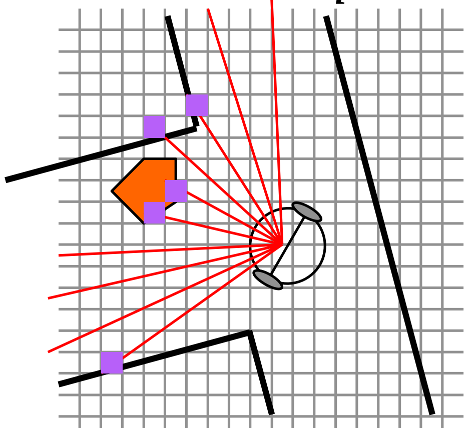
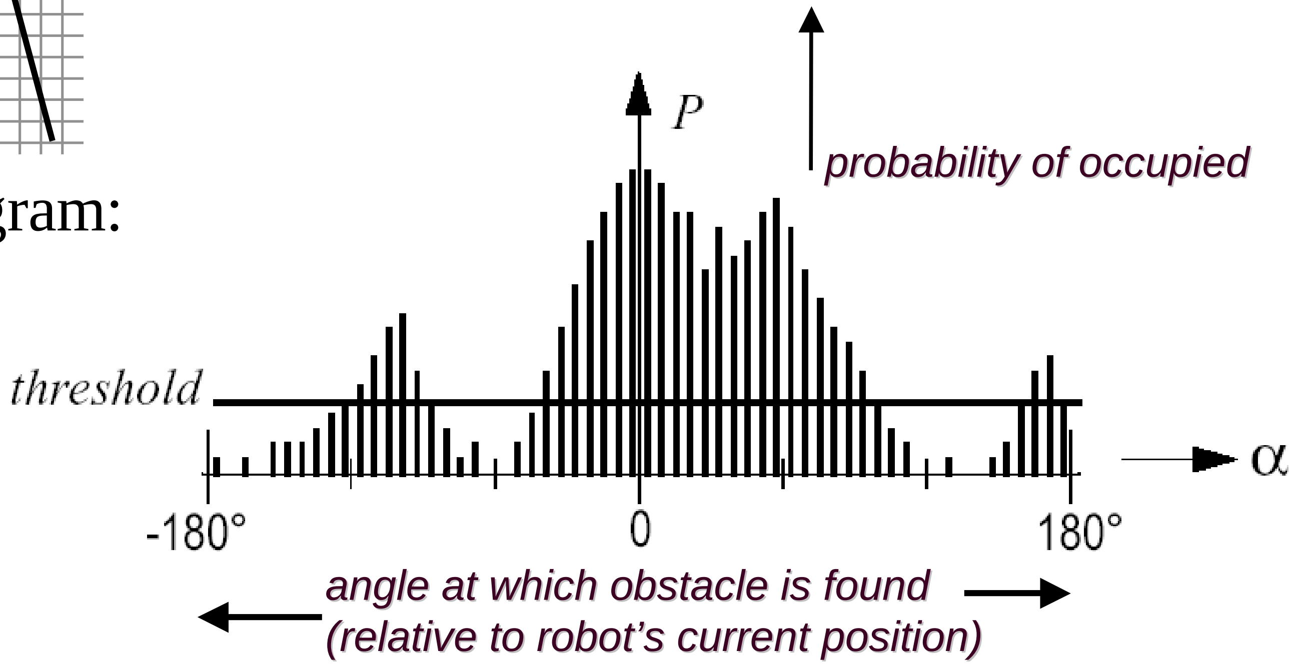
Recap: Vector Field Histogram (VFH)

Koren & Borenstein, ICRA 1990

- Environment represented in a grid (2 DOF)
  - *cell values are equivalent to the probability that there is an obstacle*



- Generate polar histogram:



probability of occupied

angle at which obstacle is found (relative to robot’s current position)

Adapted from © R. Siegwart, I. Nourbakhsh

Recap: Vector Field Histogram (VFH)

Koren & Borenstein, ICRA 1990

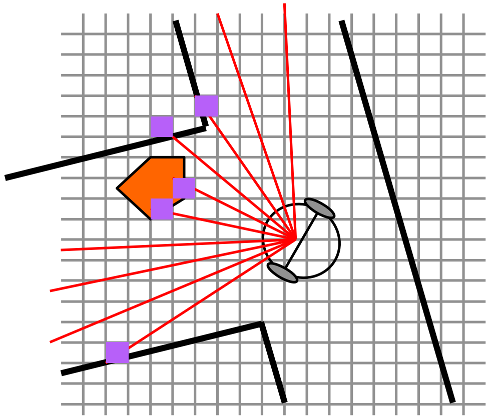
- From histogram, calculate steering direction:
  - ➢ *Find all openings large enough for the robot to pass through*
  - ➢ *Apply cost function G to each opening*

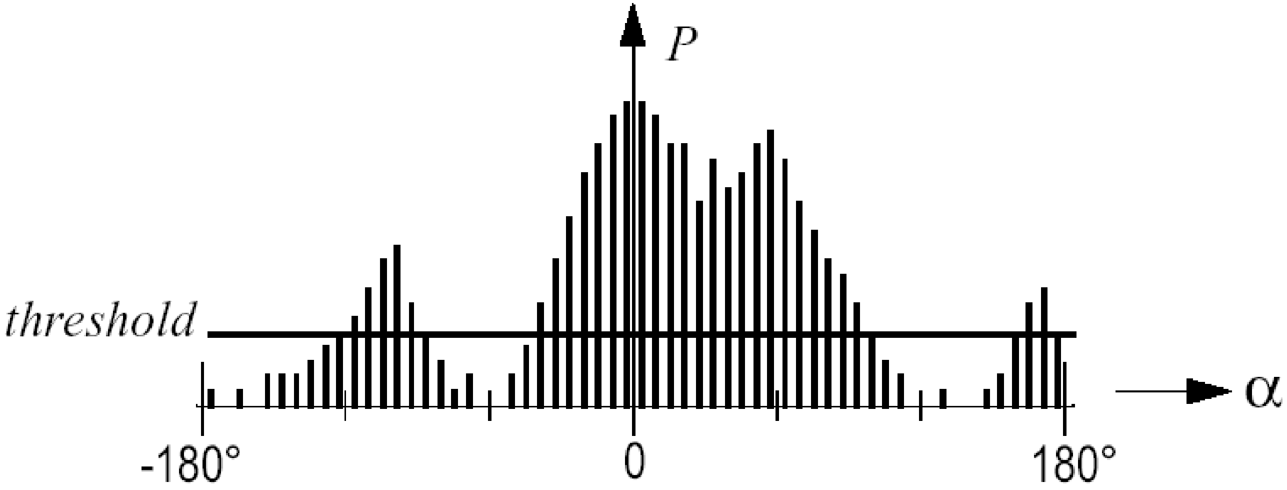
$$G = a \cdot \text{target\_direction} + b \cdot \text{wheel\_orientation} + c \cdot \text{previous\_direction}$$

where:

- o *target_direction = alignment of robot path with goal*
- o *wheel_orientation = difference between new direction and current wheel orientation*
- o *previous_direction = difference between previously selected direction and new direction*

- ➢ *Choose the opening with lowest cost function value*



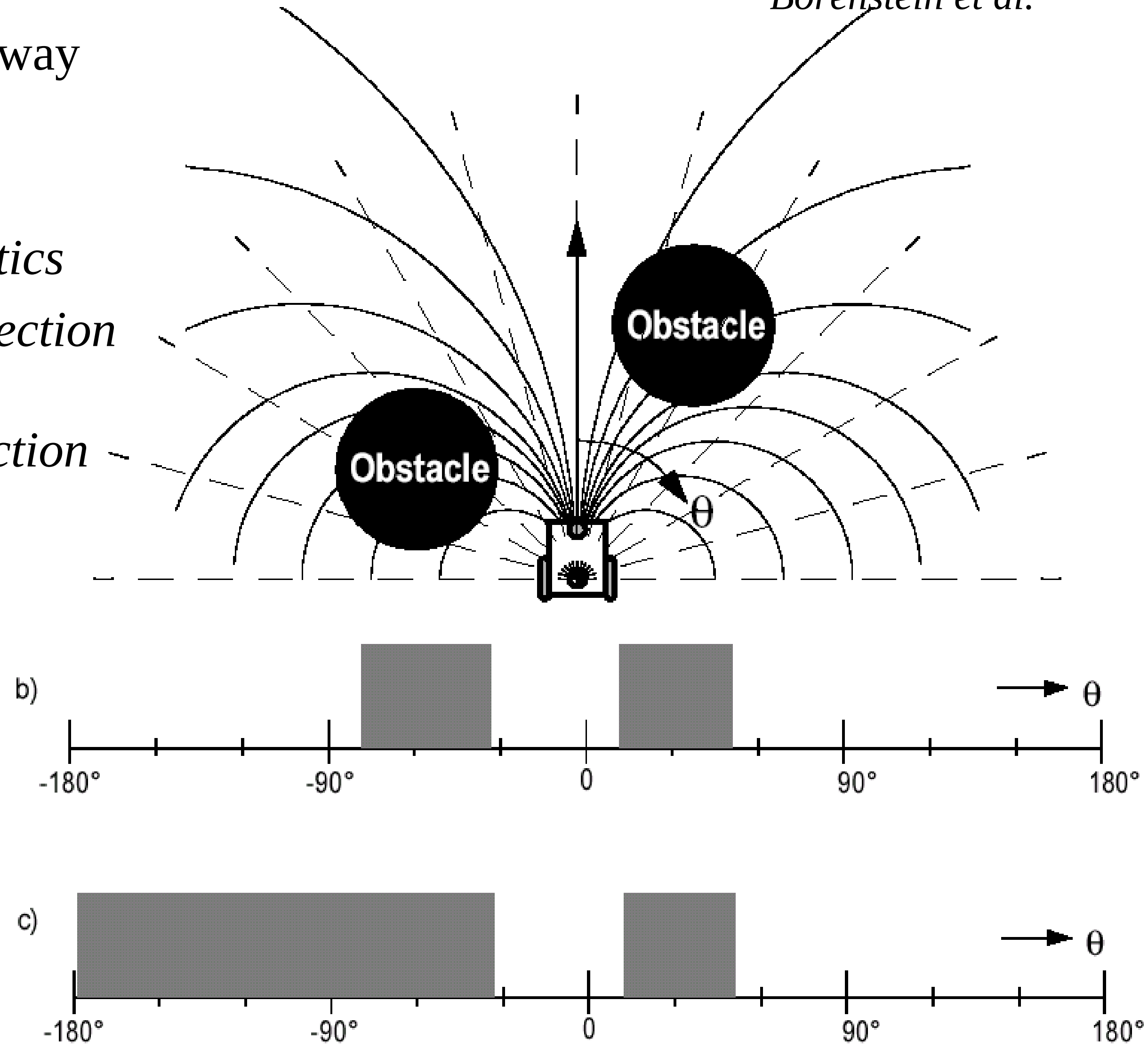


Adapted from © R. Siegwart, I. Nourbakhsh

Obstacle Avoidance: **Vector Field Histogram $^+$ (VFH+)**

- Accounts also in a very simplified way for the moving trajectories
  - *robot can move on arcs*
  - *arcs take into account kinematics*
  - *obstacles blocking a given direction also block all the trajectories (arcs) going through this direction*

Borenstein et al.



Caprari et al. 2002

Adapted from © R. Siegwart, I. Nourbakhsh

Obstacle Avoidance: **VFH**

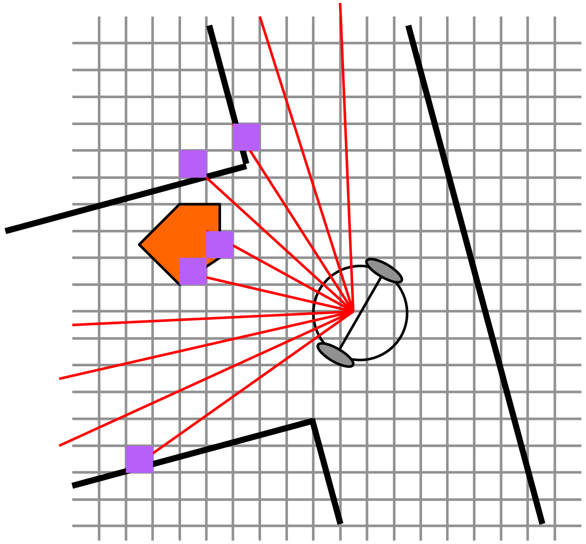
Borenstein et al.

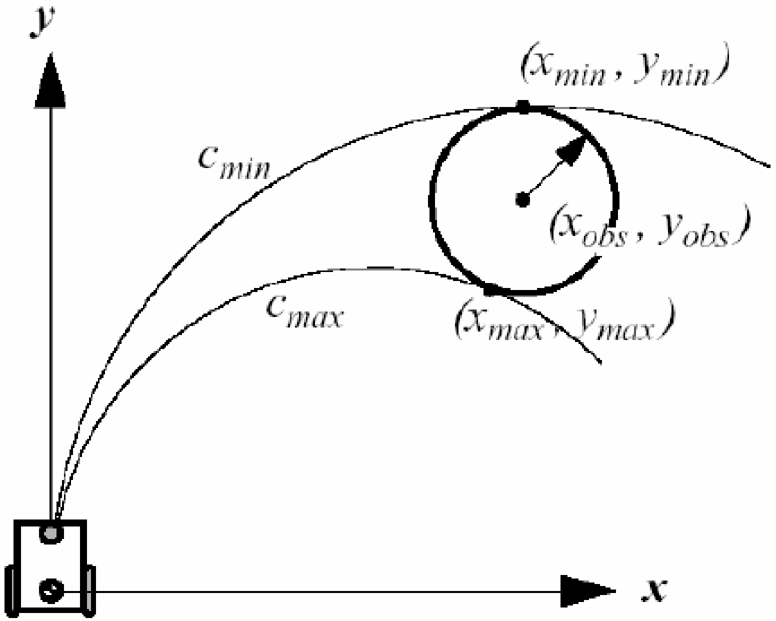
- Limitations:
  - *Can be problematic if narrow areas (e.g. doors) have to be passed*
  - *Local minima might not be avoided*
  - *Reaching the goal cannot be guaranteed*
  - *Dynamics of the robot not really considered*

Obstacle Avoidance: **Basic Curvature Velocity Methods** (CVM)

Simmons et al.

- Adding *physical constraints* from the robot and the environment on the *velocity space* (v, ω) of the robot
  - *Assumption that robot is traveling on arcs* $(c = \omega / v)$
  - *Constraints:* $-v_{max} < v < v_{max} \qquad -\omega_{max} < \omega < \omega_{max}$
  - *Obstacle constraints: Obstacles are transformed in velocity space*
  - *Objective function used to select the optimal speed*





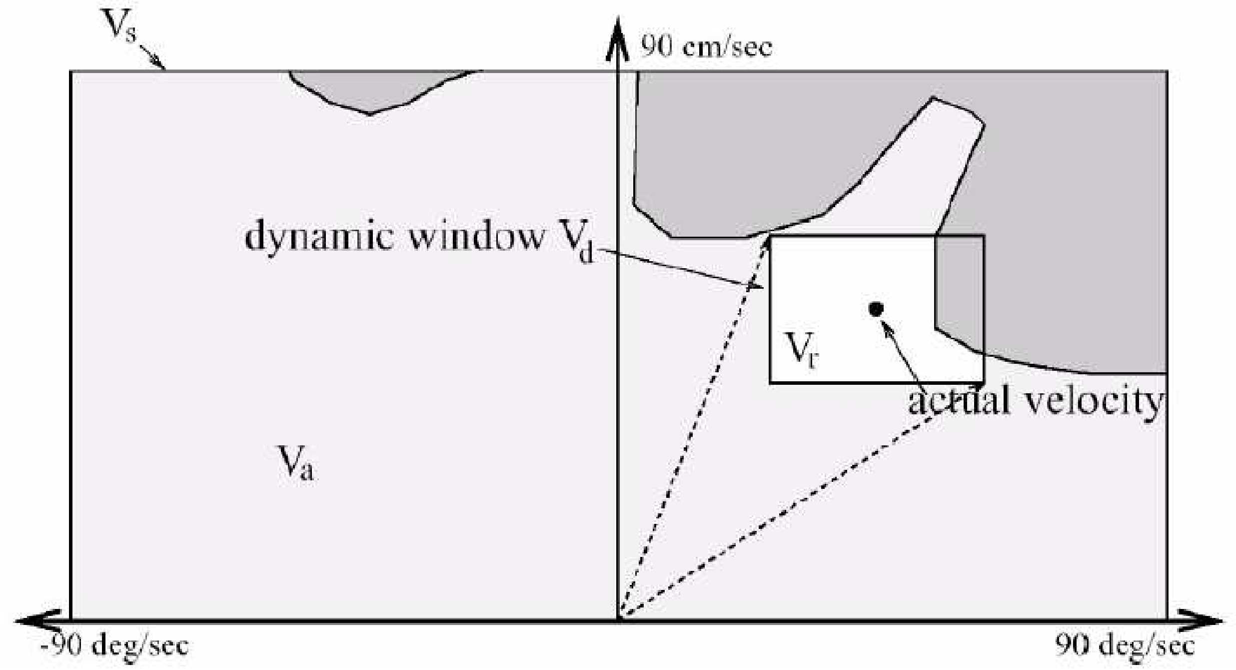
rom © R. Siegwart, I. Nourbakhsh

Obstacle Avoidance: **Dynamic Window Approach**

Fox and Burgard, Brock and Khatib

- The kinematics of the robot is considered by searching a well chosen velocity space
  - *velocity space -> some sort of configuration space*
  - *robot is assumed to move on arcs*
  - *ensures that the robot comes to stop before hitting an obstacle*
  - *objective function is chosen to select the optimal velocity*

$$O = a \cdot heading(v, \omega) + b \cdot velocity(v, \omega) + c \cdot dist(v, \omega)$$

