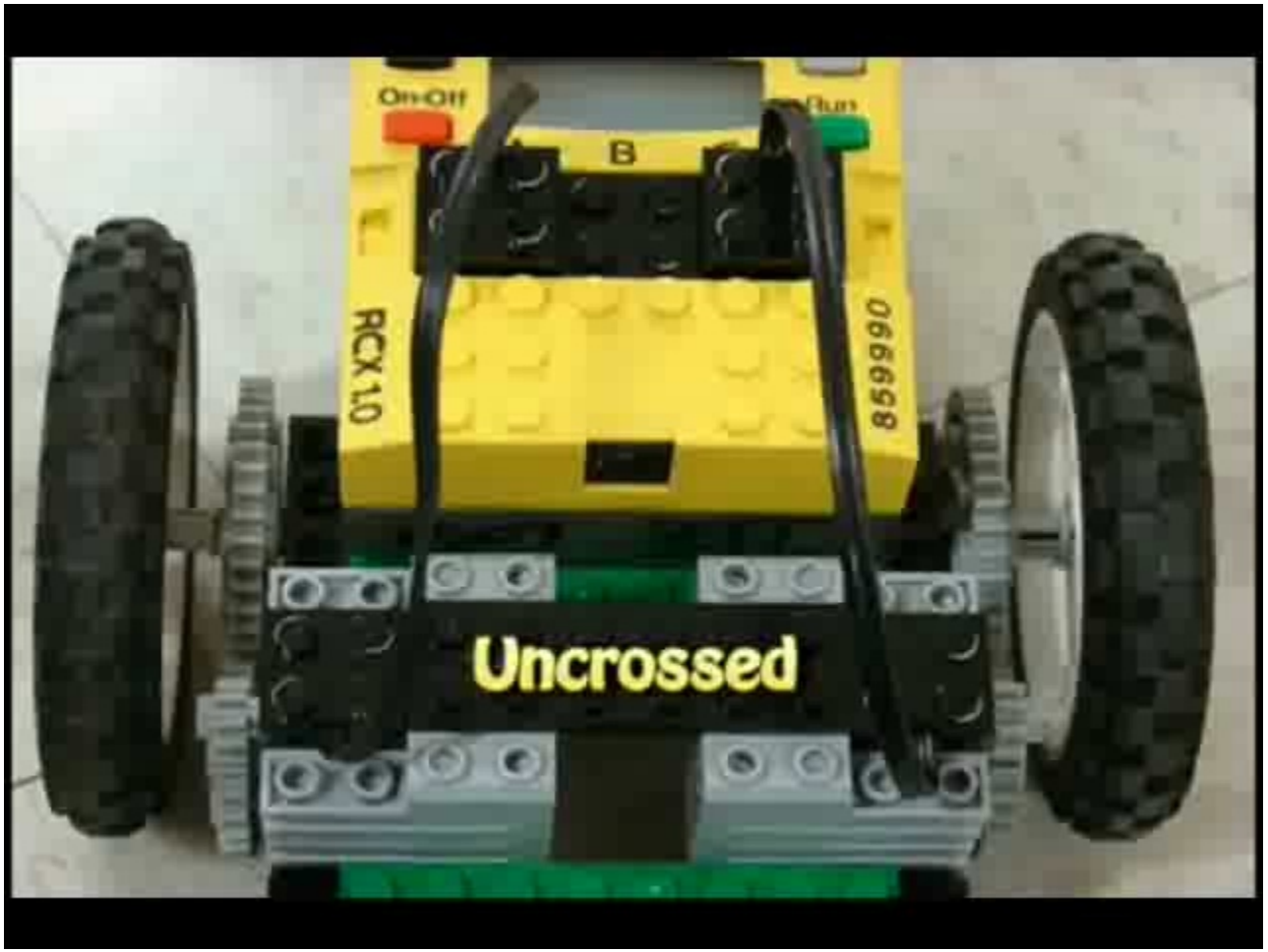
Braitenberg Vehicles

- Sensors for different stimuli (e.g. temperature, oxygen, etc) can be combined
- Vehicle will exhibit complex and dynamic behaviour



- Behaviour
  - is goal-directed, fast, flexible and adaptive
  - might even appear intelligent
- But
  - No information processing or cognitive processes
  - Agent is purely reactive
  - Simple architecture leads to robustness (why have cockroaches survived for millions of years)

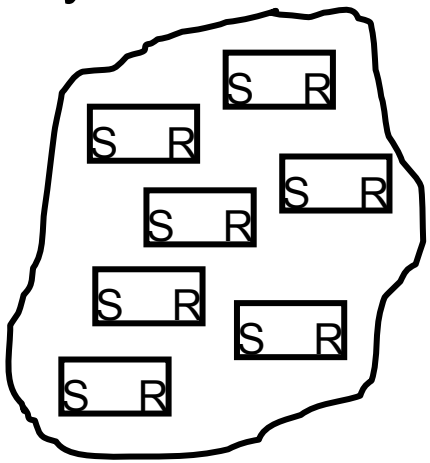
Braitenberg Vehicles



Braitenberg Vehicles

Behaviour-Based Architecture

- Brook's assumptions [Brooks 85]:
  - Complex behaviour does not require complex control system
  - Things should be simple to increase stability and robustness
  - Robots should be autonomous to survive long without human intervention
  - Environment is 3D and does not consist only of polyhedra
  - Absolute coordinate system leads to cumulative errors



Behaviour-Based Architecture

- Core ideas of a behaviour-based system:

Situatedness

Robot must deal with sensory and motor eventualities where it operates and not with abstract descriptions of the world

Embodiment

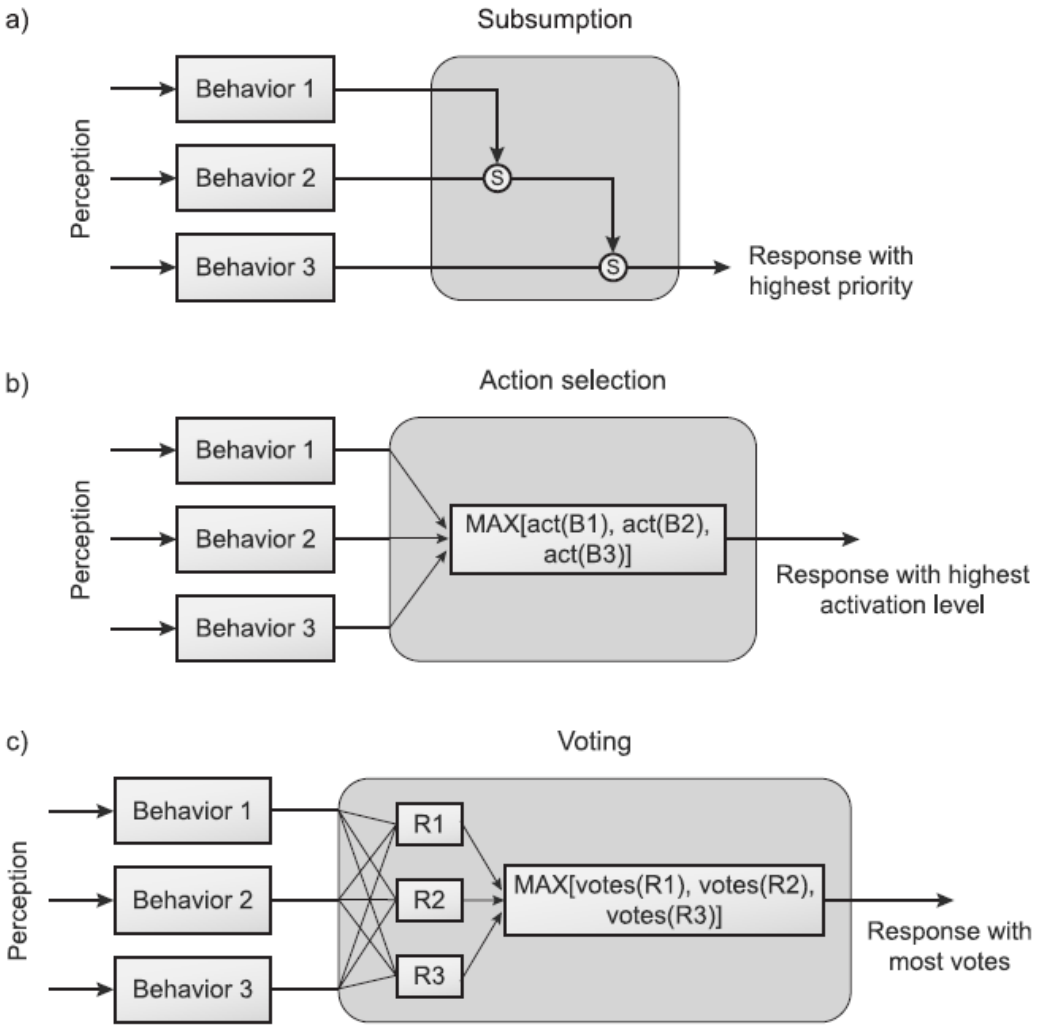
Experience the world directly through sensors and physically act on the environment rather than operate in simulated worlds

