Robotic Control Paradigms

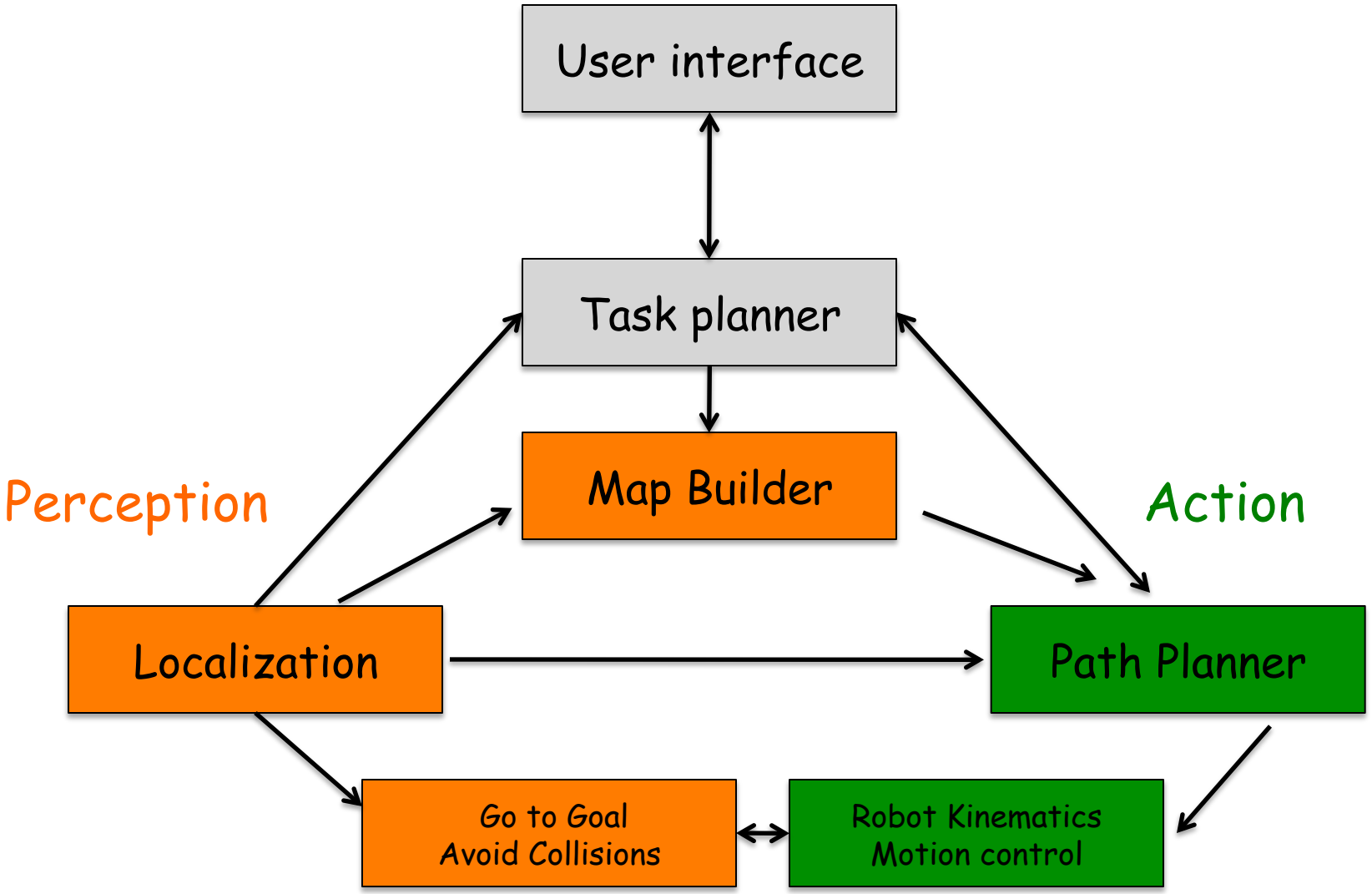
Previously basics of control
- trajectory generation
- closed feedback-loop control

Particular control law yields robotic behavior
- going to the goal from any position – goal reaching behavior
- avoiding obstacles – obstacle avoidance behaviour
- motivated by potential field based approach – steering behaviors
- elementary behaviors often don't need an explicit model of the environment

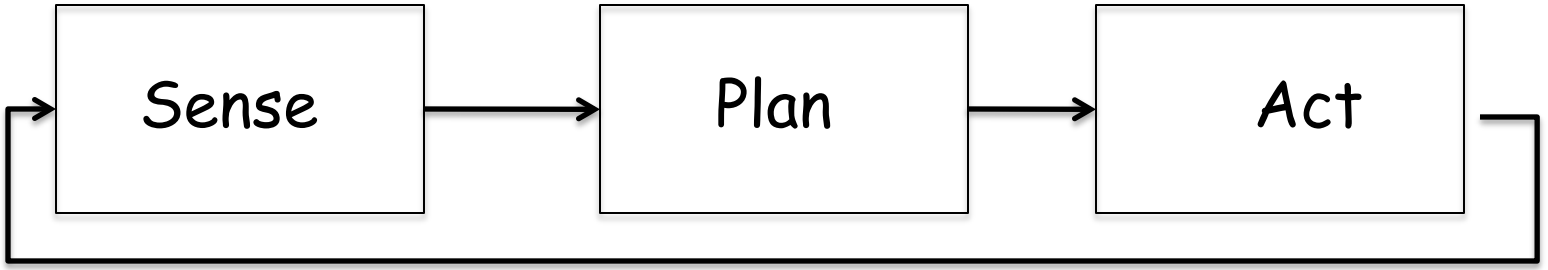
Motion planning (later)
- Representation of the environment
- Different choices
-Path planning algorithms

Different organization of these components yields different control Architectures

Typical architecture

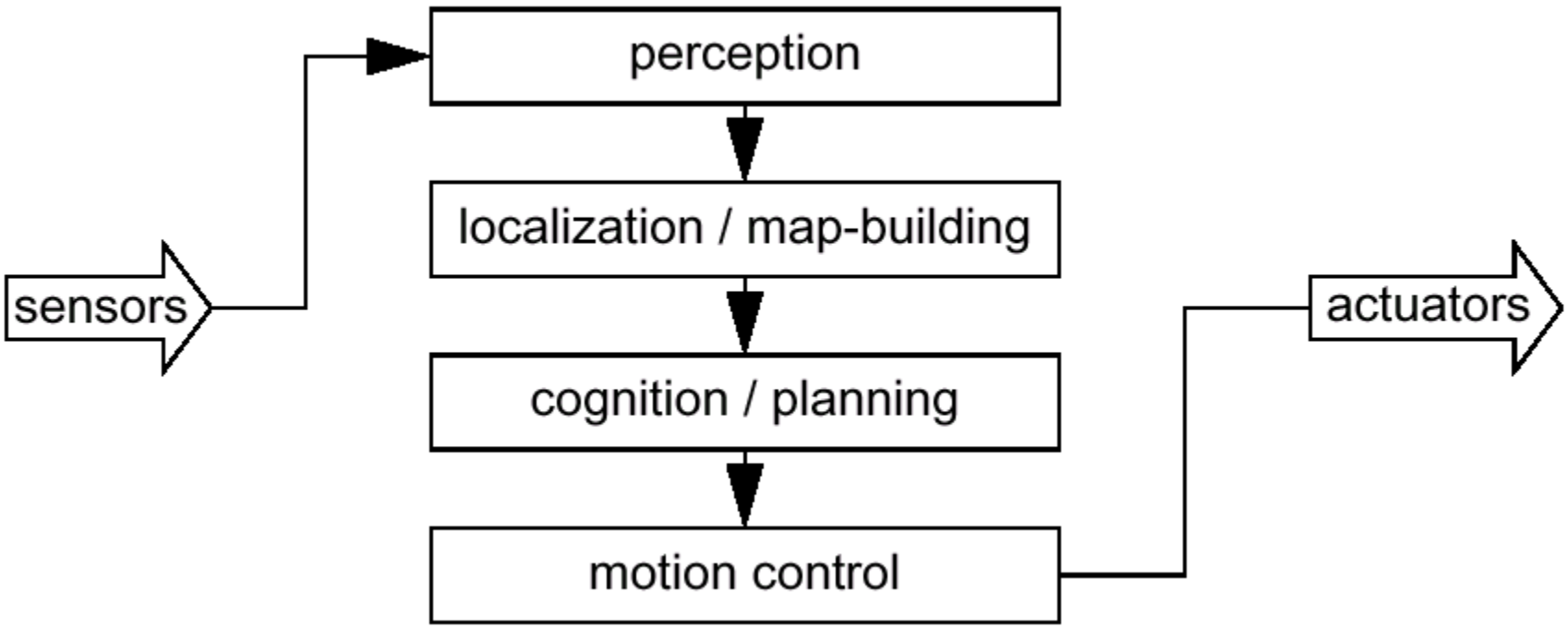


Classical Paradigm

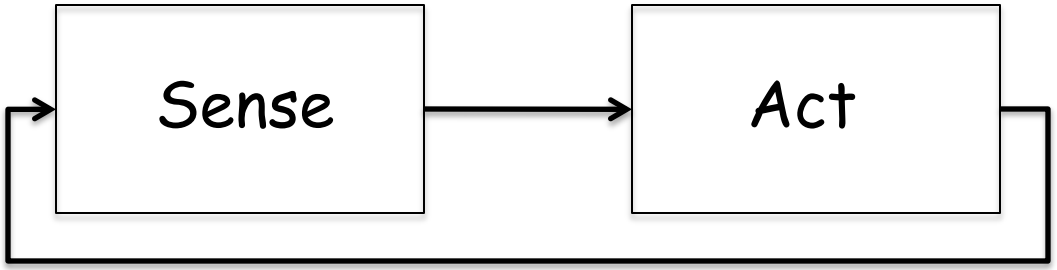


- Historically, focus on automated reasoning,
- Planning and problem solving
- STRIPS – Stanford Research Institute Problem Solver
- Blocks of words problem

Model Based Navigation



Reactive Paradigm



- No models
- motivated by biological systems
- limitations when scaling up

Reactive Architectures

- no memory – no look-ahead reacts to the current environmental stimuli/ sensory information
- reactive behaviors:
  Feedback controllers are instances of reactive controllers
(mappings between situations and actions
  mapping between state and control input)

Can we achieve bigger functionality if we combine them ?

