Georgia Institute of Technology

Dr. Magnus Egerstedt
Professor
School of Electrical and
Computer Engineering

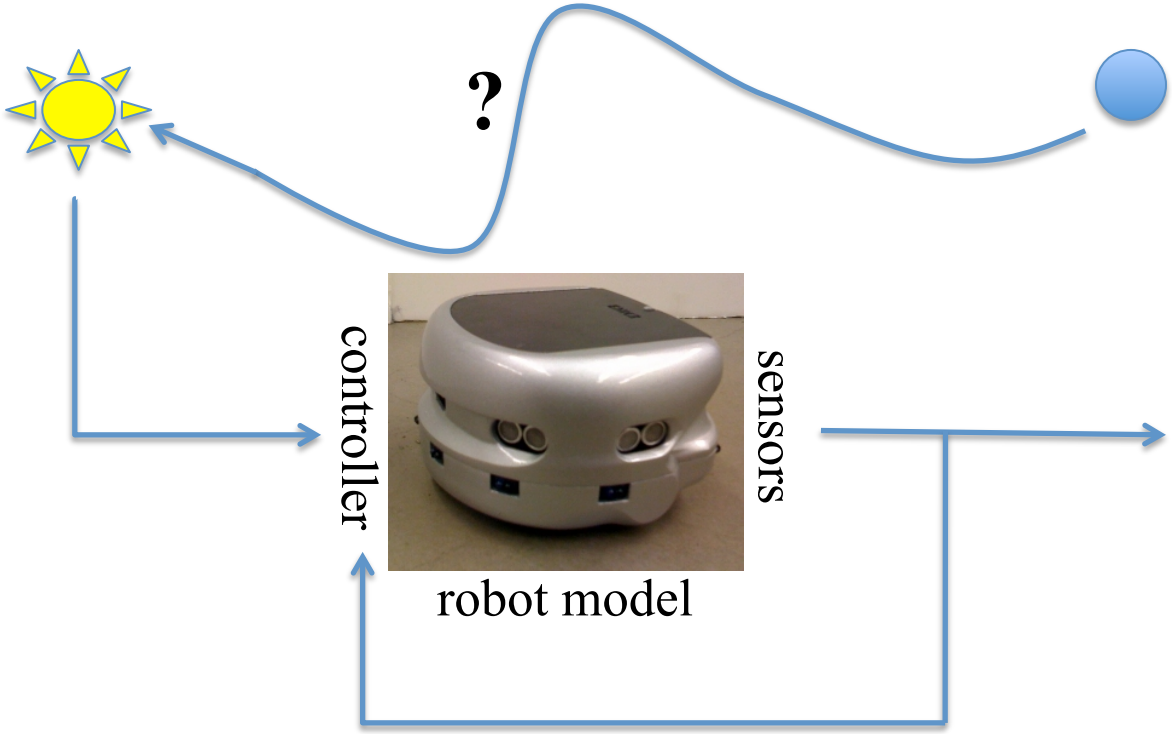
Control of Mobile Robots

Module 2
Mobile Robots

How make mobile robots move in effective, safe, predictable, and collaborative ways using modern control theory?

Lecture 2.1 – Driving Robots Around?

- What does it take to drive a robot from point A to point B?

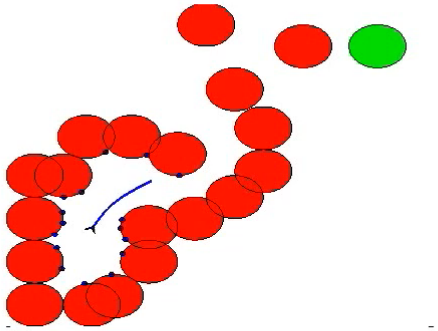


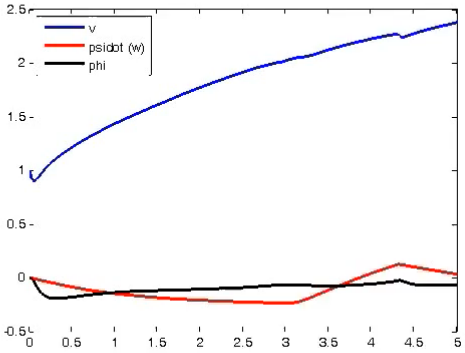
Divide and Conquer

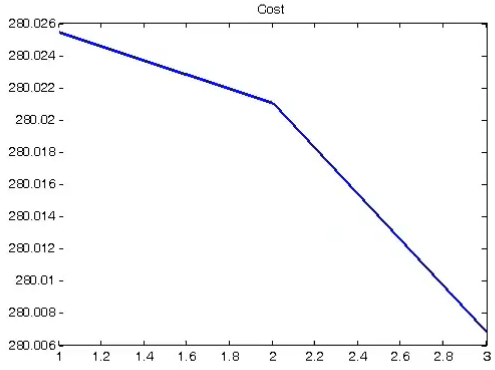
- The world is dynamic and fundamentally unknown
- The controller must be able to respond to environmental conditions
- Instead of building one complicated controller – divide and conquer: Behaviors
  - `Go-to-goal`
  - `Avoid-obstacles`
  - `Follow-wall`
  - `Track-target`
  - …

Behaviors

Behaviors

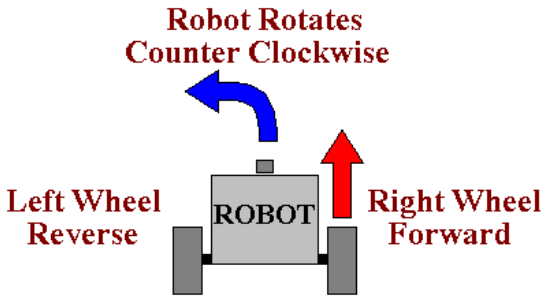


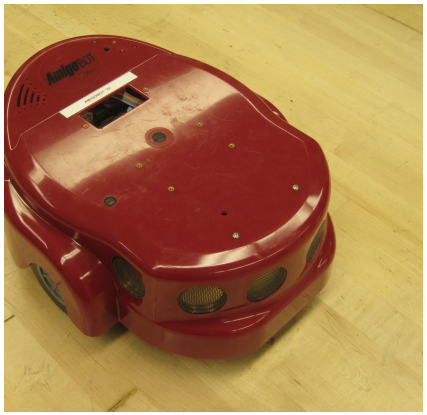

v
psidot (w)
phi
2.5
2
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Cost
280.026
280.024
280.022
280.02
280.018
280.016
280.014
280.012
280.01
280.008
280.006
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3

