# Model Predictive Control and the Unreasonable Effectiveness of Replanning

Tapomayukh Bhattacharjee

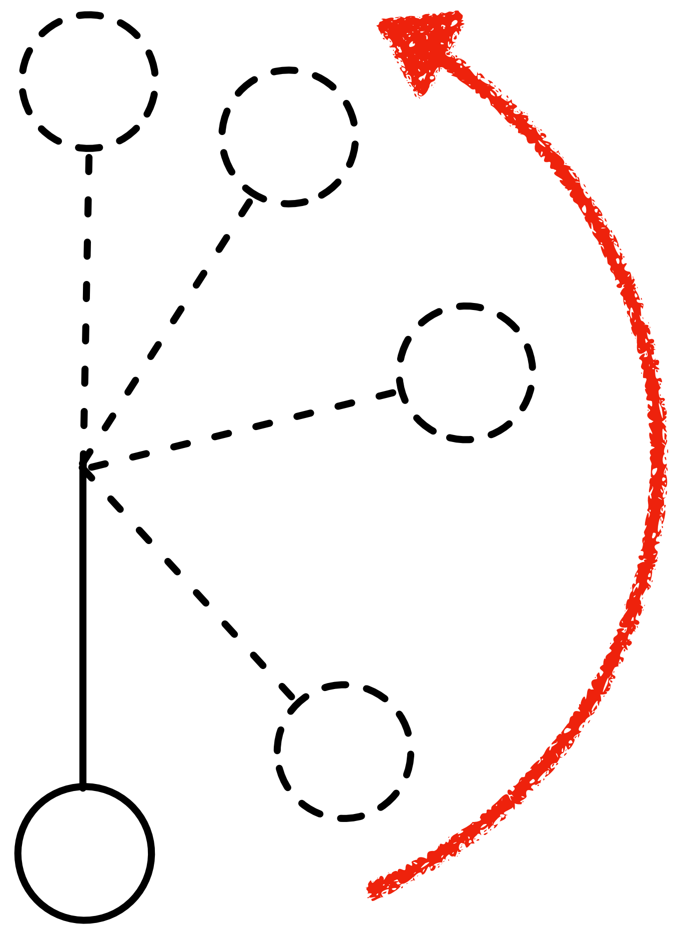
Cornell Bowers CIS
Computer Science

* Some slides from last year's CS 4756

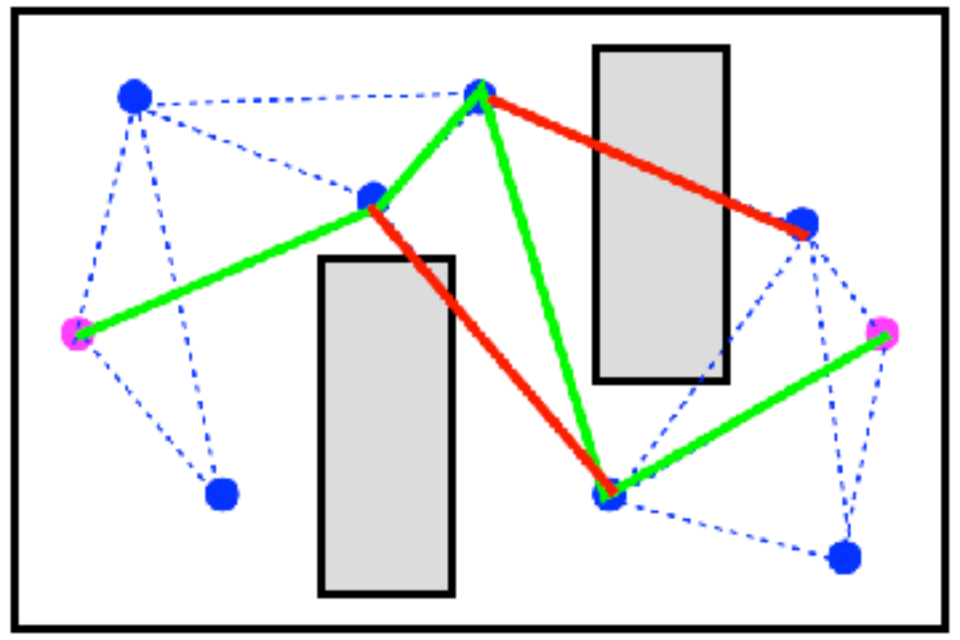
# Landscape of Planning / Control Algorithms

Low-level control

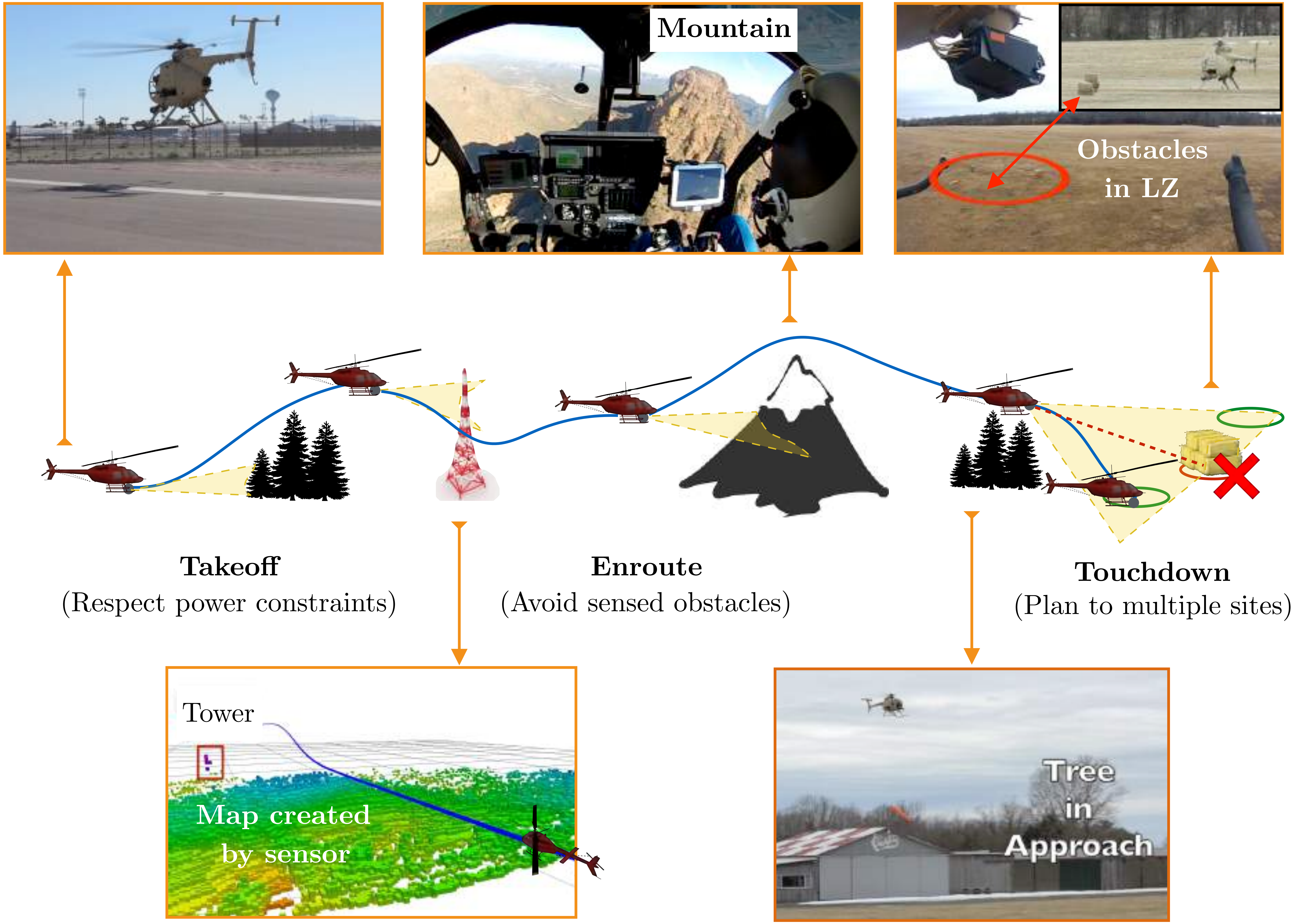
High-level path planning



LQR



LazySP


Mountain
Obstacles
in LZ
Takeoff
(Respect power constraints)
Enroute
(Avoid sensed obstacles)
Touchdown
(Plan to multiple sites)
Tower
Map created
by sensor
Tree
in
Approach

# Recap: Solving a MDP

$$\min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} c(s_t, a_t) \qquad s_{t+1} = \mathcal{T}(s_t, a_t)$$

*(Solve for a sequence of actions)*

*(Sum over all costs)*

*(Transition function)*

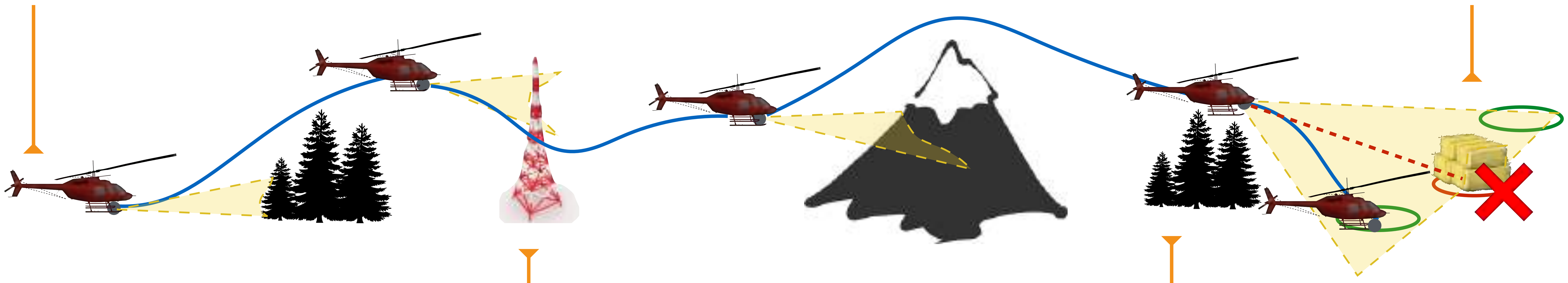
# Brainstorm: Challenges in solving MDP for helicopter

$$\min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} c(s_t, a_t) \qquad s_{t+1} = \mathcal{T}(s_t, a_t)$$

*(Solve for a sequence of actions)*

*(Sum over all costs)*

*(Transition function)*



**Takeoff**
(Respect power constraints)

**Enroute**
(Avoid sensed obstacles)

**Touchdown**
(Plan to multiple sites)

# The Big Challenges

Problem 1: Don't know the terrain ahead of time!

