

# Decision Making and FSMs

EECS 398

Intro. to Autonomous Robotics

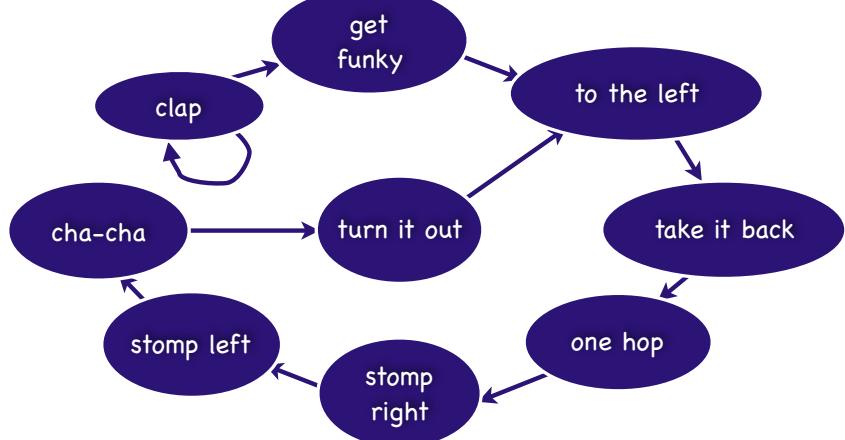
ME/EECS 567    ROB 510  
Robot Modeling and Control

Fall 2018

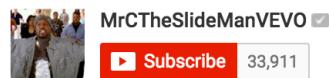


[autorob.org](http://autorob.org)

# Cha-Cha Slide



Mr. C The Slide Man - Cha Cha Slide Part 2

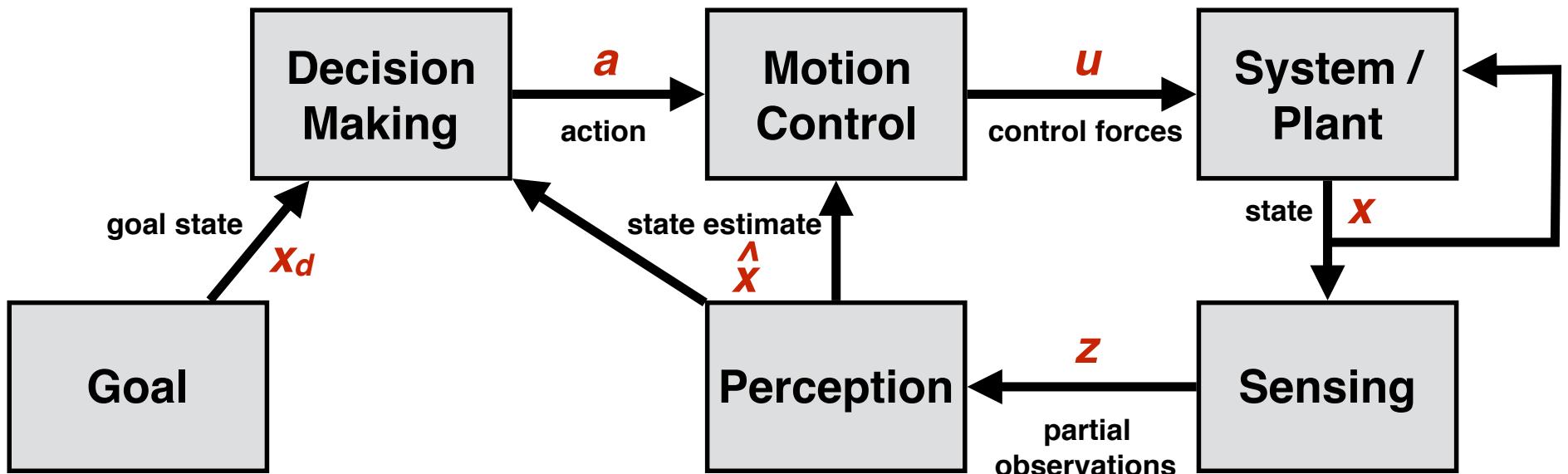


2,413,250

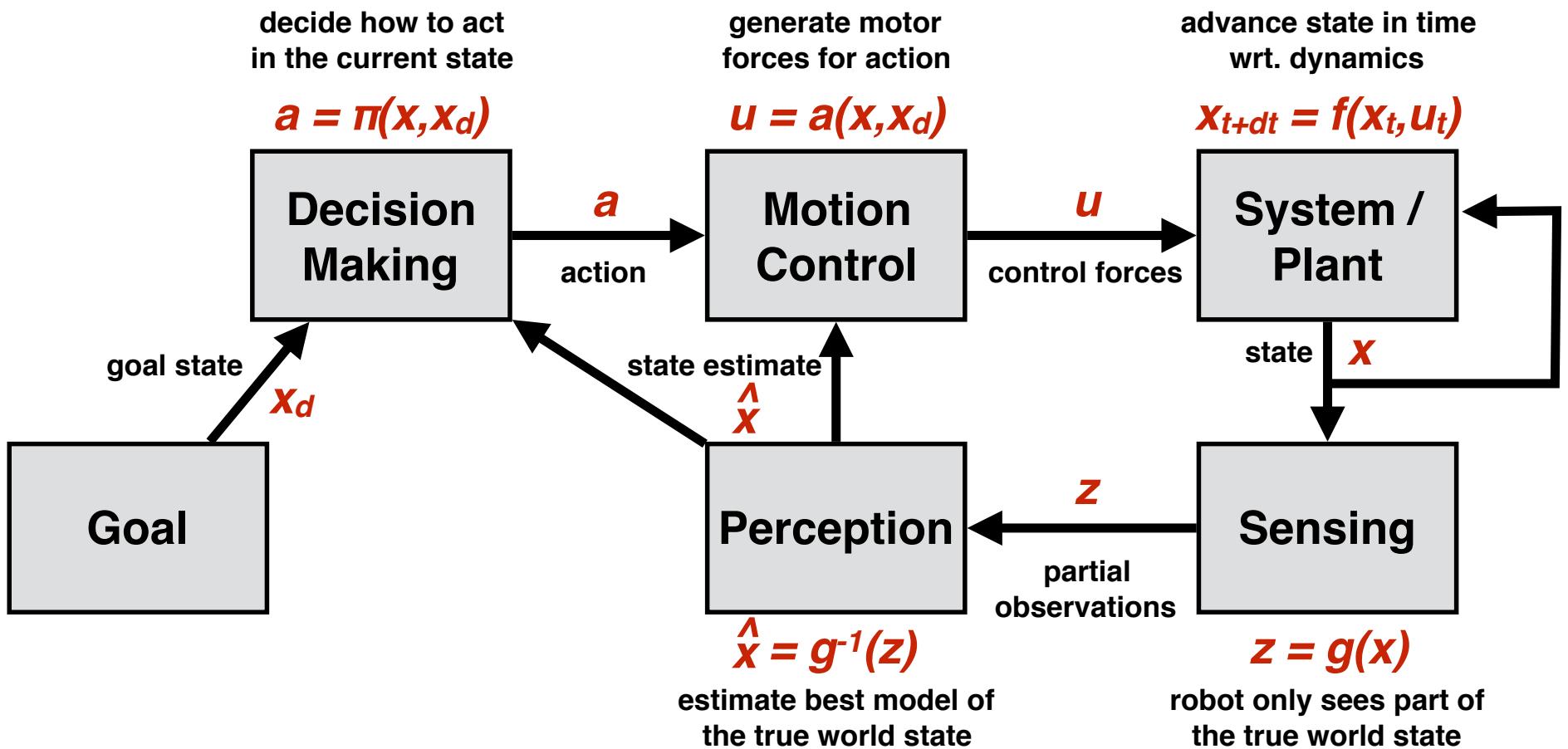
Video: <https://www.youtube.com/watch?v=LI64R1bjN7U>

Lyrics: <https://youtu.be/cb6pJ4AE0oI>

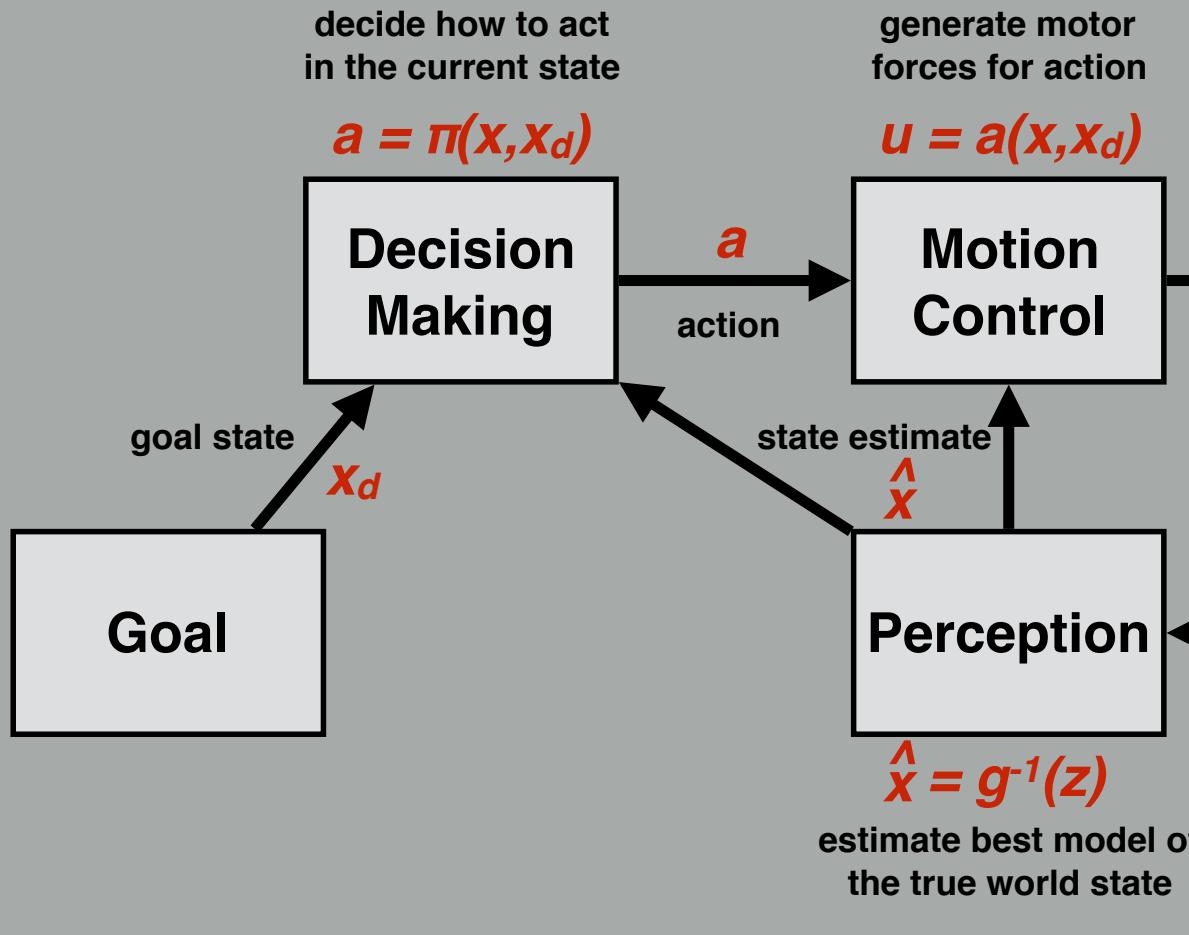
# Robot Control Loop



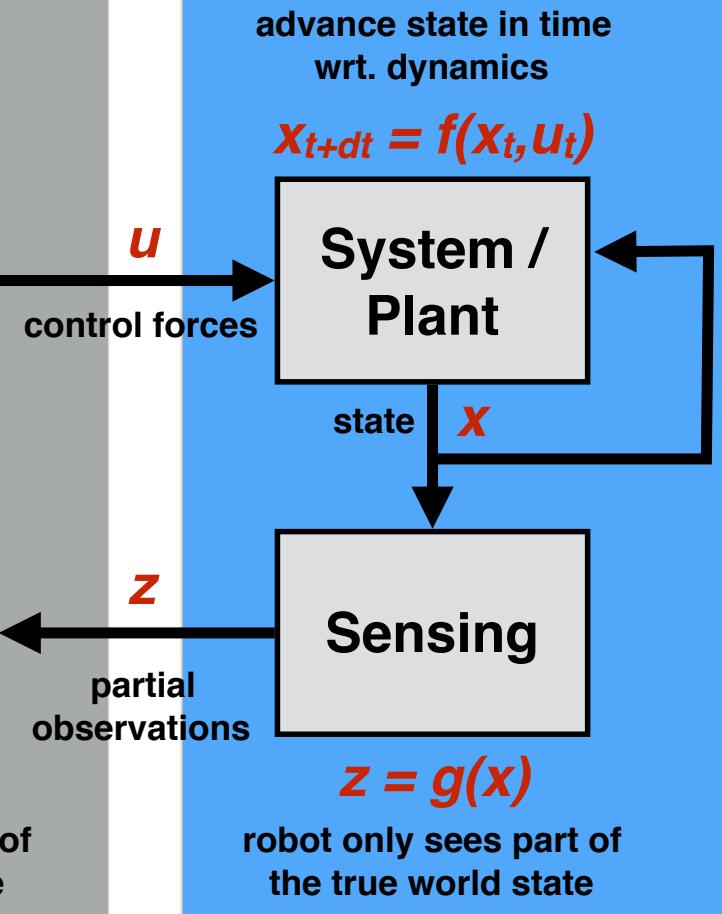
# Robot Control Loop



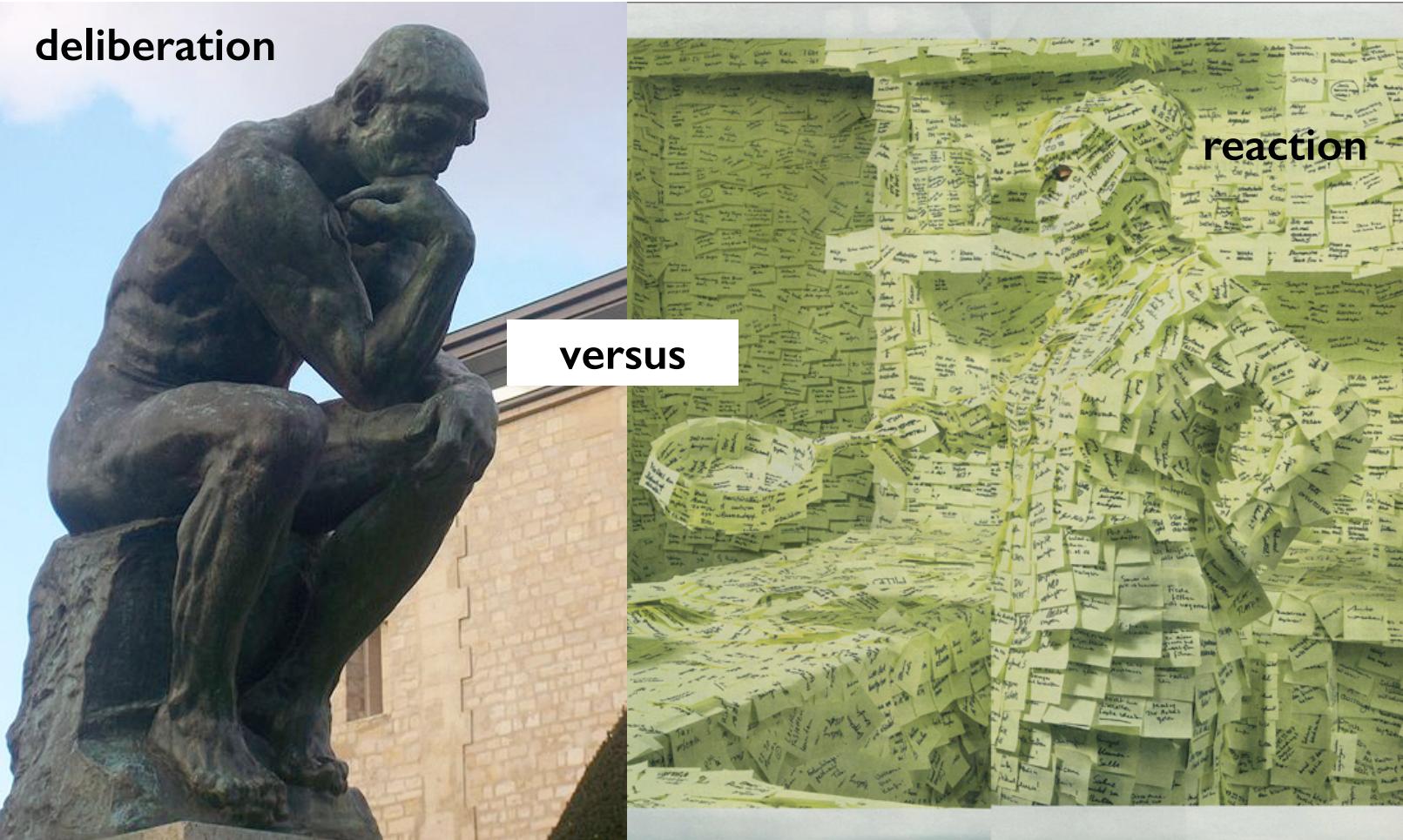
# Autonomy



# Embodiment



# Robot Decision Making



# Should your robot's decision making



OR

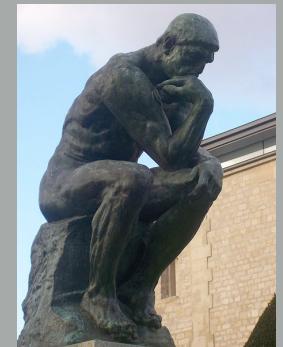
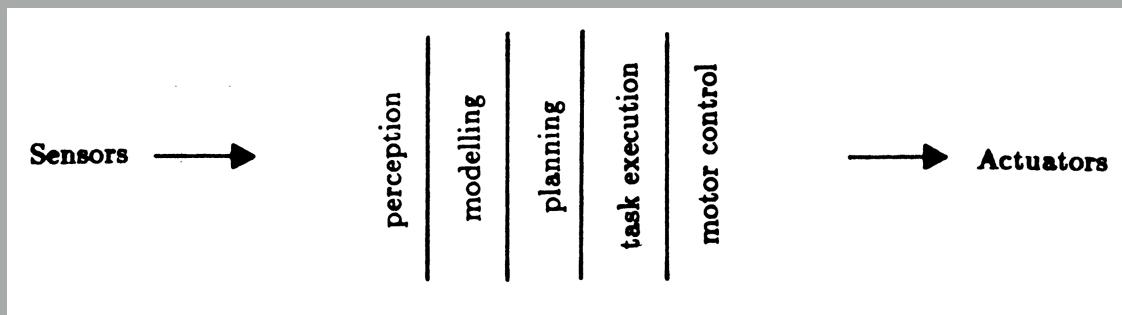


fully think through  
solving a problem?

