



**Dr. Magnus Egerstedt**  
Professor  
School of Electrical and  
Computer Engineering

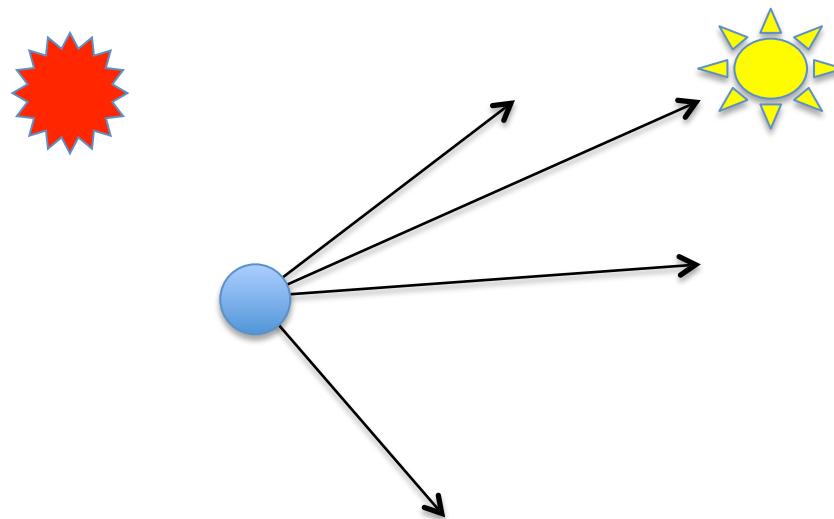
# Control of Mobile Robots

## Module 6 The Navigation Problem

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

# Lecture 6.1 – Behaviors Revisited

- Let's return to the navigation problem using behaviors
- Use control theory to describe what is going on



## Start Simple: The Model

- For the purpose of “planning”, assume that



$$\dot{x} = u, \quad x \in \mathbb{R}^2$$

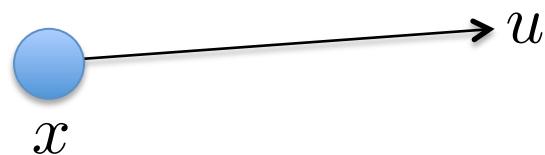
$$\dot{x} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u$$

$$\Gamma = [ \begin{array}{cc} B & AB \end{array} ] = [ \begin{array}{cc} I & 0 \end{array} ], \quad \text{rank}(\Gamma) = 2$$

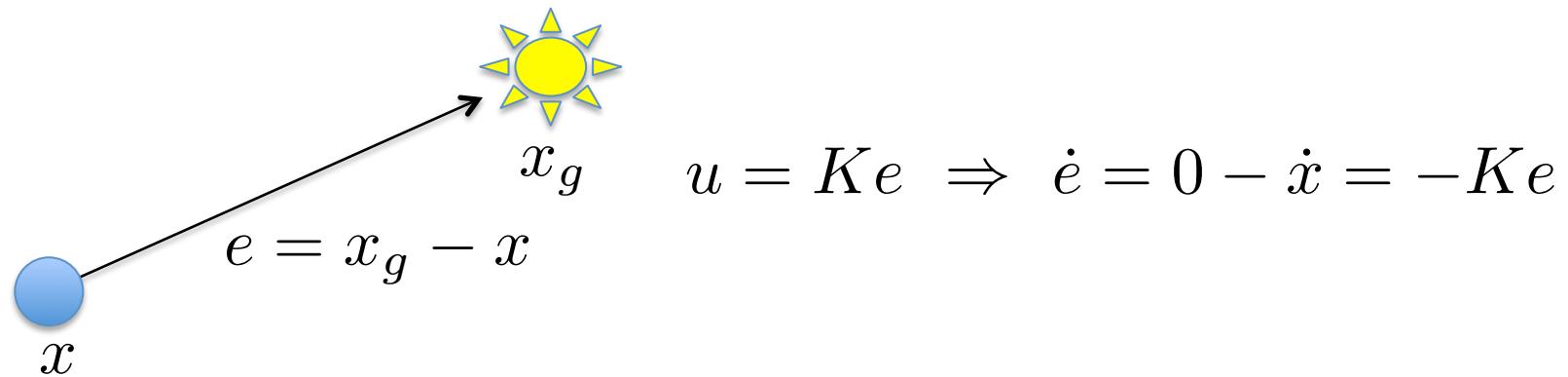
CC!

# The Dynamic Duo

- The two bread-and-butter behaviors are go-to-goal and avoid-obstacle
  - Go-to-goal: Drives the robot towards the goal
  - Avoid-obstacle: Don't slam into things
- Control design task: pick a desired motion vector and set that equal to the input  $u$



## Go-To-Goal

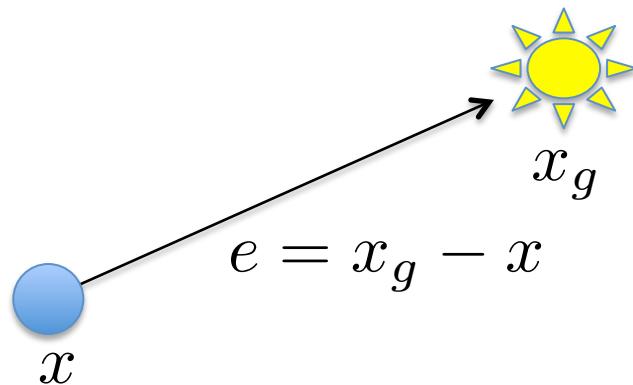


Asymptotically stable?

$K > 0$  or  $K \succ 0$  ( $\text{eig}[K] > 0$ )  
 $e \rightarrow 0$

# A Concern

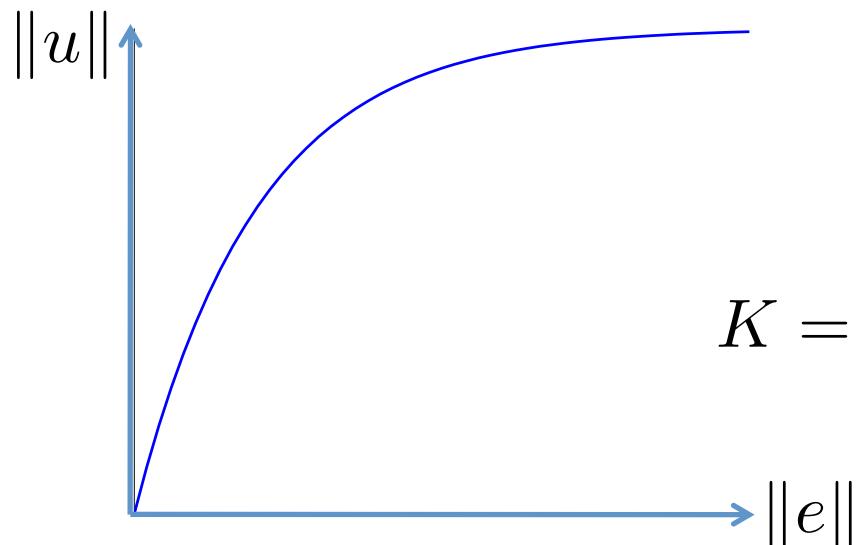
- A linear controller means the robot goes faster the further away the goal is.
- Solution, in practice, make the gain K a function of e



$$\dot{e} = -K(\|e\|)e$$

# A Concern

- A linear controller means the robot goes faster the further away the goal is.
- Solution, in practice, make the gain K a function of  $e$

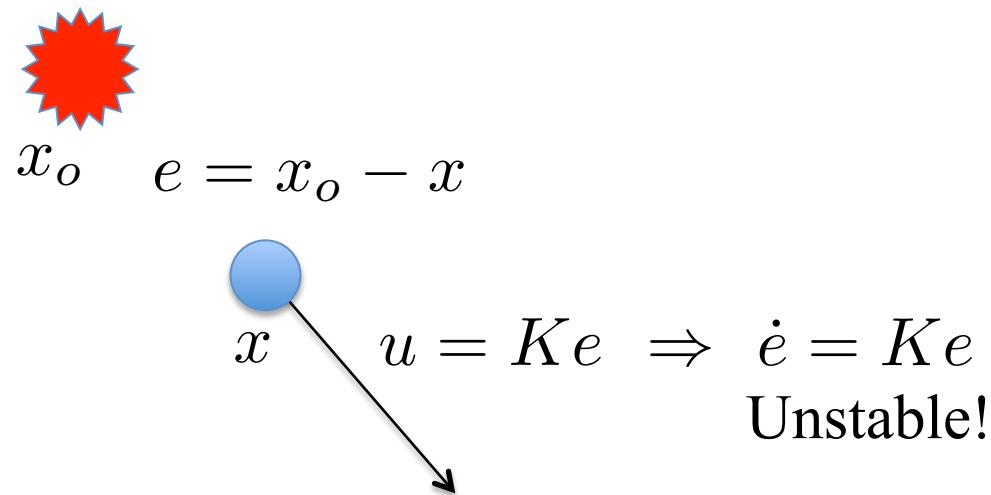


$$\dot{e} = -K(\|e\|)e$$

$$K = \frac{v_0 \left(1 - e^{-\alpha \|e\|^2}\right)}{\|e\|}$$

## Avoid-Obstacle

- To get the direction, just flip the sign in the go-to-goal behavior

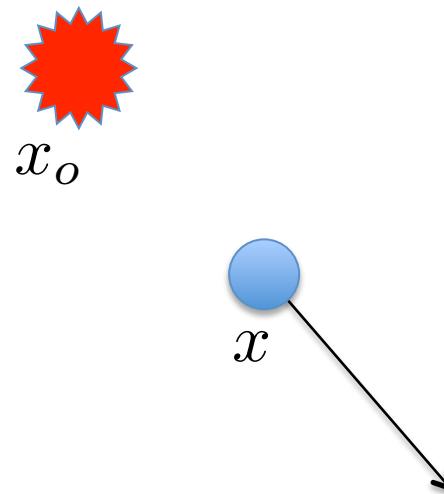


## More Concerns

- The robot drives off to infinity?
- It is overly cautious?
- We care less the closer we get?

“Solutions”

- Make K dependent on e!
- Switch between behaviors!
- Use the induced mode!