Behaviour-Based Architecture

- ***Stimuli*** relevant to behaviours $S = [s_1, s_2, \ldots, s_n]$
- Set of ***behaviours*** $B = [b_1, b_2, \ldots, b_n]$
- ***Weights/Gains*** associated with behaviours $G = [g_1, g_2, \ldots, g_n]$
- Response of behaviour i: $r_i = b_i(s_i)$
- Overall response: $\rho = C(G * B(S))$
  - ρ is a vector containing the system's overall response and is of the same type as response vector, e.g. $r_i = (\text{magnitude, angle})$
  - $*$ is special scaling operation where each scalar in G gets multiplied with magnitude component of response vector
  - C to select/combine output to produce single response vector

Competitive Behaviour Selection

- Subsumption
  - Higher level behaviours can overrule lower level behaviour output
- Action Selection
  - Selection through criterion (e.g. signal strength)
- Voting
  - Behaviours vote for action response R

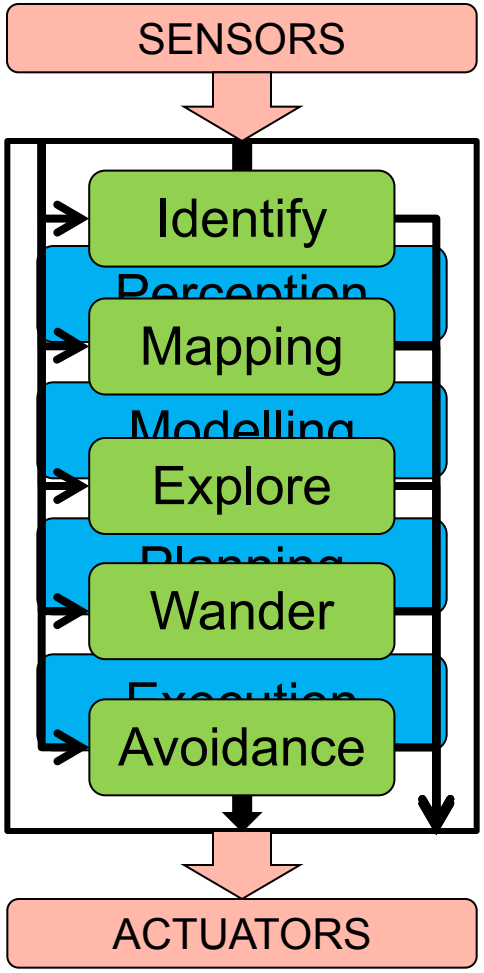


[Floreano/Mattiussi "Bio-inspired AI" adapted from Arkin 1998]

Subsumption Architecture

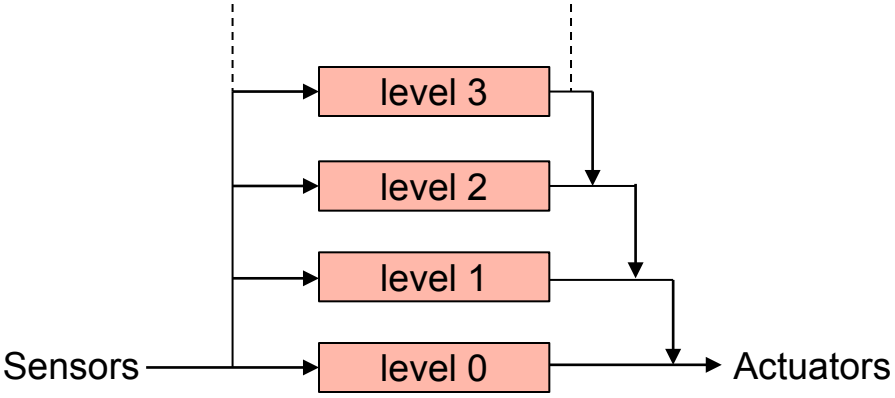
- Introduced by Rodney Brooks (1985)
- **Idea:**
  Decompose robot control in terms of task-achieving behaviours rather than functional modules

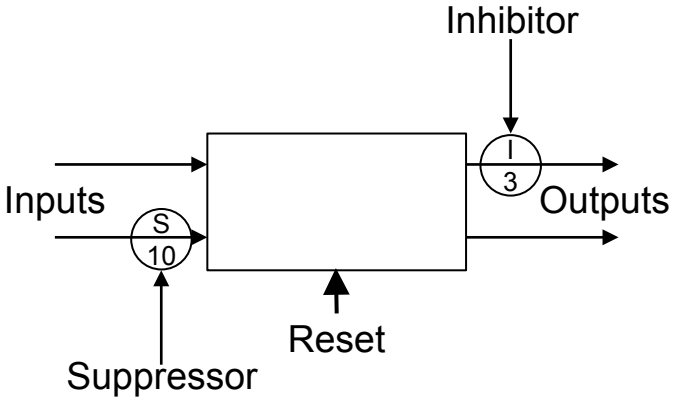
> "Complex (and useful) behaviour need not necessarily be a product of an extremely complex control system. Rather, complex behaviour may simply be the reflection of a complex environment. It may be an observer who ascribes complexity to an organism – not necessarily its designer." [Brooks 85]



Subsumption Architecture

- Layered Architecture
  - Higher level implies more specific behaviour
  - All layers have access to sensors and outputs of lower level modules
  - Higher levels can inhibit or subsume output of lower levels or reset state of module (move avoid excape example)