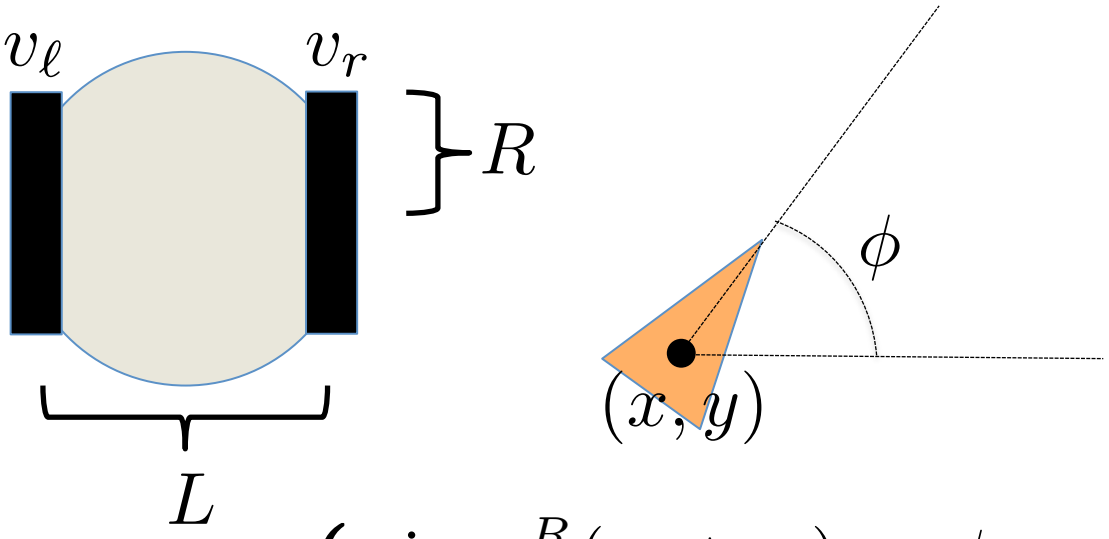
Lecture 2.2 – Differential Drive Robots

- In order to control mobile robots, we need models
- Differential drive wheeled robots – a very common type





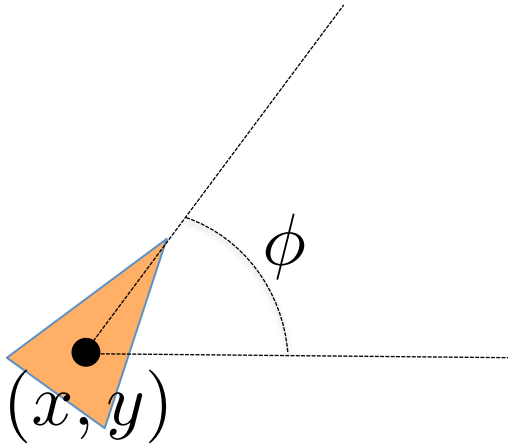
Model 1.0



$$\begin{cases} \dot{x} = \frac{R}{2}(v_r + v_\ell)\cos\phi \\ \dot{y} = \frac{R}{2}(v_r + v_\ell)\sin\phi \\ \dot{\phi} = \frac{R}{L}(v_r - v_\ell) \end{cases}$$

The "Unicycle" Model

- But it is not very natural to "think" in terms of wheel velocities
- Go directly for translational and angular velocities



- Inputs:

$$v$$
$$\omega$$

- Dynamics:

$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases}$$

Model 2.0

$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases}$$

Design for this model!

$$v = \frac{R}{2}(v_r + v_\ell) \;\Rightarrow\; \frac{2v}{R} = v_r + v_\ell$$

$$\omega = \frac{R}{L}(v_r - v_\ell) \;\Rightarrow\; \frac{\omega L}{R} = v_r - v_\ell$$

$$\begin{cases} \dot{x} = \frac{R}{2}(v_r + v_\ell)\cos\phi \\ \dot{y} = \frac{R}{2}(v_r + v_\ell)\sin\phi \\ \dot{\phi} = \frac{R}{L}(v_r - v_\ell) \end{cases}$$

Implement this model!

$$v_r = \frac{2v + \omega L}{2R}$$

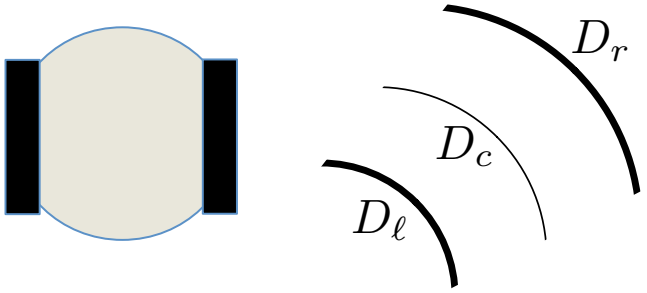
$$v_\ell = \frac{2v - \omega L}{2R}$$

Lecture 2.3 – Odometry

- The state of the robot is (x, y, ϕ)
- ***How do we obtain this state information?***
- Two possibilities:
  - External sensors
  - Internal sensors
    - Orientation: Compass, …
    - Position: Accelerometers, Gyroscopes, …
    - **Wheel Encoders**