Simplest scenario one situation one action:
Motivation – biology, V. Braitenberg's Vehicles
one can design simple continuous feedback strategies
or sets of if-then rule state rules.

Reactive Paradigm
Subsumption Architecture

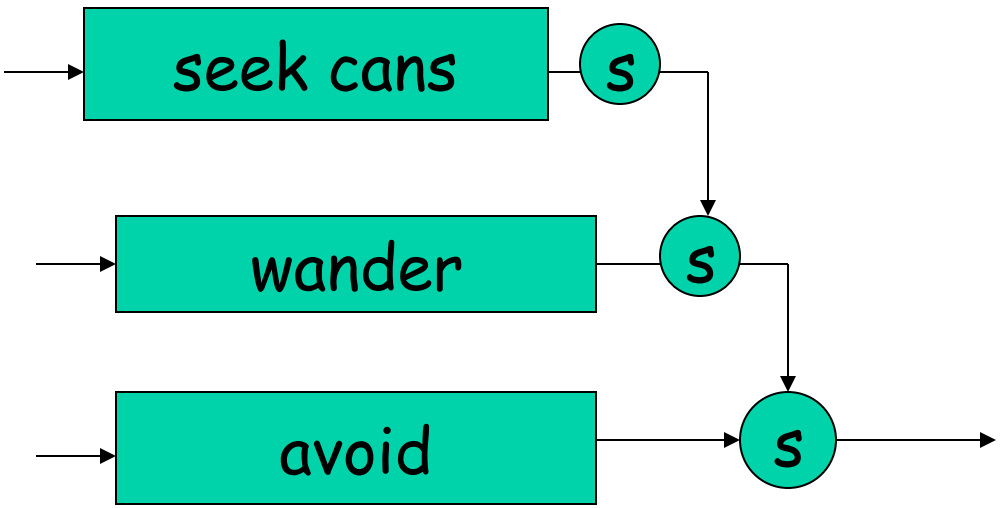
Guidelines:

- Build the system from bottom up
- Components are task achieving behaviors
- Components are executed in parallel
- Components are organized in layers
- Lowest layers handle most basic tasks
- Higher levels exploit the lower levels
- Each components has its tight connection between Perception and action

Bottom up design process

- Introduced by Brooks (MIT) 1996

Subsumption architecture



- Each module is direct mapping between sensors and actions
- Easy software design, good modularity

Subsumption architecture

No model of the world
# of tight feedback loops – reactivity, robustness
inflexibility at run-time, needs expertise for the design

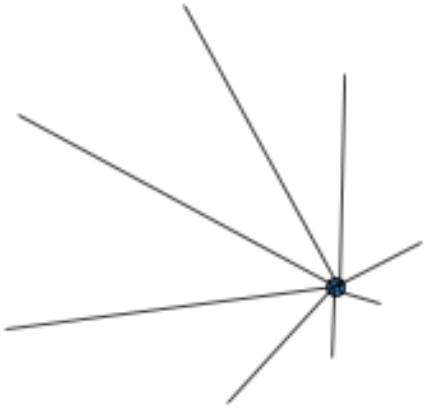
Model of the behavior is Augmented Finite State Machine
AFSM's connected with communication wires
pass input and output messages

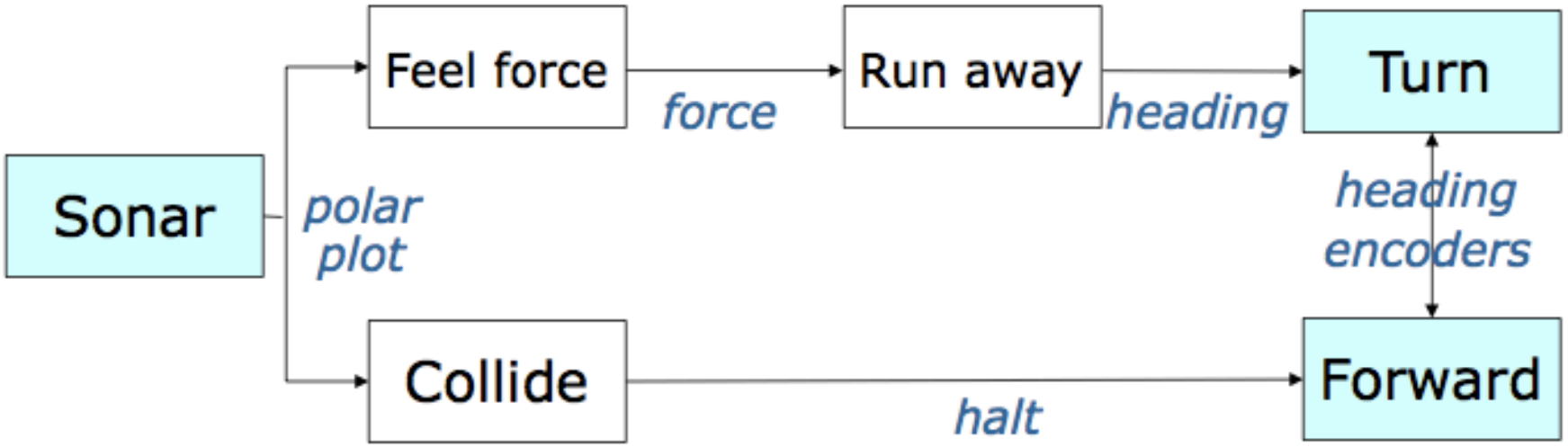
Results in fixed based priority arbitration

Coupling between the layers can be done through the world:
e.g. HERBERT soda can searching robot (Jon Connell, MIT)
1. If you see soda can grab it
2. If its heavy put it down
3. If its empty pick it up
… no notion of the model of the world or connections between behaviors

