# Learning Model Predictive Control for Iterative Tasks

## Theory and Applications

Ugo Rosolia
Principal Research Scientist @ Lyric.tech

Work (mostly) done at UC Berkeley and Caltech

December 18th, 2025

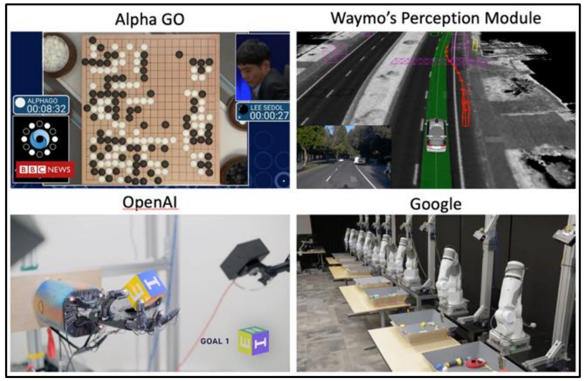
# Classic Approaches VS AI-based Strategies

# Classic Approaches VS AI-based Strategies

# Classic Approaches VS AI-based Strategies

Problem/Business knowledge

AI-based method

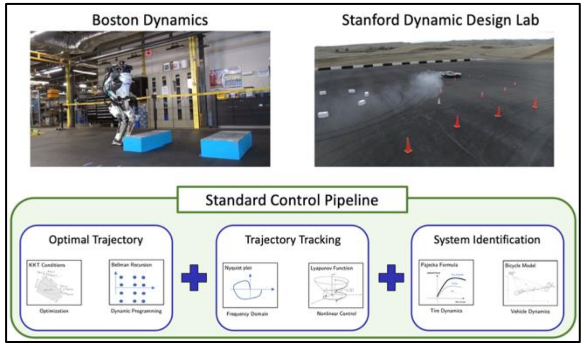



Data/Compute power

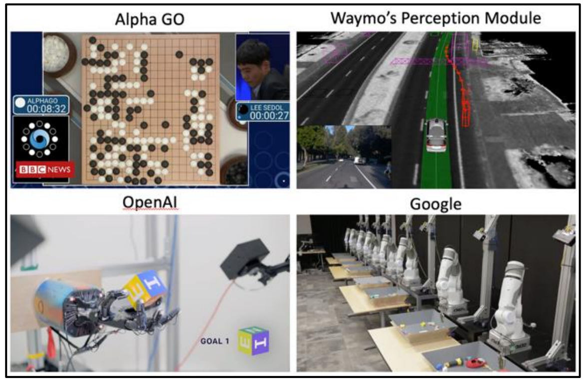
# Classic Approaches VS AI-based Strategies

