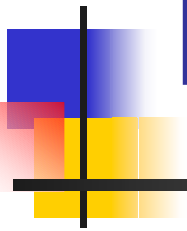
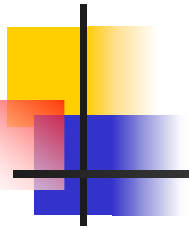


Robotic Behaviors

Version: 29.10.03



Objectives

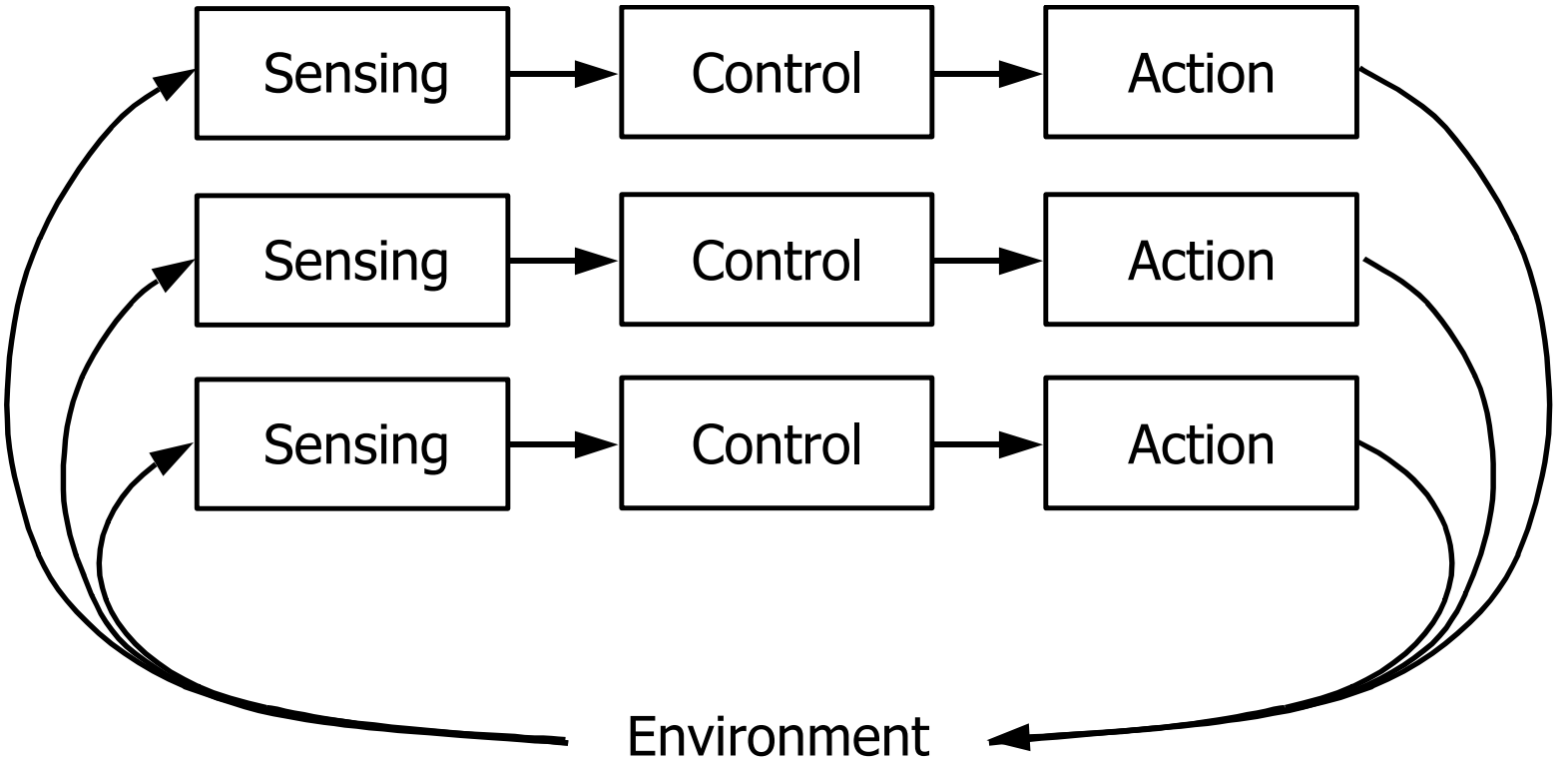
- Learn what robot behaviors are
- Understand methods to express and encode them
- Learn how to compose and coordinate behaviors
- Understand design choices

Behavior-Based Robotics' Response to Classical AI

- Reacted against classical AI
- Brooks (1987 – 1990):
  "Planning is just a way of avoiding figuring out what to do next"
  "Elephants don't play chess"
- Increased emphasis on:
  - Sensing and acting within environment
- Reduced emphasis on:
  - Knowledge representation
  - Planning

Behavior-Based / Reactive: Based on Biological Paradigm

Philosophy: "World is own best model; therefore, don't try to build another world model"



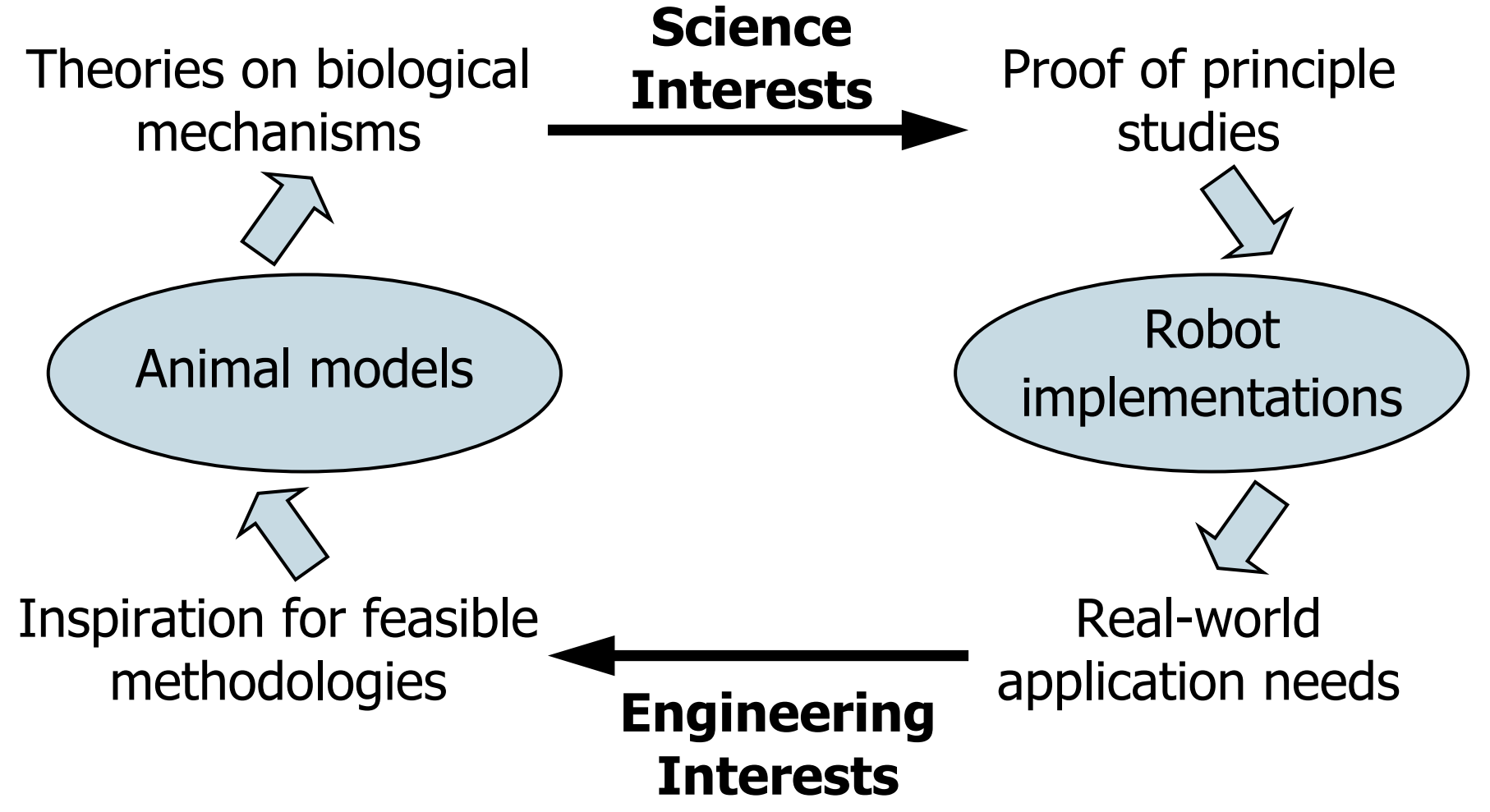
What do we mean by „Intelligence“?

- Open question: Where intelligence begins and ends
- Intelligence (our working definition): The ability to improve an animal or human’s likelihood of survival within the real world, and, where appropriate, to compete or cooperate successfully with other agents to do so
- For our purposes, we call certain animal behaviors “intelligent”

Ethology and Cognitive Psychology

- Ethology: The study of animal behavior in natural conditions
- "Founding fathers" of ethology: Konrad Lorenz and Niko Tinbergen
  - Individual animal behaviors
  - How do animals acquire behaviors
  - How do animals select or coordinate groups of behaviors
- Cognitive psychology: The study of how humans think and represent knowledge

Reasons for Operationalizing Animal Models in Robotics



What is "Behavior"?

- Behavior: Mapping of sensory inputs to a pattern of motor actions that are used to achieve a task
- Reflexive behaviors:
  - Stimulus-response
  - Hard-wired for fast response
  - Example: (physical) knee-jerk reaction
- Reactive behaviors:
  - Learned
  - "Compiled down" to be executed without conscious thought
  - Examples: "muscle memory" – playing piano, riding bicycle, running, etc.
- Conscious behaviors:
  - Require deliberative thought
  - Examples: writing computer code, completing your tax returns, etc.

Acquiring Behaviors

Lorenz and Tinbergen identified four ways to acquire a behavior:

-Innate:
- Born with it
- Simple, effective, computationally inexpensive

- Sequence of innate behaviors:
- Born with a sequence of behaviors
- Several steps follow one after the other
- Similar to computer science's Finite State Machines

- Innate with memory:
- Born with behaviors that need initialization, customization
- Agent has to learn parameters of behaviors

- Learned:
- Can be complex behaviors
- Learn the releasing mechanism and the actions, and how to put them together