Wheel Encoders

- Wheel encoders give the distance moved by each wheel
- Assume the wheels are following an arc (short time scale)

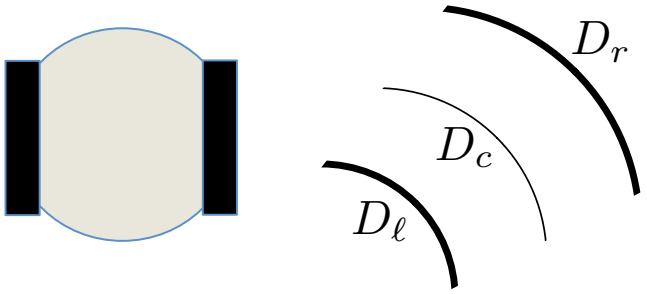


$$D_c = \frac{D_\ell + D_r}{2}$$

$$\begin{cases} x' = x + D_c \cos(\phi) \\ y' = y + D_c \sin(\phi) \\ \phi' = \phi + \dfrac{D_r - D_\ell}{L} \end{cases}$$

Wheel Encoders

- But how do we know how far each wheel has moved?



- Assume each wheel has N "ticks" per revolution
- Most wheel encoders give the total tick count since the beginning

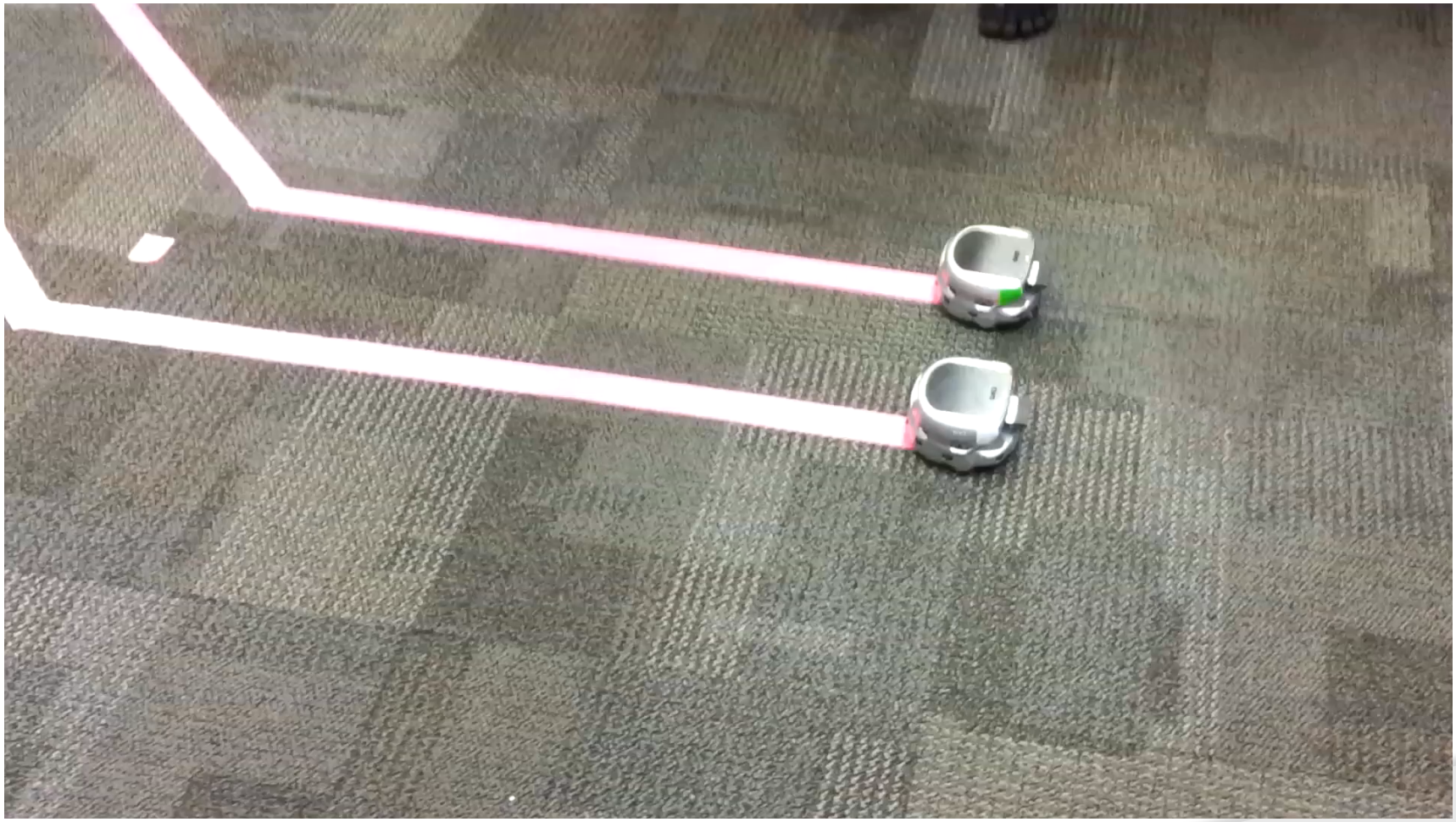
- For both wheels:

$$\Delta\text{tick} = \text{tick}' - \text{tick}$$

$$D = 2\pi R \frac{\Delta\text{tick}}{N}$$

A Major Disclaimer

DRIFT!!!