Problem 2: Don't have a perfect dynamics model!

Problem 3: Not enough time to plan all the way to the goal!

# The Big Challenges

Problem 1: Don't know the terrain ahead of time!

Problem 2: Don't have a perfect dynamics model!

Problem 3: Not enough time to plan all the way to the goal!

# Brainstorm!

Find a sequence of actions to go from start to goal.

The helicopter can only sense upto 1km.

How should it deal with unknown terrain? What assumptions can it make?

# What is the problem mathematically?

$$\min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} c(s_t, a_t) \qquad s_{t+1} = \mathcal{T}(s_t, a_t)$$

*(Solve for a sequence of actions)*

*(Sum over all costs)*

*(Transition function)*

Is the transition function fully known?

If not, then how can we solve the optimization problem?

# Idea: Plan with an optimistic model

$$\min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} c(s_t, a_t)$$

*(Solve for a sequence of actions)*

*(Sum over all costs)*

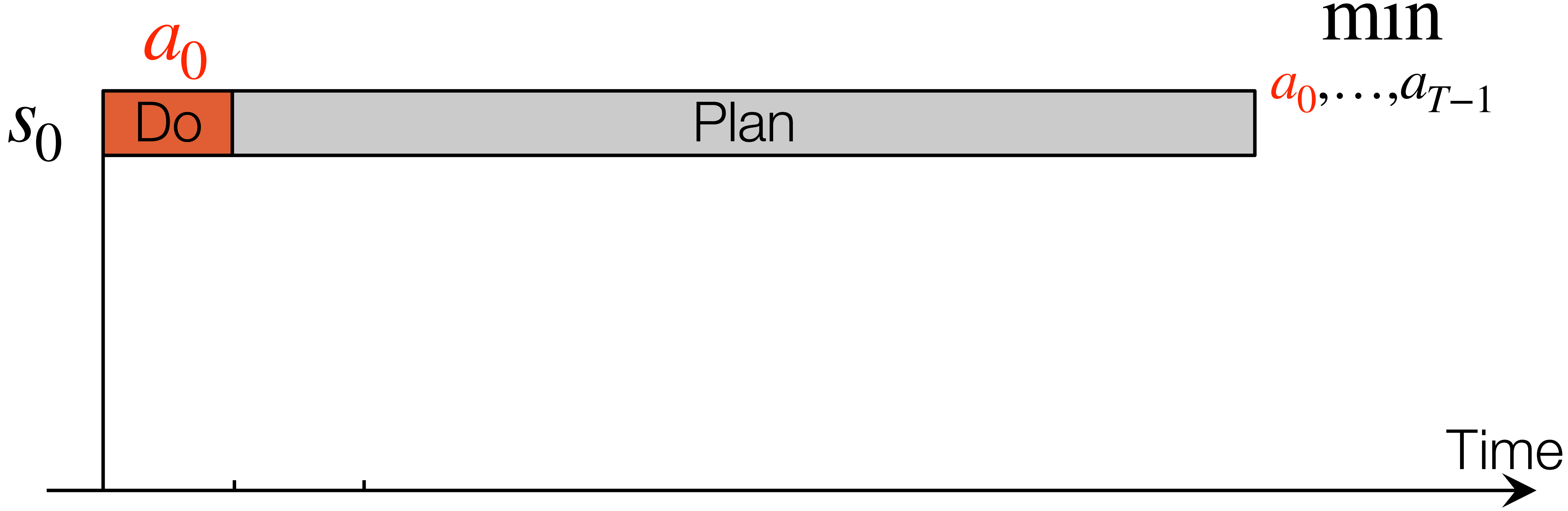
$$s_{t+1} = \hat{\mathcal{T}}(s_t, a_t)$$

*(Optimistic Model)*

Assume that any unknown space is fully traversable.

Update model as you get information from real world. Replan!

# Model Predictive Control (MPC)



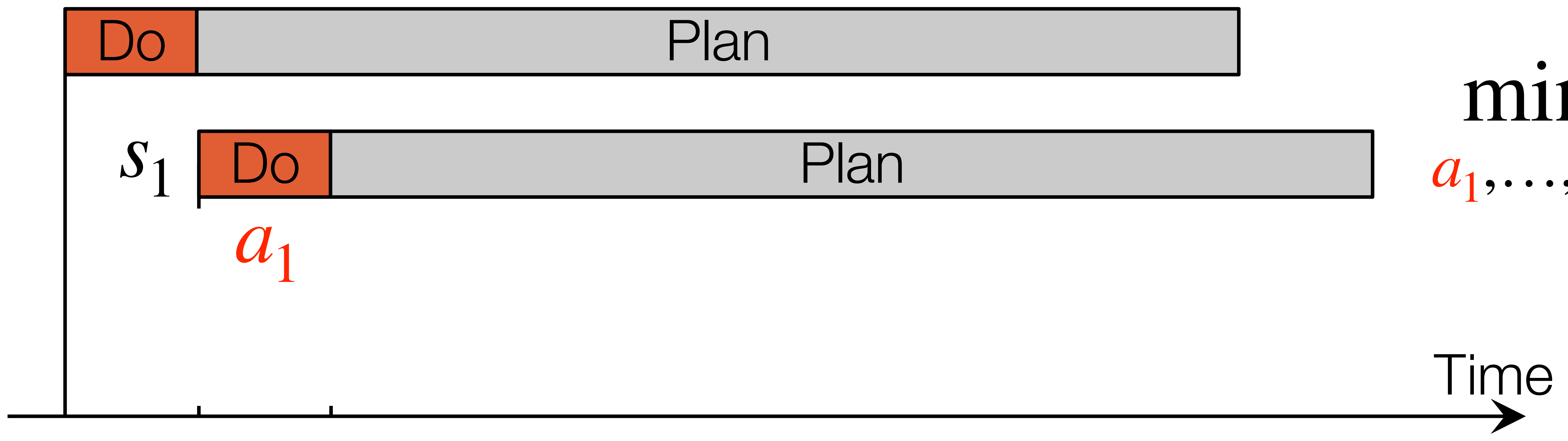

$$\min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} c(s_t, a_t)$$

Step 1: Solve current MDP (plan) to find a sequence of actions

Step 2: Execute the first action in the real world and update MDP

Step 3: Repeat!

# Model Predictive Control (MPC)



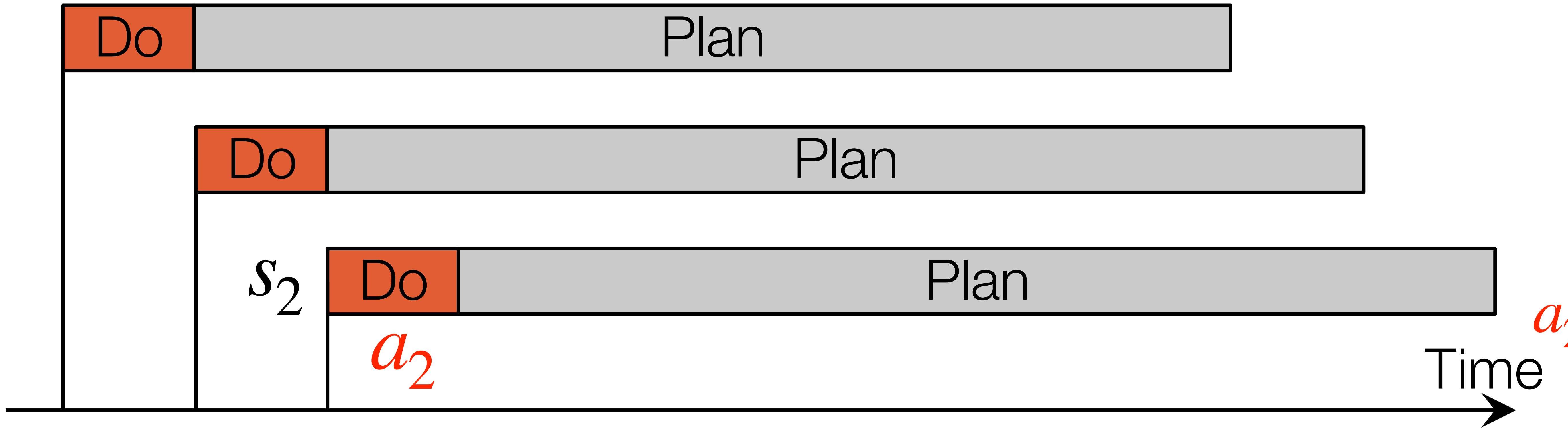

$$\min_{a_1,\ldots,a_T} \sum_{t=1}^{T} c(s_t, a_t)$$

Step 1: Solve current MDP (plan) to find a sequence of actions

Step 2: Execute the first action in the real world and update state

Step 3: Repeat!

# Model Predictive Control (MPC)




$$\min_{a_2,\ldots,a_{T+1}} \sum_{t=2}^{T+1} c(s_t, a_t)$$

Step 1: Solve current MDP (plan) to find a sequence of actions

Step 2: Execute the first action in the real world and update state

Step 3: Repeat!