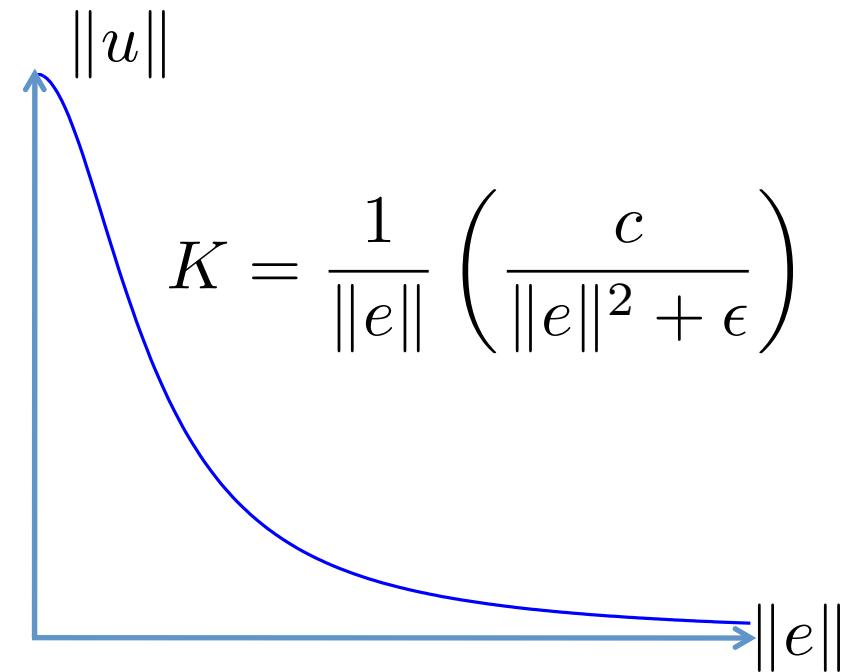


# More Concerns

- The robot drives off to infinity?
- It is overly cautious?
- We care less the closer we get?

“Solutions”

- Make K dependent on e!
- Switch between behaviors!
- Use the induced mode!

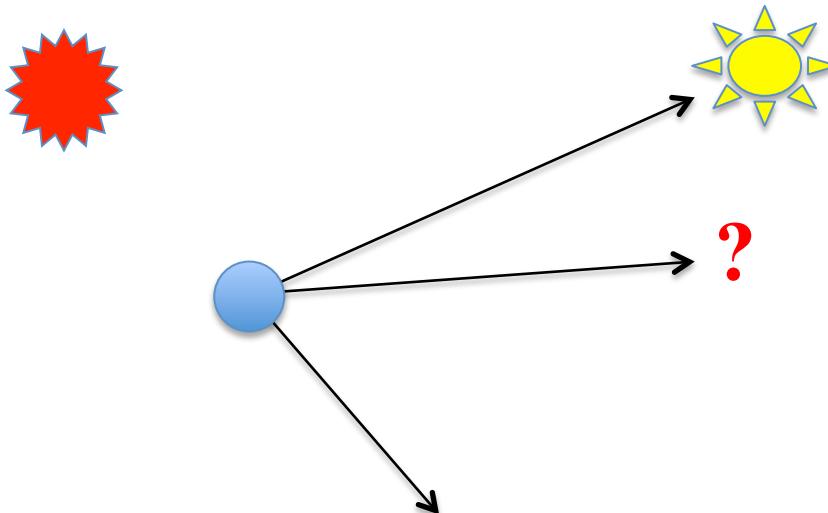


# Behavior/Mode Transitions?

- What's missing?
  - Couple actual robot dynamics to the behaviors (next module)
  - Figure out the mode transitions (next lecture)
- But first, let's try it for real!

## Lecture 6.2 –Hard Switches vs Blending

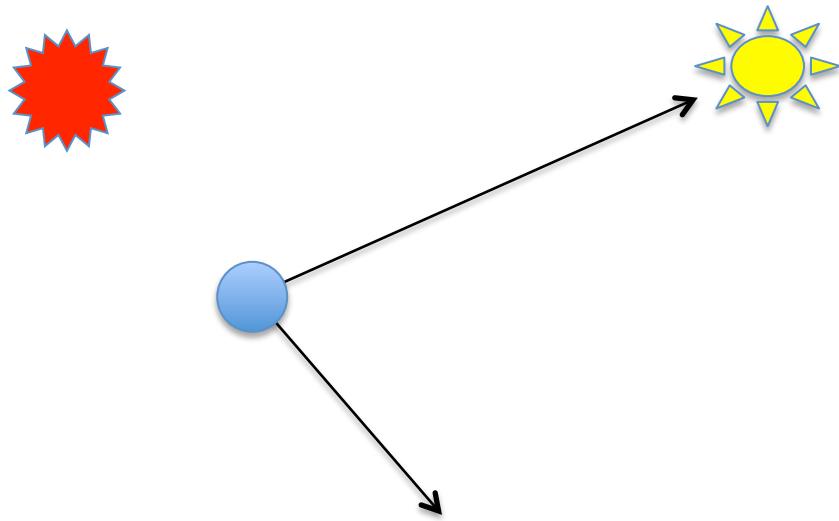
- Given two different behaviors, how should they be combined?



- Two options:
  - Hard Switches
  - Blended Behaviors

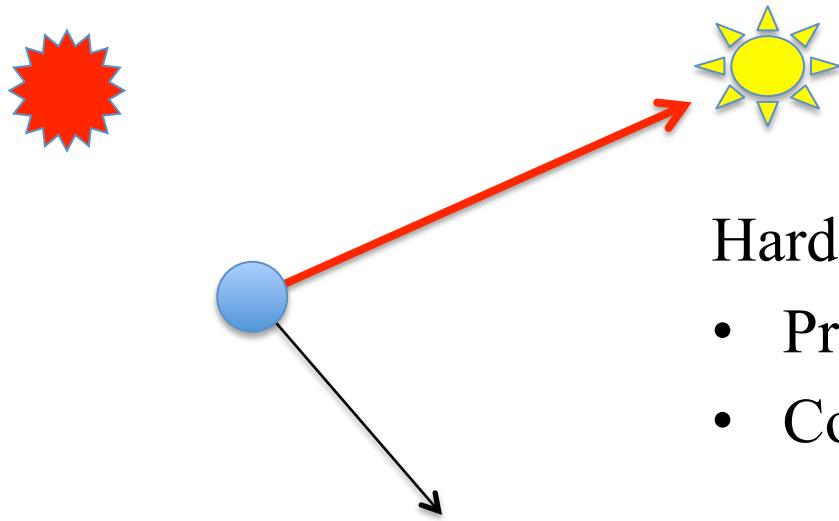
# Pros and Cons

- Philosophical Question: Can robots chew gum and walk at the same time?



## Pros and Cons

- Philosophical Question: Can robots chew gum and walk at the same time?

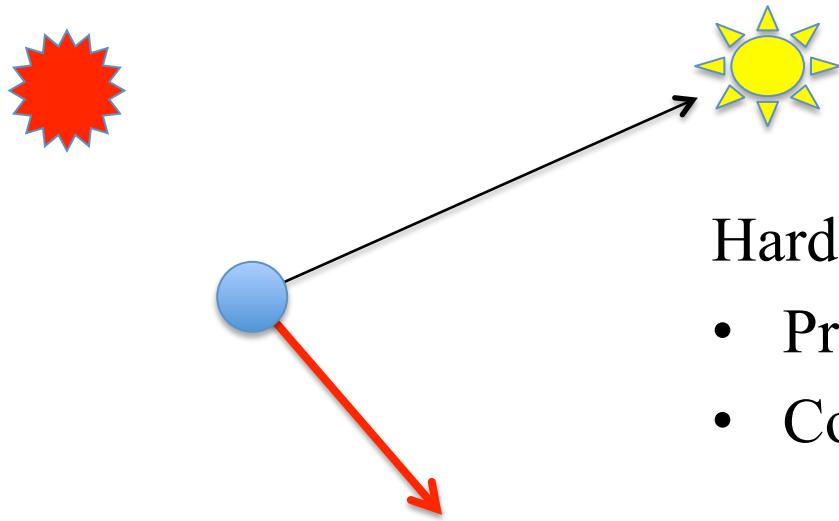


### Hard Switches:

- Pro: Performance guarantees
- Con: Bumpy ride, Zeno?

## Pros and Cons

- Philosophical Question: Can robots chew gum and walk at the same time?

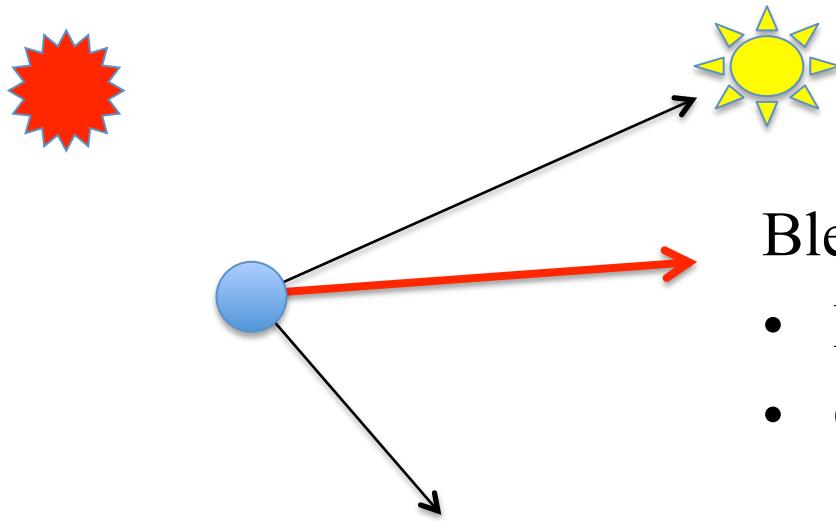


### Hard Switches:

- Pro: Performance guarantees
- Con: Bumpy ride, Zeno?