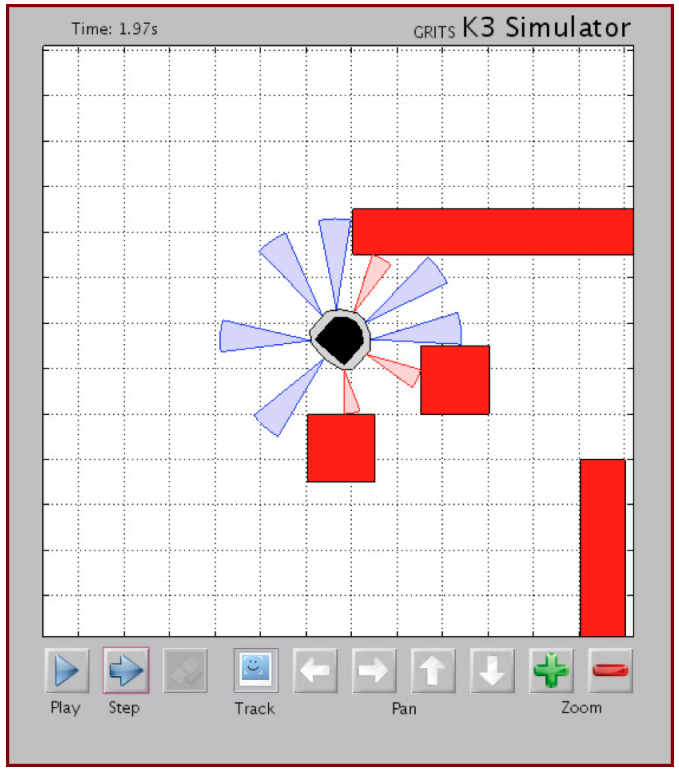


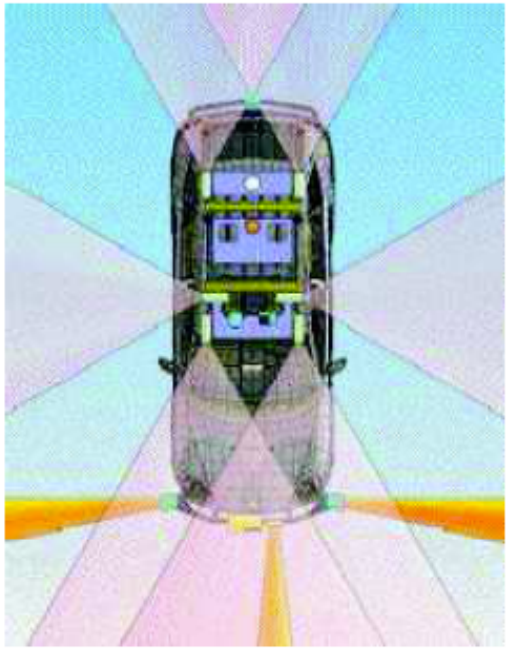
Lecture 2.4 – Sensors

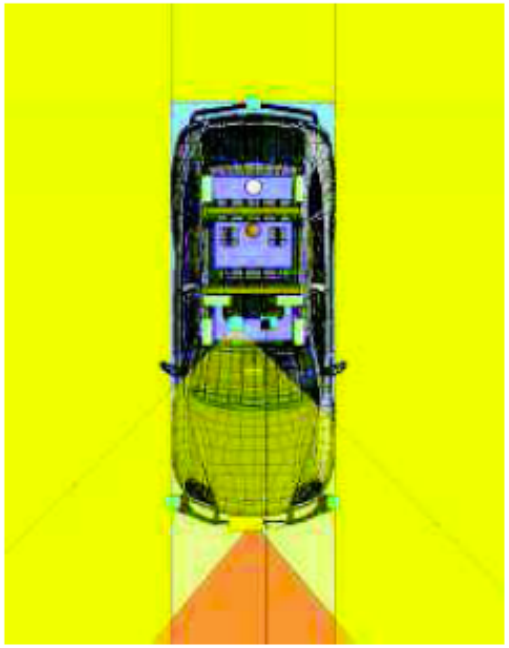
- Robots need to know what the world around them looks like
- The standard sensor suite includes a "skirt" of range sensors:
  - IR, Ultra-Sound, LIDAR,…
- Other standard external sensors include
  - Vision
  - Tactile
  - "GPS"