Level 1: Avoid

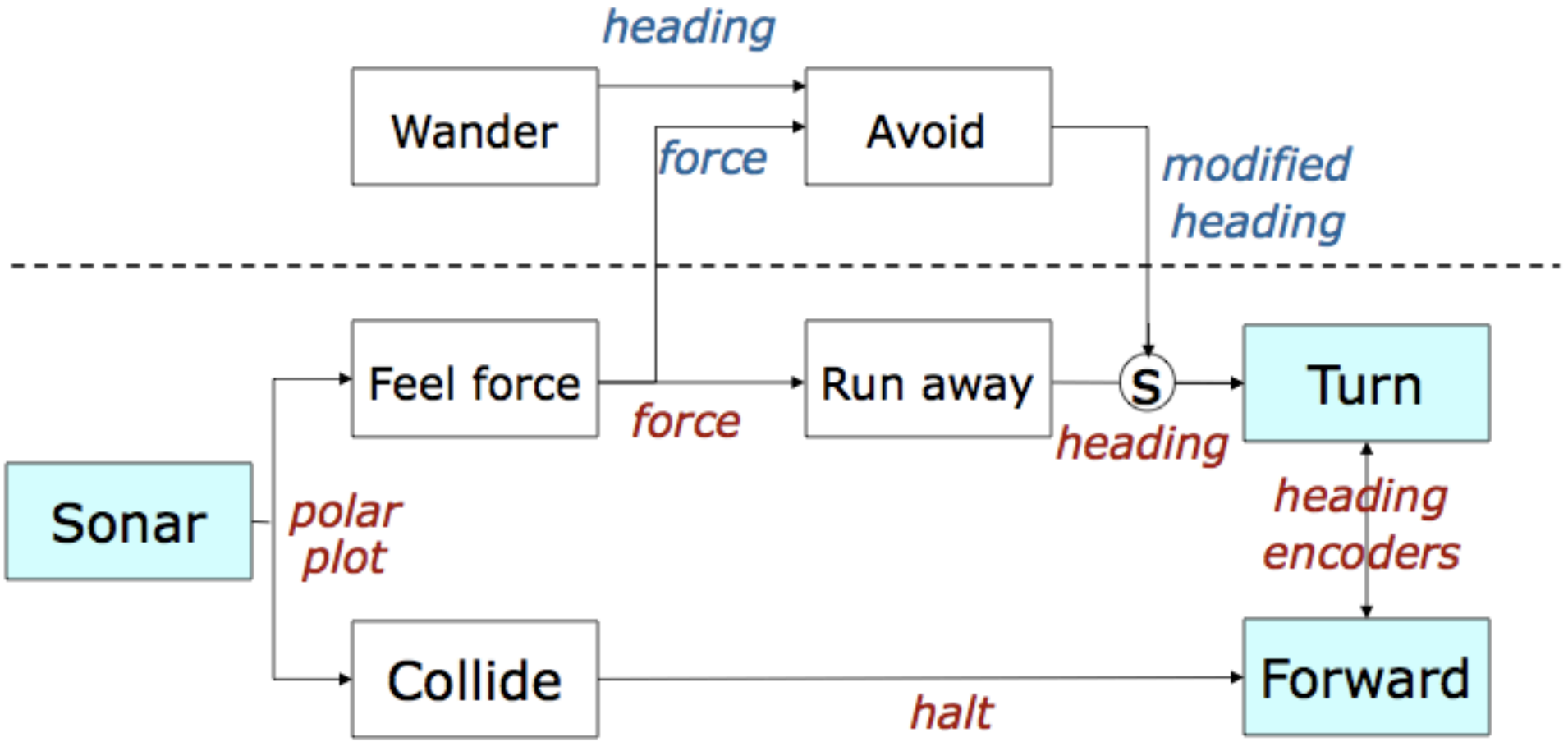
Polar plot of sonars





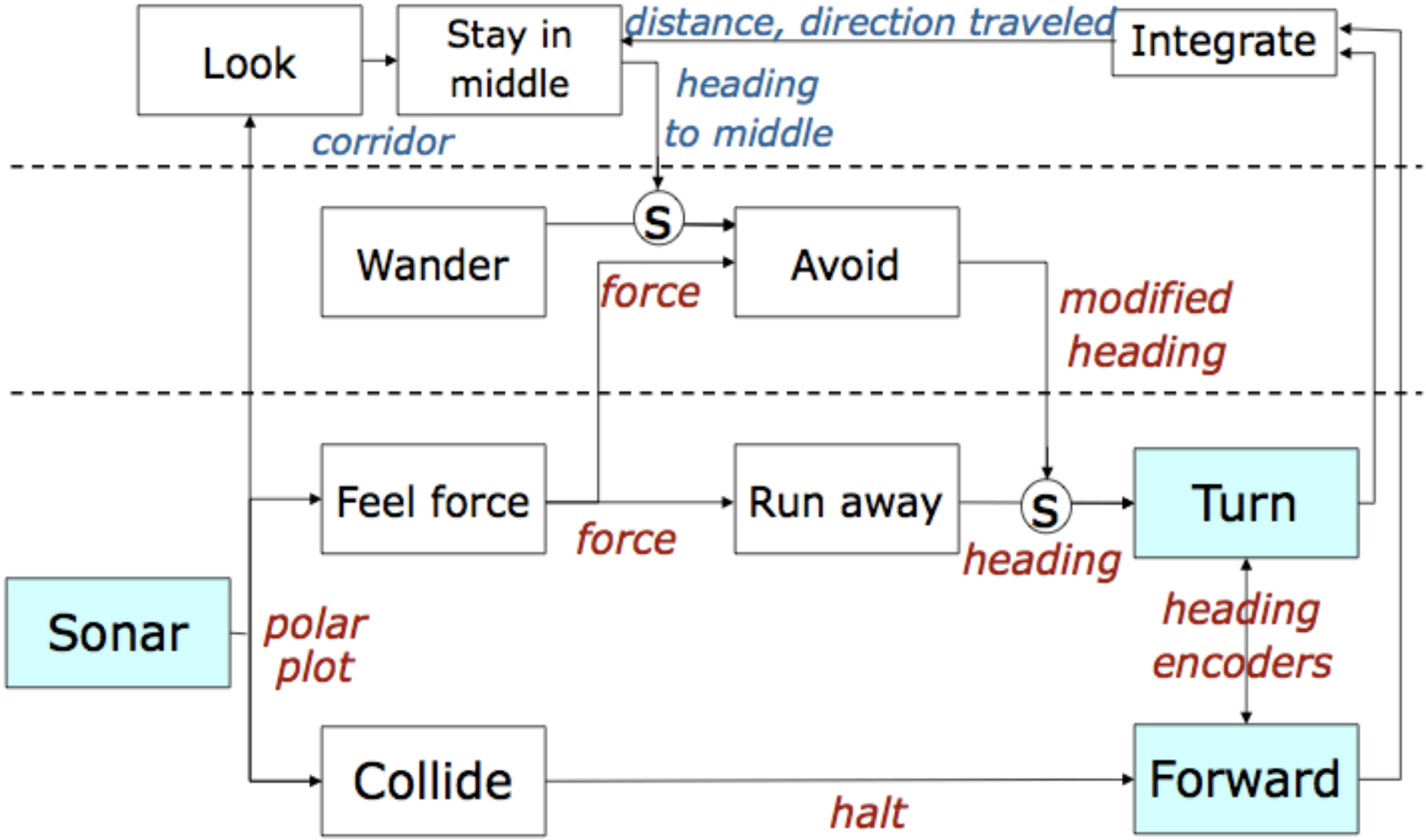
Slide coutresy: Uni Freiburg: Burgard, Stachniss, Bennewitz, Arras

Level 2: Wander



Slide coutresy: Uni Freiburg: Burgard, Stachniss, Bennewitz, Arras

Level 2: Follow Corridor



Slide coutresy: Uni Freiburg: Burgard, Stachniss, Bennewitz, Arras

Subsumption architecture claims:

World is is best model
"Intelligence" without representation

All the relevant parts of the systems interact with the each other through sensing and the world

Many instances of robots build using this philosophy.

Strengths: reactivity, parallelism, incremental design (robustness)
Weakness: needed expertise at the design and inflexibility at the run time

Issues of representation

- Before we discussed the model of the robot
  kinematics, dynamics characterize the motion of the robot
  and the way how it interacts with the environment
  (lower part of the previous diagram)
  feedback- control – no memory, stimulus-action pairs
  determined the next step

- Model of the environment where the robot resides
- Map of the environment (static/dynamic)
- Representation of the environment is the distinguishing feature
  of the robot architecture (we discuss different choices later)

Motivational examples

Motivation

Valentino Braitenberg: Vehicles

>> vehicles with different personalities

Walter Grey: Tortoise

analog implementation, one sensor per one effector

>> light seeking behavior

Behavior-Based Architecture

We had previously examples of behaviors

Feedback controllers, task-achieving behaviors
- design motivated by potential field based technique

How to composed them ? Some examples of composition
- superposition (motivated by potential field techniques)

Behaviors

1. Behaviors are feedback controllers
2. Behaviors are executed in parallel
3. Achieve specific goals (avoid-obstacles, go-to-goal)
4. Can be combined to achieve more complex networks (make inputs of one behavior, outputs of another)
5. Behaviors can be designed to look-ahead, build and maintain representation of the world

Representation of behaviors

- Behavior is mapping from state to control command

Different representations

- Feedback controllers - gradient of some potential function
- Lookup table
- Stimulus/response diagrams
- Discrete and/or continuous representations
  (differential equations or if-then rules ->
  wall-following example)

Potential Field Representations

Continuous representation

Potential field techniques (attractive, repulsive potential fields)

Schema (more general approach to vector fields design) (e.g. goto goal, follow corridor) – Arbib 81

issues with superposition - local minima, maxima, oscilatory behavior

Potential Field Methods

- Idea robot is a particle
- Environment is represented as a potential field (locally)
- Advantage – capability to generate on-line collision avoidance

Compute force acting on a robot – incremental path planning

$$F(q) = -\nabla U(q)$$

Example: Robot can translate freely , we can control independently
Environment represented by a potential function

$$U(x, y)$$

Force is proportional to the gradient of the potential function

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\nabla U(x, y)$$

Attractive potential field

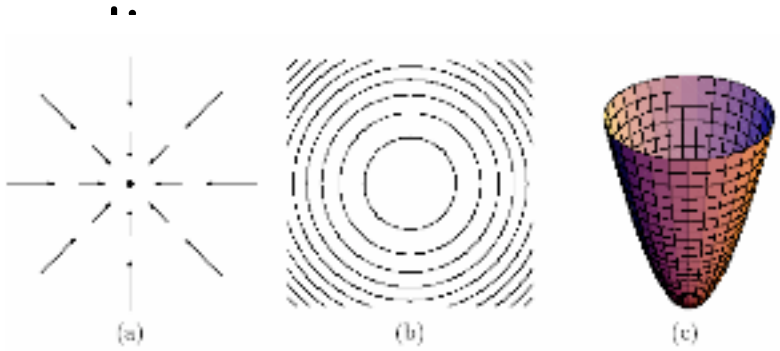
- Linear function of distance

$$U_a(q) = \xi \|q - q_{goal}\| \qquad F_a(q) = -\nabla U_a(q) = -\xi \frac{(q - q_{goal})}{\|q - q_{goal}\|}$$

- Quadratic function of distance

$$U_a(q) = \xi \frac{1}{2} \|q - q_{goal}\|^2 \quad F_a(q) = -\nabla U_a(q) = -\xi (q - q_{goal})$$

Combination of two – far away
closer by use parabolic well

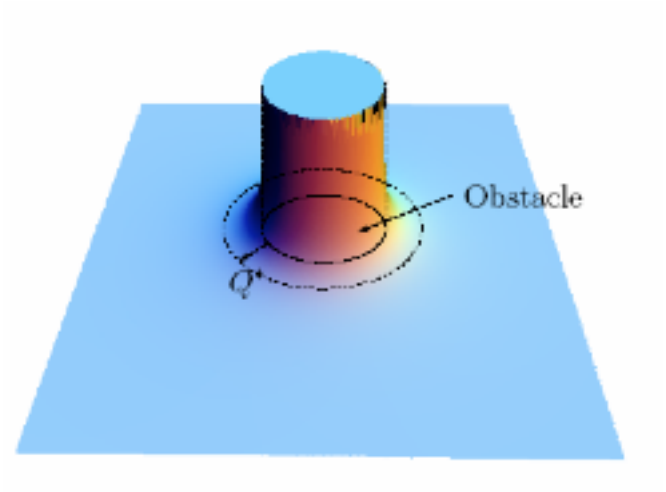


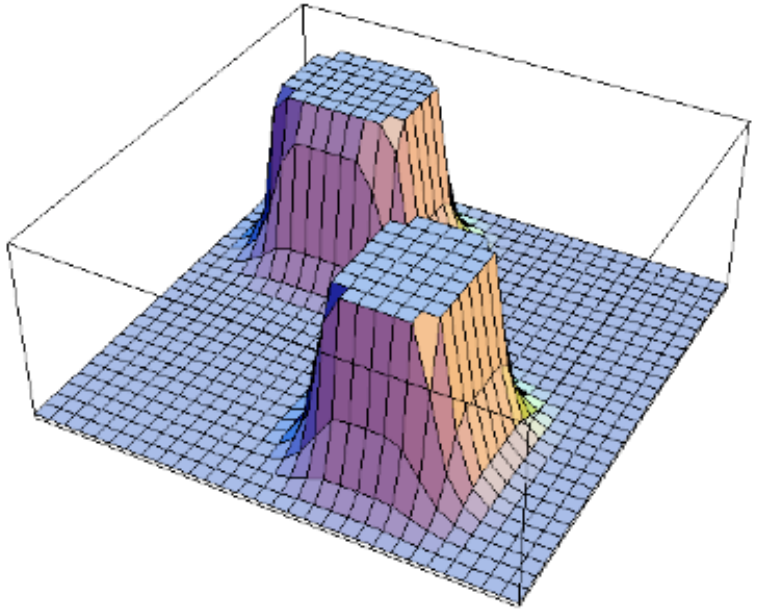
Repulsive potential field

$$U_r(q) = \frac{1}{2}\nu\left(\frac{1}{\rho(q,q_{obst})} - \frac{1}{\rho_o}\right) \qquad \text{if} \qquad \rho(q, q_{obst})\| \leq \rho_0$$