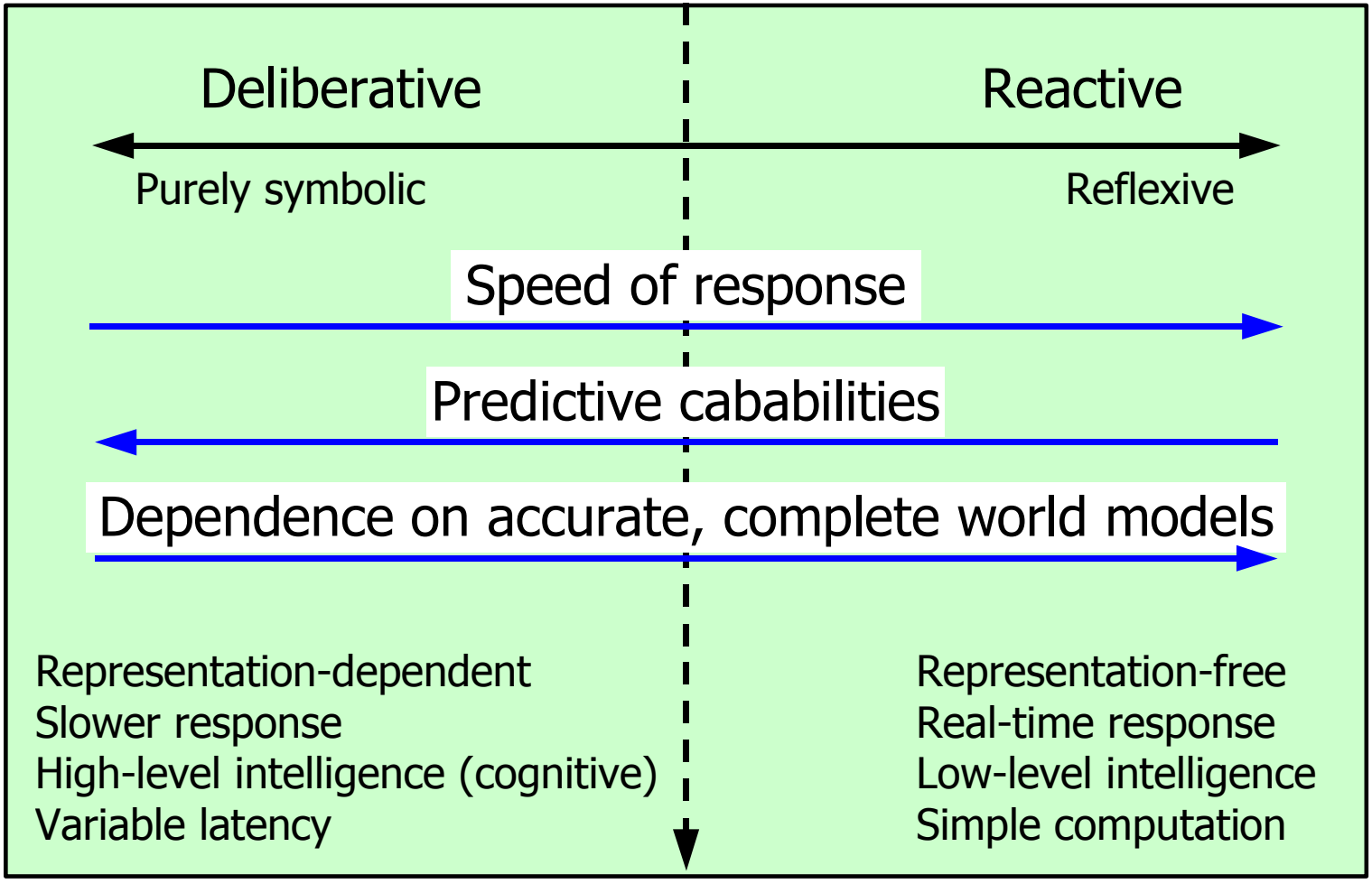
Interaction of Concurrent Behaviors

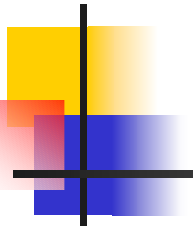
- Usually, behaviors follow a fixed sequence
- However, can have multiple behaviors activated in certain environmental situations
- How do behaviors interact?
  - Equilibrium:
    - Behaviors balance each other out
    - Example: squirrel with food close to human
  - Dominance of one / winner-take-all
  - Cancellation:
    - Example: Male stickleback fish: if want to defend and fight (due to overlapping territories) and upbuilding a nest

Perception in Behaviors

- Two functions of perception:
  - Release: to release a behavior
  - Guide: to provide information needed to accomplish a behavior
- Action-oriented perception:
  - Perception filters the incoming sensory stream to extract information specific to the task at hand
  - Note: Difference from hierarchical world-model building
- Affordance: "perceivable potentialities of the environment for an action"
  - Example: Color "red" to a baby arctic tern is perceivable and represents the potential for feeding

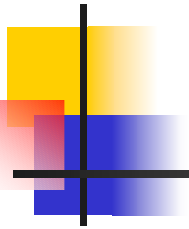
Wide Spectrum of Robot Control





Deliberative Control

- Common characteristics:
  - Hierarchical in structure
  - Clearly identifiable division of functionality
  - Communication and control is predictable and predetermined
  - Higher levels provide subgoals for lower levels
  - Planning scope changes with descent in hierarchy
  - Heavy reliance on symbolic representations
- Seemingly well-suited for structured and highly predictable environments (e. g., manufacturing)



Deliberative Systems

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

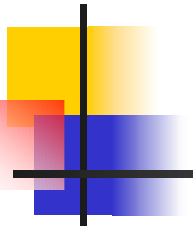
Reactive Systems

- Behavior = mapping of sensory inputs to a pattern of motor actions that are used to achieve a task

  or (equivalently) = reaction to a stimulus

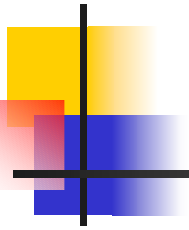
- A reactive robotic system tightly couples perception to action without the use of intervening abstract representations or time history

- Recall: reactive paradigm:



Reactive Systems

Reactive control is a technique for tightly coupling perception and action, typically in the context of motor behaviors, to produce timely robotic response in dynamic and unstructured worlds.

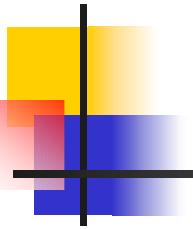


Reactive Systems

- 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

Reflexive (purely reactive) Behavior

Behavior that is generated by hardwired reactive behaviors with tight sensor-effector arcs, where sensory information is not persistent, and no world models are used whatsoever.



Hybrid Systems

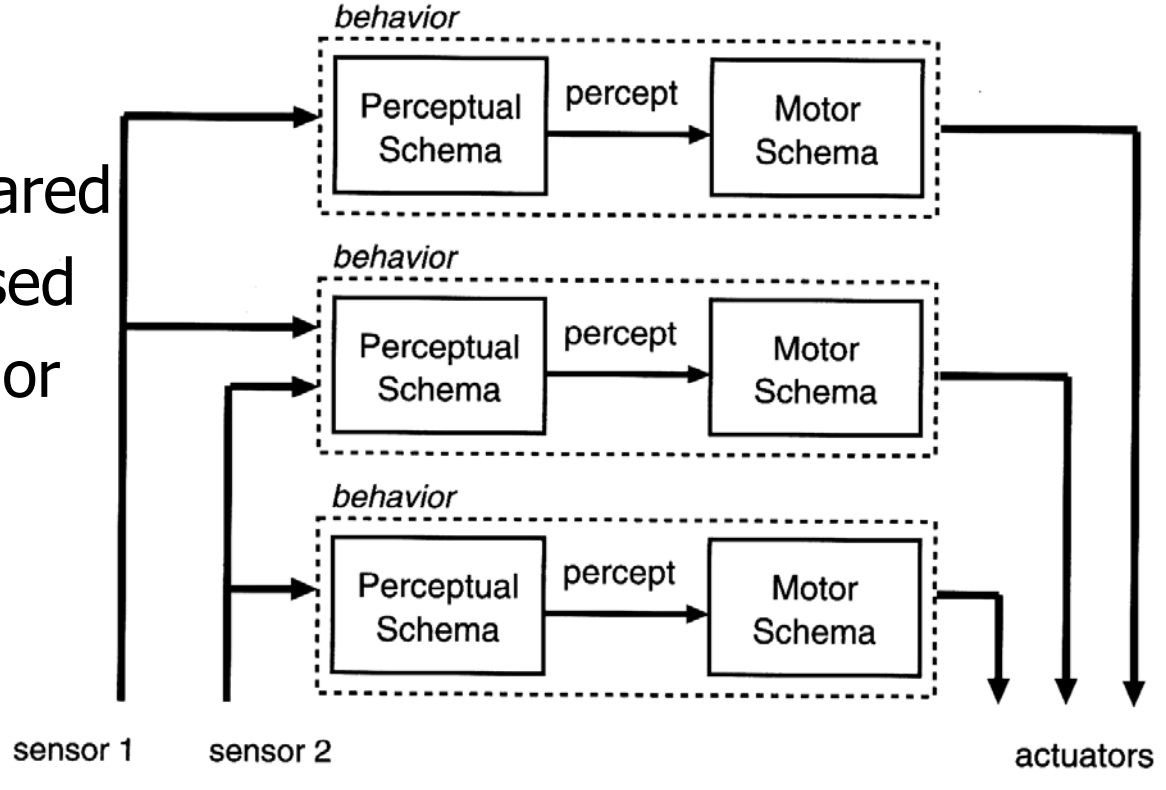
- Combine the two extremes
- reactive system on the bottom
- deliberative system on the top
- connected by some intermediate layer
- Often called 3-layer systems
- Layers must operate concurrently
- Different representations and time-scales between the layers

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

Reactive Systems

- Behaviors are a direct mapping of sensory inputs to a pattern of motor actions that are then used to achieve a task.
- Sensing is local
- Sensors can be shared
- Sensors can be fused locally by a behavior



Characteristics of Reactive Robotic Systems

- Behaviors serve as basic building blocks for robotic actions - typically, behaviors are simple sensori-motor pairs
- use of explicit abstract representational knowledge is avoided in the generation of a response
- Animal models of behavior often serve as a basis for these systems
- These systems are inherently modular from a software design perspective

Characteristics of reactive behaviors

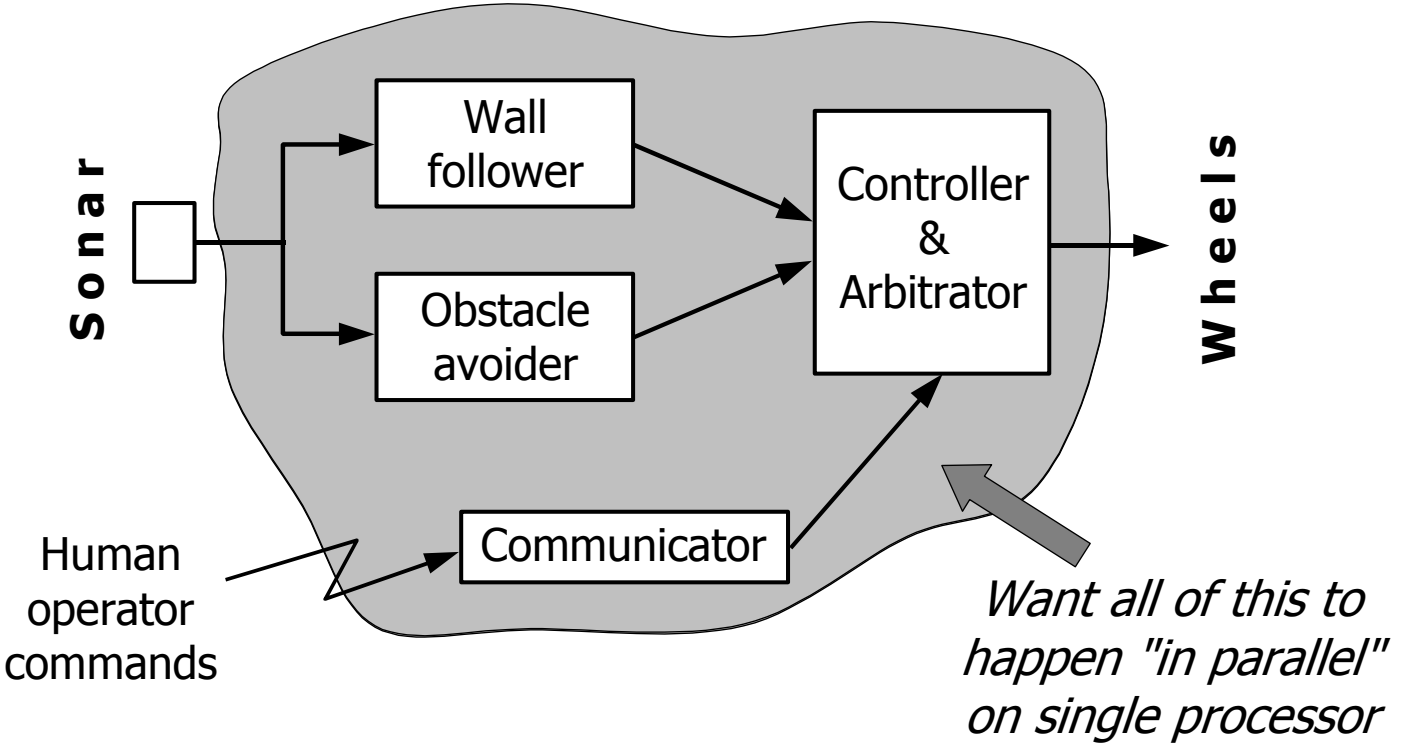
1. Robots are situated agents operating in an ecological niche. When a robot acts, it changes the world, and receives immediate feedback about the world through sensing. What the robot senses affects its goals and how it attempts to meet them, generating a new cycle of actions.
2. Behaviors serve as the basic building blocks for robotic actions, and the overall behavior of the robot is emergent. Behaviors are independent, computational entities and operate concurrently. There may be a coordinated control program in the schema of a behavior, but there is no external controller of all behaviors for a task. The overall behavior of a reactive robot emerges from the way its individual behaviors interact.

Characteristics of reactive behaviors

3. Only local, behavior-specific sensing is permitted. The use of explicit abstract representational knowledge in perceptual processing, even though it is behavior-specific, is avoided. Any sensing which does require representation is expressed in ego-centric (robot-centric) coordinates. For example, consider obstacle avoidance. An ego-centric representation means that it does not matter that an obstacle is in the world at coordinates (x, y, z), only where it is relative to the robot.
4. Behavior based systems inherently follow good software design principles. The modularity of these behaviors supports the decomposition of a task into component behaviors. The behaviors are tested independently, and behaviors may be assembled from primitive behaviors.