# Pros and Cons

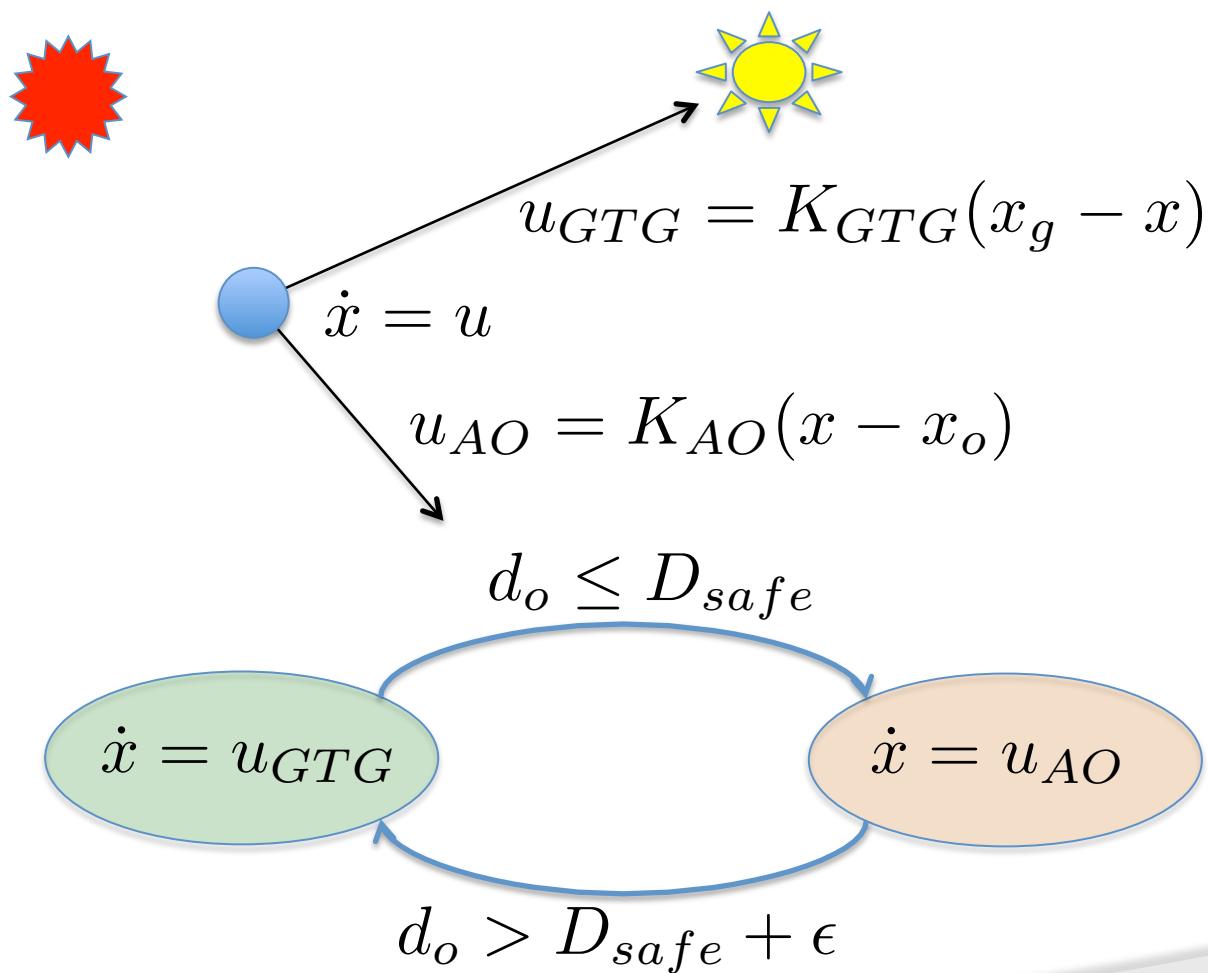
- Philosophical Question: Can robots chew gum and walk at the same time?



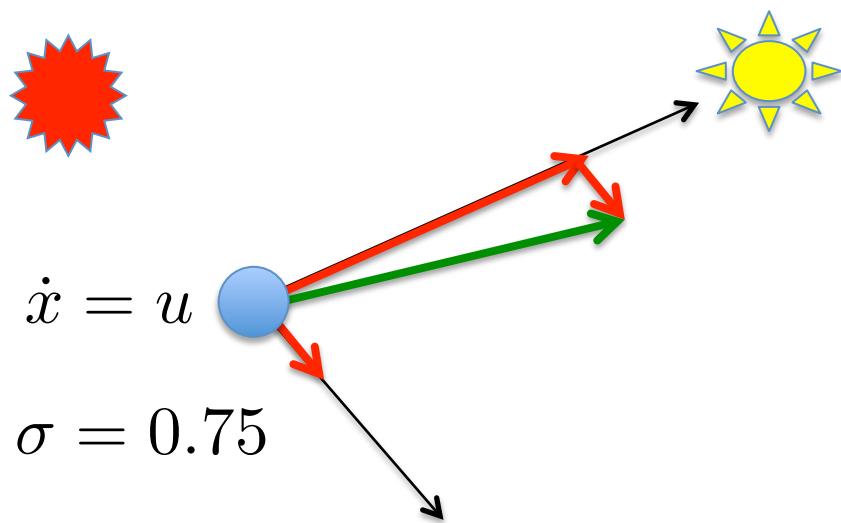
Blending:

- Pro: Smooth ride
- Con: No guarantees

# A Switch HA



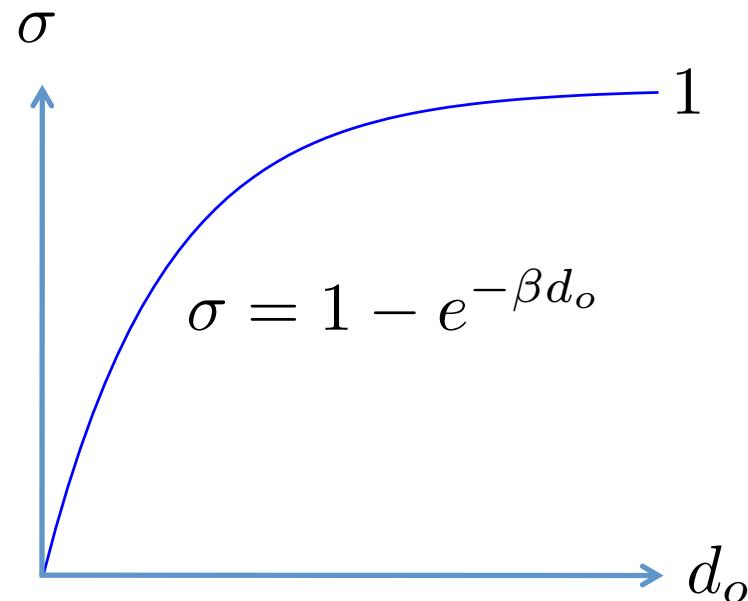
# Blending



Blending function  
 $\sigma(d_o) \in [0, 1]$

$$\dot{x} = \sigma(d_o)u_{GTG} + (1 - \sigma(d_o))u_{AO}$$

# Example



# Punchlines

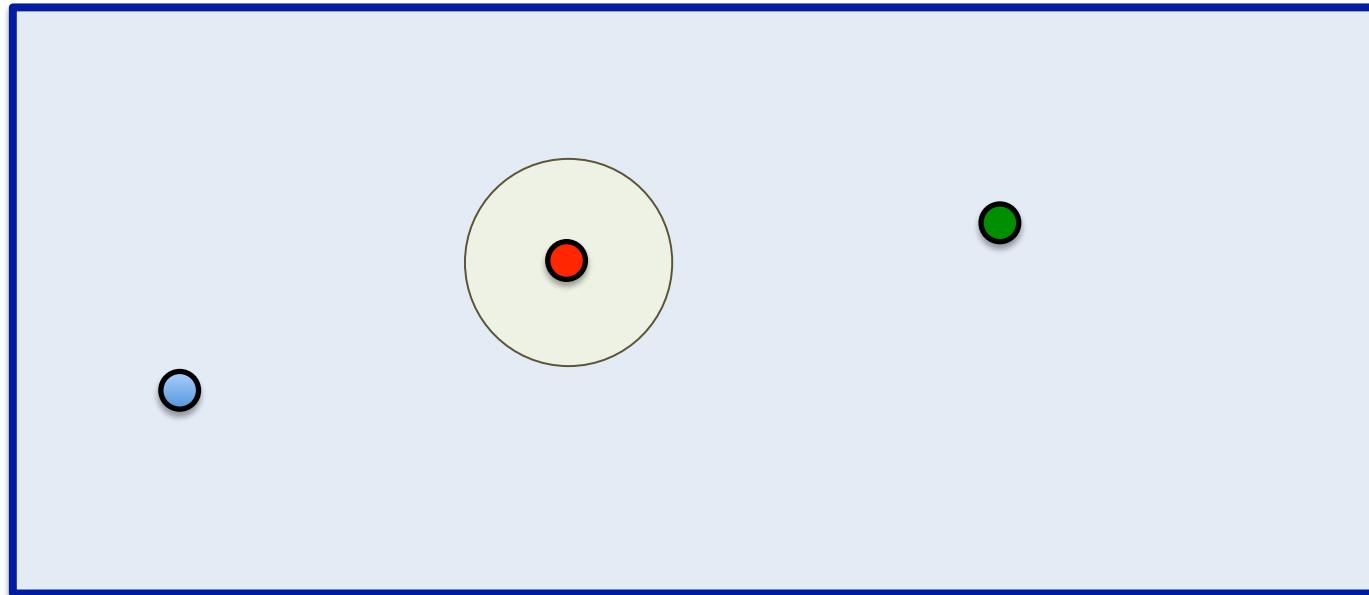
- Two choices – both of them come with pros and cons
- Would like to have our cake and eat it (smooth ride + performance guarantees)
- Enter: the induced mode

