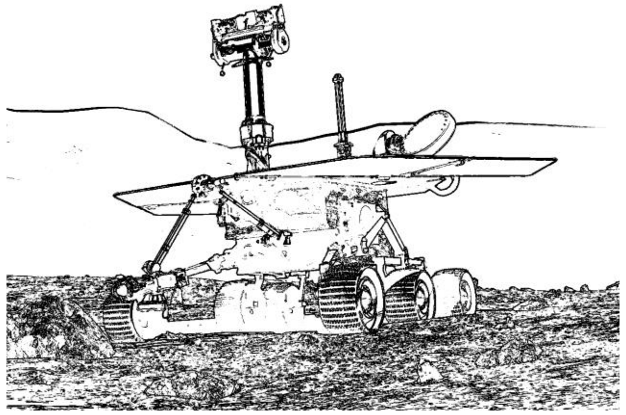
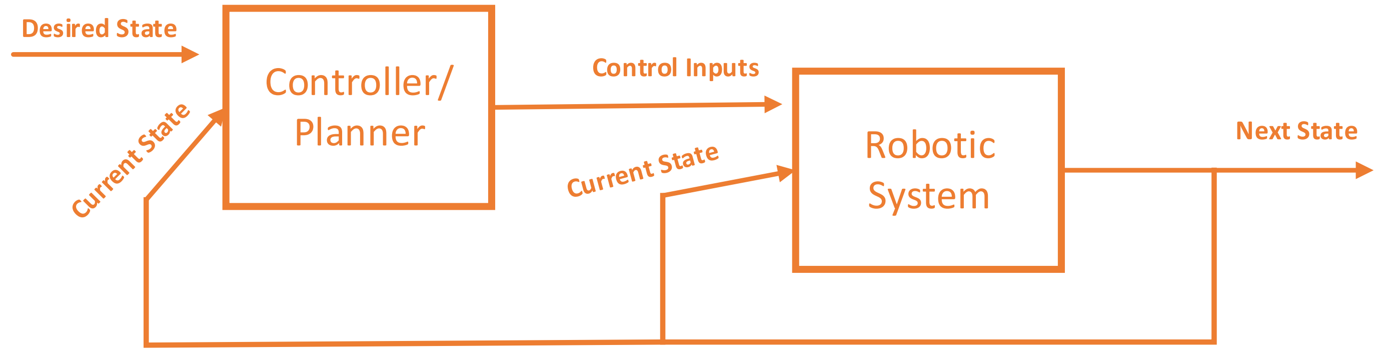


CSC477
Introduction to Mobile Robotics

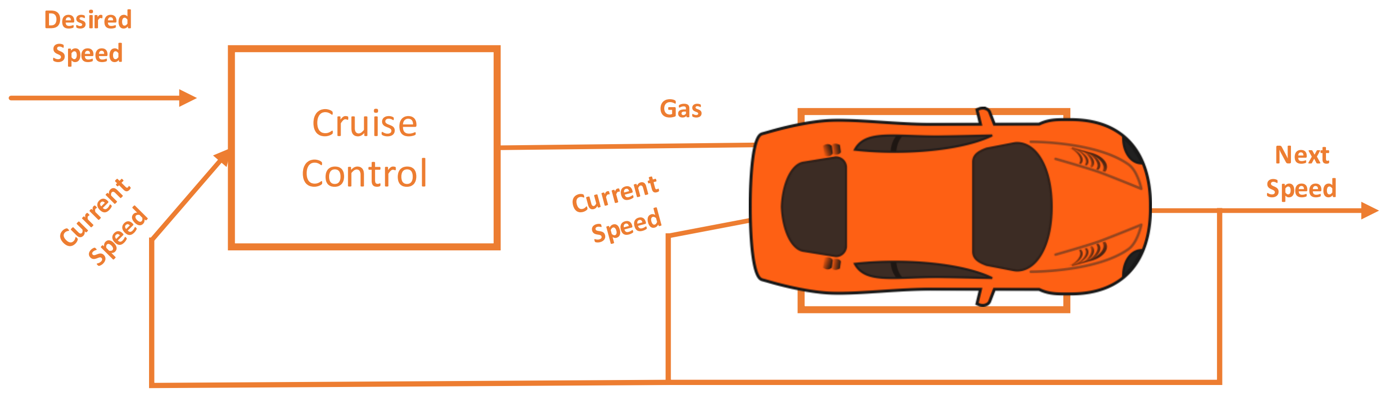
Florian Shkurti

Week #3: PID Control, Potential Fields, Obstacle Avoidance

Feedback Control



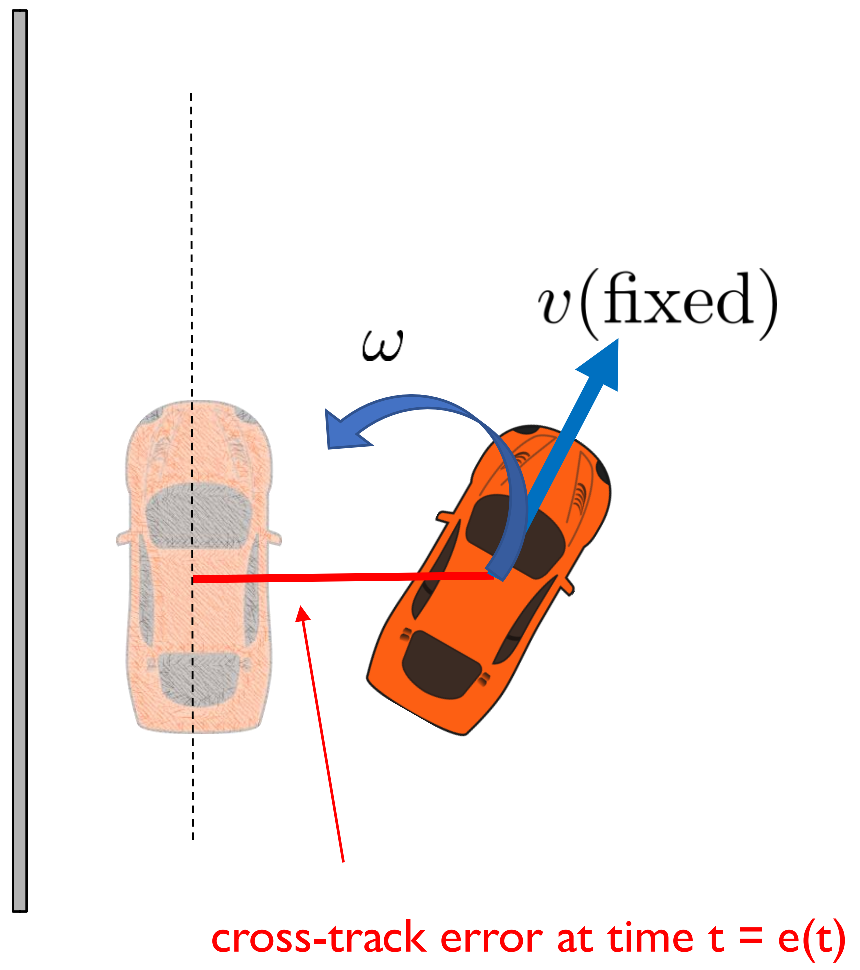
Example



Main question: what are the controls that will take the system from state A to B?

Example: wall/line following at fixed speed

You have to control the angular velocity ω



Why bother with wall/line following?

- Because it enables arbitrary path following where the line is the local tangent of the curved path

Idea 1: bang-bang control

$$\omega = \begin{cases} \omega_{\max} & \text{if} \quad \mathrm{CTE} > 0 \\ -\omega_{\max} & \text{if} \quad \mathrm{CTE} < 0 \\ 0 \end{cases}$$

What’s wrong with this?

Idea 2: proportional (P-)control

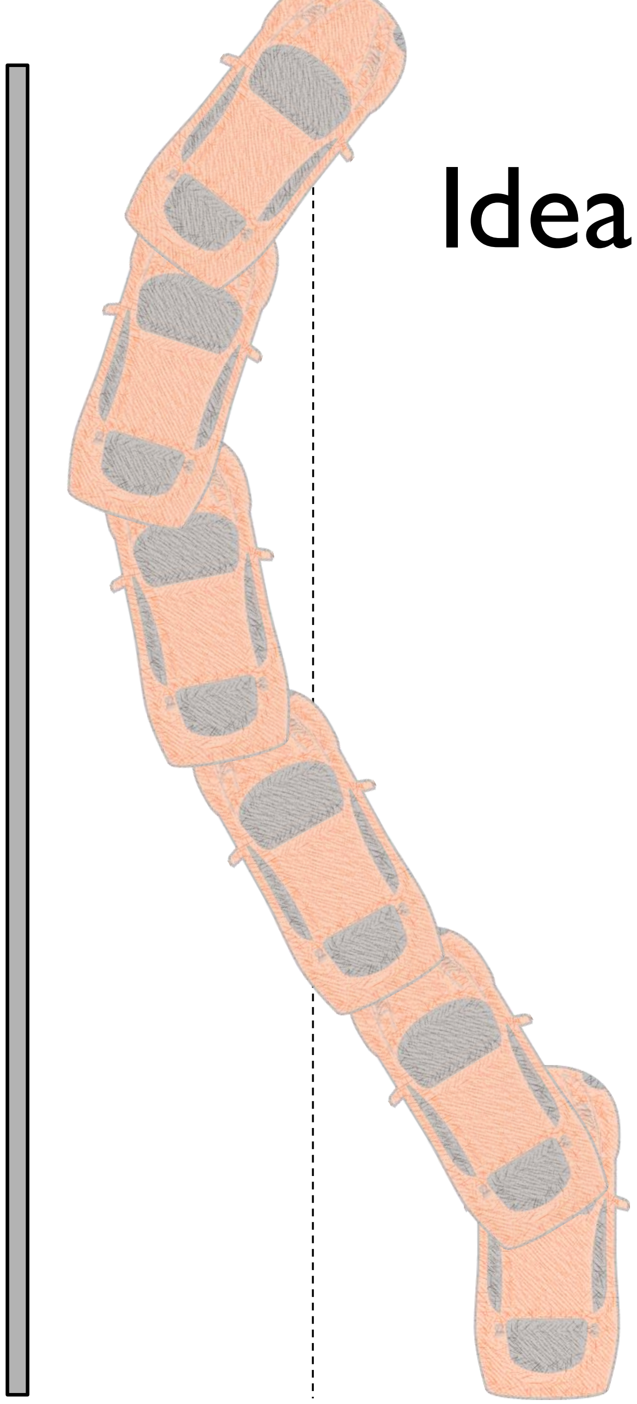
$$\omega = K_p e(t)$$

Will the car reach the target line?

Will the car overshoot the target line?

Is the asymptotic (steady-state) error zero?

Idea 2: proportional (P-)control



$$\omega = K_p e(t)$$

Will the car reach the target line? YES

Will the car overshoot the target line? YES

Is the asymptotic (steady-state) error zero? NO

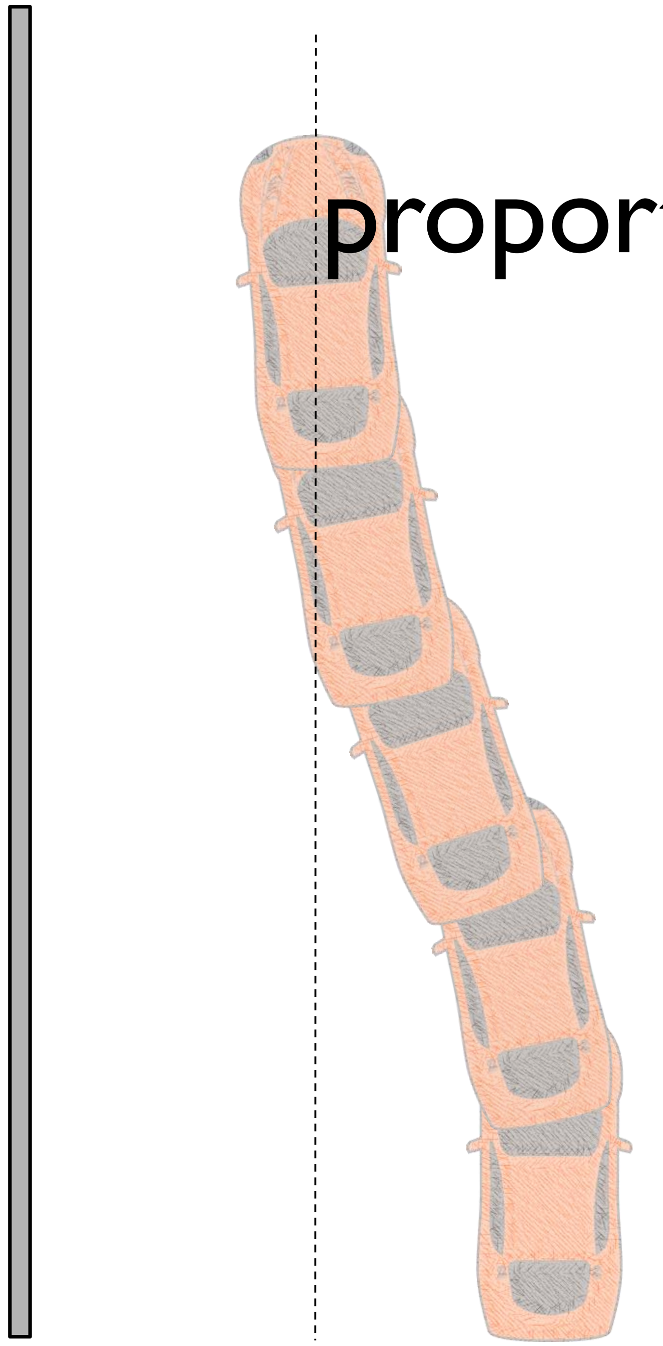
Addressing the oscillation problem

- Need to reduce turning rate **well before** the line is approached

- Idea: have a small proportional gain K_p
  Problem: that means the car doesn't turn very much

- Idea: need to **predict the error** in the near future
  This is good, as long as the error does not oscillate at a very high frequency

Idea 3: proportional-derivative (PD-)control



$$\omega = K_p e(t) + K_d \dot{e}(t)$$

How do we set the gains?