Follow walls and obey human operator commands

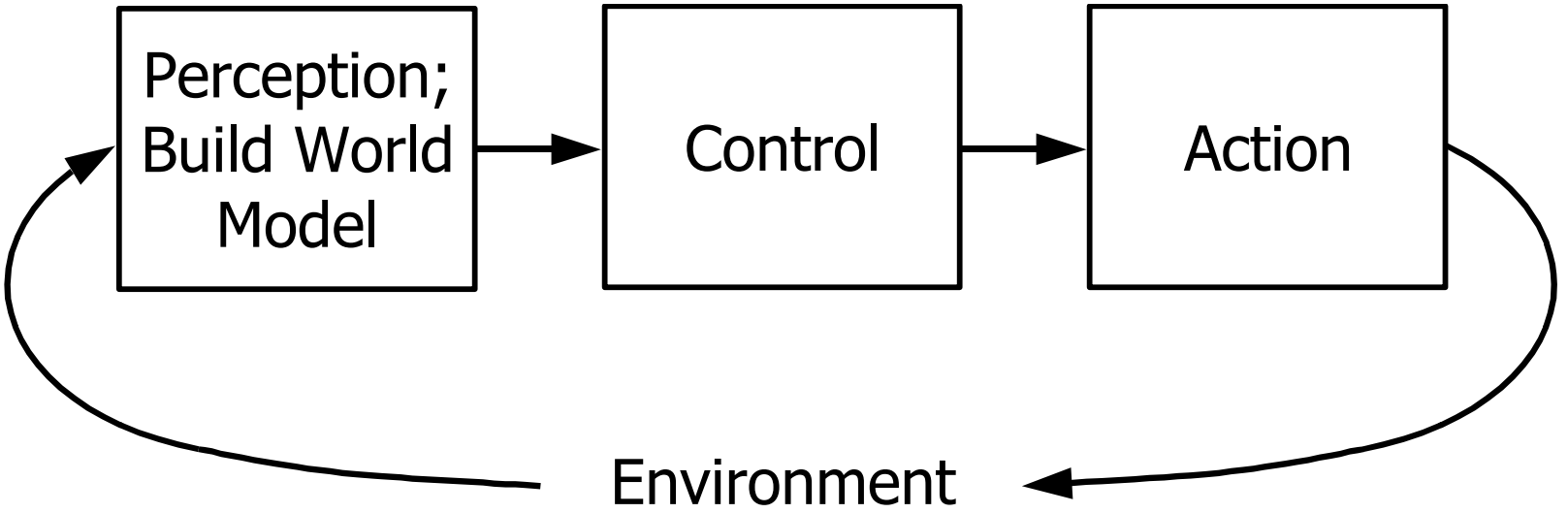
- Assume we have the following functions needed:
  - Communications to operator interface – commands such as "stop", "go", etc.
  - Sonar: used to follow wall and avoid obstacles



Key Issues of Behavior-Based Control

- Situatedness: robot operates in the real world
- Embodiment: robot has a physical presence (body)
- Emergence: intelligence arises from interaction of robot with environment
- Grounding in reality: avoid symbol grounding problem
- Ecological dynamics: cannot characterize environment
- Scalability: unknown whether behavior-based control will scale to human-level intelligence

Hierarchical Organization



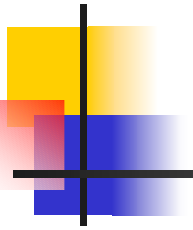
Earliest robot control projects used this approach, with limited success

Definitions

- *Individual behavior*: A stimulus/response pair for a given environmental setting that is modulated by attention and determined by intention
- *Attention*: Prioritizes tasks and focuses sensory resources and is determined by the current environmental context

Definitions

- *Intention*: determines which set of behaviors should be active based on the robotic agent's internal goals and objectives
- *Overt or emergent behavior*: the global behavior of the robot or organism as a consequence of the interaction of the individual active behaviors



Hallmarks of Behavior-based Methods

- *Situatedness* - the robot is surrounded by the real world, not abstract representations
- *Embodiment* - the robot has a physical body within the world
- *Emergence* - intelligence is a result of interactions with environment

Related Issues

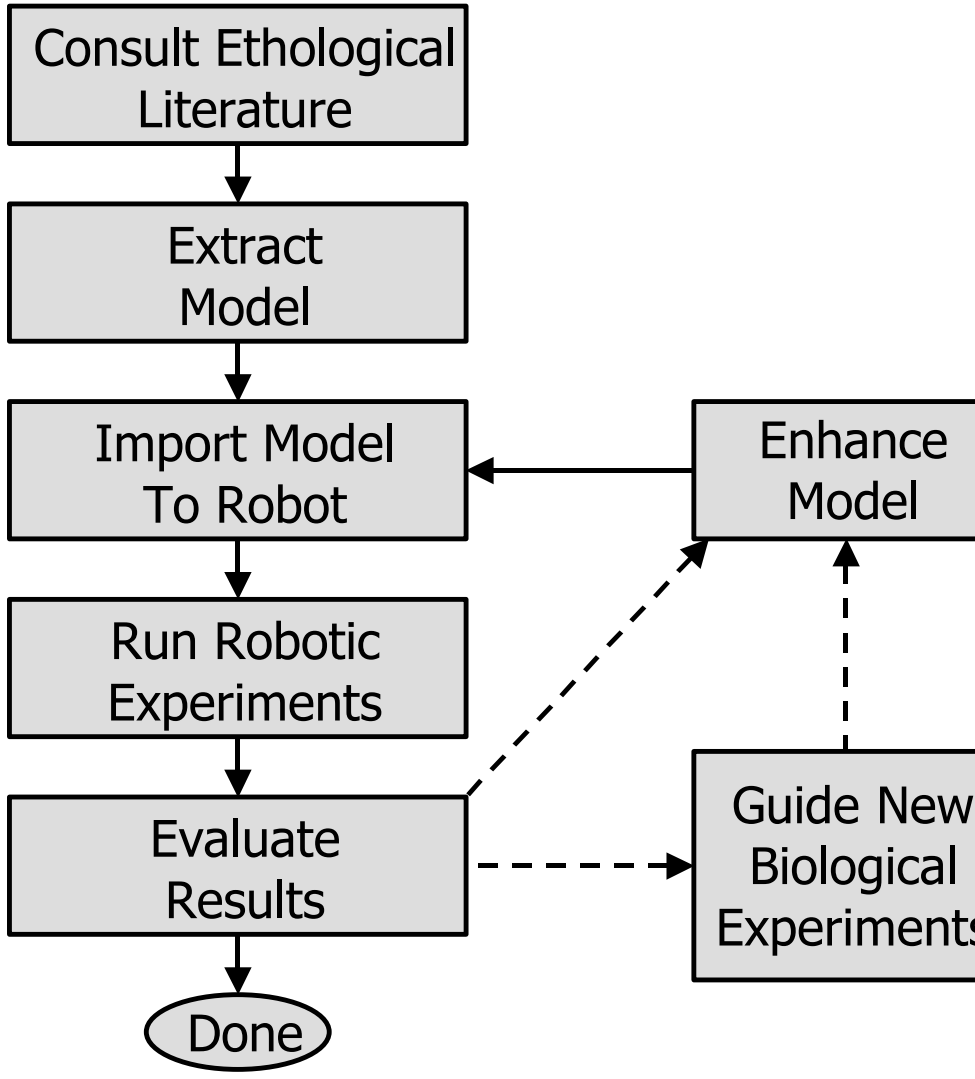
- Grounding in reality:
  - the *symbol-grounding* problem
- Ecological dynamics
  - niche finding
  - evolution and adaptation
- Scalability
  - hotly debated

Basis for Robotic Behavior

- Key questions:
  - What are the right behavioral building blocks for robotic systems?
  - What really is a primitive behavior?
  - How are these behaviors effectively coordinated?
  - How are these behaviors grounded to sensors and actuators?
- No universally agreed-upon answers
- Ultimate judge: appropriateness of the robotic response to a given task and environment

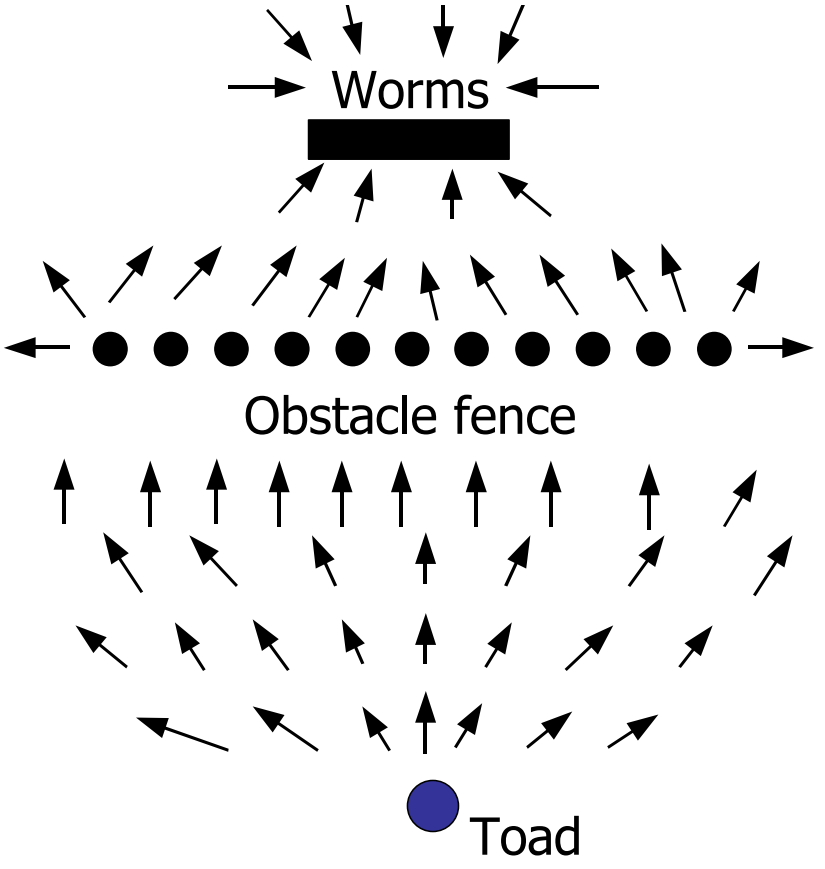
Ethologically Guided / Constrained Design

Use studies of animal behavior for inspiration / guidance



Example: Toad/Robot Navigation

- Motion divergence field from toad/robot studies
- Analogous implementation in robots of potential fields

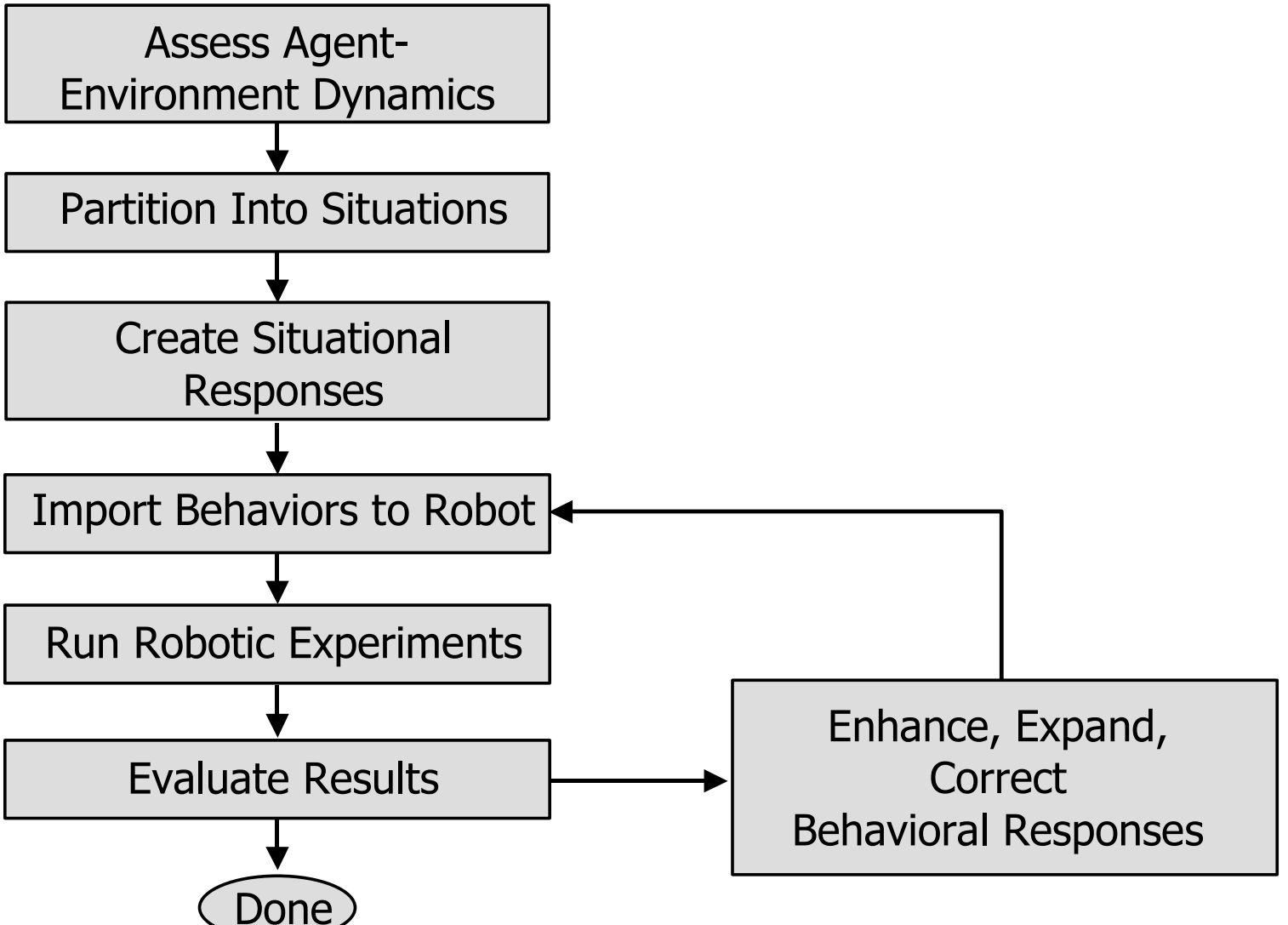


This fields defines most likely direction of motion for animal/robot at each point in space