# Lecture 6.3 –Convex and Non-Convex Worlds

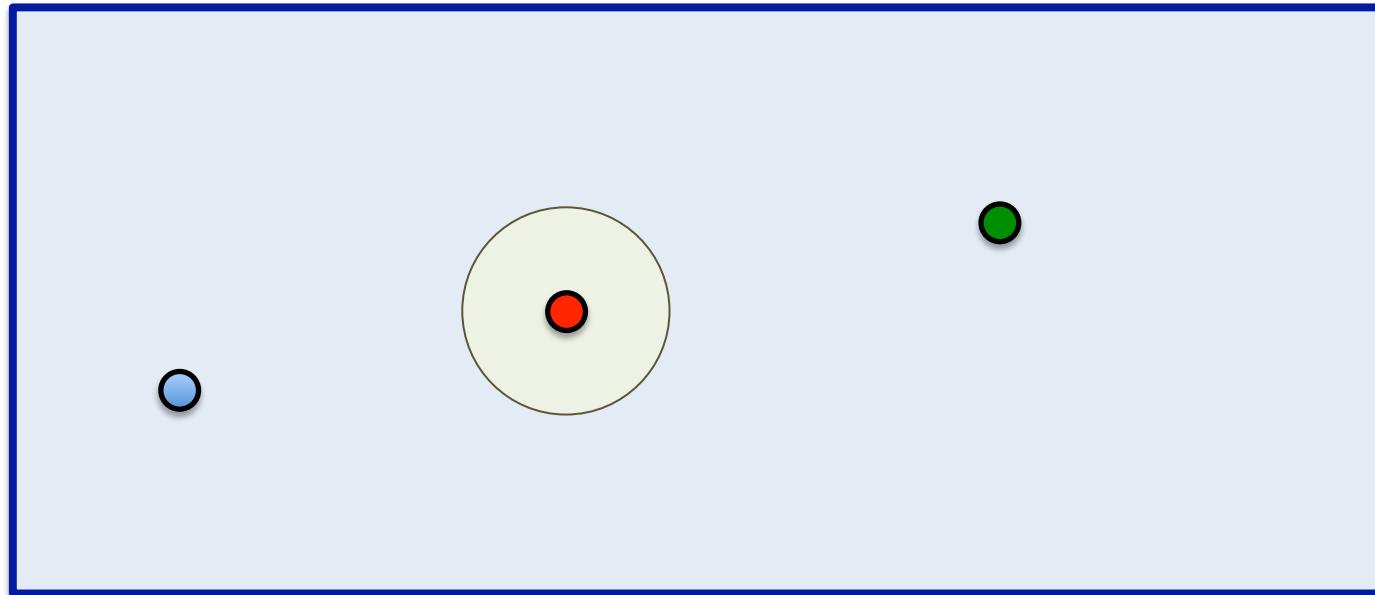
- Let's see when our two basic behaviors work and when we need more behaviors!
- We will start simple and add complexity
  - Point-worlds
  - Circular obstacles
  - Convex worlds
  - Non-convex obstacles
  - Labyrinths



# Point-Worlds

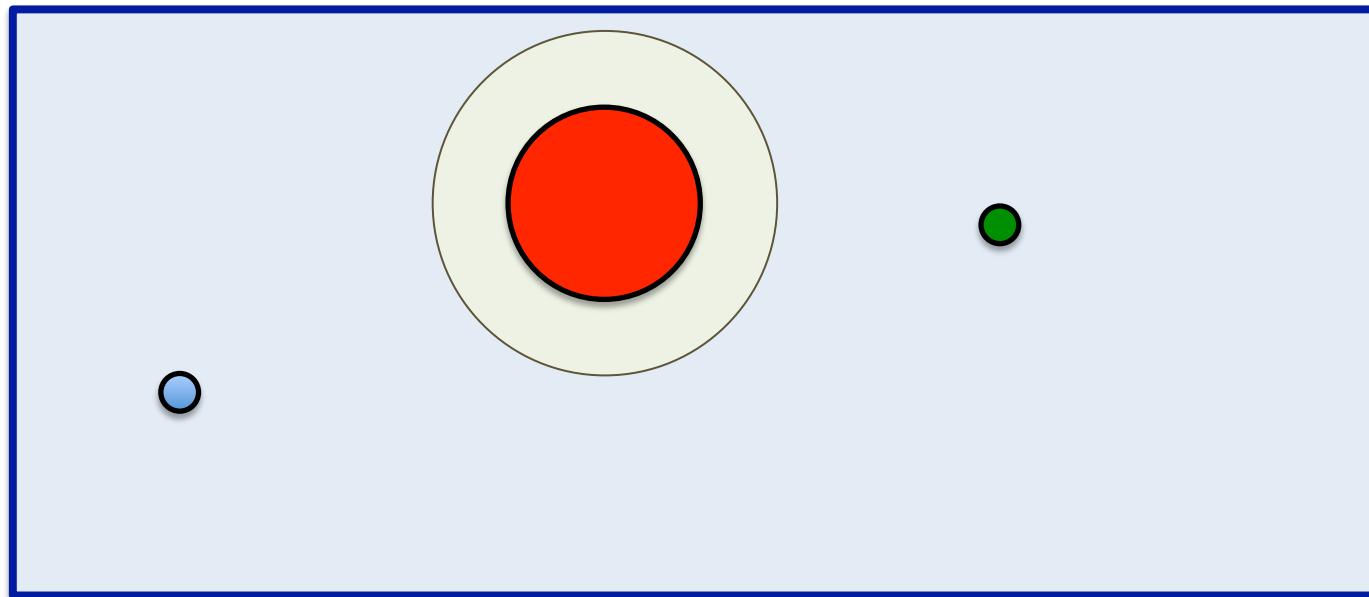


# Point-Worlds



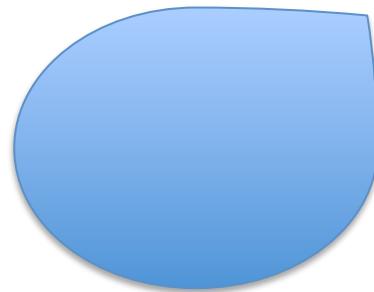
- Here the two behaviors are sufficient unless we are super-unlucky. (Maybe some “mild” Zeno...)
- Solution: Add a little bit of noise – in practice this is never really needed since the world is noisy

# Circular Obstacles

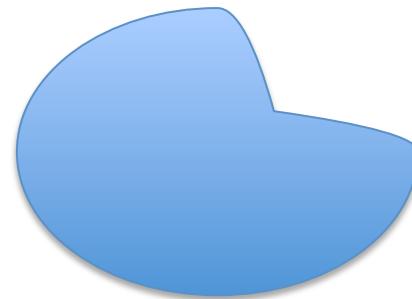


- This is the same as for point-obstacles

# Convex Obstacles



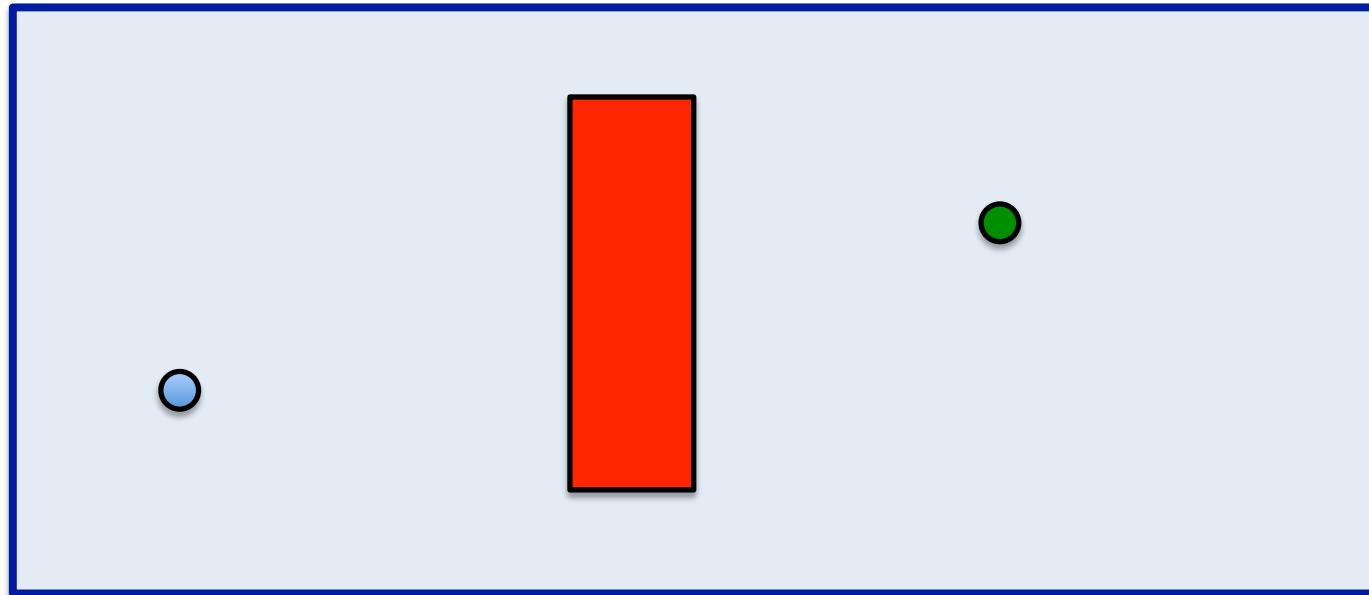
convex



non-convex

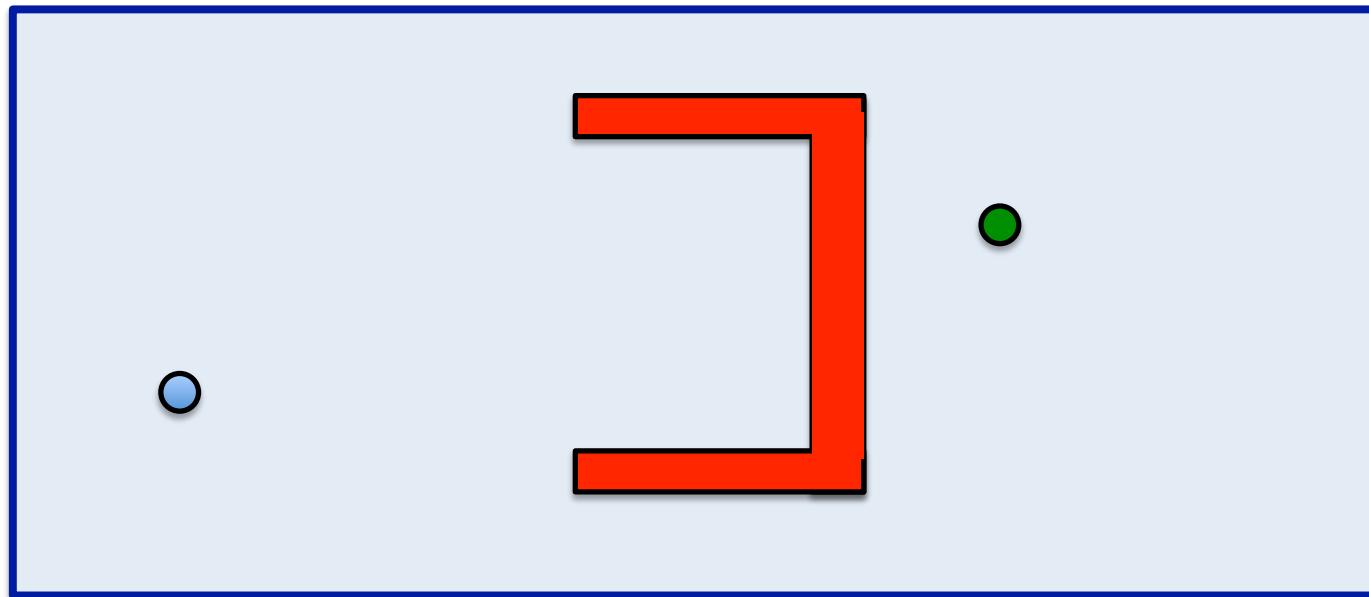
- A set is convex if every line in-between two points in the set lies in the set

