Previously

- Kinematics and Control

Control

- Trajectory following
- Line following
- Point to point control
- Pose to pose control
- Potential Fields goals and obstacles

Robotic Control Paradigms

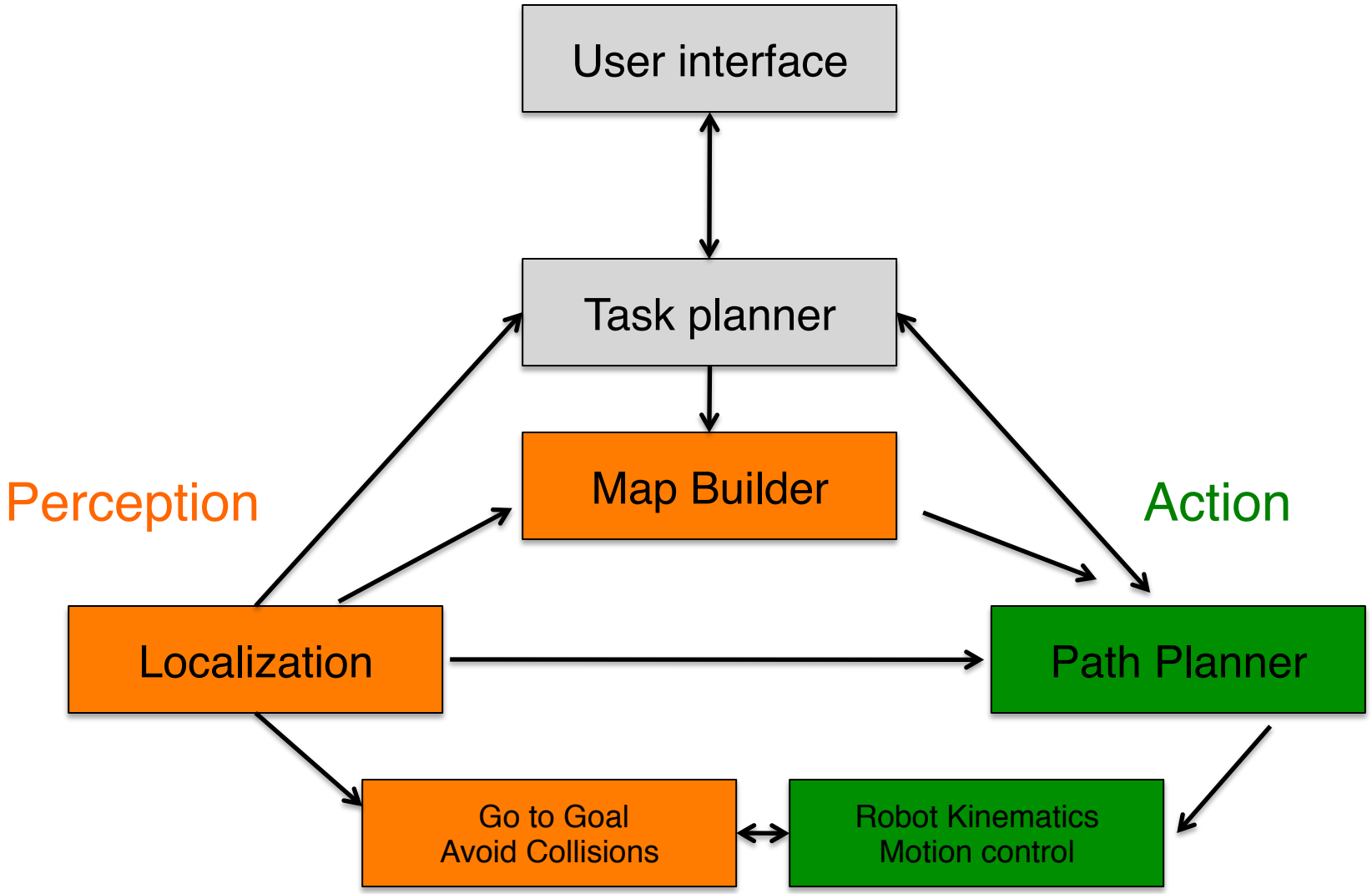
Previously basics of control
- trajectory generation
- closed feedback-loop control

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



Alternatives I. - 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)

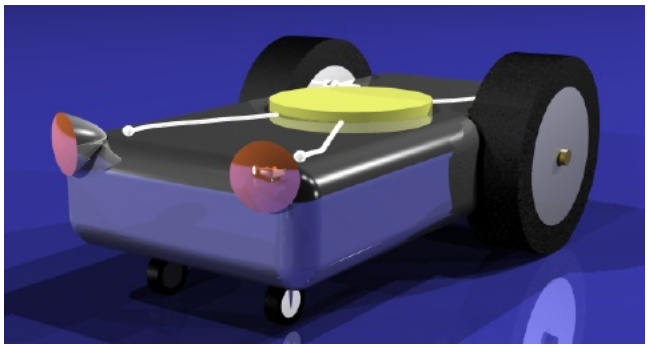
- Can we achieve bigger functionality if we combine them ?