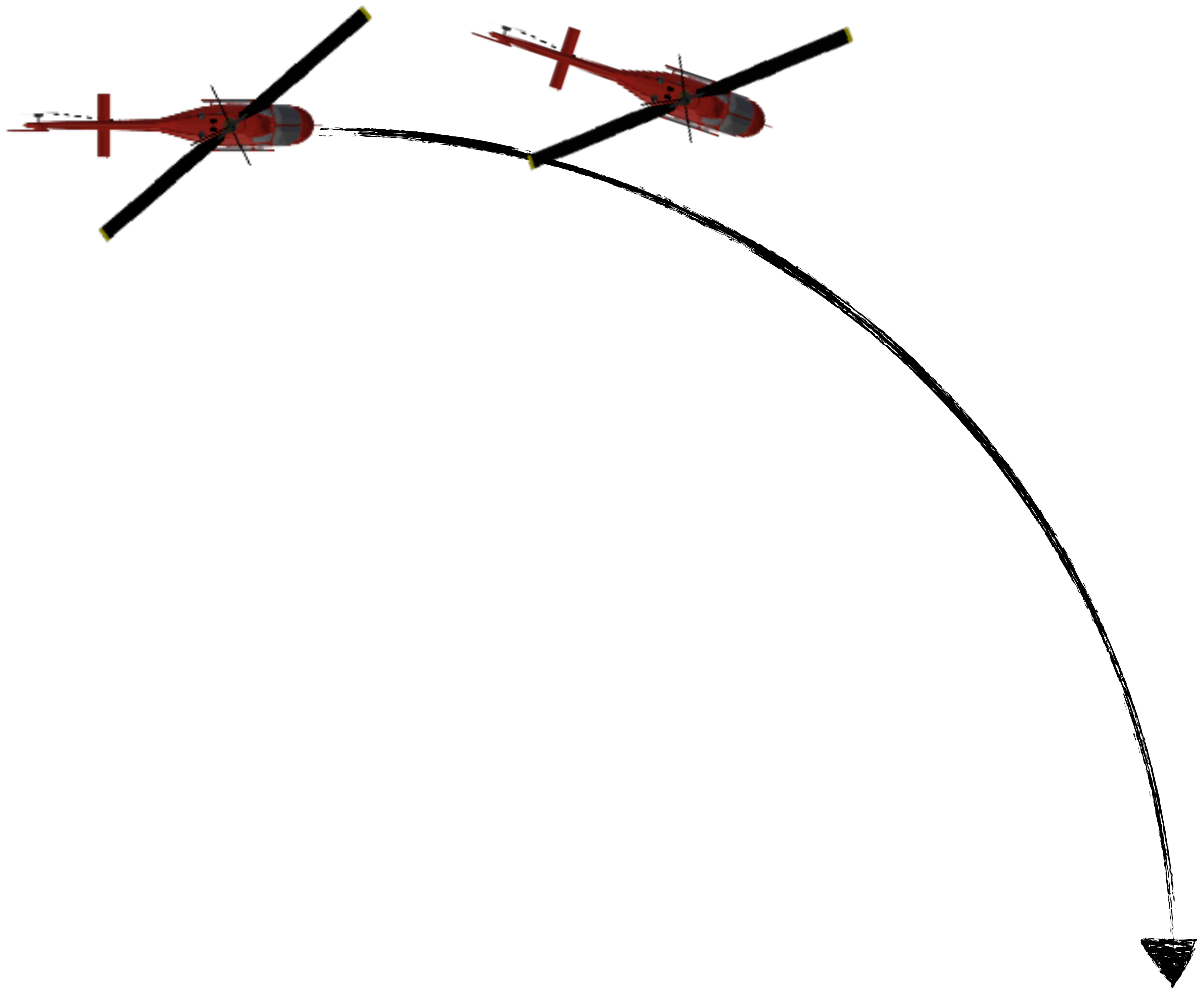
# The Big Challenges

Problem 1: Don't know the terrain ahead of time!

Problem 2: Don't have a perfect dynamics model!

Problem 3: Not enough time to plan all the way to the goal!

# Problem 2: Don't have a perfect dynamics model!



Let's say there is an unknown gust of wind pushing you off the path

# What is the problem mathematically?

$$\min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} c(s_t, a_t) \qquad s_{t+1} = \mathcal{T}(s_t, a_t)$$

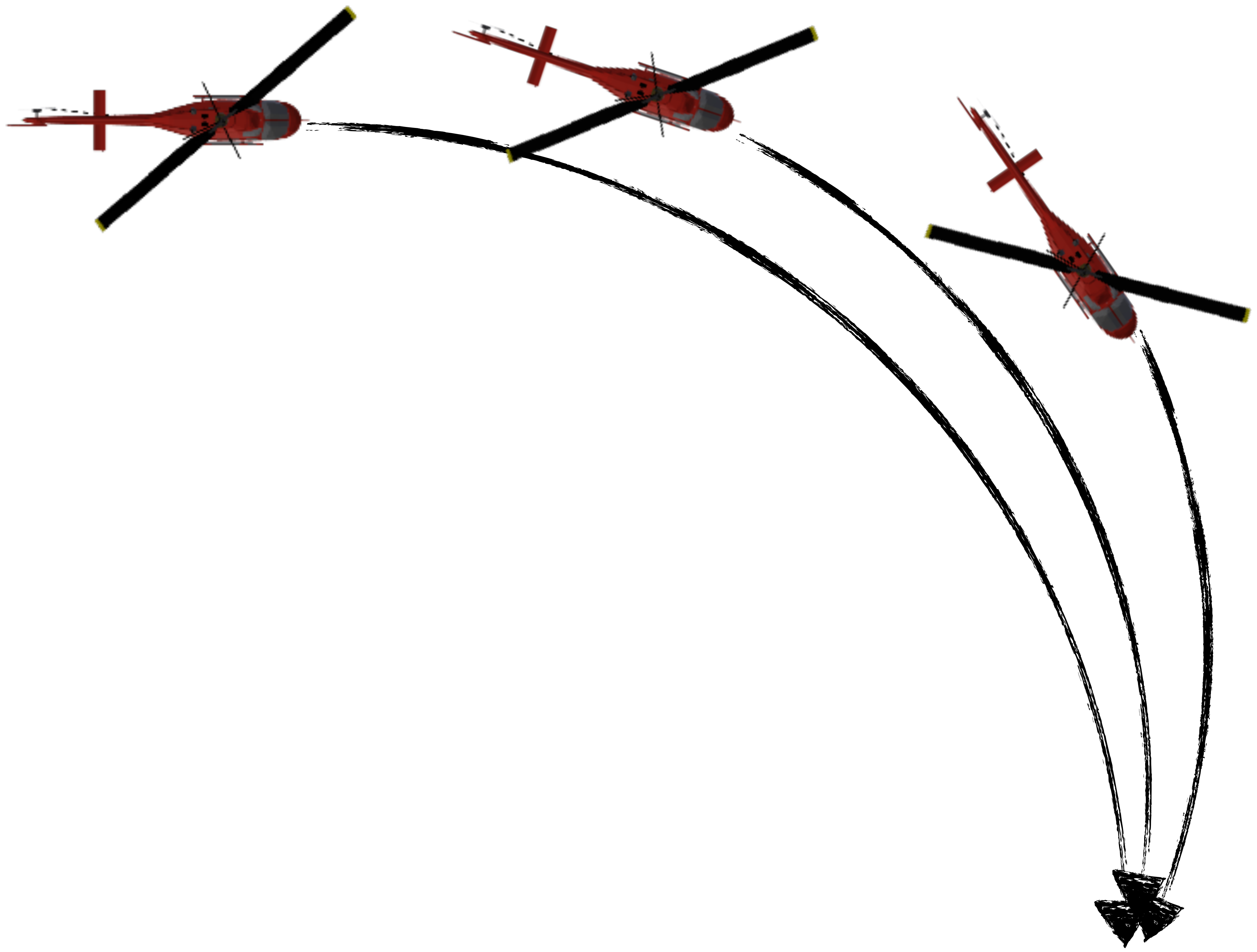
*(Solve for a sequence of actions)*

*(Sum over all costs)*

*(Transition function)*

Is the transition function fully known?

# Problem 2: Don't have a perfect dynamics model!



Plan with incorrect transition model and replan!

Theorem:
An optimal policy in an incorrect model has bounded suboptimality in the real model