# Convex Obstacles



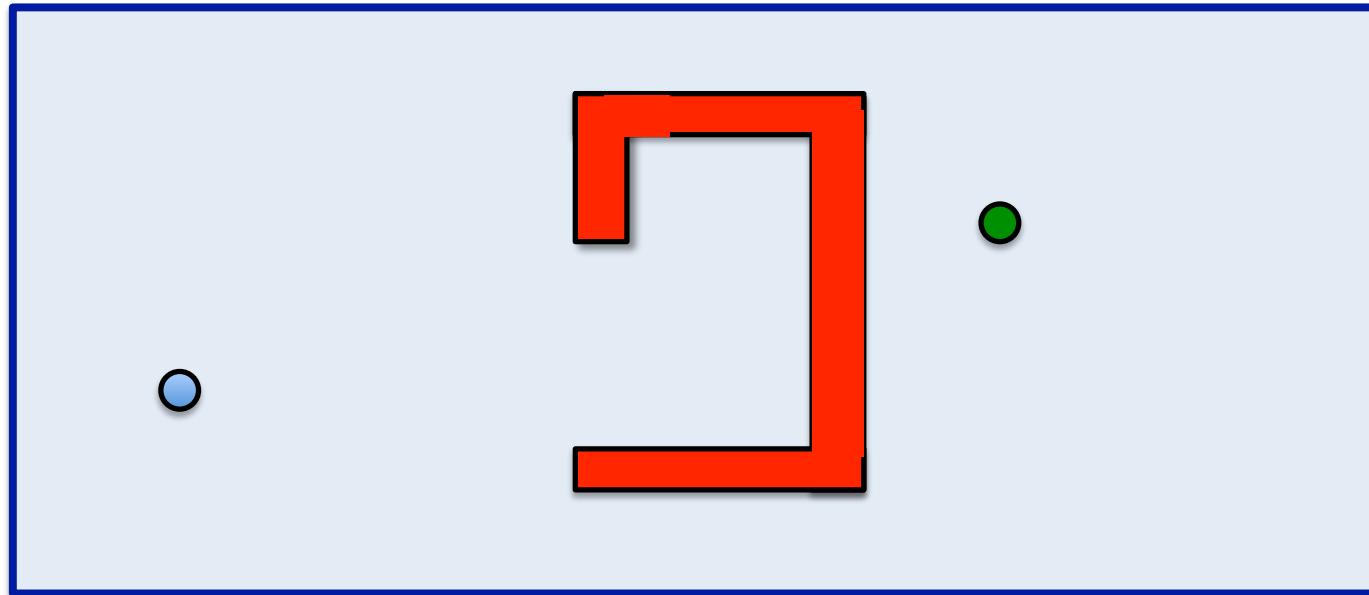
- Even though the obstacles are quite nice-looking, we can get stuck for real with only these two behaviors
- We need some way to keep going along the boundary of the obstacle!

# Non-Convex Obstacles



- Even worse!

# Labyrinths



- Forget about it!

# Next Lecture

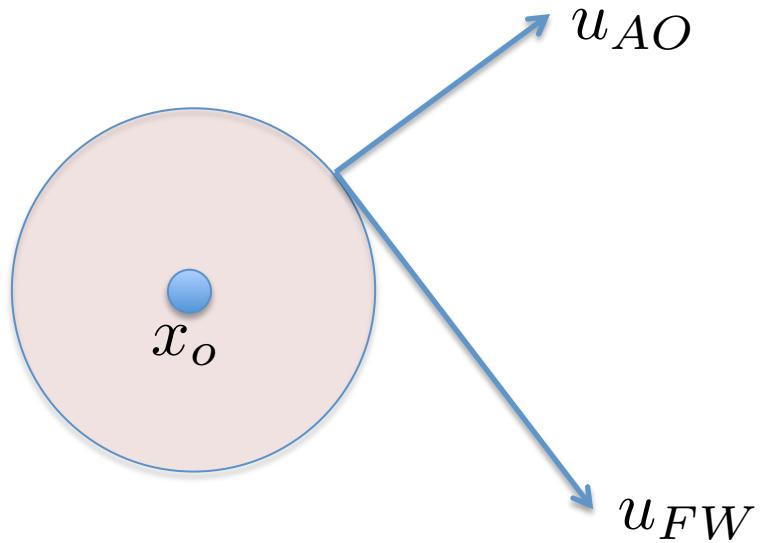
- We really need at least one more behavior to be able to deal with non-circular worlds!

# Lecture 6.4 –Wall Following

- We need some way of negotiating complex environments
- A really useful behavior is one that makes the robot follow the boundary of an obstacle/wall
- This will also allow us to introduce the induced mode in a systematic way

# How To Follow Walls

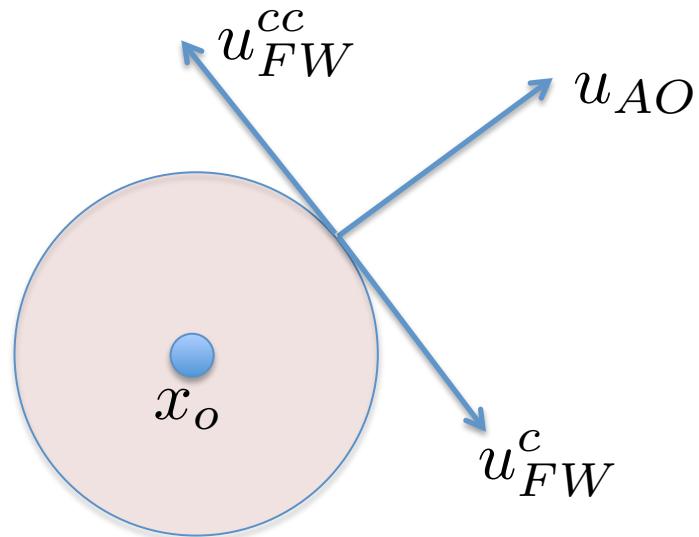
- The follow-wall behavior should maintain a constant distance to the obstacle/wall it is following



$$u_{FW} = \alpha \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} u_{AO} = \alpha R(-\pi/2) u_{AO}$$

# Clockwise or Counter-Clockwise?

- We can clearly move in two different directions along a wall!



# Clockwise or Counter-Clockwise?

- We can clearly move in two different directions along a wall!

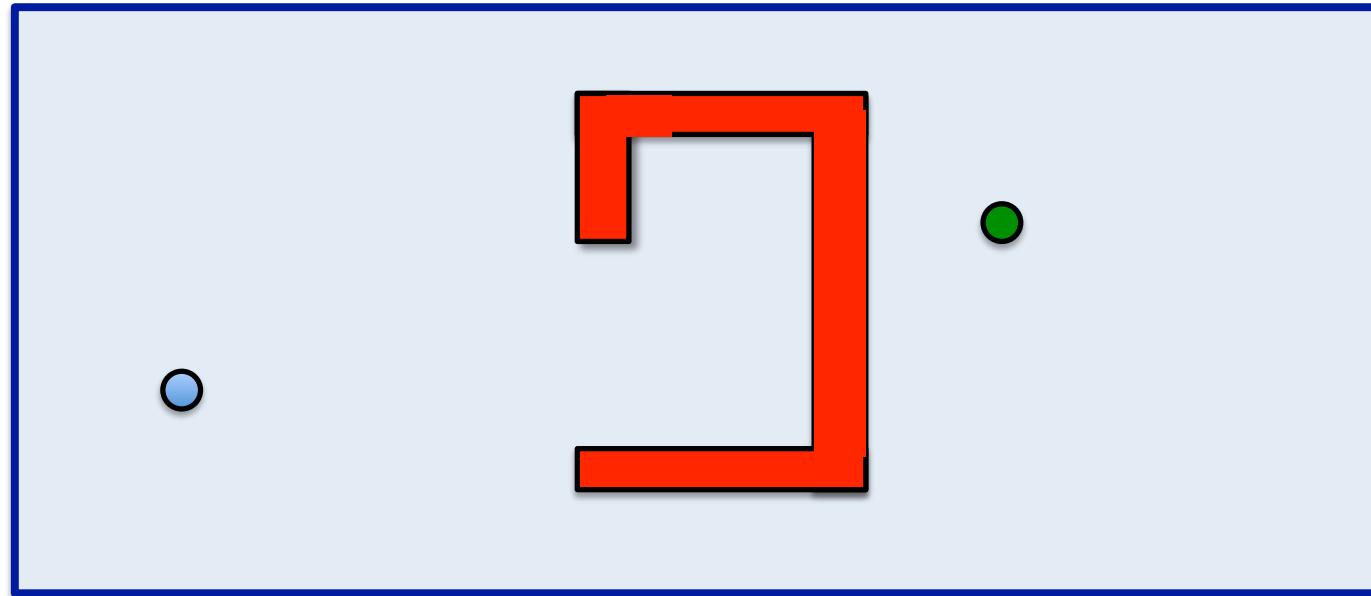
Rotation Matrix:  $R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$

$$u_{FW}^c = \alpha R(-\pi/2) u_{AO} = \alpha \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} u_{AO}$$

$$u_{FW}^{cc} = \alpha R(\pi/2) u_{AO} = \alpha \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} u_{AO}$$