Problem/Business knowledge

Control methods

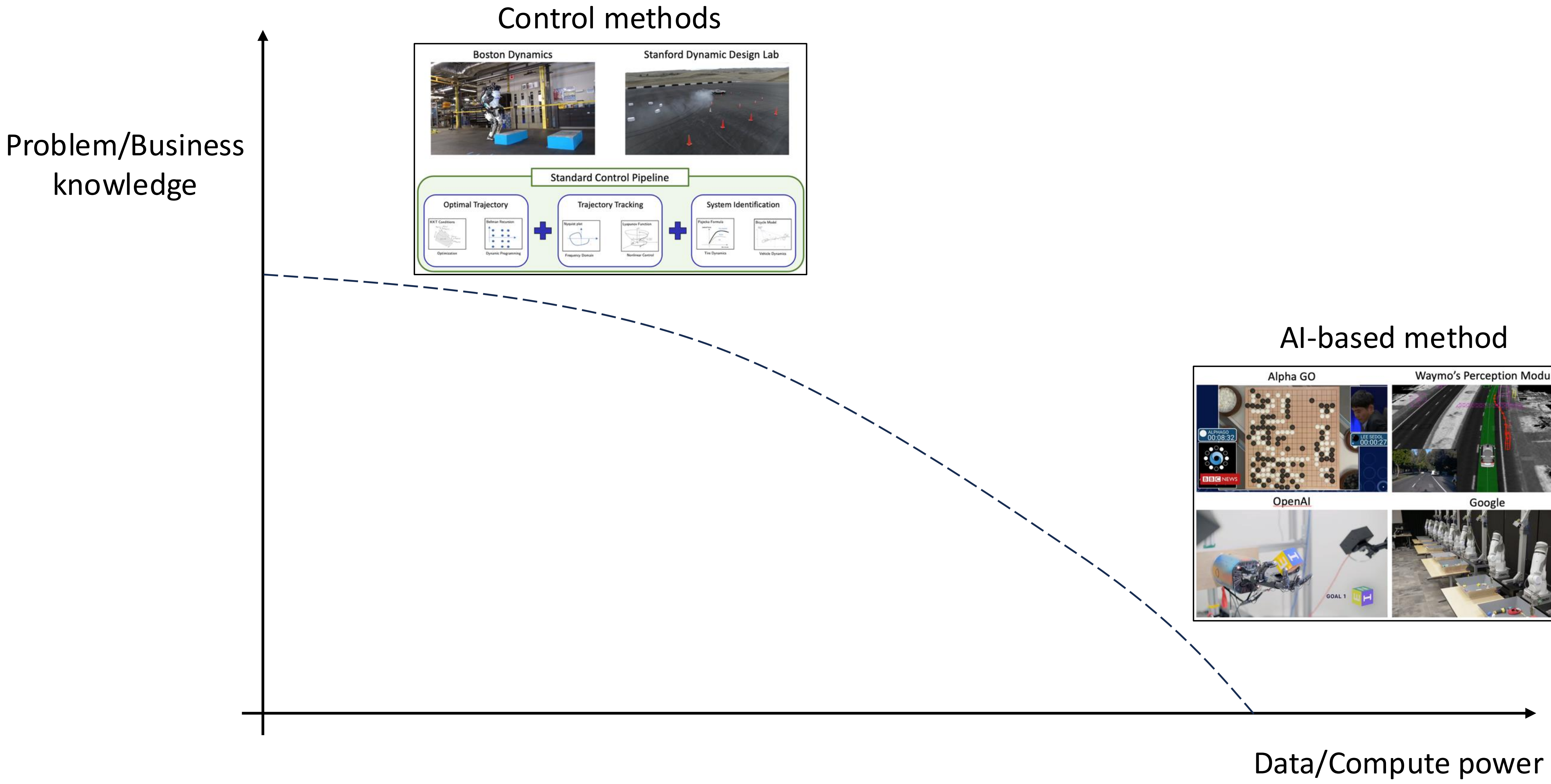



AI-based method

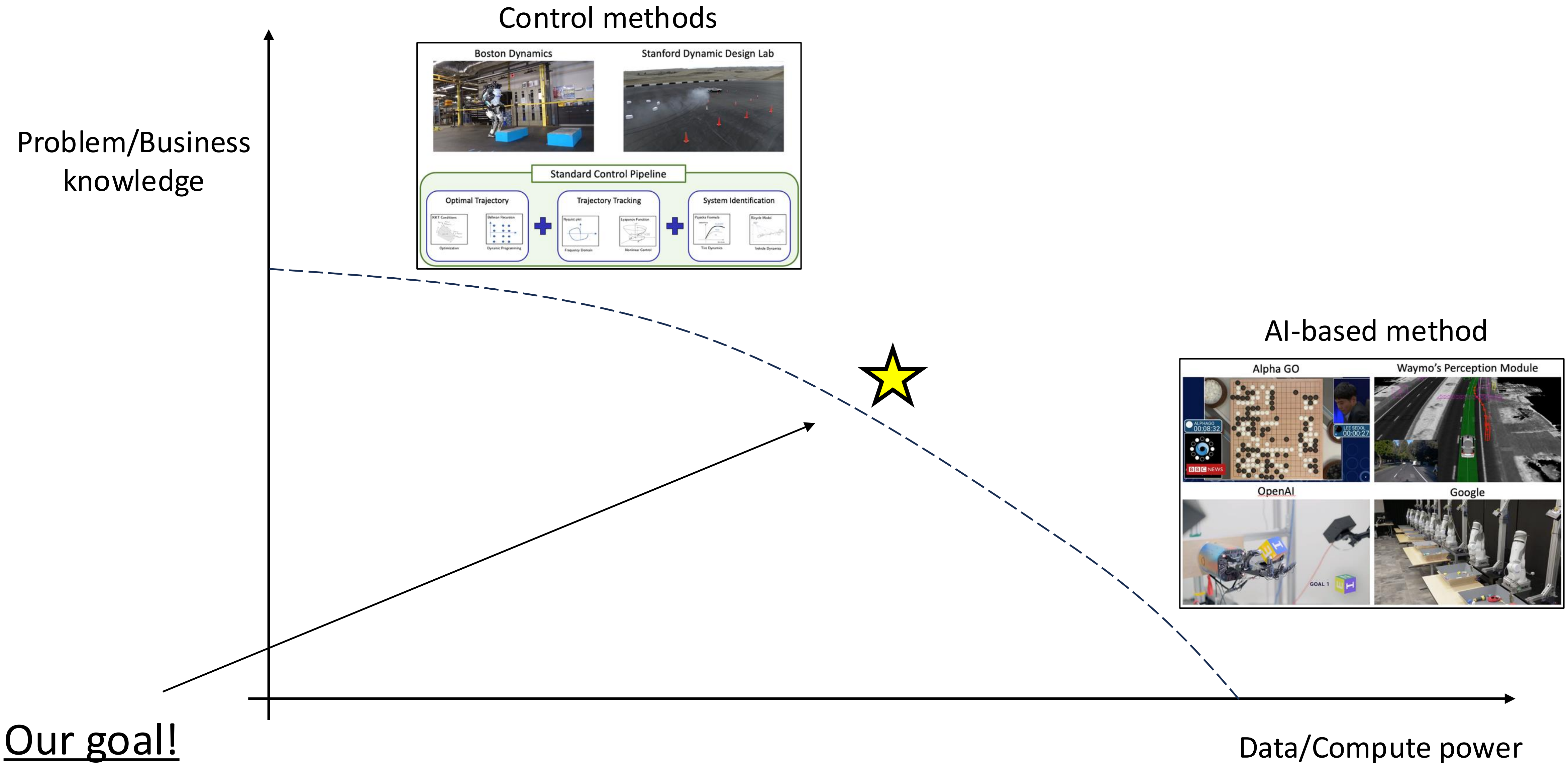



Data/Compute power

# Classic Approaches VS AI-based Strategies

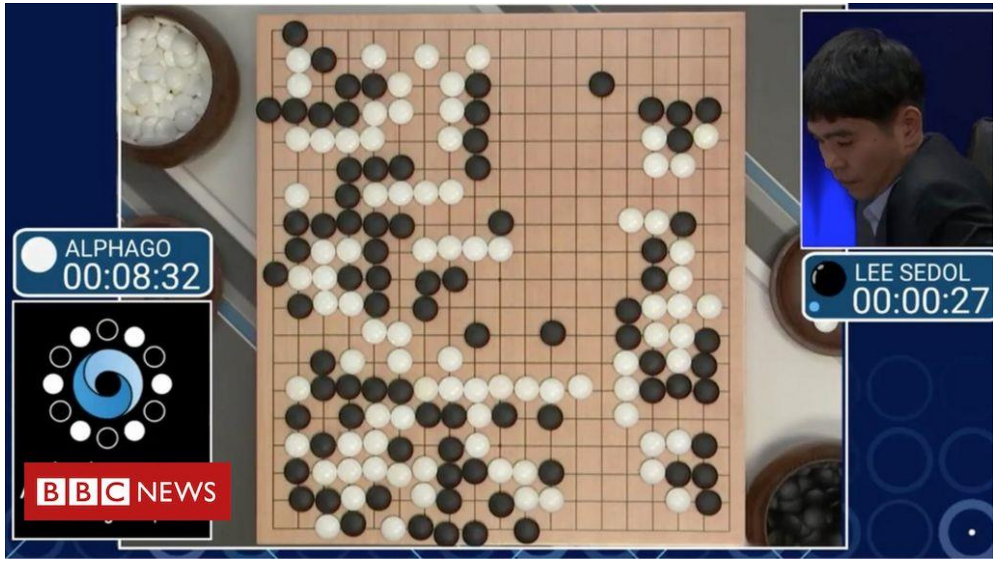