What if there are systematic errors/biases?

What if the error estimate is very noisy?

Handling systematic biases: the integral term

- Examples of systematic biases:
    - wheels are misaligned
    - car is much heavier on one side

- Add integral of the error from the beginning of time: $\int_{\tau=0}^{\tau=t} e(\tau) d\tau$

Handling systematic biases: the integral term

- Examples of systematic biases:
  - wheels are misaligned
  - car is much heavier on one side
- Add integral of the error from the beginning of time: $\int_{\tau=0}^{\tau=t} e(\tau) d\tau$
- Can the PI-controller have nonzero error asymptotically (in steady-state)?
  - NO. In steady state both the control ω_∞ and the error e_∞ must be constant. If the asymptotic error is nonzero then the control is not constant:

$$\omega_\infty = K_p e_\infty + K_i e_\infty t$$

Potential problem: integrator windup

- What happens if the control variable reaches the actuator’s limits?
- I.e. the car can’t turn as fast as the controller commands it.
- Actuator may remain at its limit for a long time while the controller modifies its commands
- Error increases, integral term winds up while controller goes back to issuing commands in the feasible region.

Potential problem: integrator windup

- Heuristic fixes:
  - Limit integral error values
  - Stop integral error while the commands are in the non feasible region
  - Reduce gain of integral error term

PID controller

$$\omega(t) = K_p e(t) + K_d \dot{e}(t) + K_i \int_{\tau=0}^{\tau=t} e(\tau) d\tau$$

Perhaps the most widely used controller in industry and robotics.
Perhaps the easiest to code.

You will also see it as:

$$\omega(t) = K_p \left[e(t) + T_d \dot{e}(t) + \frac{1}{T_i} \int_{\tau=0}^{\tau=t} e(\tau) d\tau \right]$$

Tips: how to implement PID

- Assume time is discrete
- Identify your error function e(t)= current_state(t)- target_state(t)
- Is the measurement of the state reliable?
- If the measurement of the current state is very noisy you might want to smooth/filter it, using:
  - **Moving average filter** with uniform weights

    $$\hat{x}_t = \frac{x_t + x_{t-1} + ... + x_{t-k+1}}{k} = \hat{x}_{t-1} + \frac{x_t - x_{t-k}}{k}$$

    Potential problem: the larger the window of the filter the slower it is going to register changes.
  - **Exponential filter**

    $$\hat{x}_t = \alpha\hat{x}_{t-1} + (1-\alpha)x_t, \quad \alpha \in [0,1]$$

    Potential problem: the closer α is to 1 the slower it is going to register changes.

Tips: how to implement PID

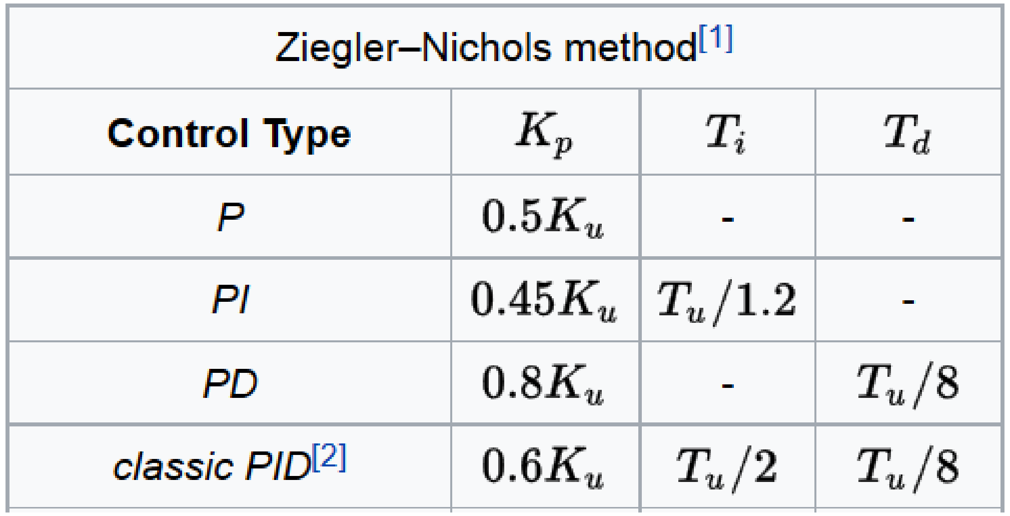
- Approximate the integral of error by a sum
- Approximate the derivative of error by:
  - Finite differences $\dot{e}(t_k) \approx \frac{e(t_k) - e(t_{k-1})}{\delta t}$
  - Filtered finite differences, e.g. $\dot{e}(t_k) \approx \alpha \dot{e}(t_{k-1}) + (1 - \alpha) \frac{e(t_k) - e(t_{k-1})}{\delta t}$
- Limit the computed controls
- Limit or stop the integral term when detecting large errors and windup

Tips: how to tune the PID

- Manually:
  - First, use only the proportional term. Set the other gains to zero.
  - When you see oscillations slowly add derivative term
    - Increasing K_d increases the duration in which linear error prediction is assumed to be valid
  - Add a small integral gain

Tips: how to tune the PID

- Ziegler-Nichols heuristic:
  - First, use only the proportional term. Set the other gains to zero.
  - When you see consistent oscillations, record the proportional gain K_u and the oscillation period T_u

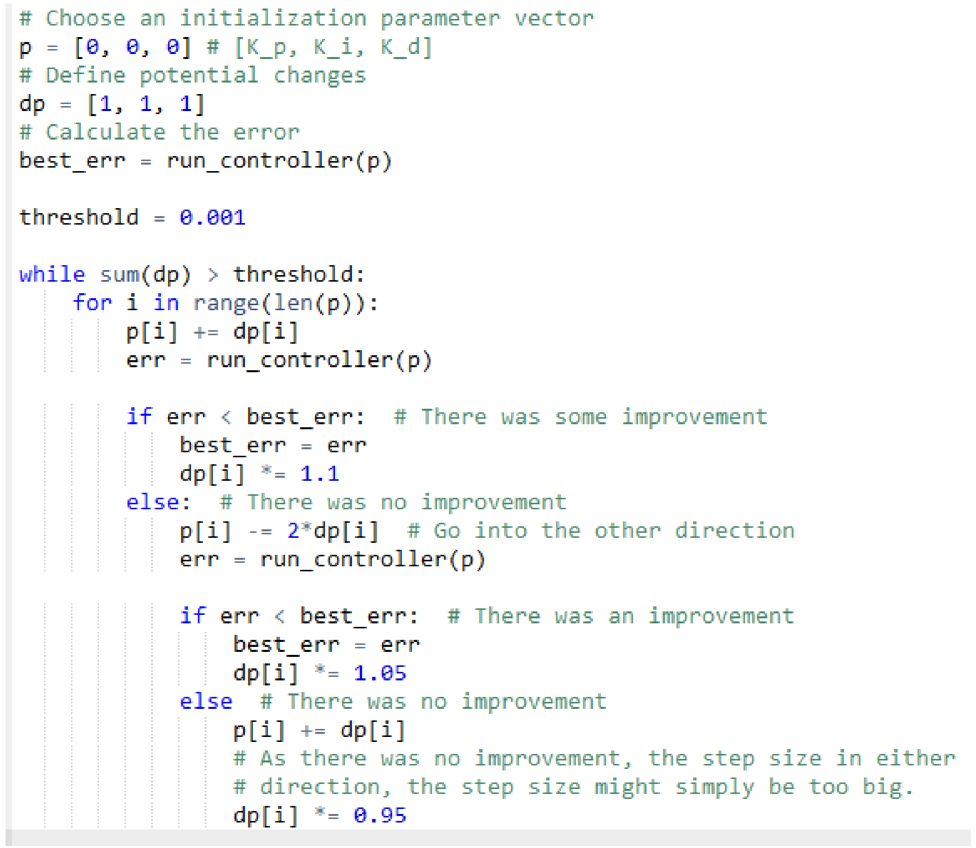


Ziegler–Nichols method[1]

Control Type	K_p	T_i	T_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$T_u/1.2$	-
PD	$0.8K_u$	-	$T_u/8$
classic PID[2]	$0.6K_u$	$T_u/2$	$T_u/8$

Tips: how to tune the PID

- After manual or Z-N tweaking you might want to use coordinate ascent to search for a better set of parameters automatically:



```
# Choose an initialization parameter vector
p = [0, 0, 0] # [K_p, K_i, K_d]
# Define potential changes
dp = [1, 1, 1]
# Calculate the error
best_err = run_controller(p)

threshold = 0.001

while sum(dp) > threshold:
    for i in range(len(p)):
        p[i] += dp[i]
        err = run_controller(p)

        if err < best_err:  # There was some improvement
            best_err = err
            dp[i] *= 1.1
        else:  # There was no improvement
            p[i] -= 2*dp[i]  # Go into the other direction
            err = run_controller(p)

            if err < best_err:  # There was an improvement
                best_err = err
                dp[i] *= 1.05
            else  # There was no improvement
                p[i] += dp[i]
                # As there was no improvement, the step size in either
                # direction, the step size might simply be too big.
                dp[i] *= 0.95
```

See Sebastian Thrun's online class "AI for robotics" on Udacity for more details on this. He calls the algorithm Twiddle and it is in Lesson 5.

Other names for this are "Self-tuning PID controllers"

When is PID insufficient?

- Systems with large time delays
- Controllers that require completion time guarantees
  - E.g. the system must reach target state within 2 secs
- Systems with high-frequency oscillations
- High-frequency variations on the target state

Example applications: self-driving cars

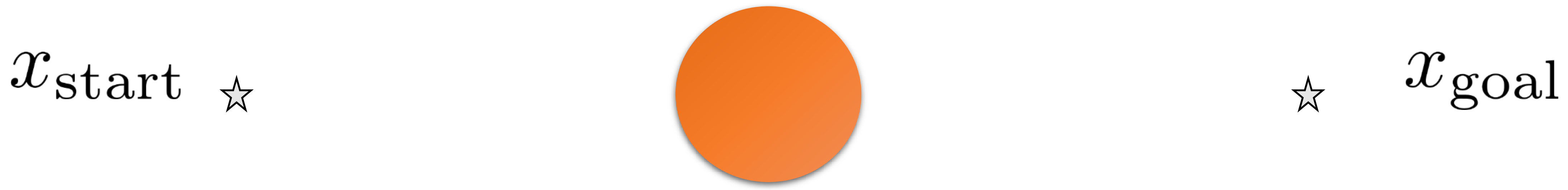
Cascading PID

- Sometimes we have multiple error sources (e.g. multiple sensors) and one actuator to control.

- We can use a master PID loop that sets the setpoint for the slave PID loop. Master (outer loop) runs at low rate, while slave (inner loop) runs at higher rate.

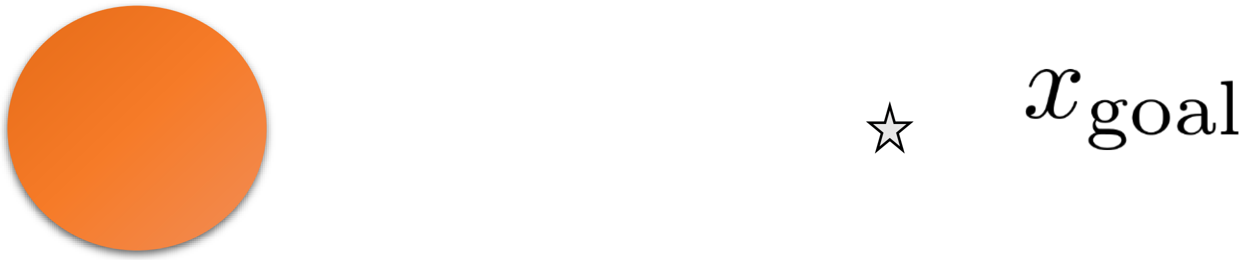
- One way of getting hierarchical control behavior.

Potential Fields
Main Motivation



Q: How do you control the robot to reach the goal state while avoiding the obstacle?