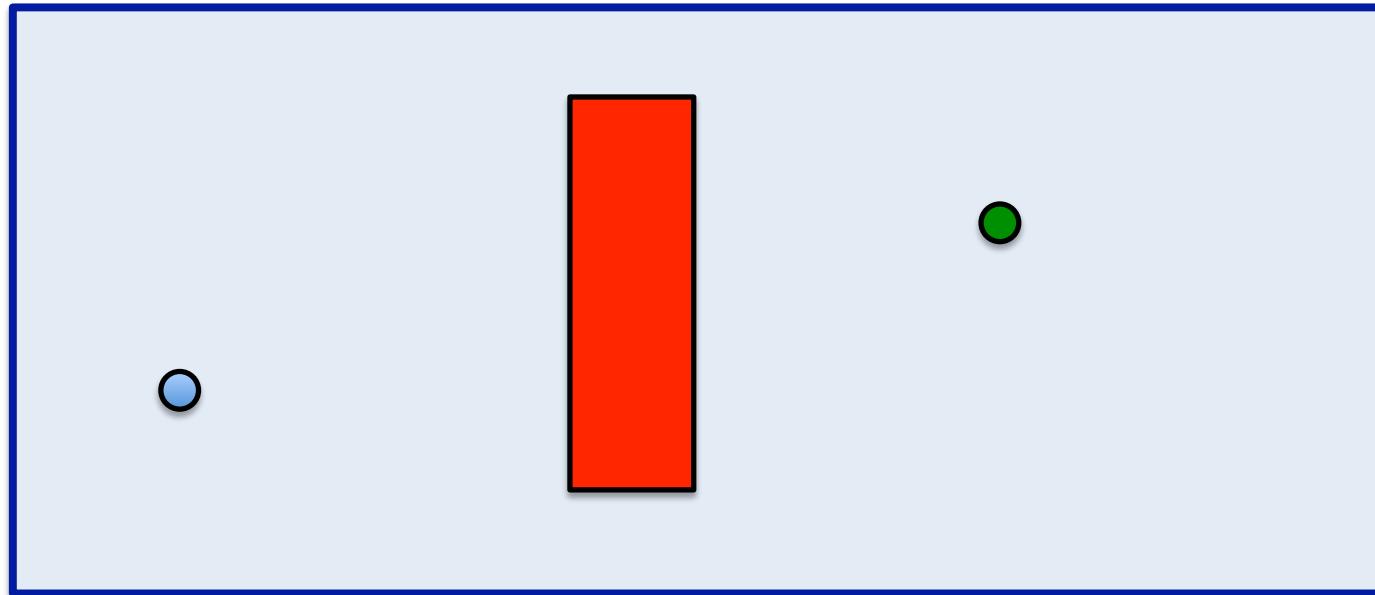
# Clockwise or Counter-Clockwise?

- Which direction to choose?



- There is no obvious answer to that!
- Maybe let the goal-to-goal direction determine?

# Clockwise or Counter-Clockwise?



$$\langle v, w \rangle = v^T w = \|v\| \|w\| \cos(\angle(v, w))$$

$$\langle u_{GTG}, u_{FW}^c \rangle > 0 \Rightarrow u_{FW}^c$$

$$\langle u_{GTG}, u_{FW}^{cc} \rangle > 0 \Rightarrow u_{FW}^{cc}$$

# Issues

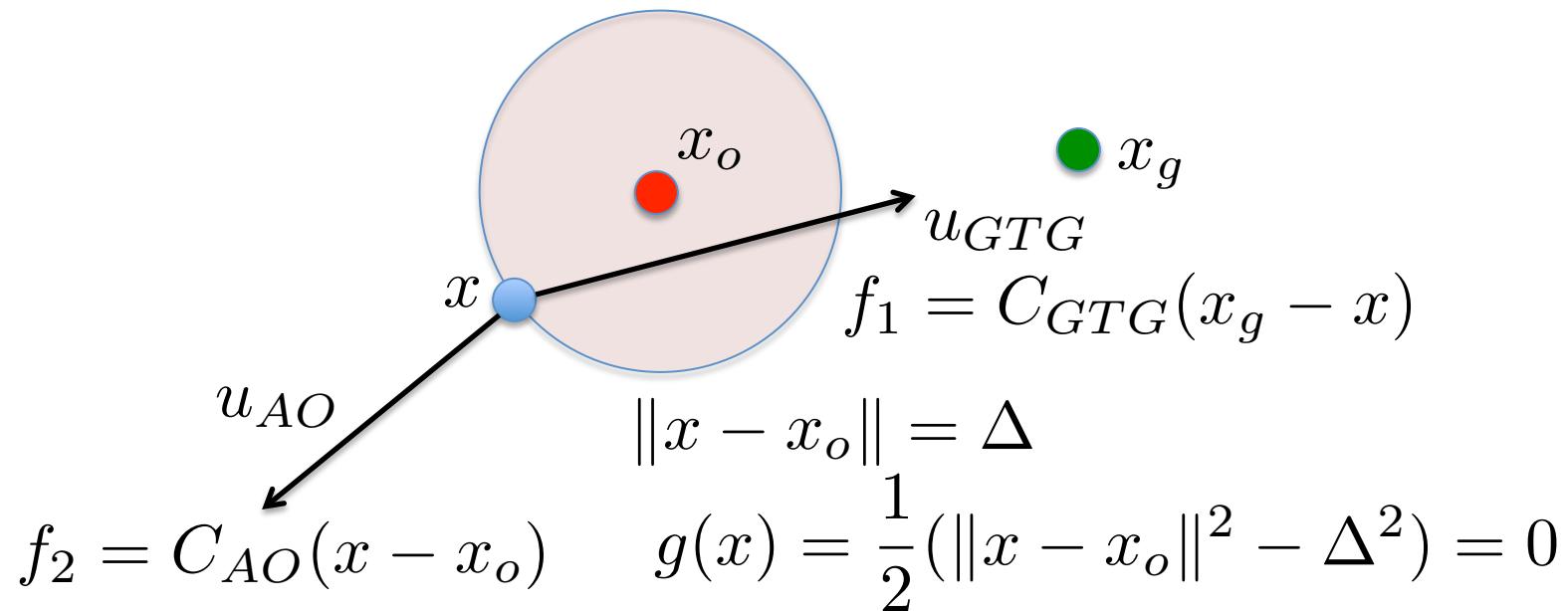
- When release the follow-wall behavior?
- Is there a systematic way to scale the behavior?

$$\alpha R(\pm\pi/2)u_{AO}$$

# Lecture 6.5 – The Induced Mode

- We have seen that we need wall-following to solve complex navigation problems
- Last lecture:  $\alpha R(\pm\pi/2)u_{AO}$
- This lecture: How can this be connected to the induced mode between go-to-goal and avoid-obstacle in a more systematic manner?

# The Setup



# Connecting to the Induced Mode

$$g(x) = \frac{1}{2}(\|x - x_o\|^2 - \Delta^2) = 0$$

$$f_1 = C_{GTG}(x_g - x)$$

$$f_2 = C_{AO}(x - x_o)$$

$$\frac{\partial g}{\partial x} = (x - x_o)^T$$

$$L_{f_2}g = \frac{\partial g}{\partial x} f_2 = (x - x_o)^T C_{AO}(x - x_o) = C_{AO}\|x - x_o\|^2$$

$$L_{f_1}g = \frac{\partial g}{\partial x} f_1 = C_{GTG}(x - x_o)^T(x_g - x)$$

The Induced Mode:

$$\dot{x} = \frac{1}{L_{f_2}g - L_{f_1}g} \left( L_{f_2}g f_1 - L_{f_1}g f_2 \right)$$

# Connecting to the Induced Mode

$$L_{f_1}g = C_{GTG}(x - x_o)^T(x_g - x)$$

$$L_{f_2}g = C_{AO}\|x - x_o\|^2$$

The Induced Mode:

$$\dot{x} = \frac{1}{L_{f_2}g - L_{f_1}g} \left( L_{f_2}gf_1 - L_{f_1}gf_2 \right)$$

Recall:

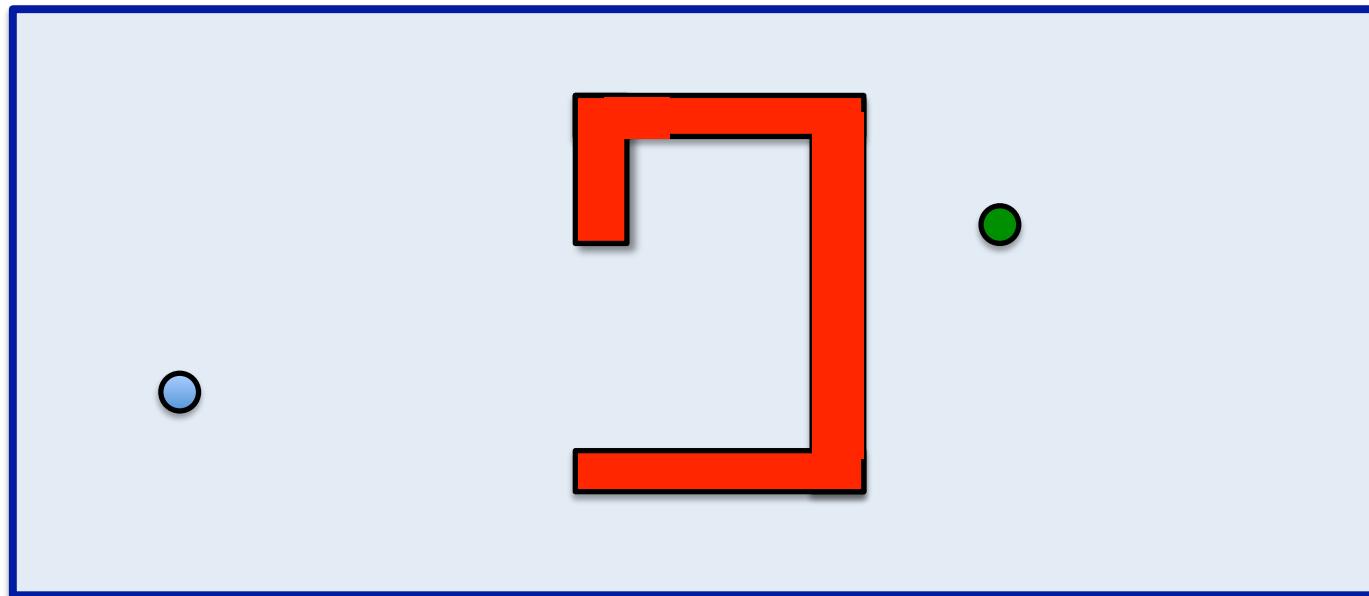
$$u_{FW}^c = \alpha R(-\pi/2)u_{AO}$$

$$u_{FW}^{cc} = \alpha R(\pi/2)u_{AO}$$

Quite a mess!