$$\text{else} \qquad U_r(q) = 0$$

Minimal distance between the robot and the obstacle

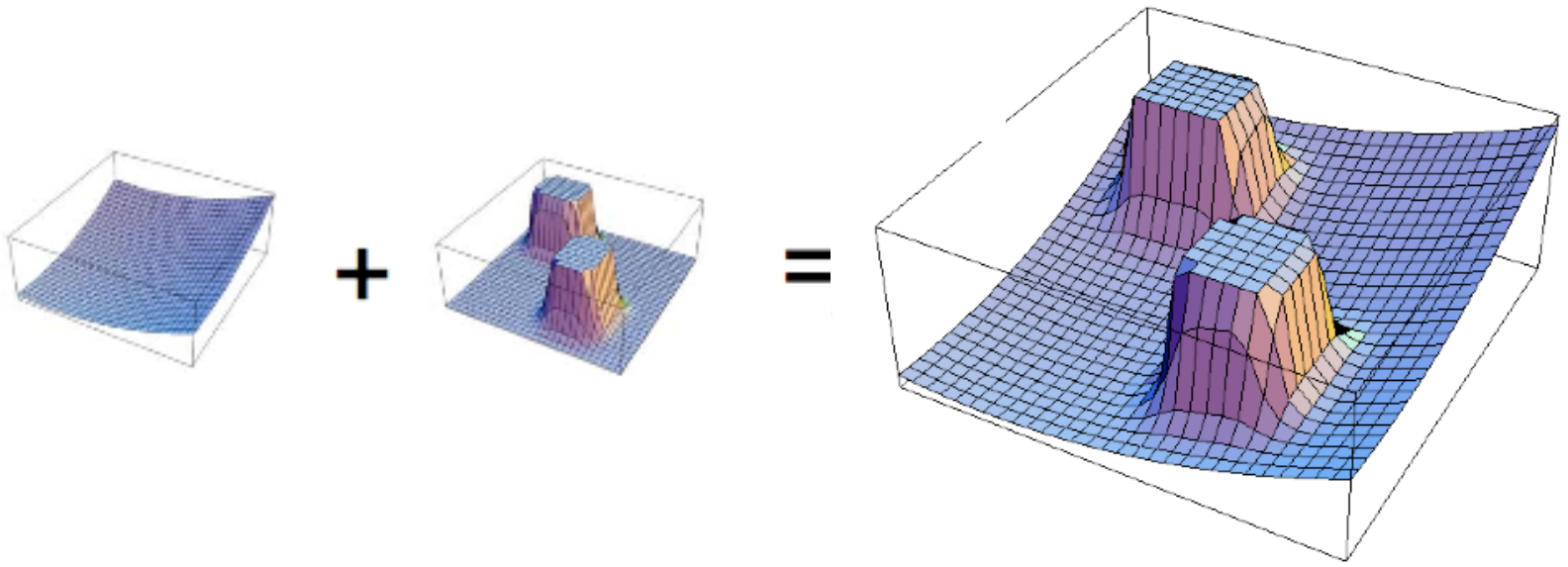




Potential Function

Resulting force $F(q) = -\nabla(U_a(q) + U_r(q))$

Iterative gradient descent planning $q_{i+1} = q_i + \delta_i \frac{F(q)}{\|F(q)\|}$

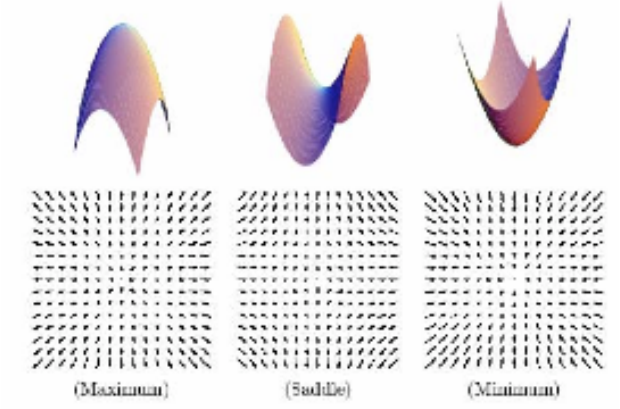


Potential Fields

- Simple way to get to the bottom, follow the gradient

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\nabla U(x, y) \qquad \dot{q} = -\nabla U(q)$$

- i.e. Gradient descent strategy $q_{i+1} = q_i + \delta_i \frac{F(q)}{\|F(q)\|}$
- A critical, stationary point is such that $\nabla U(q) = 0$
- Equation is stationary at the critical point



- To check whether critical point is a minimum – look at the second order derivatives (Hessian for m -> n function)

Repulsive potential field

$$U_r(q) = \frac{1}{2}\nu\left(\frac{1}{\rho(q,q_{obst})} - \frac{1}{\rho_o}\right) \qquad \text{if} \qquad \rho(q, q_{obst})\| \leq \rho_0$$

$$\text{else} \quad U_r(q) = 0$$

Minimal distance between the robot and the obstacle

Resulting force $$F(q) = -\nabla(U_a(q) + U_r(q))$$

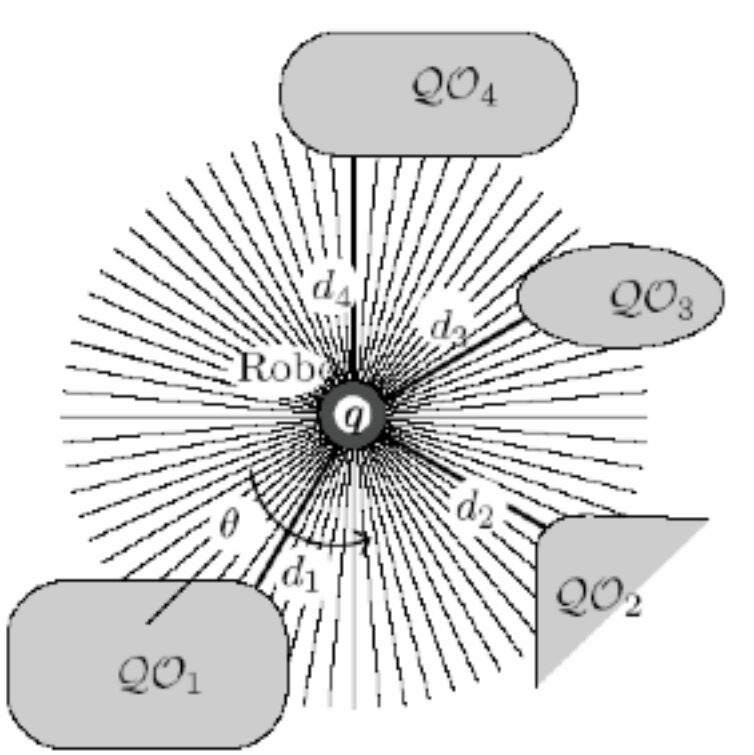
Iterative gradient descent planning $$q_{i+1} = q_i + \delta_i \frac{F(q)}{\|F(q)\|}$$

Issues – multiple obstacles – nonconvex obstacles – how to compute distance
Can be computed for polygonal and polyhedral obstacles

Issues – local minima
Heuristics for escaping the local minima
Can be used in local and global context
Numerical techniques
Random walk methods

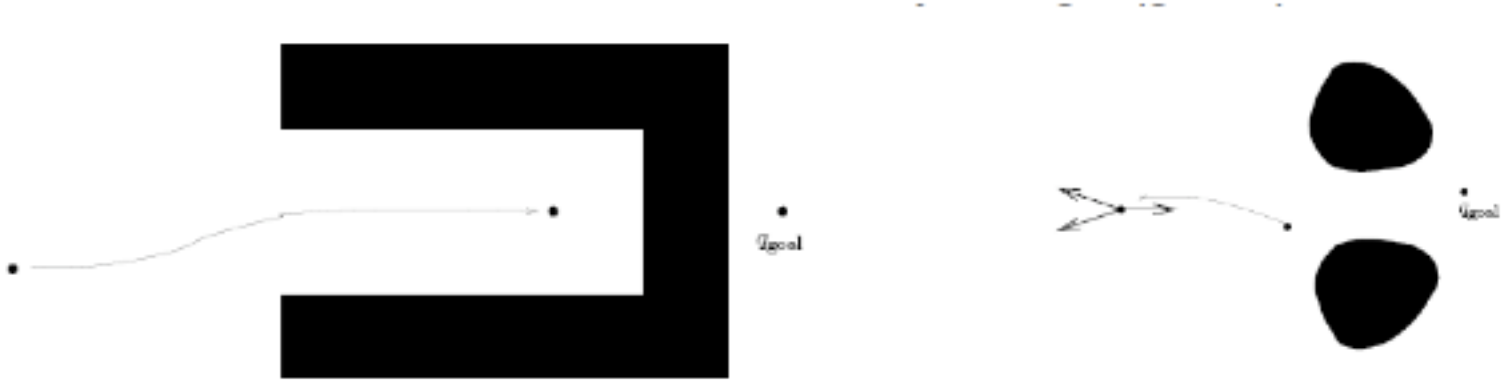
Computing Distances

- Using sensor measurements



Potential Functions

- How do we know we have single global minimum ?