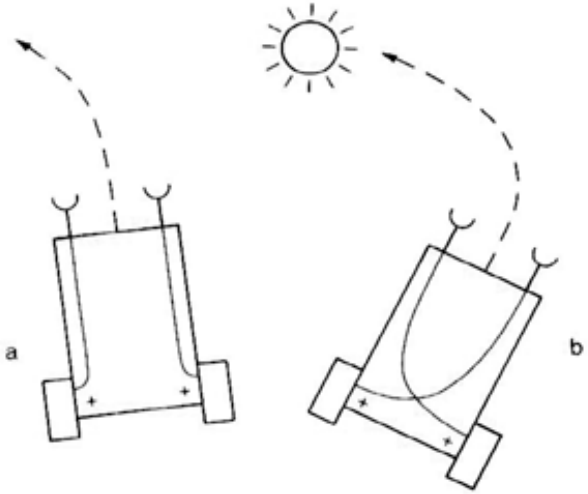
- Simplest scenario one situation one action:

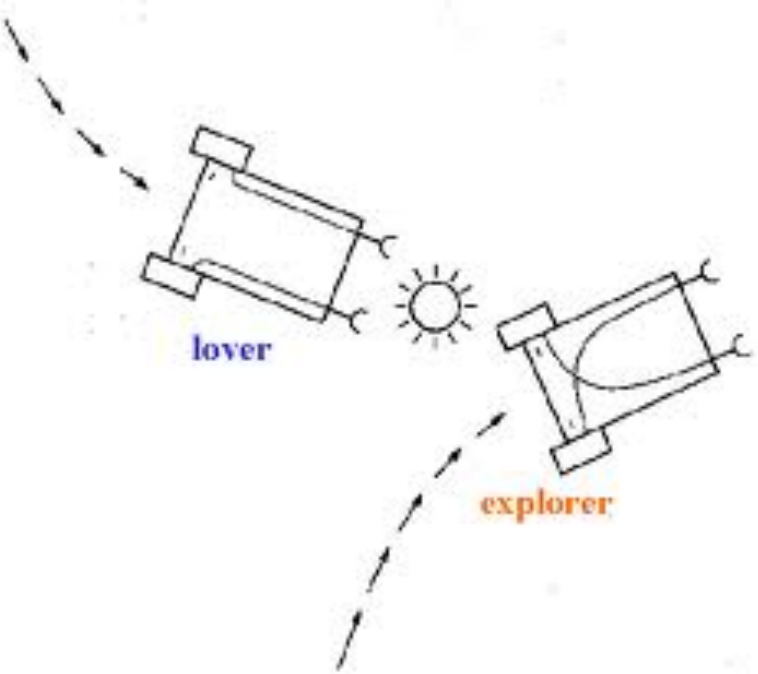
Motivation: Biology, V. Braitenberg's Vehicles
one can design simple continuous feedback strategies
or sets of if-then rule state rules.

Braitenberg's Vehicles

- Simple sensor-actuator wirings
- Seemingly complex behaviors
- Light seeking
- Light avoidance
- Attraction, Repulsion







Reactive Paradigm
Subsumption Architecture

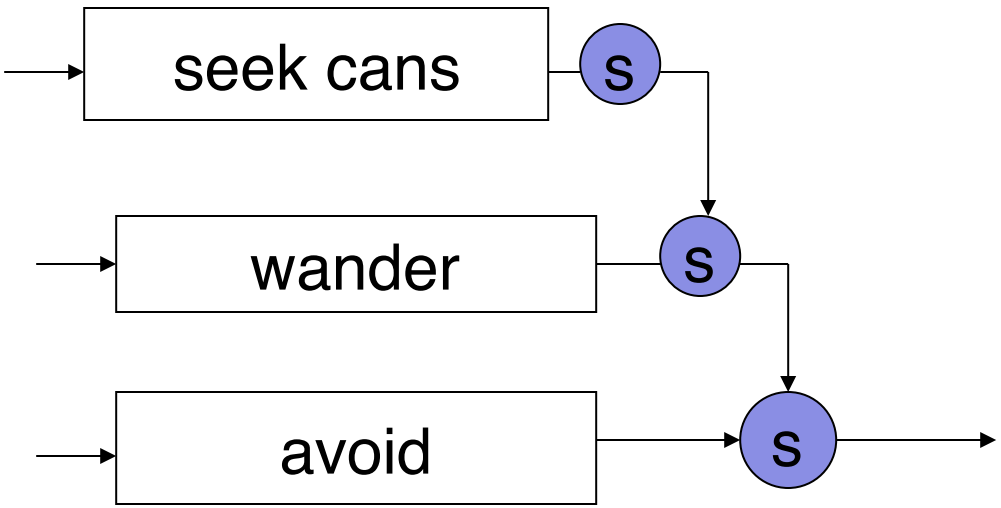
Guidelines:
- Build the system from bottom up
- Components are task achieving behaviors
- Components are executed in parallel
- Components are organized in layers
- Lowest layers handle most basic tasks
- Higher levels exploit the lower levels
- Each components has its tight connection between

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

- Augmented Finite State Machine
- AFSM's connected with communication wires
  pass input and output messages

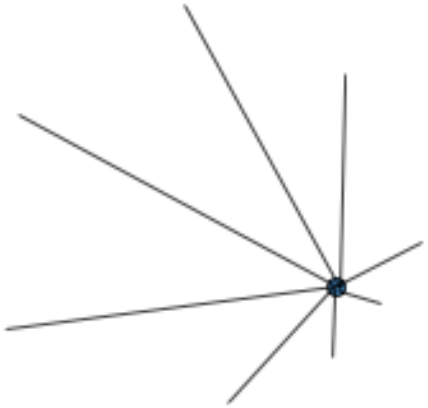
Results in fixed based priority arbitration