# Classic Approaches VS AI-based Strategies

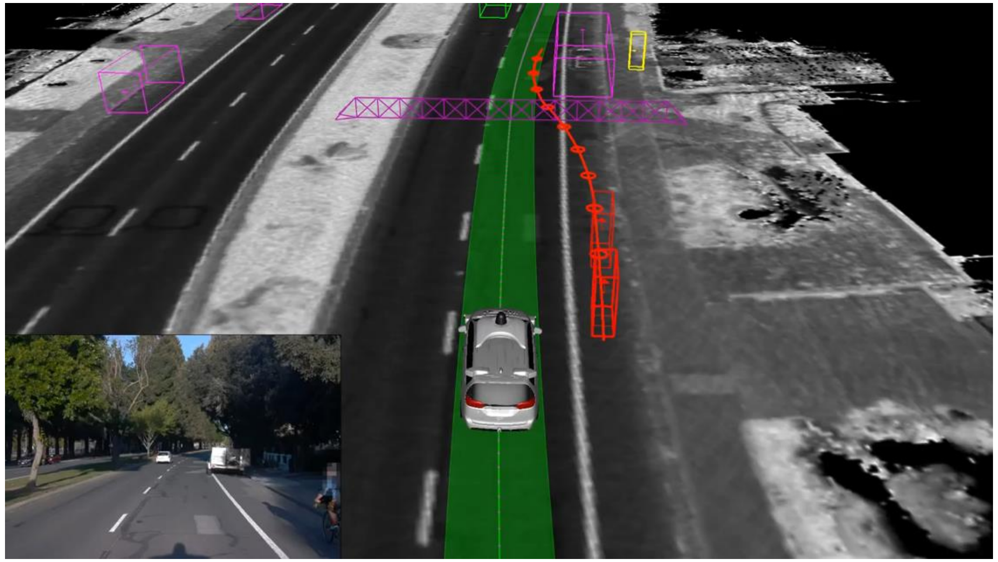


# Success Stories from AI

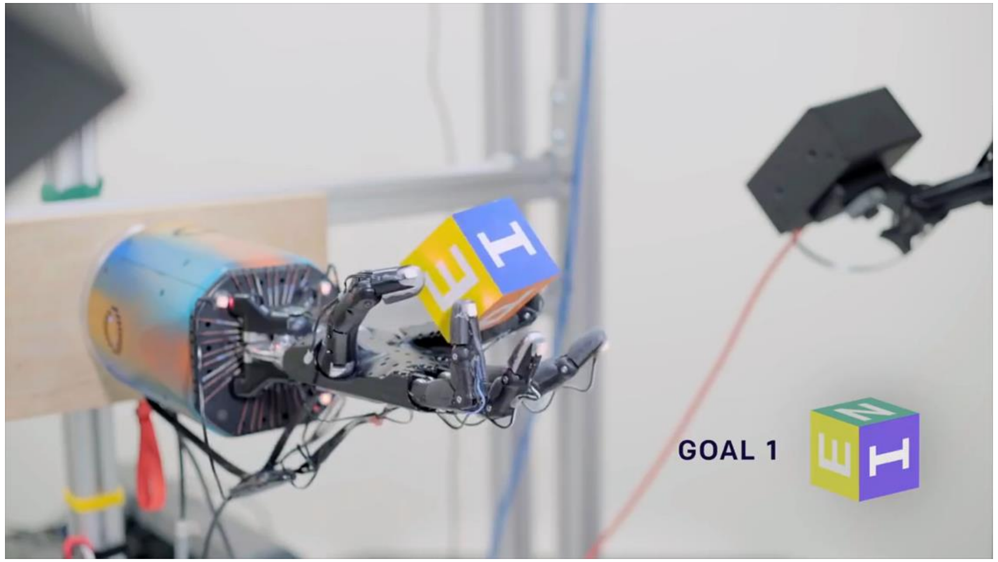
Alpha GO




Waymo's Perception Module



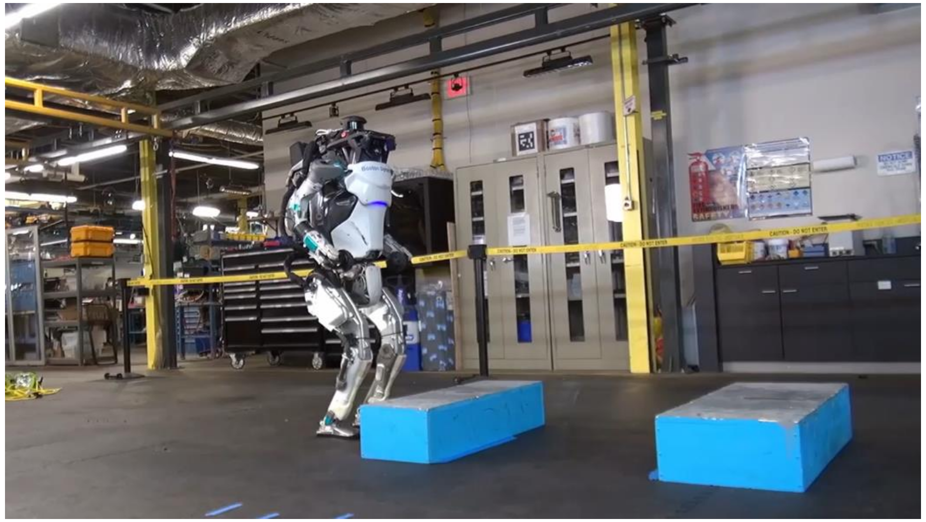
OpenAI




Google

# Success Stories from Control Theory

## Boston Dynamics

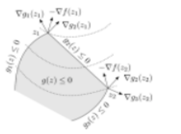