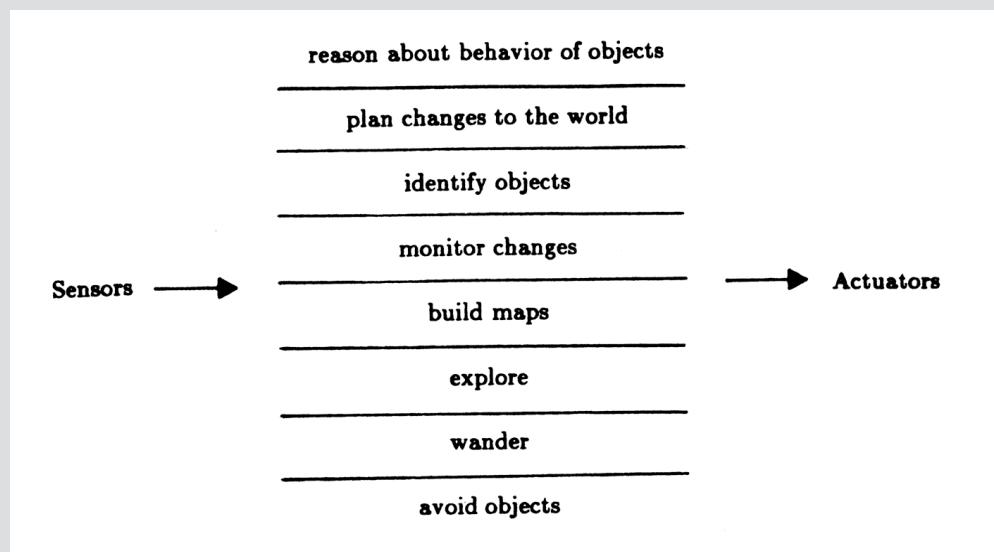
react quickly to  
changes in its world?

# Deliberation v. Reaction

**deliberative:**  
sense-plan-act,  
path planning  
motion planning



**reaction:**  
controllers acting in parallel  
subsumption,  
Finite State Machine



[Arkin 1998]

# Deliberation-Reaction spectrum



## DELIBERATIVE

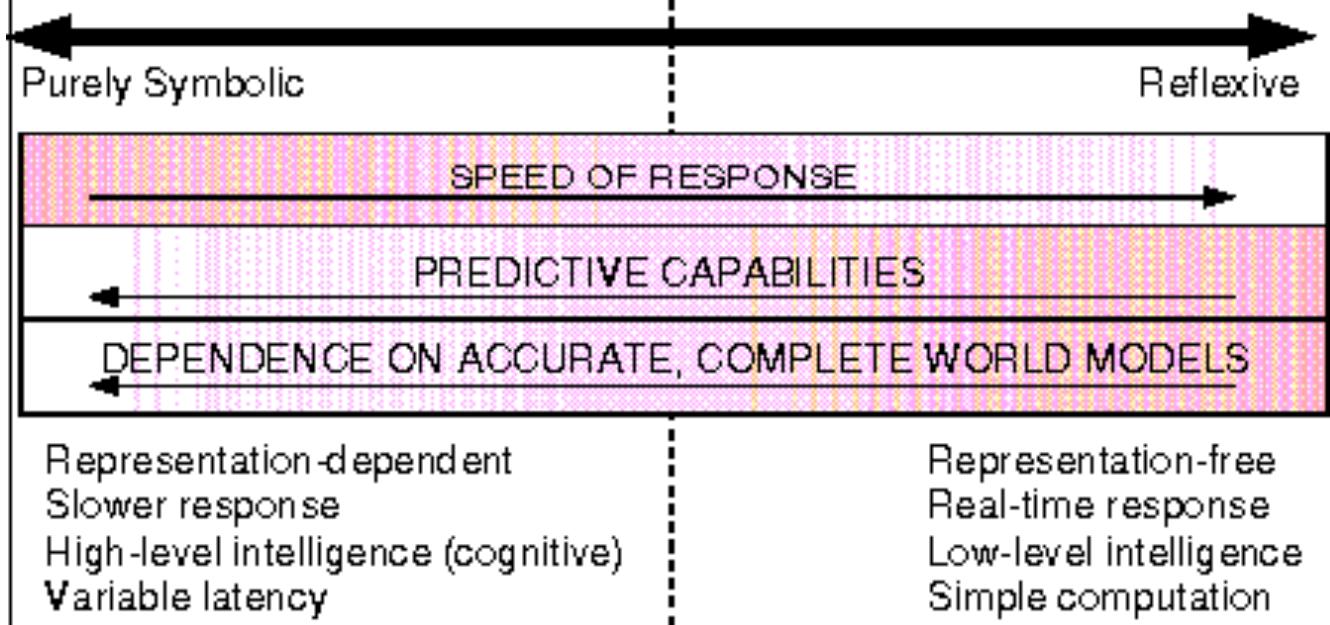
## REACTIVE

Complete  
Adaptive  
Optimal  
Slower

Faster  
Cheaper  
More robust  
Forgetful

Requires  
complete model  
of the world

Requires a  
complete design  
of the problem



Considerations: time-scale/tractability, generality, representation/state

# Examples?



DELIBERATIVE



REACTIVE

Purely Symbolic

Reflexive

example???

# Examples?



**DELIBERATIVE**



**REACTIVE**

Purely Symbolic

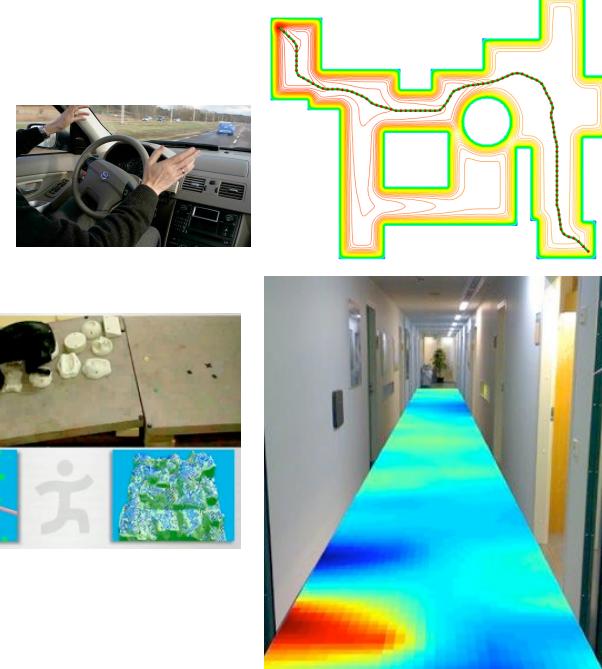
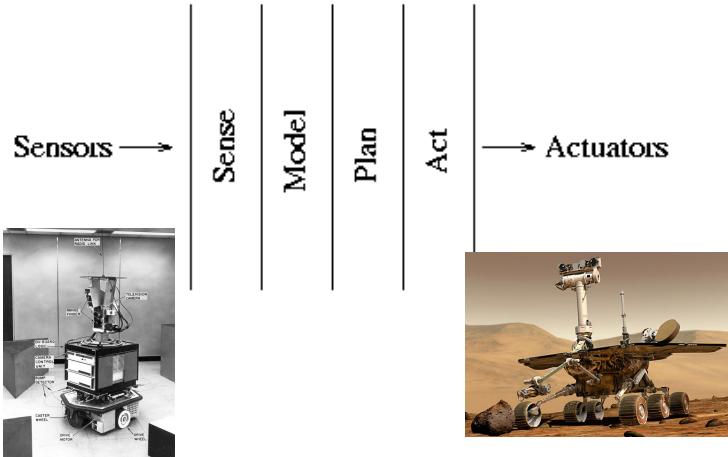
Reflexive



# Deliberation

## “Sense-Plan-Act” paradigm

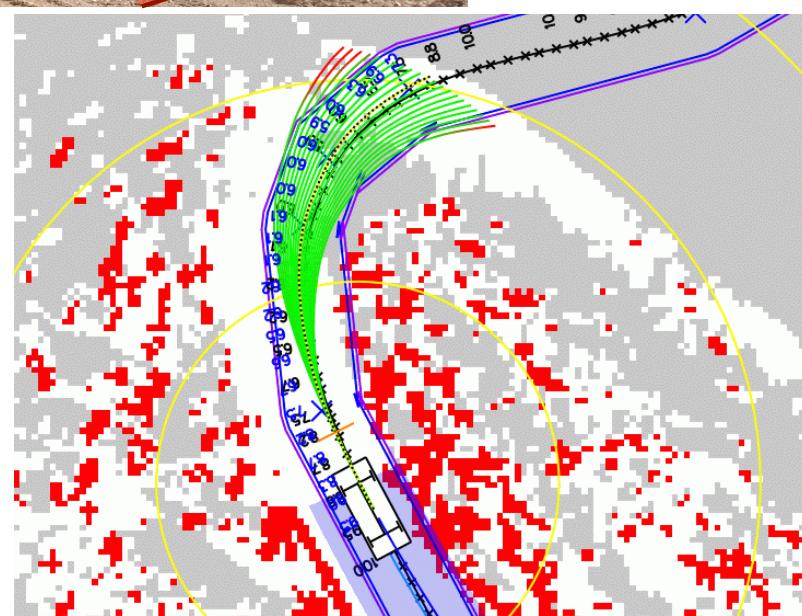
- sense: build most complete model of world
  - GPS, SLAM, 3D reconstruction, affordances
- plan: search over all possible outcomes
  - BFS, DFS, Dijkstra, A\*, RRT
- act: execute plan through motor forces



# Stanley (Grand Challenge)



Navigation  
costmap



Road detection

2005

# MIT Talos (Urban Challenge)



2007



2013

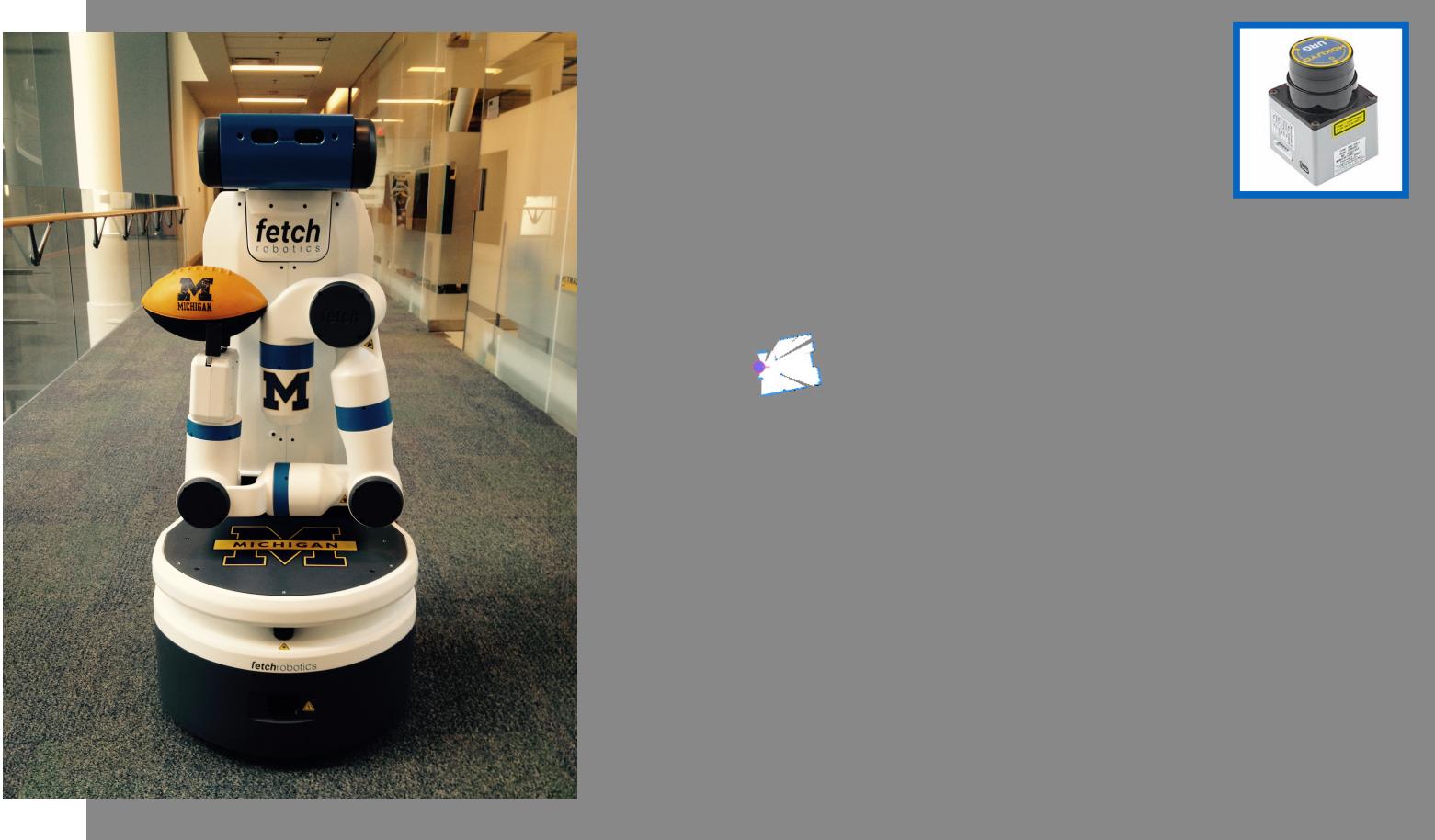
Deliberation  
requires a model of the world