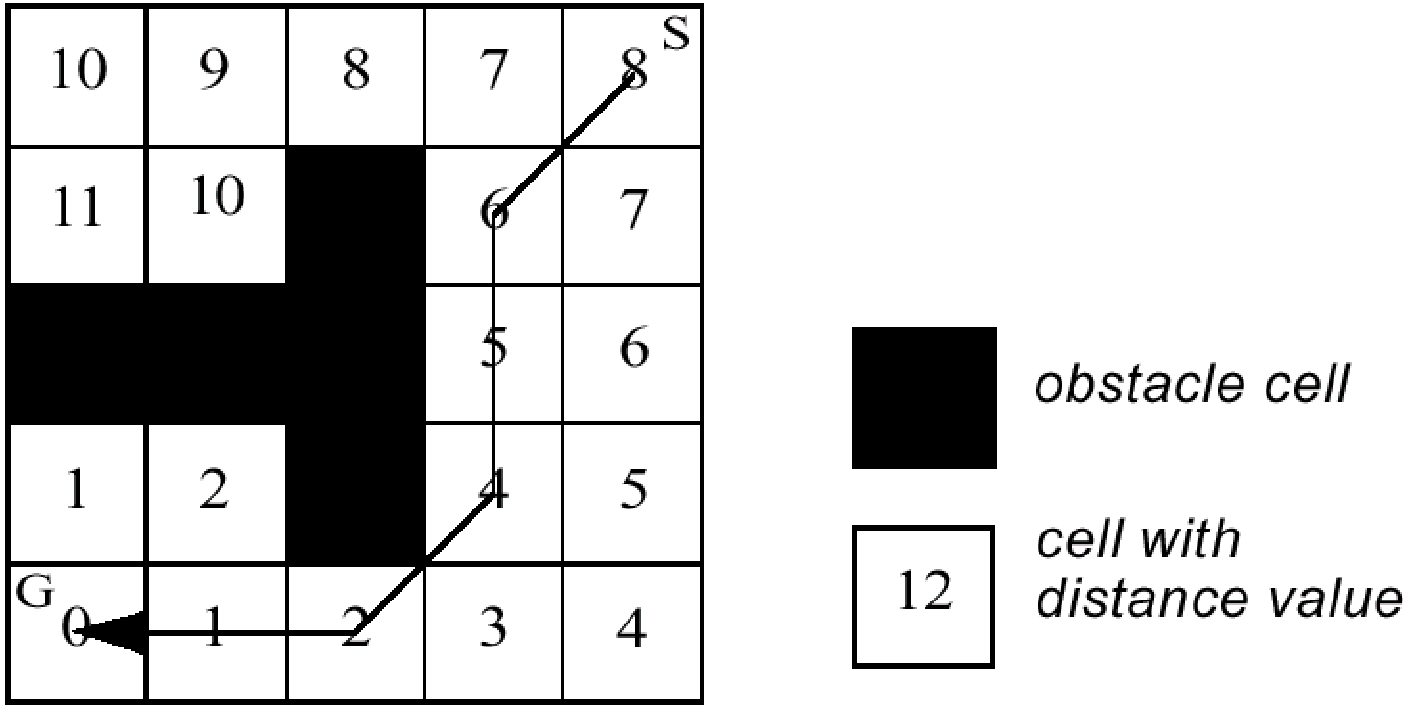


- *heading* = progress toward goal
- *velocity* = forward velocity of robot (encourages fast movements)
- *dist* = distance to closest obstacle in trajectory

Adapted from © R. Siegwart, I. Nourbakhsh

Obstacle Avoidance: **Global Dynamic Window Approach**

- Global approach:
  - ➢ *This is done by adding a minima-free function named NF1 (wave-propagation) to the objective function* O *presented above.*
  - ➢ *Occupancy grid is updated from range measurements*



Adapted from © R. Siegwart, I. Nourbakhsh

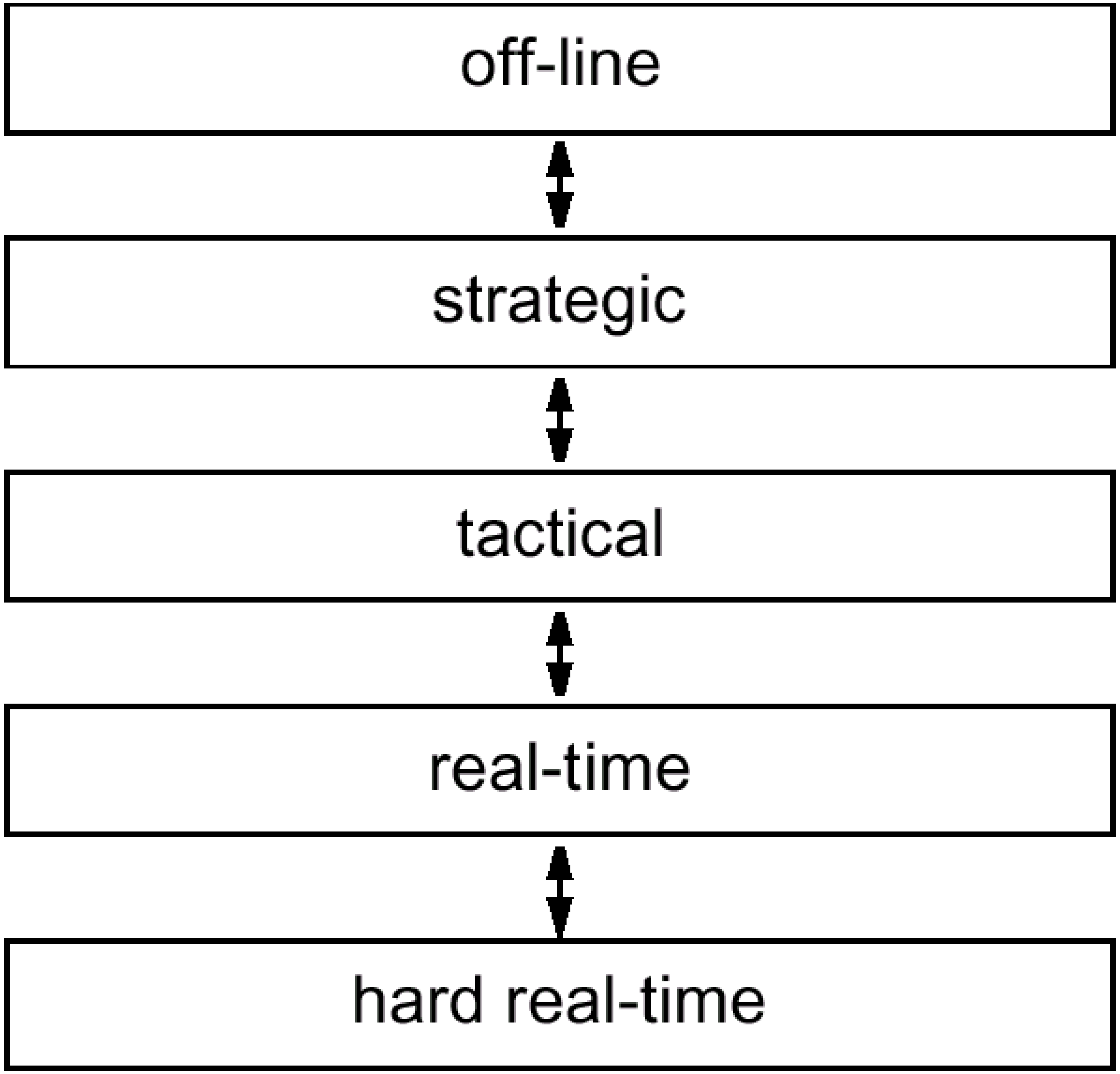
Now – let’s consider overall robot control architecture

- Need to combine hard real-time control with higher-level planning
  - ➔ *temporal decomposition*

- Need to combine multiple control capabilities (e.g., obstacle avoidance + go-to-goal)
  - ➔ *control decomposition*

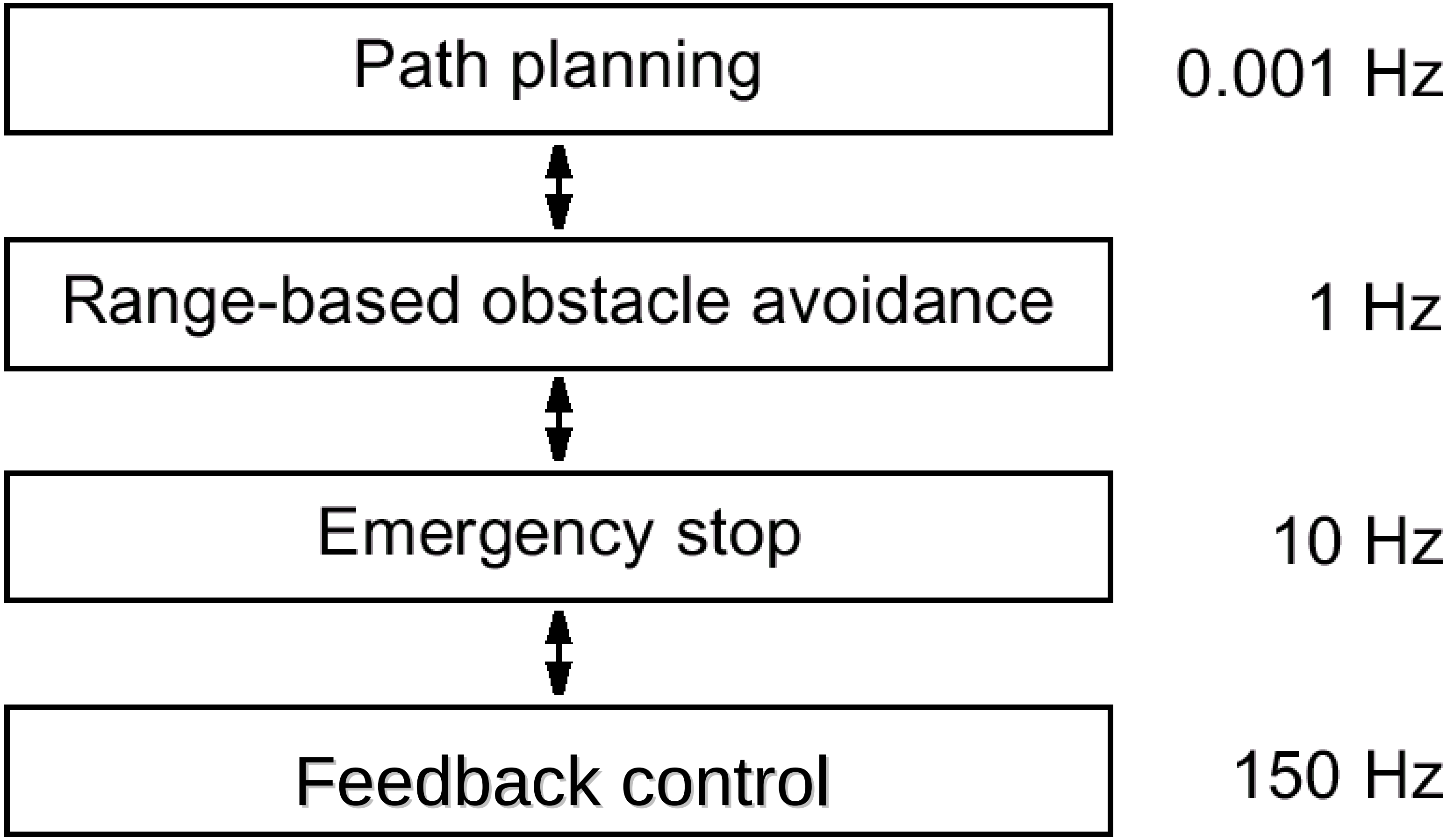
Adapted from © R. Siegwart, I. Nourbakhsh

Generic temporal decomposition



om © R. Siegwart, I. Nourbakhsh

4-level temporal decomposition



Path planning 0.001 Hz

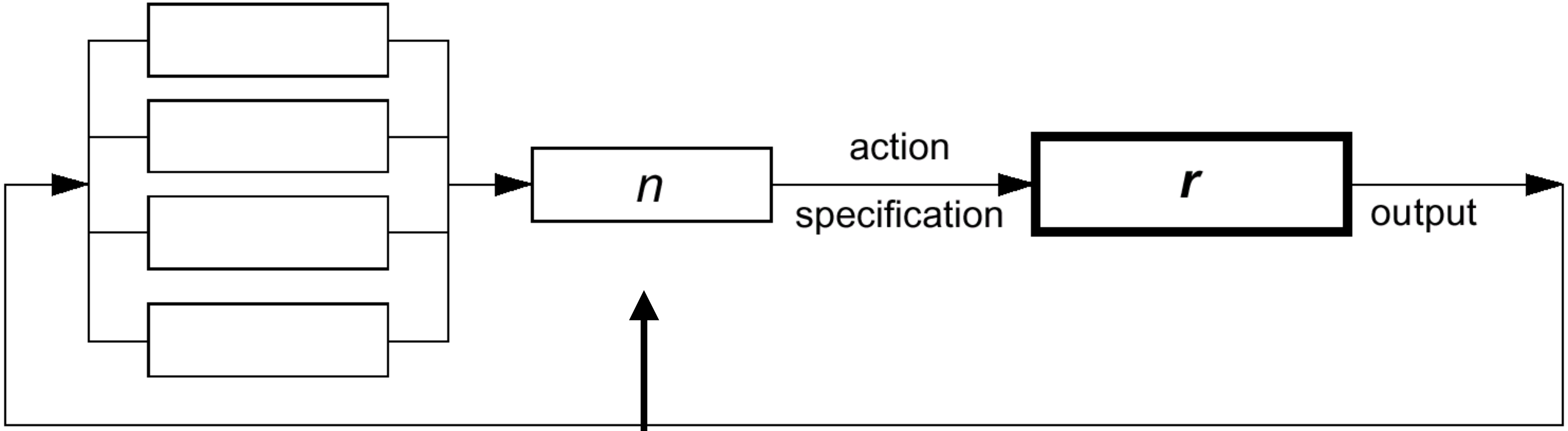
Range-based obstacle avoidance 1 Hz

Emergency stop 10 Hz

Feedback control 150 Hz

Control decomposition

- Parallel decomposition



decides how to combine outputs from multiple control actions

control actions (e.g., wall-follow, avoid obstacles, go-to-goal)

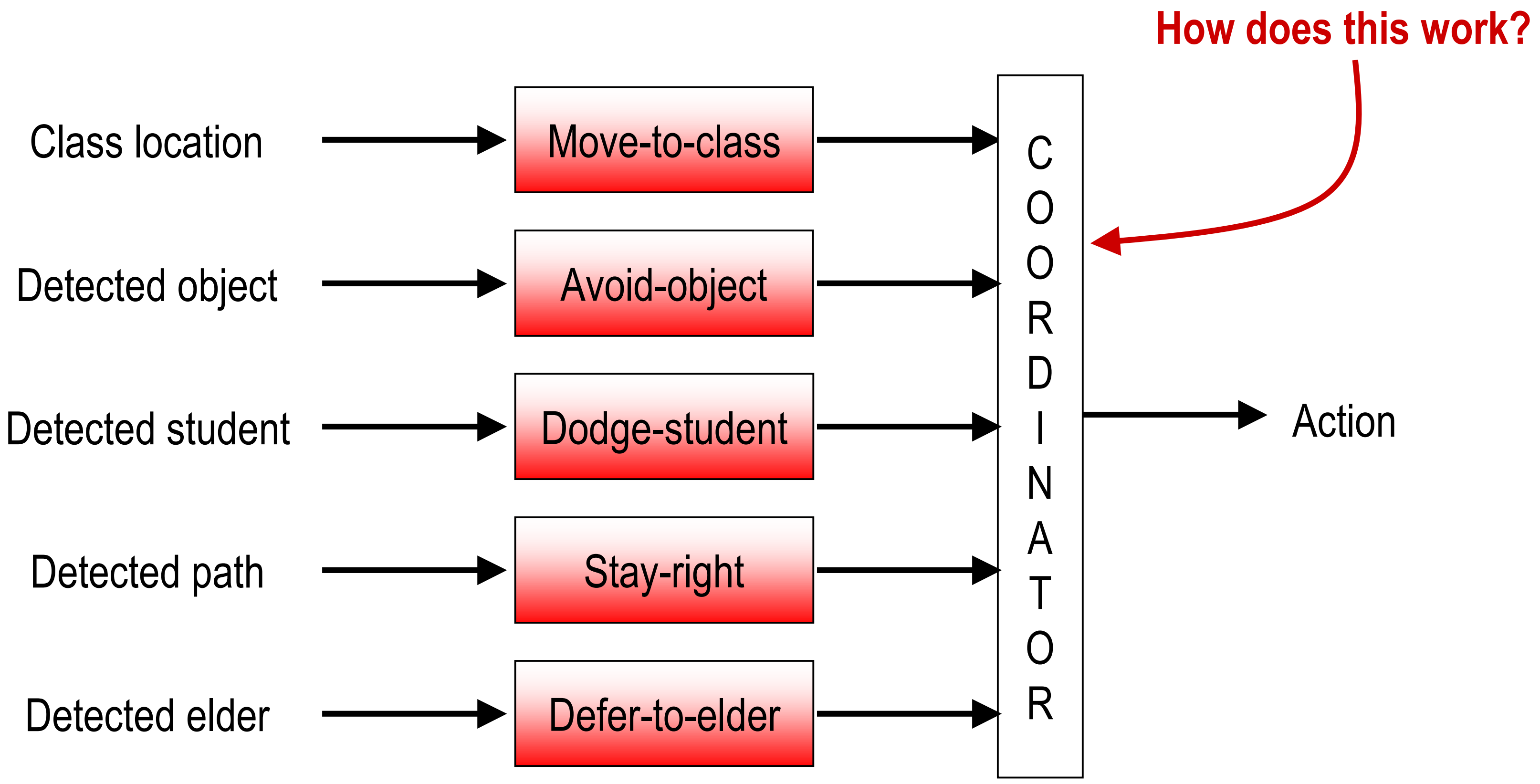
Adapted from © R. Siegwart, I. Nourbakhsh

Key question: How to combine multiple control modules?

- Consider a “Classroom Navigation Example”:
- Here, student is going from one room to another. What is involved?
  - *Getting to your destination from your current location*
  - *Not bumping into anything along the way*
  - *Skillfully negotiating your way around other students who may have the same or different intentions*
  - *Observing cultural idiosyncrasies (e.g., deferring to someone of higher priority – age, rank, etc.; or passing on the right (in the U.S.), …)*
  - *Coping with change and doing whatever else is necessary*

Assembling Multiple Control Modules

- Issue: When have multiple behaviors, how do we combine them?
- Think about in terms of: Navigation example



Adapted from © R. Siegwart, I. Nourbakhsh

Key question: How to combine multiple control modules?