Range-Sensor Skirts

- We will mainly deal with range-sensors

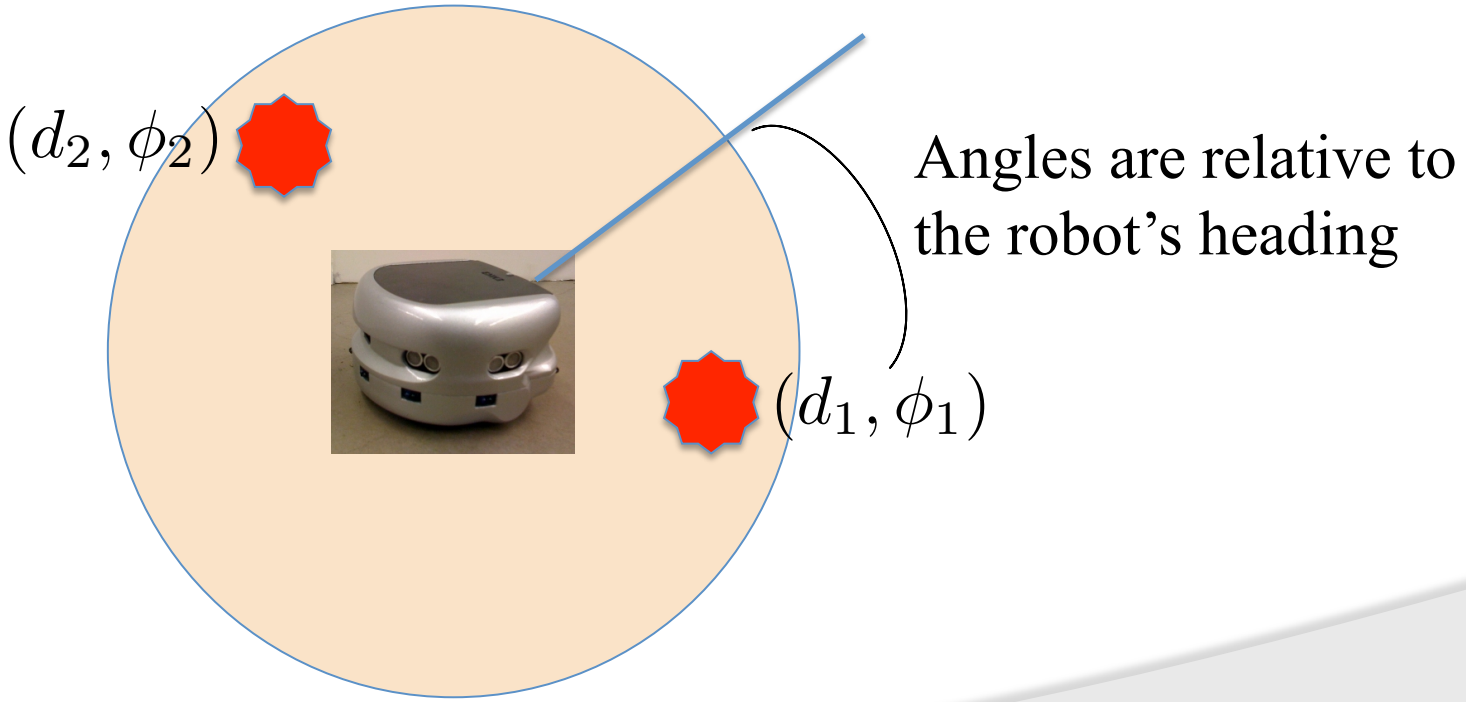






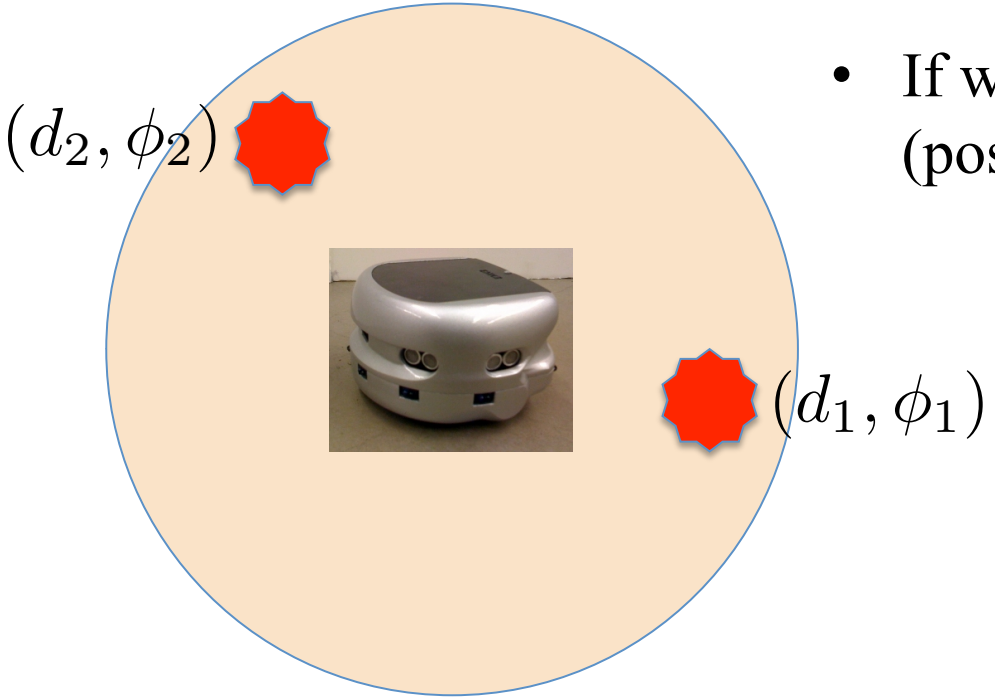
The Disk Abstraction

- Instead of worrying about the resolution of the sensors, assume we know the distance and direction to all obstacles around us (that are close enough)



The Disk Abstraction

- Instead of worrying about the resolution of the sensors, assume we know the distance and direction to all obstacles around us (that are close enough)