They are the same!!

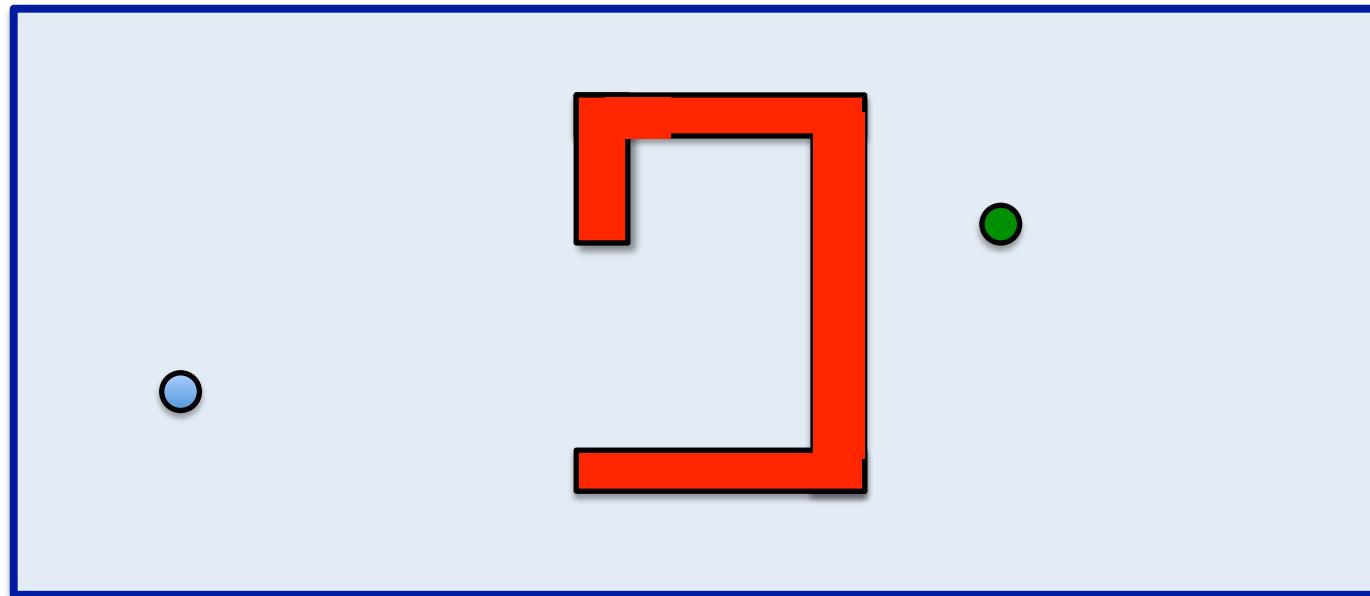
# When to Stop Sliding?



- Not so easy!
- Next time....

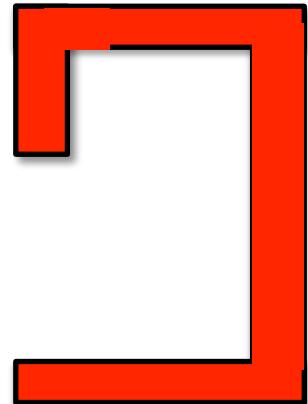
# Lecture 6.6 – A Complete Navigation System

- When do we stop following walls?
- Not when we stop sliding!



# When To Stop?

- We should stop when enough progress has been made and we have a “clear” shot to the goal!

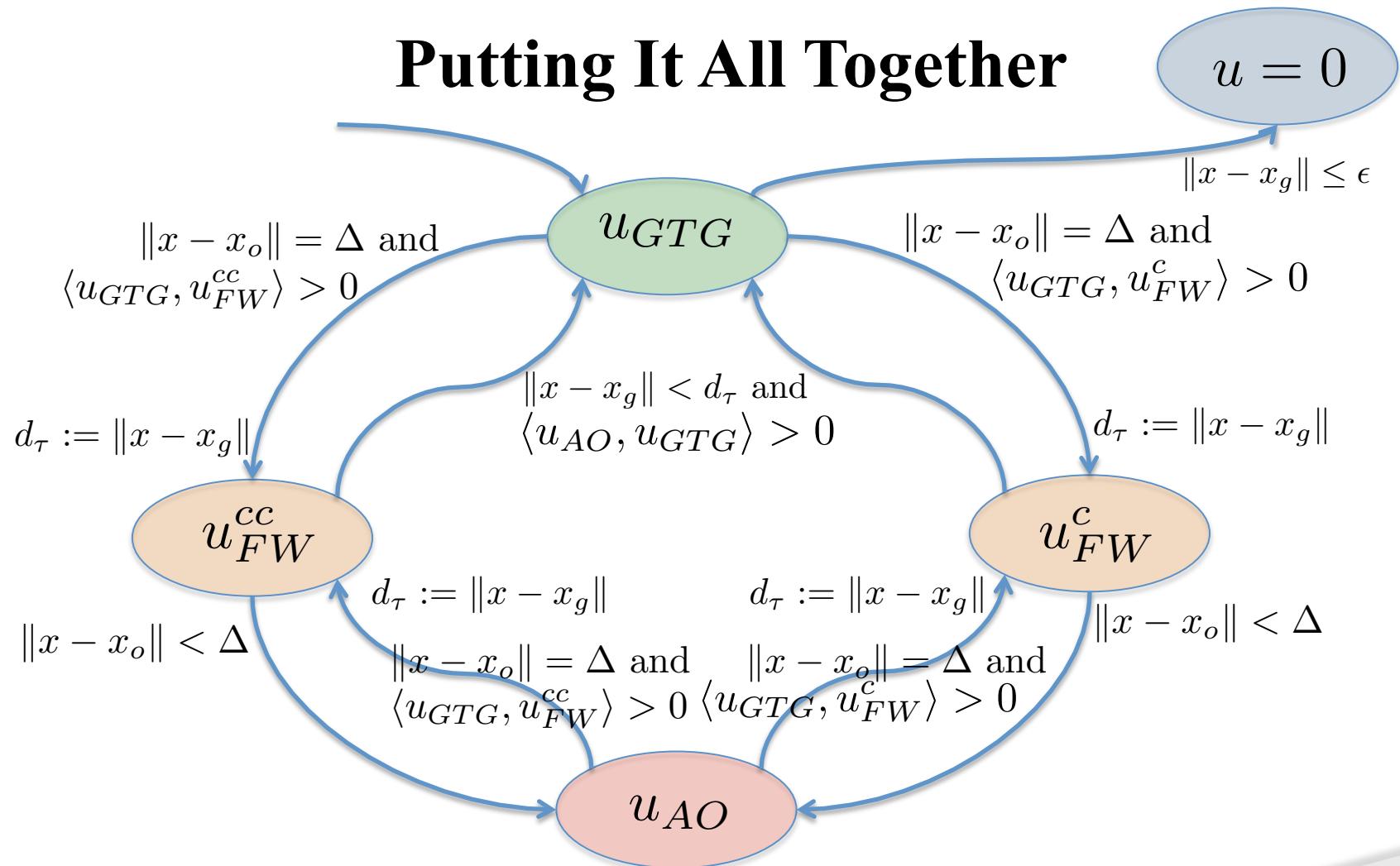


$\tau =$  time of last switch

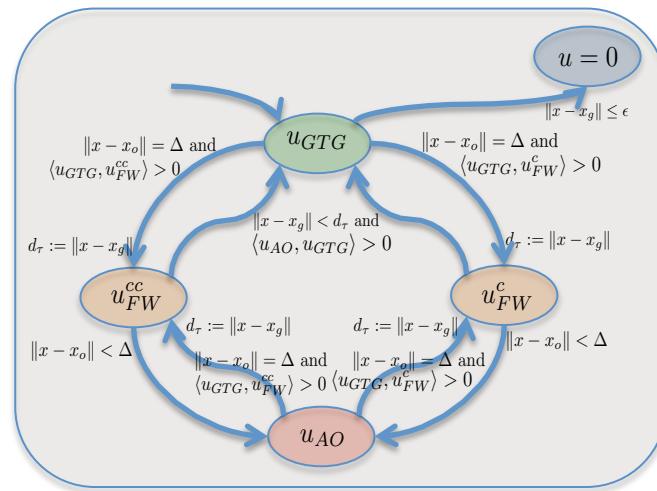
progress:  $\|x - x_g\| < \|x(\tau) - x_g\|$

clear shot:  $\langle u_{AO}, u_{GTG} \rangle > 0$

# Putting It All Together

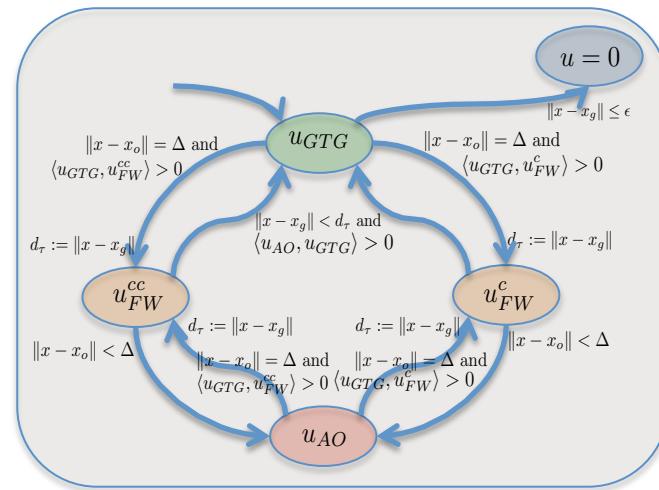


# Does This Work?



- Nope!
- Next time: Practical considerations!