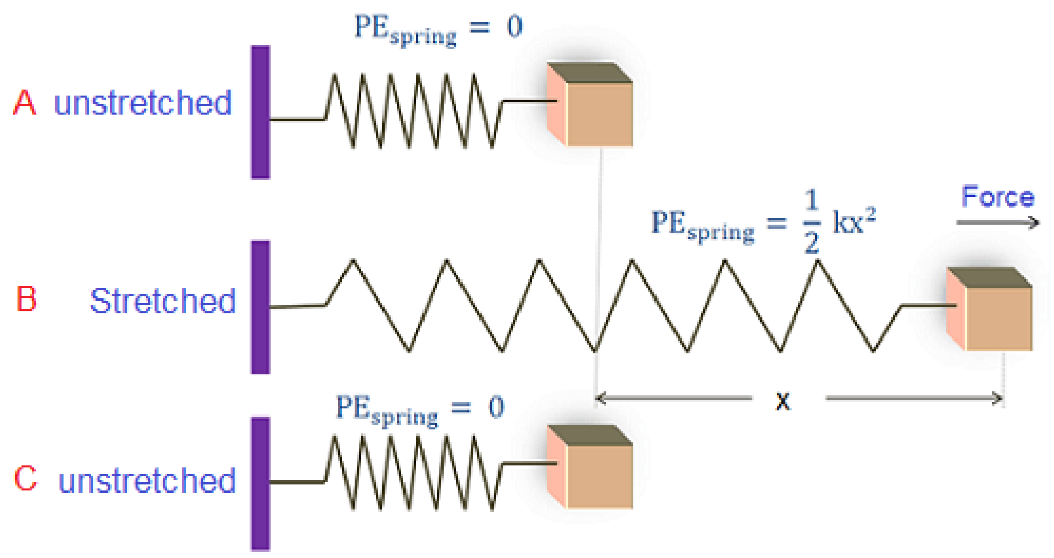
Main Motivation



Q: How do you control the robot to reach the goal state while avoiding the obstacle?

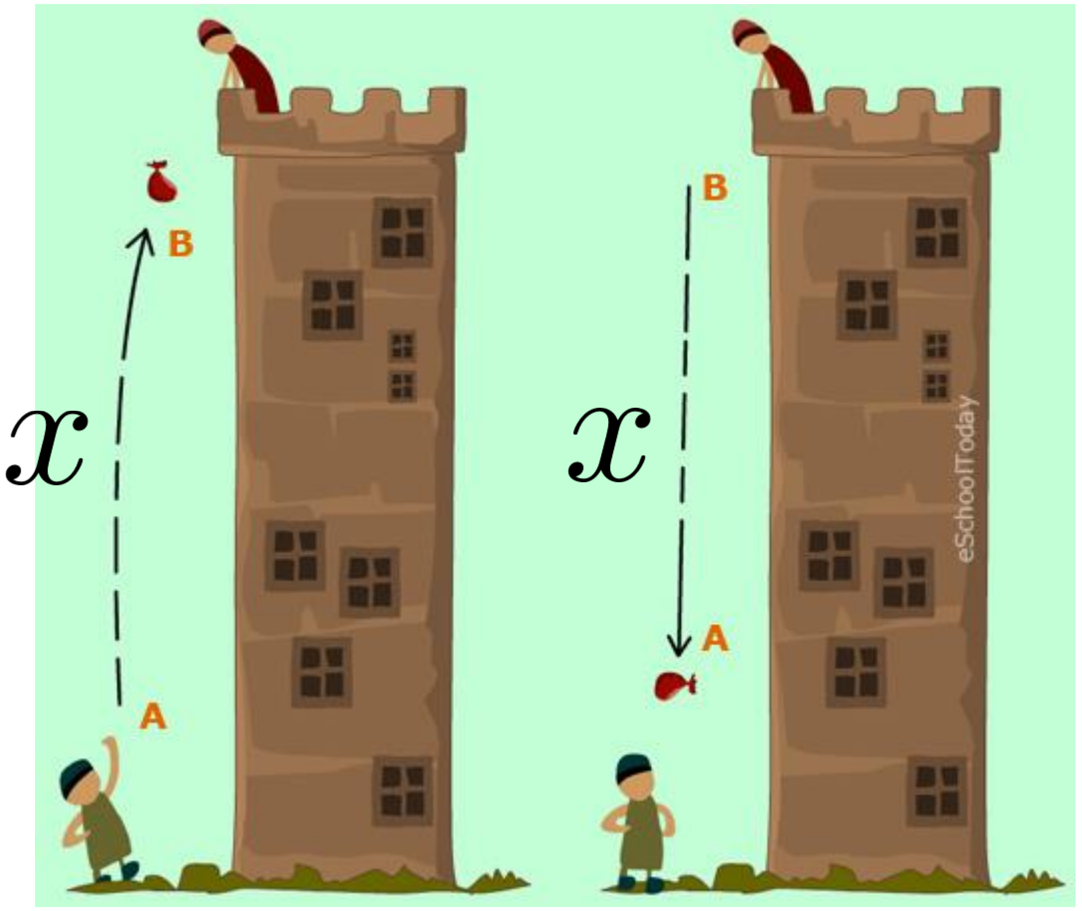
Assume omnidirectional robot.

Background: potential energy and forces



$$U(x) = \frac{1}{2} kx^2$$
$$F(x) = -kx$$

Background: potential energy and forces



$$U(x) = mgx$$

$$F(x) = -mg$$

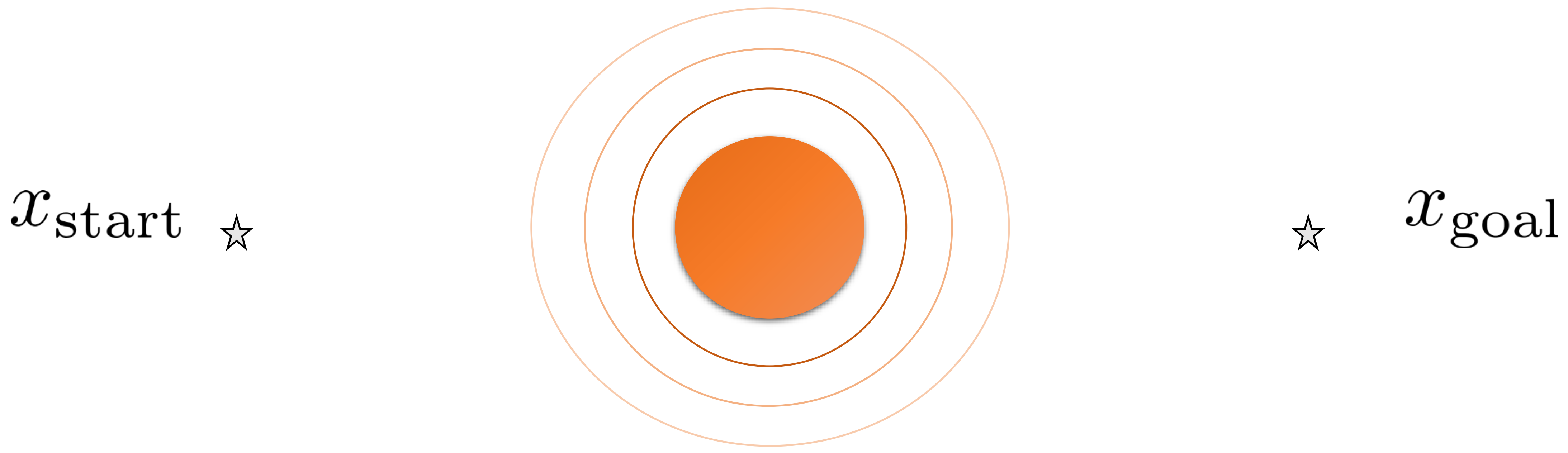
Background: potential energy and forces

In both cases we have conversion from kinetic energy to potential energy $U(x)$.

In both cases there is a force resulting from the potential field, and $F(x)=-dU(x)/dx$.

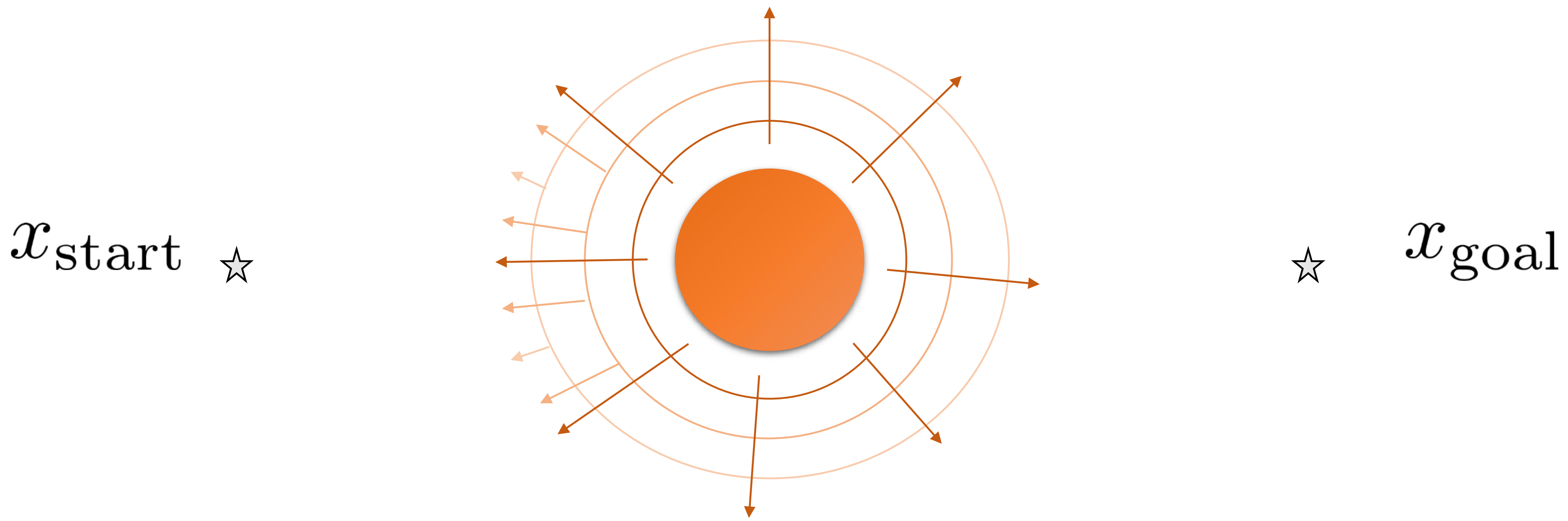
This is a general rule for conservative systems with no external forces.

Main Motivation



Q: How do you control the robot to reach the goal state while avoiding the obstacle?

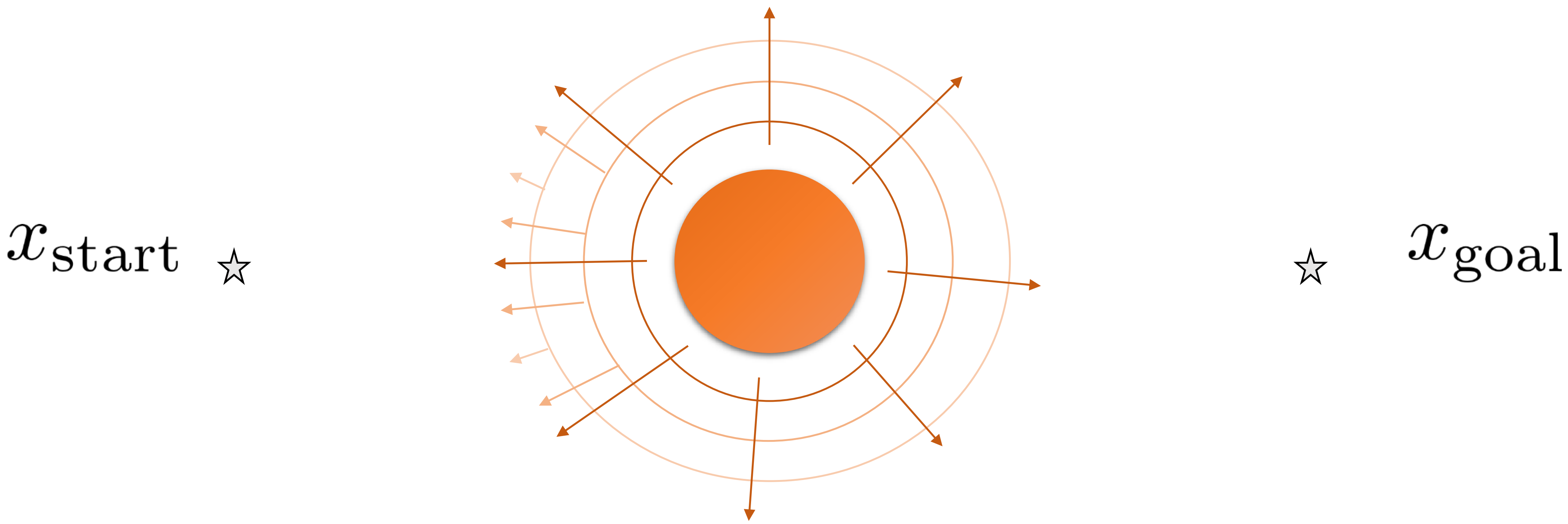
Artificial Potential Fields



Q: How do you control the robot to reach the goal state while avoiding the obstacle?