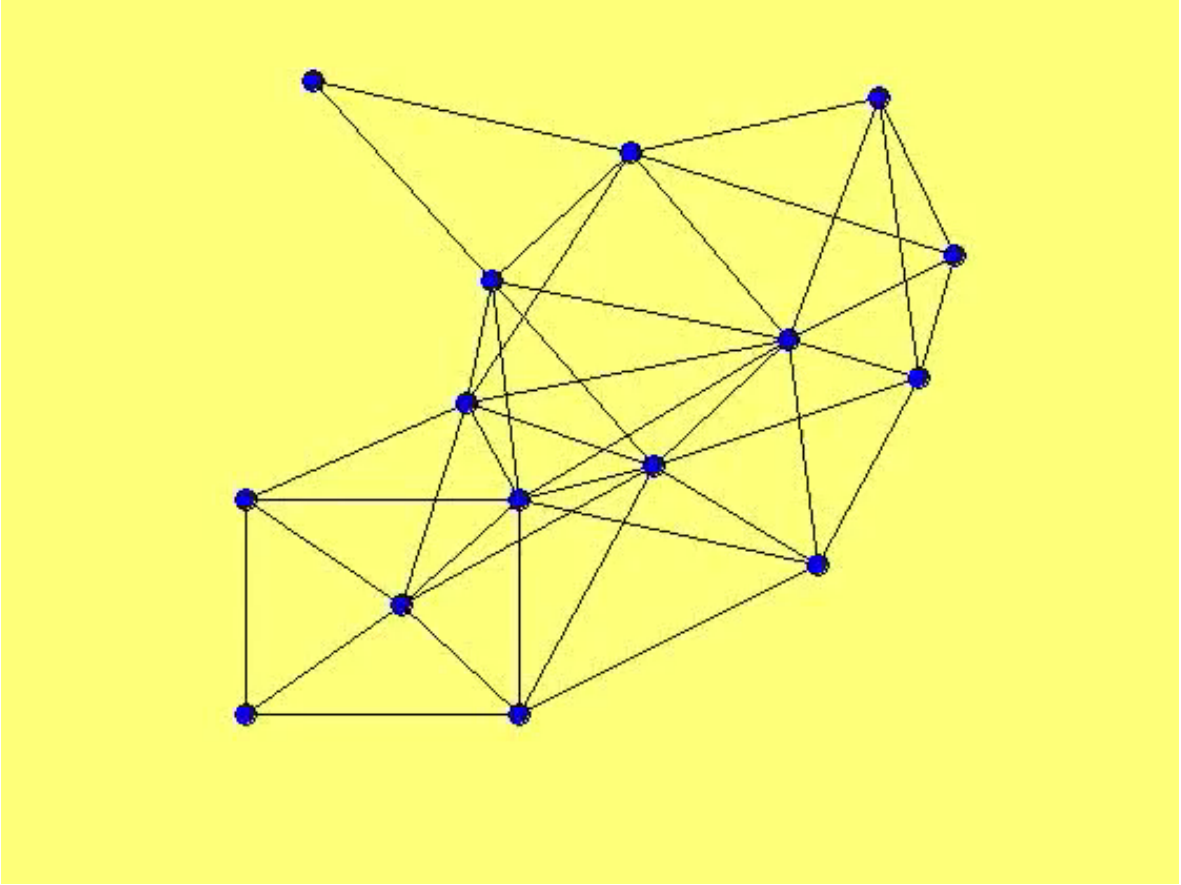


- If we know our own pose (position and orientation):

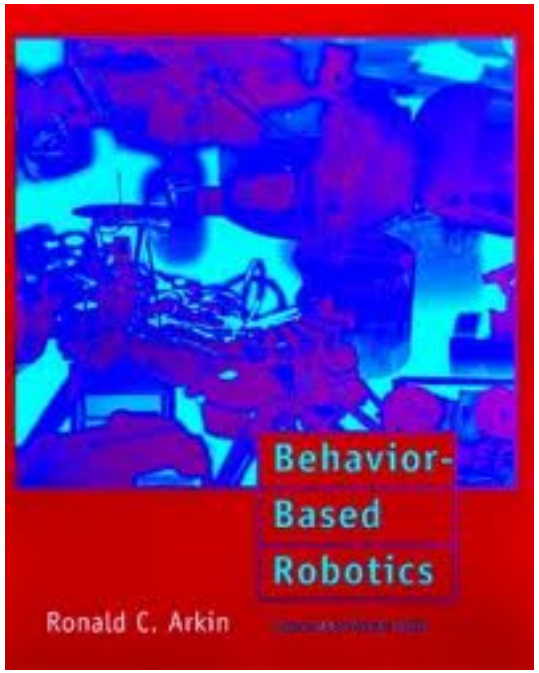
$$x_1 = x + d_1 \cos(\phi_1 + \phi)$$
$$y_1 = y + d_1 \sin(\phi_1 + \phi)$$

Example: Rendezvous

Example: Rendezvous



Lecture 2.5 – Behavior-Based Robotics

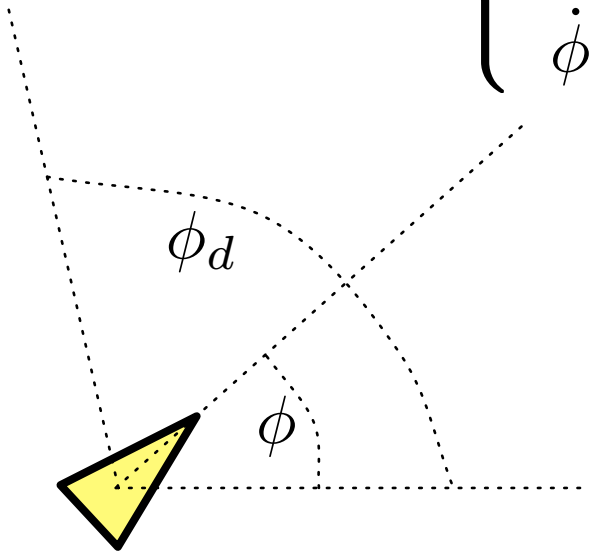


- The world is fundamentally unknown and changing
- Does not make sense to over-plan
- Key idea: Develop a library of useful controllers (=behaviors)
- Switch among controllers in response to environmental changes

Building a Behavior v.1

- Assume we have a differential-drive, wheeled mobile robot driving at a constant speed

$$\begin{cases} \dot{x} = v_0 \cos\phi \\ \dot{y} = v_0 \sin\phi \\ \dot{\phi} = \omega \end{cases}$$



- Want to drive in a desired heading

$$\omega = ???$$

Building a Behavior v.1

- We have a reference, a model, a control input, and a tracking error:

$$r = \phi_d, \quad e = \phi_d - \phi, \quad \dot{\phi} = \omega$$

- We not use PID?

$$\omega = K_P e + K_I \int e d\tau + K_D \dot{e}$$

Dealing with Angles

- This typically will not work since we are dealing with angles:

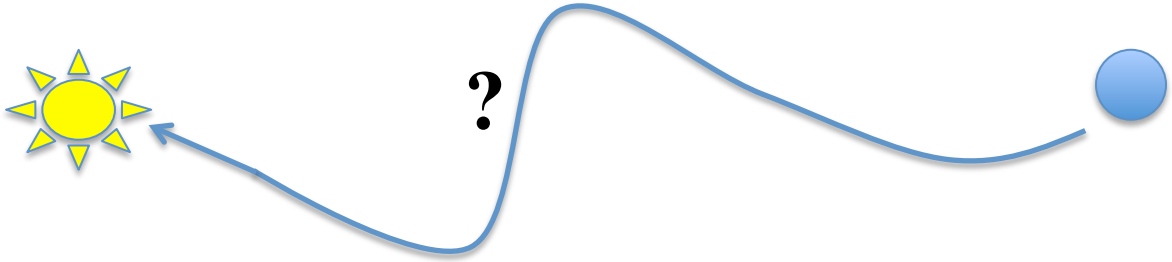
$$\phi_d = 0, \quad \phi = 100\pi \quad \Rightarrow \quad e = -100\pi$$