Color+Depth Camera



Laser Rangefinder

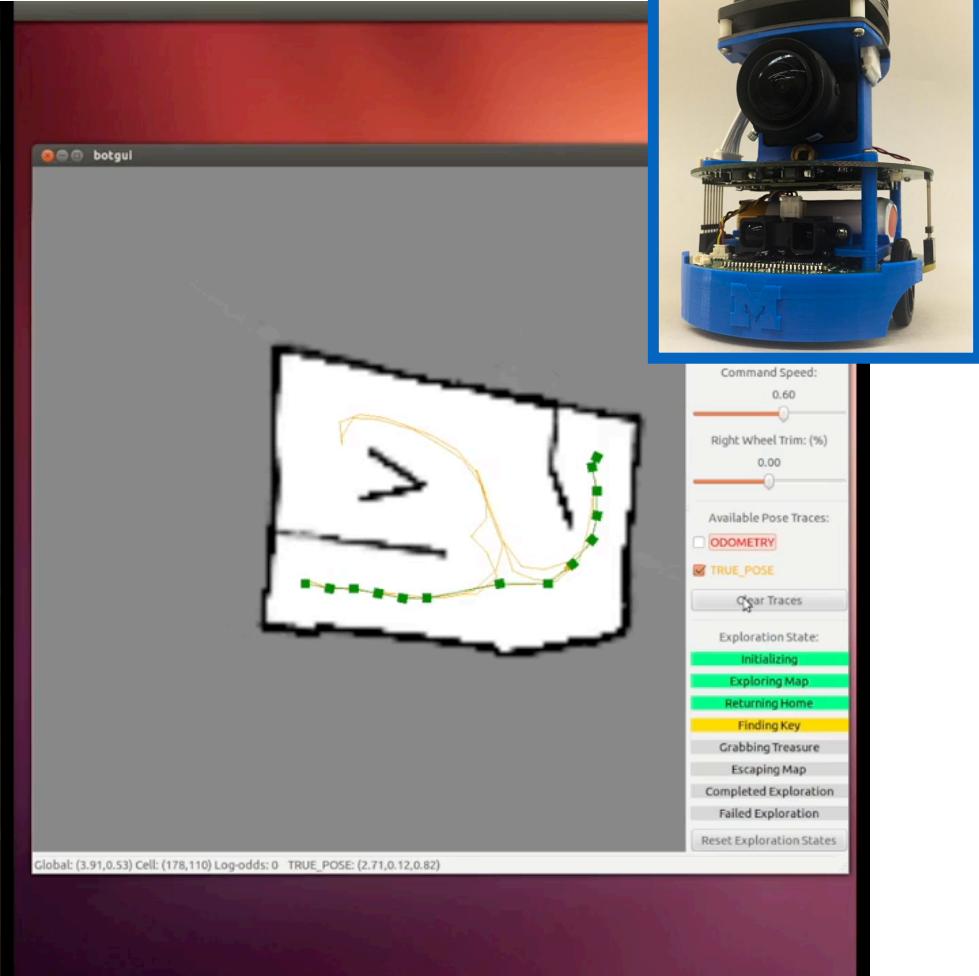
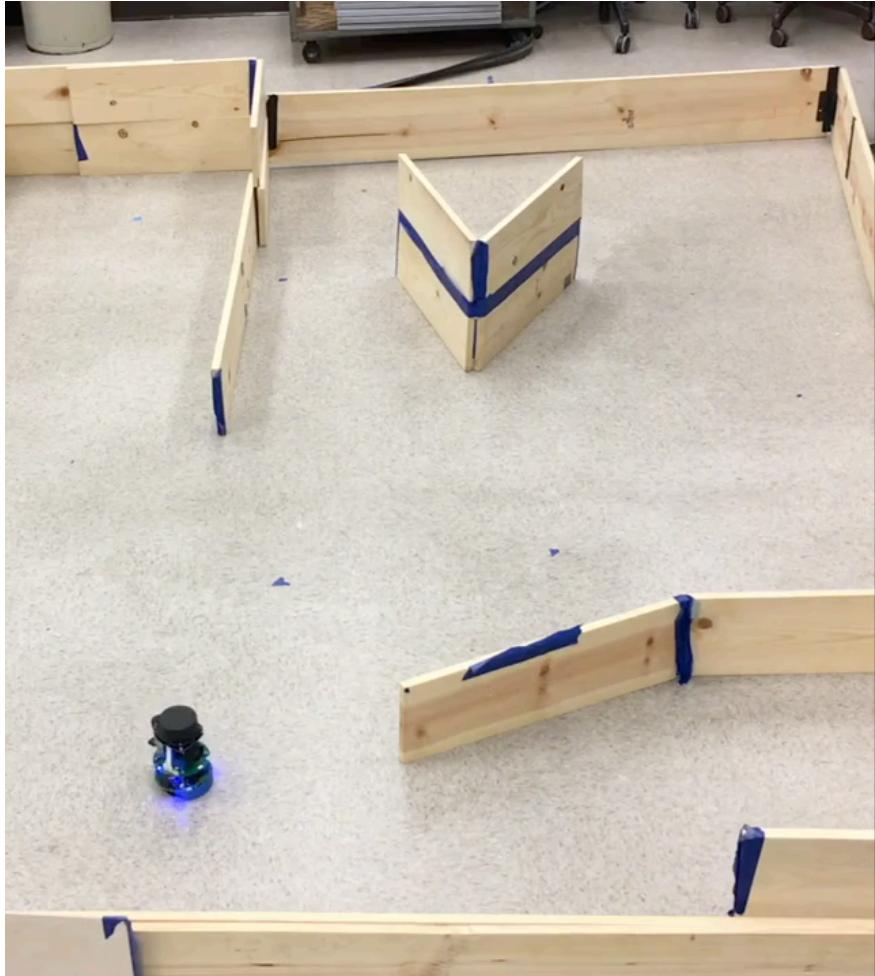


# Simultaneous Localization and Mapping



Autonomous robot navigation  
from previously built map

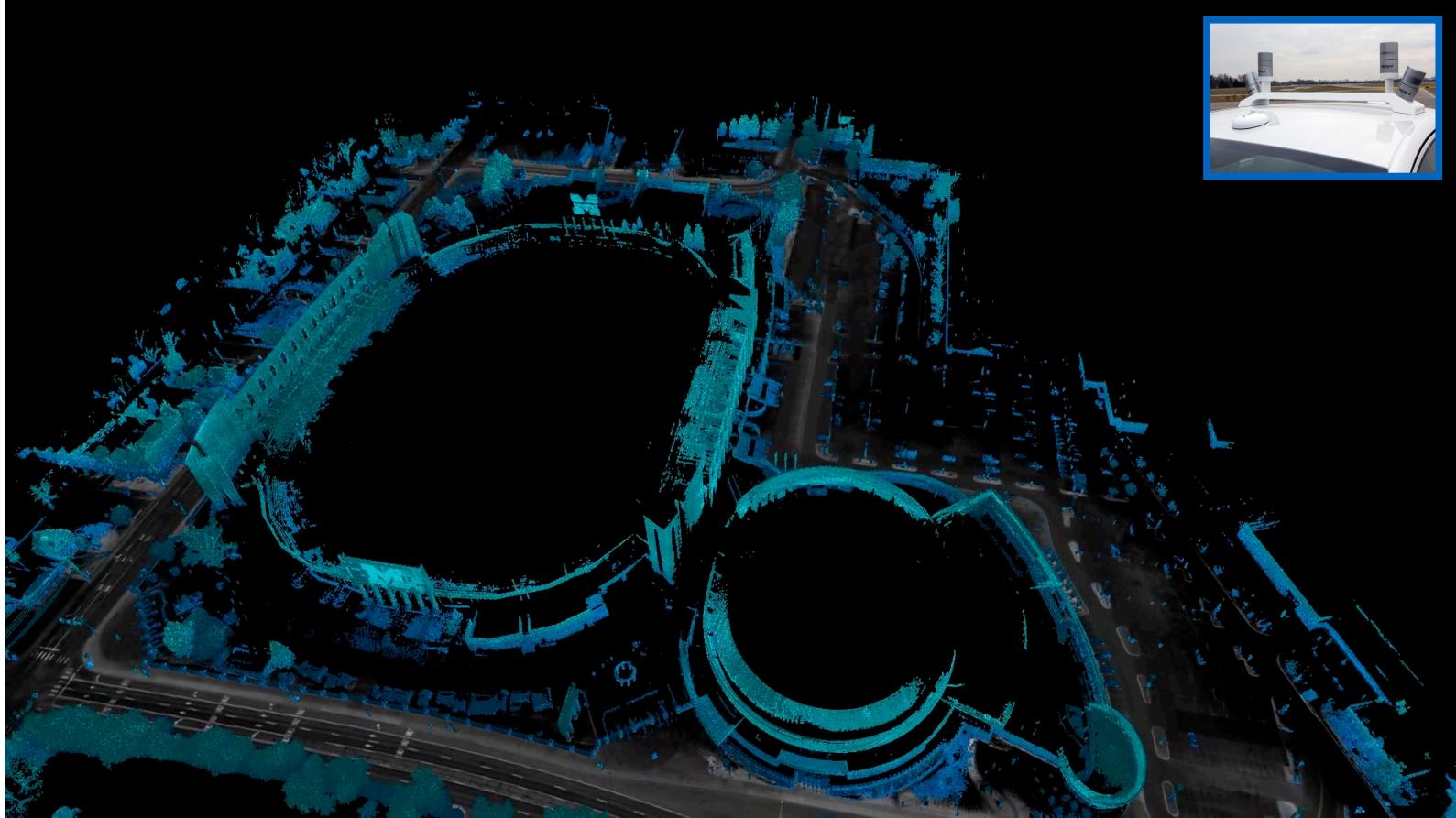
# ROB 550 BotLab / EECS 467 Escape Challenge



UM EECS 467 [autobot.github.io](https://autobot.github.io) Jasmine Liu et al.



Michigan Next Generation Vehicle (Eustice, Olson et al.)



# Autonomous Transportation

Michigan Next Generation Vehicle (Eustice, Olson et al.)

# Examples?



**DELIBERATIVE**

Purely Symbolic



**REACTIVE**

Reflexive



example???

# Examples?



**DELIBERATIVE**

Purely Symbolic

**REACTIVE**



Reflexive



<https://www.youtube.com/watch?v=jCB3pd-wBw0> Autonomous cars again?



Follow

Sign in

Get started



Edwin Olson [Follow](#)

CEO of May Mobility, Associate Professor of Computer Science at University of Michigan

Sep 4 · 5 min read



SELF-DRIVING CARS SHOULD BE **GOOD** DRIVERS,  
NOT **SO-SO** DRIVERS. BUT ARE THEY?

## MPDM: Multipolicy Decision-Making in Dynamic, Uncertain Environments for Autonomous Driving

Alexander G. Cunningham, Enric Galceran, Ryan M. Eustice, and Edwin Olson

**Abstract**—Real-world autonomous driving in city traffic must cope with dynamic environments including other agents with uncertain intentions. This poses a challenging decision-making problem, e.g., deciding when to perform a passing maneuver or how to safely merge into traffic. Previous work in the literature has typically approached the problem using ad-hoc solutions that do not consider the possible future states of other agents, and thus have difficulty scaling to complex traffic scenarios where the actions of participating agents are tightly conditioned on one another. In this paper we present multipolicy decision-making (MPDM), a decision-making algorithm that exploits knowledge from the autonomous driving domain to make decisions online for an autonomous vehicle navigating in traffic. By assuming the controlled vehicle and other traffic participants execute a policy from a set of plausible closed-loop policies at every timestep, the algorithm selects the best available policy for the controlled vehicle to execute. We perform policy election using forward simulation of both the controlled vehicle and other agents, efficiently sampling from the high-lielihood outcomes of their interactions. We then score

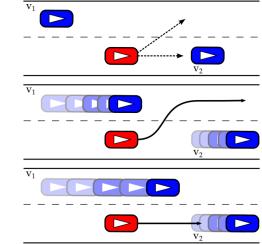


Fig. 1. In the top image, the red car is faced with a discrete choice as to whether change lanes to pass vehicle  $v_2$  in front of vehicle  $v_1$  (middle) or remain behind slow vehicle  $v_2$  (bottom). The MPDM algorithm evaluates these possibilities, while simulating forward the other vehicles, shown in the center and bottom images. By considering the *closed-loop interactions*

<https://medium.com/may-mobility/the-problem-with-so-so-driving-4a46f60bac37>



# Autonomous Navigation in Dynamic Social Environments using Multi-Policy Decision Making

Dhanvin Mehta

Gonzalo Ferrer

Edwin Olson

IROS 2016

# Examples?



DELIBERATIVE

Purely Symbolic



REACTIVE

Reflexive



more common  
example???

# Examples?



**DELIBERATIVE**

Purely Symbolic

**REACTIVE**

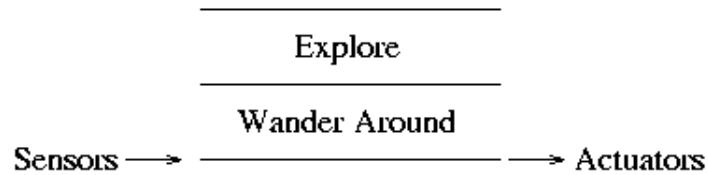


Reflexive



# Reaction

- No representation of state
  - Typically, fast hardcoded rules
- Embodied intelligence
  - behavior ← control + embodiment
  - Stigmergy: ant analogy
- Finite State Machines
  - most common
- Subsumption architecture
  - prioritized reactive policies



# Roomba cleaning pattern



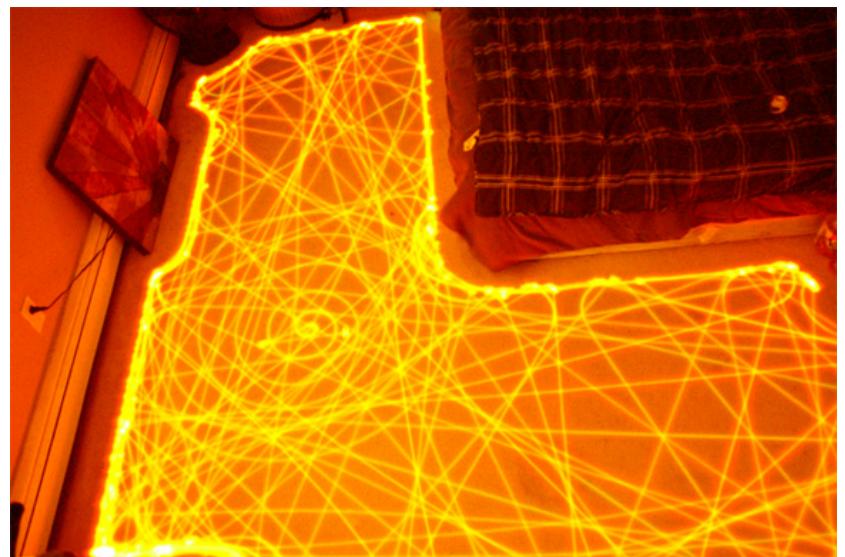
roomba cleaning "pattern"



miro ledajaks

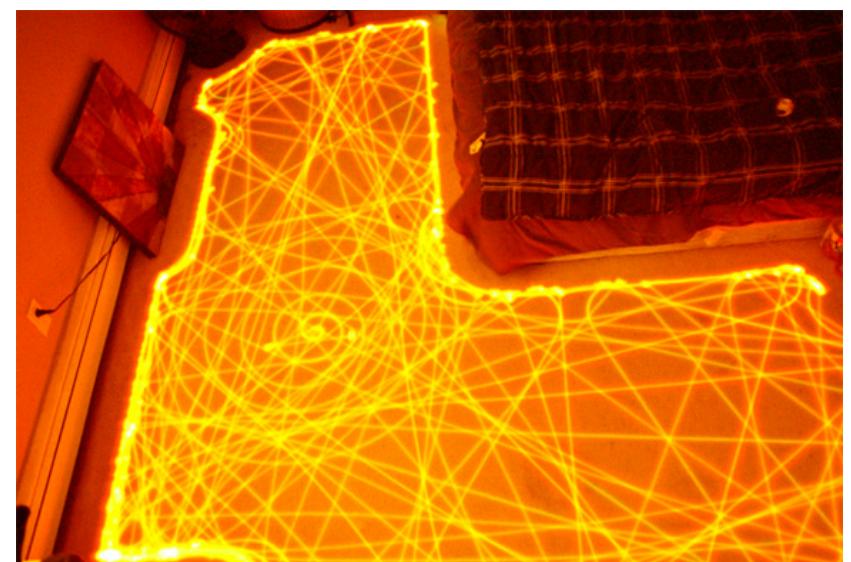
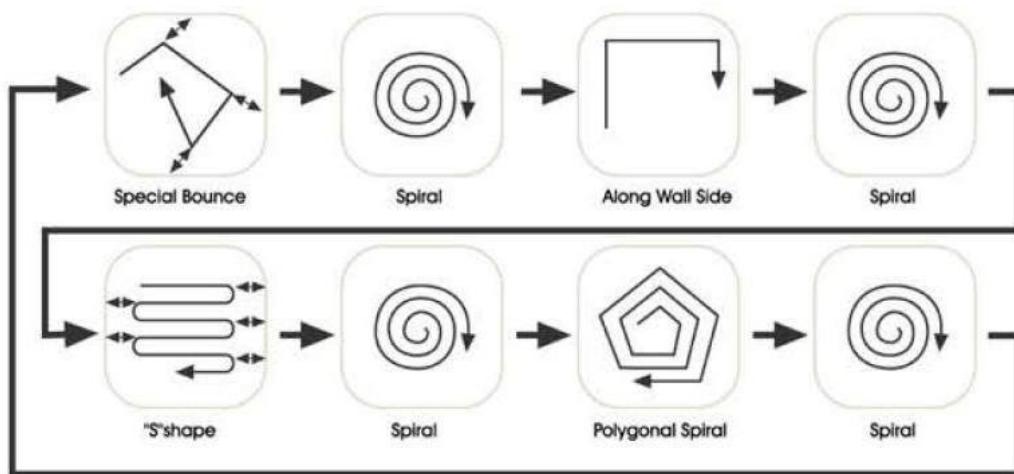
[Subscribe](#) 21