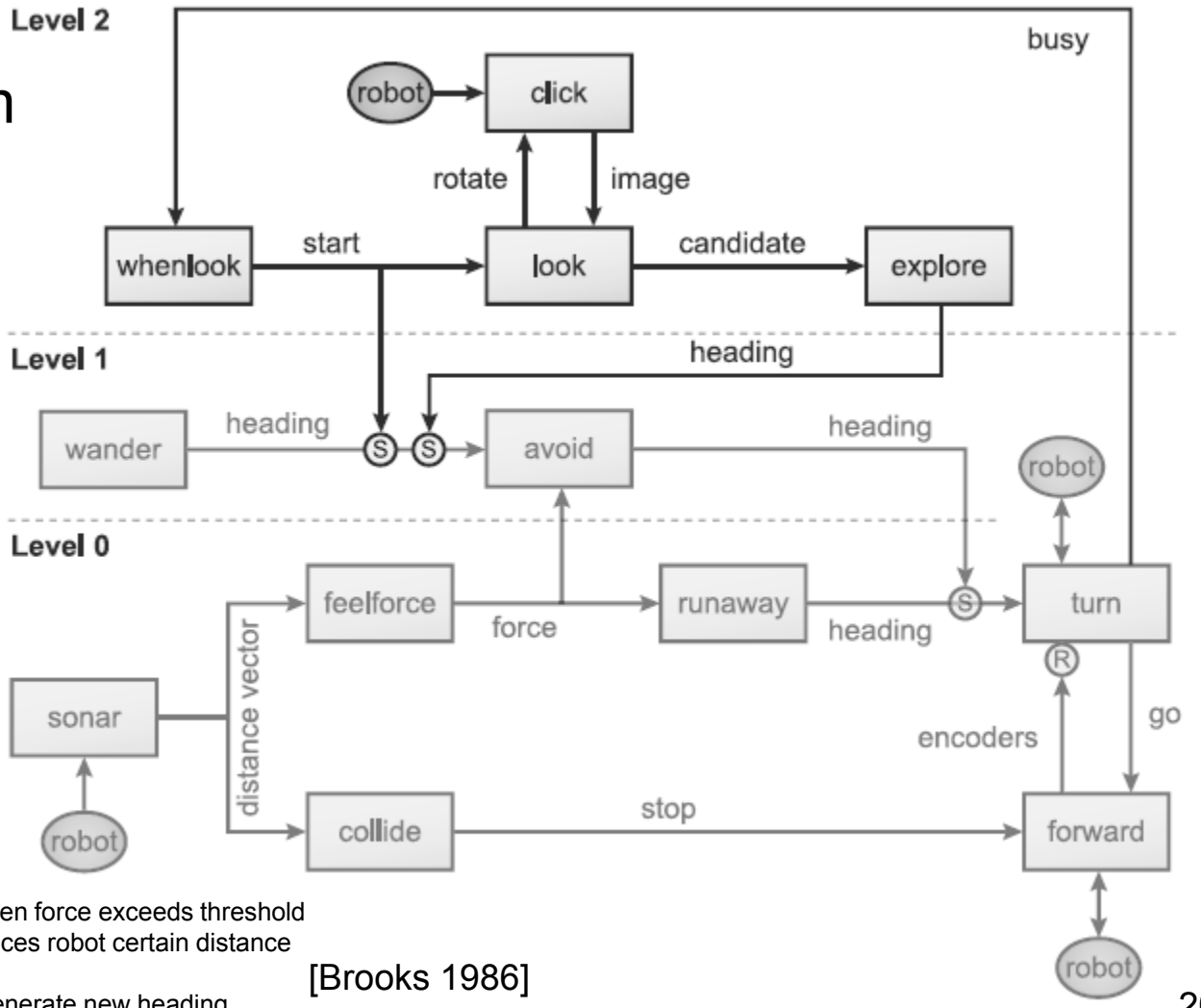




[Brooks 1985]

Subsumption Architecture

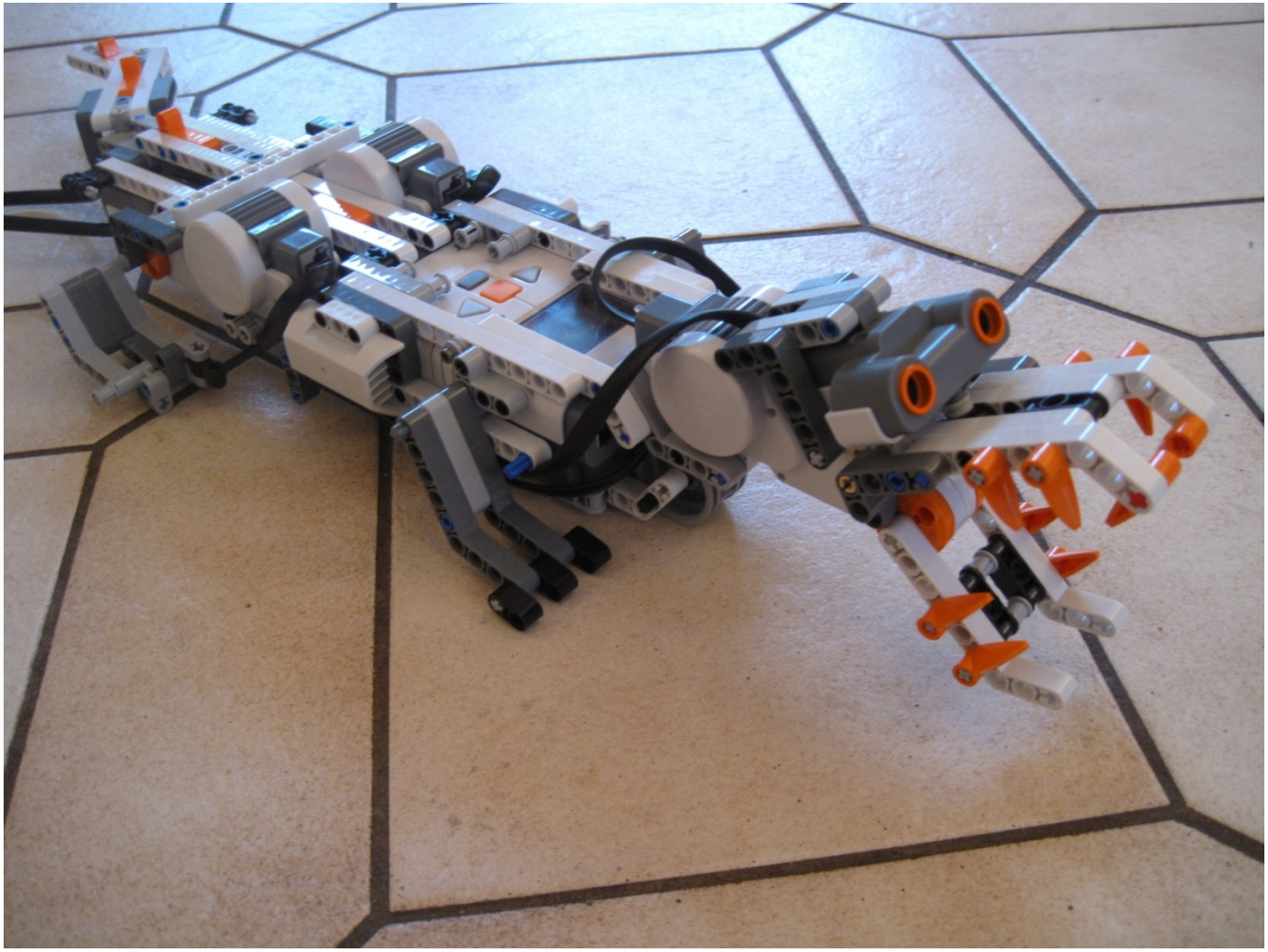
- Development from the bottom up
- Next level only when last level was tested



Robot with circular sonar sensor array
Sonar: Provides vector with distances
Collide: stops if any distance is below threshold
Feelforce: Calculates repulsive vector
Runaway: Calculates heading away from force when force exceeds threshold
Forward: returns wheel encoder values and advances robot certain distance
Wander: creates new heading every 10 intervals
Avoid: combines heading and repulsive force to generate new heading