- Switched parallel: at any instant, the output is from one specific module
  - *Advantage: If switching is rare, then it is easy to characterize result*
  - *Disadvantage: If switching is frequent, resulting robot behavior may be unstable*

- Mixed parallel: at any instant, control is shared between multiple modules
  - *Advantages: More general, can achieve multiple objectives at once*
  - *Disadvantages: More difficult to predict outcome; can cause unstable behavior*

Notation for Combining “Behaviors” (i.e., control modules)

- **S** denotes vector of all stimuli, s_i ,relevant for each behavior β_i detectable at time t.
- **B** denotes a vector of all active behaviors β_i at a given time t
- **G** denotes a vector encoding the relative strength or gain g_i of each active behavior β_i.
- **R** denotes a vector of all responses $\mathbf{r_i}$ generated by the set of active behaviors.
- Behavioral coordination function **C** is defined such that:

$$\rho = \boldsymbol{C}(\boldsymbol{G} * \boldsymbol{B}(\boldsymbol{S}))$$

or, alternatively:

$$\rho = \boldsymbol{C}(\boldsymbol{G} * \boldsymbol{R})$$

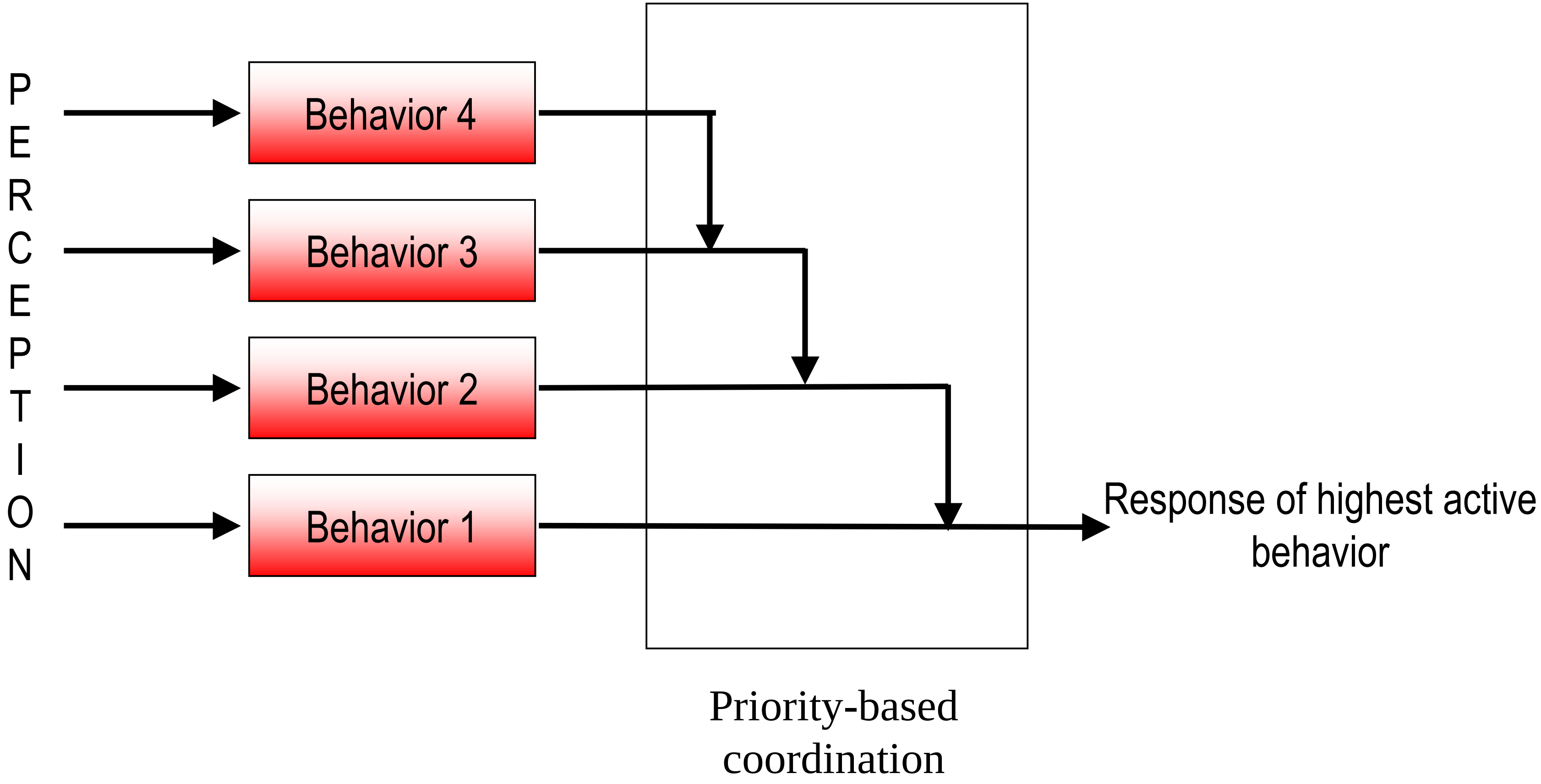
(where B(S) (i.e., behavior B with sensor input S) outputs response R)

Competitive (i.e., "Switched Parallel") Methods for Defining Combination Function, C

- Provide a means of coordinating behavioral response for conflict resolution
- Can be viewed as "winner take all"

- Arbitration can be:
  - *Fixed prioritization*
  - *Action selection*
  - *Vote generation*

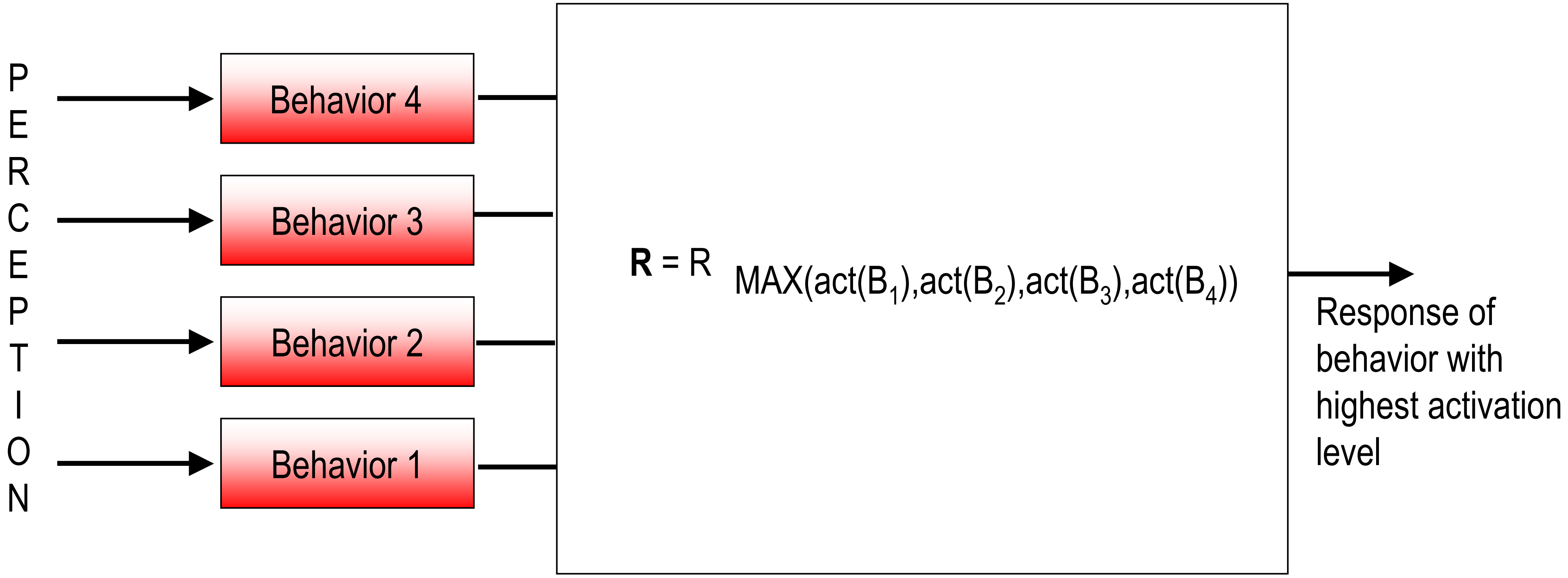
Competitive Method #1: Arbitration via Fixed Prioritization



Prioritization fixed at run-time

Adapted from © R. Siegwart, I. Nourbakhsh

Competitive Method #2: Arbitration via Action Selection



$\mathbf{R} = R_{\mathrm{MAX}(act(B_1),act(B_2),act(B_3),act(B_4))}$

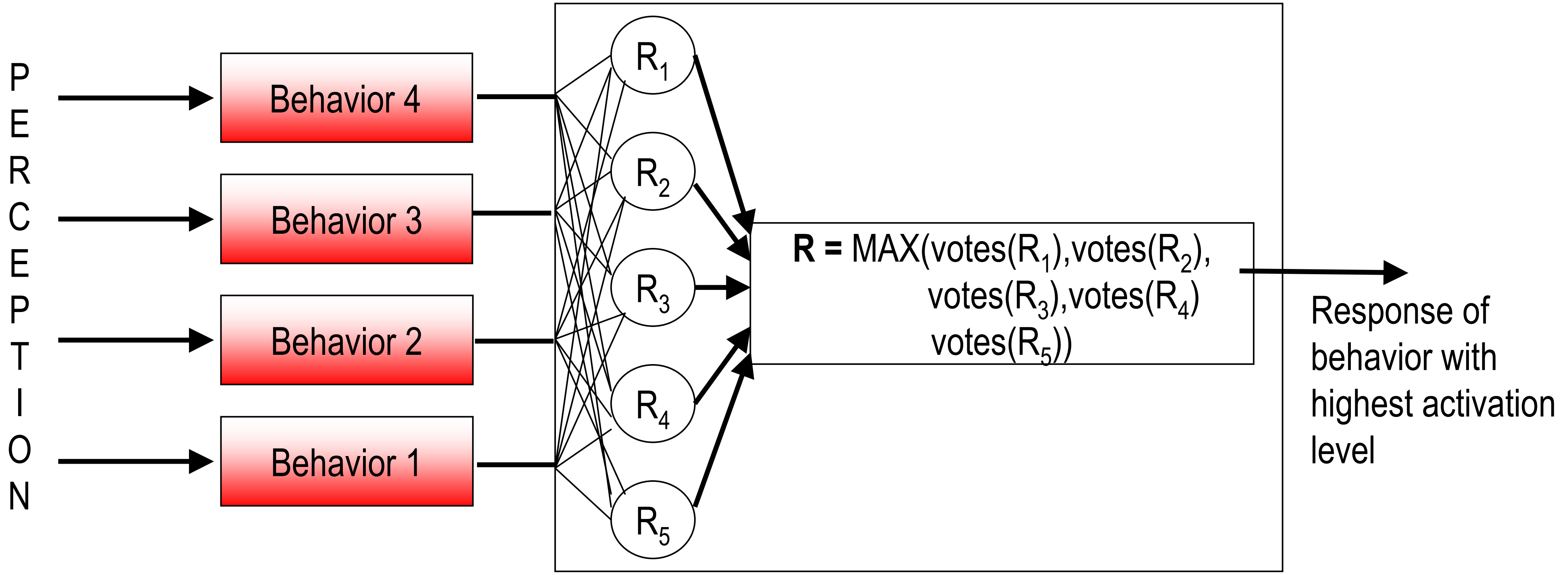
Action-Selection coordination

- Behaviors compete through use of activation levels driven by agent's goals

Prioritization varies during mission

Adapted from © R. Siegwart, I. Nourbakhsh

Competitive Method #3: Arbitration via Voting



PERCEPTION

Behavior 4

Behavior 3

Behavior 2

Behavior 1

R_1

R_2

R_3

R_4

R_5

$\mathbf{R} = \mathrm{MAX}(\mathrm{votes}(R_1), \mathrm{votes}(R_2), \mathrm{votes}(R_3), \mathrm{votes}(R_4)\ \mathrm{votes}(R_5))$

Response of behavior with highest activation level

Voting-based coordination

- Pre-defined set of motor responses;
- Each behavior allocates votes (in some distribution) to each motor response
- Motor response with most votes is executed

Prioritization varies during mission

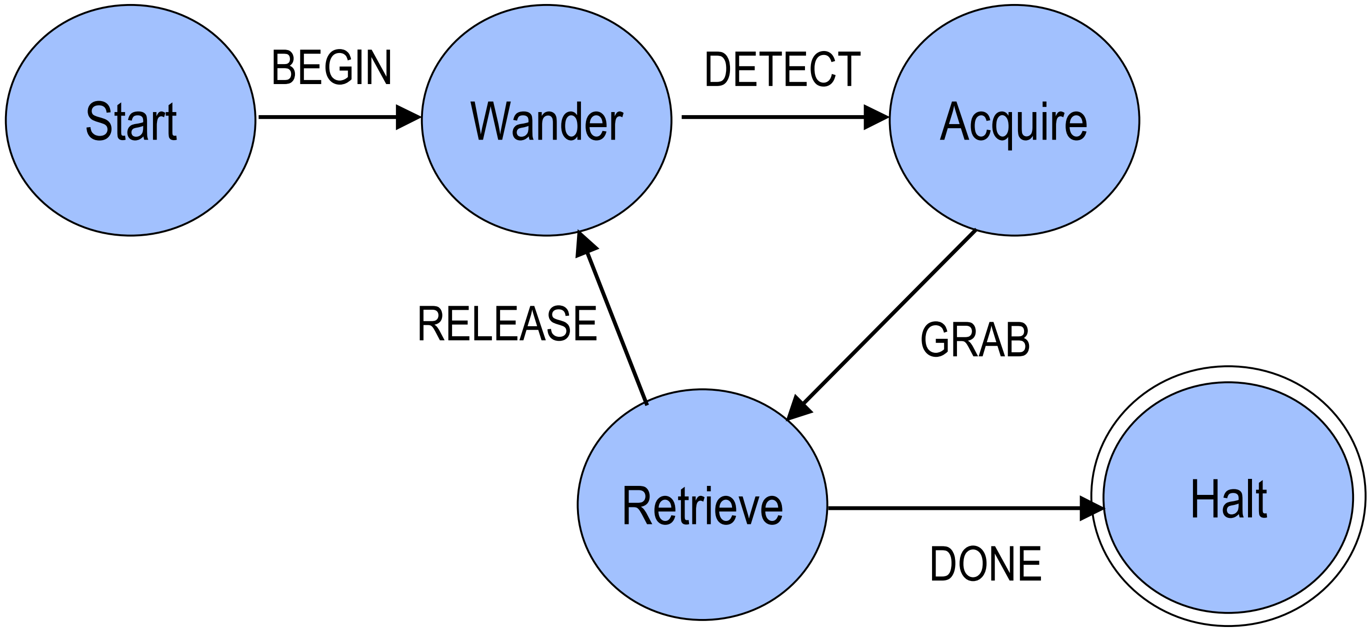
Adapted from © R. Siegwart, I. Nourbakhsh

Competitive Method #4: Sequencing based on Finite State Automata

- As an example to consider, let’s look at robotic foraging
- Foraging:
  - *Robot moves away from home base looking for attractor objects*
  - *When detect attractor object, move toward it, pick it up, and return it to home base*
  - *Repeat until all attractors in environment have been returned home*
- High-level behaviors required to accomplish foraging:
  - Wander: *move through world in search of an attractor*
  - Acquire: *move toward attractor when detected*
  - Retrieve: *return the attractor to the home base once required*

Competitive Method #4: Sequencing based on FSA (con't.)