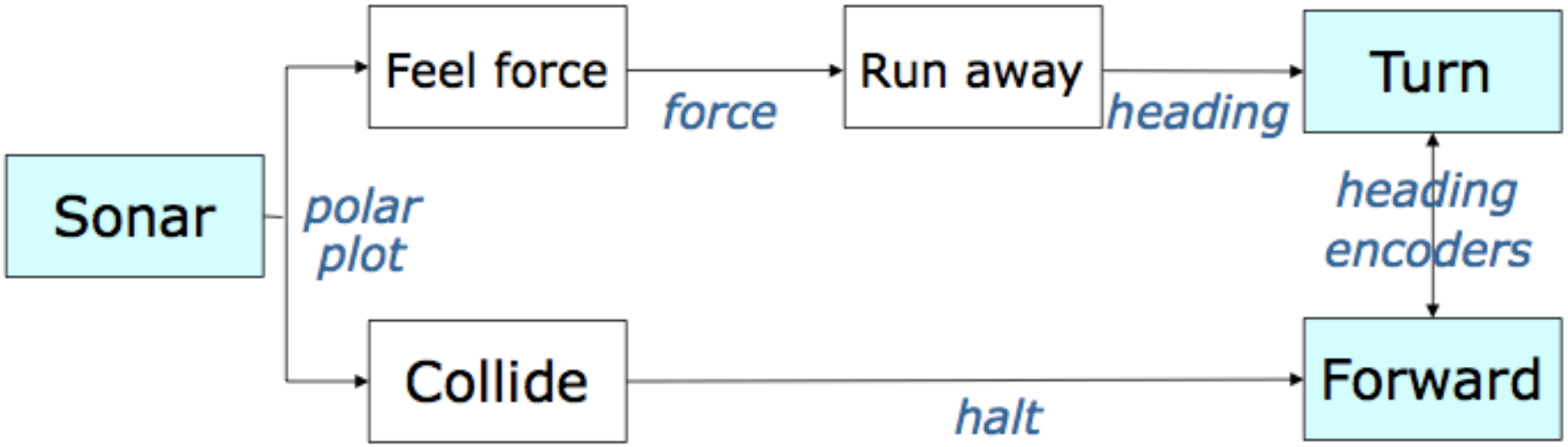
Coupling between the layers can be done through the world:
e.g. HERBERT soda can searching robot (Jon Connell, MIT)
1. If you see soda can grab it
2. If its heavy put it down
3. If its empty pick it up