196 views



<https://www.youtube.com/watch?v=G4ocrevf4ng>

# Vacuuming Finite State Machine



# MIT Technology Review



Wealth & Investment  
Management

See how realigning  
can help maximiz

Ghengis hexapod robot

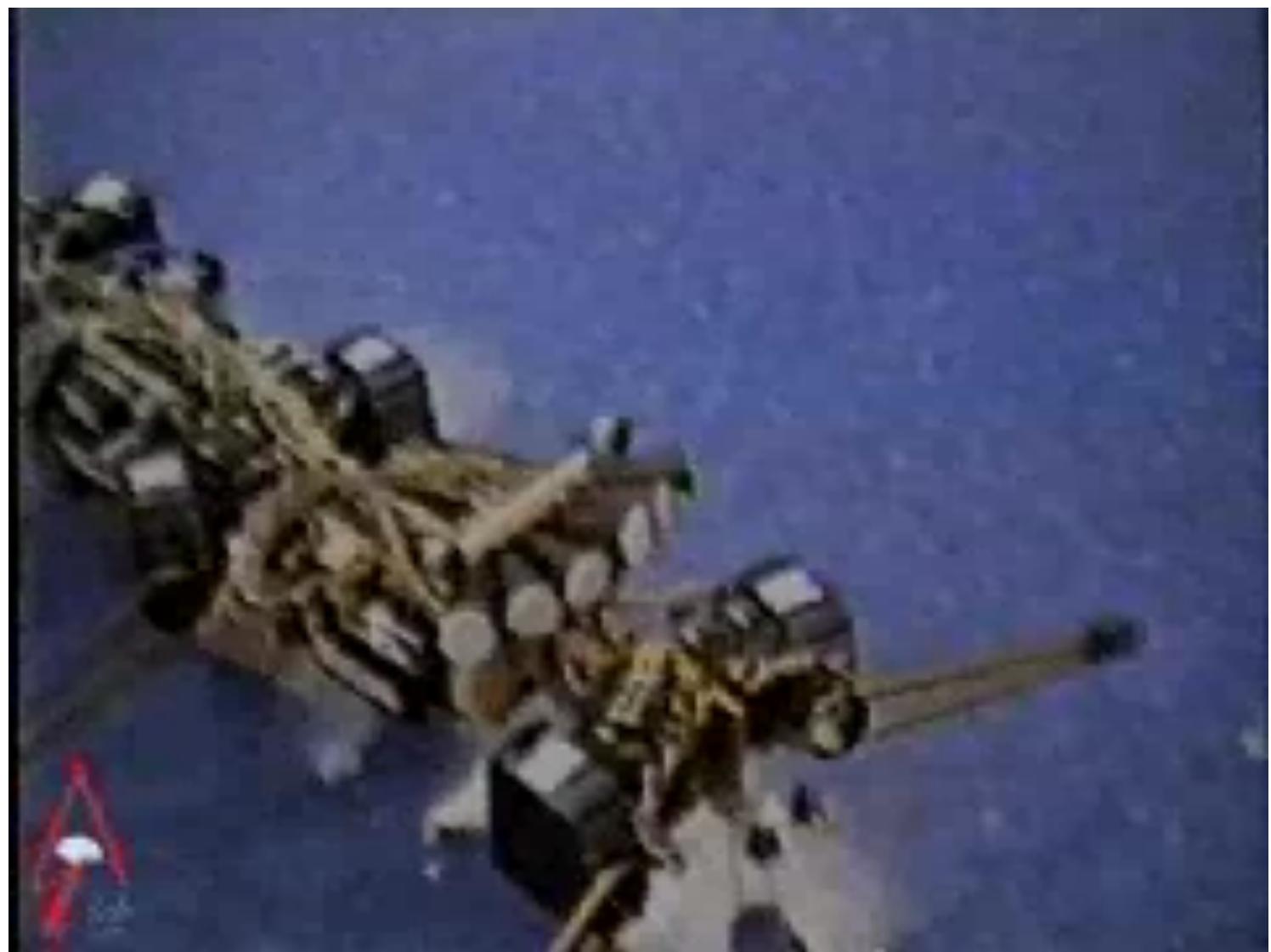


## Intelligent Machines

# Honey, They've Shrunk the Rover

The Sojourner vehicle that trekked across the Martian surface captured the world's fancy last summer, but we haven't seen anything yet. Next will come rovers that can roam miles across Mars and "aerobots" able to survey other planets.

by Eric Scigliano    January 1, 1998

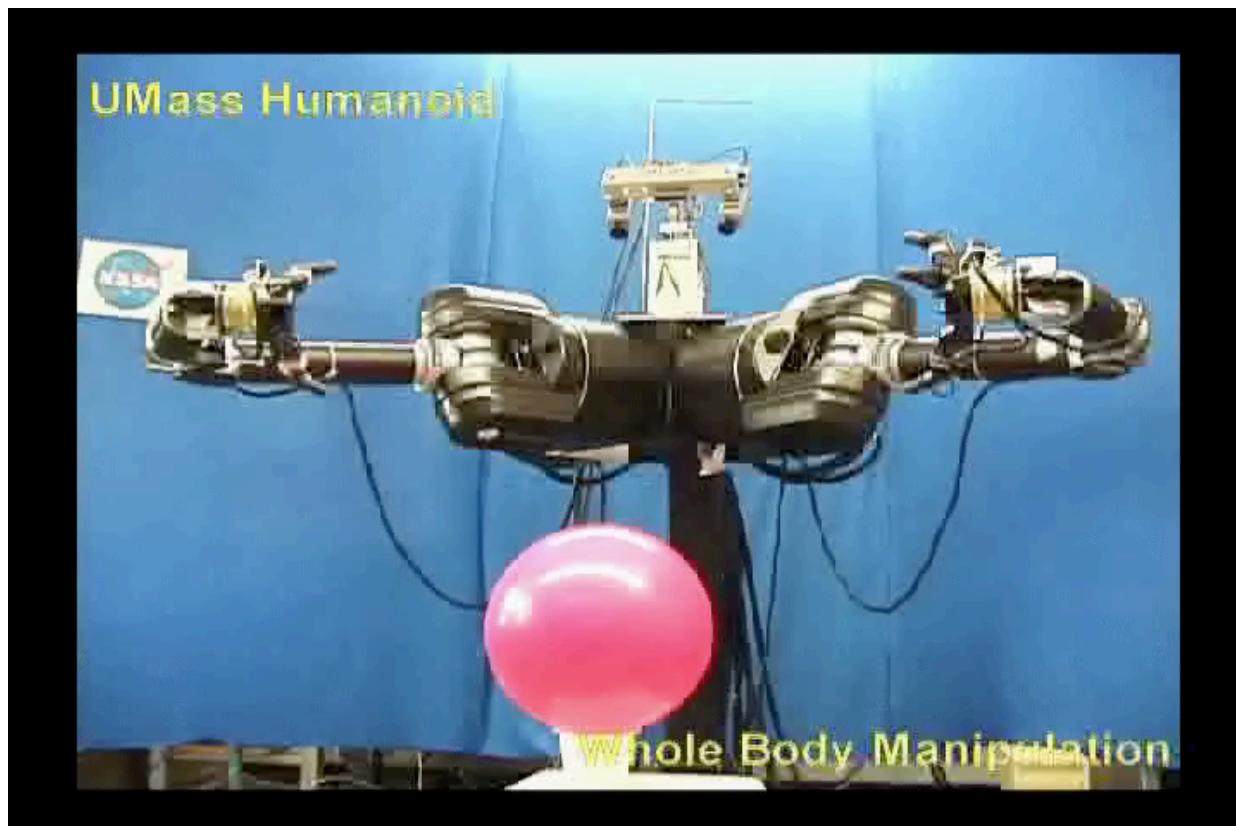


MIT Genghis

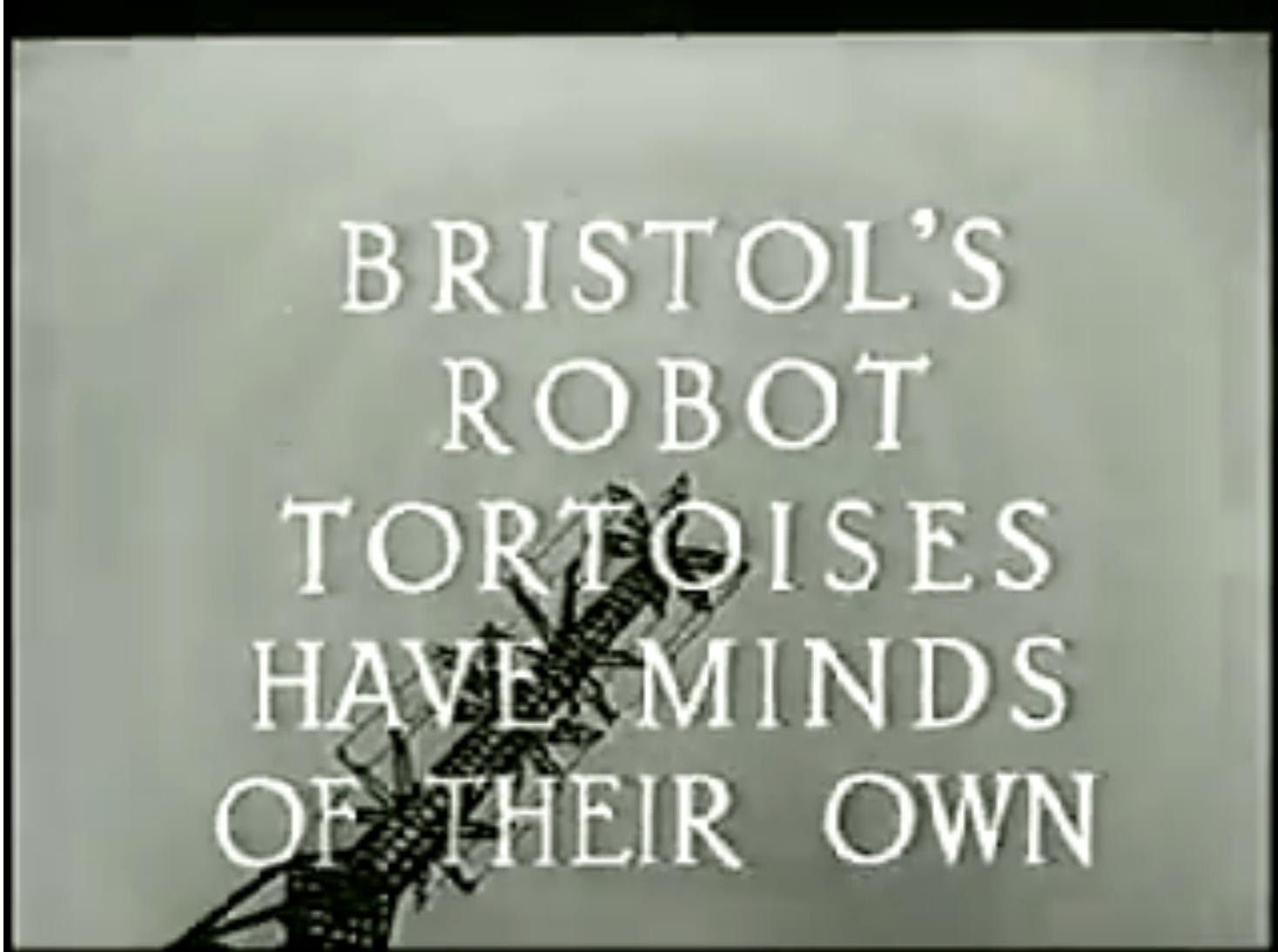


MIT Genghis

# Manipulation Gaits



Collections of robust  
manipulation controllers

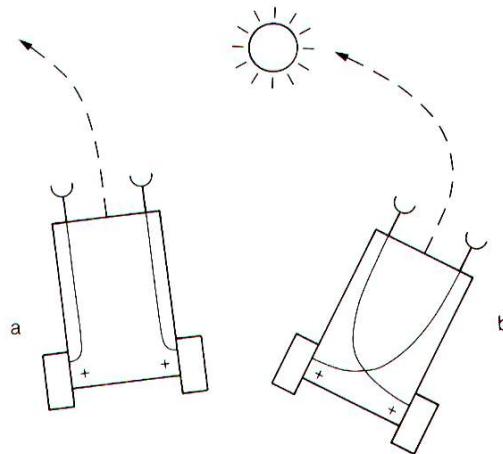


BRISTOL'S  
ROBOT  
TORTOISES  
HAVE MINDS  
OF THEIR OWN

Grey Walter's Tortoise (1949) - <https://youtu.be/lLULRlmXkKo>

# Embodied Intelligence

- Do we need to model the world or build sophisticated controllers?
- Embodied intelligence: robot behavior results from embodiment acting in environment
- Braitenberg vehicle
  - “Vehicles: Experiments in synthetic psychology” (1984)



Tiny Braitenberg Vehicle  
<http://vimeo.com/5664333>

**Figure 3**  
Vehicles 2a and 2b in the vicinity of a source (circle with rays emanating from it). Vehicle 2b orients toward the source, 2a away from it.

# Random Walk: Goal Seeking

- Move in a random direction until you hit something
- Then go in a new direction
- Stop when you get to the goal, assuming it can be recognized

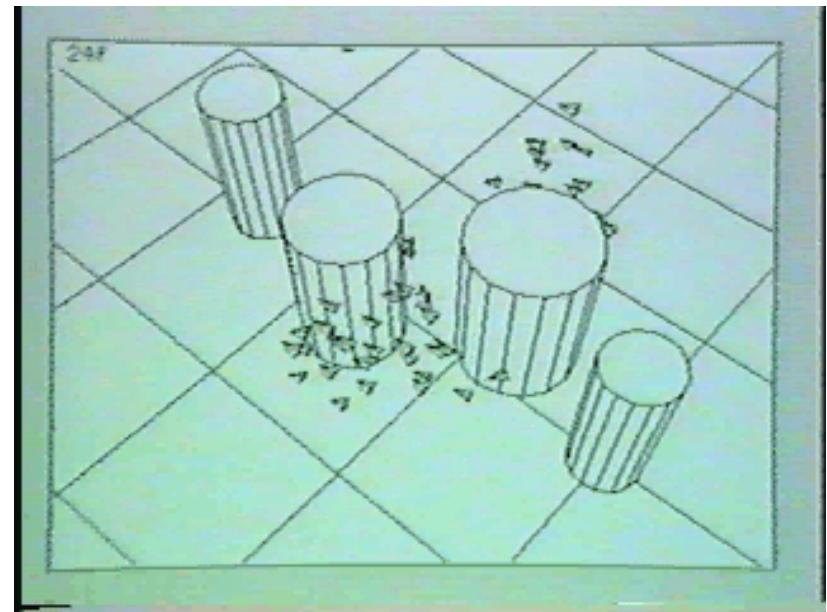


Lisa Miller, <http://www.youtube.com/watch?v=VBzXDrz8rMI>

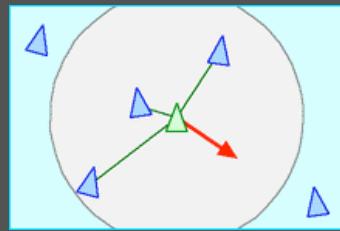
goal: exit here

# Swarm Intelligence: Formations

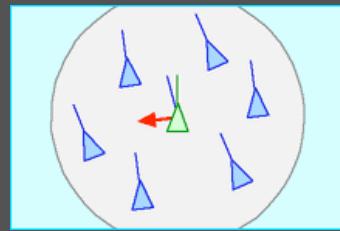
- collective behavior of decentralized, self-organized systems, natural or artificial
- complex group behavior emerges from combination of simple local controllers
- Reynolds' Boids (1987) for flocking



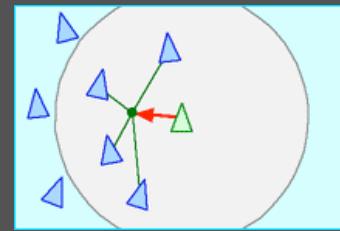
**Separation:** steer to avoid crowding local flockmates



**Alignment:** steer towards the average heading of local flockmates



**Cohesion:** steer to move toward the average position of local flockmates



# Swarm Intelligence: Box clustering

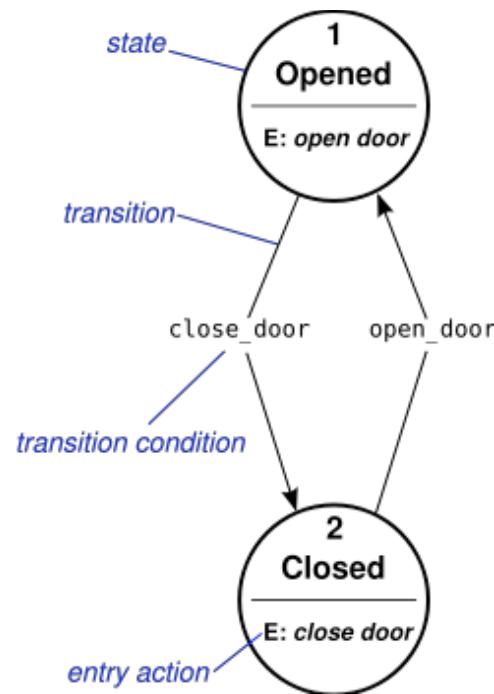


[https://www.youtube.com/watch?v=b\\_kZmatqAaQ](https://www.youtube.com/watch?v=b_kZmatqAaQ)

How do we computationally  
represent reactive control?