- Can sequence behaviors if one activity needs to be completed before another.
- Example: Foraging – Finite State Automata (FSA) diagram:



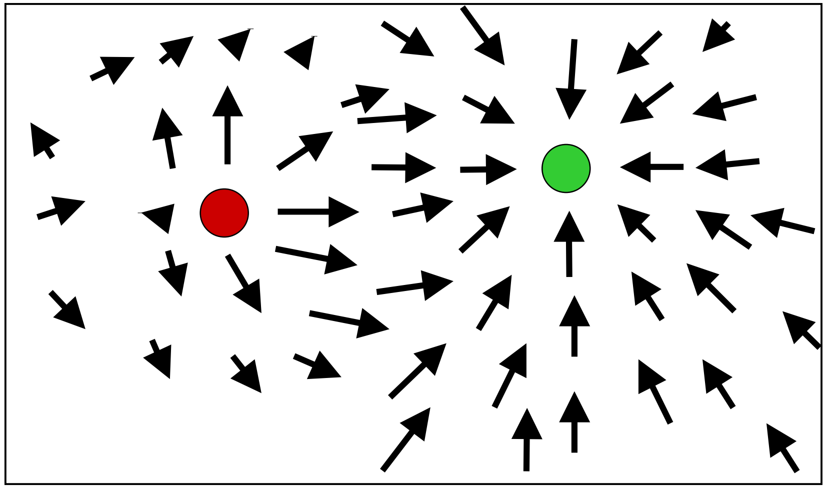
Adapted from © R. Siegwart, I. Nourbakhsh

Cooperative (i.e., "Mixed Parallel") Methods for Defining Combination Function, C

- Behavioral fusion provides ability to concurrently use the output of more than one behavior at a time

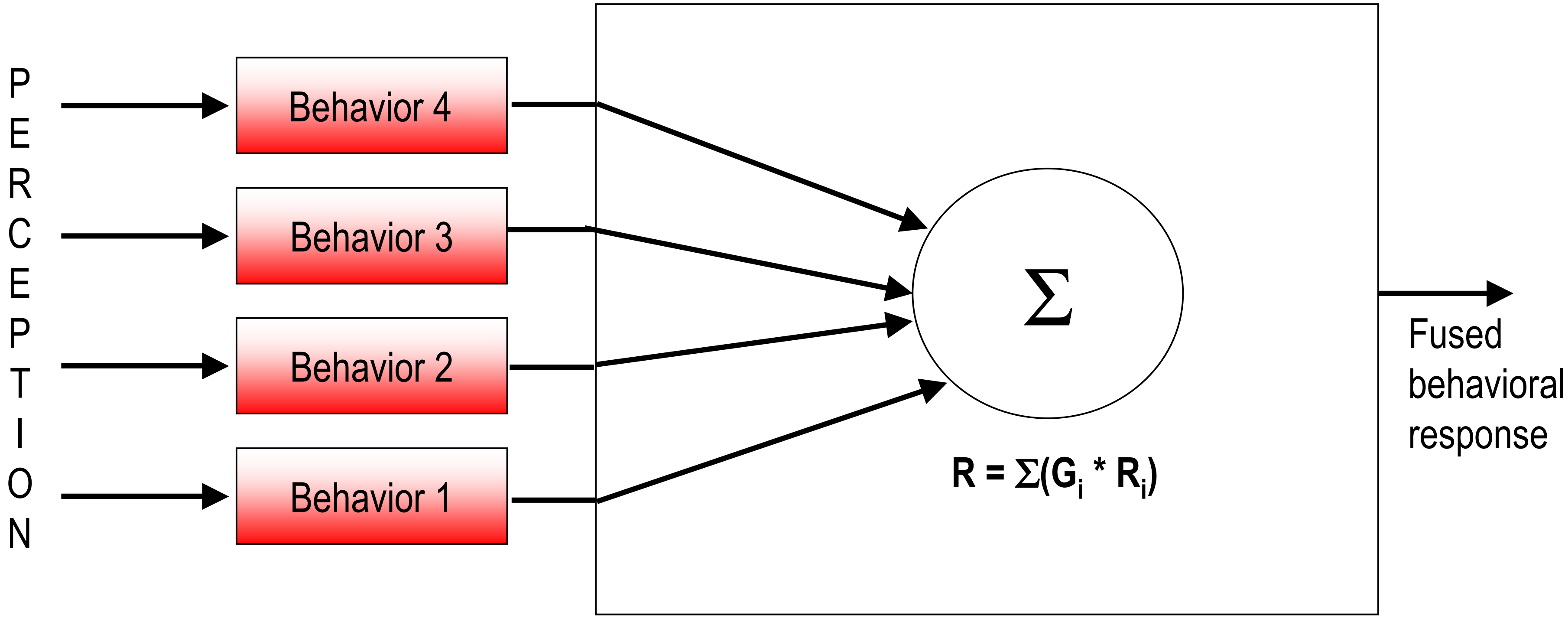
- Central issue: finding representation amenable to fusion

- Common method:
  - *Vector addition using potential fields*
  - *Recall potential field approach:*



Adapted from © R. Siegwart, I. Nourbakhsh

Cooperative Method: Behavioral Fusion via Vector Summation



PERCEPTION

Behavior 4

Behavior 3

Behavior 2

Behavior 1

Σ

$R = \Sigma(G_i * R_i)$

Fused behavioral response

Behavioral fusion

Summarizing Behavior Coordination

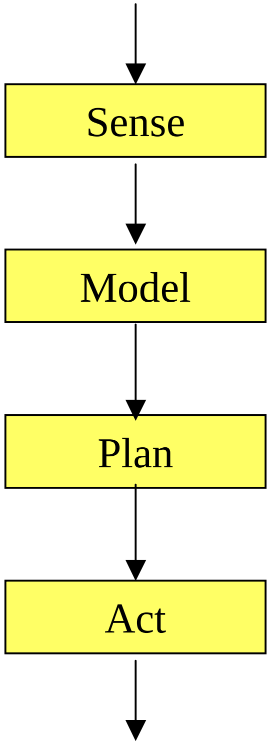
- Two main strategies:
  - *Competitive*
    - *Fixed prioritization*
    - *Action selection*
    - *Voting*
    - *Sequencing*
    - *Etc.*

  - *Cooperative*
    - *Vector addition*
    - *Etc.*

- Can also have combination of these two

Case Study: Subsumption Architecture
Example of *Competitive (Switched Parallel)* Combination

- Developed in mid-1980s by Rodney Brooks, MIT



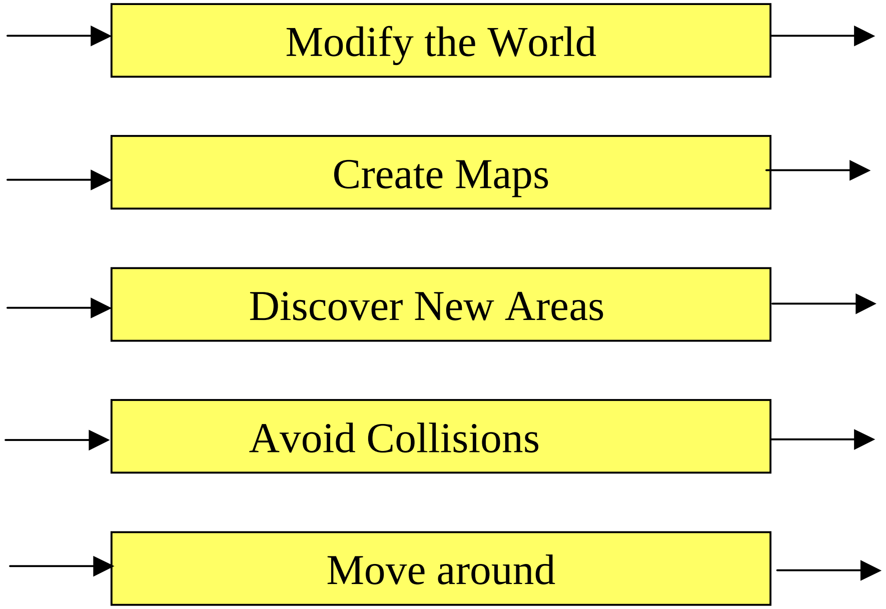
Sense

Model

Plan

Act

Old Sense-plan-act model



Modify the World

Create Maps

Discover New Areas

Avoid Collisions

Move around

New subsumption model

Tenets of the Subsumption Architecture

- Complex behavior need not be the product of a complex control system
- Intelligence is in the eye of the observer
- The world is its own best model
- Simplicity is a virtue
- Robots should be cheap
- Robustness in the presence of noisy or failing sensors is a design goal
- Planning is just a way of avoiding figuring out what to do next
- All onboard computation is important
- Systems should be built incrementally
- No representation. No calibration. No complex computers. No high-bandwidth communication.

Categorization of Subsumption Architecture

Name	Subsumption architecture
Background	Well-known early reactive architecture
Precursors	Braitenberg (1984), Walter (1953)
Principal design method	Experimental
Developer	Rodney Brooks (MIT)
Response encoding	Predominantly discrete (rule-based)
Coordination method	Competitive (priority-based arbitration via inhibition and suppression)
Programming method	Old method uses AFMs; newer method uses Behavior Language
Robots fielded	Allen, Genghis, Squirt, Toto, Seymour, Polly, etc.
References	Brooks 1986; Brooks 1990

Subsumption Robots

- Allen
- Tom and Jerry
- Genghis and Attila
- Squirt
- Toto
- Seymour
- Tito
- Polly
- Cog

Adapted from © R. Siegwart, I. Nourbakhsh

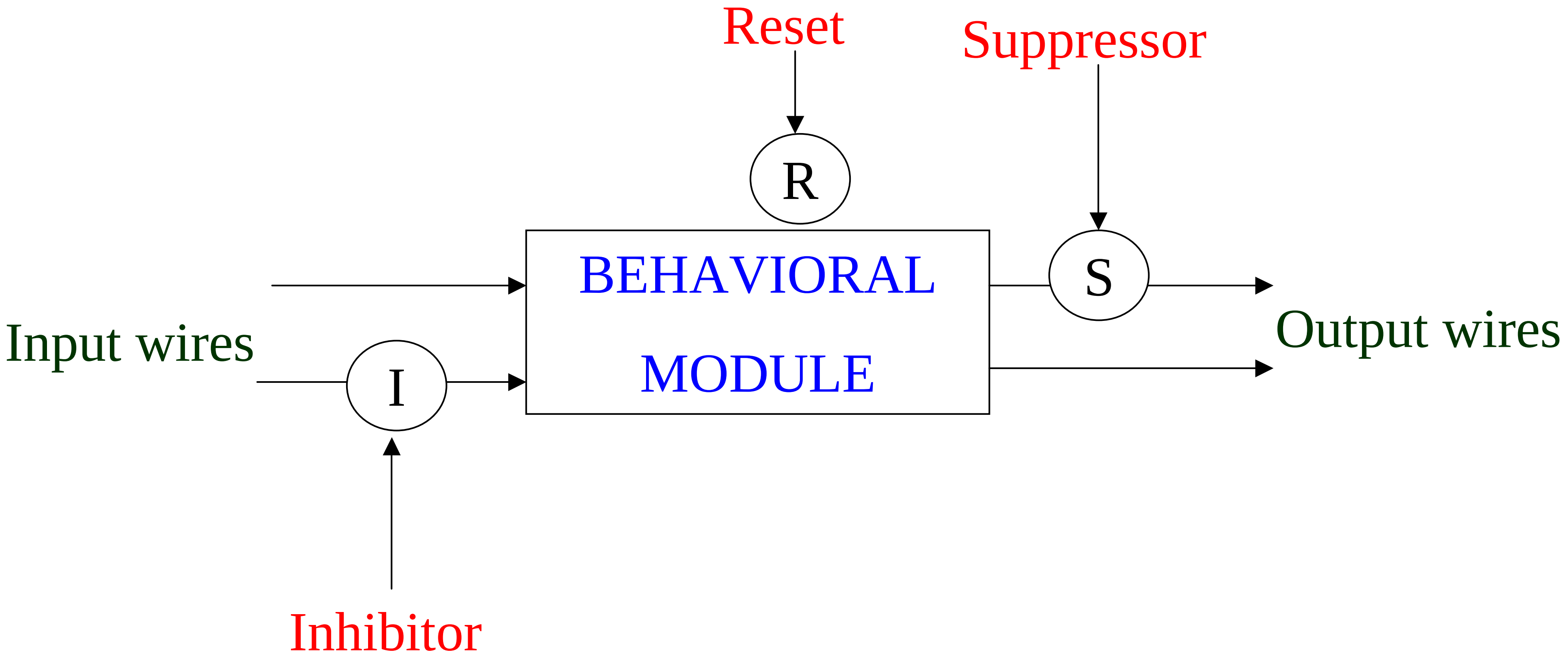
Coordination in Subsumption

- "Subsumption" comes from coordination process used between layered behaviors of architecture
- Complex actions subsume simpler behaviors
- Fixed priority hierarchy defines topology
- Lower levels of architecture have no "awareness" of upper levels

- Coordination has two mechanisms:
  - *Inhibition: used to prevent a signal being transmitted along an AFSM wire from reaching the actuators*
  - *Suppression: prevents the current signal from being transmitted and replaces that signal with the suppressing message*

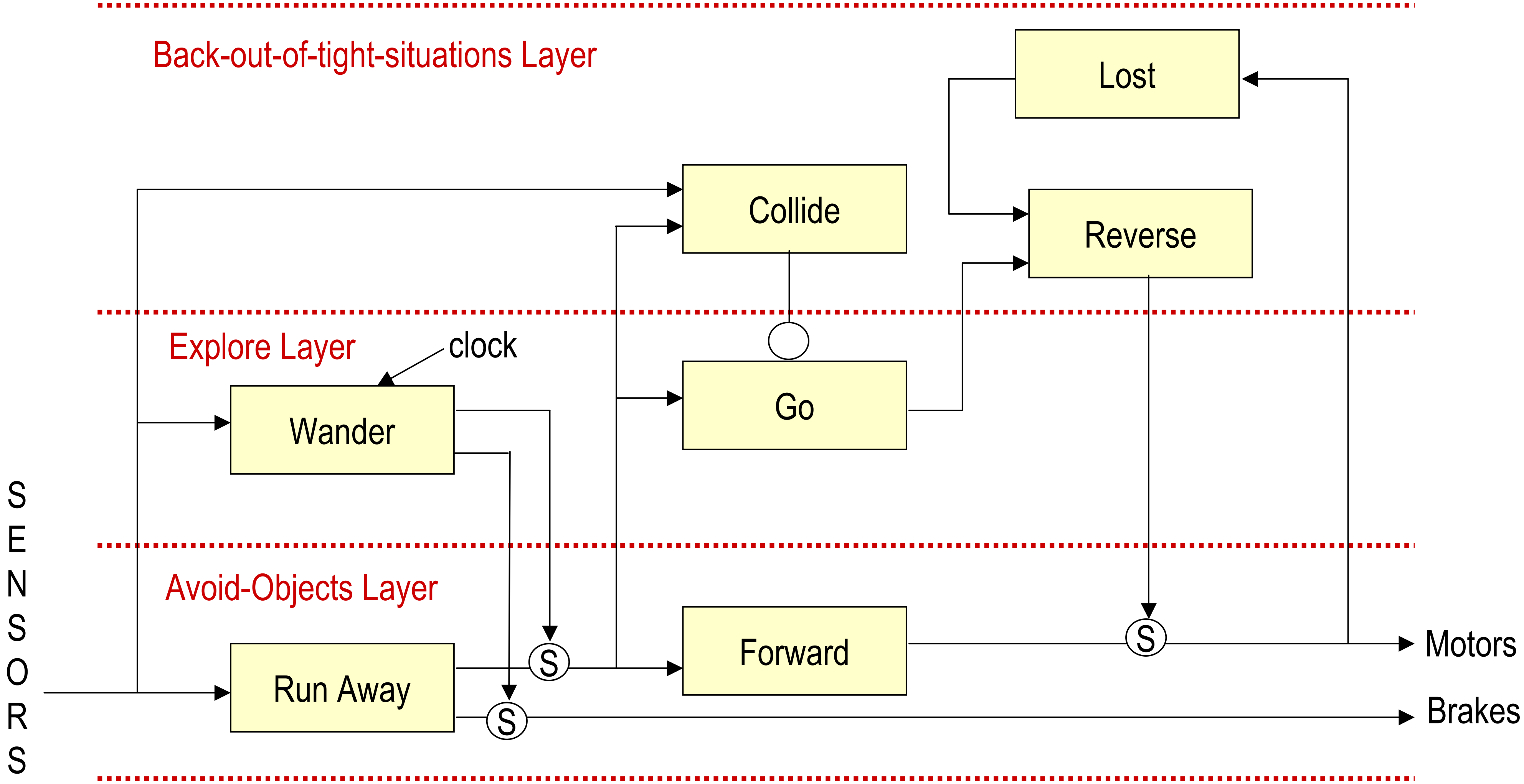
Adapted from © R. Siegwart, I. Nourbakhsh

Subsumption Based on Augmented Finite State Machines (AFSM)



Adapted from © R. Siegwart, I. Nourbakhsh

Example of 3-Layered Subsumption Implementation



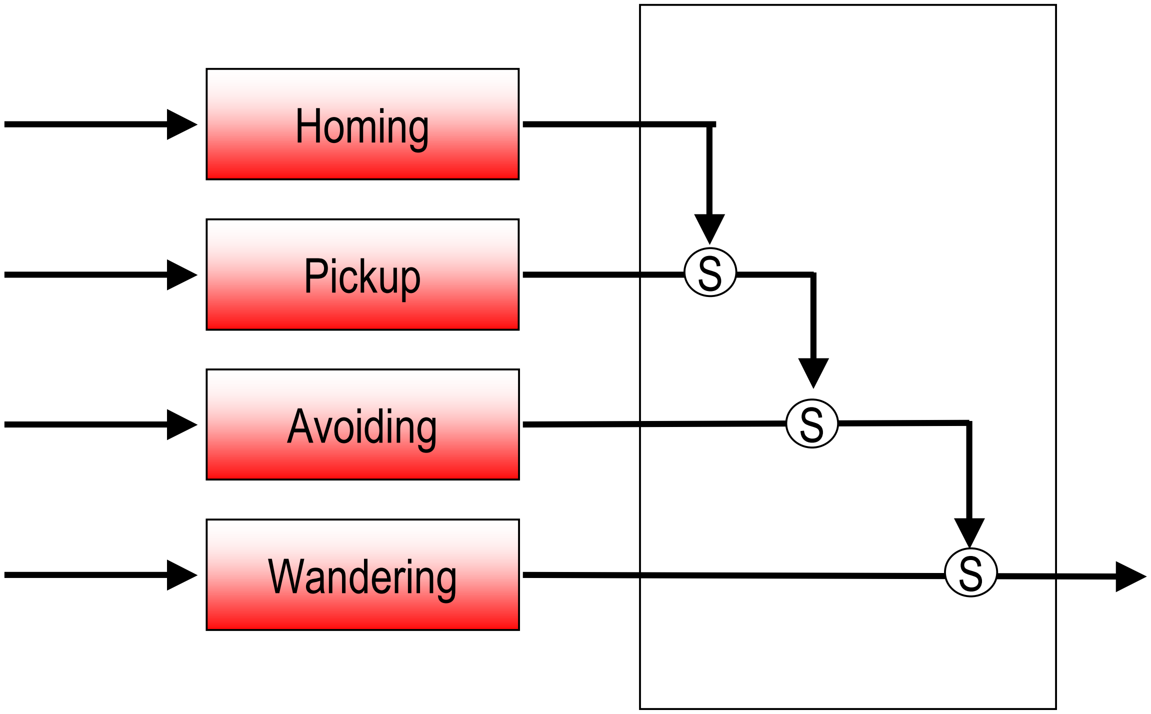
Adapted from © R. Siegwart, I. Nourbakhsh

Foraging Example

- Behaviors Used:
  - *Wandering: move in a random direction for some time*
  - *Avoiding:*
    - *Turn to the right if the obstacle is on the left, then go*
    - *Turn to the left if the obstacle is on the right, then go*
    - *After three attempts, back up and turn*
    - *If an obstacle is present on both sides, randomly turn and back up*
  - *Pickup: Turn toward the sensed attractor and go forward. If at the attractor, close gripper.*
  - *Homing: Turn toward the home base and go forward, otherwise if at home, stop.*