- If global minimum is not guaranteed, need to do something else then gradient descent
- Design functions in such a way that global minimum can be guaranteed

Potential function

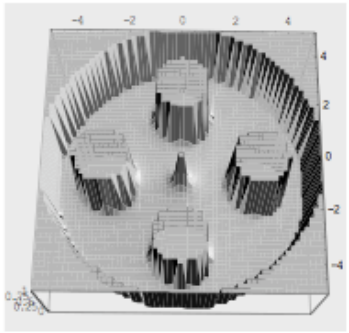
- Heuristics for escaping the local minima
- Can be used in local and global context
- Numerical techniques, Random walk methods

- Navigation functions (Rimon & Kodistchek, 92)
- Navigations in sphere worlds and worlds diffeomorphic to them

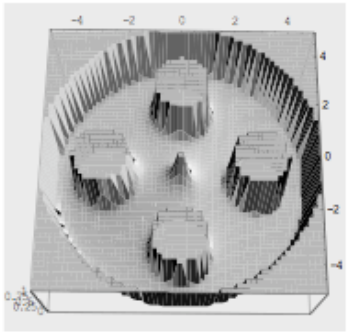
Navigation functions

$$\phi(q) = -\frac{d^2(q, q_{goal})}{[d(q, q_{goal})^{2k} + \beta(q)]^{1/k}} \quad \beta(q)$$ Obstacle term

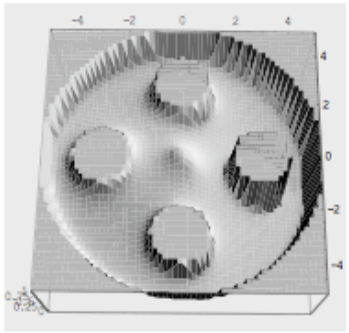
- For sufficiently large k – this is a navigation function [Rimon-Koditschek, 92]



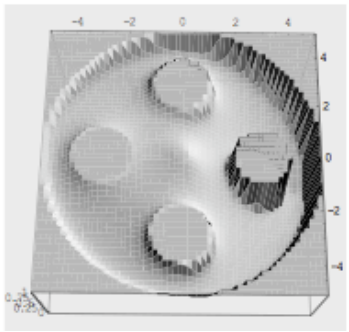
k=3

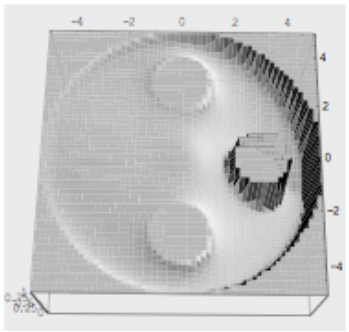


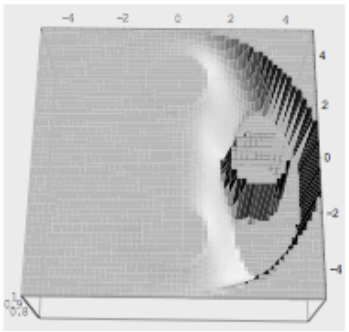
k=4



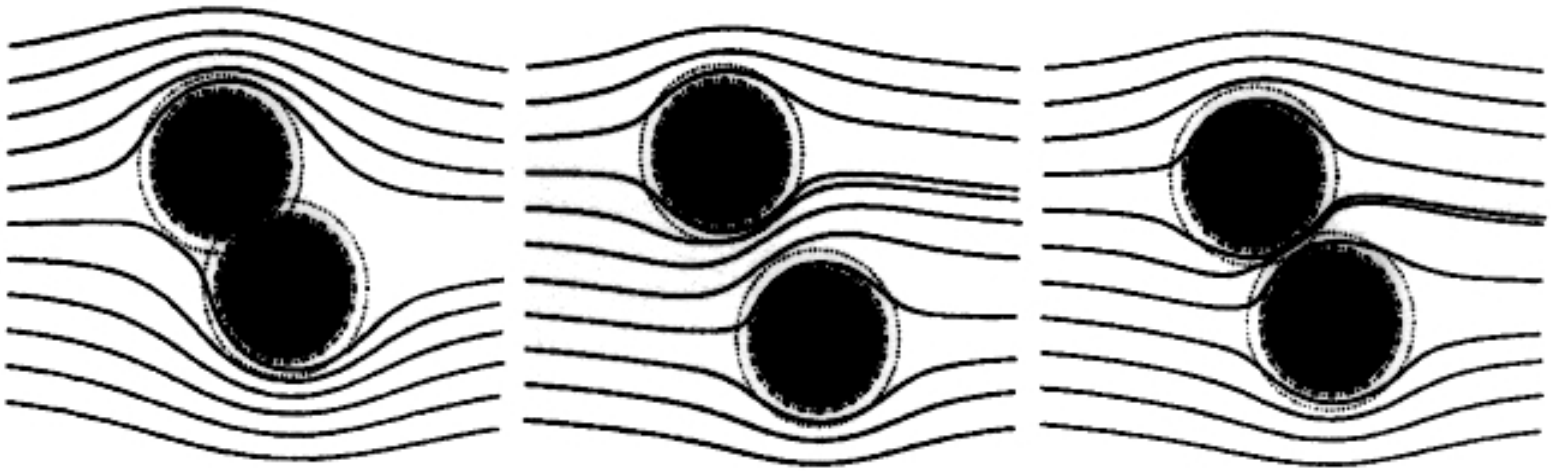
k=6





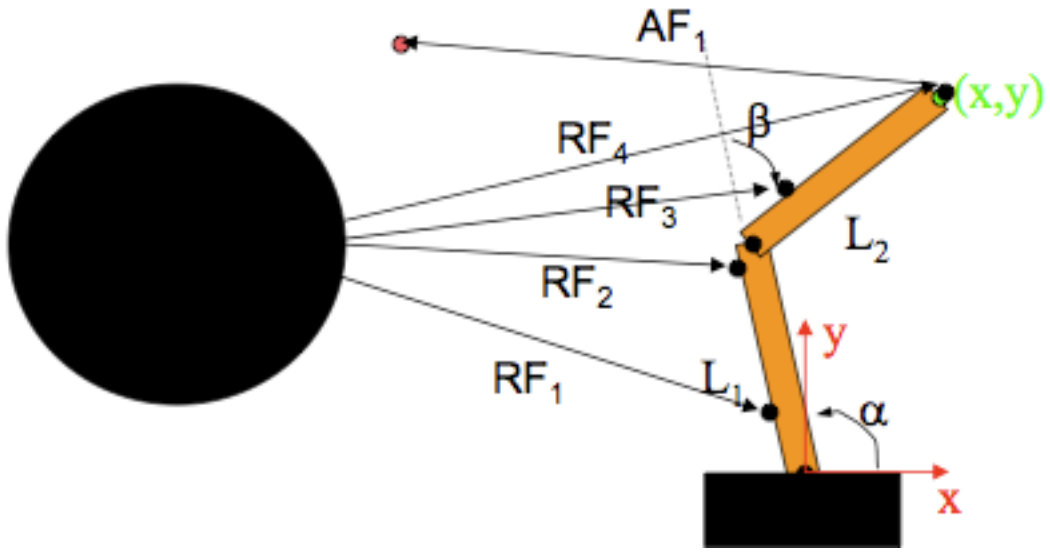


Potential Field Path Planning: Using Harmonic Potentials

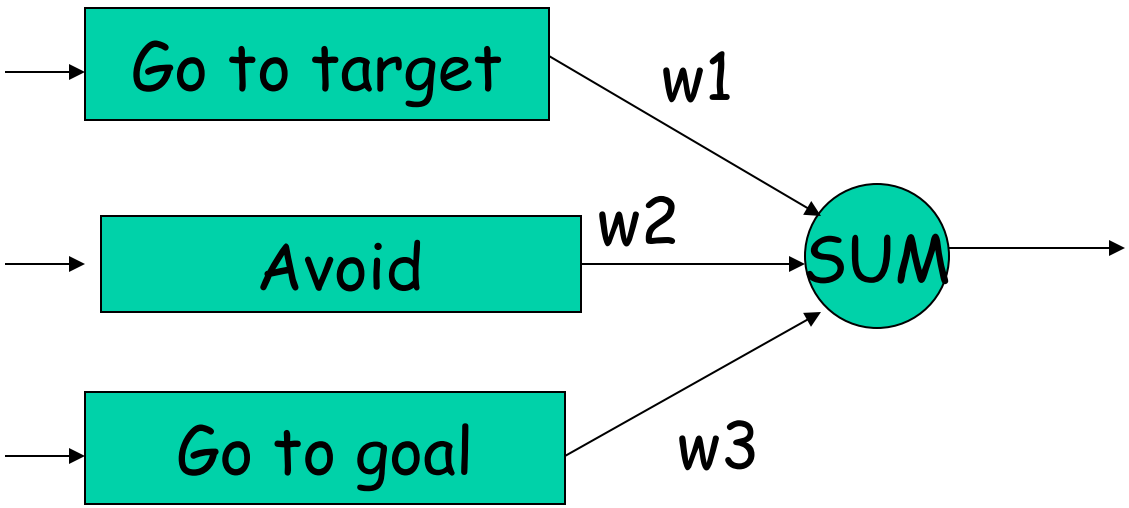
- Hydrodynamics analogy
  - robot is moving similar to a fluid particle following its
- 
- Note:
  - Complicated, only simulation shown

Potential fields for Rigid Bodies

- So far robot was considered a point- gradient of the potential function – force acting on a point
- how to generalize to manipulators of objects ?
- Idea – forces acting on objects – forces acting on multiple points of the object
- http://www.cs.cmu.edu/~motionplanning/
- For robots, pick enough control points to pin down the robot – define forces in workspace – map them to configuration space



Superposition of different behaviors



Behavior-based Architecture

Motivation

1. To keep all the advantages of the Reactive Control
2. Allow representation of the environment
3. Allow bigger flexibility and reconfiguration depending on the task

(… this is what subsumption architecture was lacking)