- No notion of the model of the world or connections between behaviors

Level 1: Avoid

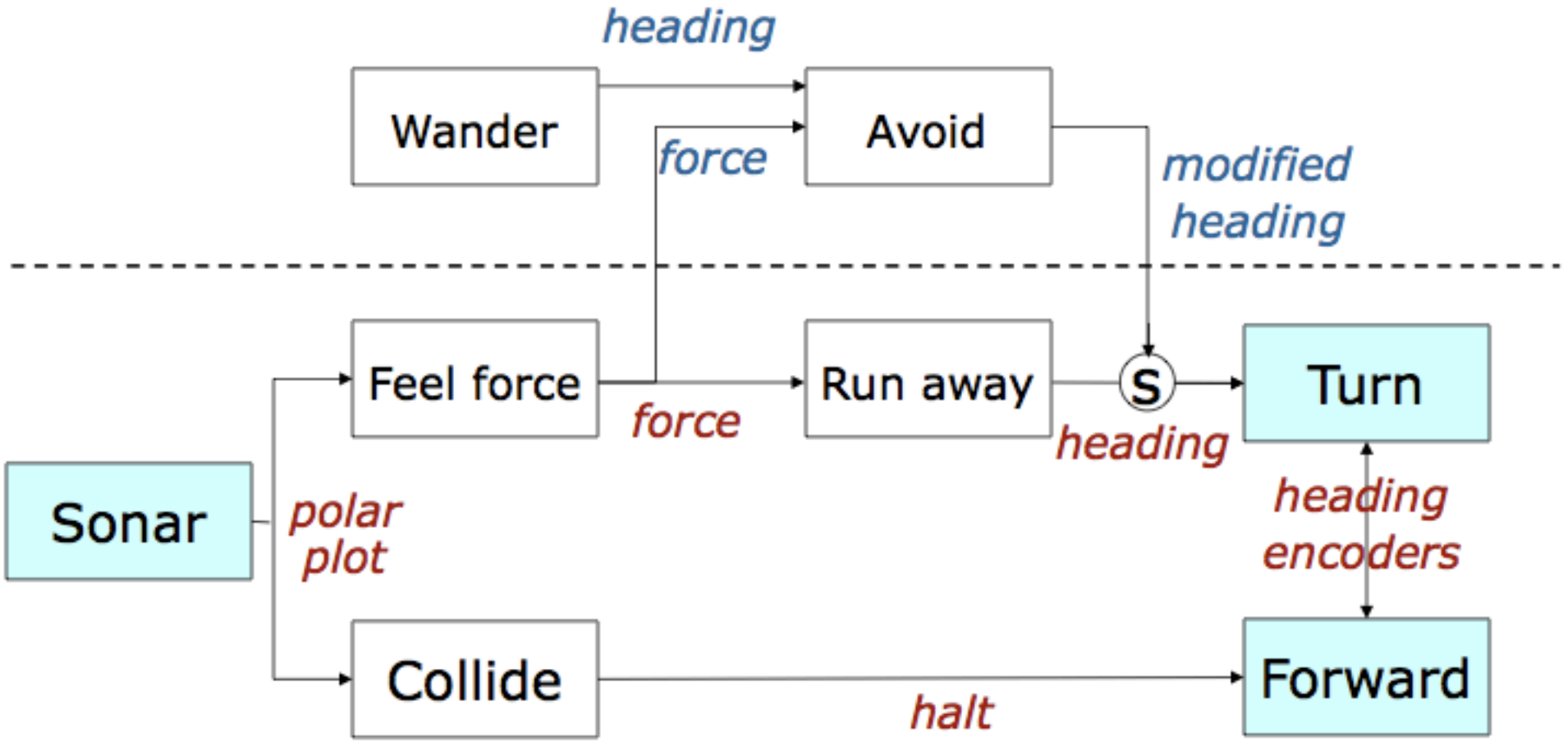
Polar plot of sonars





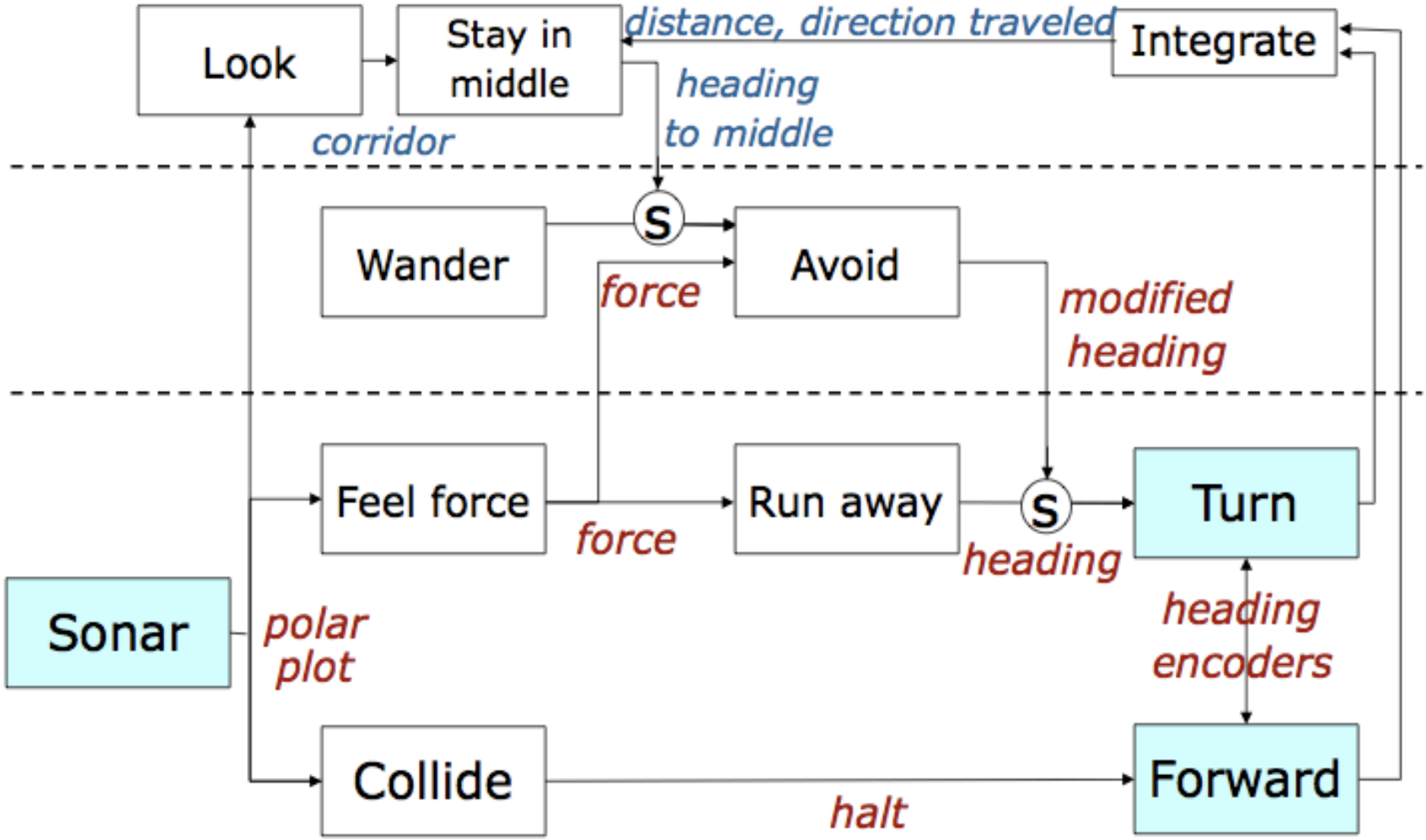
Slide coutresy: Uni Freiburg: Burgard, Stachniss, Bennewitz, Arras

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:

- 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

- Hypothesis at the time: world is its best model - “Intelligence” without representation

Issues of representation

- 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

Behavior-Based Architecture

- Previously examples of behaviors, point to point, pose-to-pose, target tracking, trajectory tracking

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

- Potential Field Techniques – addition of attractive and repulsive potential

- We can have more general compositions of elementary behaviors (control strategies)

Behaviors

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

Representation of behaviors

- Behavior is mapping from state *x* to control command *u*

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 or finite state machines)

Potential Field Representations

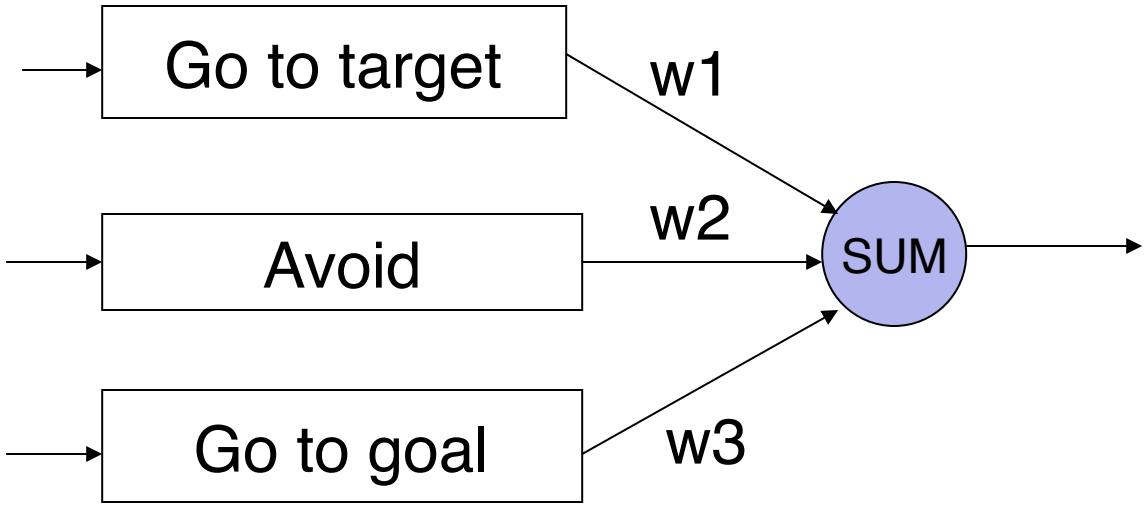
- Continuous representation
- Potential field techniques
- Attractive, repulsive potential fields
- Control Law - follow gradient of some potential function
- Gradient defines the vector field