Adapted from © R. Siegwart, I. Nourbakhsh

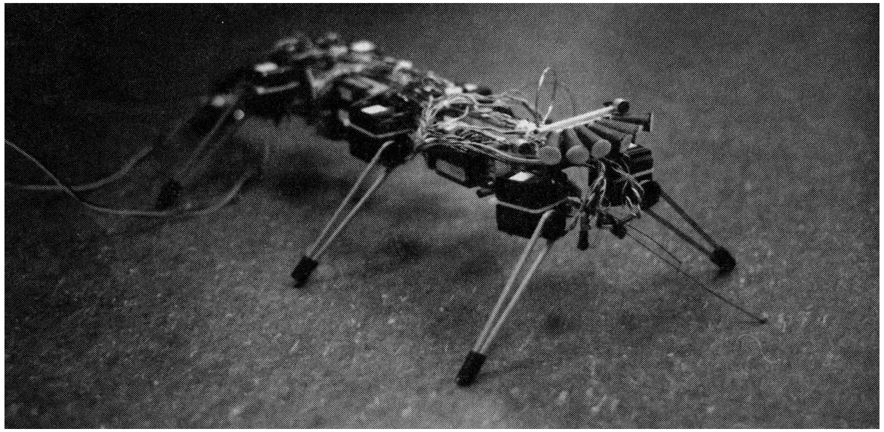
Organization for Subsumption-Based Foraging Robot



Adapted from © R. Siegwart, I. Nourbakhsh

Genghis Subsumption Design

- Behavioral layers implemented:
  - *Standup*
  - *Simple walk*
  - *Force balancing*
  - *Leg lifting*
  - *Whiskers*
  - *Pitch stabilization*
  - *Prowling*
  - *Steered prowling*

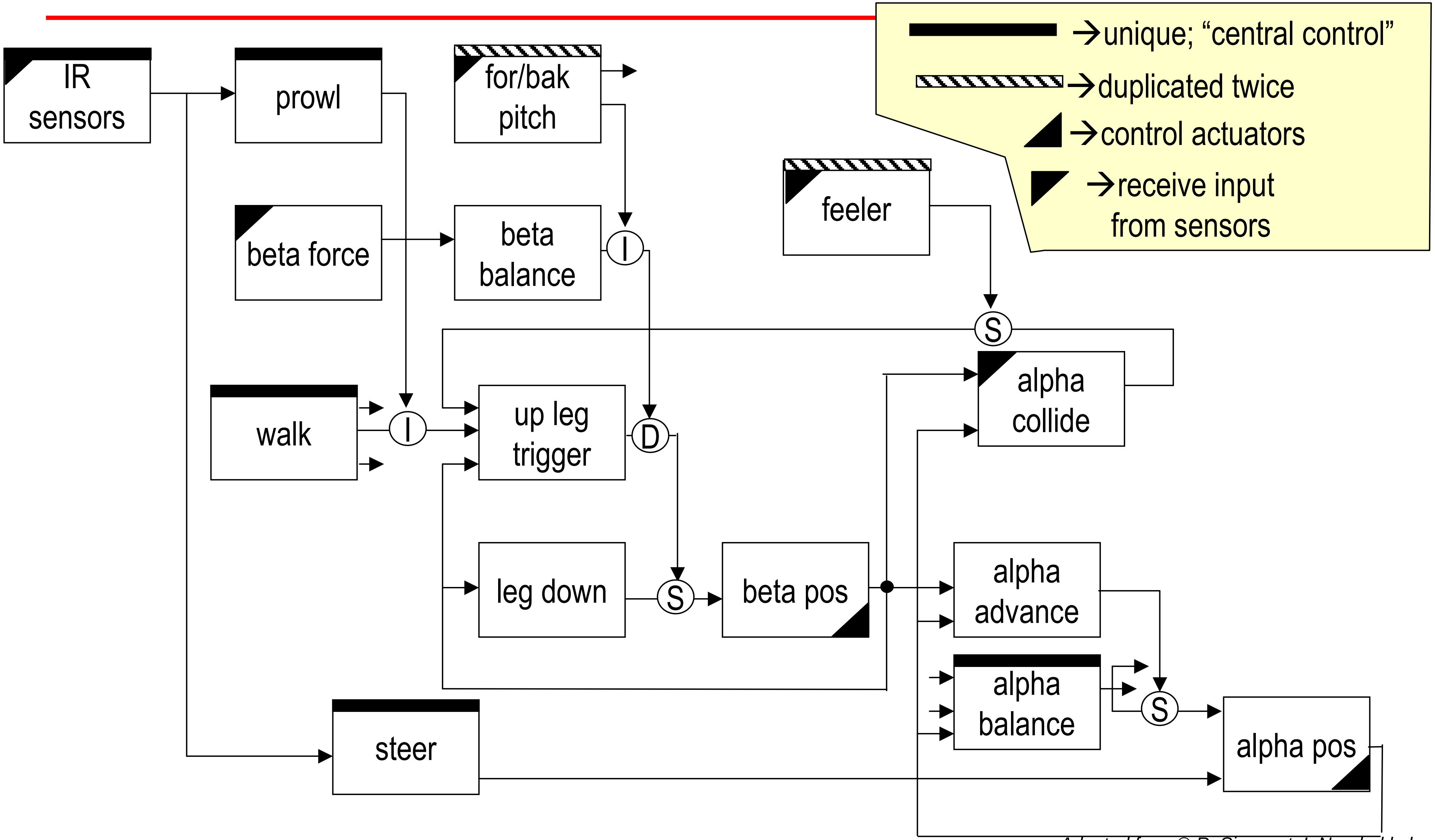


Two motors per leg:

α = *advance*, which swings leg back and forth
β = *balance*, which lifts leg up and down

Adapted from © R. Siegwart, I. Nourbakhsh

Genghis AFSM Network



▬ →unique; “central control”

▨ →duplicated twice

◢ →control actuators

◤ →receive input from sensors

Adapted from © R. Siegwart, I. Nourbakhsh

“Core Subset” of Genghis AFSM Network

Enables robot to walk without any feedback:

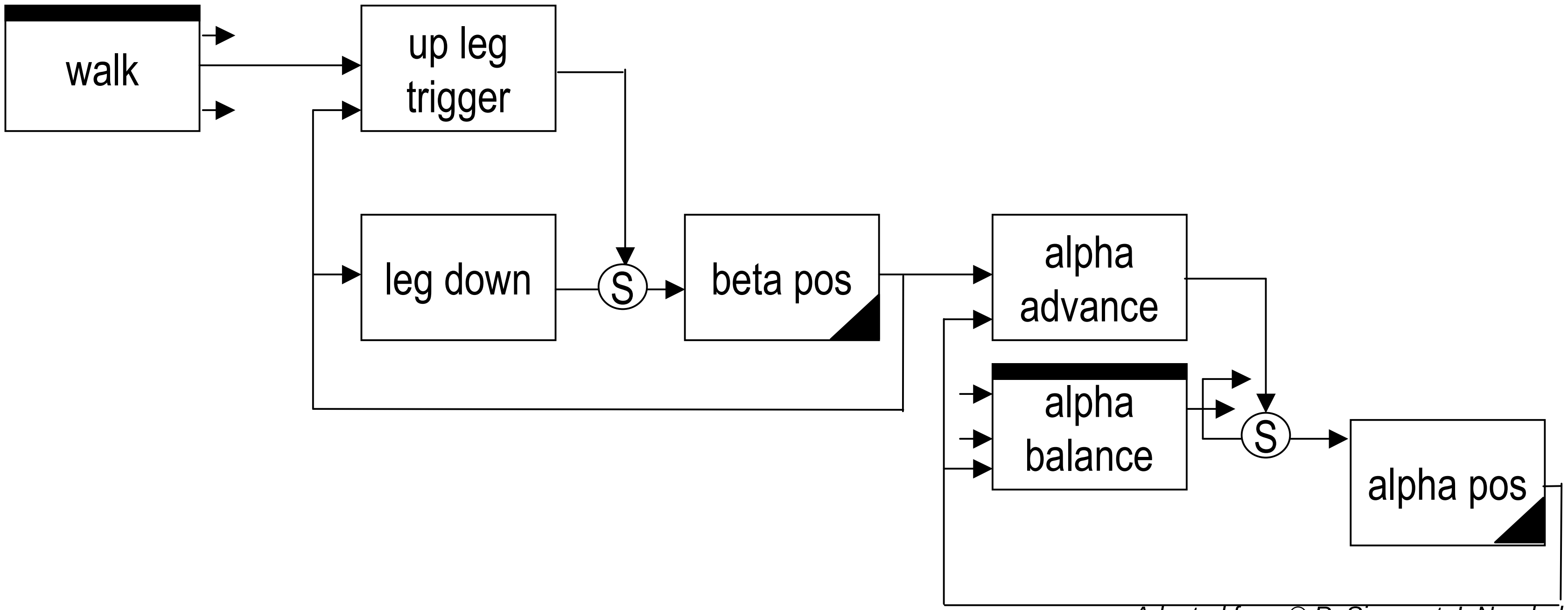
- Standup
- Simple walk

▬ →unique; “central control”

▨ →duplicated twice

◢ →control actuators

◤ →receive input from sensors



Adapted from © R. Siegwart, I. Nourbakhsh

Evaluation of Subsumption

- Strengths:
  - *Hardware retargetability: Subsumption can compile down directly onto programmable-array logic circuitry*
  - *Support for parallelism: Each behavioral layer can run independently and asynchronously*
  - *Niche targetability: Custom behaviors can be created for specific task-environment pairs*

- Null (not strength/not weakness):
  - *Robustness: Can be successfully engineered into system but is often hard-wired and hard to implement*
  - *Timeliness for development: Some support tools exist, but significant learning curve exists*

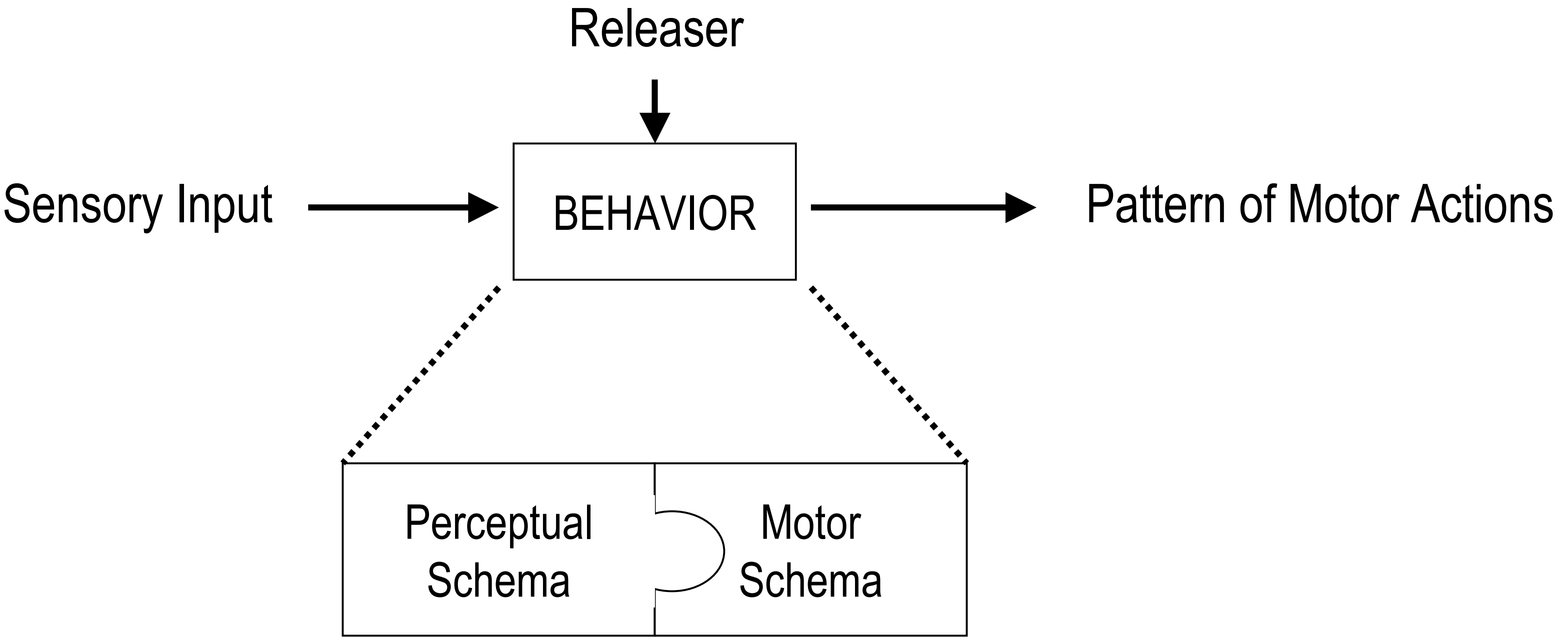
- Weaknesses:
  - *Run time flexibility: priority-based coordination mechanism, ad hoc aspect of behavior generation, and hard-wired aspects limit adaptation of system*
  - *Support for modularity: behavioral reuse is not widely done in practice*

Example of Cooperative (i.e., mixed parallel) Combination Motor Schemas (with Potential Fields)

- Motor Schemas -- Based upon schema theory:
  - *Explains motor behavior in terms of concurrent control of many different activities*
  - *Schema stores both how to react and the way that reaction can be realized*
  - *A distributed model of computation*
  - *Provides a language for connecting action and perception*
  - *Activation levels are associated with schemas that determine their readiness or applicability for acting*
  - *Provides a theory of learning through acquisition and tuning*

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


Releaser

Sensory Input

BEHAVIOR

Pattern of Motor Actions

Perceptual Schema

Motor Schema

Adapted from © R. Siegwart, I. Nourbakhsh

Categorization of Motor Schemas

Name	Motor Schemas
Background	Reactive component of AuRA Architecture
Precursors	Arbib (1981); Khatib (1985)
Principal design method	Ethologically guided
Developer	Ronald Arkin (GaTech)
Response encoding	Continuous using potential field analog
Coordination method	Cooperative via vector summation and normalization
Programming method	Parameterized behavioral libraries
Robots fielded	HARV, George, Ren and Stimpy, Buzz, blizzards, mobile manipulator, etc.
References	Arkin (1987), Arkin (1989), Arkin (1992)

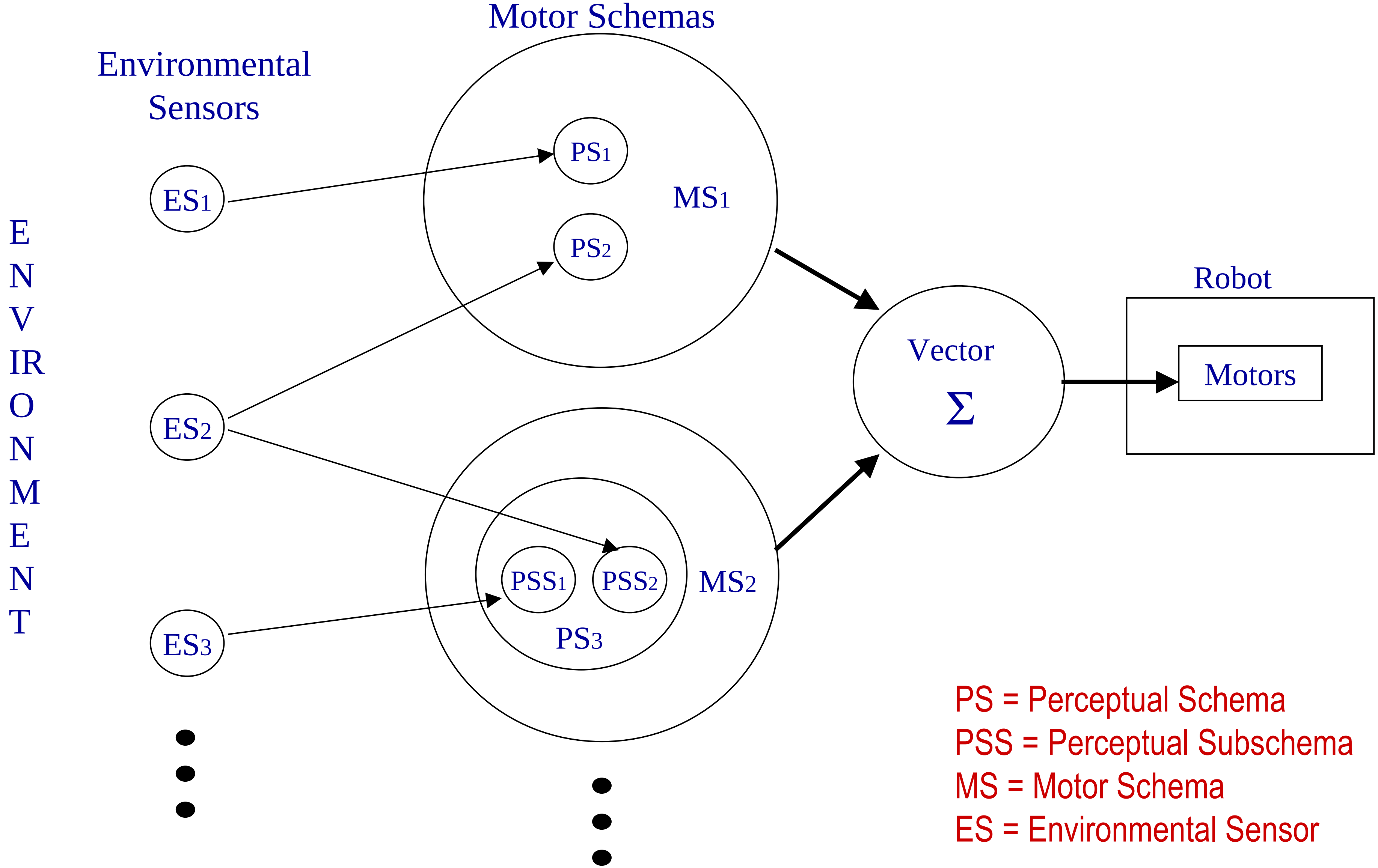
Adapted from © R. Siegwart, I. Nourbakhsh

Differences of Motor Schemas versus Other Behavioral Approaches

- Behavioral responses are all represented as vectors generated using a potential fields approach
- Coordination is achieved by vector addition
- No predefined hierarchy exists for coordination; instead, behaviors are configured at run-time
- Pure arbitration is not used; each behavior can contribute in varying degrees to robot's overall response

Adapted from © R. Siegwart, I. Nourbakhsh

Perception-Action Schema Relationships



Motor Schemas

Environmental Sensors

E N V IR O N M E N T

ES_1

ES_2

ES_3

PS_1

PS_2

MS_1

PSS_1

PSS_2

PS_3

MS_2

Vector

Σ

Robot

Motors

PS = Perceptual Schema

PSS = Perceptual Subschema

MS = Motor Schema

ES = Environmental Sensor

Adapted from © R. Siegwart, I. Nourbakhsh

Defined Motor Schemas

- Move-ahead
- Move-to-goal
- Avoid-static-obstacle
- Dodge
- Escape
- Stay-on-path
- Noise
- Follow-the-leader
- Probe
- Dock
- Avoid-past
- Move up, move-down, maintain altitude
- Teleautonomy

Each of these is defined as a potential field of output vector responses.