Behavior Assemblages

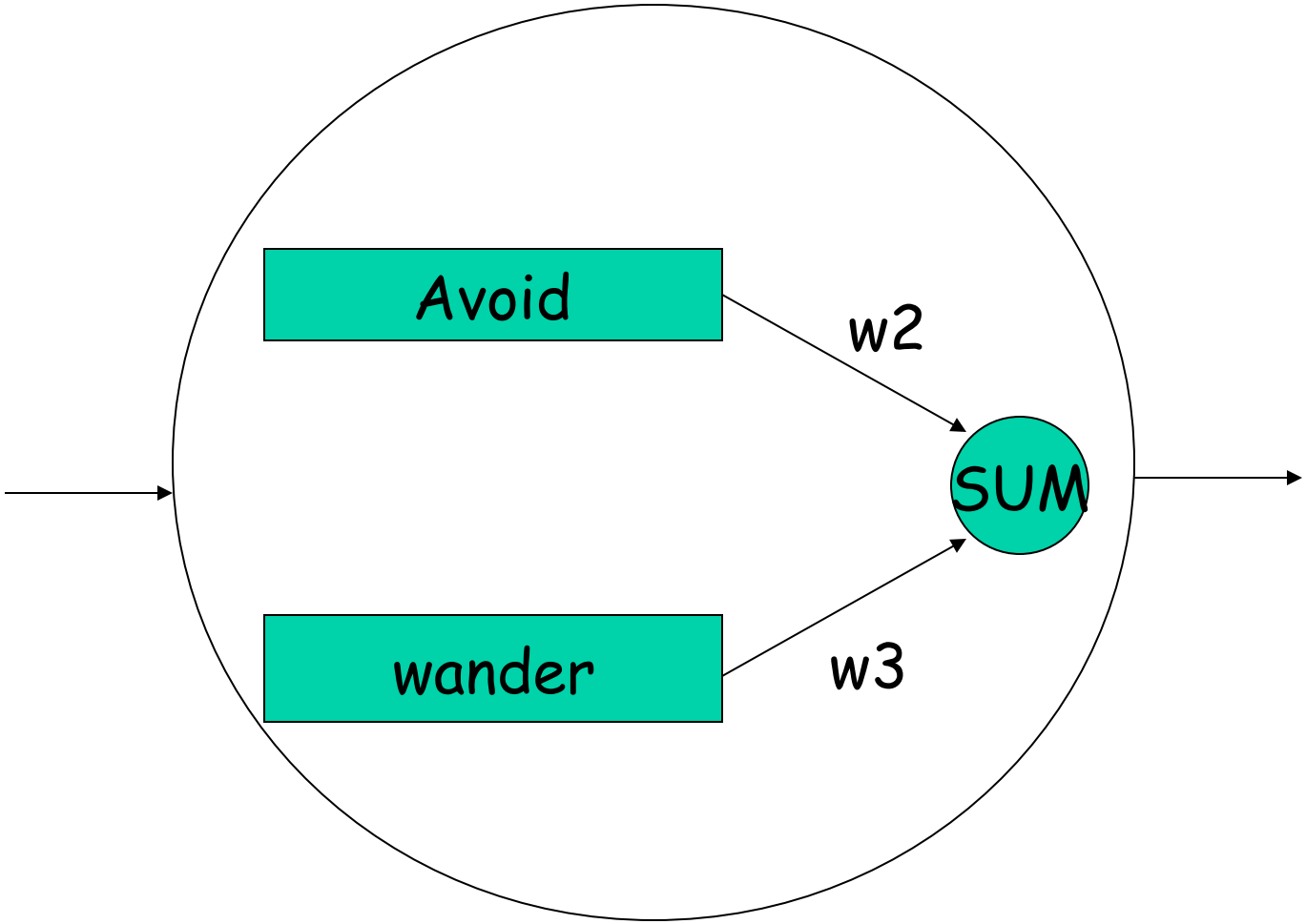
- Power of abstraction
- Modularity
- Reuse of elementary behaviors
- "reason" over them
- Coarser level of granularity – good for adaptation and learning

Abstraction's in terms of FSM's

Composition of the behaviors

Examples: FSM for navigation
Pole finding robot (*AAA* competition)

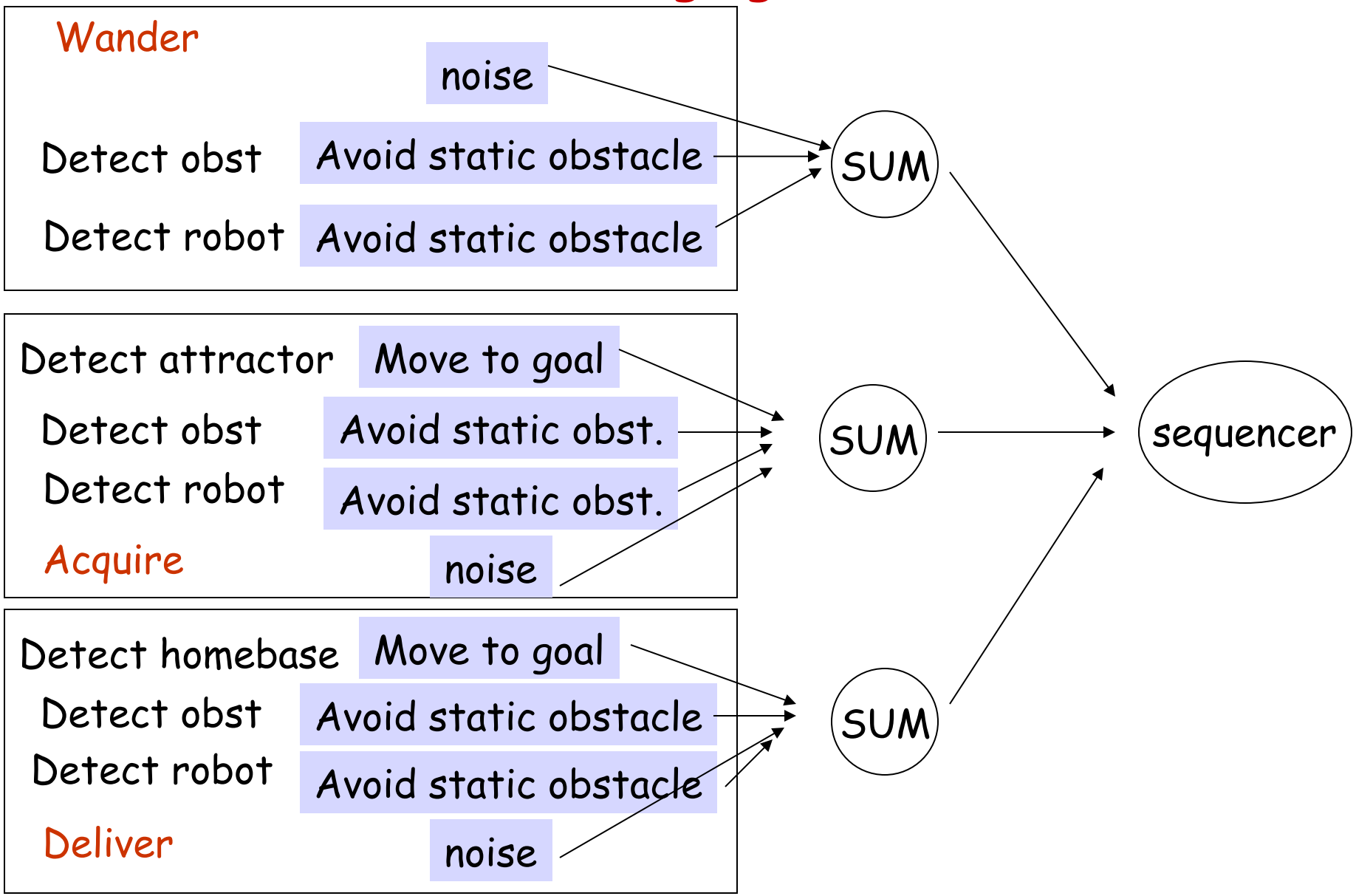
Wander and avoid behavior assemblage

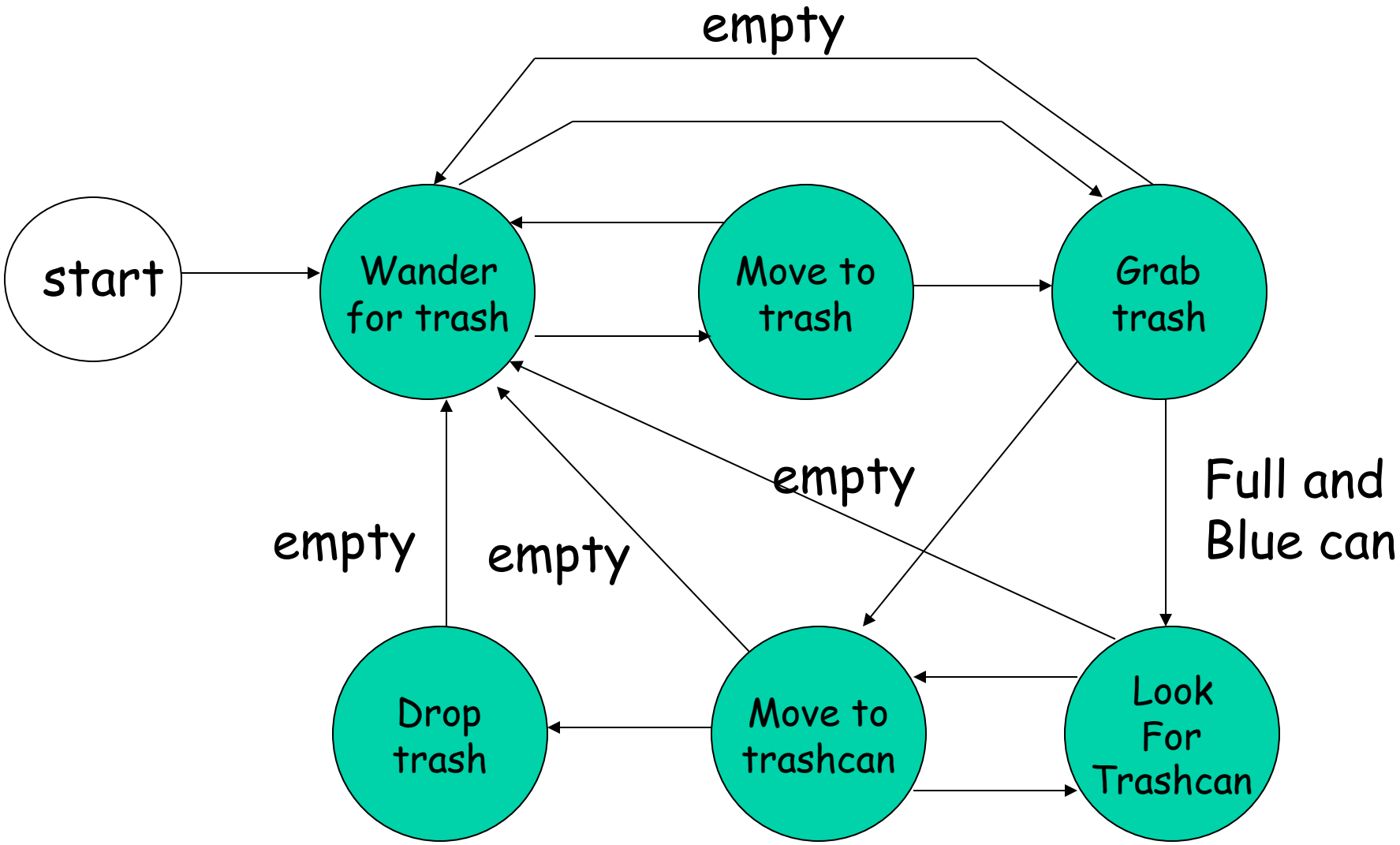


Different behaviors/motor schemas

- Move-ahead
- Move-to-goal
- Avoid-obstacle
- Dodge – sidestep an approaching ballistic projectile
- Escape – move away from projected intercept point between robot and approaching predator
- Stay-on-path
- Noise
- Follow the leader
- Probe – move towards open areas
- Dock- approach an object from particular direction
- Avoid-past – move away from recently visited areas
- Move-up, down maintain altitude