- Behavior Schema's
- more general approach to vector fields design
- Go-to goal, follow corridor – Arbib 81

- Issues with superposition - local minima, maxima, oscillatory behavior

Superposition of different behaviors

- Each behavior will generate a vector of suggested heading
- Composite behaviors – linear combination of vectors



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
4. Modularity
5. Reuse of elementary behaviors
6. Coarser level of granularity – good for adaptation and learning
7. Abstraction's in terms of FSM's

Examples: FSM for navigation
Trash Collecting Robot

Behavior Composition

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

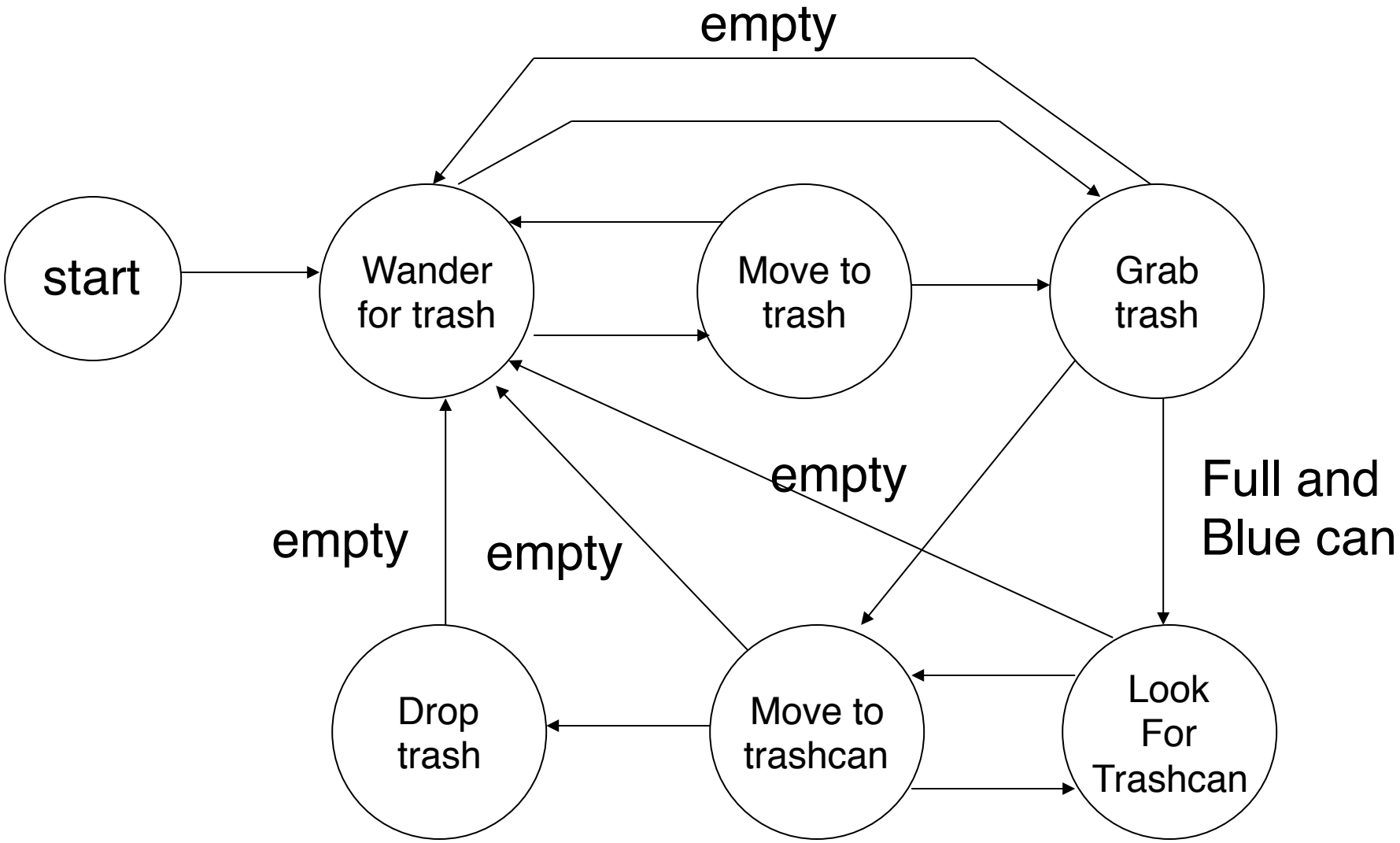
1. Sequential B1; B2
2. Conditional B1 : B2
3. Parallel B1 ll B2
4. Disabling B1 # B2
5. Iterative B1:; B2, B1 :: B2