A: Place a repulsive potential field around obstacles

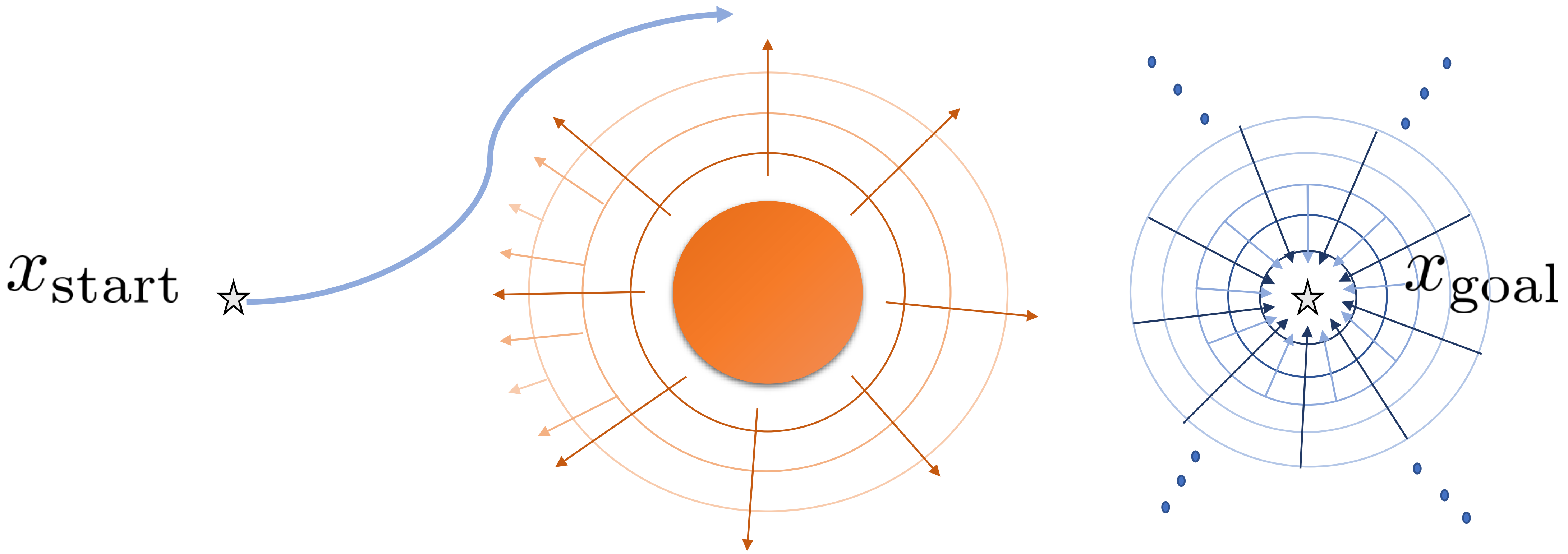
Artificial Potential Fields



Q: How do you control the robot to reach the goal state while avoiding the obstacle?

A: Place a repulsive potential field around obstacles

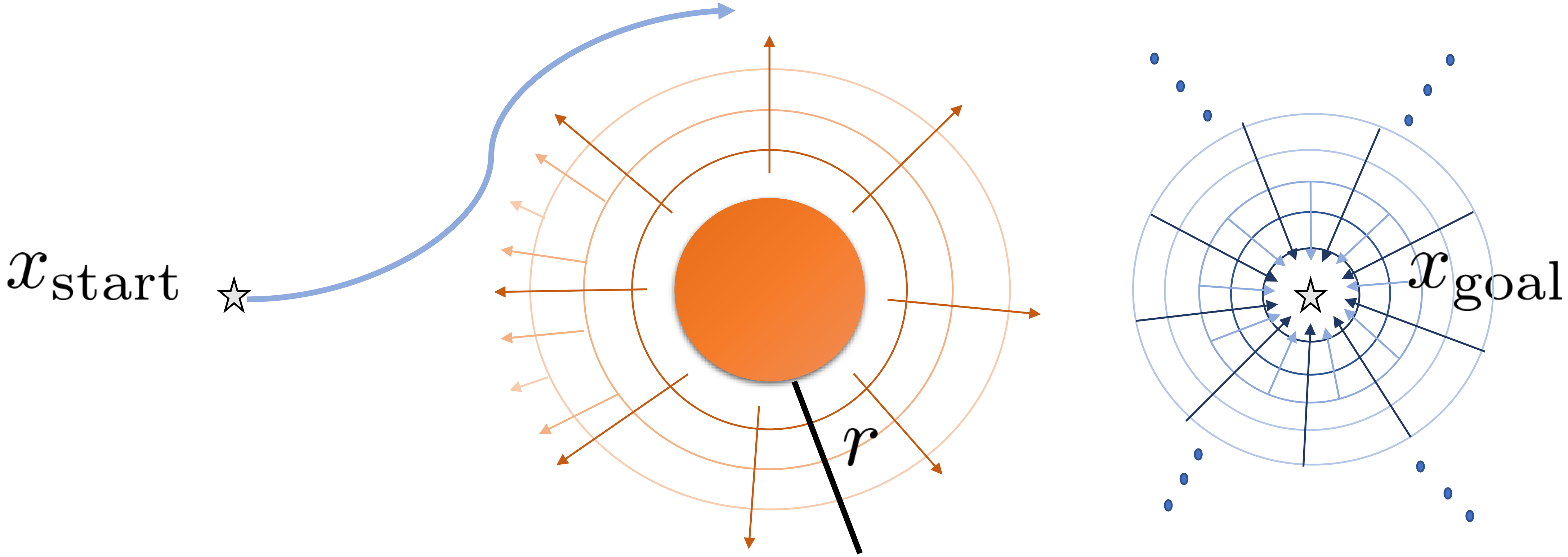
Artificial Potential Fields



Q: How do you control the robot to reach the goal state while avoiding the obstacle?

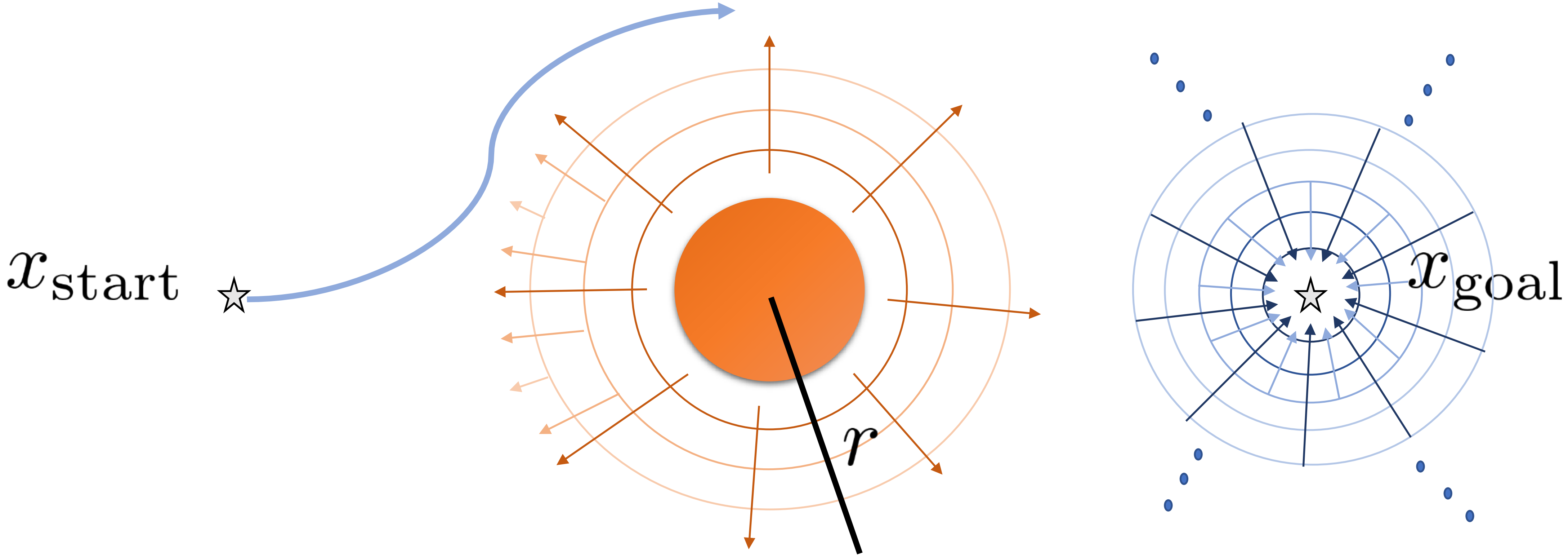
A: Place a repulsive potential field around obstacles
and an attractive potential field around the goal

Artificial Potential Fields



$$U_{\text{repulsive}}(x) = \begin{cases} \left(\frac{1}{d(x,\text{obs})} - \frac{1}{r}\right)^2 & \text{if} \quad d(x,\text{obs}) < r \\ 0 & \text{if} \quad d(x,\text{obs}) \geq r \end{cases}$$

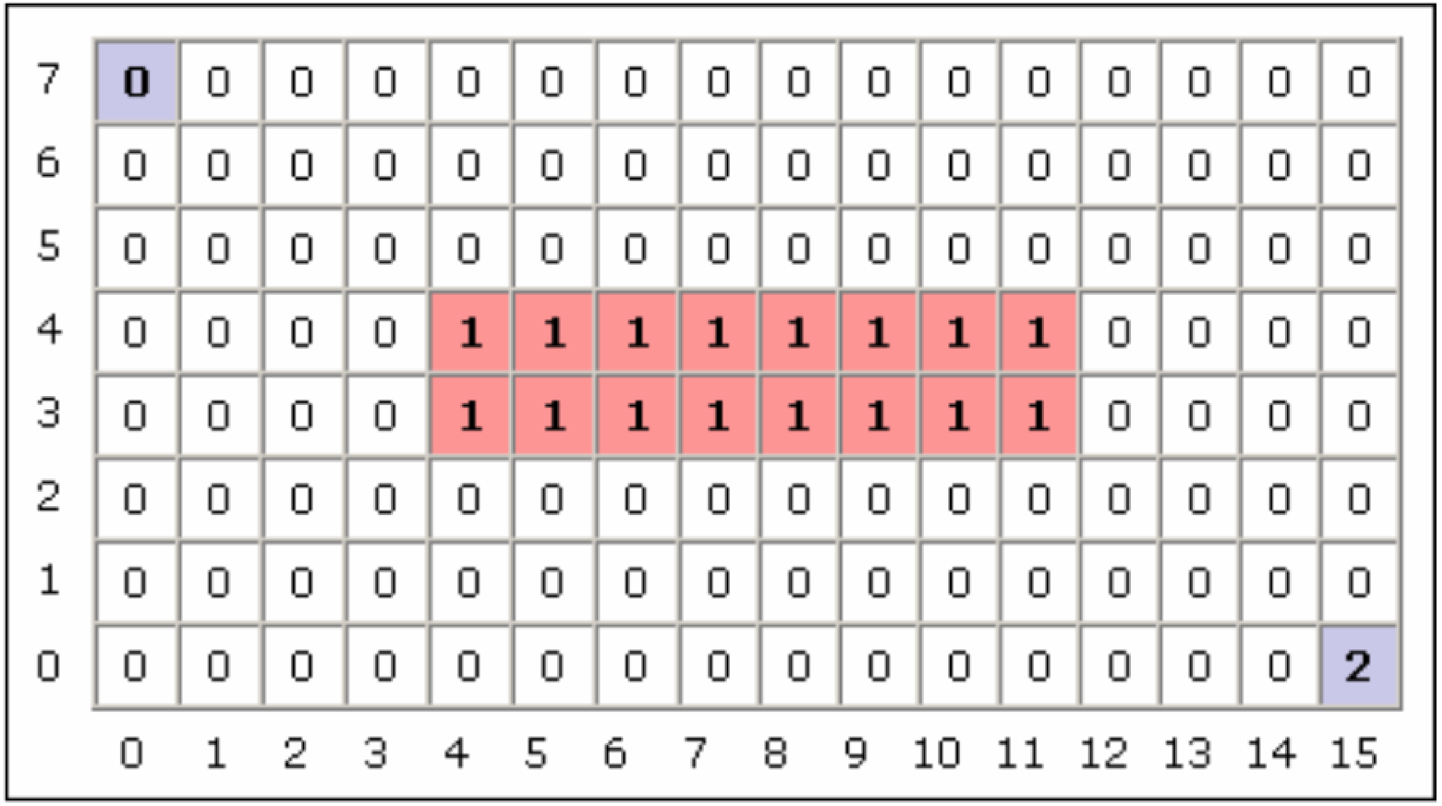
Artificial Potential Fields



$$U_{\text{repulsive}}(x) = \begin{cases} (\frac{1}{d(x,\text{obs})} - \frac{1}{r})^2 & \text{if} \quad d(x,\text{obs}) < r \\ 0 & \text{if} \quad d(x,\text{obs}) \geq r \end{cases}$$