Schema-Based Robots (Mostly at Georgia Tech)

- HARV
- George
- Ren and Stimpy
- Buzz
- Io, Callisto, Ganymede
- Mobile manipulator

Io, Callisto, Ganymede

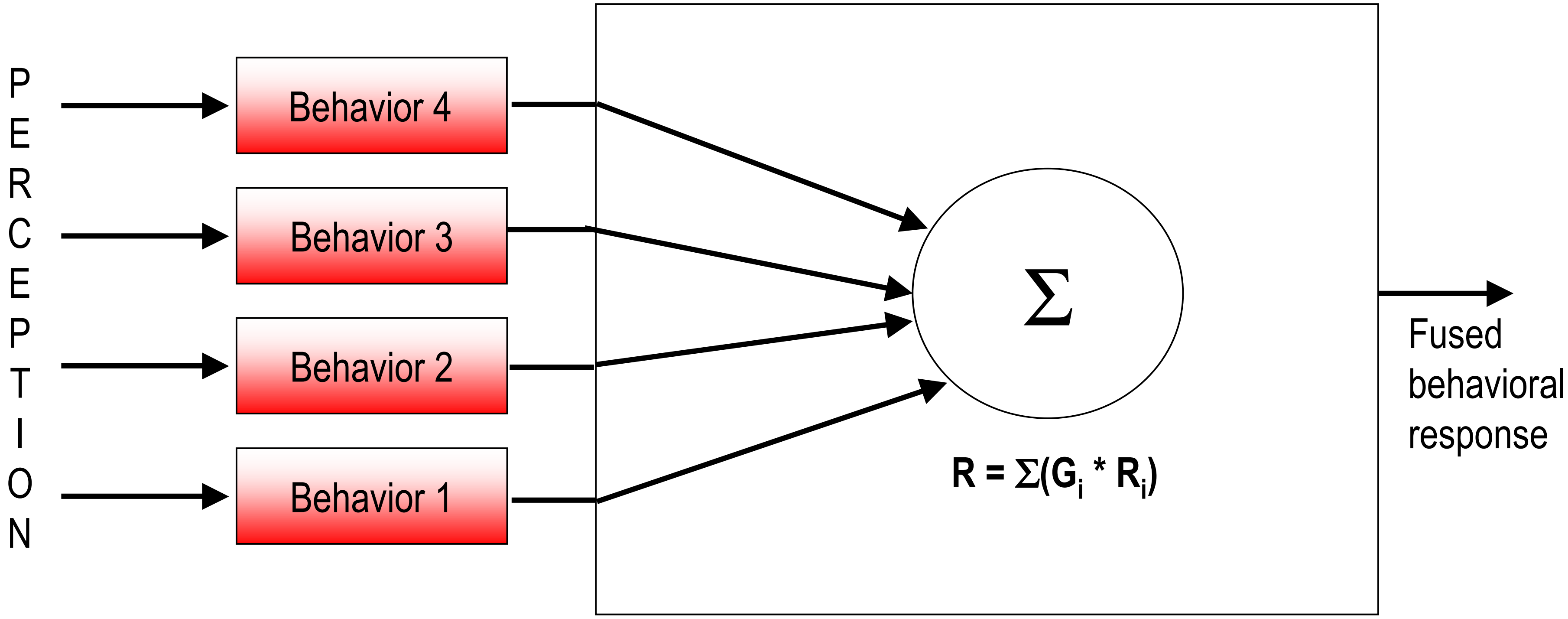
Output of Motor Schemas Defined as Vectors

- Output Vector: consists of both orientation and magnitude components

- $V_{magnitude}$ denotes magnitude of resultant response vector
- $V_{direction}$ denotes orientation

Adapted from © R. Siegwart, I. Nourbakhsh

Motors Schemas Achieve Behavioral Fusion via Vector Summation



PERCEPTION

Behavior 4

Behavior 3

Behavior 2

Behavior 1

Σ

$R = \Sigma(G_i * R_i)$

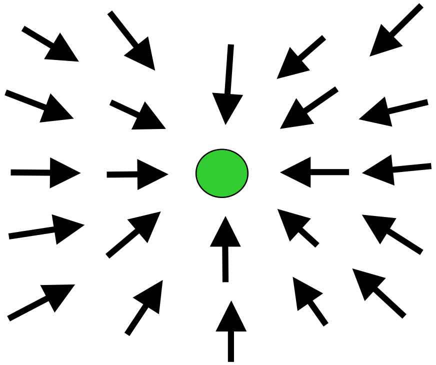
Fused behavioral response

Behavioral fusion

Example Motor Schema Encodings

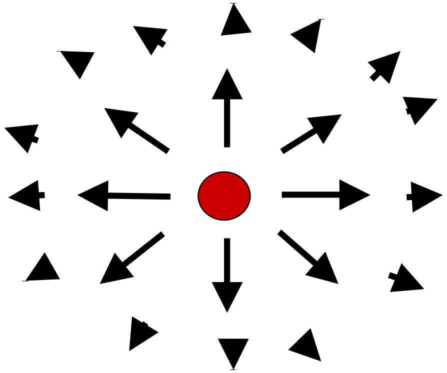
- Move-to-goal (ballistic):

  $V_{\text{magnitude}}$ = *fixed gain value*
  $V_{\text{direction}}$ = *towards perceived goal*



- Avoid-static-obstacle:

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



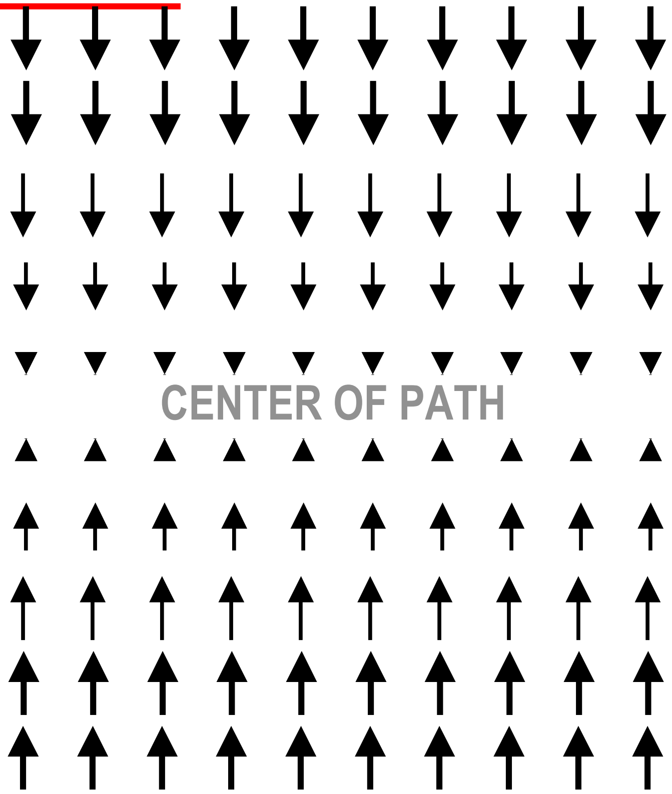
where S = *sphere of influence of obstacle*
R = *radius of obstacle*
G = *gain*
d = *distance of robot to center of obstacle*

Adapted from © R. Siegwart, I. Nourbakhsh

More Motor Schema Encodings

- Stay-on-path:

$$V_{magnitude} = \begin{cases} P & \text{for } d > (W/2) \\ \dfrac{d}{W/2} * G & \text{for d} \leq (\text{W}/2) \end{cases}$$



where:

W = *width of path*

P = *off-path gain*

G = *on-path gain*

D = *distance of robot to center of path*

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

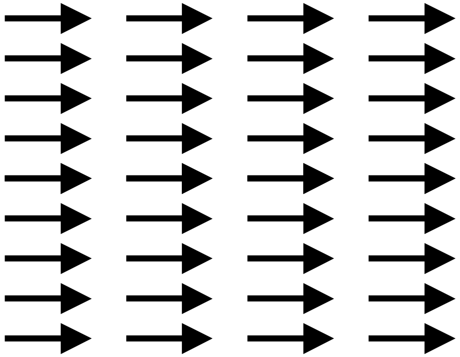
Adapted from © R. Siegwart, I. Nourbakhsh

More Motor Schema Encodings (con’t.)

- Move-ahead:

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

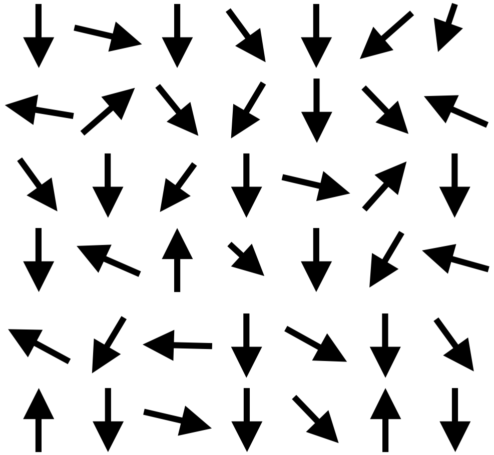
$V_{\text{direction}}$ = *specified compass direction*



- Noise:

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

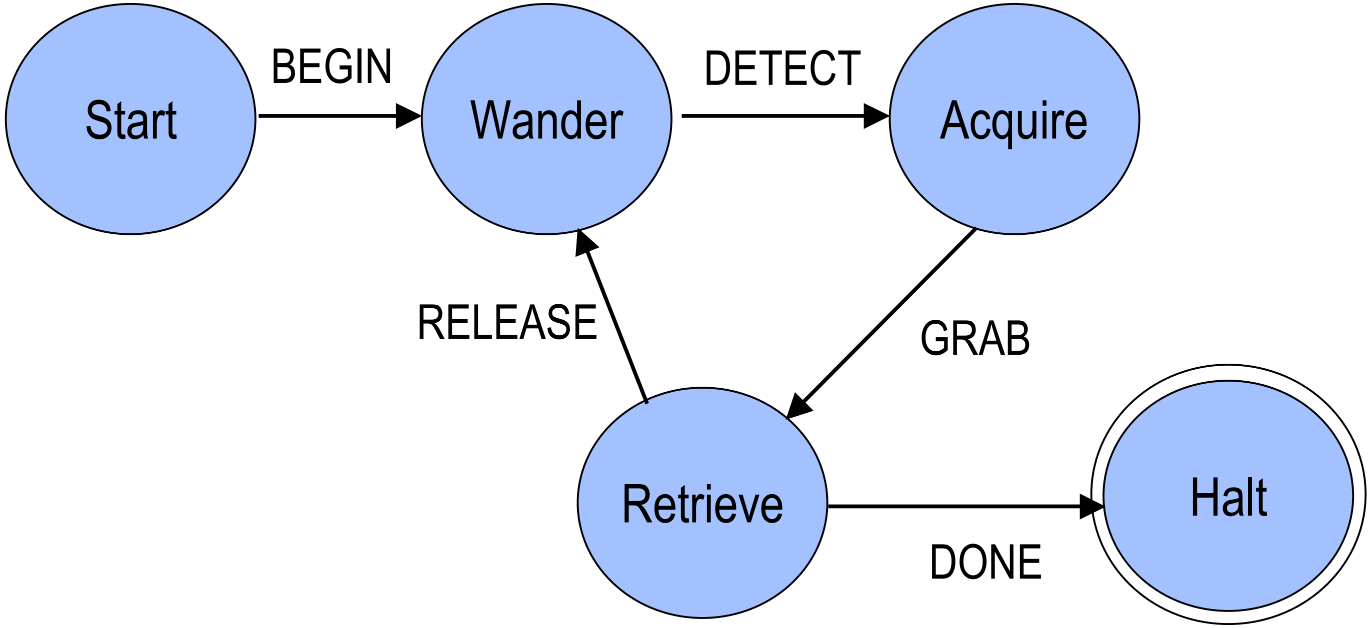
$V_{\text{direction}}$ = *random direction changed every*
p *time steps*



Adapted from © R. Siegwart, I. Nourbakhsh

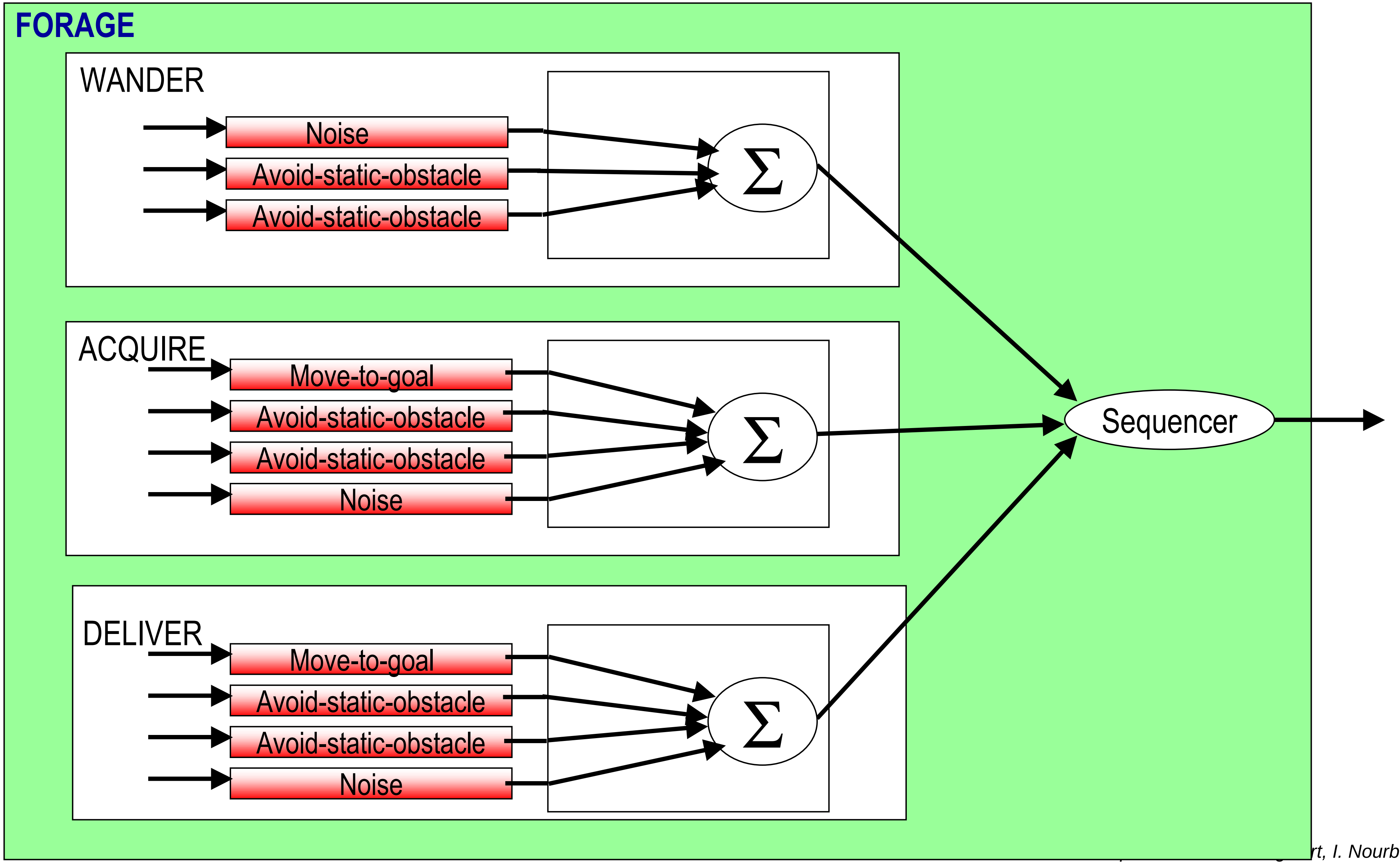
Sequencing of Motor Schemas

- Can sequence motor schemas if one activity needs to be completed before another.
- Recall Foraging – FSA diagram:



Adapted from © R. Siegwart, I. Nourbakhsh

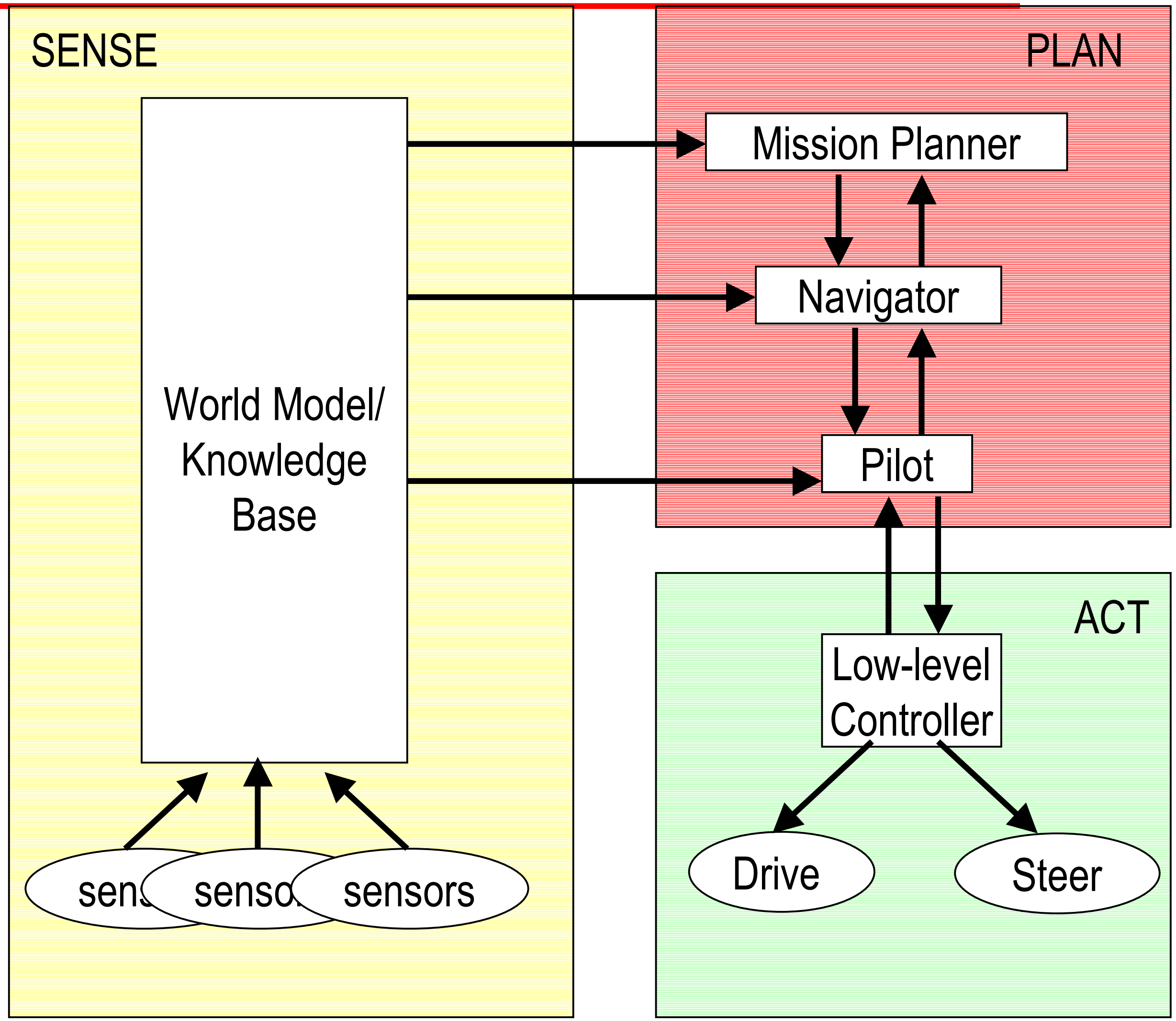
Stimulus-Response Diagram for Schema-Based Foraging



More generally – What do overall robot control architectures look like?