Foraging


Wander
noise
Detect obst
Avoid static obstacle
Detect robot
Avoid static obstacle
Detect attractor
Move to goal
Detect obst
Avoid static obst.
Detect robot
Avoid static obst.
Acquire
noise
Detect homebase
Move to goal
Detect obst
Avoid static obstacle
Detect robot
Avoid static obstacle
Deliver
noise
SUM
SUM
SUM
sequencer



Example of trash collecting robot – each node is an assemblage of behaviors - more details on the transitions

Behavior Composition

Programming language for behavior composition
Elementary behaviors FSM's
Composition operators:

1. Sequential B1; B2
2. Conditional B1 : B2
3. Parallel B1 || B2
4. Disabling B1 # B2
5. Iterative B1:; B2, B1 :: B2

Examples of more complex tasks as networks of elementary behaviors (behaviors can communicate via shared memory)

Example : Classroom navigation , Clean Up, Foraging

Emergent Behaviors

Apparently new behaviors can "emerge" from

Interactions of rules
Interactions of behaviors
Interactions with the environment

- Since the behavior is just input output mapping externally observed
- Occasionally explicitly un-modeled interactions/behaviors can be observed
- Notion of emergent behavior – intuitive, not well defined except for some simple scenarios (wall following example, flocking, dispersing, foraging)

Emergent behaviors

Flocking example

1. Don't run into another robot
2. Don't get too far from other robots
3. Keep moving

Unexpected vs. emergent

- depends on the observer - subjective notion

Due to the un-modeled uncertainties, the behaviors are not exactly repeatable/predictable

Emergent behaviors can be achieved from parallel execution of many behaviors

For the purpose of analysis – undesirable phenomena

Streering behaviors

http://www.red3d.com/cwr/steer/

Behavior Composition

So far all the examples of behaviors were just reactive controllers represented in continuous or discrete manner

How to design more sophisticated behaviors ?
e.g. to achieve map building of the environment

- Map building in the context of Behavior-Based Architecture
- What type of map representation would fit ?
- Representation needs to satisfy the premises of the behavior-based architecture

Example: Toto the map building robot
M. Mataric, Learning a distributed map representation of the environment based on navigation behaviors, 1992

Distributed Map Building

- Map cannot be centralized CAD-CAM model
- Idea: distribute different parts of the map over different behaviors
- Individual behaviors will be responsible for the portions of the map
- They will be connected in such a way that the connections will reflect the adjacency in the actual model of the environment

- Notion of a landmark – particular part of the environment which the robot can easily recognize
- Design behaviors to detect particular landmarks – the choice of landmarks, depends on the types of sensors used

Distributed Map building

Three types of landmarks
1. Walls
2. Corridors
3. Irregular – messy areas

Toto moves around, safely and detects landmarks
Each landmark behavior remembered some information about (landmark type, heading and length)

Map building - continued

1. Every time new landmark was discovered, new behavior was Added and connected to the previous behaviors via communication wires

2. If no behavior was activated – landmark is new – added
   if existing behavior was activated – inhibit other behaviors (example)

3. Was also used for path planning
   Path can be optimized based on the length of individual segments, localization within the topological map

Distributed Map Building

Different layers:

1. Move around safely
2. Detect new landmarks
3. Build a map and localize within the map
4. Find a path towards a goal

Real-time reactive behavior – no central place for keeping The representation of the environment (as opposed to deliberative architecture)

Behavior Based Architecture - Summary

Pros and cons

1. Distributed, operate at larger time scales
2. Maintain the reactivity
3. Can form and maintain representations
4. Notion of the emergent behaviors
   (something which is hard to model and hard to analyze)
   can be achieved by superposition
   Difficult to provide any guarantees
5. Behavior Based approached exploit emergent behaviors
   in some instances they have to be avoided

Multi-agent Architectures

- Advantages of having groups of robots achieving some tasks (exploration, manipulation, demining) simpler less expensive robots + cooperation, can achieve more complex tasks
- Taxonomy of the collectives – size, communication range, communication topology, reconfigurability (of topology or physical)
- Homogeneous or Heterogeneous
- Processing ability – centralized, distributed
- Mothership and team concepts

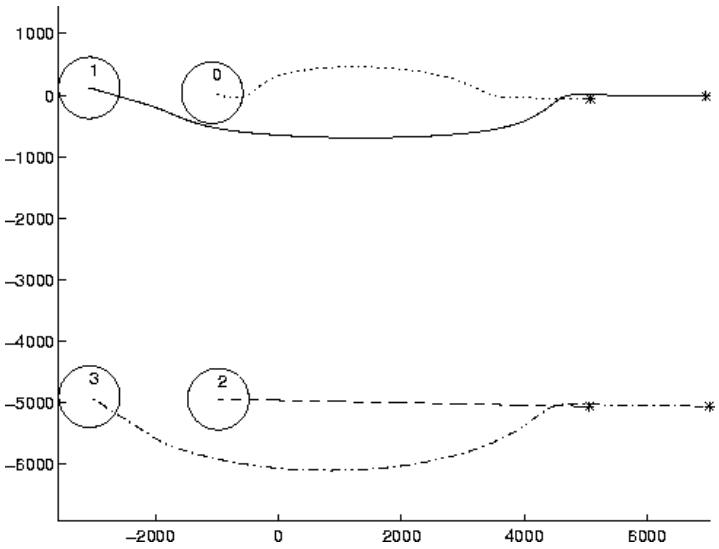
Example – Multi-robot Collision Avoidance

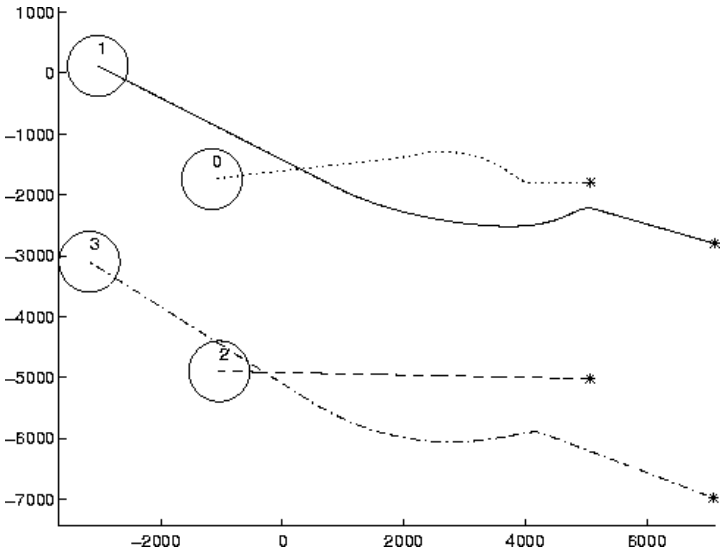
Air traffic Control – Collision Avoidance

- Distributed multi-agent motion planning approach
- Potential and Vortex field based motion planning
- Generation of the prototype maneuvers
- 2D planar conflict resolution
- 2-1/2D conflict resolution

Collision Avoidance - Vector field based approach

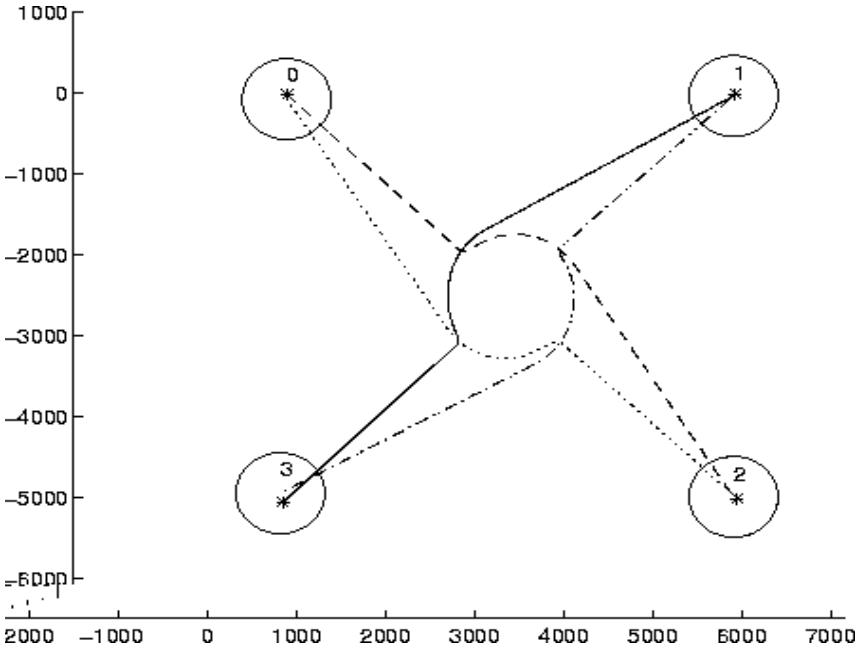
- Superposition of participating vector fields