Situated Activity-Based Design

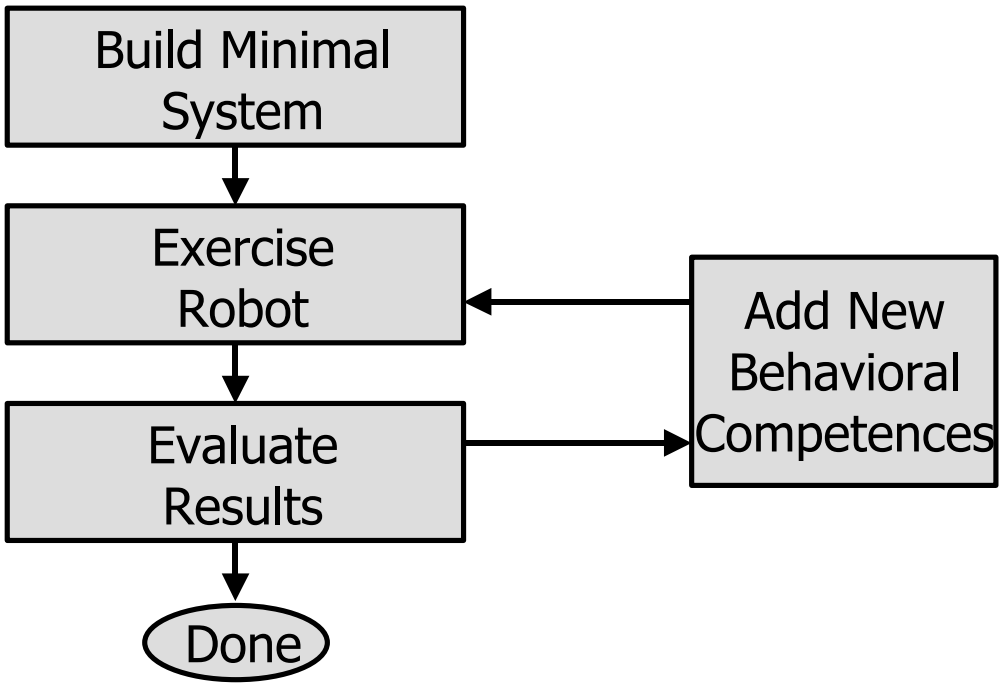
- Situated activity: robot's actions are predicated upon the situations in which the robot finds itself
- Therefore, the perception problem is reduced to recognizing what situation(s) the robot is in and then choosing one action (out of perhaps many) to undertake
- When robot finds itself in new situation, it selects more appropriate action
- Using this design methodology: requires solid understanding of relationship between robotic agent and its environment

Situated Activity Design



Experimentally Driven Design

- Created in a bottom-up manner
- Iterative design



Example of Experimentally Designed Robot

Genghis: Designed at MIT AI Lab in late 1980s by Rodney Brooks

Built in a bottom-up fashion using experimentally driven design

"A robot that walks; emergent behaviors from a carefully evolved network", by Brooks, MIT AI Lab memo, 1989.

Genghis

Generic Classification of Robot Behaviors

- Whatever the design basis, general categories of ways that robotic agent can interact with the world are:

- Exploration/directional behaviors (more in a general direction
  - Heading-based
  - Wandering
- Goal-oriented appetitive behaviors (taxes-more towards an attractor)
  - Discrete object attractor
  - Area attractor
- Aversive/protective behaviors (prevent collision)
  - Avoid stationary objects
  - Elude moving objects
  - Aggression

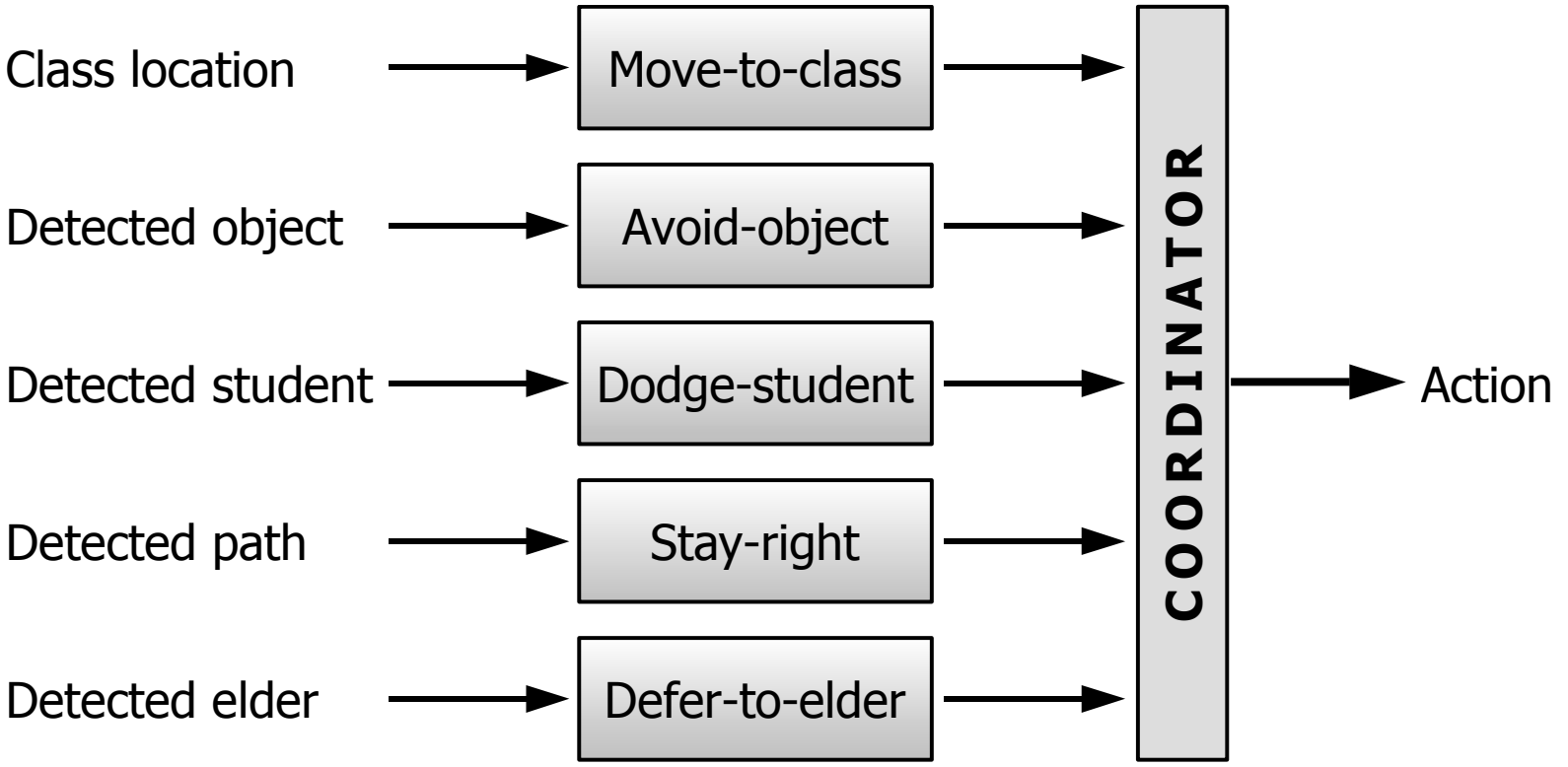
Generic Classification of Robot Behaviors (cont.)

- Path following behaviors (move on a designated path)
  - Road following
  - Hallway navigation
  - Stripe following
- Postural behaviors
  - Balance
  - Stability
- Social/cooperative behaviors
  - Sharing
  - Foraging
  - Flocking/herding
- Teleautonomous behaviors (coordinate with a human operator
  - Influence
  - Behavioral modification

Generic Classification of Robot Behaviors (cont.)

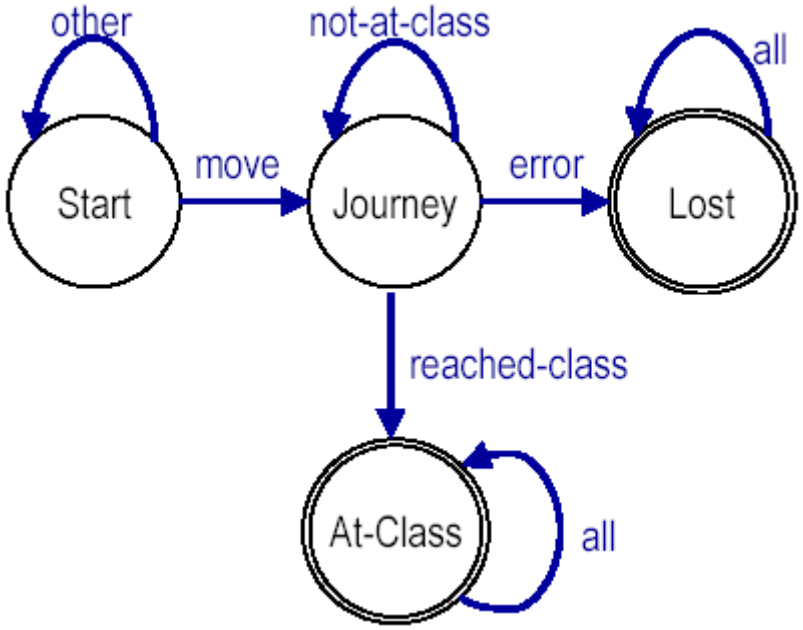
- Perceptual behaviors
  - Saccades
  - Visual search
  - Ocular reflexes
- Walking behaviors (for legged robots)
  - Gait control
- Manipulator-specific behaviors (for arm control)
  - Reaching
- Gripper/dexterous hand behaviors (for object acquisition)
  - Grasping
  - Enveloping

Stimulus-Response Diagram



Finite State Acceptor Diagram for Classroom Navigation Example

M = {{*start, journey, lost, at-class*}, δ, *start*, {*lost, at-class*}}



δ	*q*	*input*	δ(*q*,*input*)
	start	move	journey
	start	other	start
	journey	error	lost
	journey	not-at-class	journey
	journey	reached-class	at-class
	at-class	all	at-class
	lost	all	lost

NOTE: "Journey" consist of an assemblage of the five other low-level-behaviors mentioned earlier

(move-to-classroom, avoid-objects, dodge-students, stay-to-right-on-path, defer-to-elders)

Behavioral Encodings

- Behavioral encoding: creates functional mapping from the stimulus plane to the motor plane
- Behavior expressed as a triple: (S, R, β)

where:

S denotes domain of all interpretable stimuli
R denotes range of possible responses
β denotes mapping $\beta: S \rightarrow R$

Behavioral Encodings

- Behaviors can be represented as triples (S, R, β)

  $\beta(s) = r$, where:

  β = behavior

  s = stimulus

  r = response

- A strength multiplier, or gain g, can be used to turn off behaviors or alter the response's relative strength.