- One example: Nested Hierarchical Controller

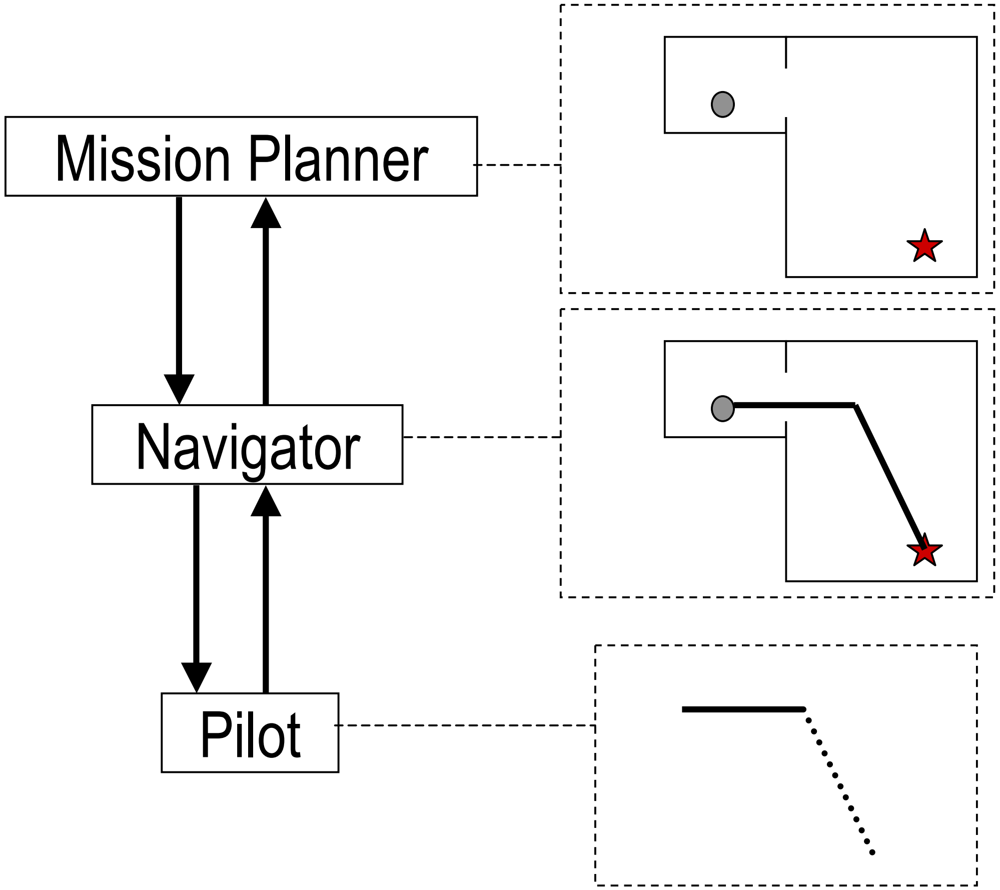
Nested Hierarchical Controller



Major contribution of NHC: Decomposition of planning into three subsystems

Adapted from © R. Siegwart, I. Nourbakhsh

Planning is Hierarchical



Uses map to locate self and goal

Generates path from current position to goal

Generates actions robot must execute to follow path segment

Adapted from © R. Siegwart, I. Nourbakhsh

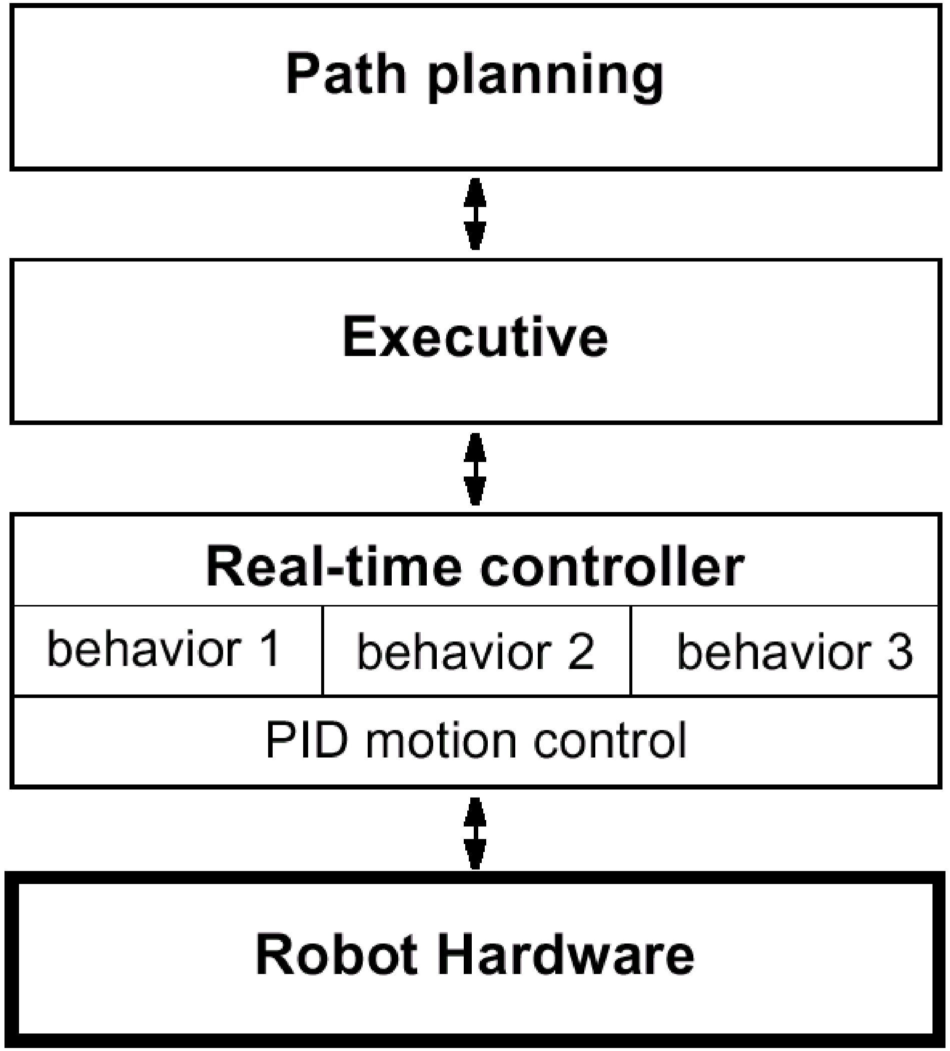
Advantage/Disadvantage of NHC

- Advantage:
  - *Interleaves planning and acting*
    - *Plan is changed if world is different from expected*

- Disadvantage:
  - *Planning decomposition is only appropriate for navigation tasks*

General Tiered Architecture

- Executive Layer
  - *activation of behaviors*
  - *failure recognition*
  - *re-initiating the planner*



Path planning

Executive

Real-time controller

behavior 1

behavior 2

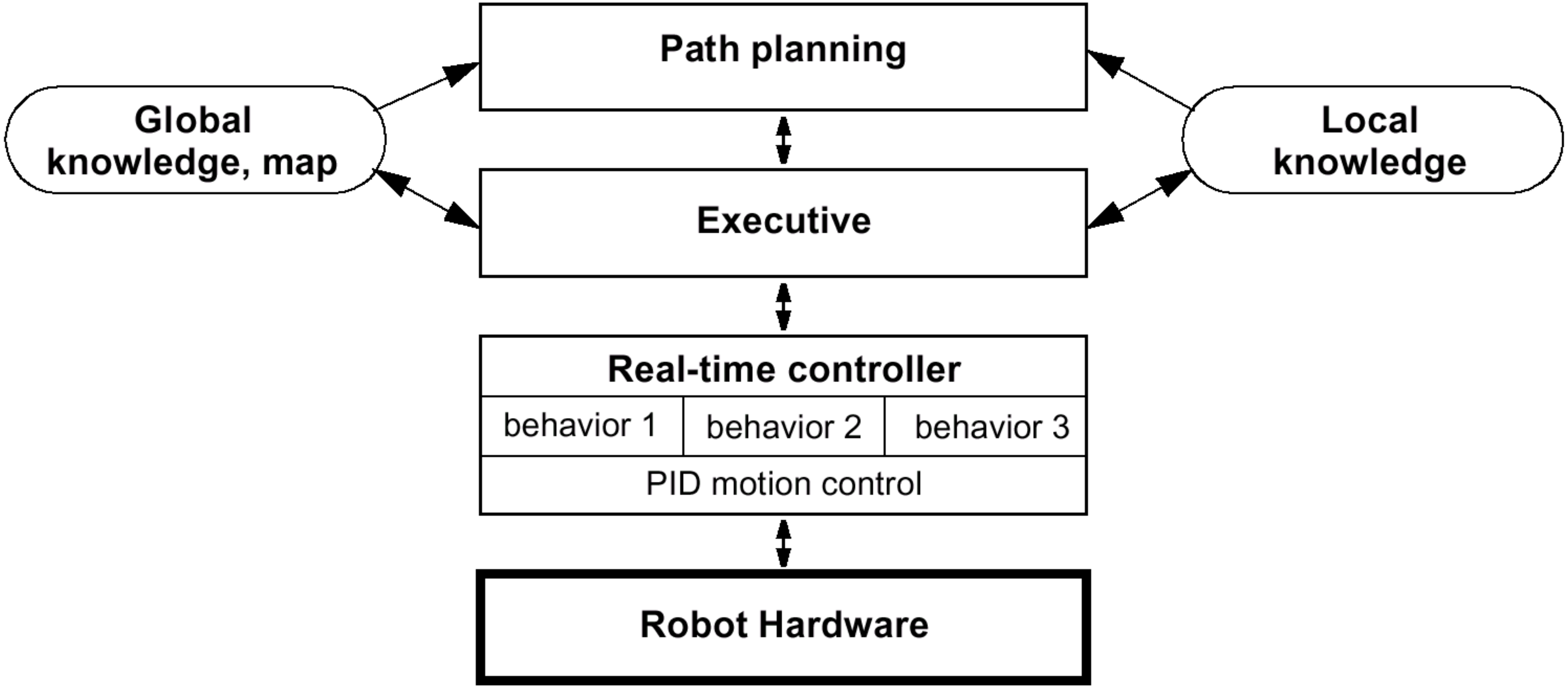
behavior 3

PID motion control

Robot Hardware

Adapted from © R. Siegwart, I. Nourbakhsh

A Three-Tiered Episodic Planning Architecture.



Path planning

Global knowledge, map

Local knowledge

Executive

Real-time controller

behavior 1

behavior 2

behavior 3

PID motion control

Robot Hardware

- Planner is triggered when needed: e.g. blockage, failure

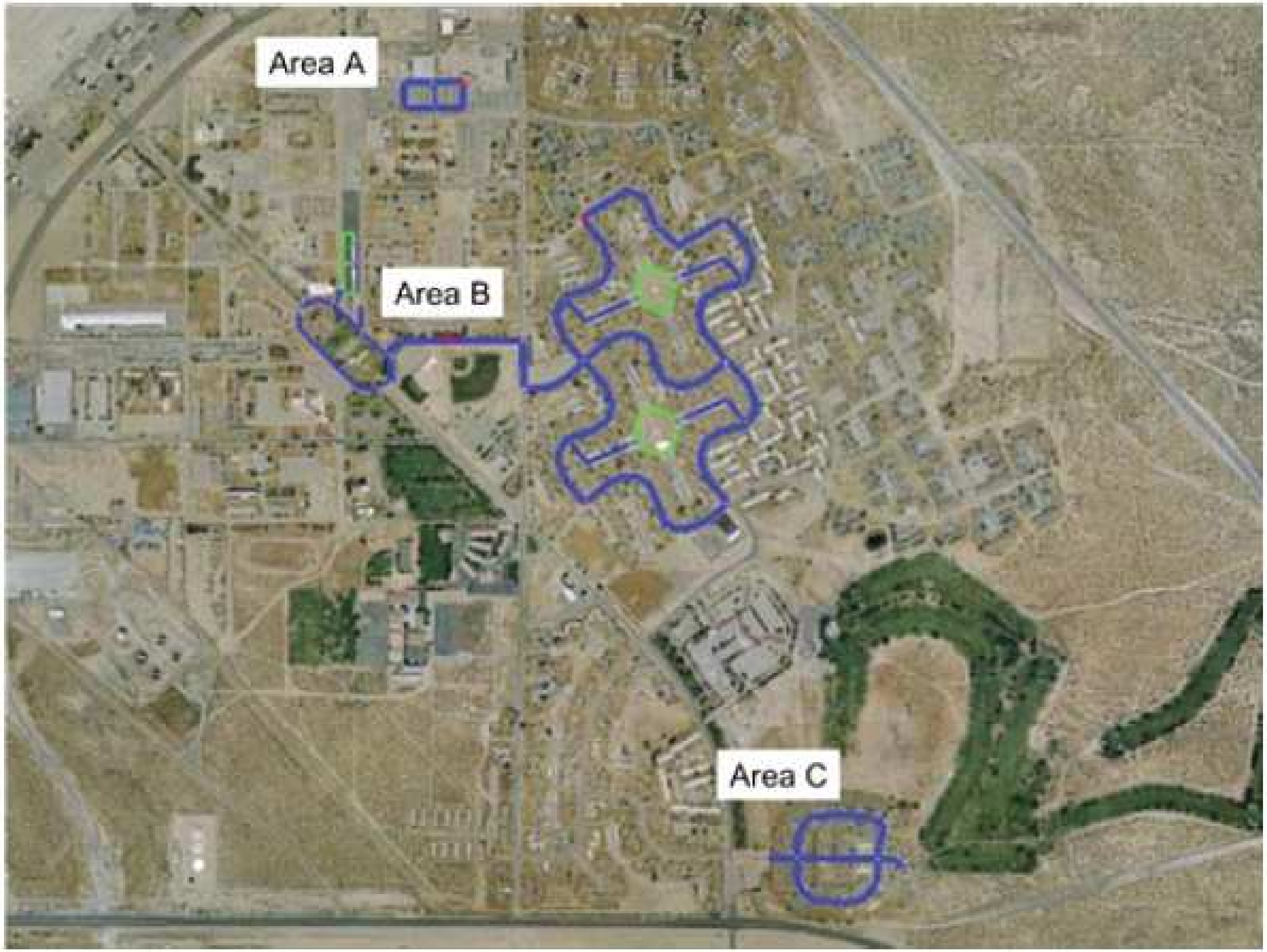
Adapted from © R. Siegwart, I. Nourbakhsh

Recent practical examples from DARPA Urban Challenge

- Objective:
  Autonomous vehicle drives 97km through an urban environment, interacting with other moving vehicles and obeying the California Driver Handbook.