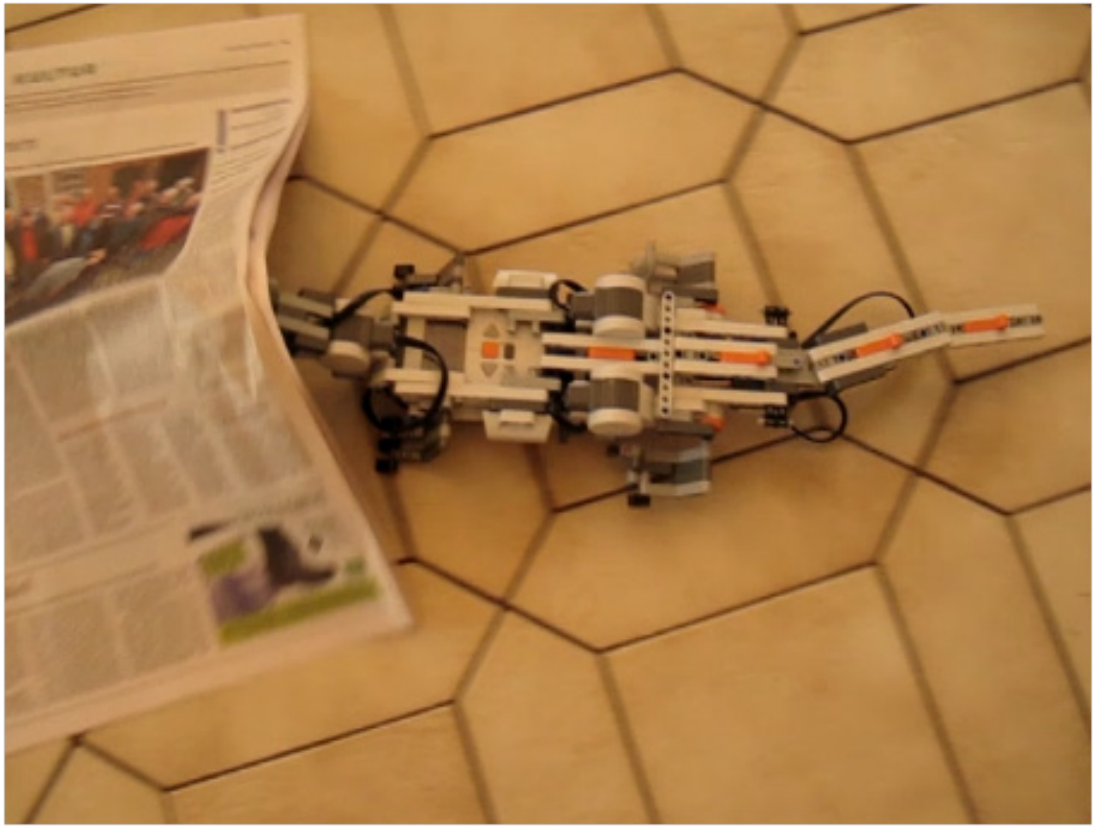
[Brooks 1986]

Behaviour-Based Example (student project)



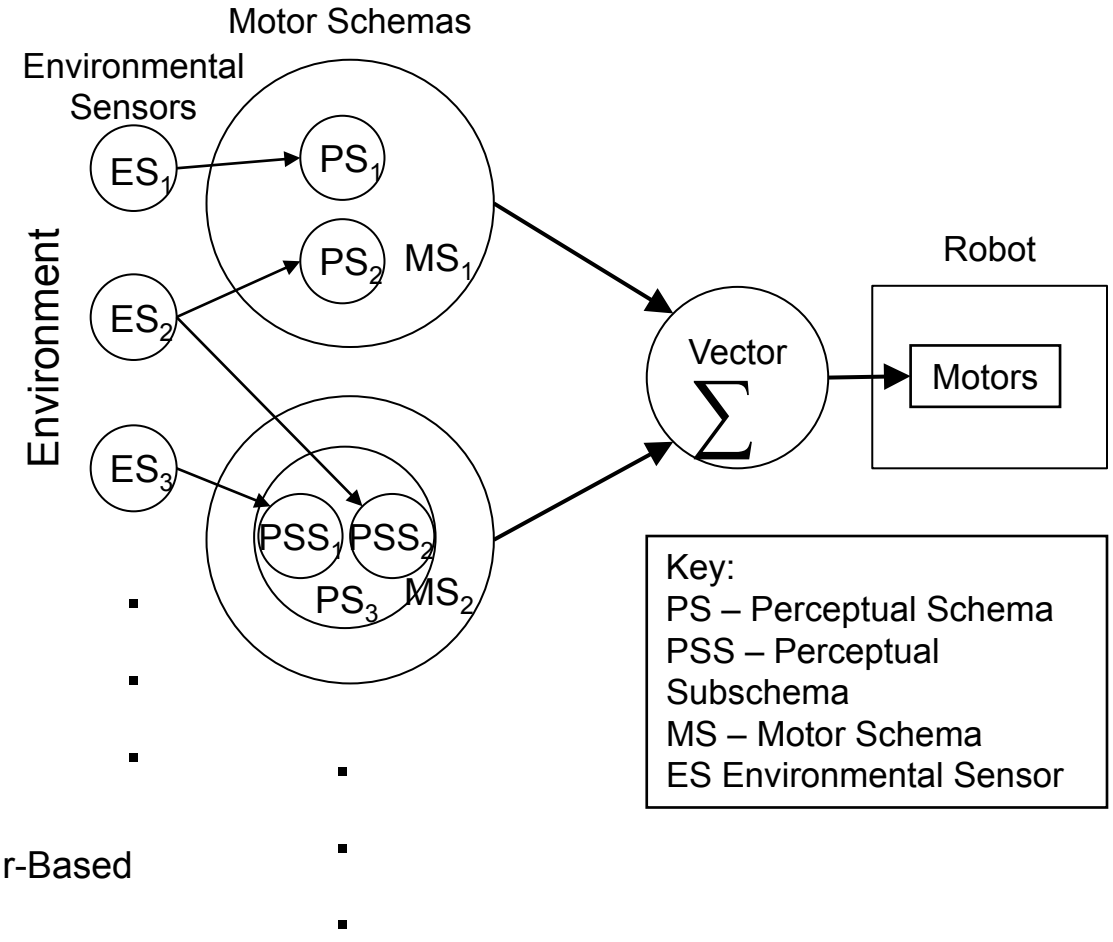
Behaviour-Based Example (student project)

Behaviour-Based Example



Motor Schemas

- More biologically motivated behaviour-based architecture
- Motor schema´s output vector containing direction and strength
- Example for cooperative behaviour selection



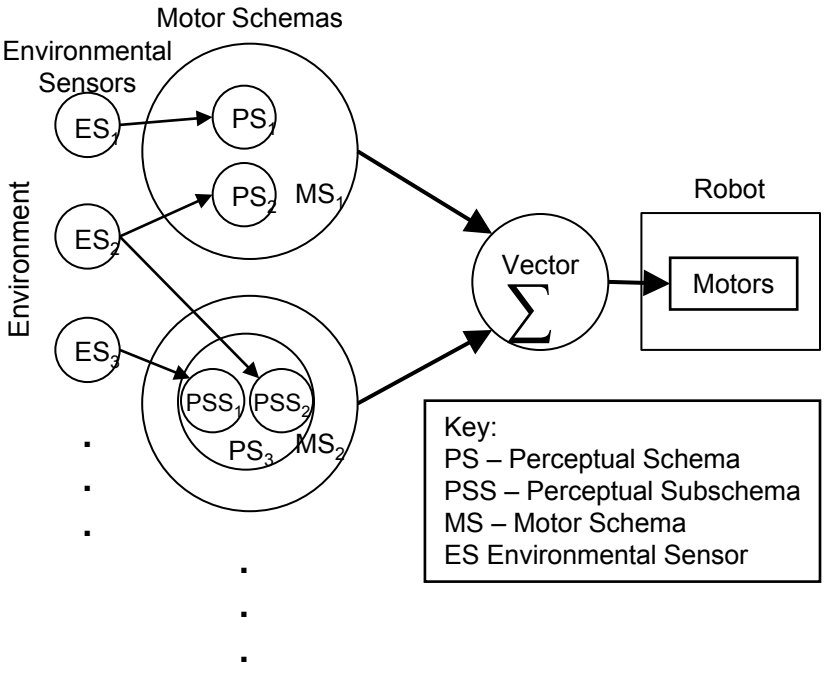
[Ronald C. Arkin (1998) “Behaviour-Based Robotics”, MIT Press]

Motor Schemas

- Perceptional Schema:
  - Provide environmental information specific for motor schema

- Motor Schema:
  - Takes input and produces response vector
  - Can have internal state

    (Braitenberg vehicles do not have an internal state)