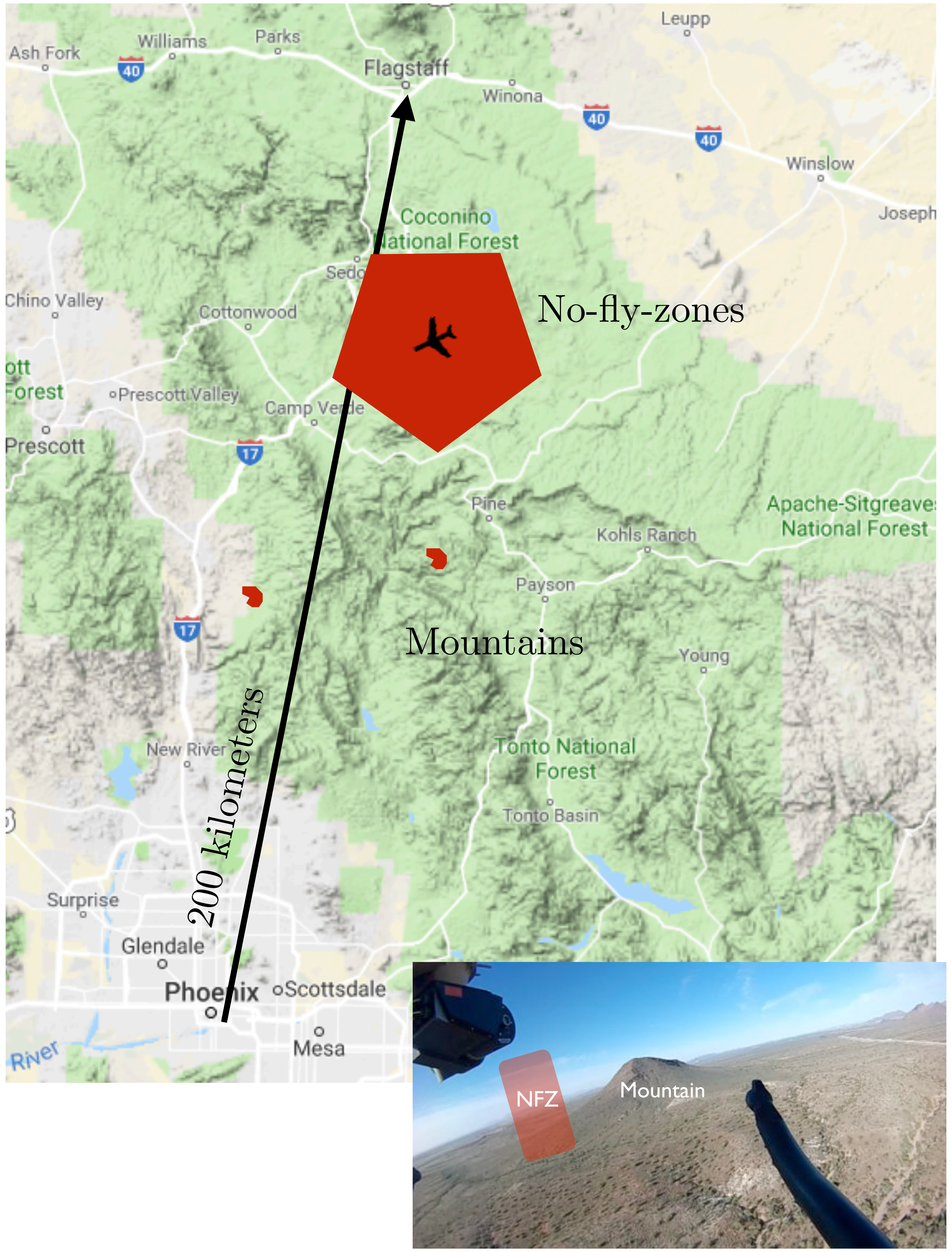
# The Big Challenges

Problem 1: Don't know the terrain ahead of time!

Problem 2: Don't have a perfect dynamics model!

Problem 3: Not enough time to plan all the way to the goal!

# Problem 3: Not enough time to plan all the way to goal!




## Example mission:

Fly from Phoenix to Flagstaff as fast as possible (200 km)

## Problem:

Take forever to plan at high resolution ALL the way to goal

# What is the problem mathematically?

$$\min_{a_0,\ldots,a_{T-1}} \sum_{t=0}^{T-1} c(s_t, a_t)$$

How large can T be?

*(Solve for a sequence of actions)*

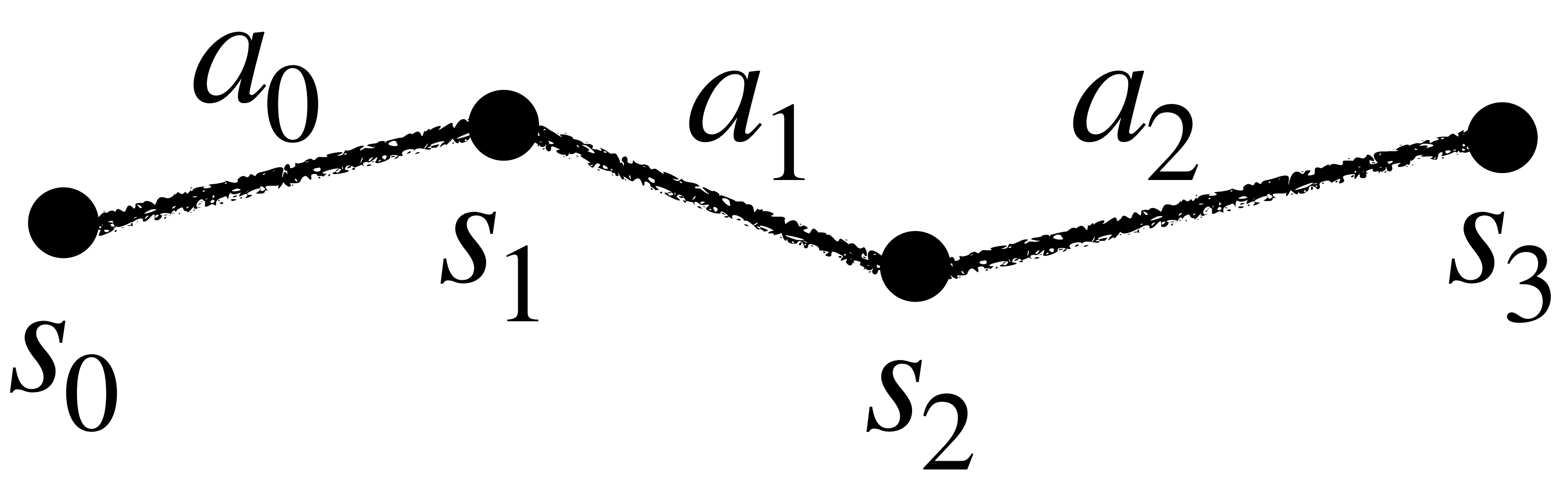
*(Sum over all costs)*

# What if we planned till a shorter time horizon T'?

$$\min_{a_0,\dots,a_{T'-1}} \sum_{t=0}^{T'-1} c(s_t, a_t)$$

*(Solve for a sequence of actions)*

*(Sum over all costs)*




Is this even allowed???

Would we get the same solution for $a_0$?

# We have to add in a terminal value for the final state

$$\min_{a_0,\ldots,a_{T'-1}} \sum_{t=0}^{T'-1} c(s_t, a_t) \quad + V^\star(s'_T)$$

*(Solve for a sequence of actions)*

*(Sum over all costs)*

*(Optimal value of state $s_{T'}$ )*

Can we compute the optimal value V*?

If not, how can we approximate it

# Idea: Use a global planner to approximate $\hat{V}^\star$

$$\min_{a_0,\ldots,a_{T'-1}} \sum_{t=0}^{T'-1} c(s_t, a_t) \quad + \hat{V}^\star(s'_T)$$

*(Solve for a sequence of actions)*

*(Sum over all costs)*

*(Approximate value of state $s_{T'}$ )*

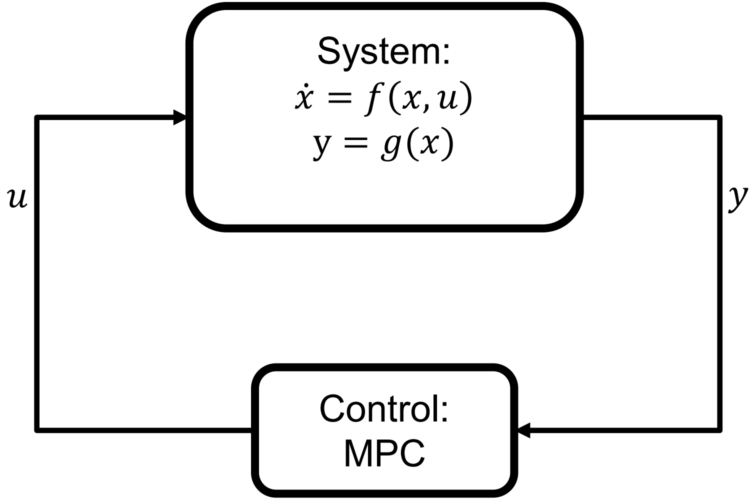
For example: Run a 2D planner from $s_T$ to the goal

Use the cost of that plan to compute approximate value

# MPC: Key idea

MPC is an optimization-based method for feedback control

Follows the idea of optimizing for the next control **u** by reasoning about the system states over a time window i.e. horizon **T**