- Responses are encoded in two forms:
  - Discrete encoding
  - Continuous functional encoding

Functional Notation

- Mathematical methods used to describe relationships using a functional notation,

$$\beta(s) = r$$

where

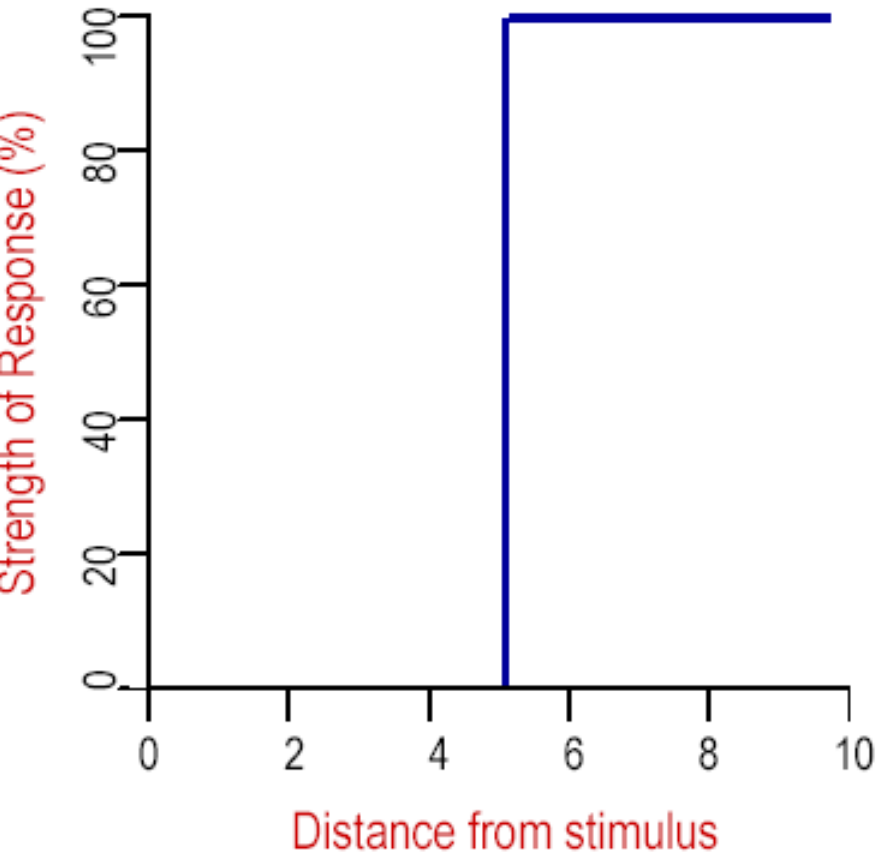
β = behavior
s = stimulus
r = response

- In purely reactive system, time is not an argument of β, since behavioral response is instantaneous and independent of system's history

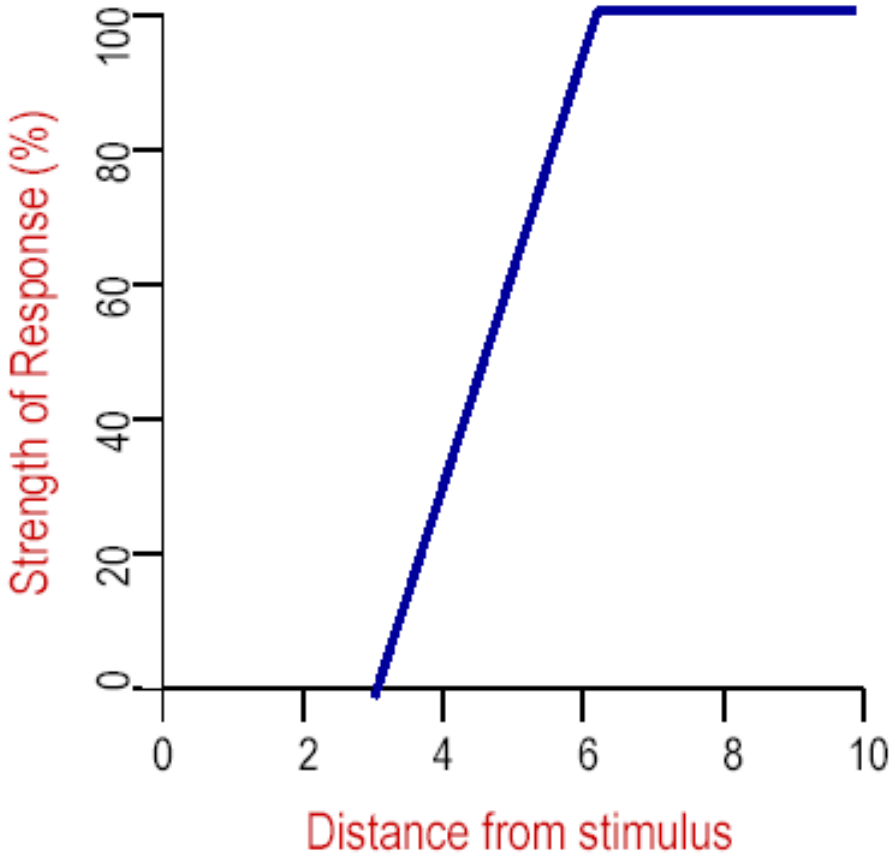
R: Range of Responses

- *R* denotes range of possible responses
- Motor response: consists of two orthogonal components:
- strength:
  - denotes magnitude of response
  - speed, force
- orientation:
  - denotes direction of action for the response
  - moving away from aversive stimulus, or towards an attractor

Stimulus/Response Strengths



Step function



Linear increase

Categories of Behavioral Mappings

- Null: Stimulus produces no motor response
- Discrete: Stimulus produces a response from an enumerable set of prescribed choices
  E. g.: turn-right, go-straight, stop, travel-at-speed 5
- Continuous: Stimulus produces a motor response that is continuous over *R*'s range

Discrete Encoding

- Situated action, β = finite set of (situation,response) pairs
  - (shortest-front < 5, turn-right-10)
- Rule-based systems using IF-THEN rules

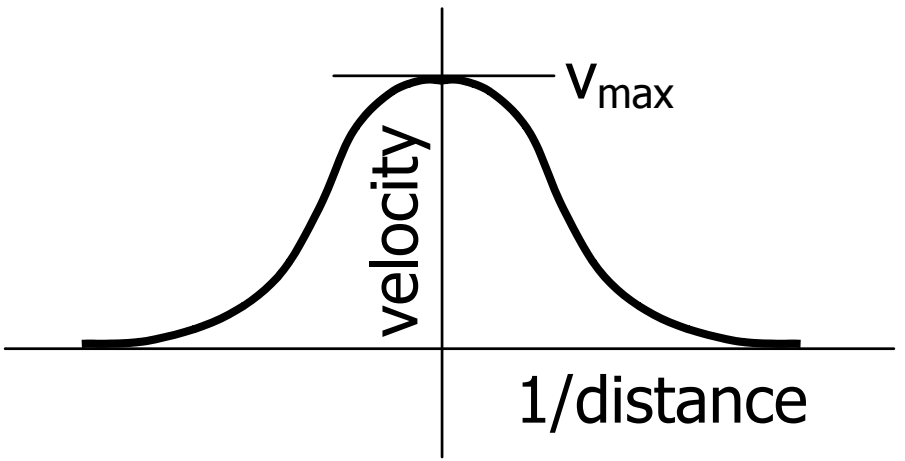
```
IF (shortest-front-distance > 20)
        omega=0

ELSE IF (shortest-front-heading < -30)
        omega=10

    ELSE IF (shortest-front-heading < 0)
        omega=20

        ELSE IF (shortest-front-heading < 30)
          omega=-20

           ELSE

          omega=-10
```

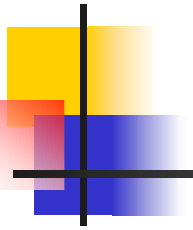
Continuous Functional Encoding

- Continuous function transforms sensory input into behavioral reaction
- Common approach: inverse square law:

  Force is proportional to: $1/distance^2$, maximizing at some maximum force

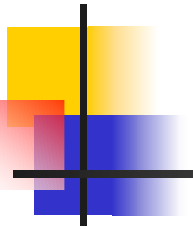
Example:





Example Behaviors

- Move
  - Moves at constant velocity
- Goto Goal
  - Moves towards a goal point (light source)
- Stop
  - Slows down near obstacles
- Avoid
  - Avoids running into obstacles



Move

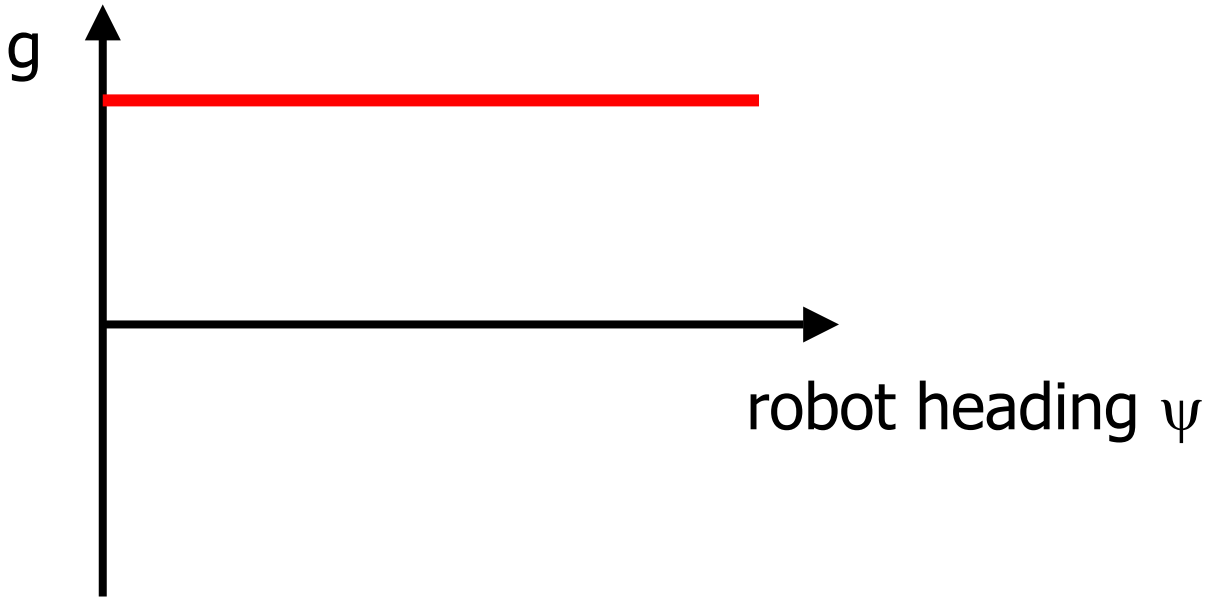
- Stimulus
  - none
- Response
  - constant translational velocity
- Strength
  - constant

Goto Goal

- Stimulus
  - desired heading (light source)
- Response
  - Turn rate ω
- Strength
  - constant