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

Example : Classroom navigation , Clean Up, Foraging

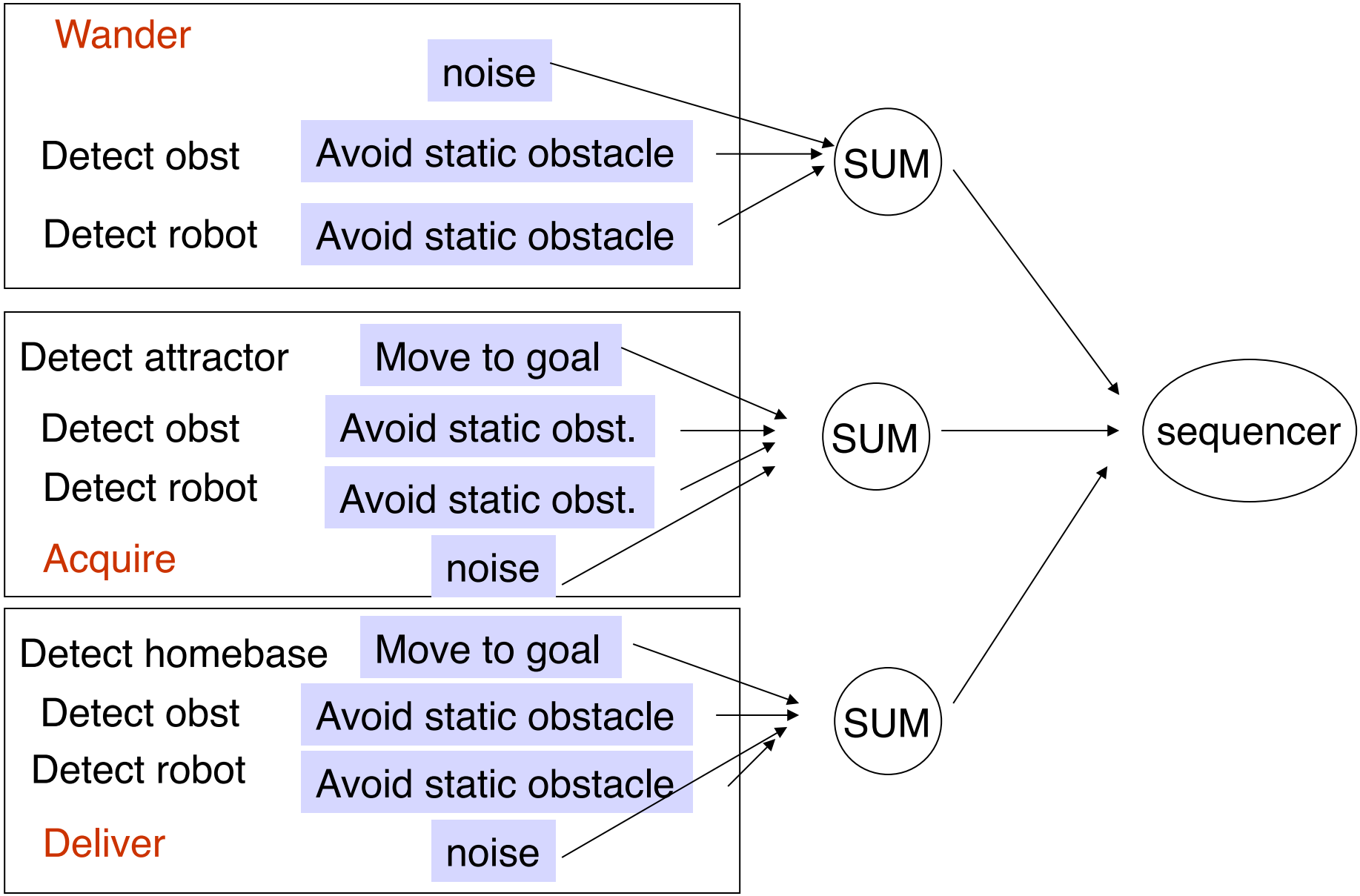
Different behaviors/motor schemas

- 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



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

Foraging Multi-Agent Behaviours



Emergent Behaviors

- New behaviors can “emerge” from

Interactions of rules
Interactions of behaviors
Interactions with the environment

- Behavior is just input output mapping observed externally
- 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's subjective notion

- Due to the un-modeled uncertainties, the behaviors are not exactly repeatable/predictable
- For the purpose of analysis – undesirable phenomena

Streering behaviors

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

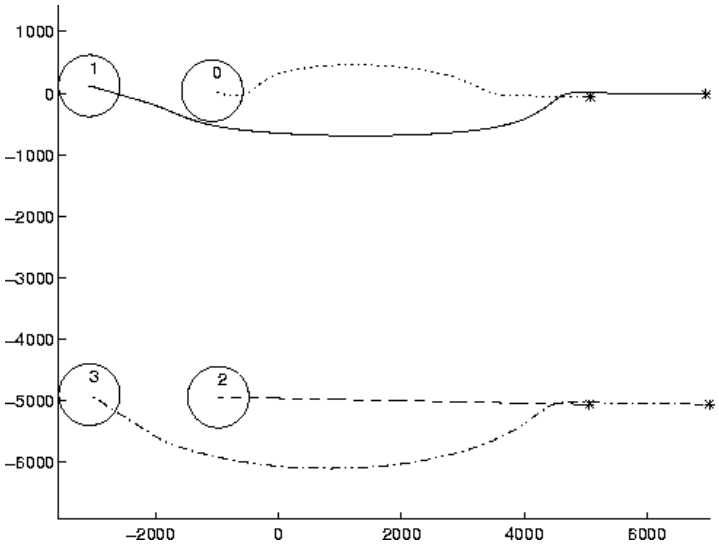
Example – Multi-robot Collision Avoidance

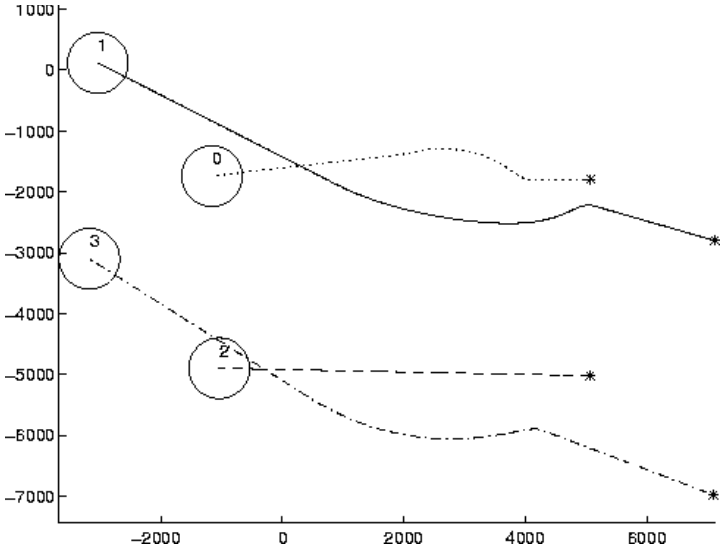
Air traffic Control – Collision Avoidance