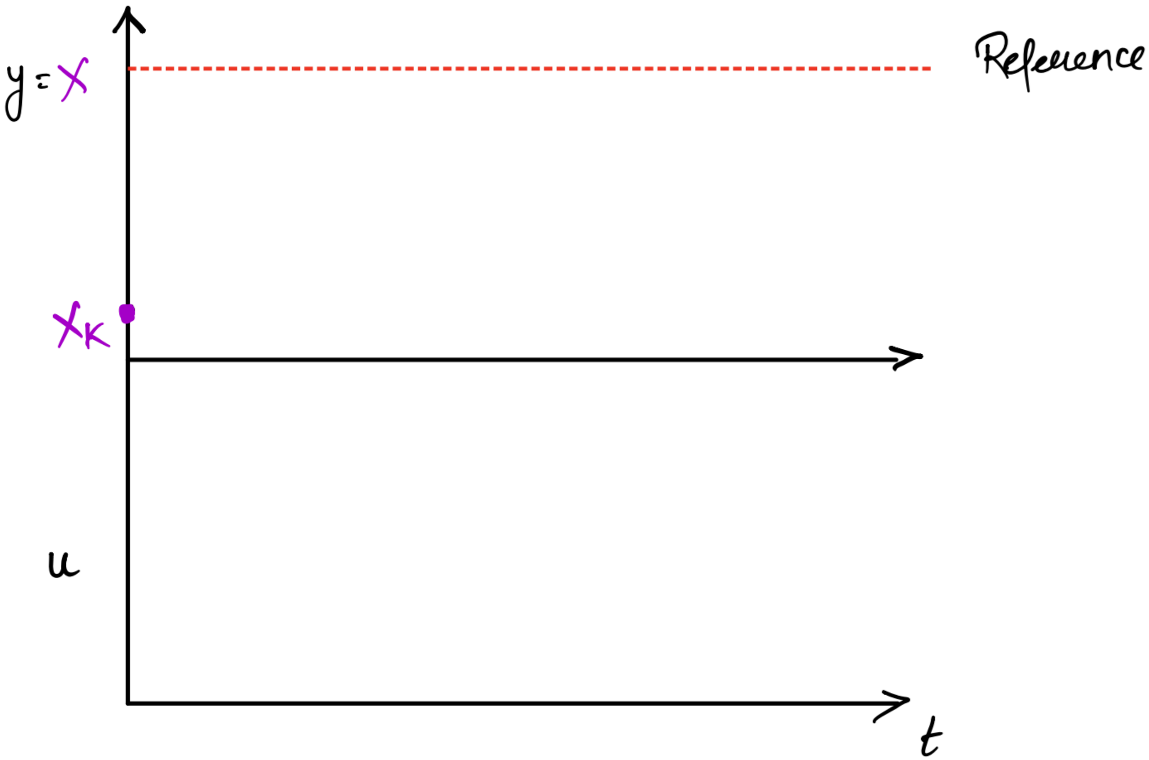


# Stepping back, a bit of history…

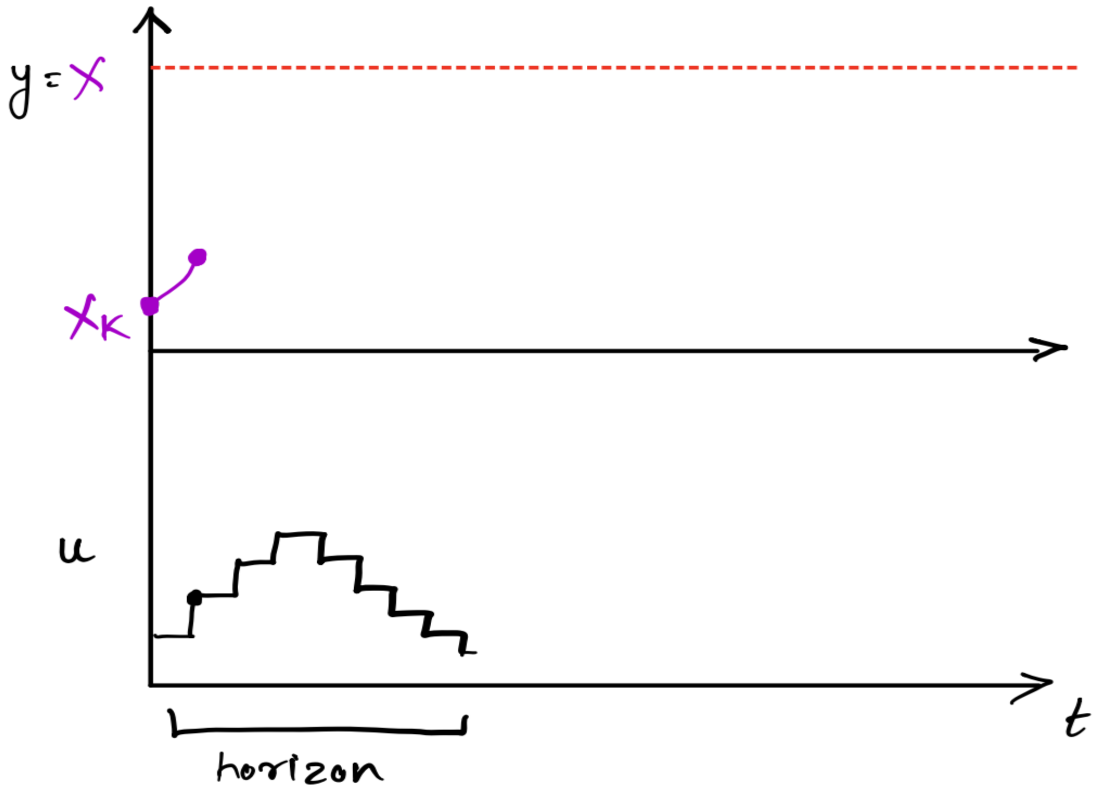
- Originally developed independently in the 1970's by two pioneering industrial research groups (Dynamic Matrix Control by Shell Oil and ADERSA)

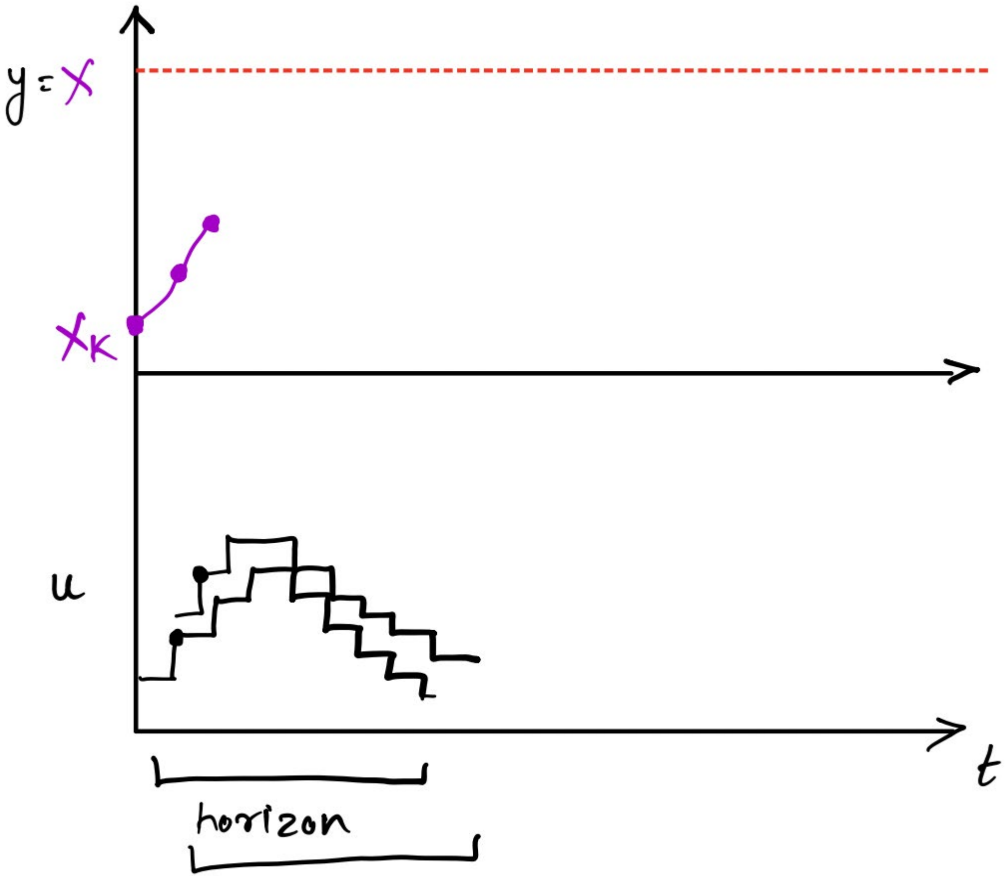
- By 1999, 4500 different application domains world-wide!

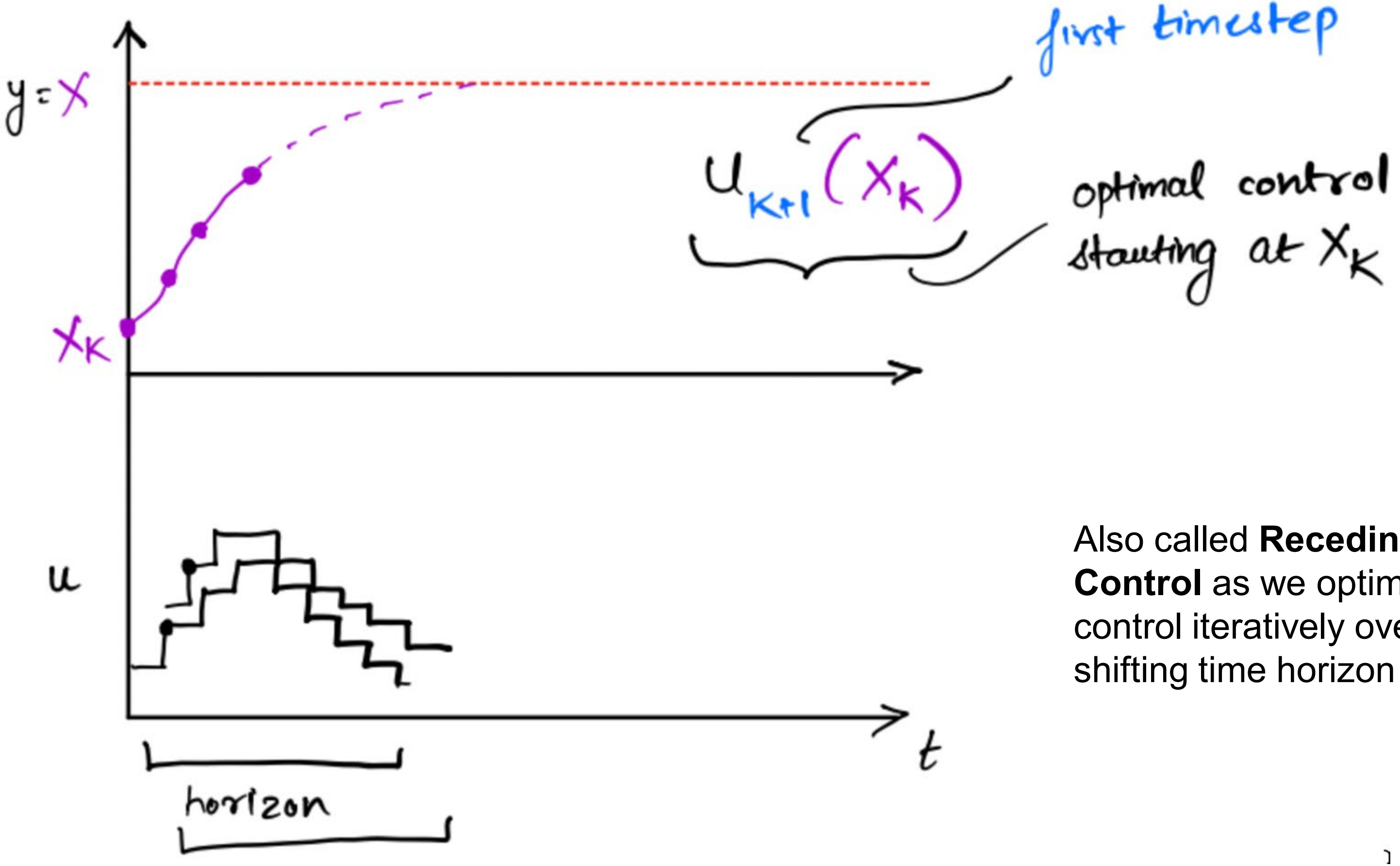
- Was primarily used in oil refineries and petrochemical plants, then in chemical, pulp and paper, before being used widely in robotics



y = X
Reference
$X_K$
u
t


y = x
x_K
u
t
horizon


y = x
x_k
u
t
horizon




Also called **Receding Horizon Control** as we optimize for next control iteratively over a forward-shifting time horizon

MPC primarily involves three main components:

MPC primarily involves three main components:

**Dynamics model** / **Process model**: Informs about the possible future states of the system as well as the constraints associated with it

MPC primarily involves three main components:

**Dynamics model** / **Process model**: Informs about the possible future states of the system as well as the constraints associated with it

**Objective function** / **Cost function**: Allows us to specify the behavior we expect for our system given the potential future states informed by the dynamics model

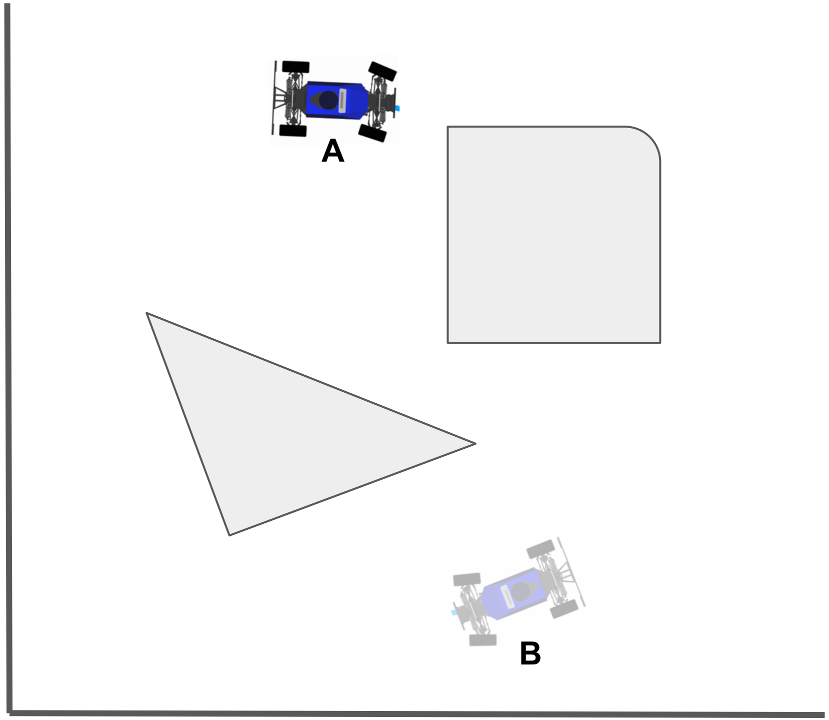
MPC primarily involves three main components:

**Dynamics model** / **Process model**: Informs about the possible future states of the system as well as the constraints associated with it

**Objective function** / **Cost function**: Allows us to specify the behavior we expect for our system given the potential future states informed by the dynamics model

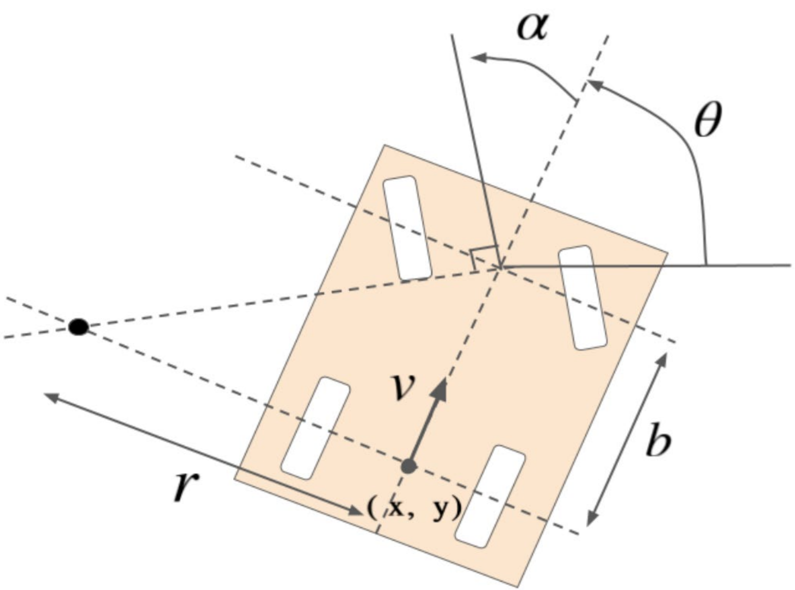
**Optimization algorithm**: Used to solve for next control given the objective function

Consider the scenario where a car needs to navigate from point A to B in a map filled with obstacles



# Dynamics Model

# Dynamics Model



# Dynamics Model

$$\dot{x} = v\cos\theta$$

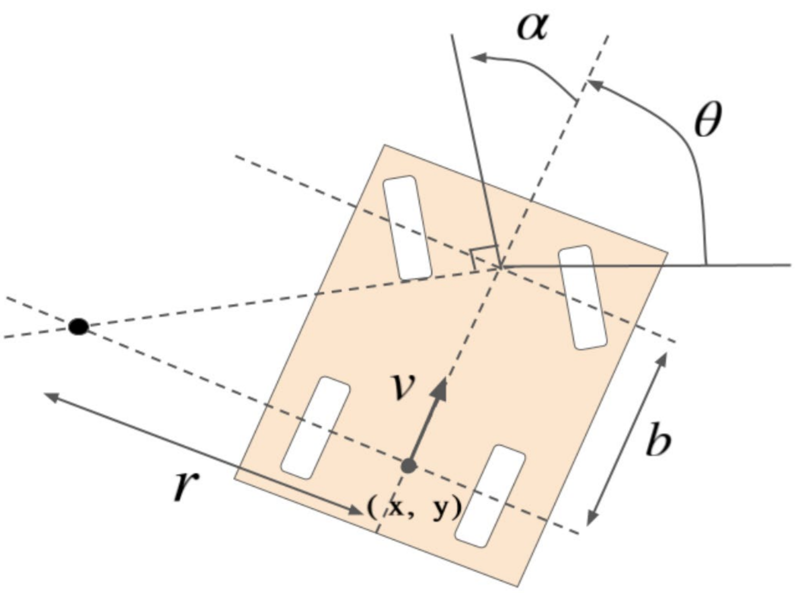
$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \omega$$

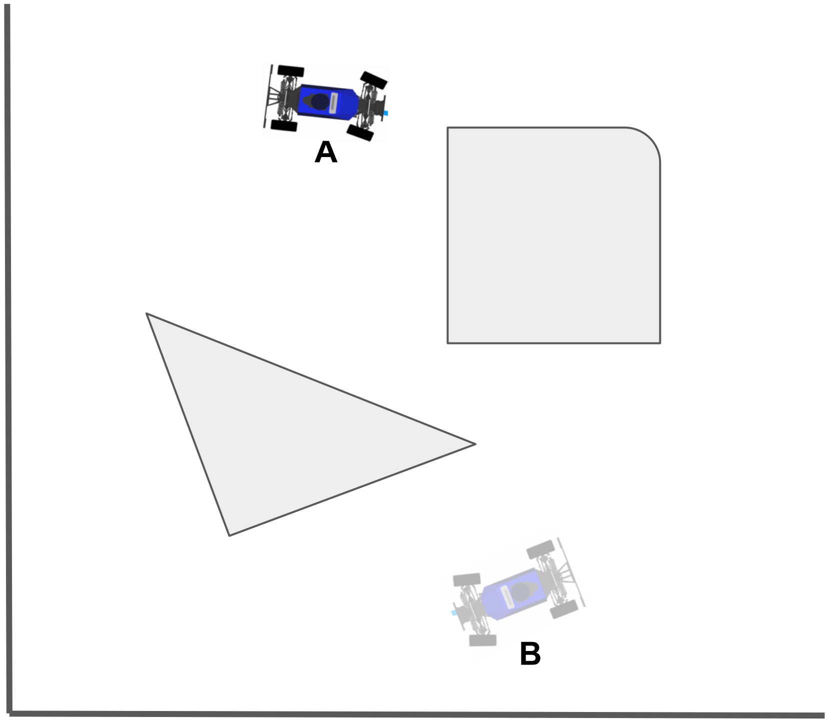
$$\theta_{t+1} = \theta_t + \frac{v}{b}\tan\alpha\Delta t$$

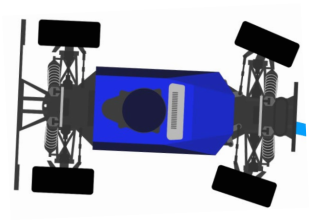
$$x_{t+1} = x_t + \frac{b}{\tan\alpha}[\sin\theta_{t+1} - \sin\theta_t]$$