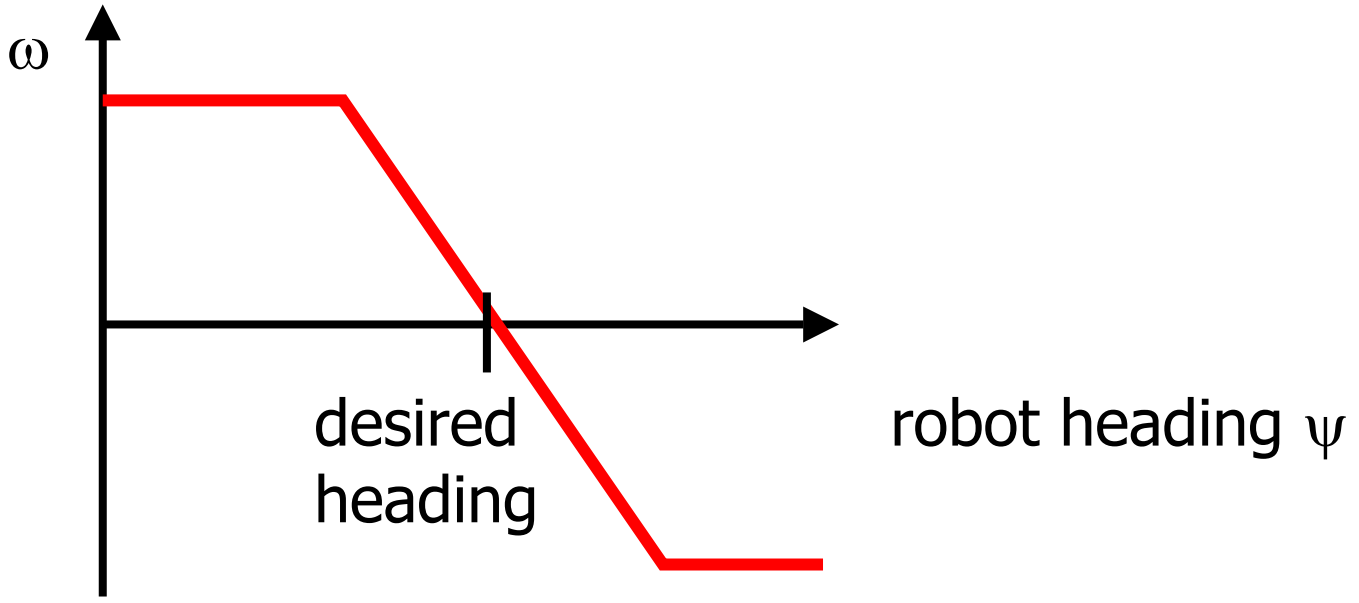
Goto Goal

- behavior strength



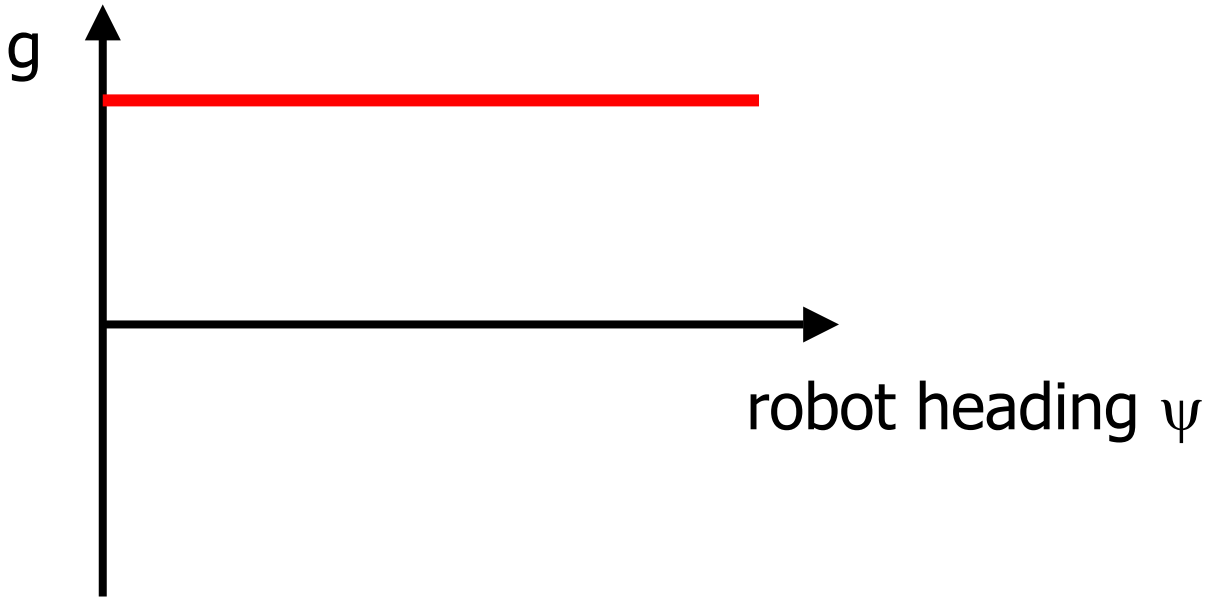
Goto Goal

- Turn rate, continuous encoding



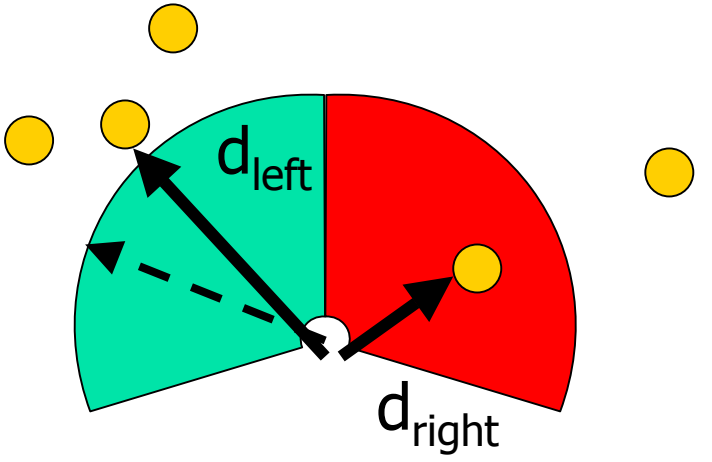
Goto Goal

- behavior strength



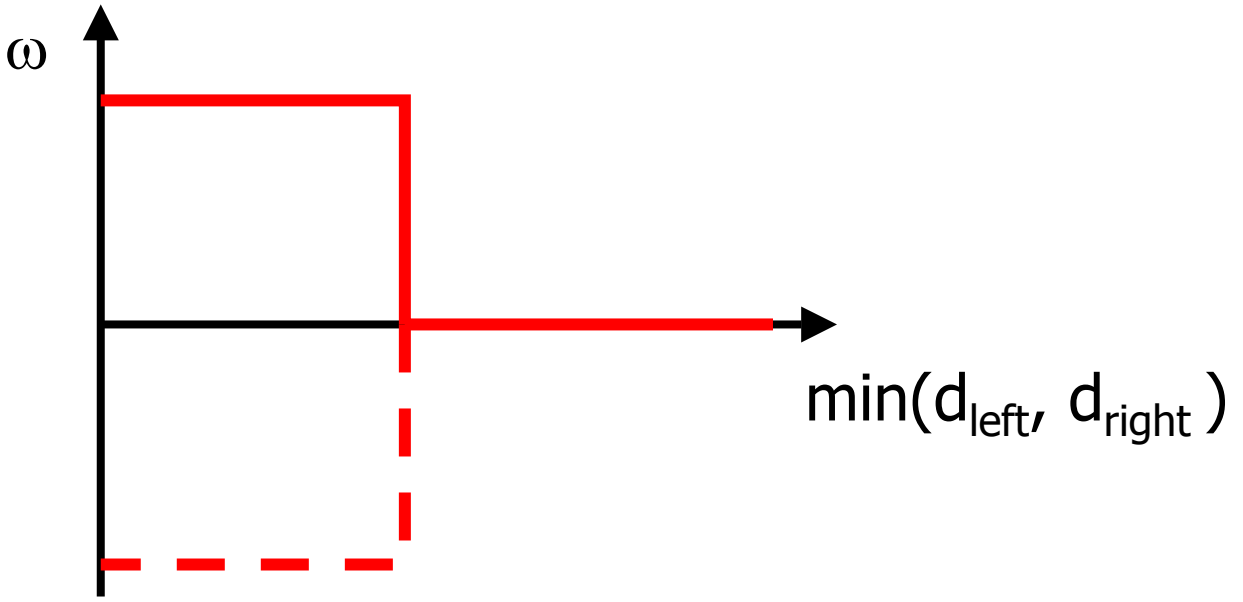
Avoid

- Stimulus
  - distance to nearest left and right obstacle d_{left}, d_{right}
- Response
  - turn rate ω
- Internal state
  - last turning direction
- Strength
  - decreases with increasing distance to obstacle



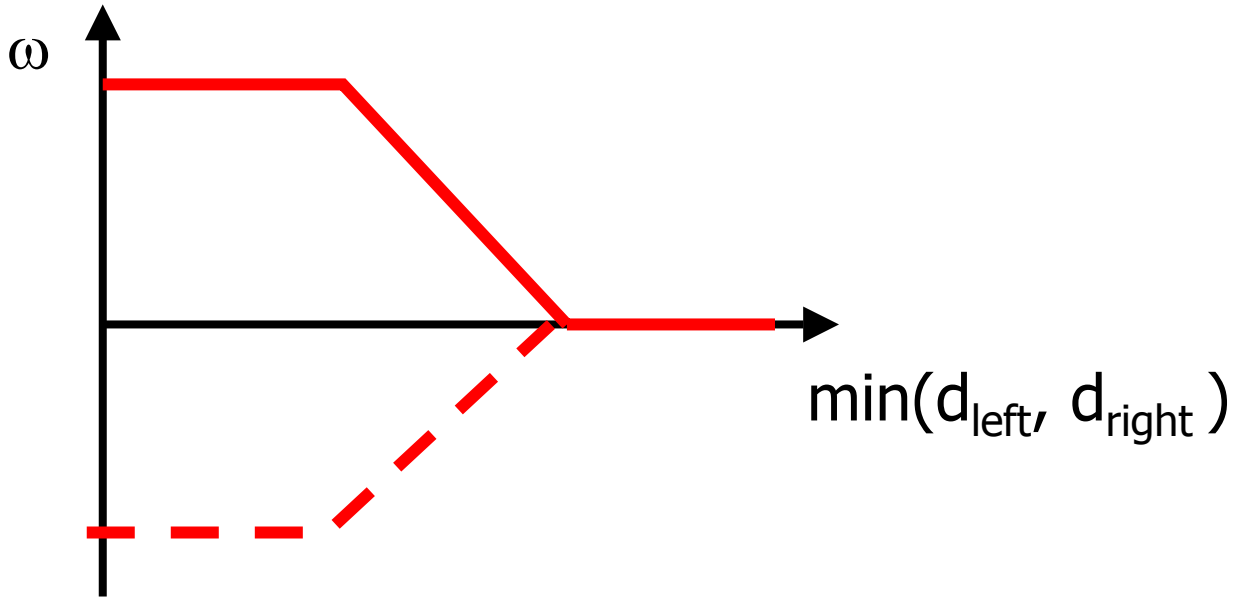
Avoid

- Turn rate, discrete encoding



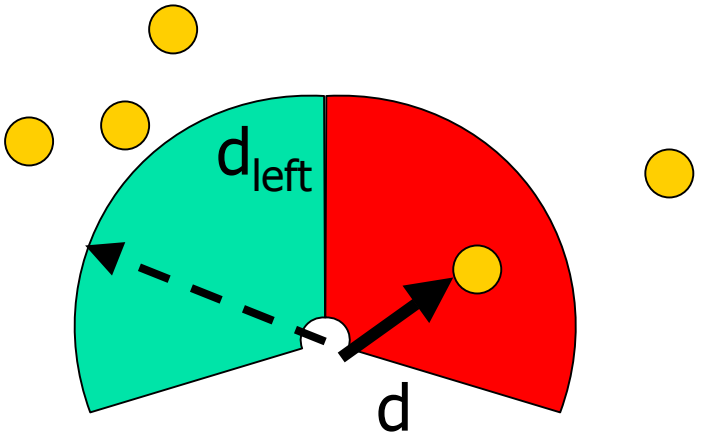
Avoid

- Turn rate, continuous encoding



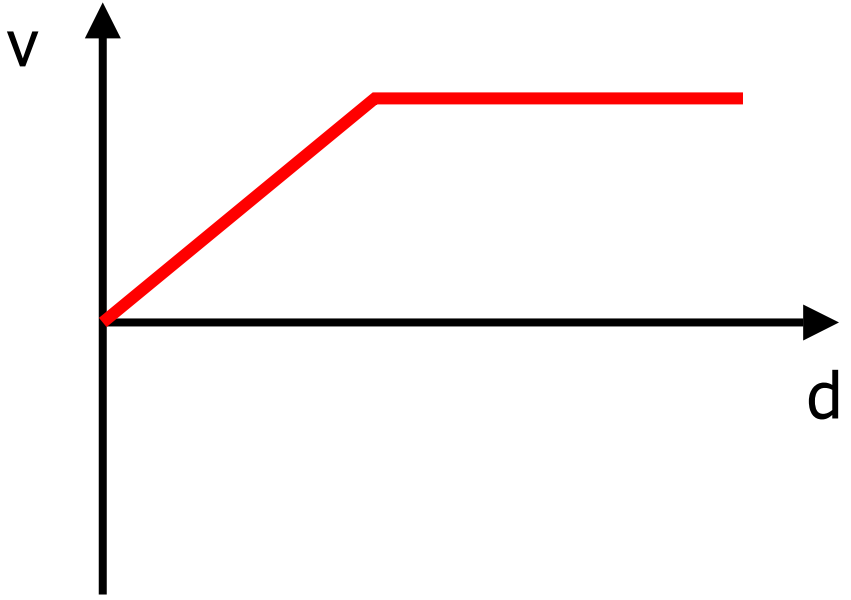
Stop

- Stimulus
  - distance to nearest obstacle d
- Response
  - translational velocity v
- Strength
  - decreases with increasing distance to obstacle