- Solution: Ensure that $e \in [-\pi, \pi]$
- Standard trick is to use `atan2`

$$e' = \text{atan2}(\sin(e), \cos(e)) \in [-\pi, \pi]$$

Example: Navigation

- Problem: Go to a goal location without bumping in to obstacles:



- At a minimum, we need two behaviors: Go-to-goal and Avoid-obstacles

Lecture 2.6 – Go-To-Goal

- How drive a robot to a goal location?

$$\begin{cases} \dot{x} = v_0 \cos\phi \\ \dot{y} = v_0 \sin\phi \\ \dot{\phi} = \omega \end{cases} \qquad e = \phi_d - \phi, \quad \omega = \text{PID}(e)$$

(x_g, y_g)

$$\phi_d = \arctan\left(\frac{y_g - y}{x_g - x}\right)$$

(x, y)

Attempt 1

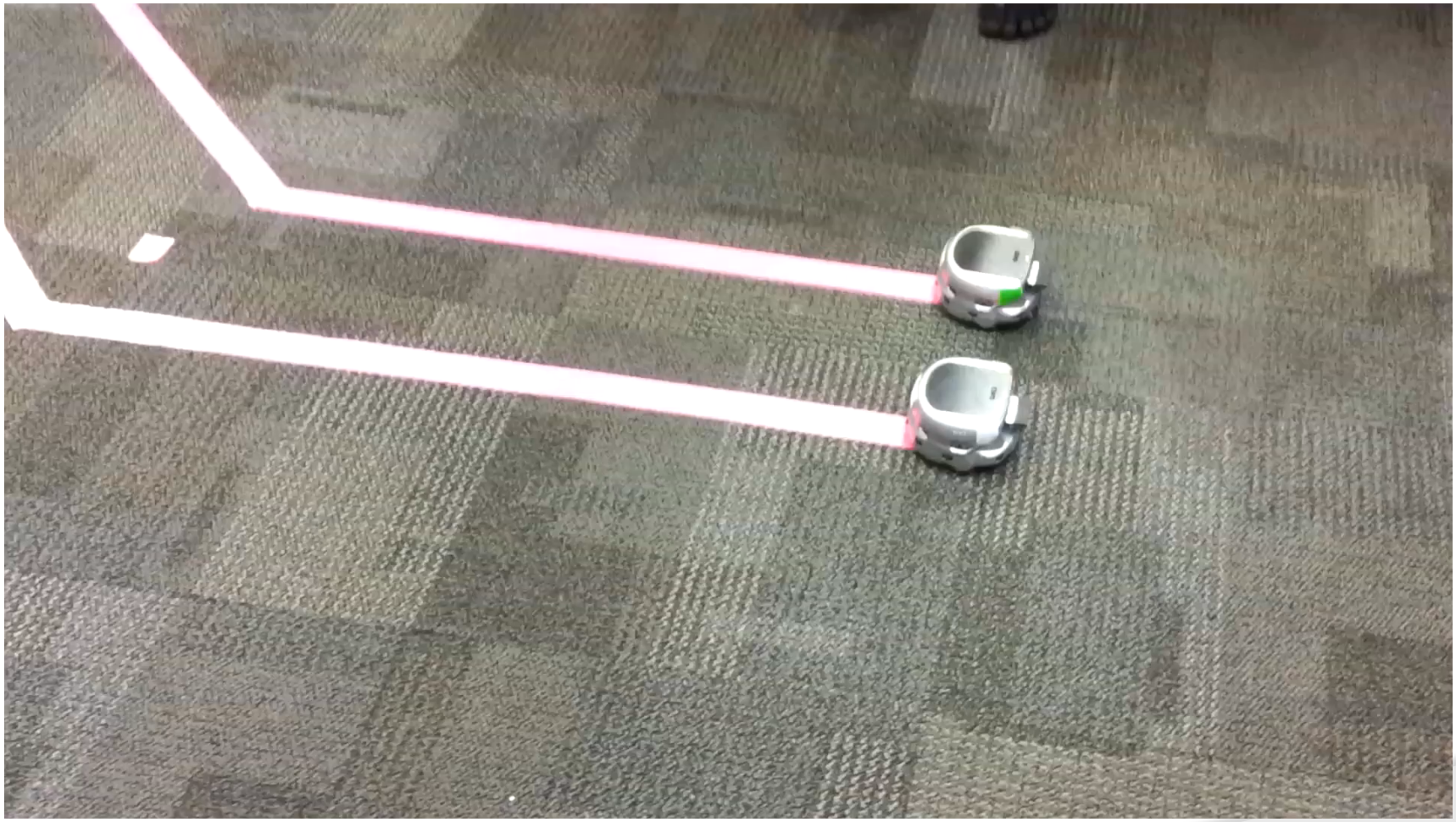
$$\omega = K(\phi_d - \phi)$$

Attempt 2

$$\text{“}\omega = K(\phi_d - \phi)\text{”}$$

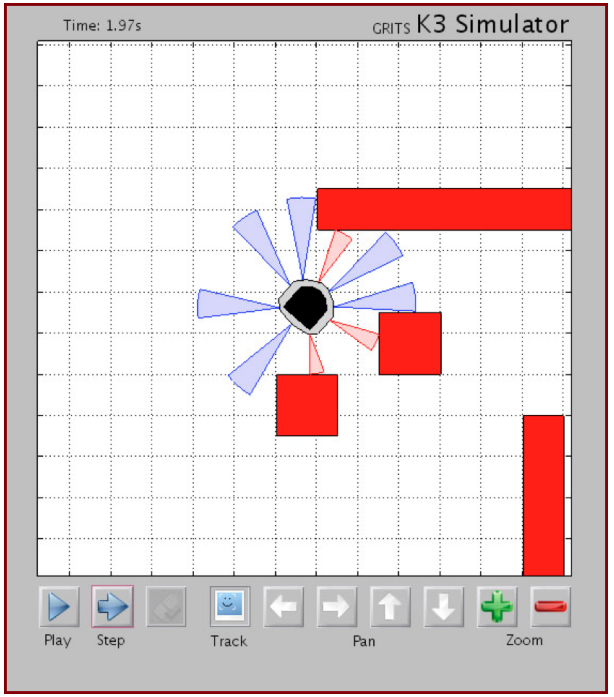
Attempt 3

"$\omega = K_{\text{big}}(\phi_d - \phi)$"