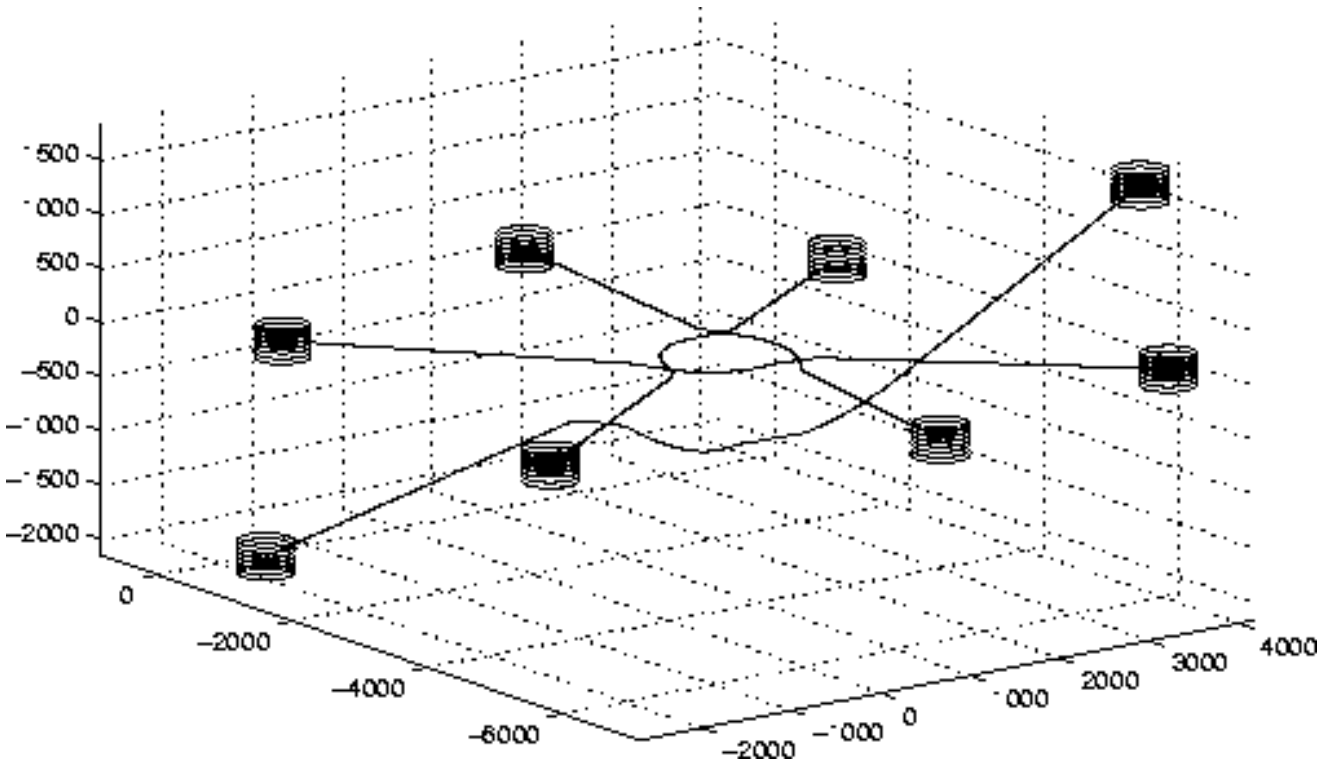
Overtake maneuver

Collision Avoidance – Prototype Maneuvers



Roundabout maneuver

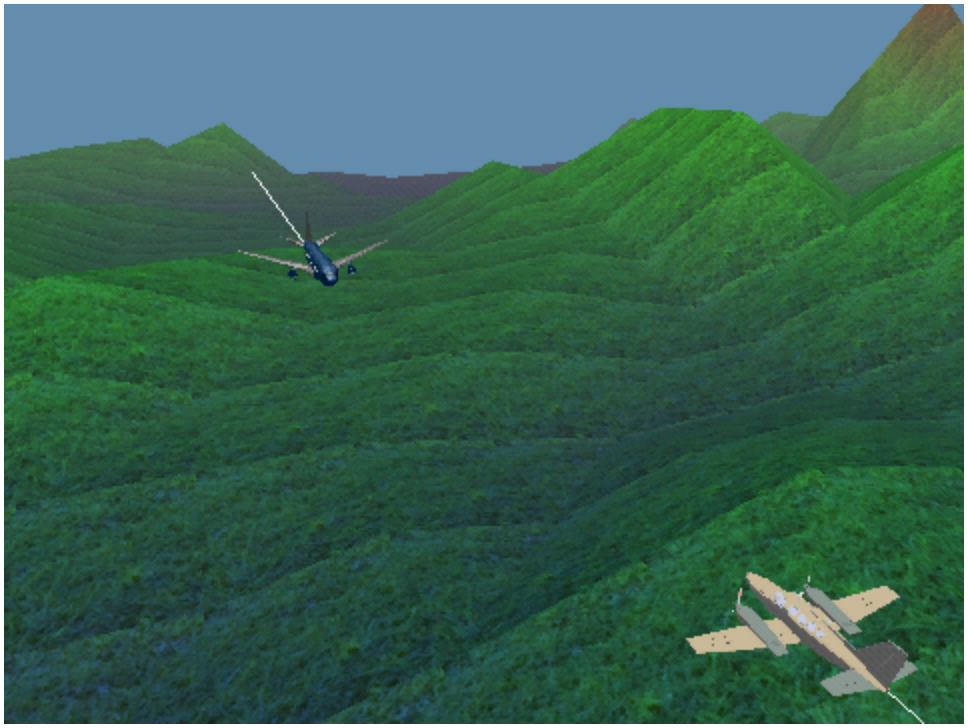
Collision Avoidance – Prototype Maneuvers



2-1/2 D avoidance maneuver,
horizontal and vertical conflict resolution

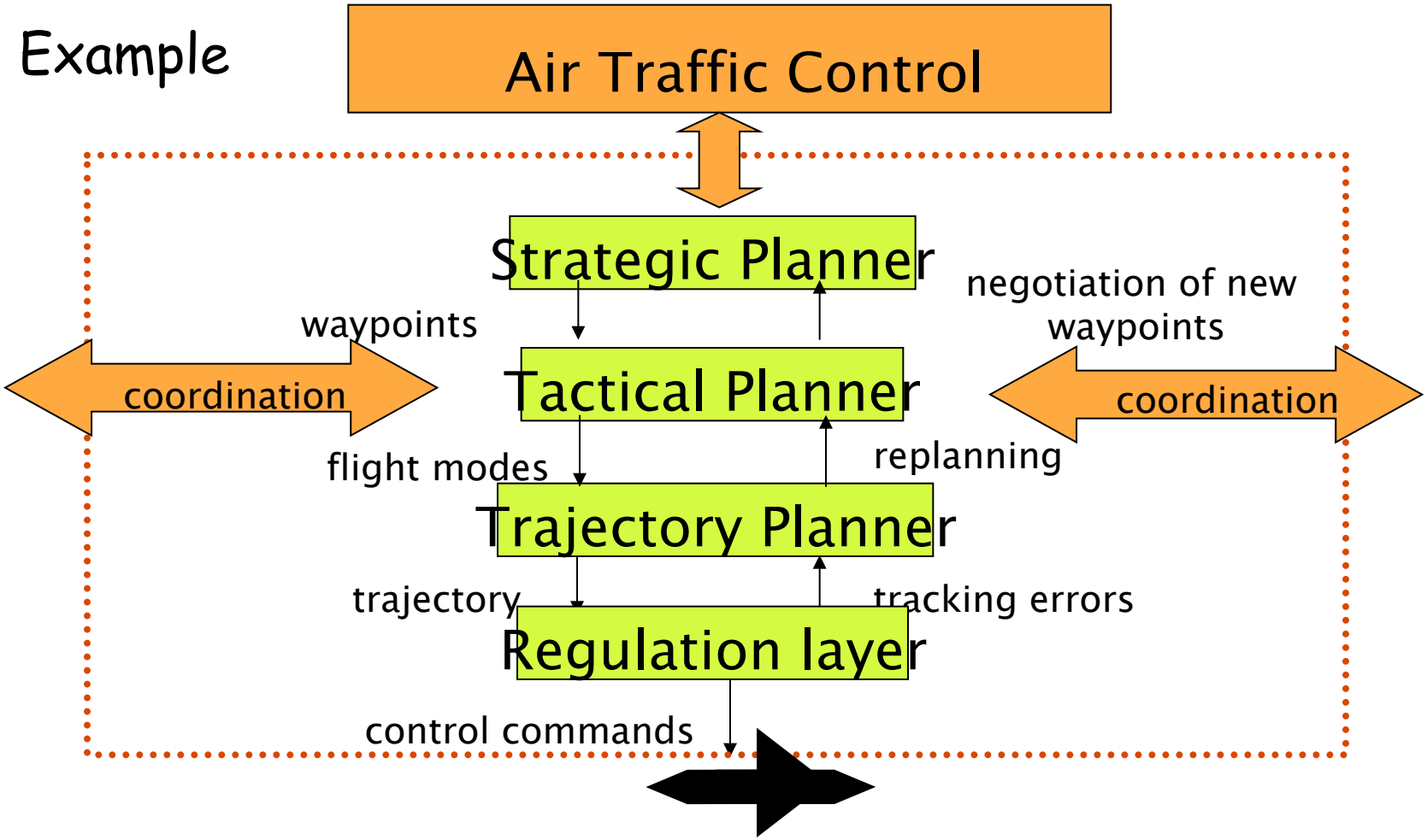
Collision Avoidance – **Visualization**

Collision Avoidance – **Visualization**



Hierarchical Architecture

Example



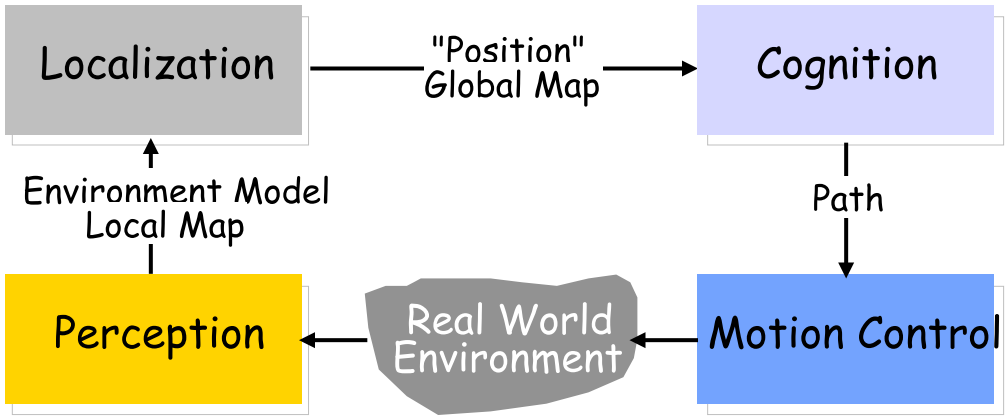
Design and modeling of a hierarchical hybrid systems:

Safety guarantees, Safety vs performance tradeoff

How to do the right thing ?

- How to do the right thing ?
- Go from A to B in the context of navigation
- Taking the environment into account
- Taking the uncertainty into account

- Using the model of the environment or not ?



Perception, Model, Plan, Execute (Motion Controller)

To localize or not?

- How to navigate between A and B
  - navigation without hitting obstacles
  - detection of goal location
- Possible by following always the left wall
  - However, how to detect that the goal