# Finite State Machines

- Components
  - alphabet (or inputs)
    - “observations” in robotics
  - states (some robot action)
  - transitions (between states)
  - stopping condition
- Commonly, implemented as switch-case or if-else within a while loop



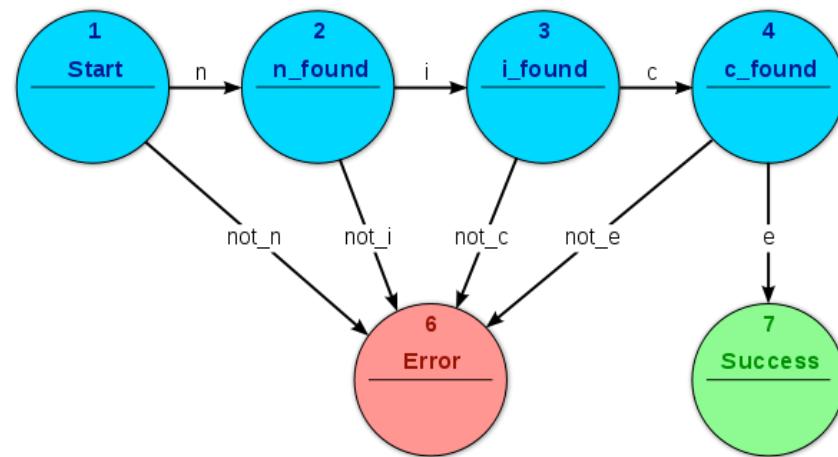
[http://en.wikipedia.org/wiki/Switch\\_statement](http://en.wikipedia.org/wiki/Switch_statement)

# “nice” recognizer

- recognize the string “nice” from input

- if input is “nice”
  - output **success**
- if input not “nice”
  - output **error**

- robotics uses
  - preconditions (enter state)
  - postconditions (exit state)

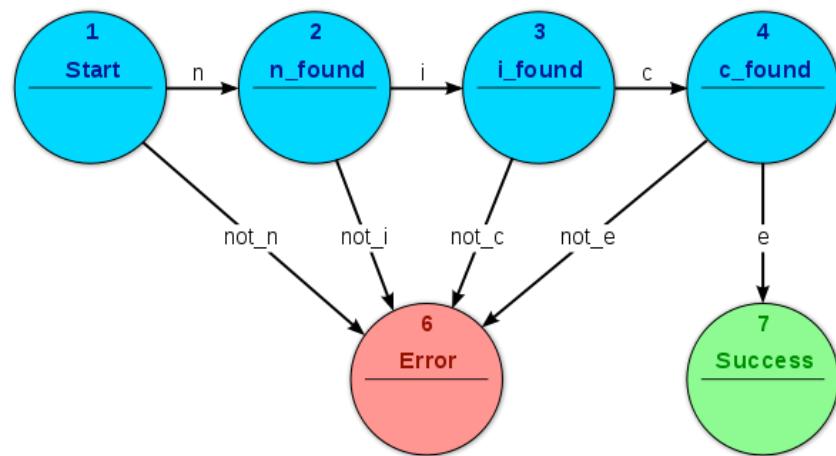


```

state ← start
while state != success and state != error
    token ← <next string character from input>
    switch (state):
        case start:
            if token = "n" then state ← n_found
            else state ← error
            break
        case n_found:
            if token = "i" then state ← i_found
            else state ← error
            break
        case i_found:
            if token = "c" then state ← c_found
            else state ← error
            break
        case c_found:
            if token = "e" then state ← success
            else state ← error
            break
    end while loop
    output ← state

```

“nice” recognizer



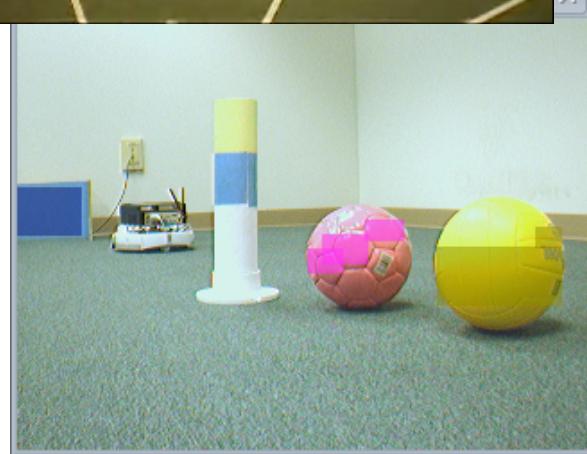
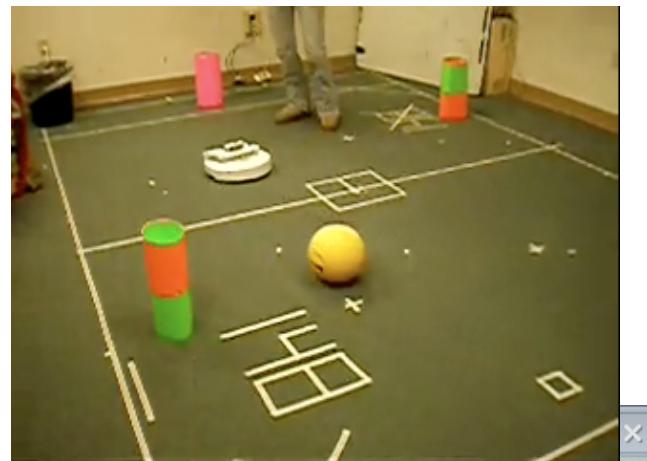
Consider input: “nice”

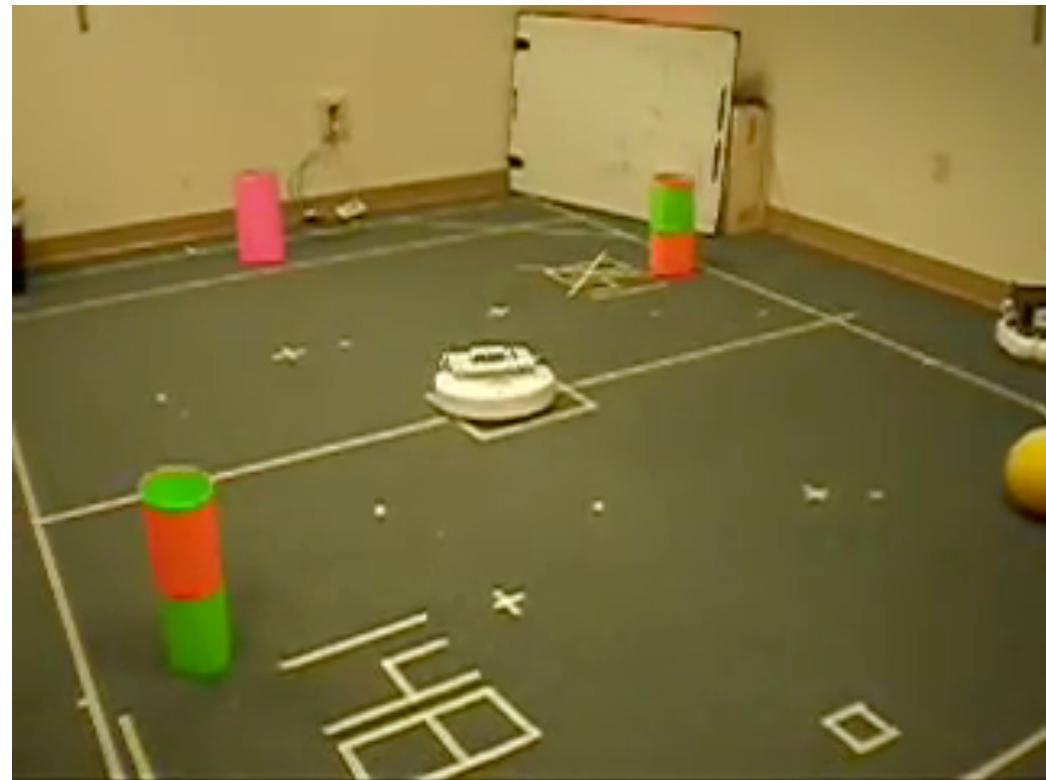
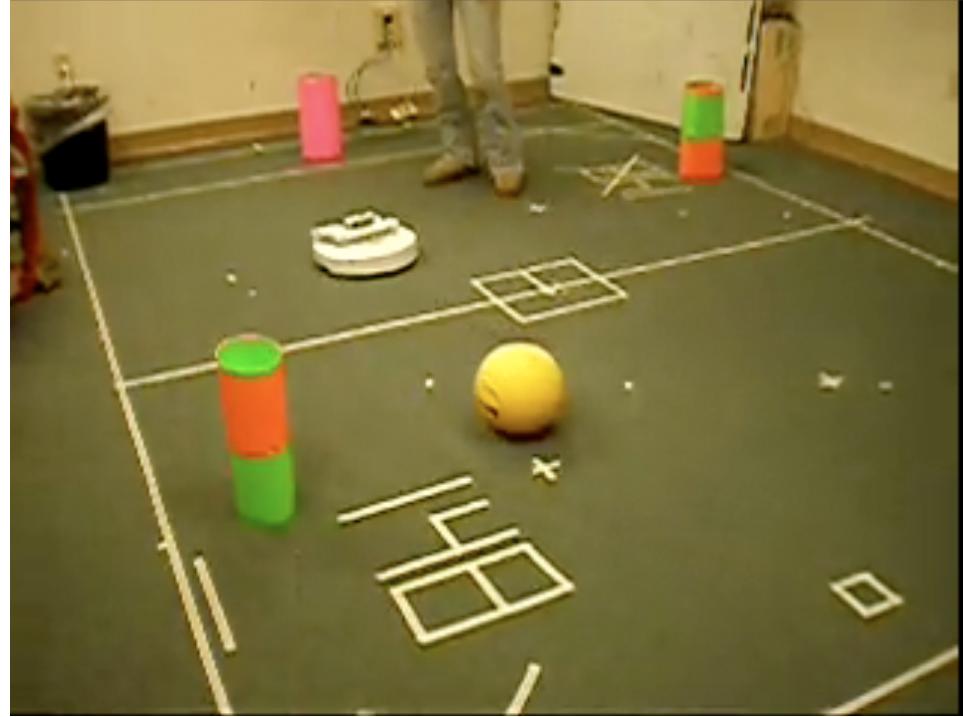
Consider input: “robotics”

Consider input: “niece”

# Move to objects in sequence?

- How to move a mobile robot to a given sequence of objects?
  - yellow ball
  - green/orange landmark
  - pink landmark
  - orange/green landmark



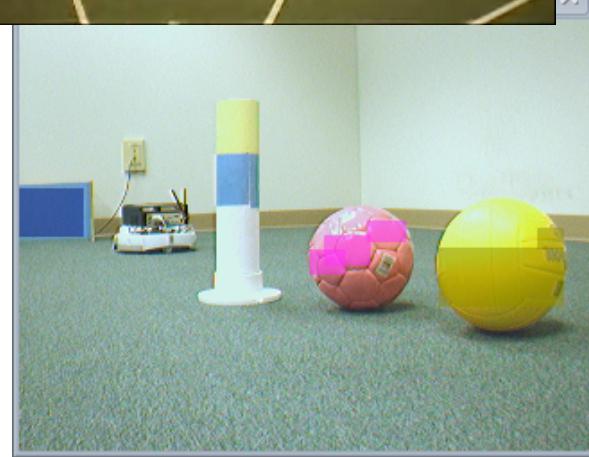
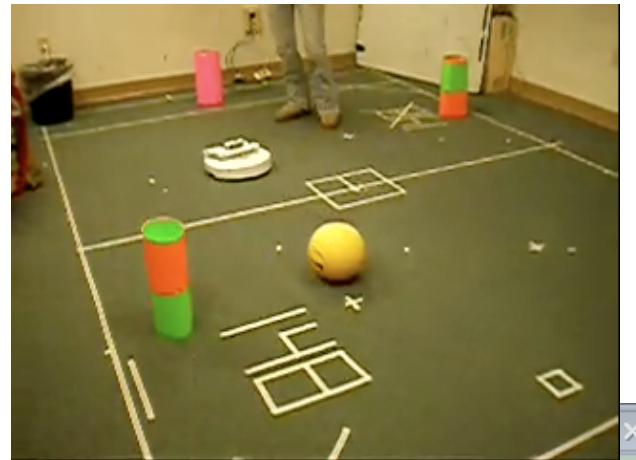


# Object Seeking

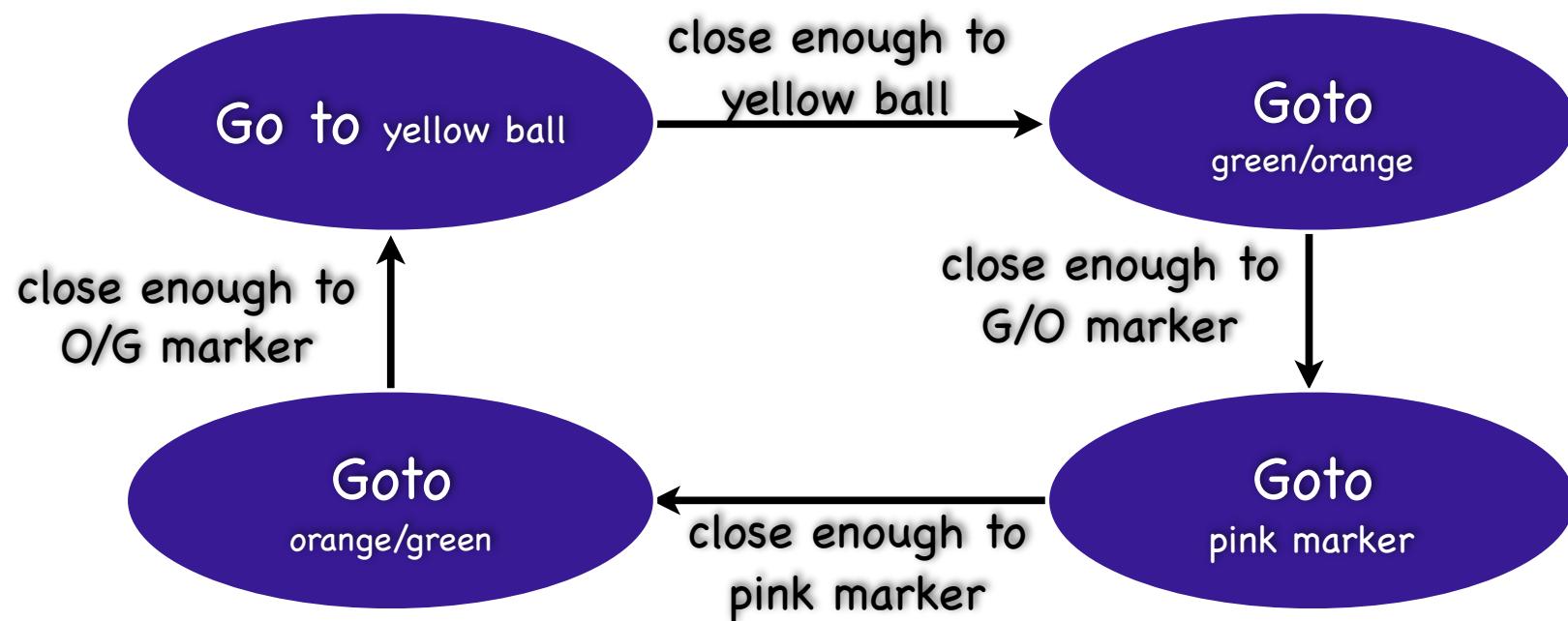
<http://www.youtube.com/watch?v=-hOA0jMUggg>

# Move to objects in sequence?

- What are the states?
- What are the transitions?
- Preconditions for states?
- Postconditions for states?
- Can someone sketch on board?