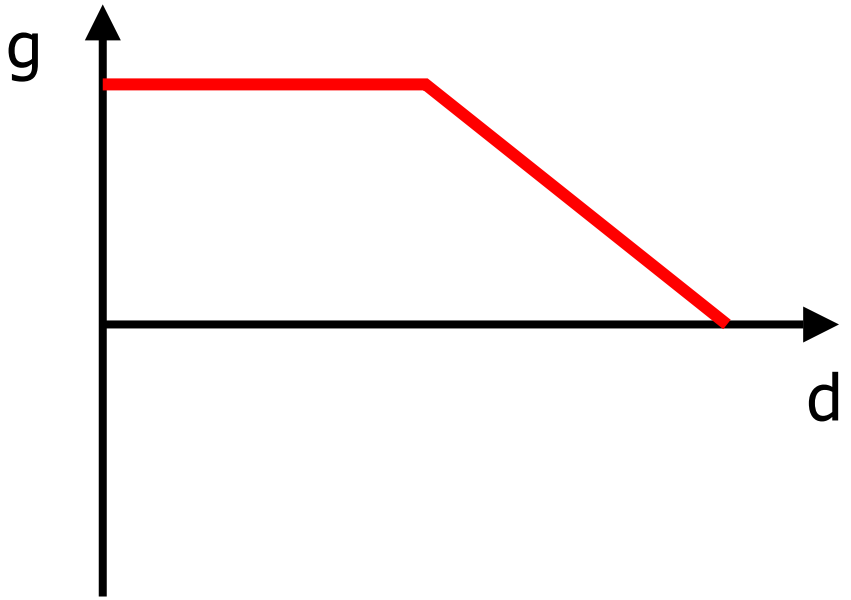
Stop

- translational velocity



Stop

- behavior strength



Obstacle Avoidance

Parameters:

myTurnThreshold : distance at which to start avoiding

myTurnAmount : magnitude of turn rate for avoiding obstacles

myTurning : direction of last turn (-1, 0 , 1)

```
// Get the left readings and right readings off of the sonar
  leftRange = (mySonar->currentReadingPolar(0, 100) -
            myRobot->getRobotRadius());
  rightRange = (mySonar->currentReadingPolar(-100, 0) -
             myRobot->getRobotRadius());
```

Obstacle Avoidance

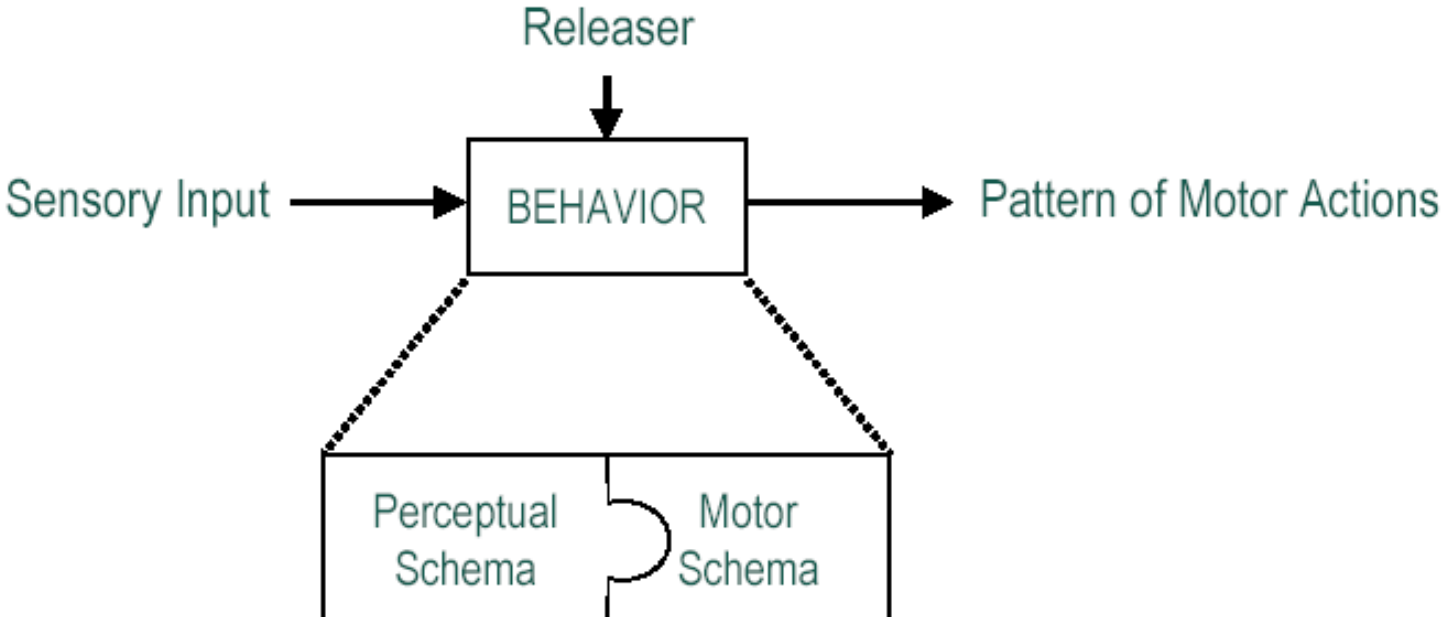
```
// if neither left nor right range is within the turn threshold,
  // reset the turning variable and don't turn

  if (leftRange > myTurnThreshold && rightRange > myTurnThreshold)
  {
    myTurning = 0;
    myDesired.setDeltaHeading(0);
  }
  // if we're already turning some direction, keep turning that direction
  else if (myTurning)
  {
    myDesired.setDeltaHeading(myTurnAmount * myTurning);
  }
```

Obstacle Avoidance

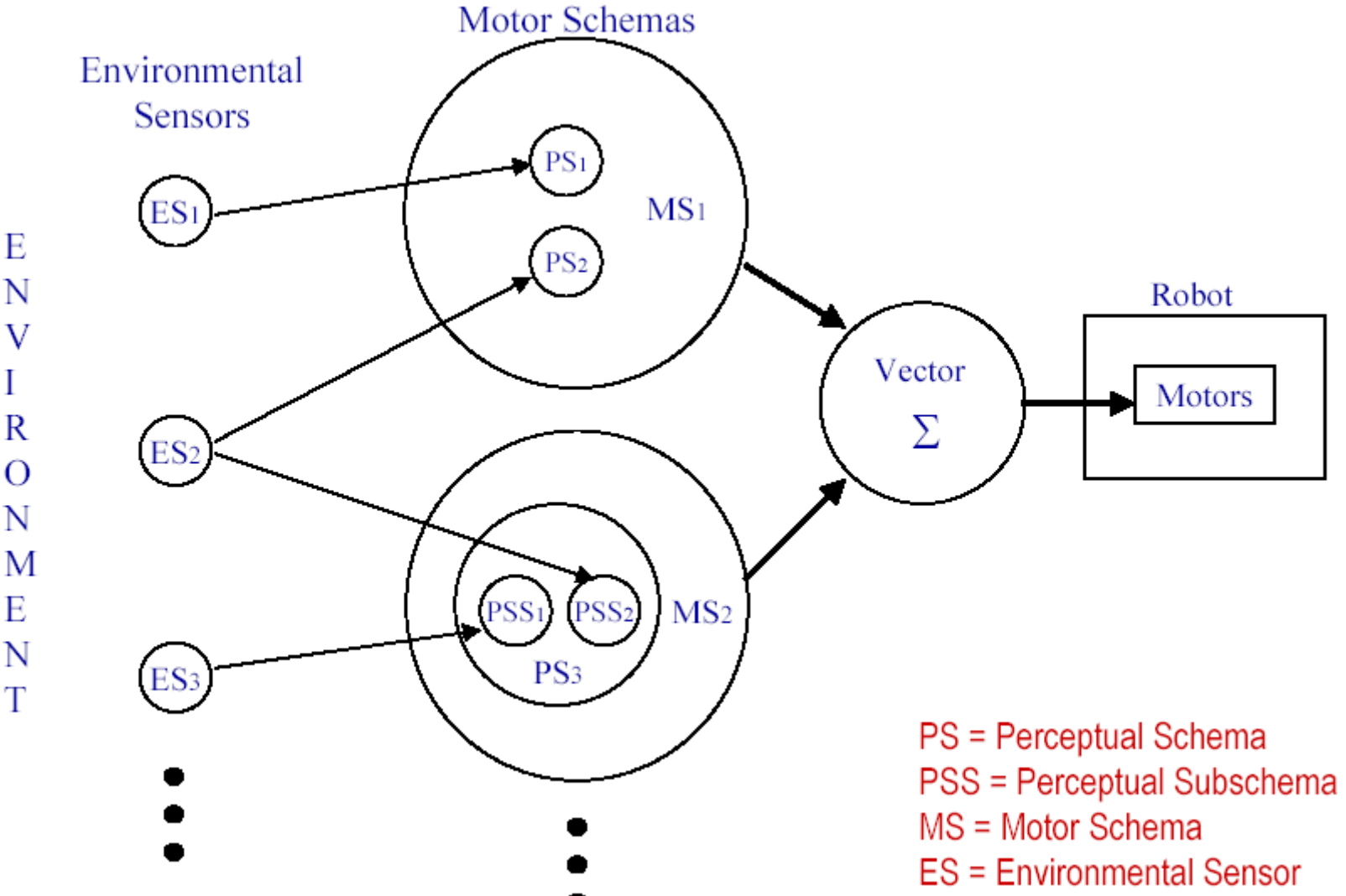
```
// if we're not turning already, but need to, and left is closer, turn right
// and set the turning variable so we turn the same direction for as long as
// we need to
else if (leftRange < rightRange)
{
  myTurning = -1;
  myDesired.setDeltaHeading(myTurnAmount * myTurning);
}
// if we're not turning already, but need to, and right is closer, turn left
// and set the turning variable so we turn the same direction for as long as
// we need to
else
{
  myTurning = 1;
  myDesired.setDeltaHeading(myTurnAmount * myTurning);
}
```

Motor Schemas

- Developed by Arkin in 1980s
- Based on biology's schema theory
- Behavioral responses are all represented as vectors generated using a potential fields approach
- Coordination is achieved by vector addition



Perception-Action Schema Relationships



Potential Fields

Introduction to Potential Fields:

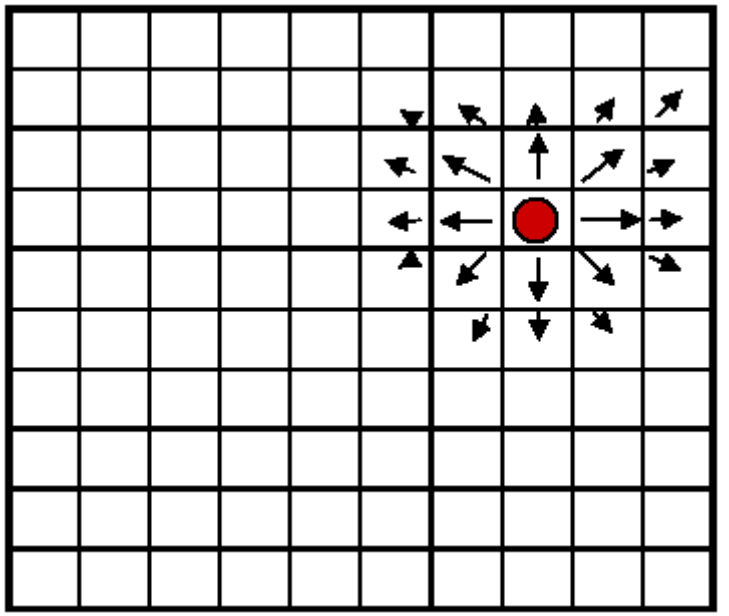
- Potential field: array (or field) of vectors representing space
- Vector **v** = (*m*,*d*)*:* consists of magnitude (*m*) and direction (*d*)
- Vector represents a force

Length of arrow = *m* = magnitude

Angle of arrow = *d* = direction

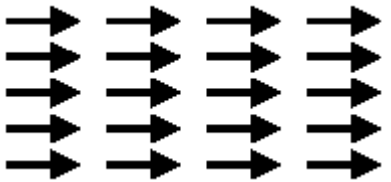
Potential Fields

- Vector space is 2D world
- Map divided into squares, creating (x,y) grid
- Each element represents square of space
- Perceivable objects in world exert a force field on surrounding space