- 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

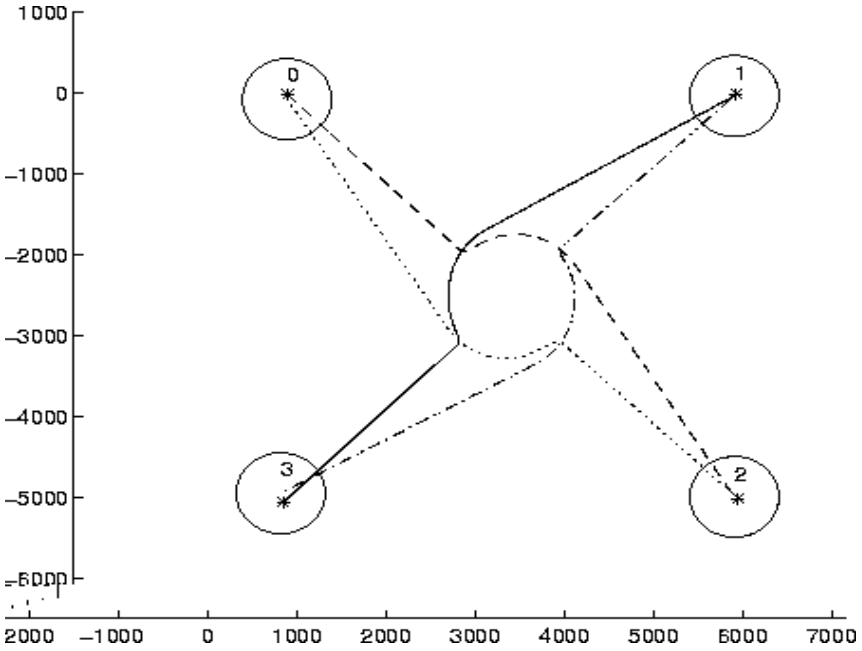
- Superposition of participating vector fields