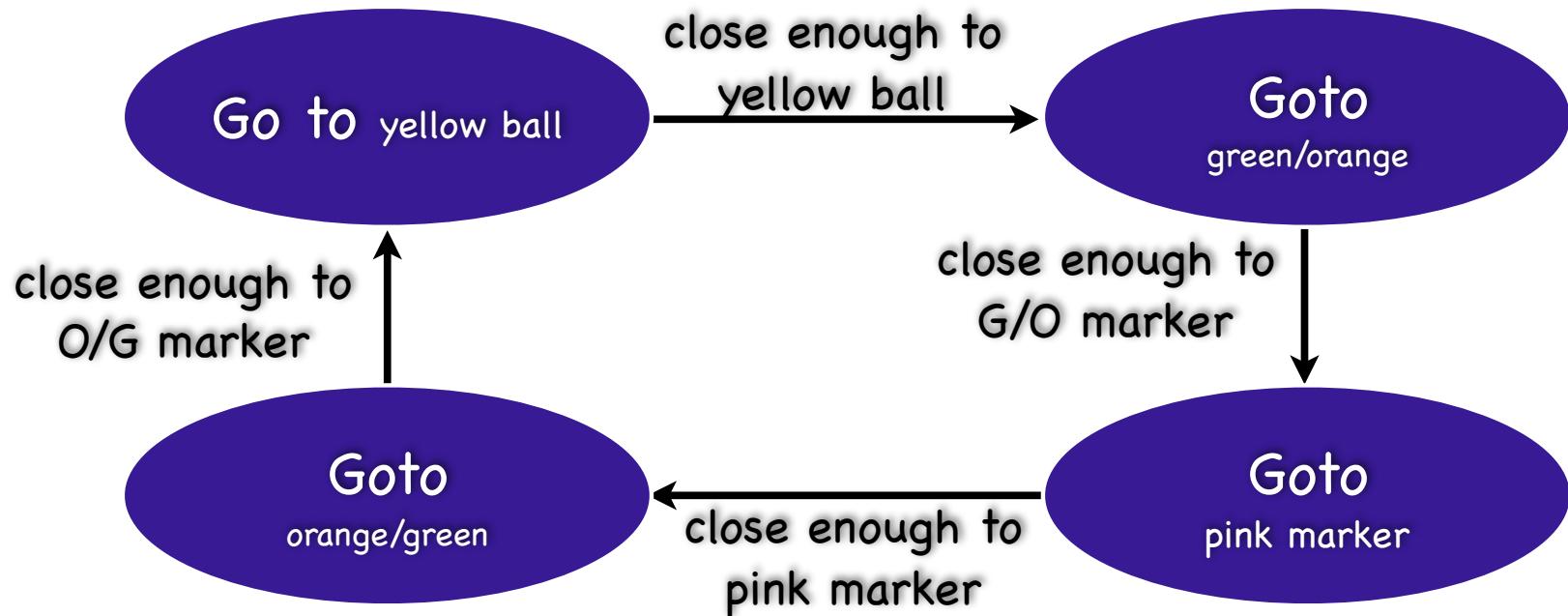


# Object seeking FSM



# Object seeking FSM

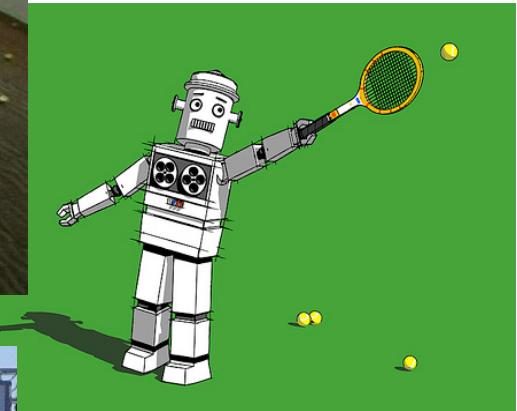
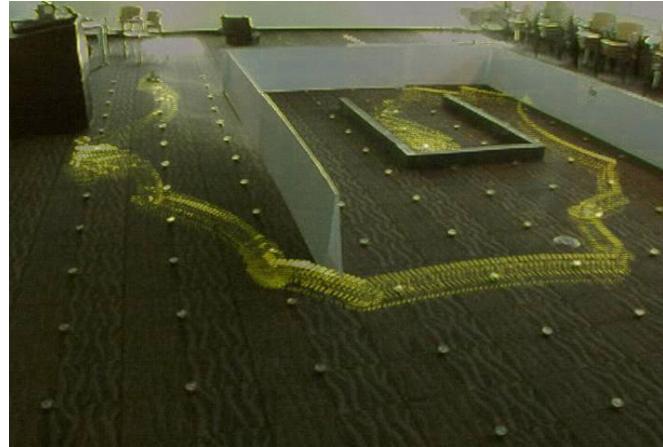
How to implement state?



How to detect “close enough”?

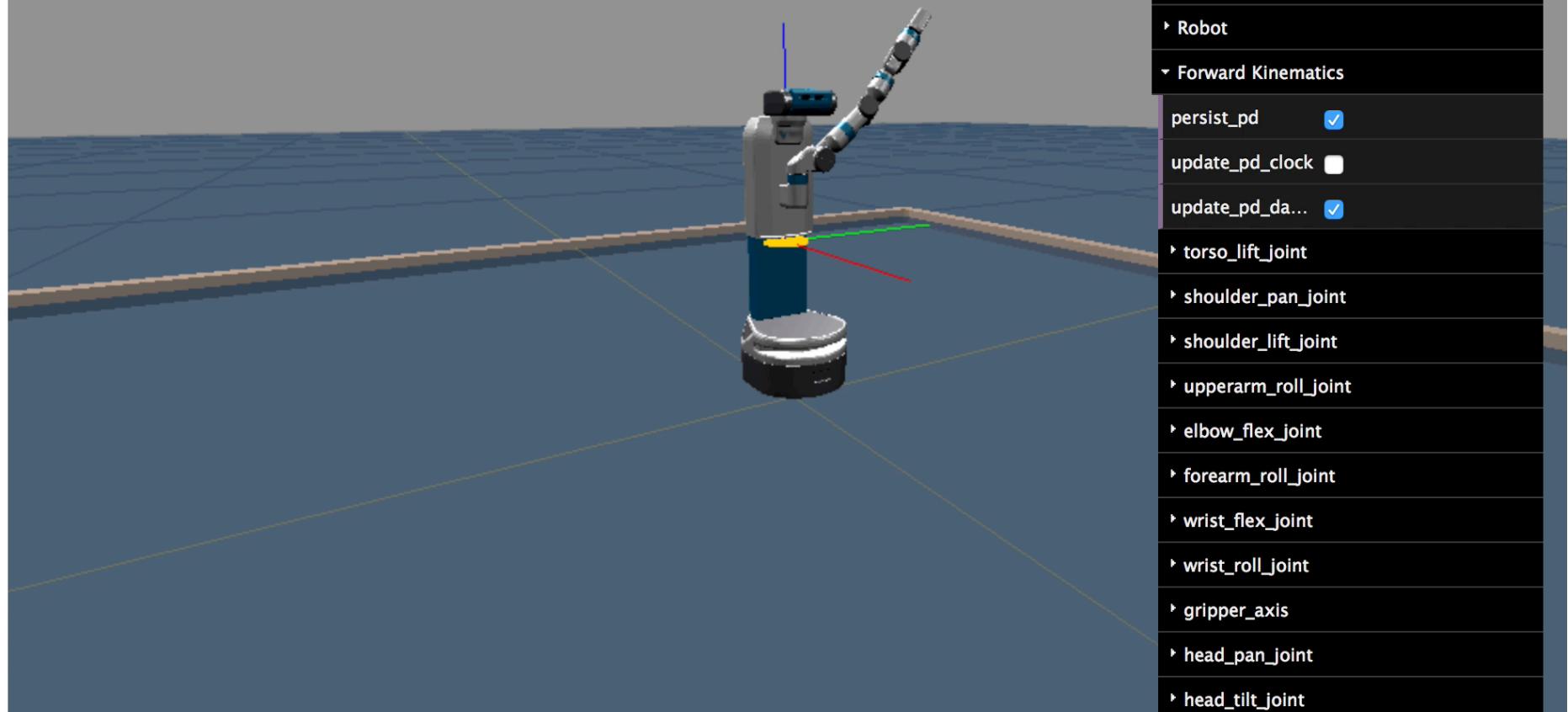
# FSMs for Other Tasks

- Robot foraging?
- Robot tennis/pong?
- Pushing a ball into a goal?
- Vacuuming a room
- Driving a car?
- Robot dancing!



# Assignment 4: FSMs for robot dancing

```
joint servo controller has been invoked  
executing dance routine, pose 2 of 10
```



Assignment 4 robot dance (ohseejay)  
<https://www.youtube.com/watch?v=WyQ9aoB3bpl>



oj 3:55 PM

@cszechy: My best attempt to recreate. Just add the following below to kineval.js in initScene() just before the endeffector geometry is created:

```
light_drake3 = new THREE.AmbientLight(0x602845);
scene.add( light_drake3 );

var geometry = new THREE.PlaneGeometry( 8, 20, 32 );
var material = new THREE.MeshLambertMaterial( {color: 0xb05890, side:
THREE.DoubleSide} );
var plane = new THREE.Mesh( geometry, material );
plane.rotateOnAxis({x:0,y:0,z:1},-Math.PI*1.2/2); //+Math.PI),
plane.rotateOnAxis({x:1,y:0,z:0},-Math.PI/2),
plane.position.set(-3,0,0);
plane.material.transparent = false;
plane.material.opacity = 1;
scene.add( plane );

plane = new THREE.Mesh( geometry, material );
plane.rotateOnAxis({x:0,y:0,z:1},-Math.PI*1.2/2),
plane.rotateOnAxis({x:1,y:0,z:0},-Math.PI/2),
plane.position.set(3,0,0);
plane.material.transparent = false;
plane.material.opacity = 1;
scene.add( plane );
```

```
light_drake3 = new THREE.AmbientLight(0x602845);
scene.add( light_drake3 );

var geometry = new THREE.PlaneGeometry( 8, 20, 32 );
var material = new THREE.MeshLambertMaterial( {color: 0xb05890, side: THREE.DoubleSide} );
var plane = new THREE.Mesh( geometry, material );
plane.rotateOnAxis({x:0,y:0,z:1},-Math.PI*1.2/2);
plane.rotateOnAxis({x:1,y:0,z:0},-Math.PI/2),
plane.position.set(-3,0,0);
plane.material.transparent = false;
plane.material.opacity = 1;
scene.add( plane );

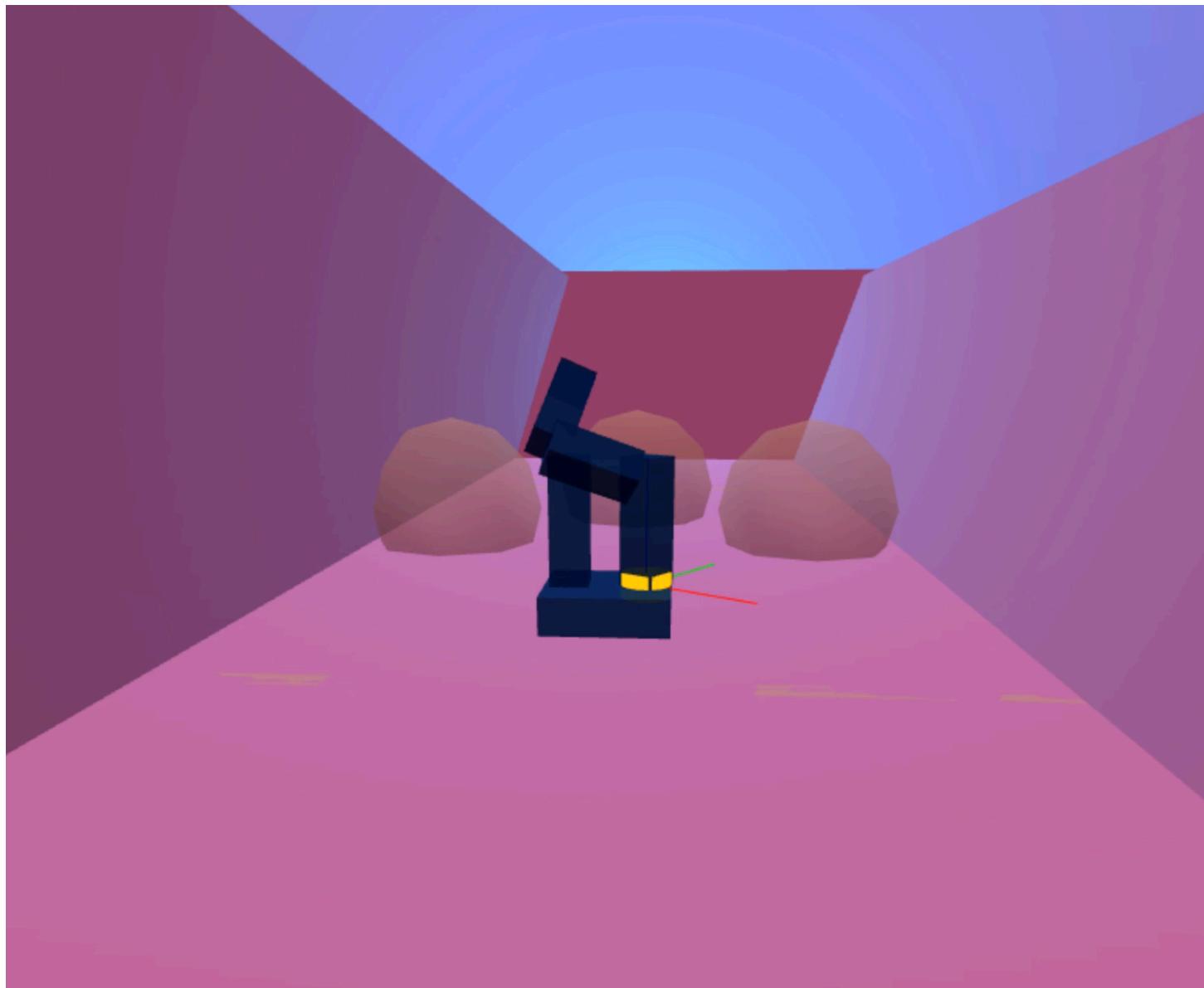
plane = new THREE.Mesh( geometry, material );
plane.rotateOnAxis({x:0,y:0,z:1},-Math.PI*1.2/2),
plane.rotateOnAxis({x:1,y:0,z:0},-Math.PI/2),
plane.position.set(3,0,0);
plane.material.transparent = false;
plane.material.opacity = 1;
scene.add( plane );

[3:56]
plane = new THREE.Mesh( geometry, material );
plane.rotateOnAxis({x:1,y:0,z:0},-Math.PI/2),
plane.position.set(0,0,1,0);
plane.material.transparent = false;
plane.material.opacity = 1;
scene.add( plane );

plane = new THREE.Mesh( geometry, material );
material = new THREE.MeshPhongMaterial( {color: 0xff8db0, side: THREE.DoubleSide} );
plane.rotateOnAxis({x:0,y:0,z:1},Math.PI/2),
plane.position.set(0,0,-11);
plane.material.transparent = false;
plane.material.opacity = 1;
scene.add( plane );

var geometry = new THREE.PlaneGeometry( 800, 800, 800 );
material = new THREE.MeshPhongMaterial( {color: 0x8070cf, side: THREE.DoubleSide} );
plane = new THREE.Mesh( geometry, material );
plane.position.set(0,0,-19);
plane.material.transparent = false;
plane.material.opacity = 1;
scene.add( plane );

var light_drake4 = new THREE.PointLight( 0xfffff, 1, 30, 0.1 );
light_drake4.position.set( 0, 2, -13 );
scene.add( light_drake4 );
```



cszechy  
ohseejay

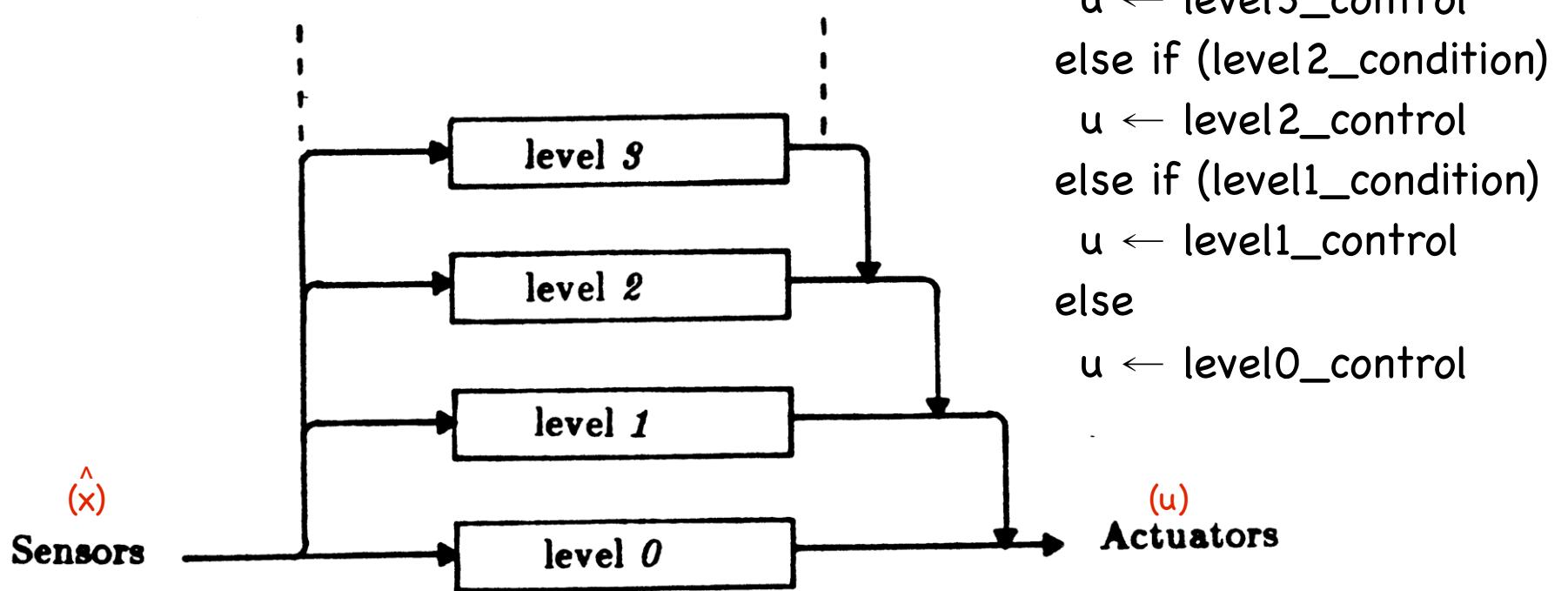
Let's generalize FSMs  
for robot control