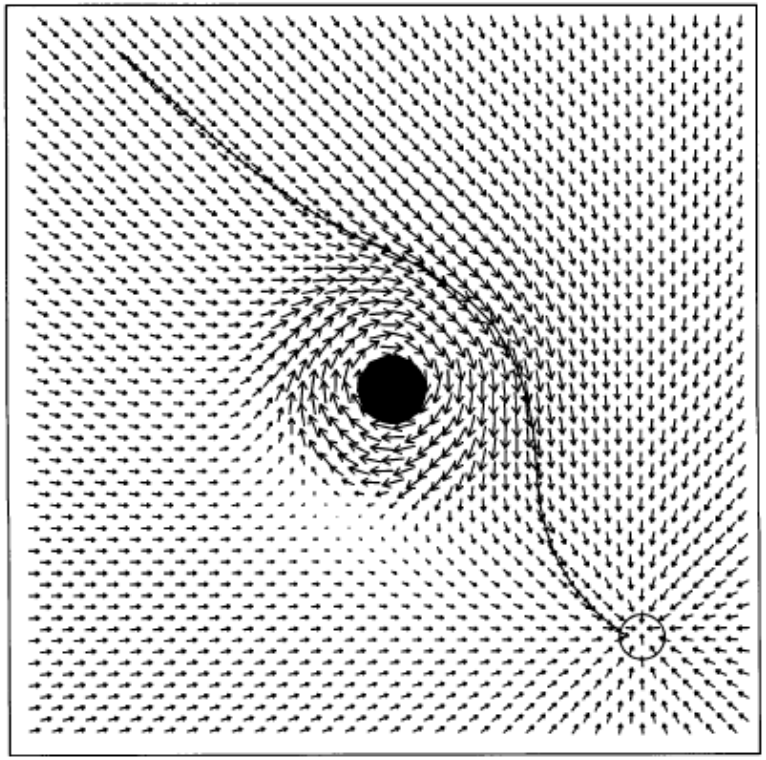


[Ronald C. Arkin (1998) "Behaviour-Based Robotics", MIT Press]

Motor Schemas

- Output of schemas can be visualised as ***potential field***
- Response vector contains direction and strength
- Each schema can be defined independently

➔ Schemas are combined by combining response vectors (vector addition)



- At any position the movement is predefined

Schema Examples

- ***Towards-Goal***
  - Ballistic:

    $V_{\text{magnitude}} = \text{fixed gain value}$

    $V_{\text{direction}} = \text{towards goal}$
  - Guarded:

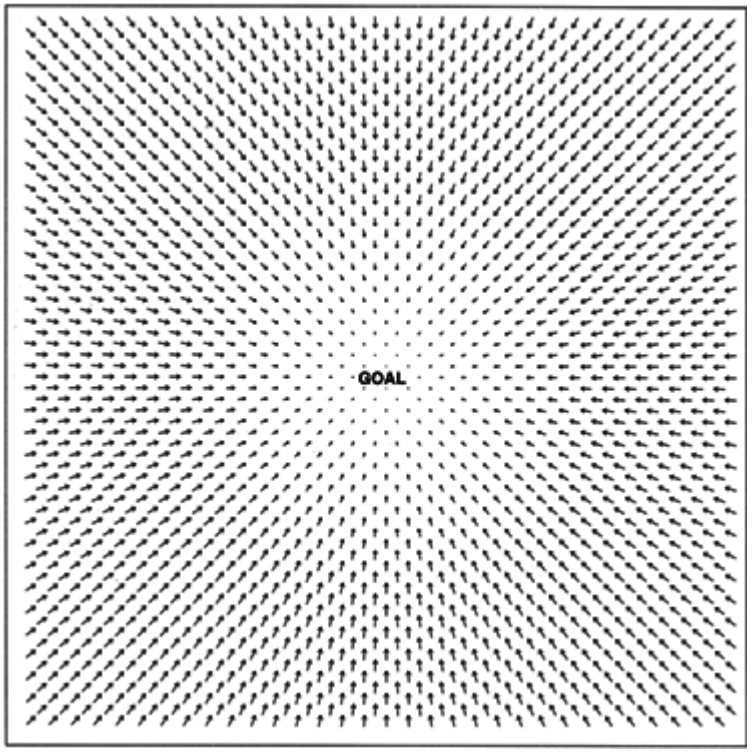
$$V_{\text{magnitude}} = \begin{cases} G & \text{for } d \geq S \\ G \cdot \dfrac{d}{S} & \text{for } d < S \end{cases}$$

$V_{\text{direction}} = \text{towards goal}$

with

d = distance to goal

S = radius of influence



Schema Examples

- ***Avoid-Obstacle***

$$V_{\text{magnitude}} = \begin{cases} 0 & \text{for } d > S \\ \dfrac{\text{S - d}}{\text{S - R}} \cdot G & \text{for } R < d \le S \\ \infty & \text{for } d \le R \end{cases}$$