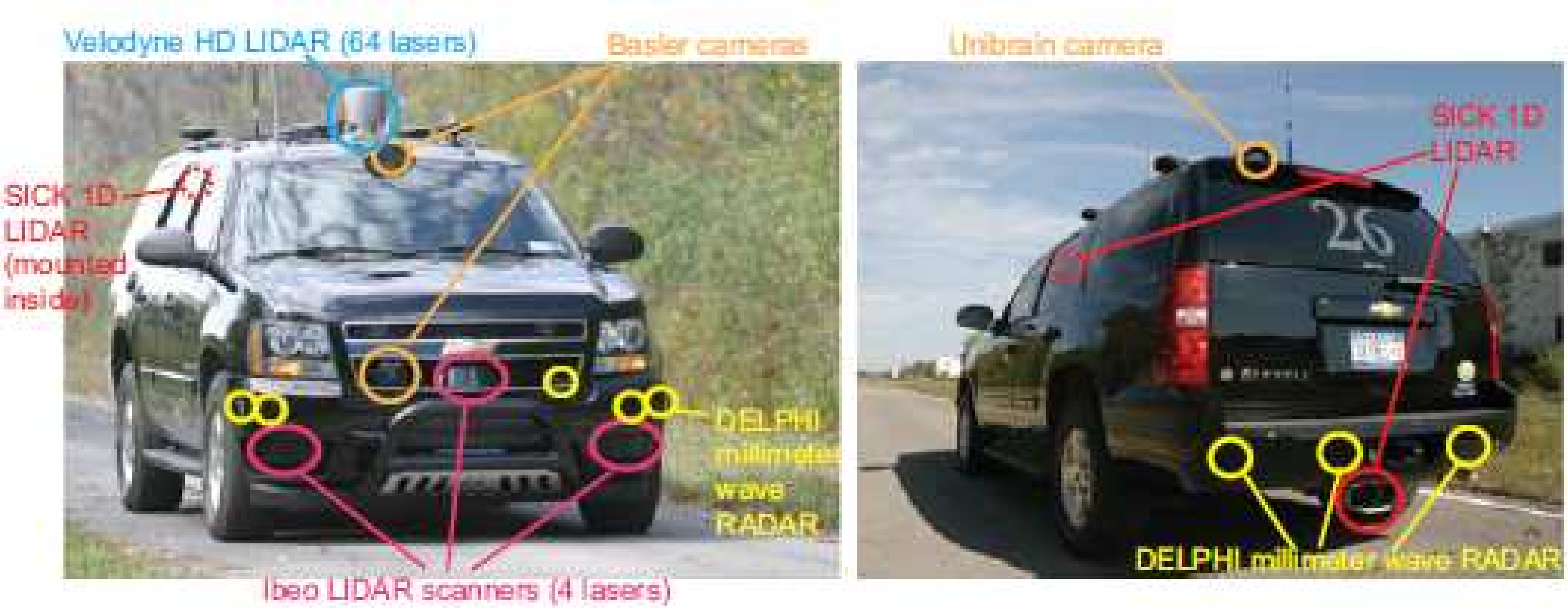
Qualification Event --

- Area A: tested merging with moving traffic
- Area B: tested navigation
- Area C: tested rerouting and intersection skills



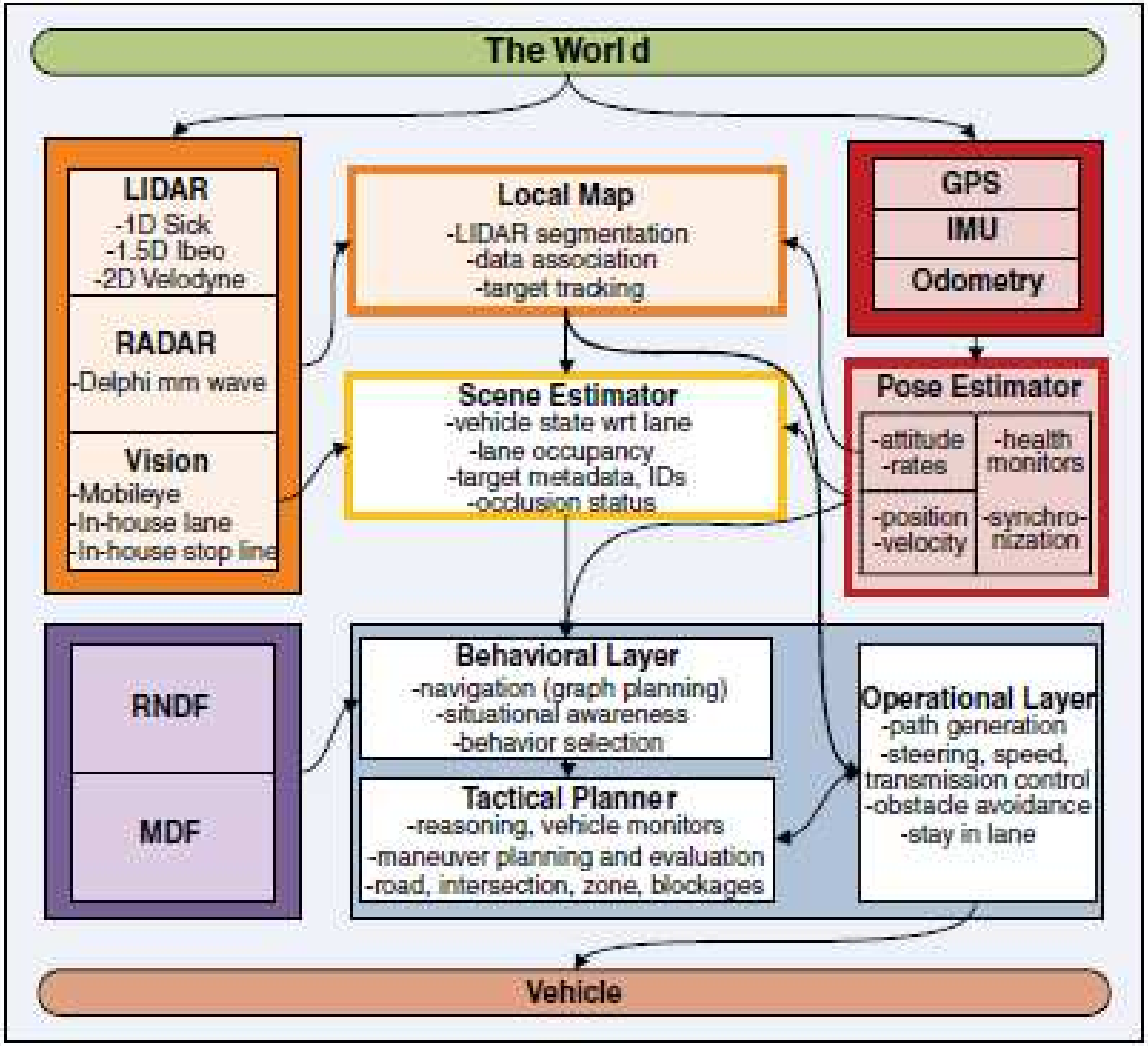
Example: DARPA Urban Challenge Vehicle Architecture

“Team Cornell’s Skynet”



Adapted from © R. Siegwart, I. Nourbakhsh

Skynet Architecture



The World

LIDAR
-1D Sick
-1.5D Ibeo
-2D Velodyne

RADAR
-Delphi mm wave

Vision
-Mobileye
-In-house lane
-In-house stop line

Local Map
-LIDAR segmentation
-data association
-target tracking

GPS

IMU

Odometry

Scene Estimator
-vehicle state wrt lane
-lane occupancy
-target metadata, IDs
-occlusion status

Pose Estimator
-attitude
-rates
-health monitors
-position
-velocity
-synchro-nization

RNDF

MDF

Behavioral Layer
-navigation (graph planning)
-situational awareness
-behavior selection

Tactical Planner
-reasoning, vehicle monitors
-maneuver planning and evaluation
-road, intersection, zone, blockages

Operational Layer
-path generation
-steering, speed, transmission control
-obstacle avoidance
-stay in lane

Vehicle

RNDF: Route network definition file
MDF: Mission data file

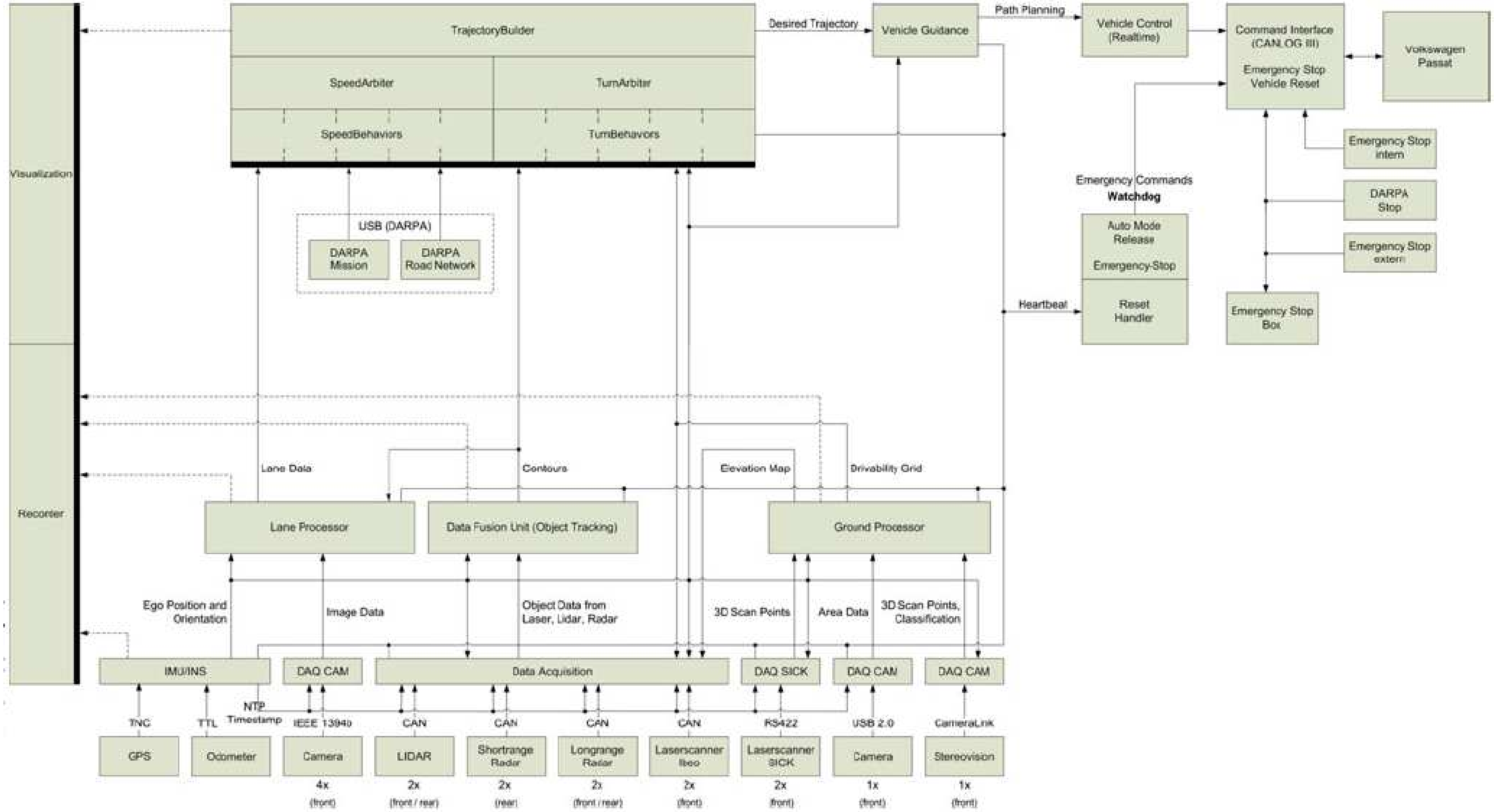
Adapted from © R. Siegwart, I. Nourbakhsh

Another Urban Challenge Example

"Caroline" (Germany)

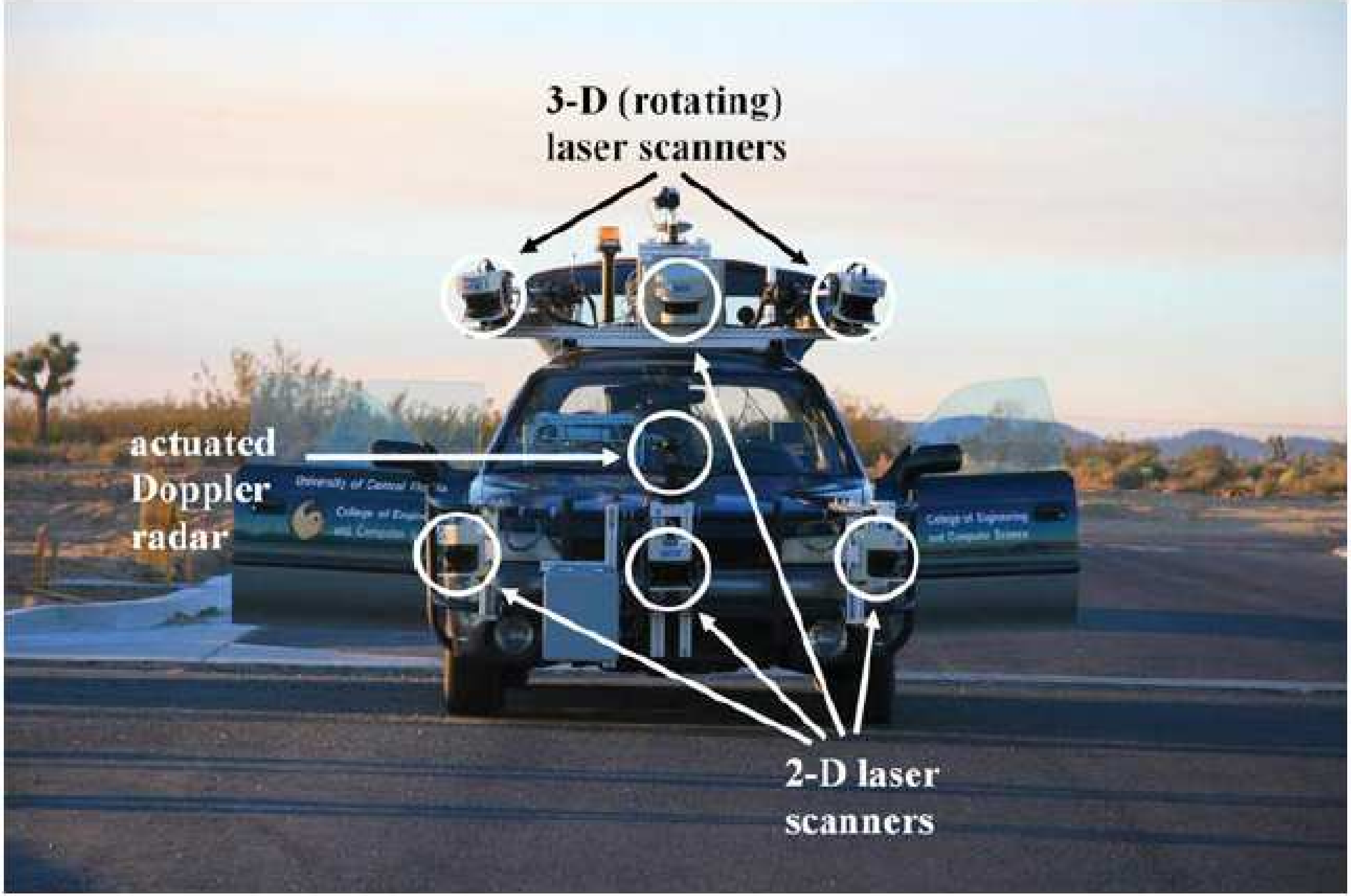
Adapted from © R. Siegwart, I. Nourbakhsh

Example: “Caroline” Architecture



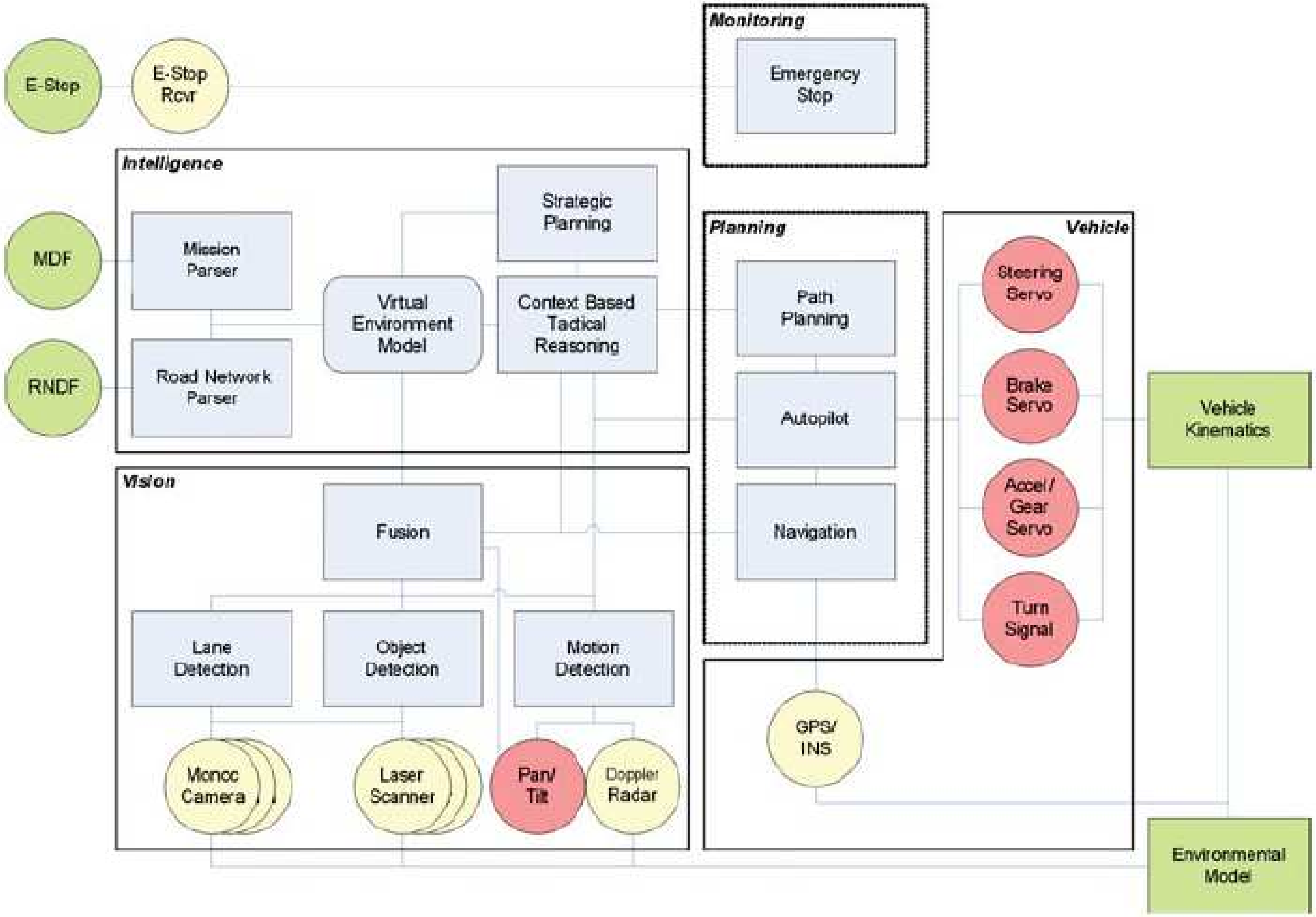
Yet Another Urban Challenge Example

"Knight Rider" (Coleman, Old Dominion, U. Central Fla.)



Adapted from © R. Siegwart, I. Nourbakhsh

Example “Knight Rider” Architecture



Adapted from © R. Siegwart, I. Nourbakhsh

Bottom Line: Lots of Alternative Architecture Designs

- No “one size fits all” approach
- Many approaches will work
- Design particular architecture to meet needs of given application