# Subsumption Architecture

[Brooks 1986]

- Generalization of FSM-based control
- Collection of modular reactive controllers in a priority hierarchy
- Controllers can be FSMs
- Large nested if-else statement
- Most robots are controlled by some form of subsumption

# Subsumption Architecture

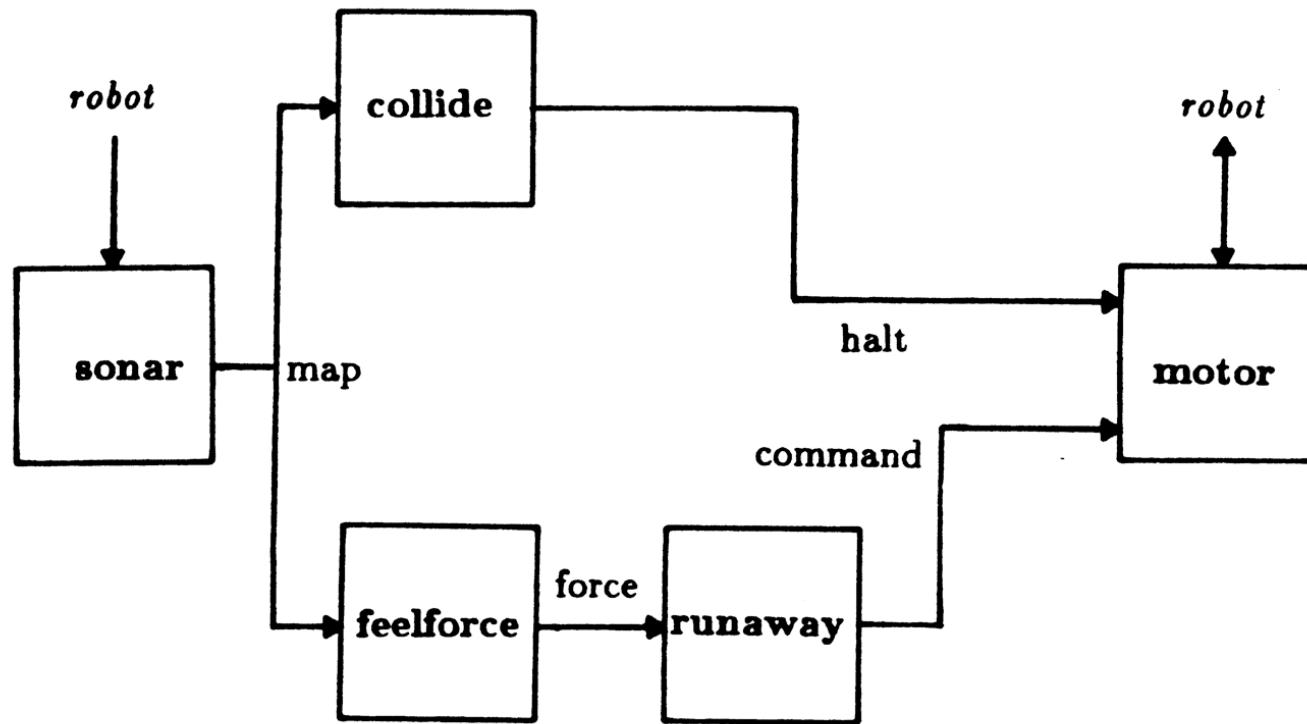


# Subsumption Design Process

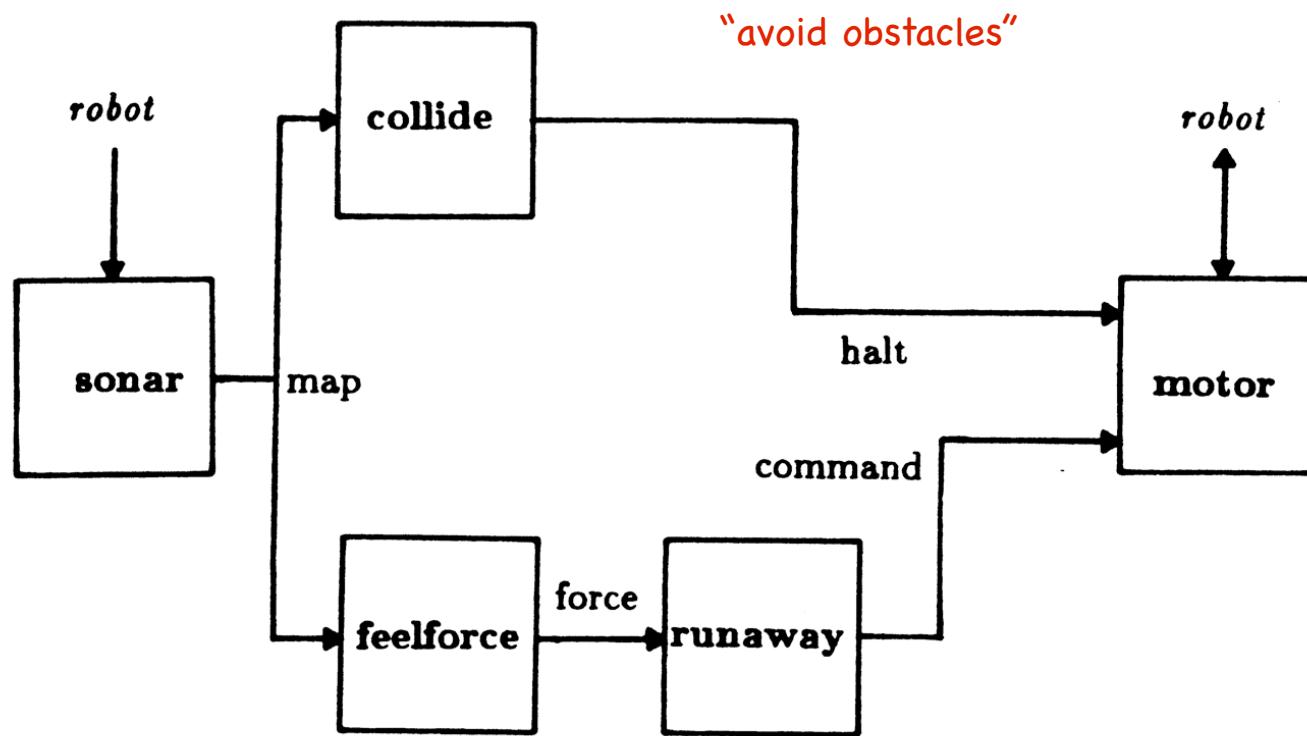
1. Divide your problem into basic competencies ordered simple to more complex.  
Designate a level for each basic competency.
2. Further subdivide each level into multiple simple components which interact through shared variables. Limit the sharing of variables among levels to avoid incomprehensible code.
3. Implement each module as a separate light-weight thread. You might think of setting the priorities for these threads so that modules in a given level have the same priority.
4. Implement suppression and inhibition as one or more separate "arbitration" processes that serve to control access to shared variables. You might want to control access using semaphores.

<http://www.cs.brown.edu/~tld/courses/cs148/02/architectures.html>

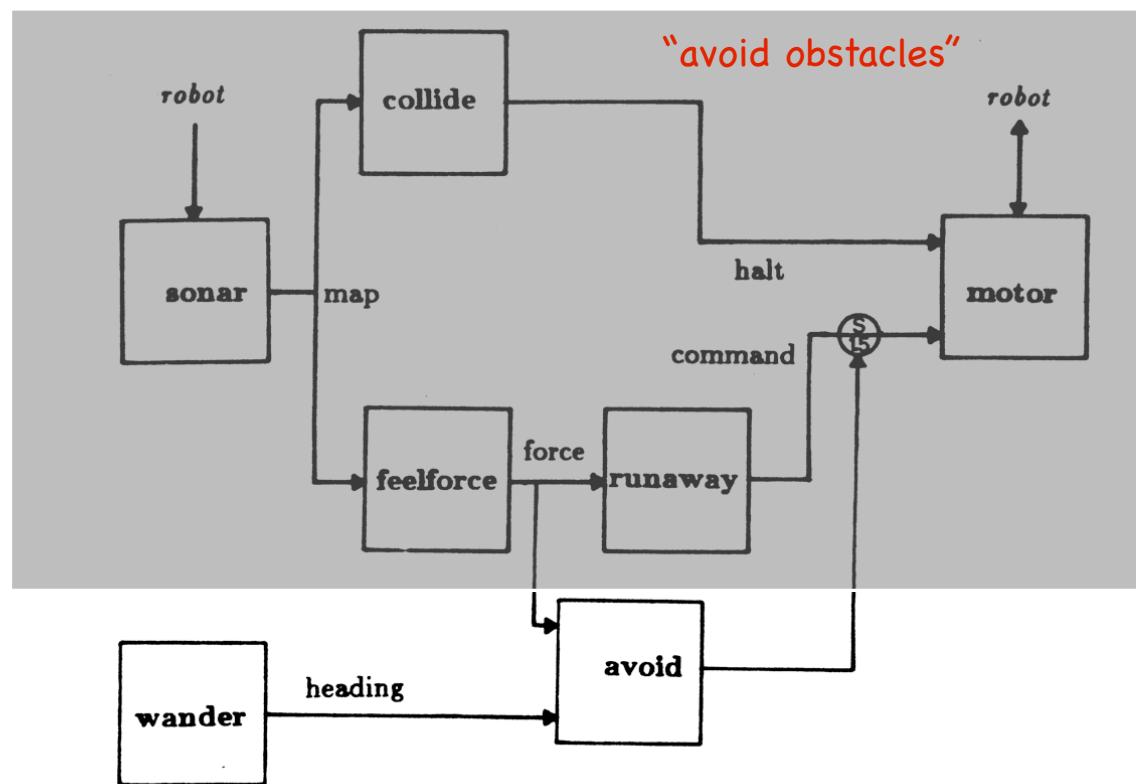
# What will this do?



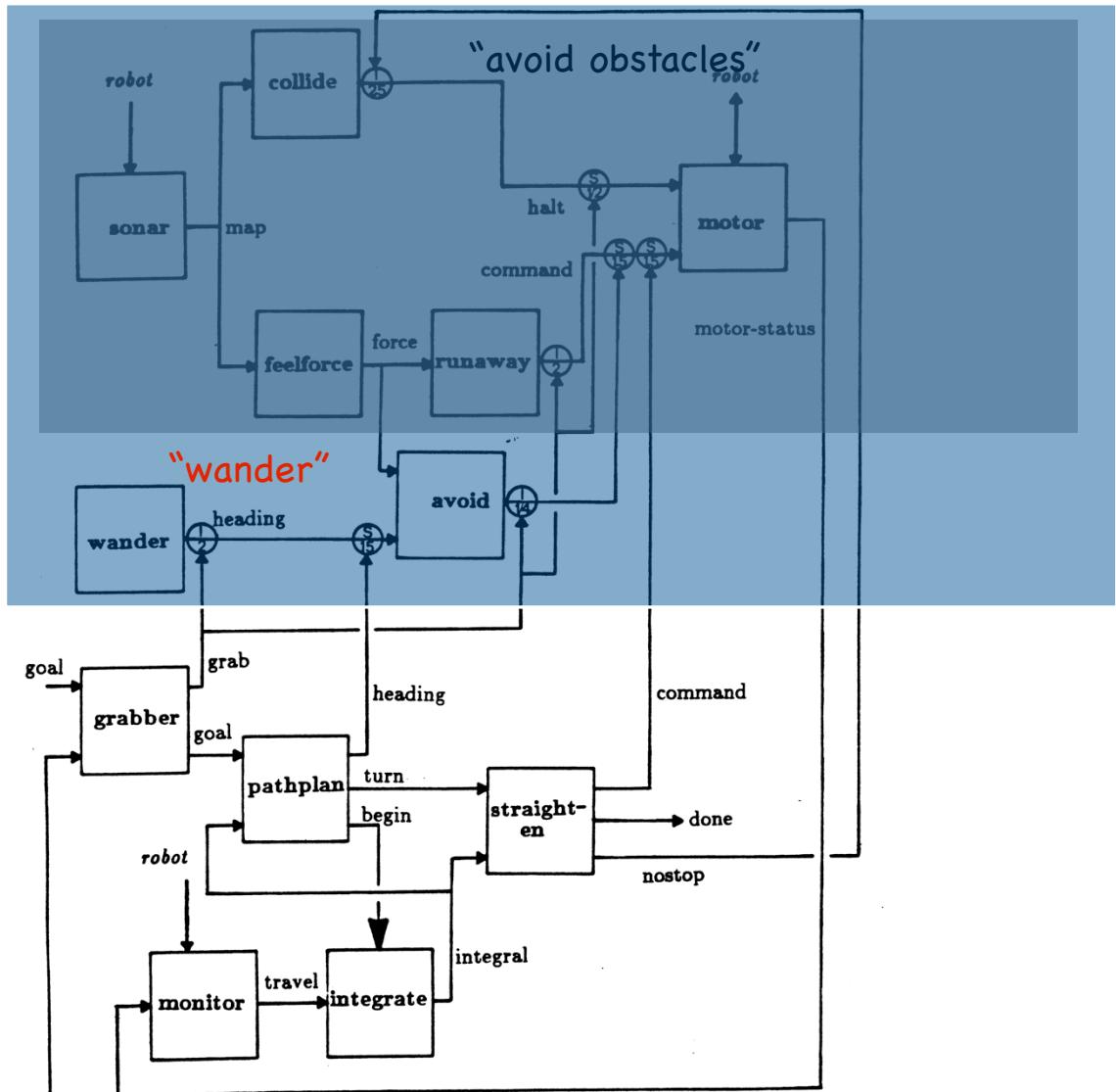
# What will this do?