$V_{\text{direction}}$ = radially away from object

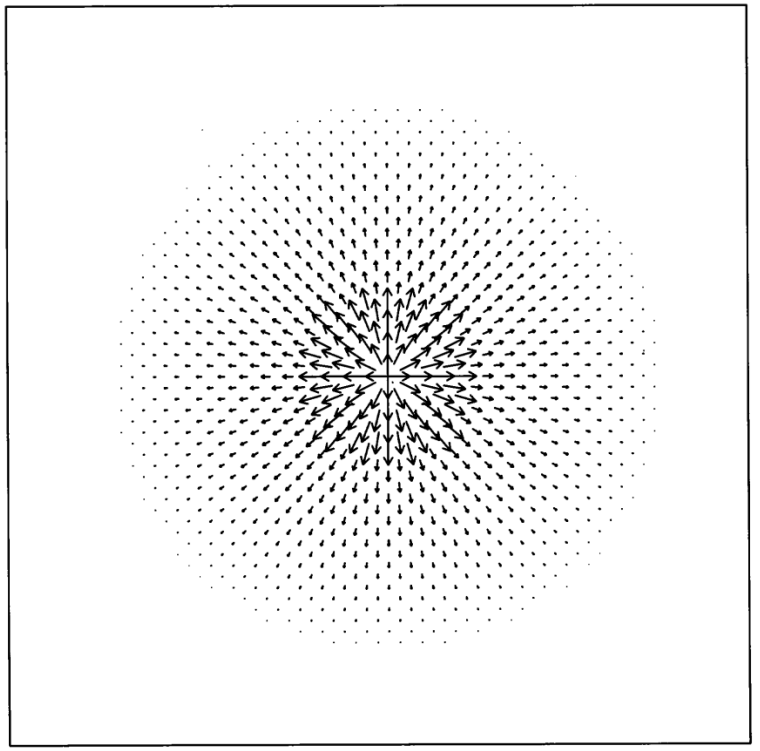
with

G = gain

d = distance to object

R = radius of object

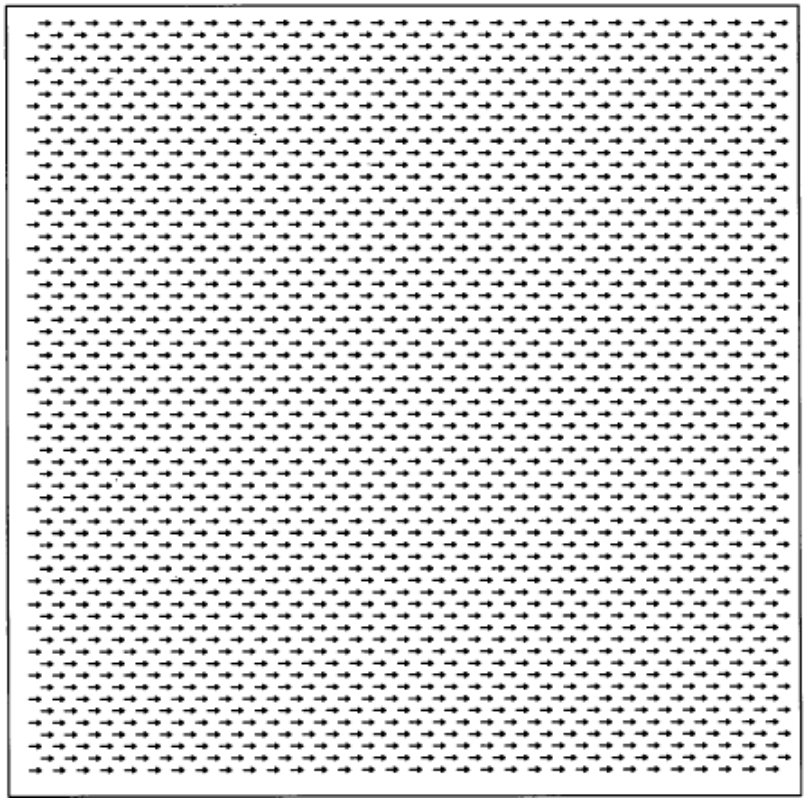
S = sphere of influence

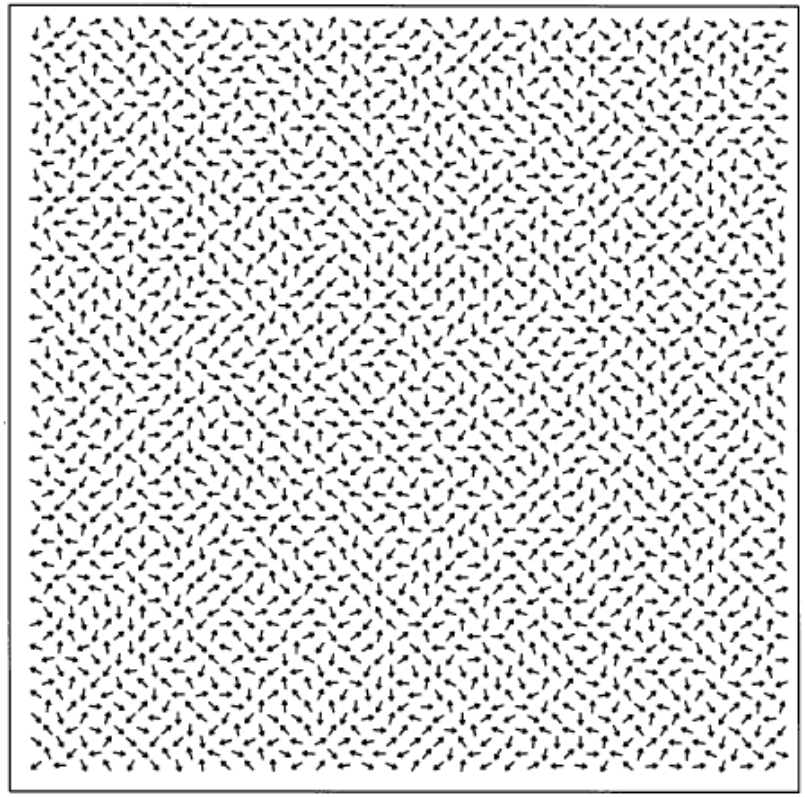


(A)

Schema Examples

- Simple: ***Move-Ahead*** and ***Noise***

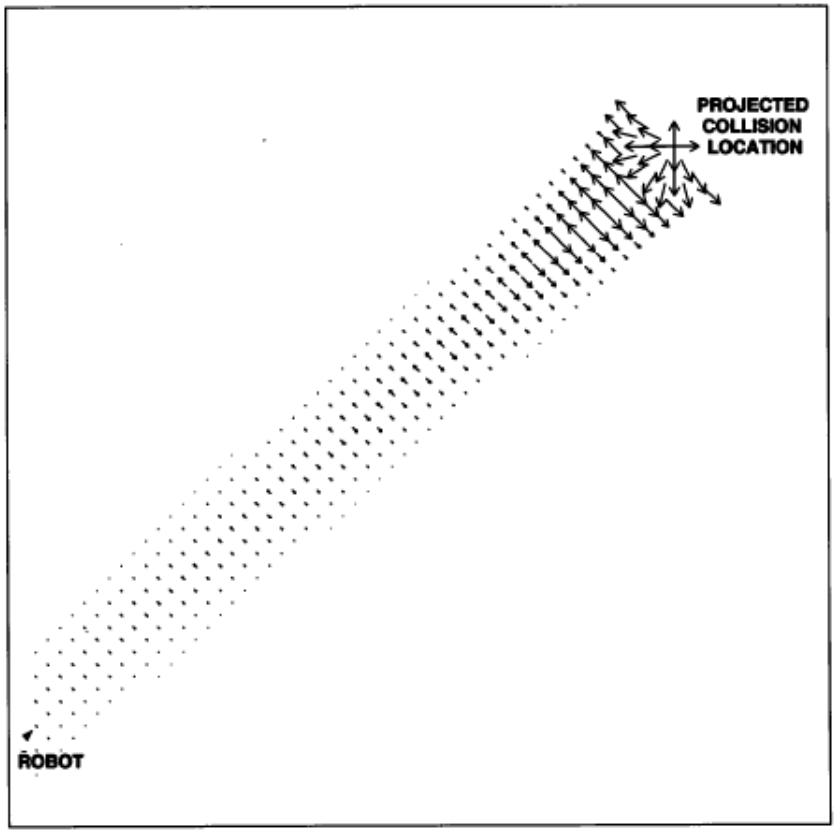


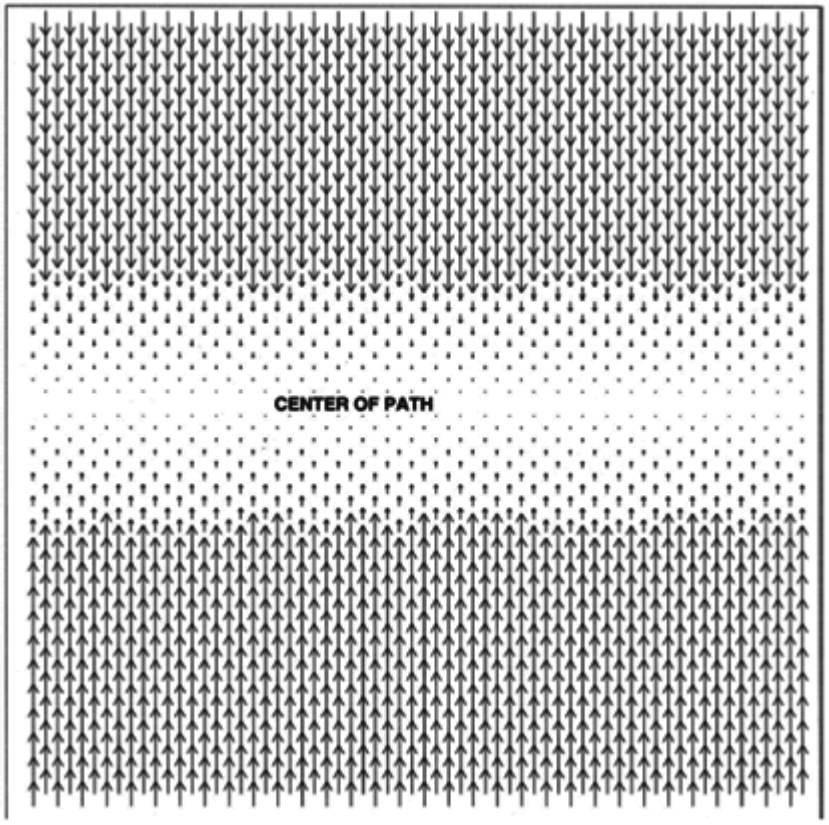


How is this different from other representations? Cellular automata?