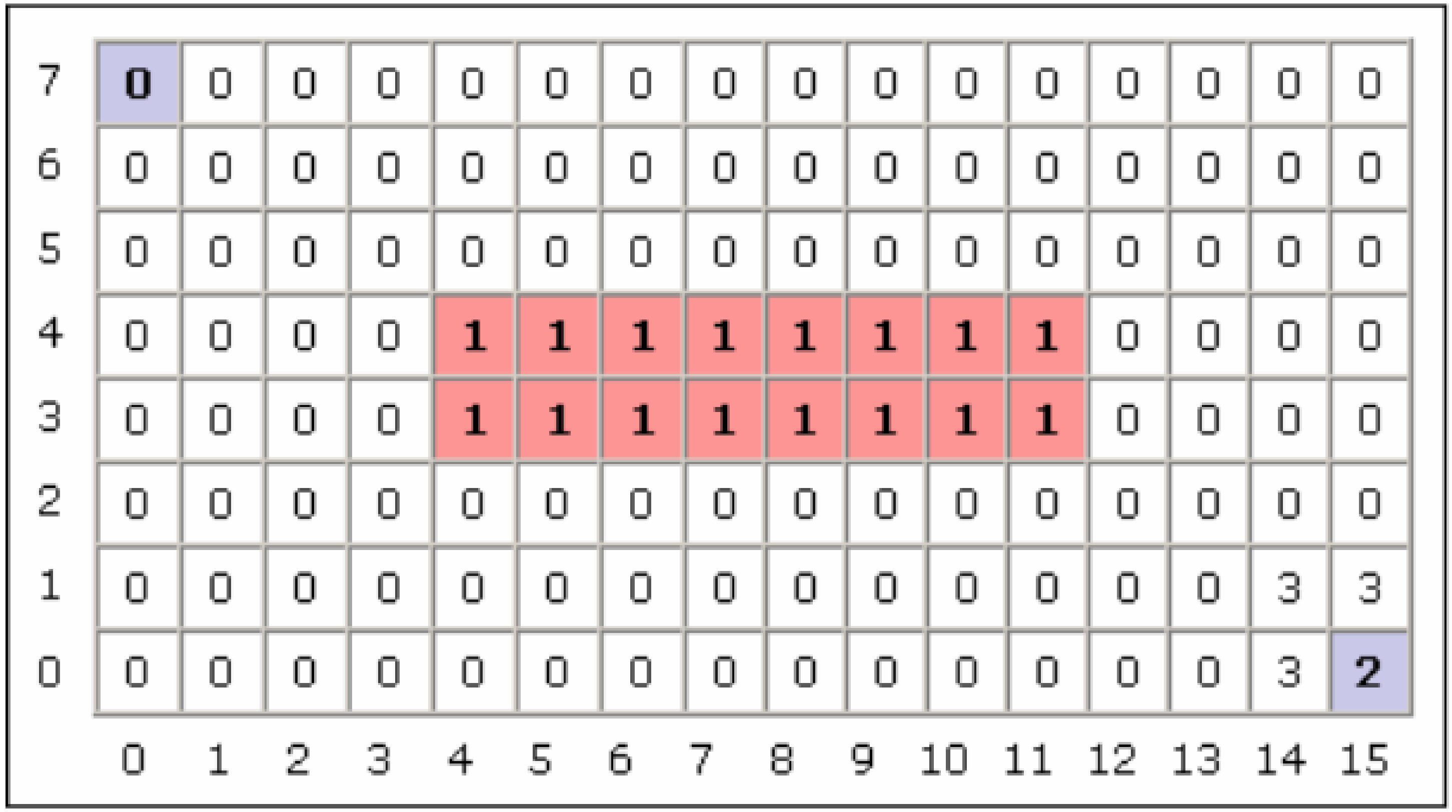
$$U_{\text{attractive}}(x) = d(x, x_{\text{goal}})^2$$

How do we compute these distances from obstacles?



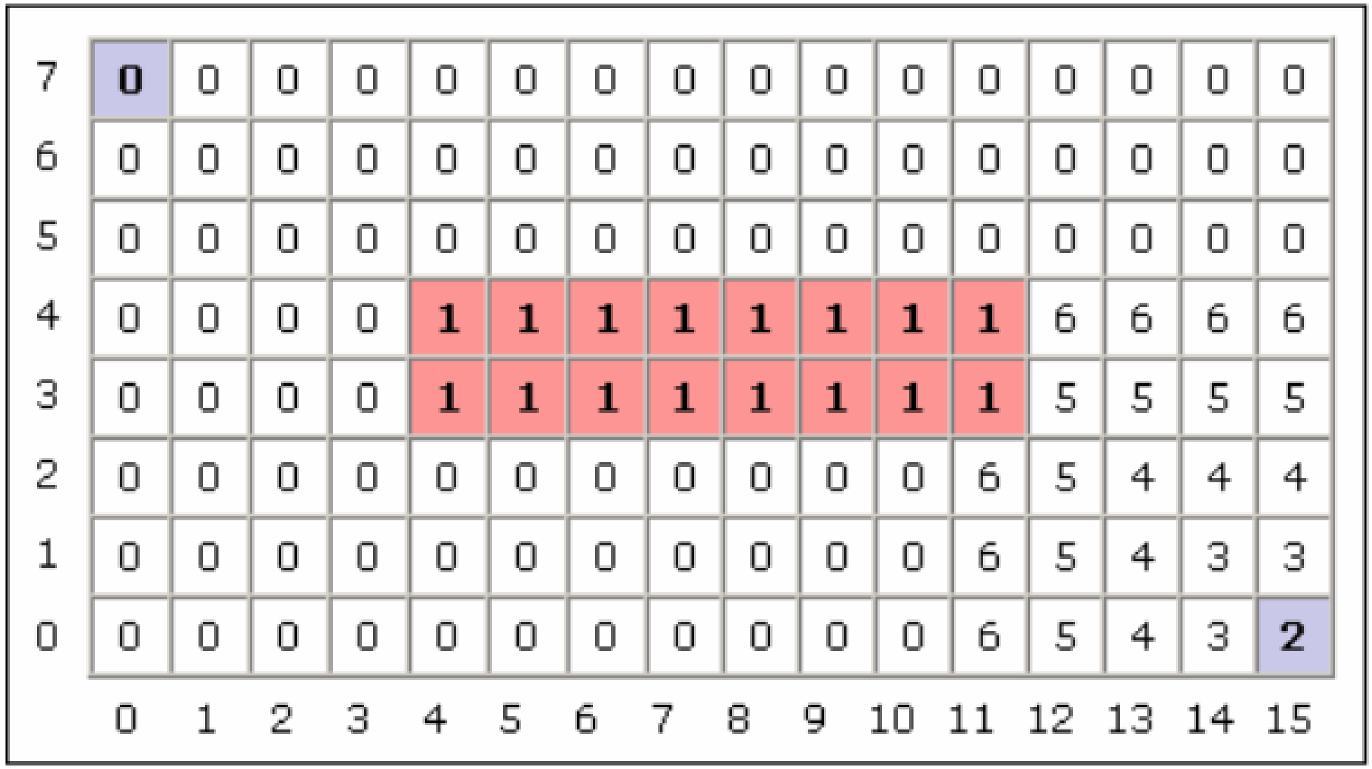
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

How do we compute these distances from obstacles?



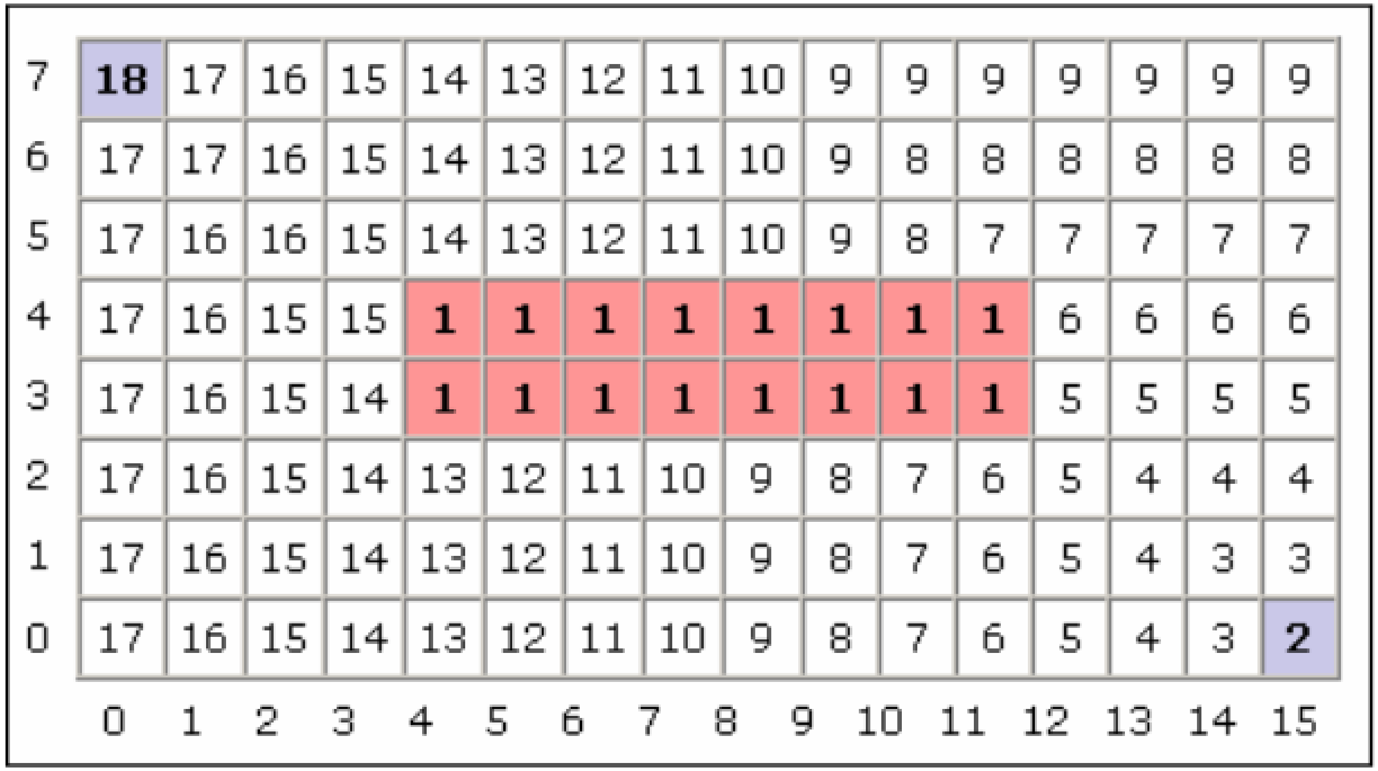
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

How do we compute these distances from obstacles?



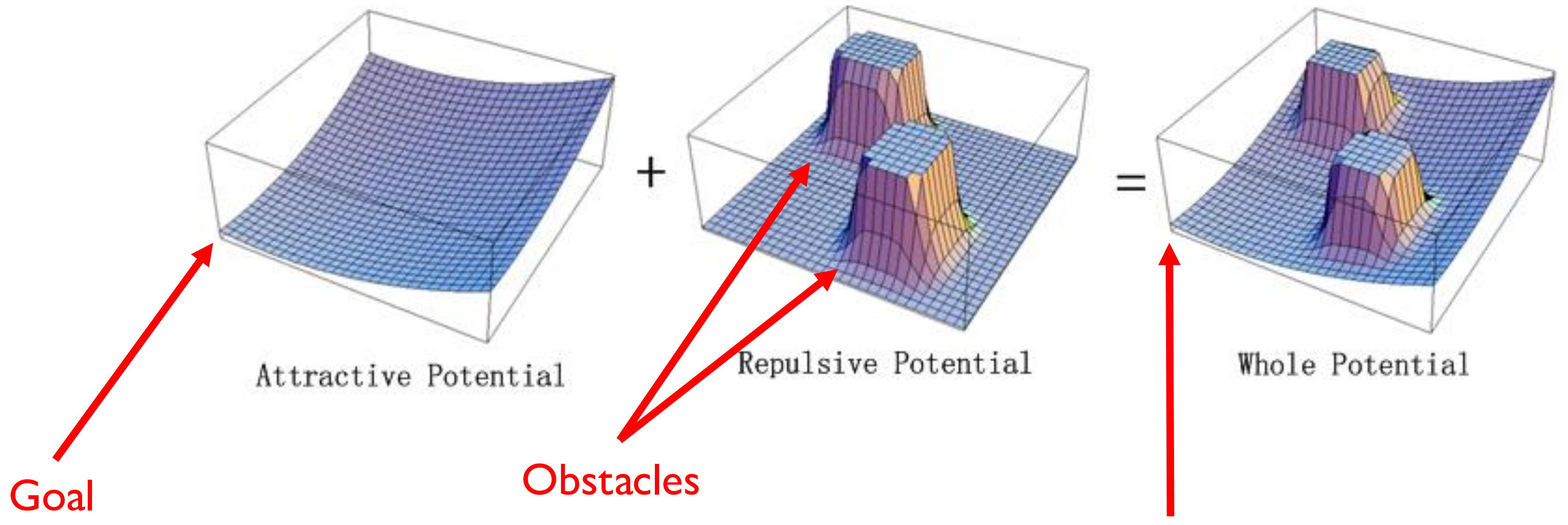
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	6	6	6	6
3	0	0	0	0	1	1	1	1	1	1	1	1	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

How do we compute these distances from obstacles?



7	**18**	17	16	15	14	13	12	11	10	9	9	9	9	9	9	9
6	17	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8
5	17	16	16	15	14	13	12	11	10	9	8	7	7	7	7	7
4	17	16	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	17	16	15	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	3
0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Attractive and Repulsive Potential Fields



$$U(x) = \alpha U_{\text{attractive}}(x) + \beta U_{\text{repulsive}}(x)$$

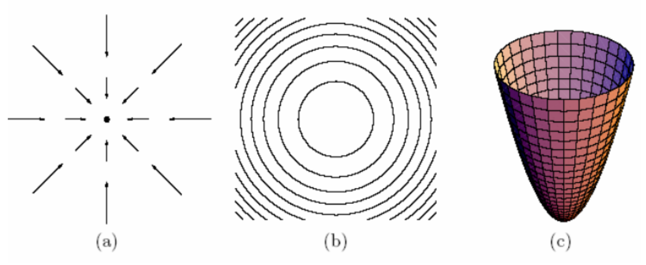
Q1: How do we reach the goal state from an arbitrary state?