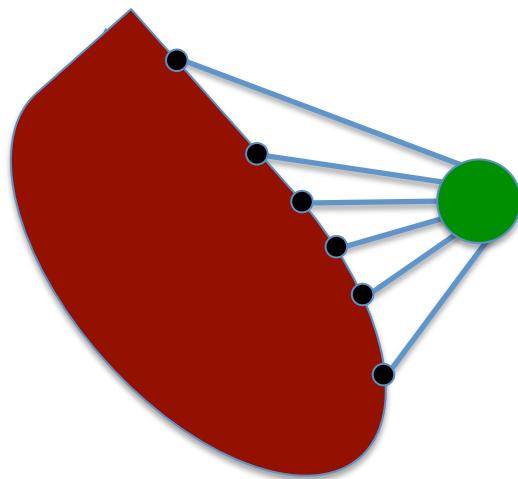
# Lecture 6.7 – Practical Considerations



- Non-Point-Obstacles
- Fat Guards
- Tweak, Tweak, Tweak

# Obstacles Aren't Points

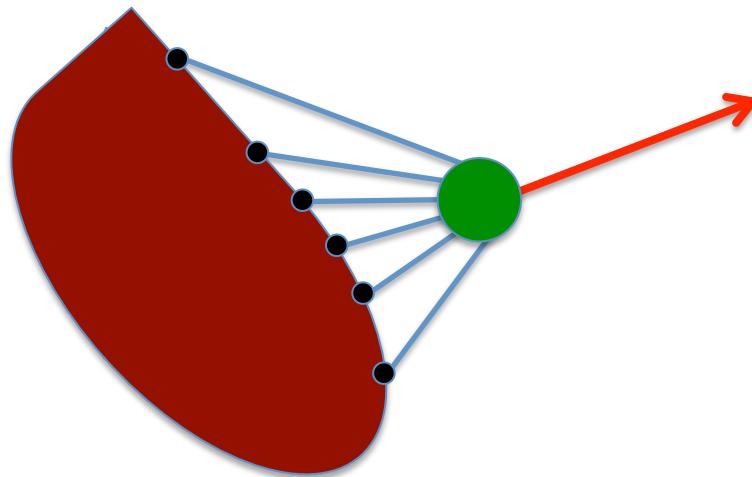
- We have to allow for non-point obstacles!
- But most (all) sensors really return points



How to deal with this situation?

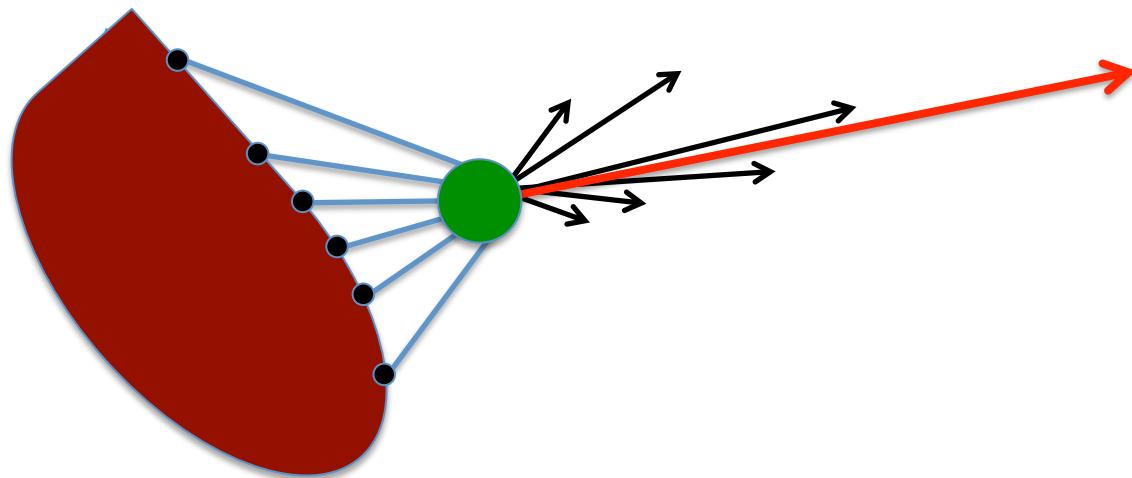
# Some Options

1. Go with the closest obstacle point (not bad)



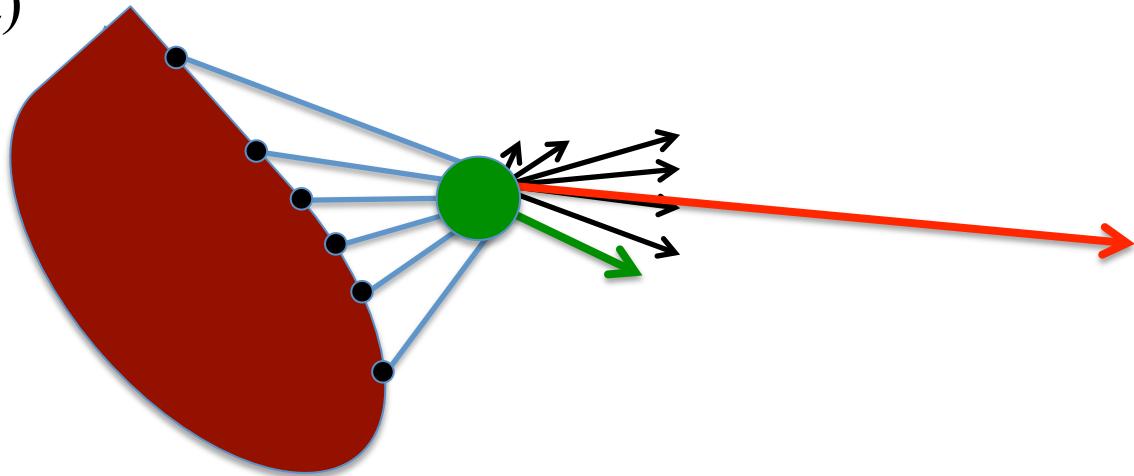
## Some Options

1. Go with the closest obstacle point (not bad)
2. Weigh and add the “obstacle vectors” depending on distance (better)



## Some Options

1. Go with the closest obstacle point (not bad)
2. Weigh and add the “obstacle vectors” depending on distance (better)
3. Weight and add depending on distance and direction of travel (best)



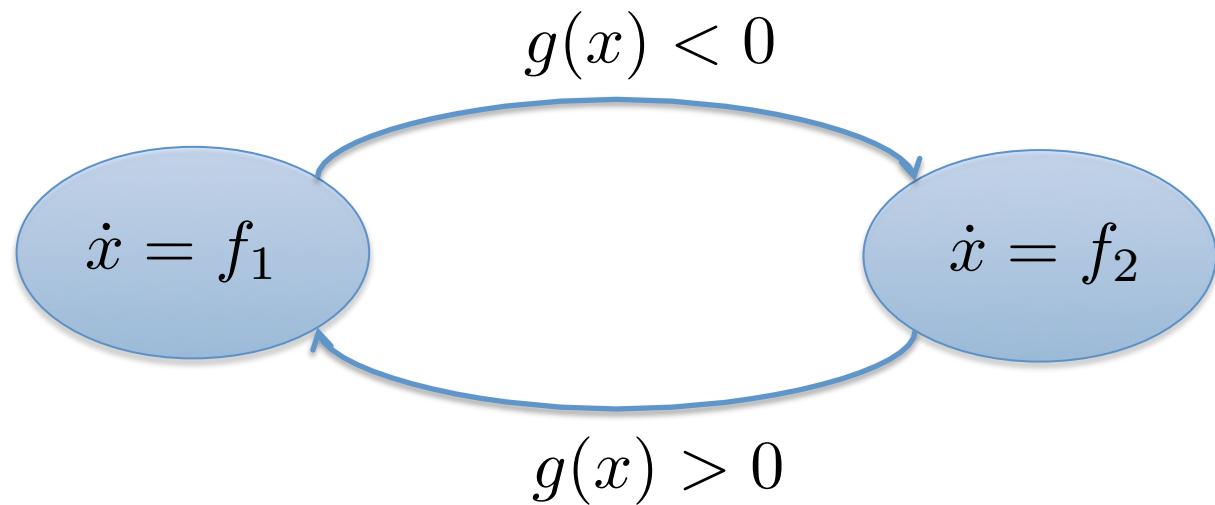
## Some Options

1. Go with the closest obstacle point (not bad)
2. Weigh and add the “obstacle vectors” depending on distance (better)
3. Weight and add depending on distance and direction of travel (best)
4. Make a map and plan (most bestest – not in this class)



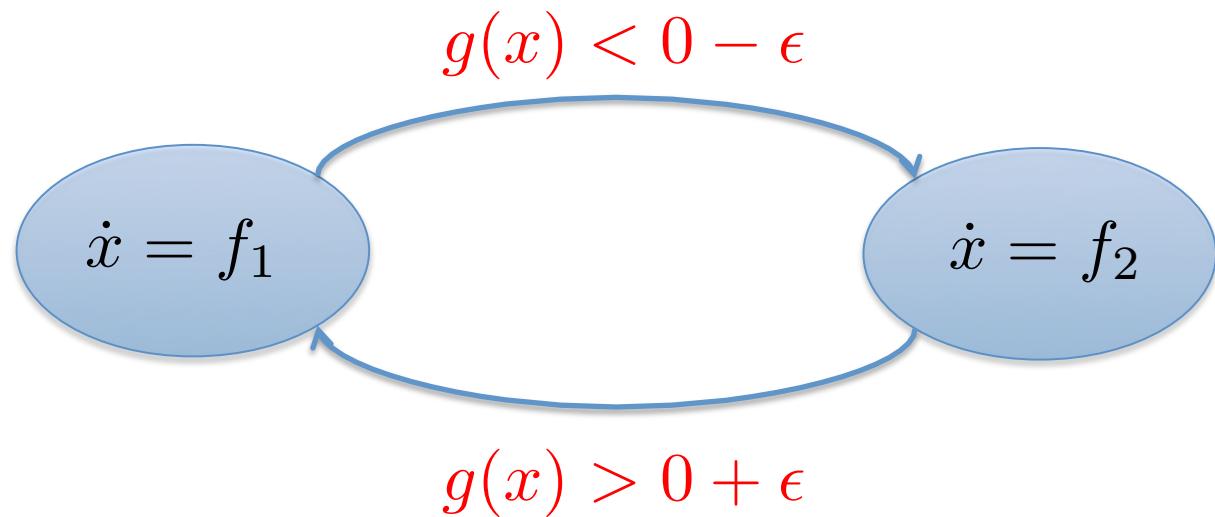
# Fat Guards

- Since no sensor is perfect, always allow for “fat” guards in the navigation HA



# Fat Guards

- Since no sensor is perfect, always allow for “fat” guards in the navigation HA



# And The Final Advise Is....

- TWEAK, TWEAK, TWEAK!!!!

# Lecture 6.8 –Let's Do It!

- Enough Talk!