Schema Examples

- More complex: ***Dodge*** and ***Stay-On-Path***

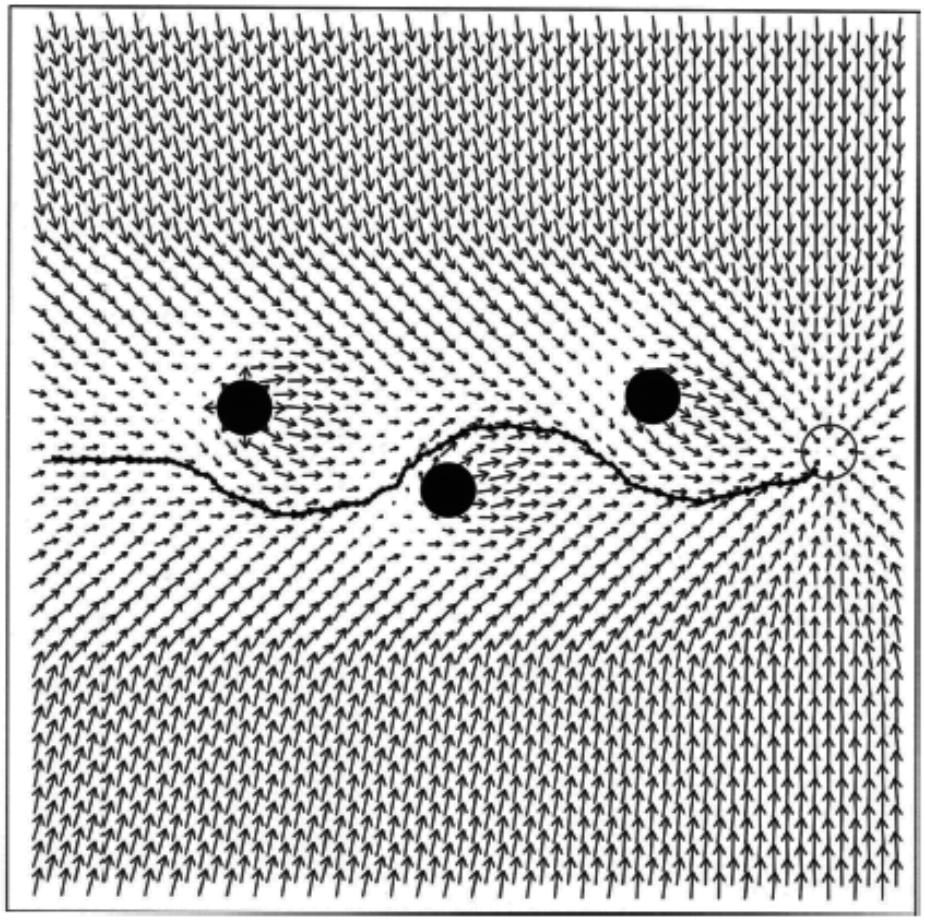




Compliant actuators vs stiff actuators…

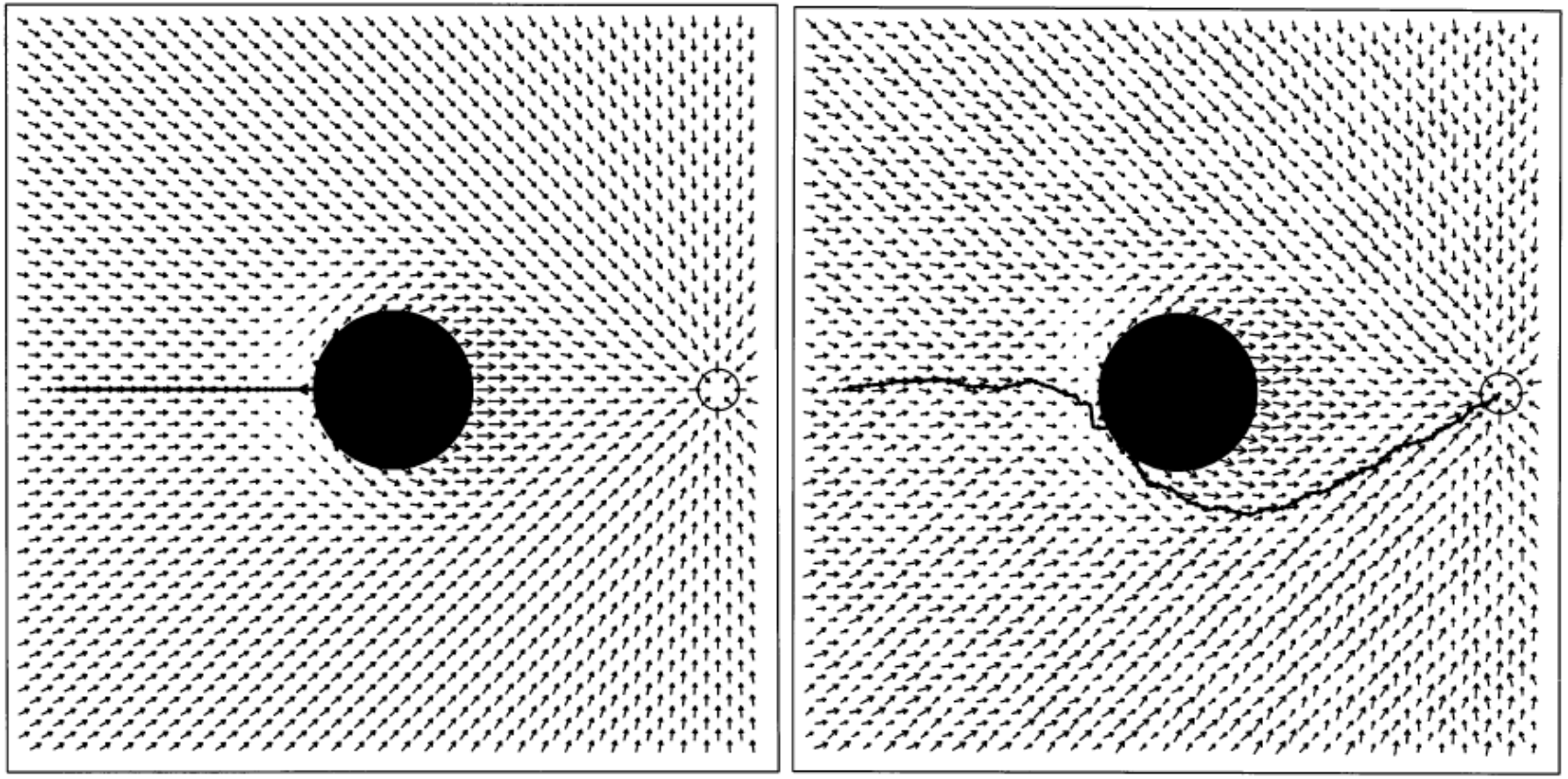
Schema Examples

- Overall robot behaviour is integration of all active schemas, e.g.
  - Stay-On-Path
  - Avoid-Obstacle
  - Towards-Goal (guarded)
  - E.g. vector addition