Lecture 2.7 – The GRITS Robot Simulator

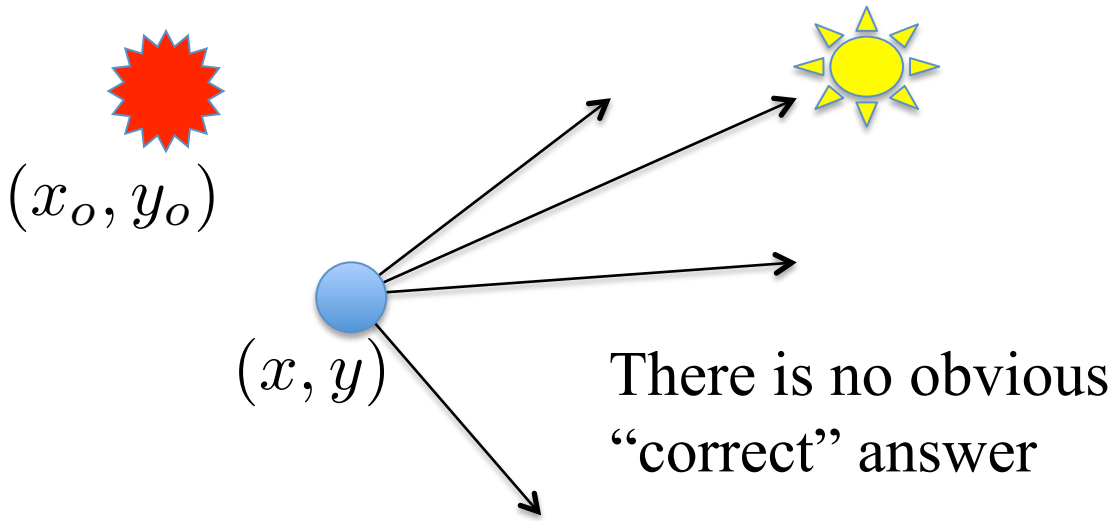
- A MATLAB-based simulator
- Simulates Khepera III differential drive mobile robot(-s) with IR range-sensors and wheel-encoders in cluttered environments



- Available at
  http://gritslab.gatech.edu/projects/robot-simulator/
  as stand-alone executable and MATLAB package

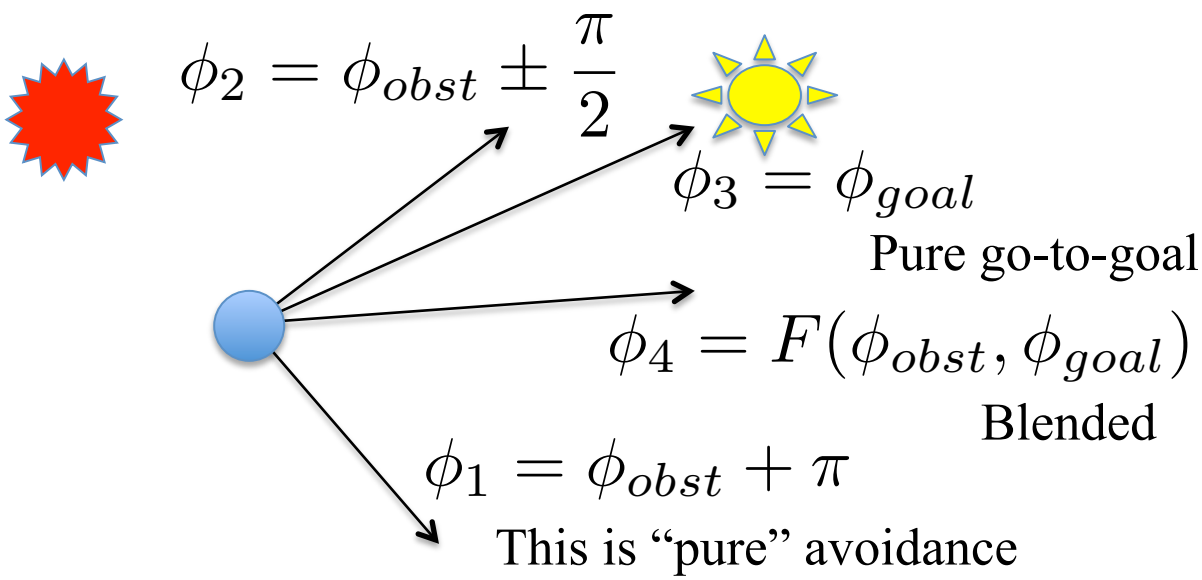
Lecture 2.8 – Obstacle-Avoidance

- How avoid driving into obstacles?
- We can use the same idea by defining a desired heading



Where to Go?

Choice depends on direction to goal.
Not "pure" but "blended"



Arbitration Mechanisms

- This example illustrate two fundamentally different "arbitration mechanisms"
  - Winner takes all = Hard switches
  - Blending = Combined behaviors

- Both approaches have merit in different situations
  - Performance?
  - Analysis?
- We will see how to design systematic behaviors and arbitration mechanisms

Module 2: Mobile Robots

- Module 2: Mobile robots
- Module 3: *Enough chit-chat!* Let's start over in a more systematic and formal manner!
  - **Linear Systems**