Q2: In this example there is an unambiguous way to reach the goal from any state. Is this true in general?

From Potential Fields to Forces

Make the robot move by applying forces resulting from potential fields

$$U_{\text{attractive}}(x) = d(x, x_{\text{goal}})^2 \Longrightarrow F_{\text{attractive}}(x) = -\nabla_x U_{\text{attractive}}(x) = -2(x - x_{\text{goal}})$$



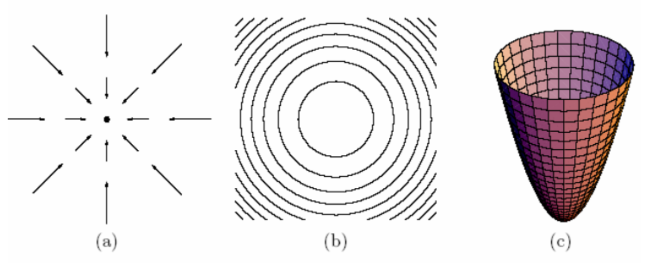
$\Downarrow$

Attractive force makes state x go to the bottom
of the potential energy bowl. Bottom=Goal = low-energy state.

From Potential Fields to Forces

Make the robot move by applying forces resulting from potential fields

$$U_{\text{attractive}}(x) = d(x, x_{\text{goal}})^2 \Rightarrow F_{\text{attractive}}(x) = -\nabla_x U_{\text{attractive}}(x) = -2(x - x_{\text{goal}})$$



$\Downarrow$

Attractive force makes state x go to the bottom of the potential energy bowl. Bottom=Goal = low-energy state.

$\Downarrow$

Move the robot using F=ma, for m=1:

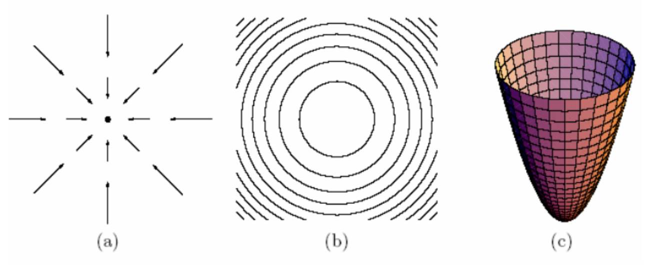
$$\dot{x}_{t+1} = \dot{x}_t + \delta t F(x_t)$$

Gradient descent down the potential bowl

From Potential Fields to Forces

Make the robot move by applying forces resulting from potential fields

$$U_{\text{attractive}}(x) = d(x, x_{\text{goal}})^2 \Longrightarrow F_{\text{attractive}}(x) = -\nabla_x U_{\text{attractive}}(x) = -2(x - x_{\text{goal}})$$



$$\Downarrow$$

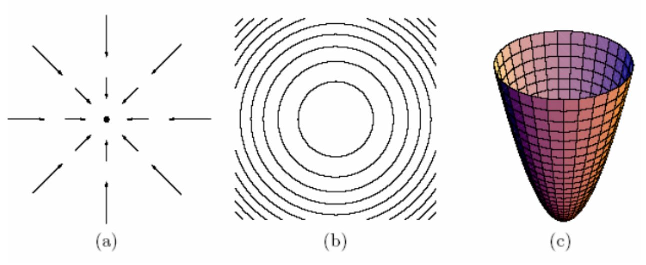
Attractive force makes state x go to the bottom
of the potential energy bowl. Bottom=Goal = low-energy state.

Q: Do you see any problems with this potential energy and force if x is far away from goal?

From Potential Fields to Forces

Make the robot move by applying forces resulting from potential fields

$$U_{\text{attractive}}(x) = d(x, x_{\text{goal}})^2 \quad \Longrightarrow \quad F_{\text{attractive}}(x) = -\nabla_x U_{\text{attractive}}(x) = -2(x - x_{\text{goal}})$$



$\Downarrow$

Attractive force makes state x go to the bottom of the potential energy bowl. Bottom=Goal = low-energy state.

Q: Do you see any potential problems with this if x is far away from goal?

A: The farther the robot is the stronger the force. May need to normalize the force vector. Alternatively:

$$U_{\text{attractive}}(x) = d(x, x_{\text{goal}}) \quad \Longrightarrow \quad F_{\text{attractive}}(x) = -\frac{(x - x_{\text{goal}})}{d(x, x_{\text{goal}})}$$

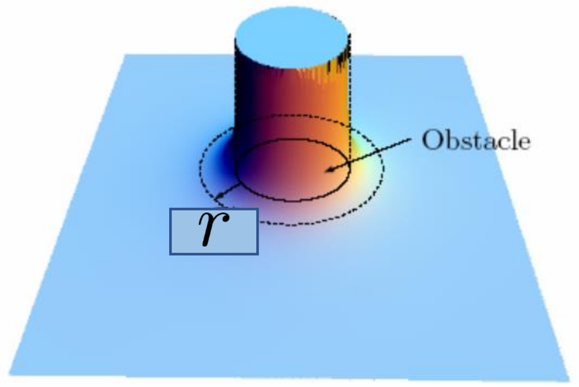
From Potential Fields to Forces

Make the robot move by applying forces resulting from potential fields

$$U_{\text{repulsive}}(x) = \begin{cases} (\frac{1}{d(x,\text{obs})} - \frac{1}{r})^2 & \text{if} \quad d(x,\text{obs}) < r \\ 0 & \text{if} \quad d(x,\text{obs}) \geq r \end{cases}$$

$$\Rightarrow$$

$$F_{\text{repulsive}}(x) = \begin{cases} 2(\frac{1}{d(x,\text{obs})} - \frac{1}{r})\frac{\nabla_x d(x,\text{obs})}{d(x,\text{obs})^2} & \text{if} \quad d(x,\text{obs}) < r \\ 0 & \text{otherwise} \end{cases}$$



$$\Downarrow$$

Repulsive force makes state x go away from the obstacle
to lower potential energy states. Free space = {low-energy states}

$$\Downarrow$$

Move the robot using F=ma, for m=1:

$$\dot{x}_{t+1} = \dot{x}_t + \delta t F(x_t)$$

Gradient descent until obstacle is cleared

Combining Attractive and Repulsive Forces

Potential energy

$$U_{\text{total}}(x) = \alpha U_{\text{attractive}}(x) + \beta U_{\text{repulsive}}(x)$$

results in forces

$$F_{\text{total}}(x) = \alpha F_{\text{attractive}}(x) + \beta F_{\text{repulsive}}(x)$$

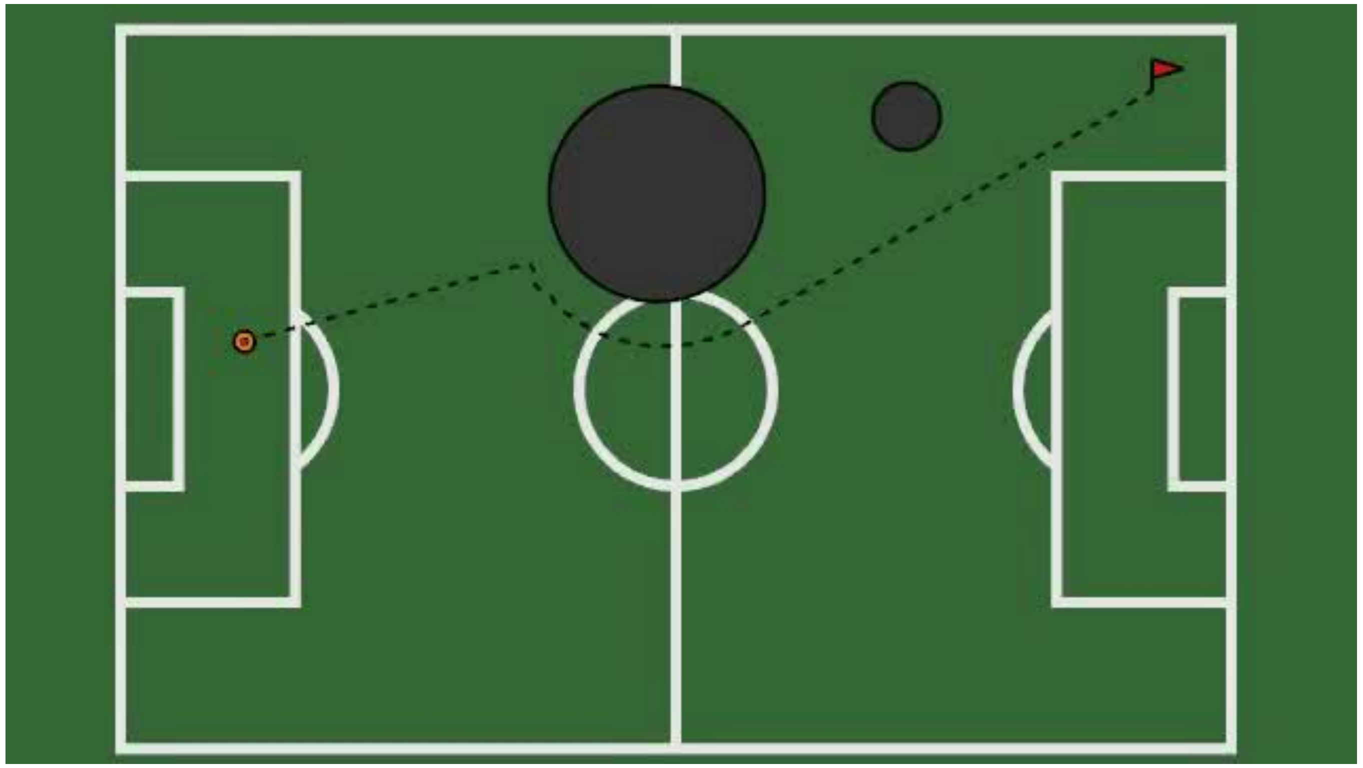
makes robot accelerate

$$\dot{x}_{t+1} = \dot{x}_t + \delta t F(x_t)$$

Artificial Potential Fields: Example

Advantages of potential fields:

- Can handle moving obstacles
- Fast and easy to compute
- Fairly reactive



Combining Attractive and Repulsive Forces

Potential energy

$$U_{\text{total}}(x) = \alpha U_{\text{attractive}}(x) + \beta U_{\text{repulsive}}(x)$$

$$\Downarrow$$

results in forces

$$F_{\text{total}}(x) = \alpha F_{\text{attractive}}(x) + \beta F_{\text{repulsive}}(x)$$

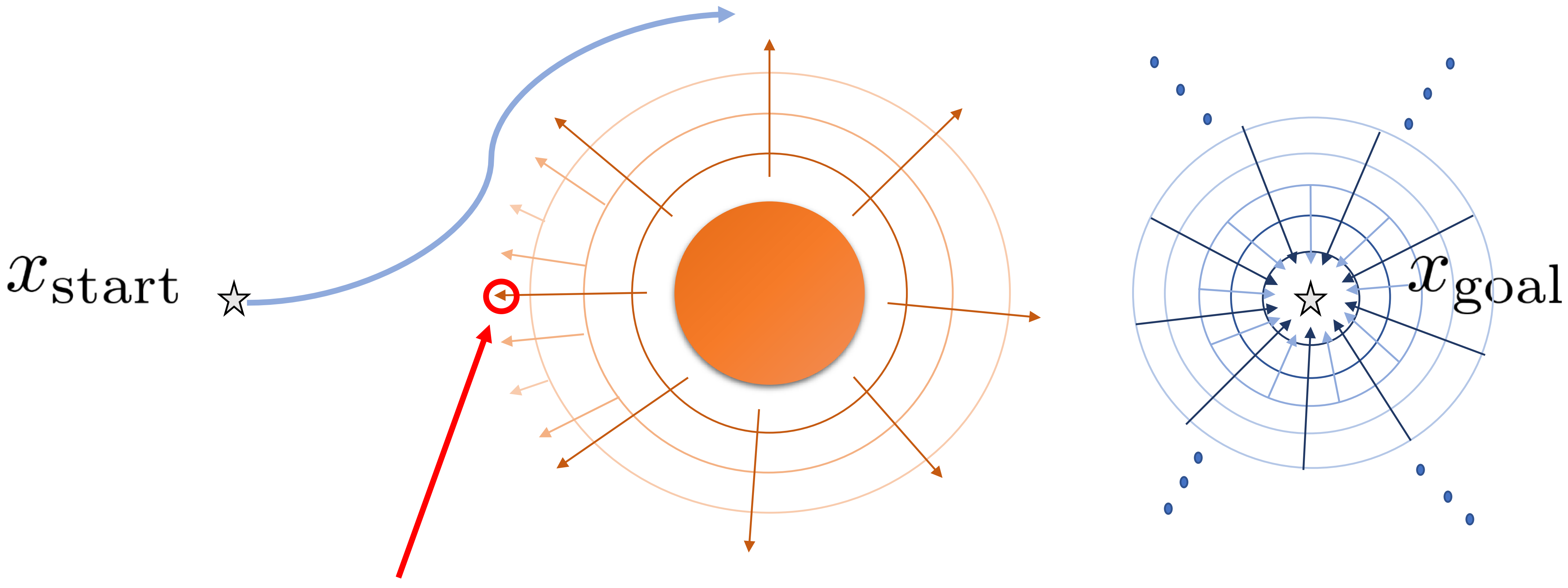
Q: What's a possible problem with addition of forces?