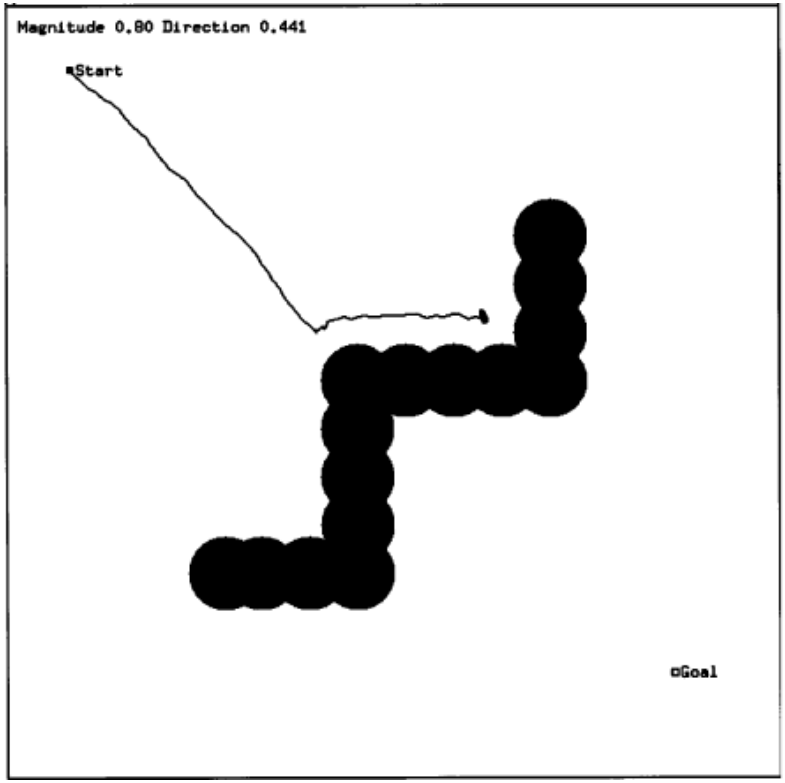
Problems with Schemas

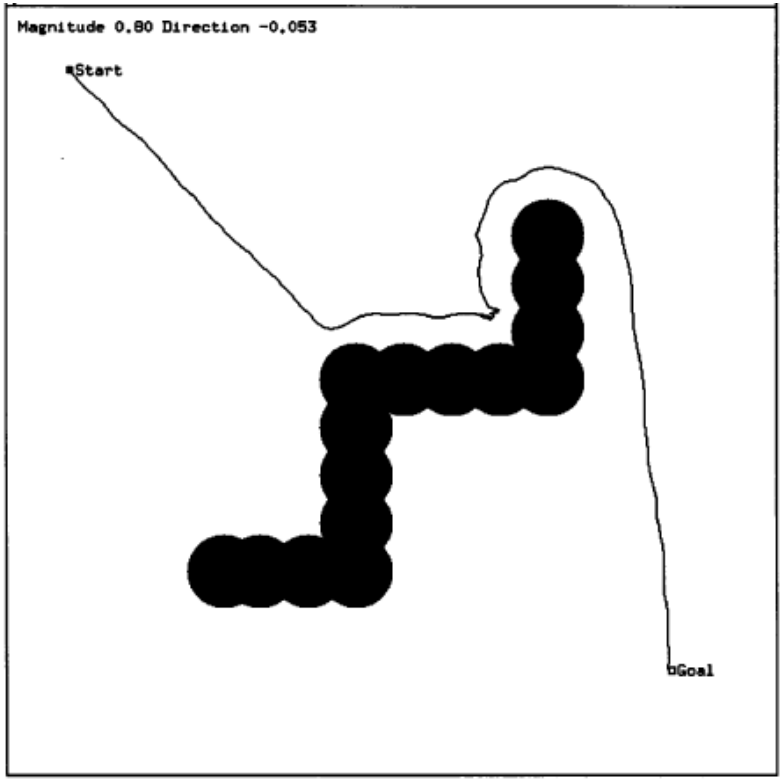
- Local Minima
  - Possible solution: Noise schema



Problems with Schemas

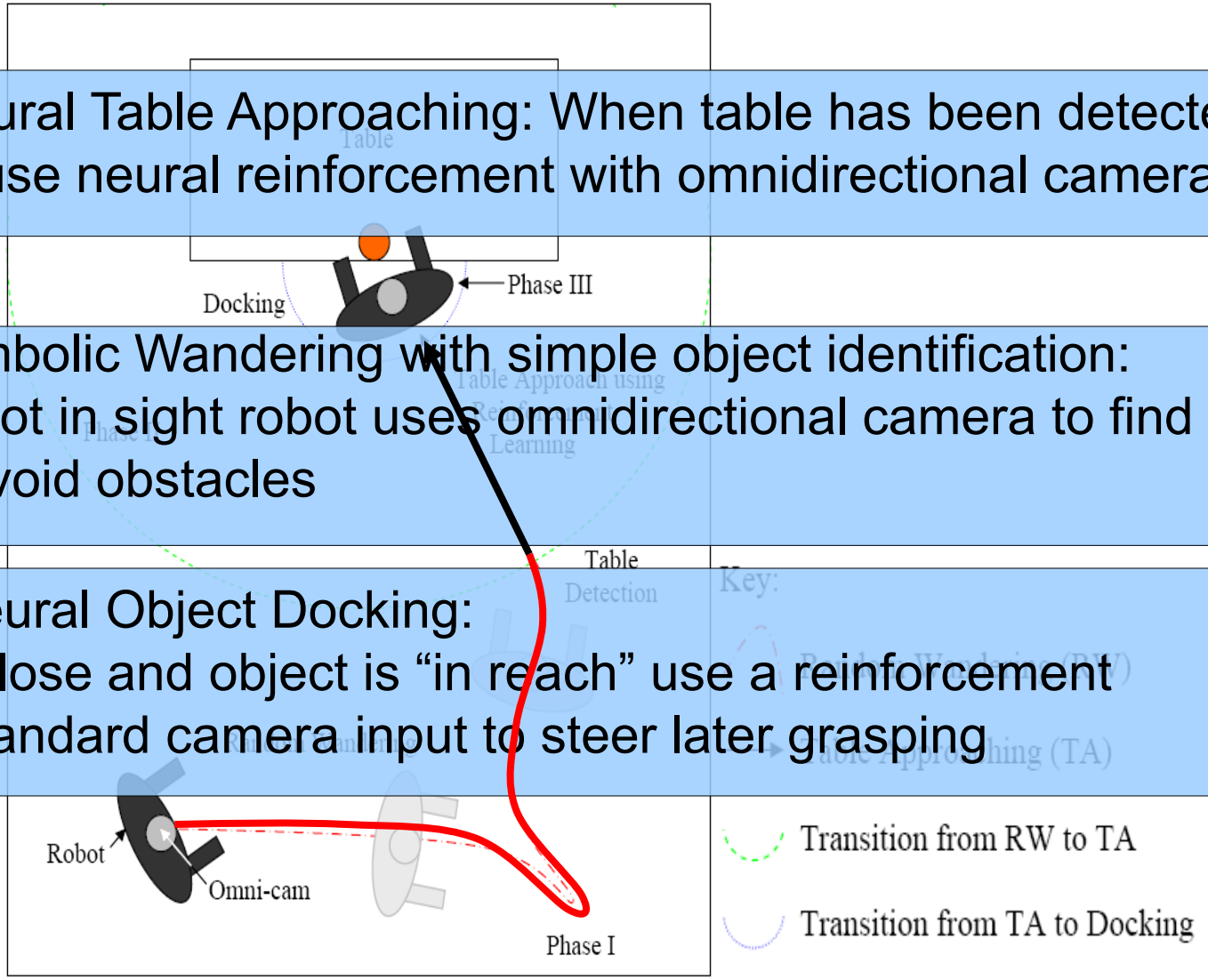
- Cyclic Behaviour
  - Possible solution: Avoid-Past schema (Repulsive force in recently visited areas)





Hybrid Approaching

- Phase II – Neural Table Approaching: When table has been detected at large range use neural reinforcement with omnidirectional camera
- Phase I – Symbolic Wandering with simple object identification: When table is not in sight robot uses omnidirectional camera to find the table and avoid obstacles
- Phase III – Neural Object Docking: When table is close and object is “in reach” use a reinforcement strategy with standard camera input to steer later grasping



Hybrid deliberative and reactive Architecture

- Mirrorbot robot (Wermter, Muse, Elshaw, Weber)

Reactive Agents Summary

- Disadvantages/Limits
  - ***Limited Information*** – Agents without environment models must have sufficient information from local environment
  - ***Non-Local information*** – how does agent take into account *non-local* information (e.g. memory?)
  - ***Learning global*** – Difficult to make reactive agents that learn global behaviour
  - ***Emergence*** – Since behavior emerges from component interactions plus environment, hard to see how to *engineer* specific agents
  - ***Complex dynamics*** – It is hard to engineer agents with large numbers of behaviors (dynamics of interactions become too complex to understand)

Reactive Agents Summary

- Advantages
  - ***Fast Reaction*** – Immediate mapping of sensory information onto motor actions
  - ***Robustness*** – If one part fails, robot may retain some behavioural competence
  - ***Multiple Goals*** – Can follow multiple goals simultaneously by combining behaviours
  - ***Extensibility*** – Easy to add new parts on top of old
  - ***Simplicity*** – Complexity derives from continuous interaction of simple modules with environment and each other
  - ***Computational tractability*** – Usually simple calculations and easy to parallelise