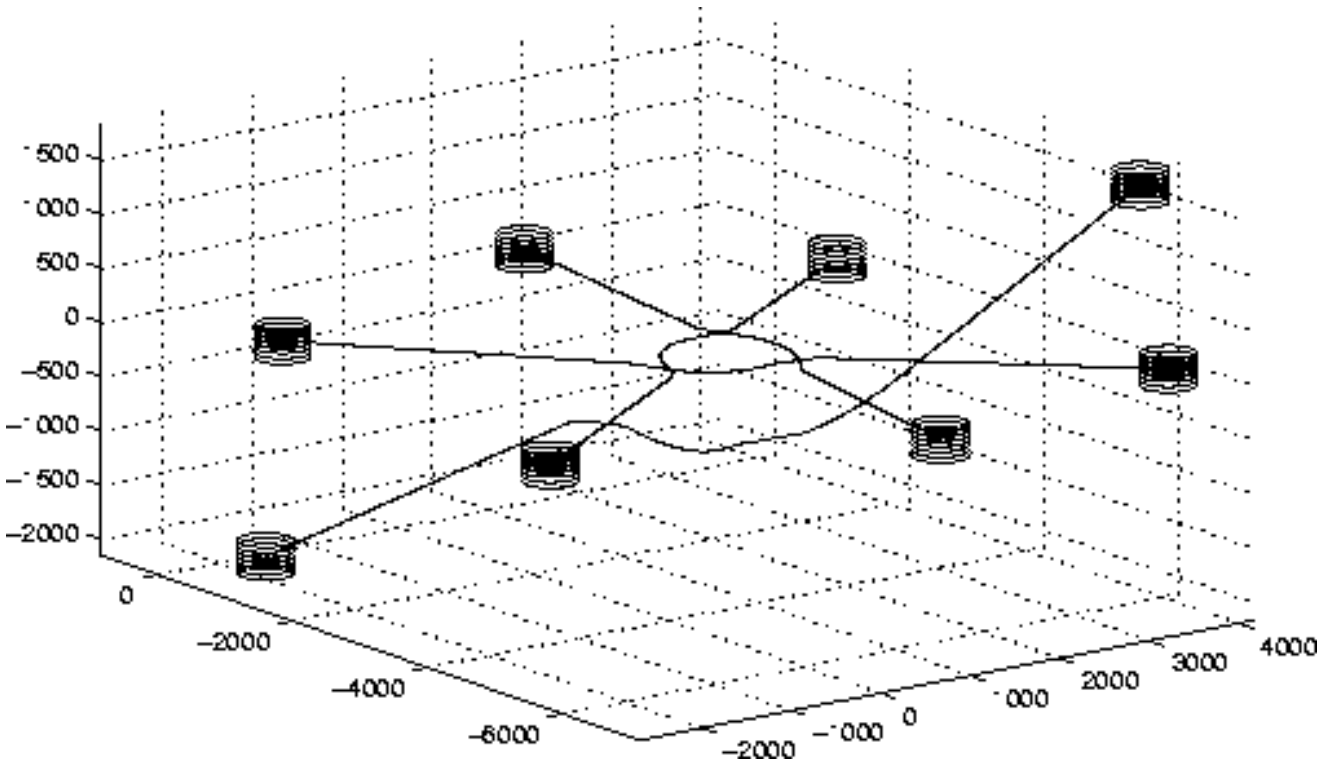
Overtake maneuver

Collision Avoidance – Prototype Maneuvers



Roundabout maneuver

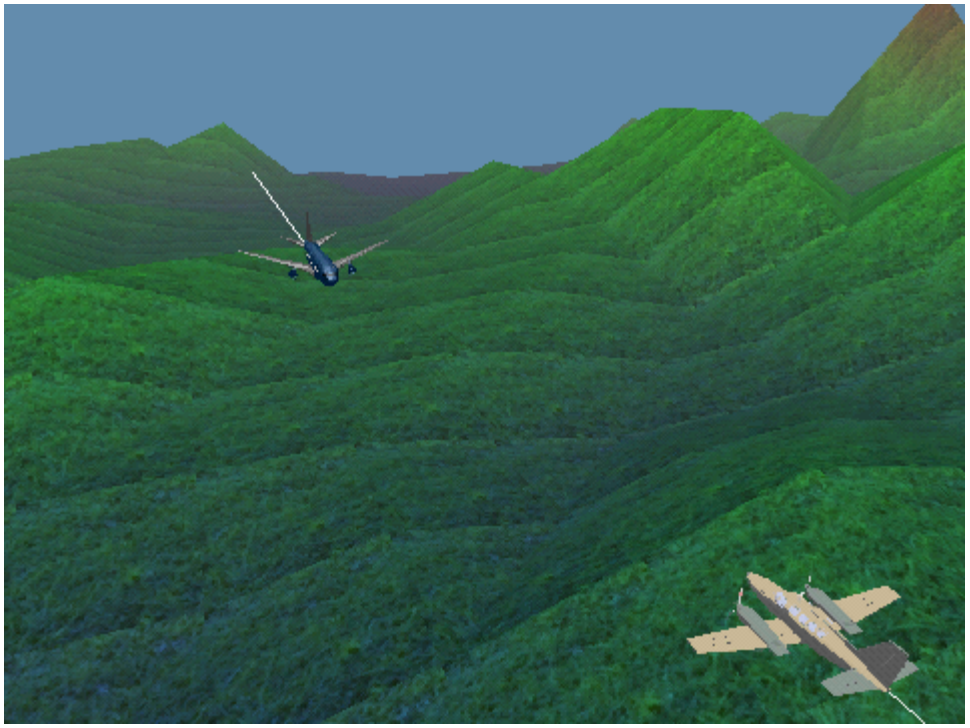
Collision Avoidance – Prototype Maneuvers



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

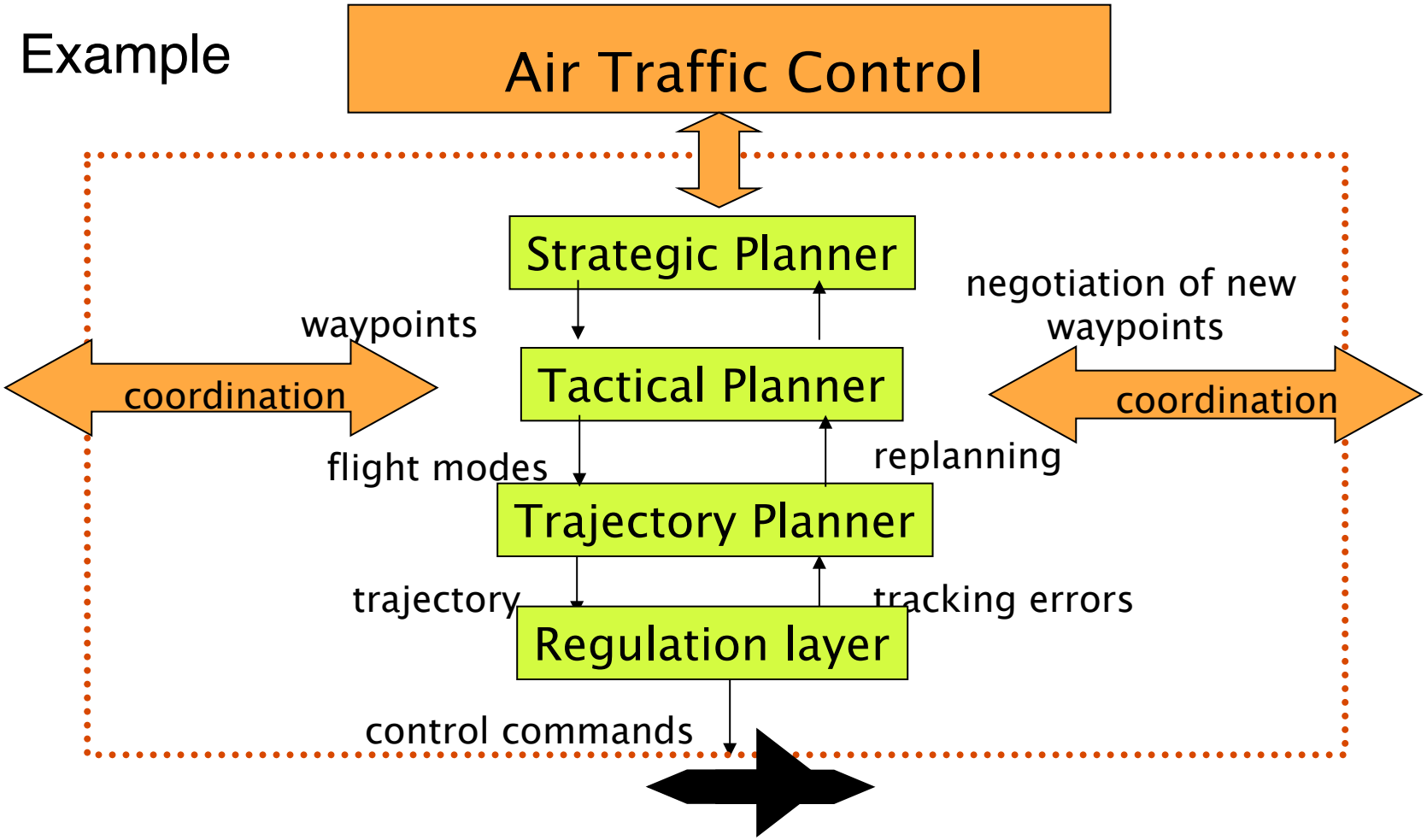
Collision Avoidance – **Visualization**

Collision Avoidance – **Visualization**



Hierarchical Architecture

Example



Design and modeling of a hierarchical hybrid systems:

Safety guarantees, Safety vs performance tradeoff