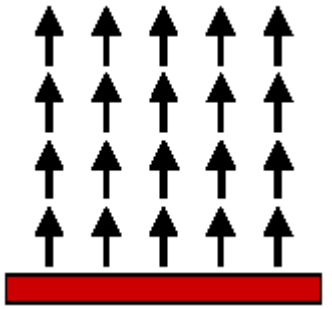


Five Primitive Potential Fields

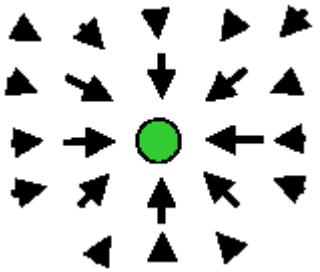
Uniform



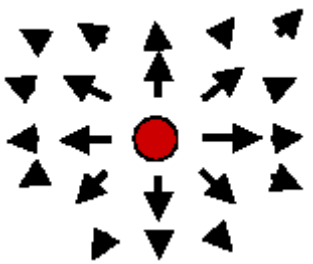
Perpendicular



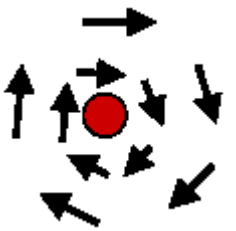
Attraction



Repulsion

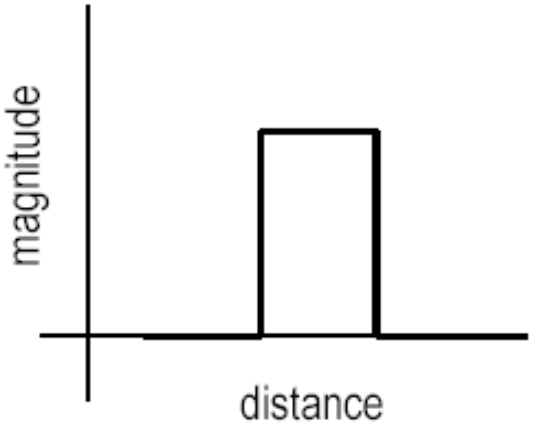


Tangential

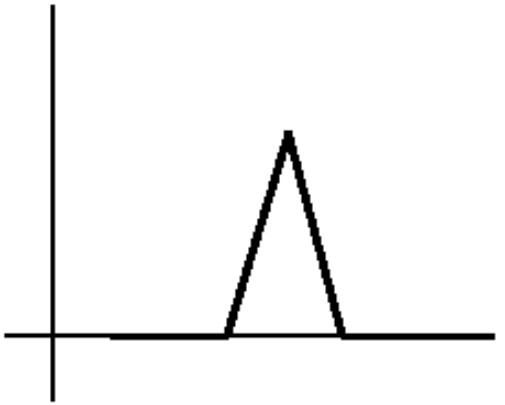


Magnitude Profiles

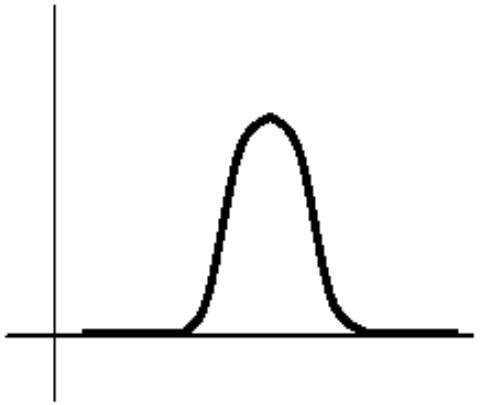
- Change in magnitude as a function of distance from the center of the field



Constant



Linear Dropoff



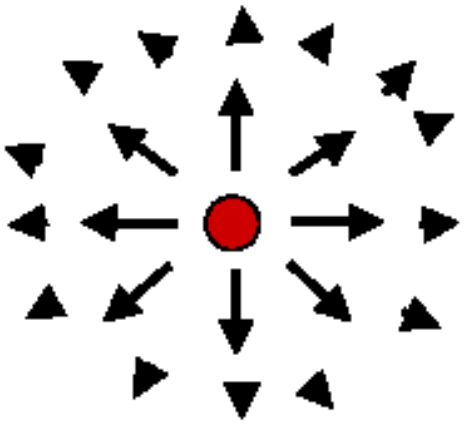
Exponential Dropoff

Single Potential Field

- Repulsive field with linear drop-off:

$$V_{direction}(\vec{x}) = \frac{\vec{x} - \vec{x}_0}{|\vec{x} - \vec{x}_0|}$$

$$V_{magnitude}(\vec{x}) = \begin{cases} \dfrac{D - |\vec{x} - \vec{x}_0|}{D} & \text{for } d \le D \\ 0 & \text{for } d > D \end{cases}$$



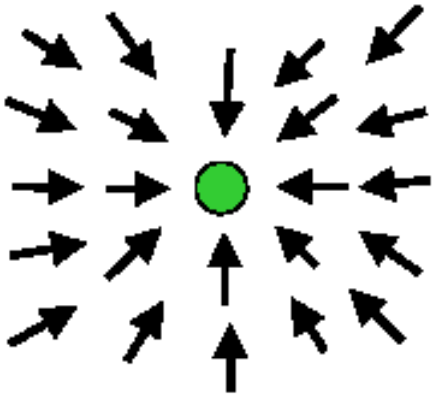
where *D* is max range of field's effect

Motor Schema Encodings

- Move-to-goal (ballistic):

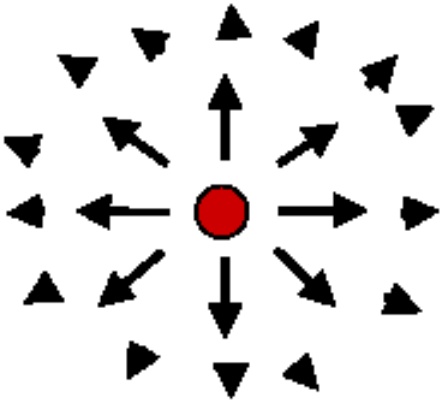
  $V_{magnitude}$ = fixed gain value

  $V_{direction}$ = towards perceived goal



- Avoid-static-obstacle:

$$V_{magnitude}(\vec{x}) = \begin{cases} 0 & \text{for } |\vec{x} - \vec{x}_0| > S \\ \dfrac{S - |\vec{x} - \vec{x}_0|}{S - R} G & \text{for } R < |\vec{x} - \vec{x}_0| \leq S \\ \infty & \text{for } |\vec{x} - \vec{x}_0| \leq R \end{cases}$$



where S = sphere of influence of obstacle

R = radius of obstacle

G = gain

D = distance of robot to center of obstacle

Motor Schema Encodings

- Stay-on-path:

$$V_{magnitude} = \begin{cases} P & \text{for } d > W/2 \\ \dfrac{d}{W/2} & \text{for } d \le W/2 \end{cases}$$

where:

W = width of path

P = off-path gain

G = on-path gain

d = distance of robot to center of path

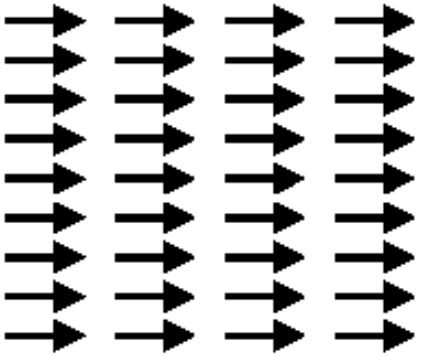
$V_{direction}$ = along a line from robot to center of path, heading toward centerline

Motor Schema Encodings

- Move-ahead:

  $V_{magnitude}$ = fixed gain value

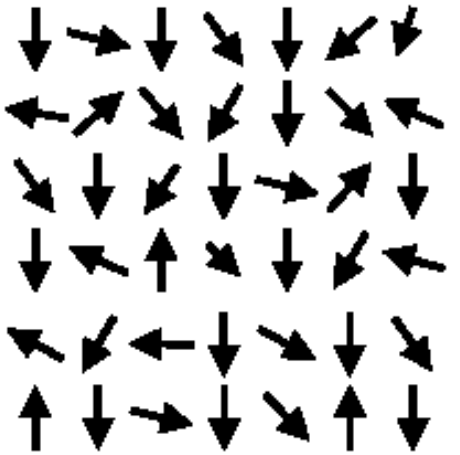
  $V_{direction}$ = specified compass direction



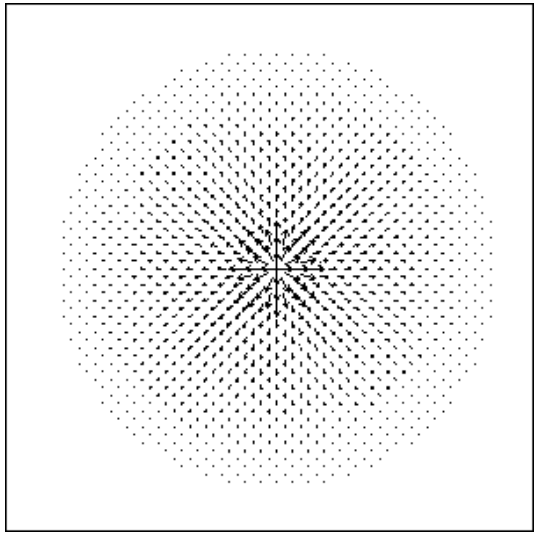
- Noise:

$V_{magnitude}$ = fixed gain value

$V_{direction}$ = random direction changed every p time steps

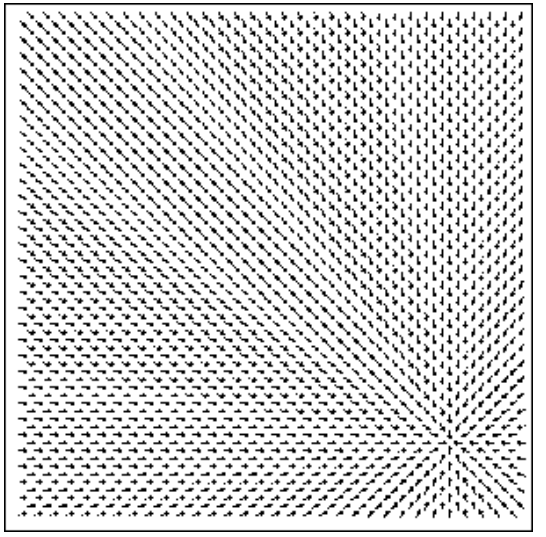


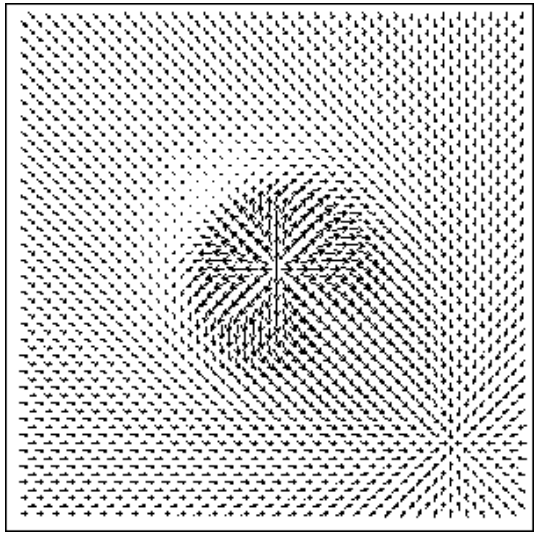
Motor Schema Encodings



avoid obstacle

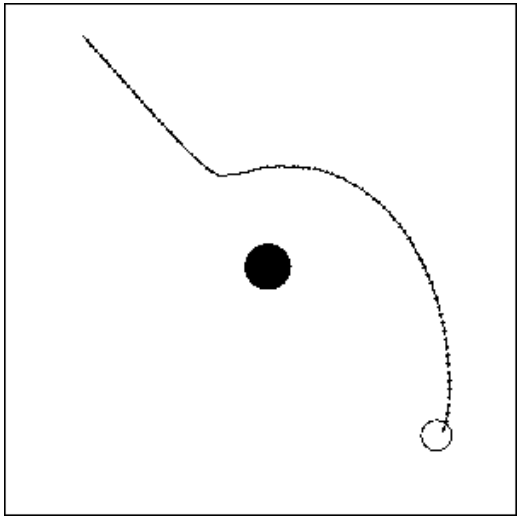
move to goal





overall vector field

resulting trajectory



Summary of Robot Behavior

- Robotic behaviors generate a motor response from a given perceptual stimulus
- Purely reactive systems avoid the use of explicit representational knowledge
- Three design paradigms:
  - Ethologically guided/constrained
  - Situated activity
  - Experimentally driven
- Expression of behaviors can be accomplished in several ways
  - SR diagrams
  - Functional notation
  - FSA diagrams
- Behaviors can be represented as triples (S, R, β)