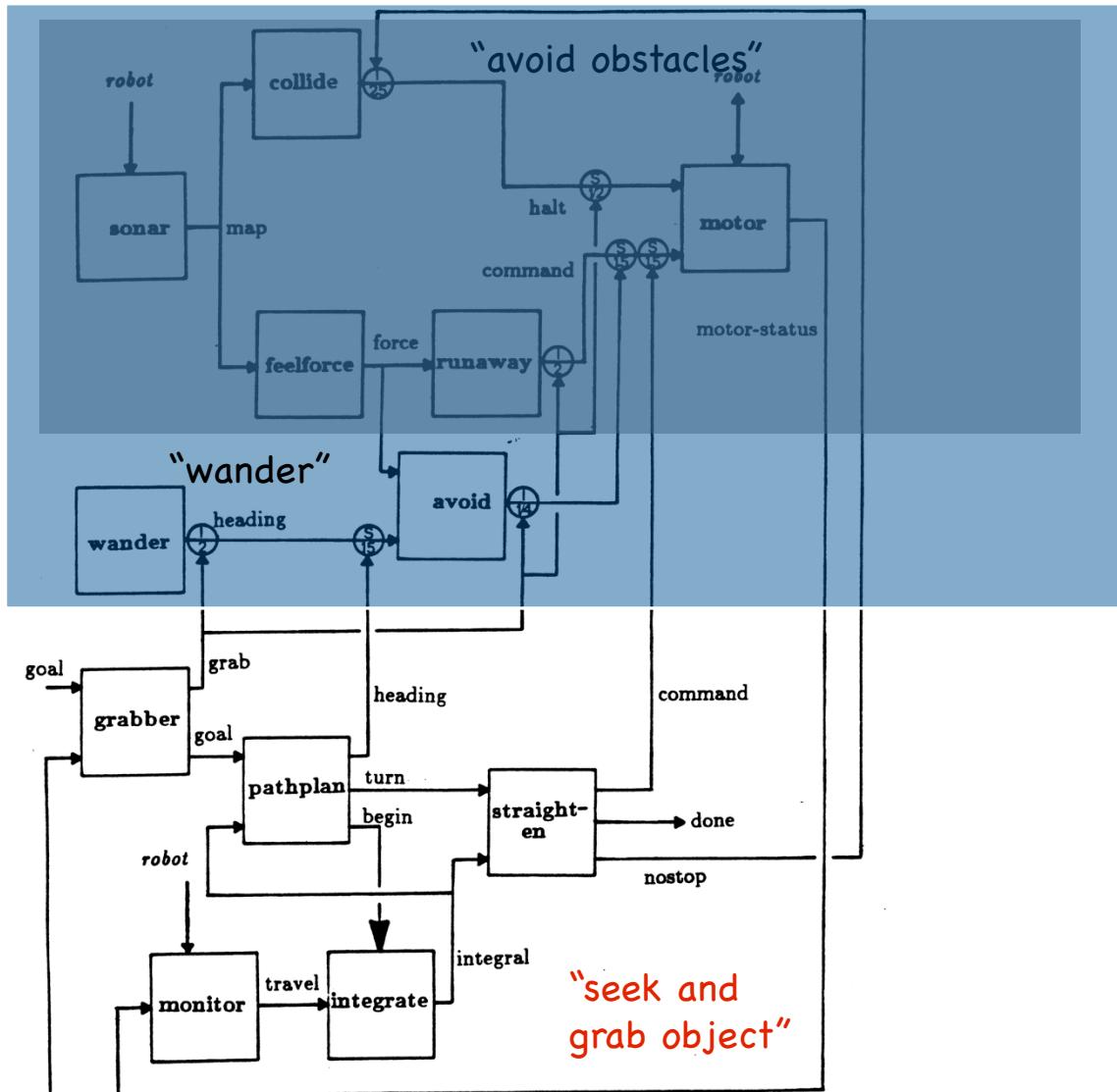
# What will this do?



# What will this do?



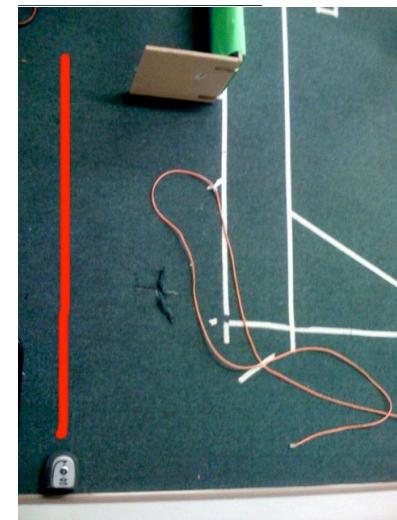
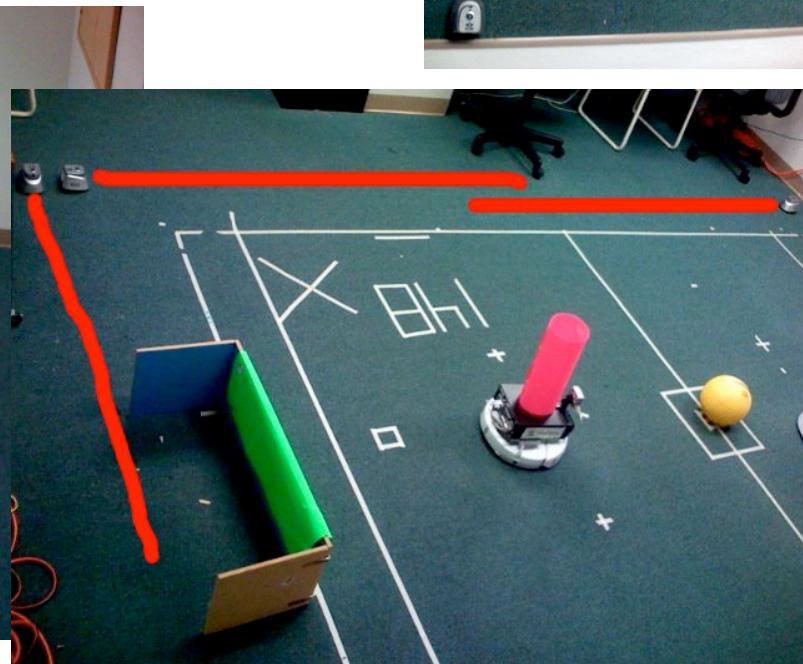
# What will this do?



# 2010: Assignment 5

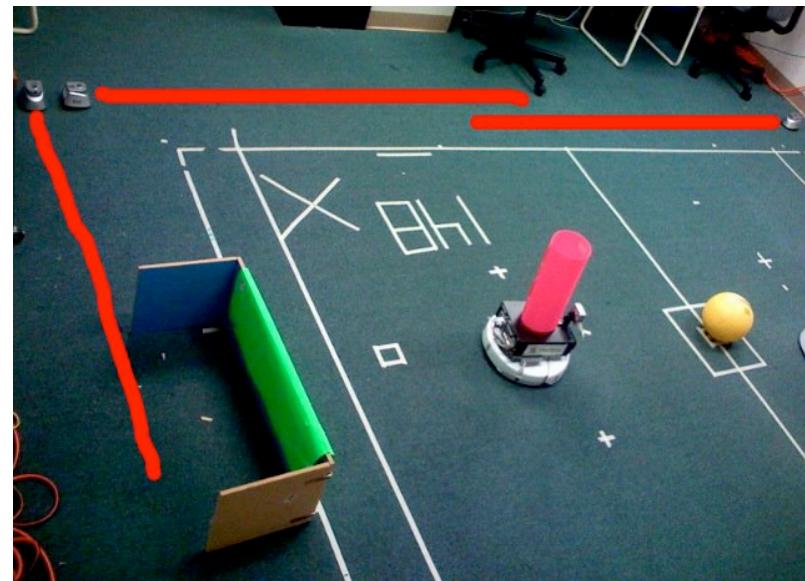
Subsumption for robot soccer

- Propose modules and priority?



# What behavior will result?

1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball



# Snappy's Subsumption: Goal Scoring

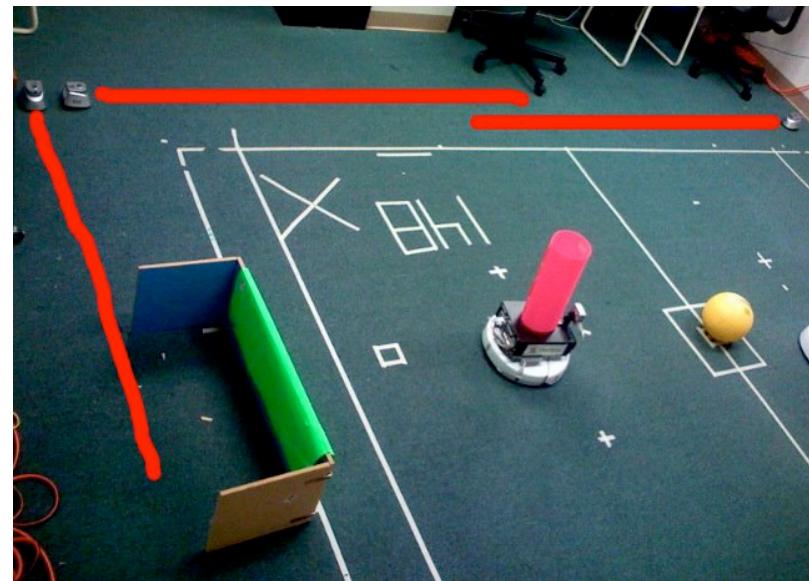
1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball



Goal Scoring Challenge - Put ball on orange

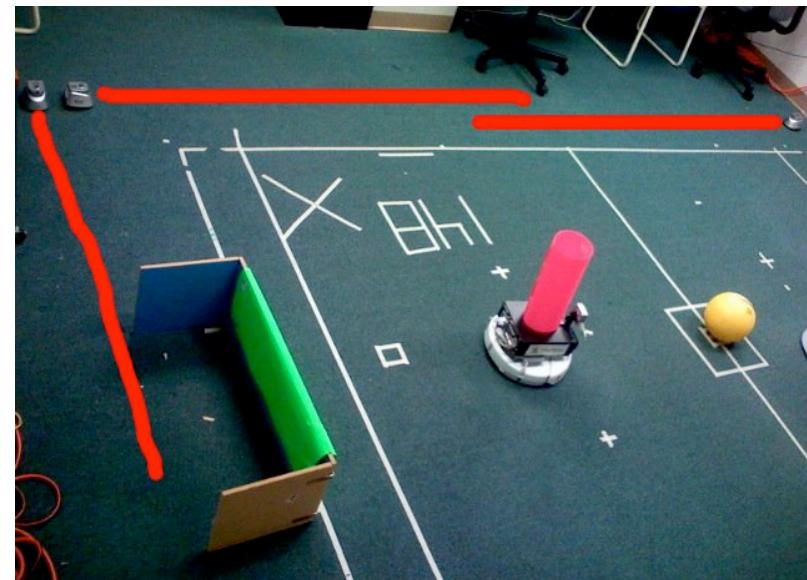
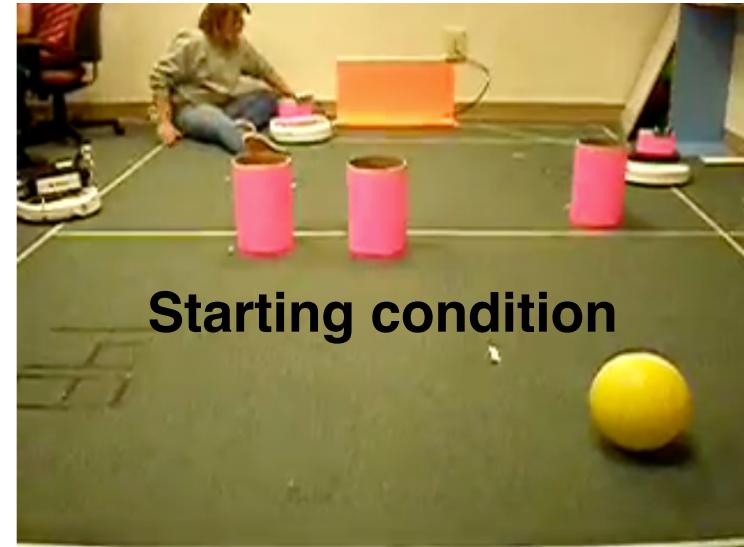
# What behavior will result?

1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball



# What behavior will result?

1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball



# Snappy's Subsumption: Navigate to Ball

1. Avoid IR Wall
2. Avoid Robot
3. Avoid Fiducial
4. Bumper Hit
5. Go To Opposite Goal
6. Go To Any Goal
7. Line Up On Ball
8. Go To Ball
9. Score Goal
10. At Ball
11. Look For Ball



Navigation Challenge - Navigate to Ball

# Snappy in competition



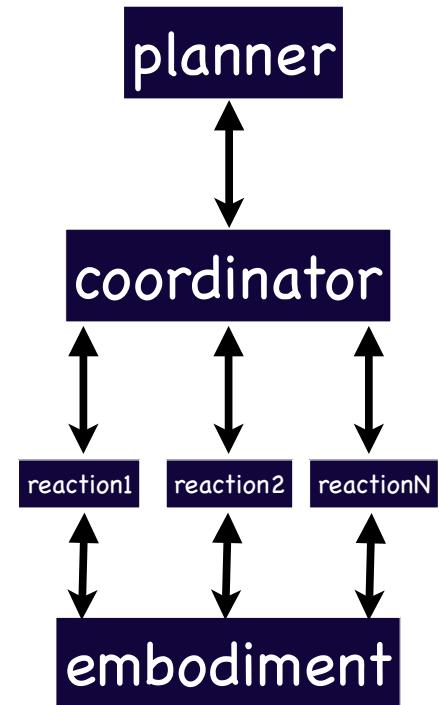
Are there other methods of  
decision making?

# Types of Decision Making

- Deliberative (Planner-based) Control
  - “Think hard, act later.”
- Reactive Control
  - “Don’t think, (re)act.”
- Hybrid Control
  - “Think and act separately & concurrently.”
- Behavior-Based Control
  - “Think the way you act.”

# Hybrid systems

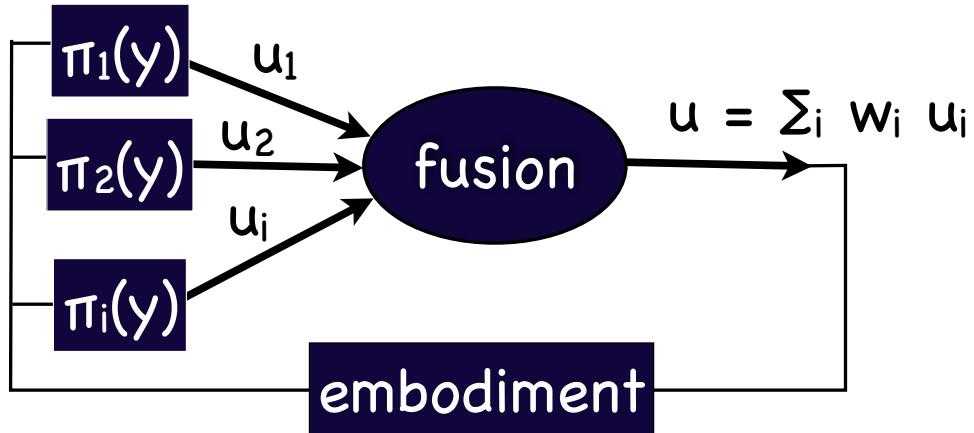
- Top-down planner for high-level goals
- Reaction for low-level immediate execution
- Interface layer to coordinate
  - how to balance long and short term
- Modern cost maps?
- Not discrete-continuous hybrid control



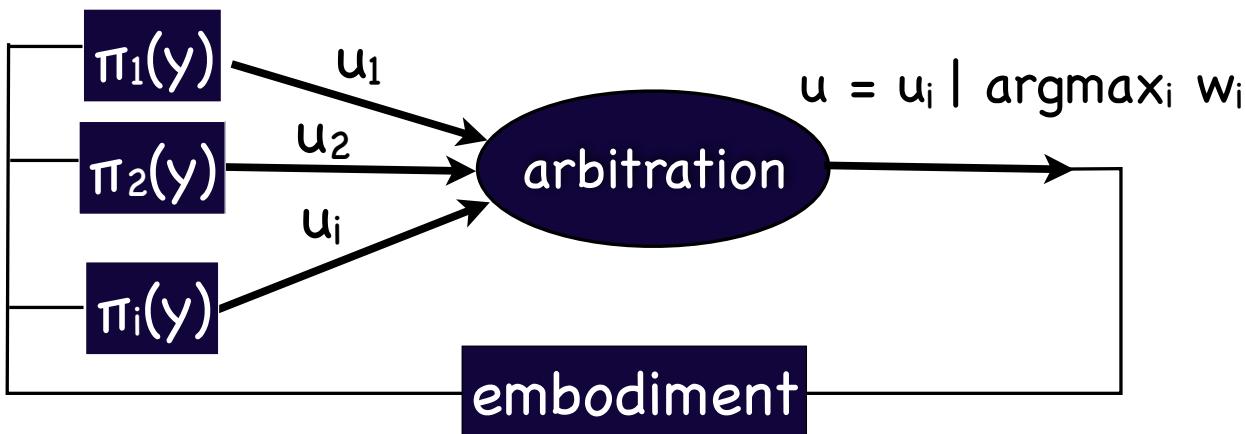
# Behavior-based systems

- All modules have equal priority, access to sensing, and output motor commands
- Modules can be deliberative, reactive, or whatever
- Commands merged through arbitration or fusion
- Potential fields?

# Arbitration vs. Fusion



Fusion:  
linearly combine  
 $u = \sum_i w_i u_i$



Arbitration:  
winner-take-all  
 $u = u_i \mid \text{argmax}_i w_i$



Next class:  
Inverse Kinematics