$$y_{t+1} = y_t + \frac{b}{\tan\alpha}[-\cos\theta_{t+1} + \cos\theta_t]$$

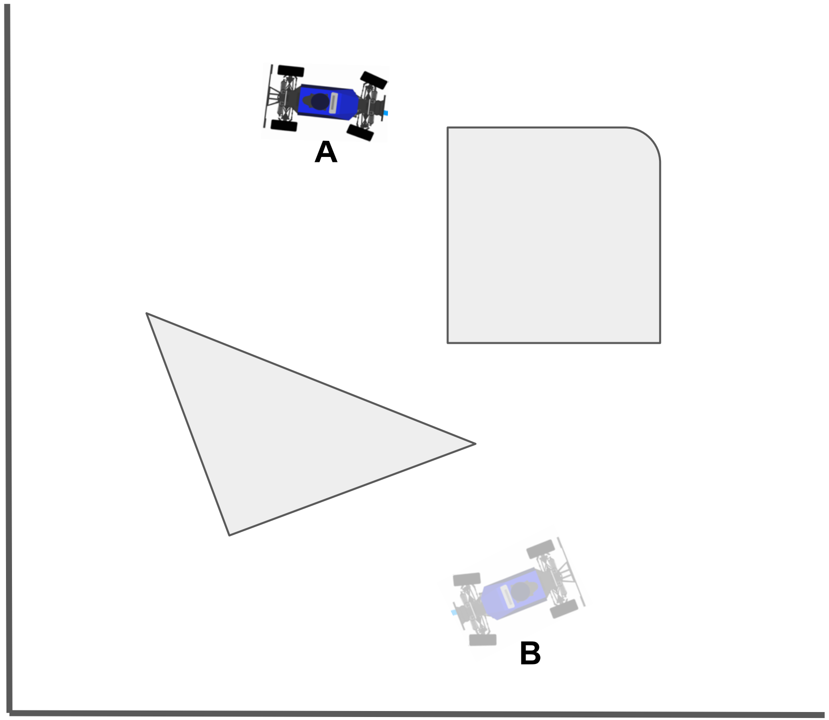


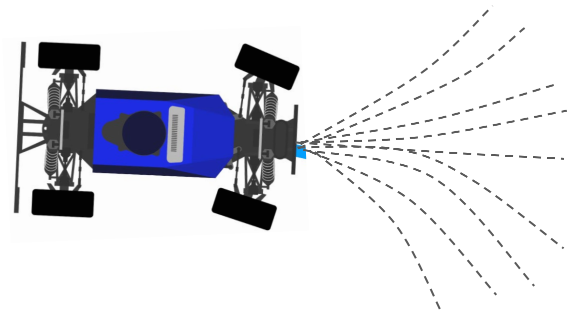
# Dynamics Model



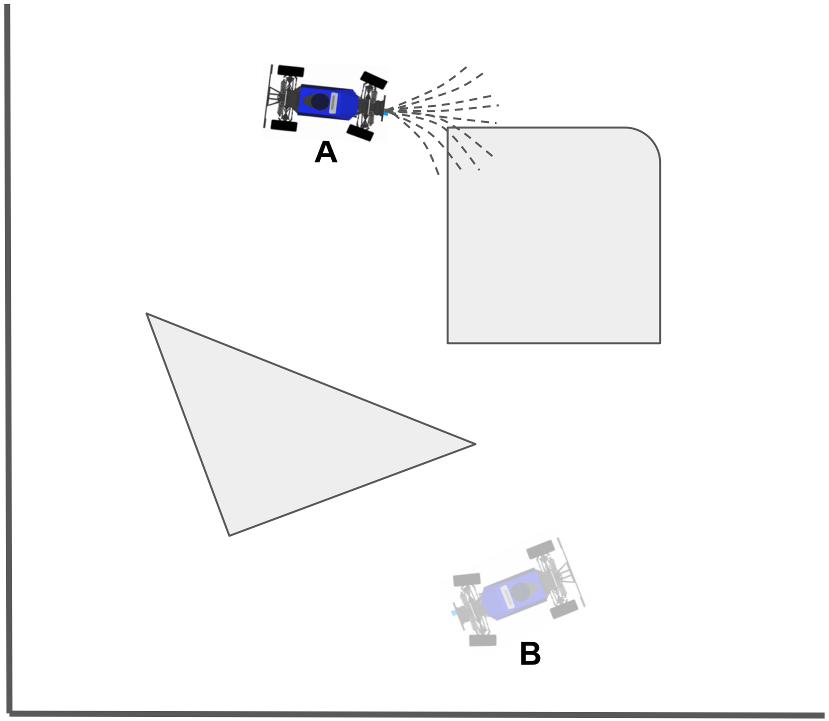



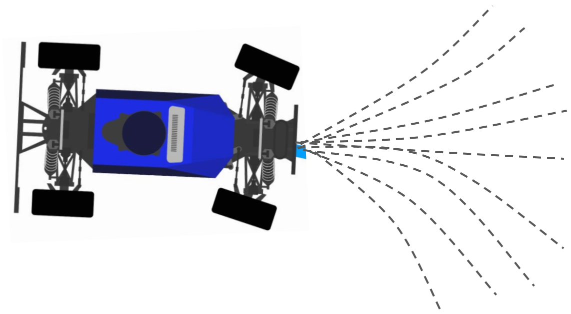
# Dynamics Model



# Dynamics Model



# Objective Function

# Objective Function

You want to move closer to the goal with every action you take

# Objective Function

You want to move closer to the goal with every action you take

If $\boldsymbol{x}^k{}_{\boldsymbol{T\text{-}1}}$ is the last state of rollout ‘k‘ and $\boldsymbol{x}_{\mathbf{goal}}$ is the goal state (use the lookahead distance to retrieve this), then

$$J^k_{dist} = \left\| x^k_{T-1} - x_{goal} \right\|_2^2$$

# Objective Function

You want to move closer to the goal with every action you take

If $\boldsymbol{x}^k{}_{T\text{-}1}$ is the last state of rollout 'k' and $\boldsymbol{x}_{\mathbf{goal}}$ is the goal state (use the lookahead distance to retrieve this), then

$$J^k_{dist} = \left\| x^k_{T-1} - x_{goal} \right\|_2^2$$

But you also want to avoid collisions

# Objective Function

You want to move closer to the goal with every action you take

If $\boldsymbol{x}^{k}{}_{T\text{-}1}$ is the last state of rollout 'k' and $\boldsymbol{x}_{\mathbf{goal}}$ is the goal state (use the lookahead distance to retrieve this), then

$$J_{dist}^{k} = \left\| x_{T-1}^{k} - x_{goal} \right\|_2^2$$

But you also want to avoid collisions and with weights for the respective costs

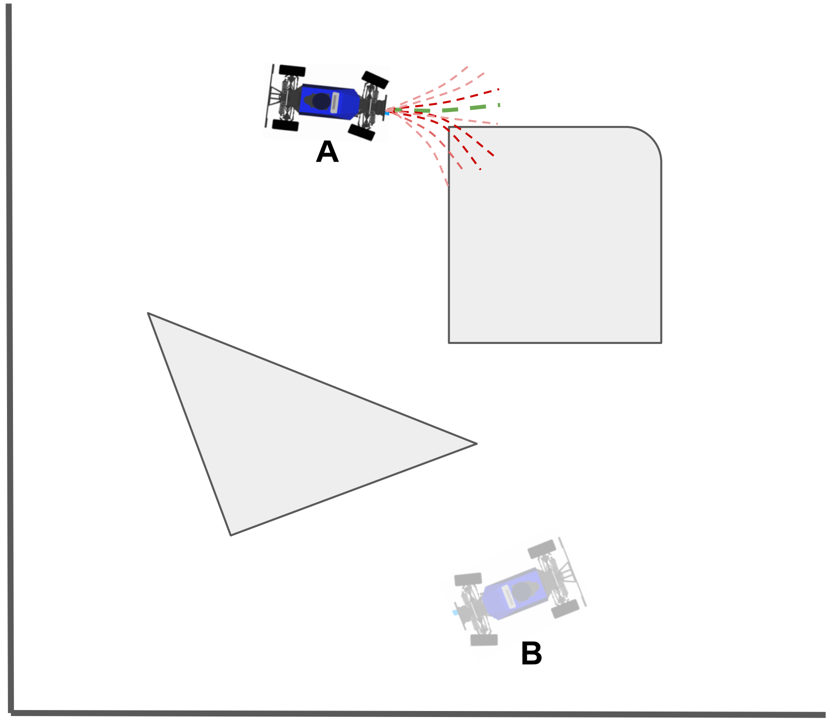
$$J_{total}^{k} = W_{dist} * J_{dist}^{k} + W_{collision} * J_{collision}^{k}$$

# Optimization Algorithm

# Optimization Algorithm

$\min J_{total}^{k}$

Obtain rollout with least cost using **argmin**



# Optimization Algorithm

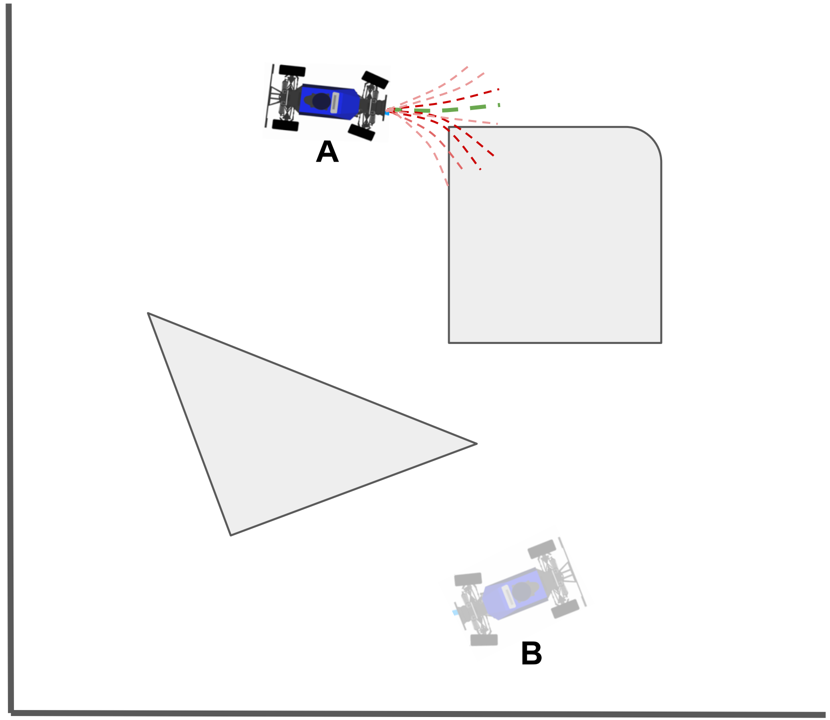
$\min \; J^{k}_{total}$

Obtain rollout with least cost using **argmin**

Alternatively,

Specify constraints for different state and control variables, use [non-linear programming (NLP) solver]

- For state variables, provide range of the state occupied by obstacles i.e. collision-free space



# MPC: Advantages

# MPC: Advantages

- Offers more flexibility
  - If system doesn't follow model closely -- we anyway reinitialize the optimization at every timestep

# MPC: Advantages

- Offers more flexibility
  - If system doesn’t follow model closely -- we anyway reinitialize the optimization at every timestep
- Allows you to impose constraints
  - Using the model and as well as during optimization
  - What kind of constraints during optimization?