$$\Downarrow$$

makes robot accelerate

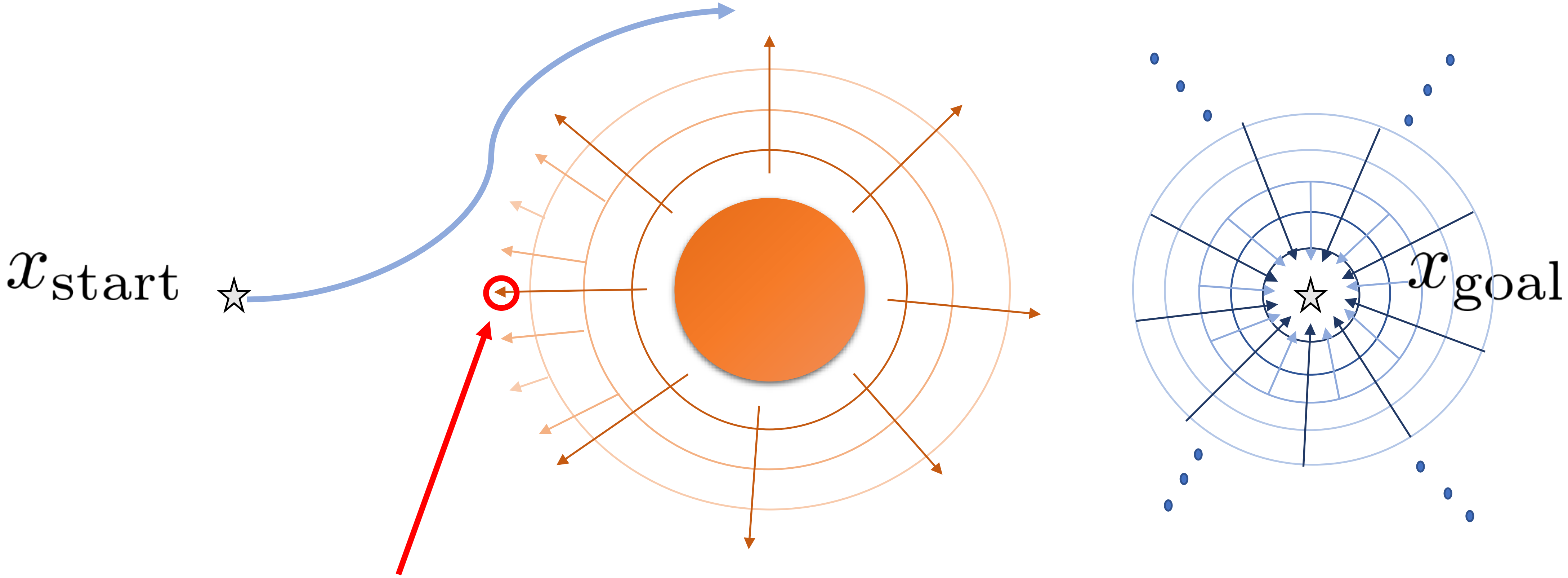
$$\dot{x}_{t+1} = \dot{x}_t + \delta t F(x_t)$$

Artificial Potential Fields



What's the total potential here?

Artificial Potential Fields



What's the total potential here?

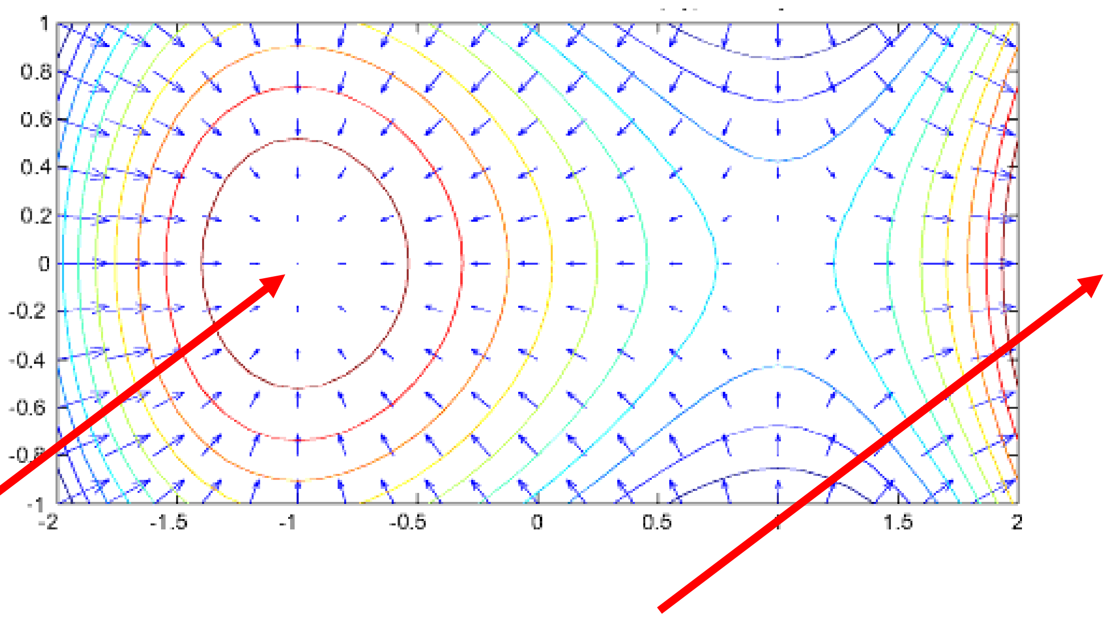
It's zero. The repulsive force is exactly the opposite of the attractive force (assuming alpha = beta)

$$F_{\text{total}}(x) = \alpha F_{\text{attractive}}(x) + \beta F_{\text{repulsive}}(x) = 0$$

Problem: gradient descent gets stuck

Local Minima on the Potential Field: Getting Stuck

States of zero total force correspond to local minima in the potential function:

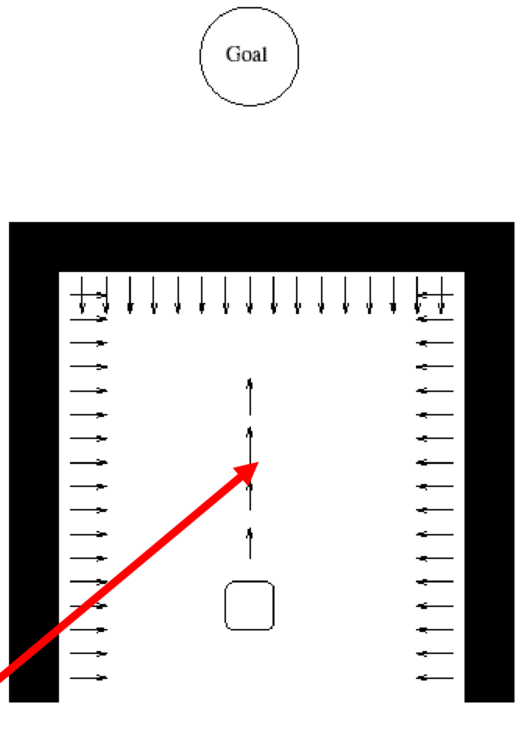


You start/end up here

Goal

Local Minima on the Potential Field: Getting Stuck

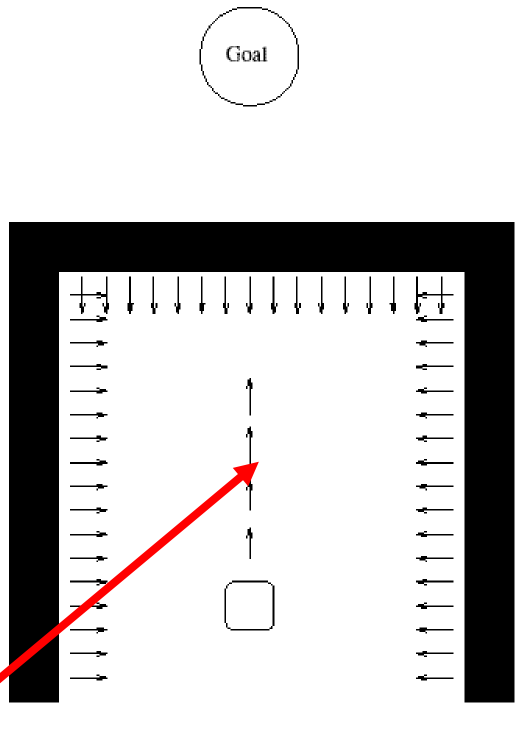
States of zero total force correspond to local minima in the potential function:



Problem: If you end up here gradient descent can't help you. All local moves seem identical in terms of value → local min

Local Minima on the Potential Field: Getting Unstuck Randomly

States of zero total force correspond to local minima in the potential function:

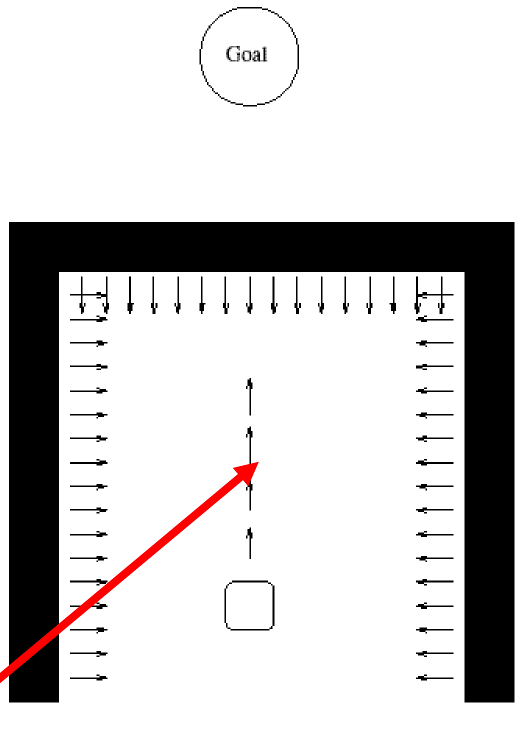


Solution #1: Do random move in case it helps you get unstuck.

Problem: If you end up here gradient descent can't help you. All local moves seem identical in terms of value → local min

Local Minima on the Potential Field: Getting Unstuck By Backing Up

States of zero total force correspond to local minima in the potential function:



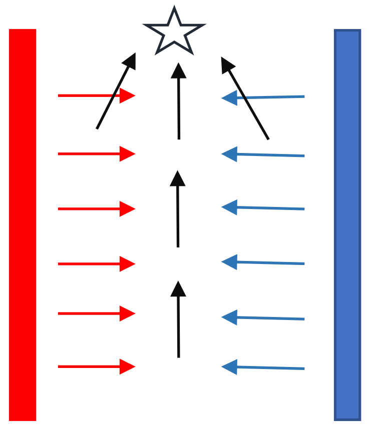
Solution #2: back up and get out from the dead end, just like you entered it.

Problem: If you end up here gradient descent can't help you. All local moves seem identical in terms of value → local min

Drawbacks of potential fields

- *Local minima*
  - Attractive and repulsive forces can balance, so robot makes no progress.
  - Closely spaced obstacles, or dead end.

- *Unstable oscillation*
  - The dynamics of the robot/environment system can become unstable.
  - High speeds, narrow corridors, sudden changes

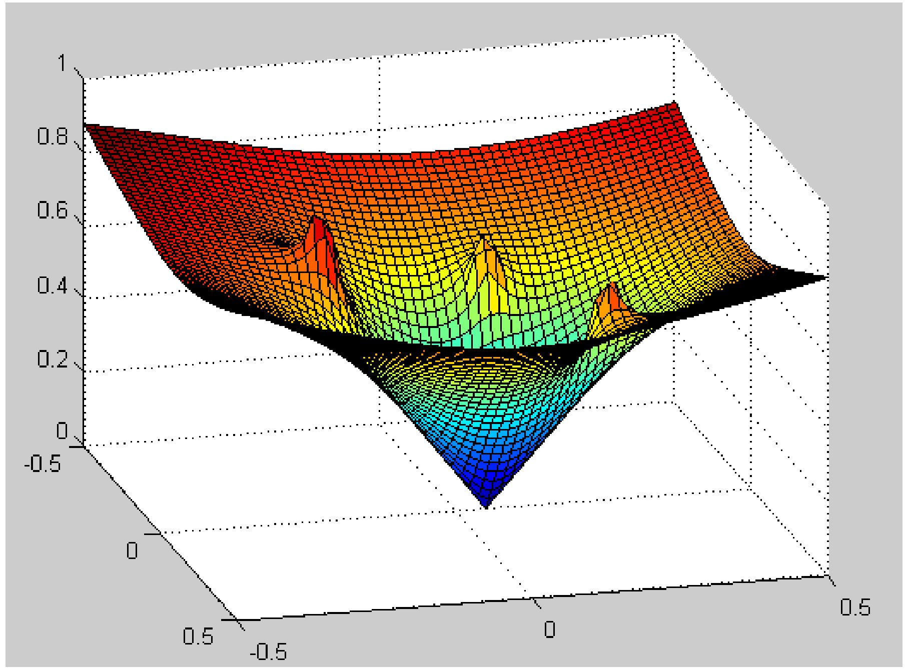


Avoiding Local Minima on the Potential Field: Navigation Functions

Potential energy function $\phi(x)$ with a **single global minimum** at the goal, and **no local minima.**

For any state x there exists a neighboring state x' such that $\phi(x') < \phi(x)$.