# MPC: Advantages

- Offers more flexibility
  - If system doesn't follow model closely -- we anyway reinitialize the optimization at every timestep
- Allows you to impose constraints
  - Using the model and as well as during optimization
  - What kind of constraints during optimization?
- Works for non-linear systems

# MPC: Advantages

- Offers more flexibility
  - If system doesn’t follow model closely -- we anyway reinitialize the optimization at every timestep
- Allows you to impose constraints
  - Using the model and as well as during optimization
  - What kind of constraints during optimization?
- Works for non-linear systems
- Scope for curating task-specific controllers using task-specific objective functions

# MPC: Disadvantages

# MPC: Disadvantages

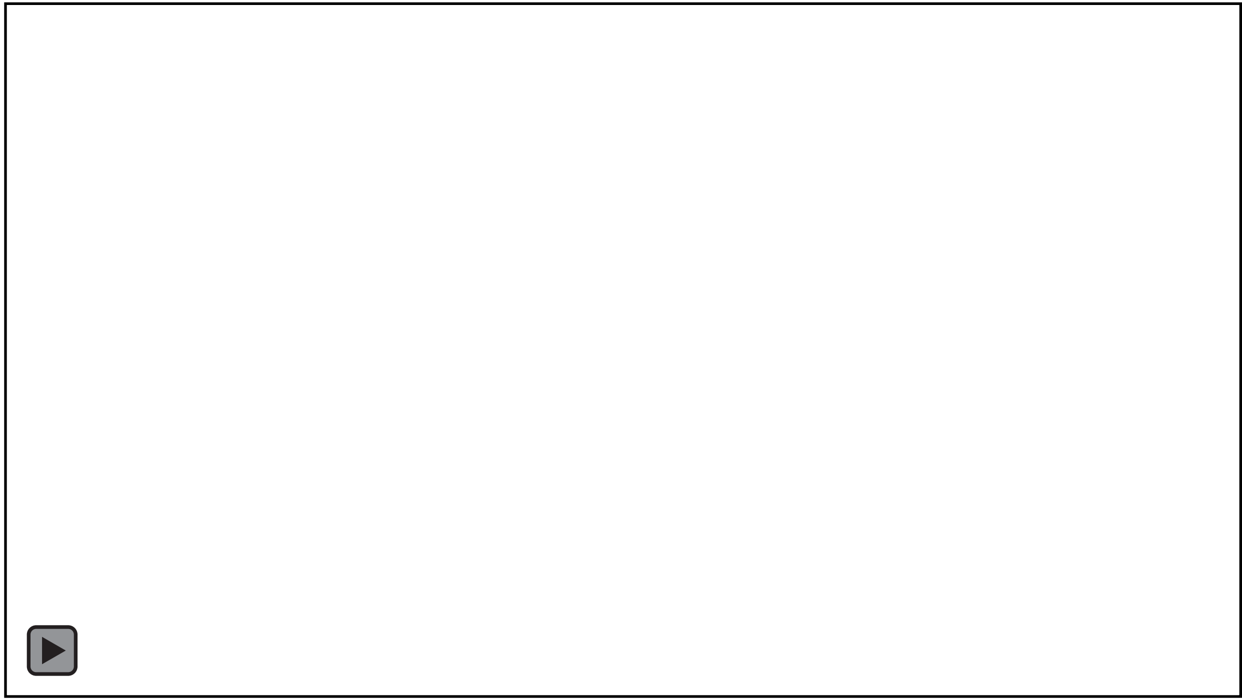
- Needs a model and needs a good one!

# MPC: Disadvantages

- Needs a model and needs a good one!
- Expensive since we are reinitializing at every timestep
    - Less of a problem as hardware is getting better

# Learning-based Model Predictive Control for Autonomous Racing

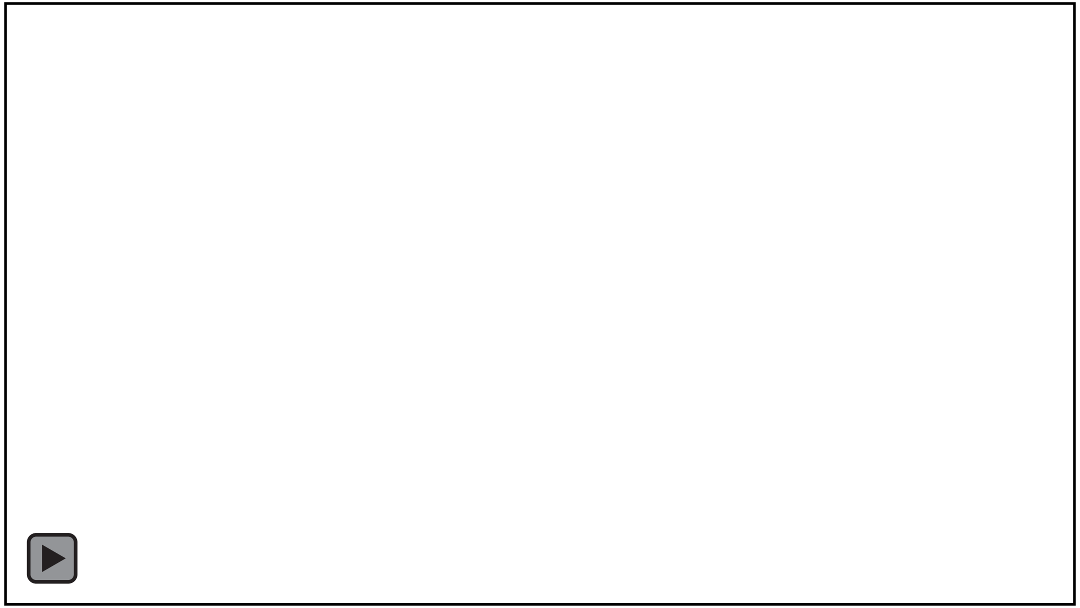
Initializes a simple bicycle model as the model and learns its parameters to improve controller



# Deep Haptic Model Predictive Control for Robot-Assisted Dressing

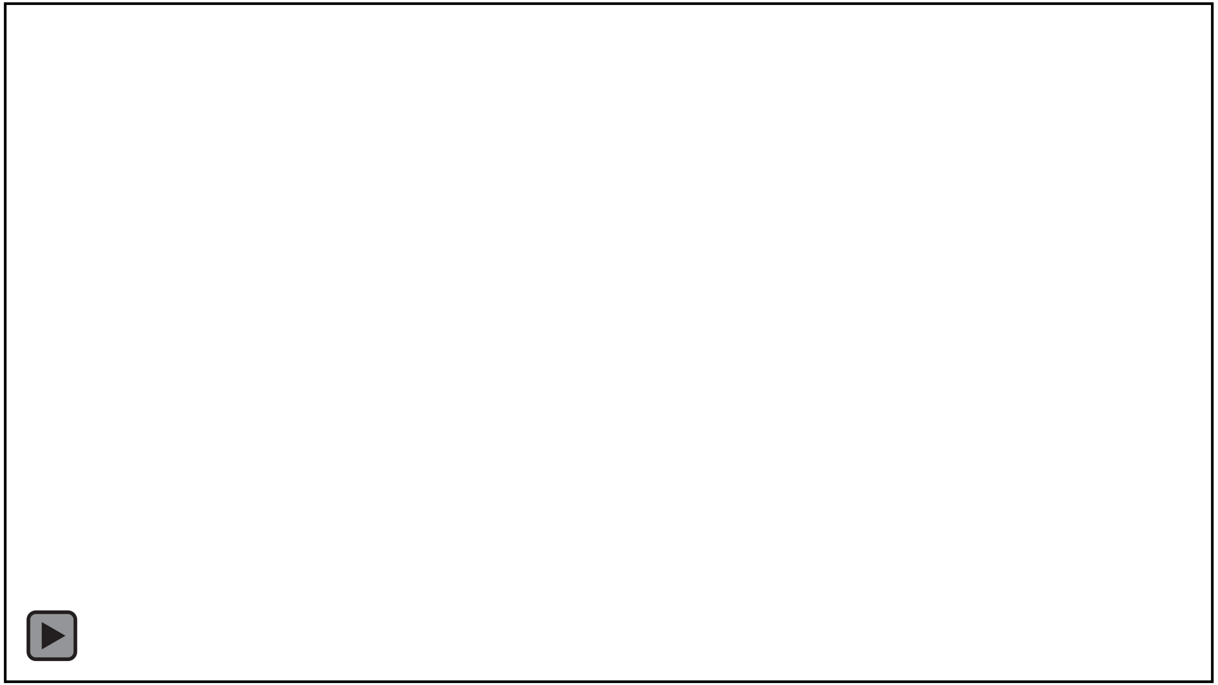
Trains neural networks to learn force being applied by the cloth on the human’s arm.

Uses them to choose actions that minimize predicted force applied during assistance.



# Model Predictive Contouring Control for Near-Time-Optimal Quadrotor Flight

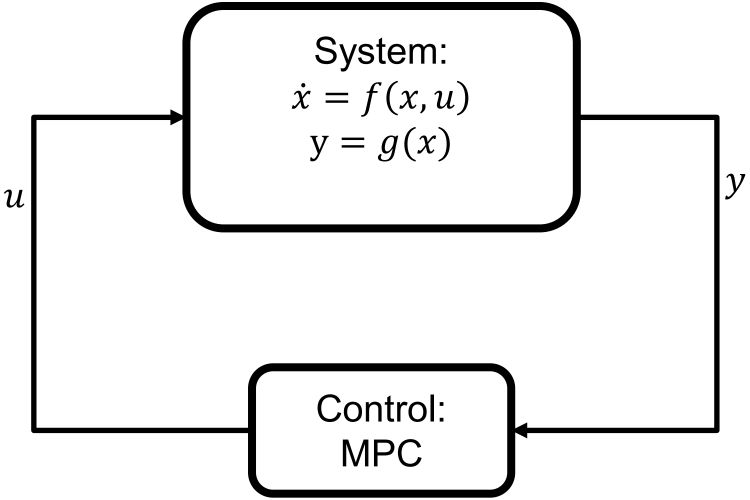
Instead of just performing reference state tracking, considers the higher-level task of minimizing Euclidean distance to a continuously differentiable 3D path while maximizing the speed at which the path is traversed.



# MPC: Key idea

MPC is an optimization-based method for feedback control

Follows the idea of optimizing for the next control **u** by reasoning about the system states over a time window i.e. horizon **T**



# Model Predictive Control and the Unreasonable Effectiveness of Replanning

Tapomayukh Bhattacharjee

Cornell Bowers CIS
Computer Science