So far not used in practice very much because they are usually as hard to compute as a planned path from the current state to the goal.

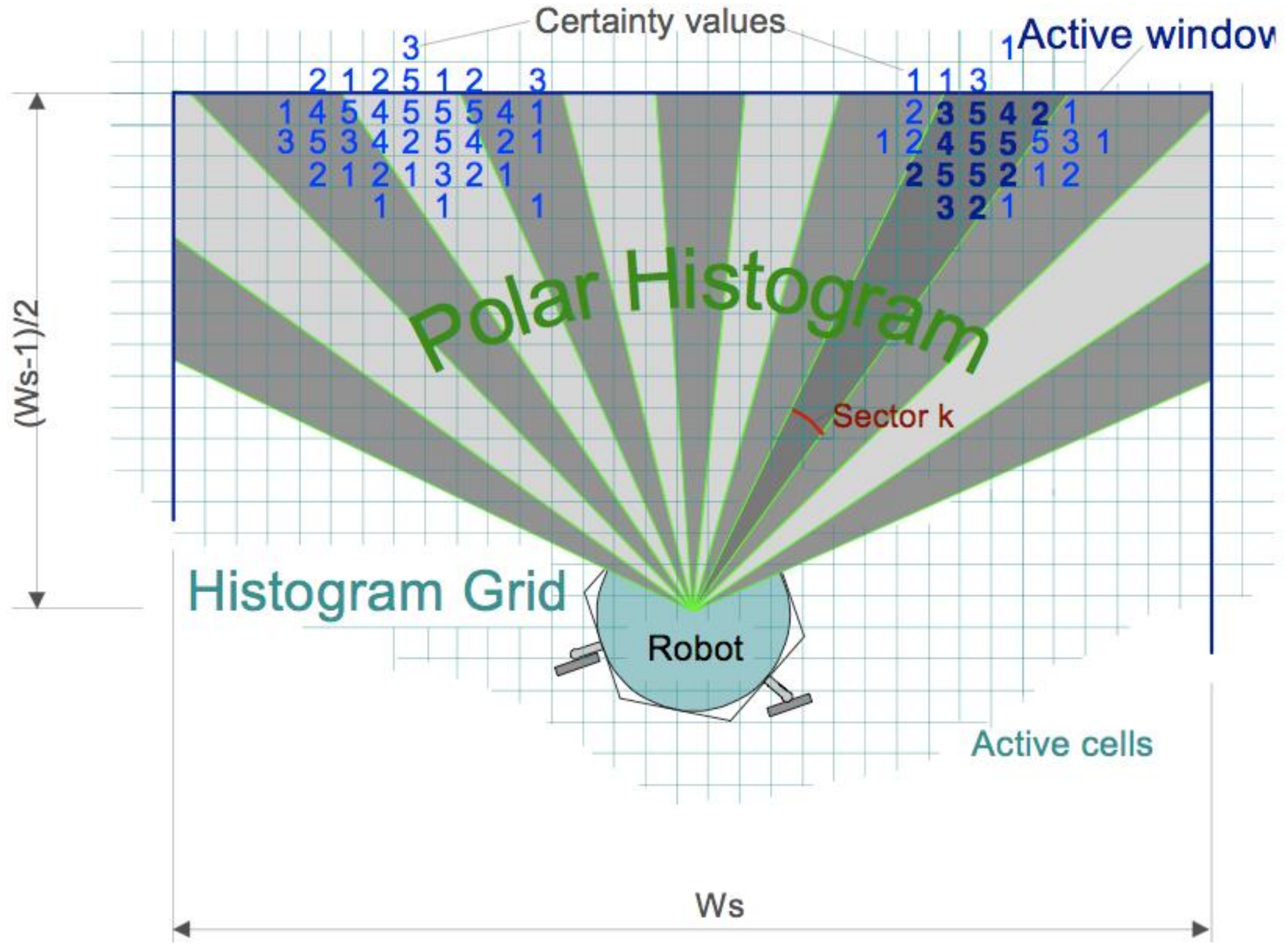


Addressing the Drawbacks of Potential Fields

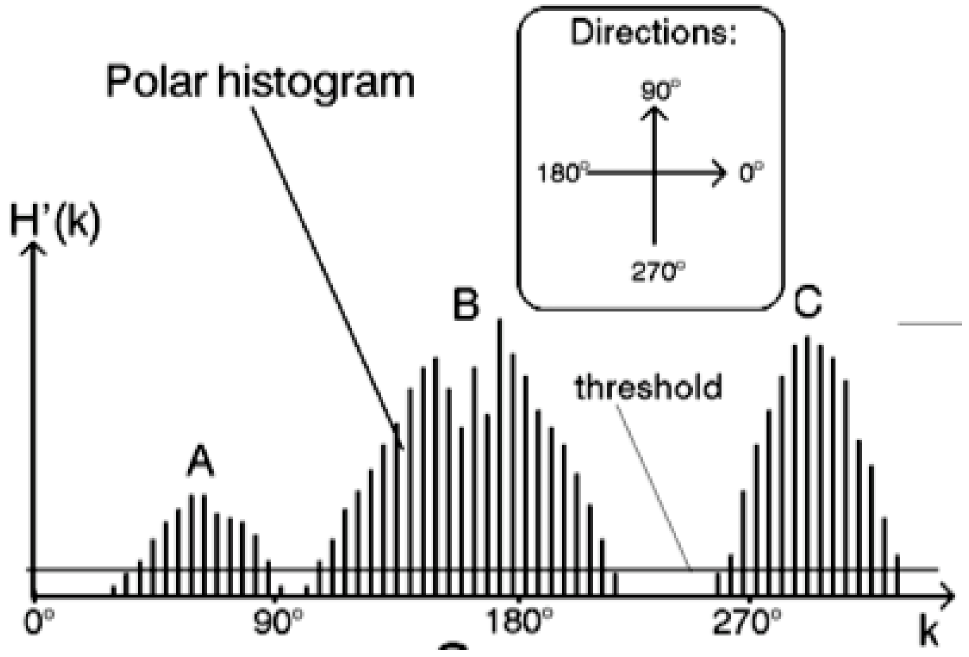
- Vector Field Histogram (VFH)
- Dynamic Window Approach

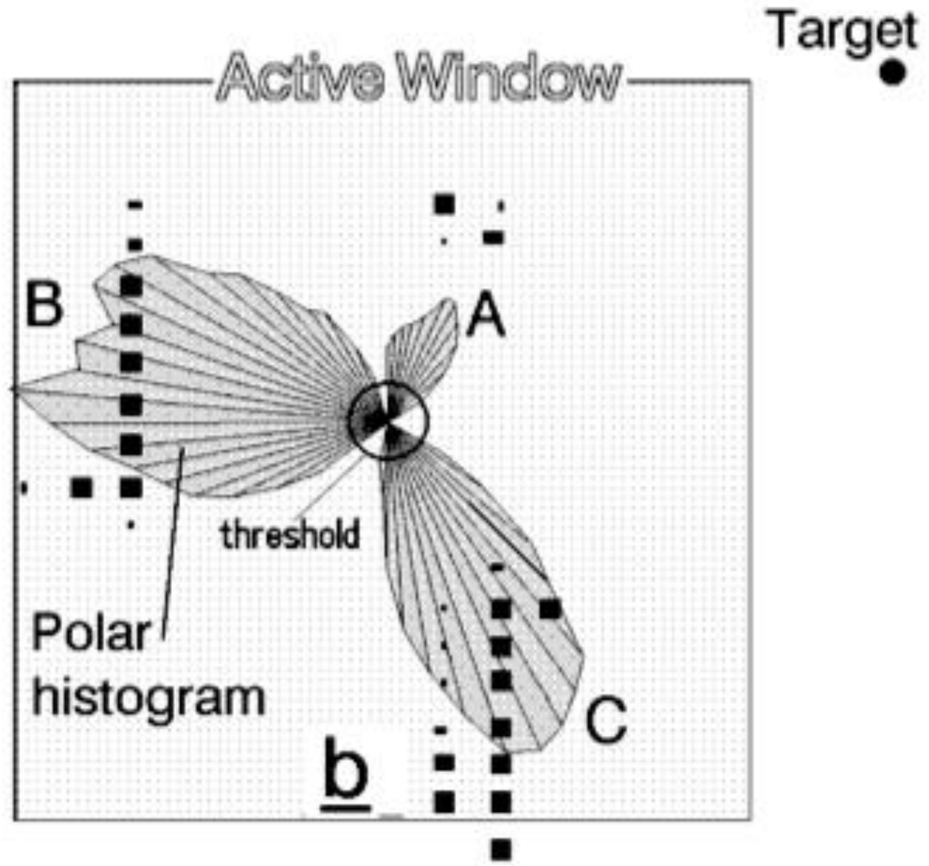
Both methods for local obstacle avoidance

VFH (Vector Field Histogram)



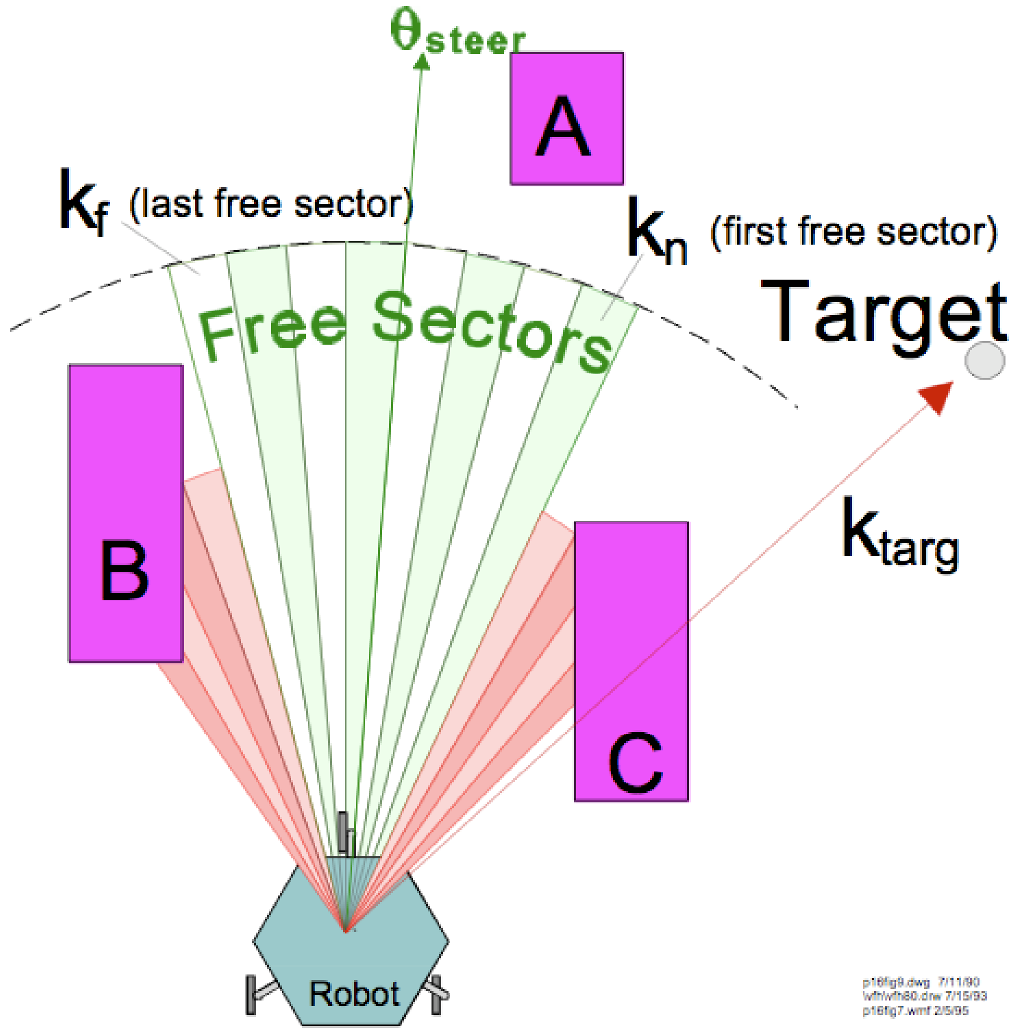
VFH (Vector Field Histogram)





High risk for cells with high probability of being occupied.
Risk inversely proportional to distance.

VFH (Vector Field Histogram)



VFH (Vector Field Histogram)

VFH (Vector Field Histogram)

DWA (Dynamic Window Approach)

Local, reactive controller

1. Sample a set of controls for x,y,theta
2. Simulate where each control is going to take the robot
3. Eliminate those that lead to collisions.
4. Reward those that agree with a navigation plan.
5. Reward high-speeds
6. Reward proximity to goal.
7. Pick control with highest score that doesn't lead to collision.