## Stanford Dynamic Design Lab

## Standard Control Pipeline

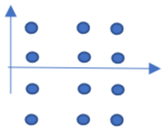
### Optimal Trajectory

KKT Conditions



Optimization

Bellman Recursion

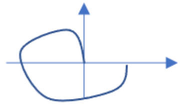


Dynamic Programming

+

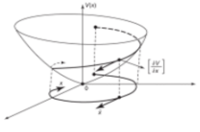
### Trajectory Tracking

Nyquist plot



Frequency Domain

Lyapunov Function



Nonlinear Control

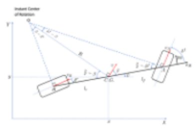
+

### System Identification

Pajecka Formula

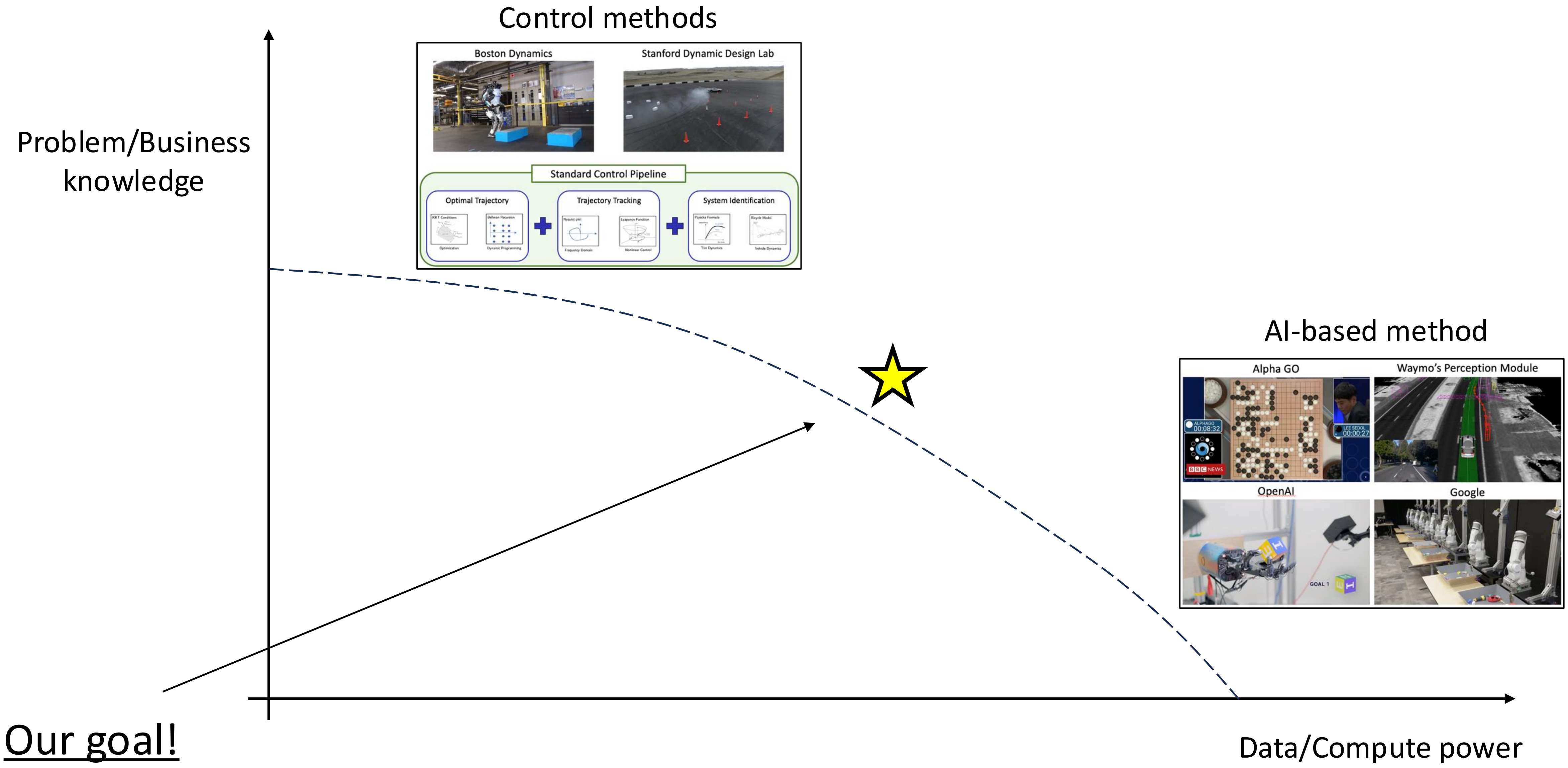
Tire Dynamics

Bicycle Model

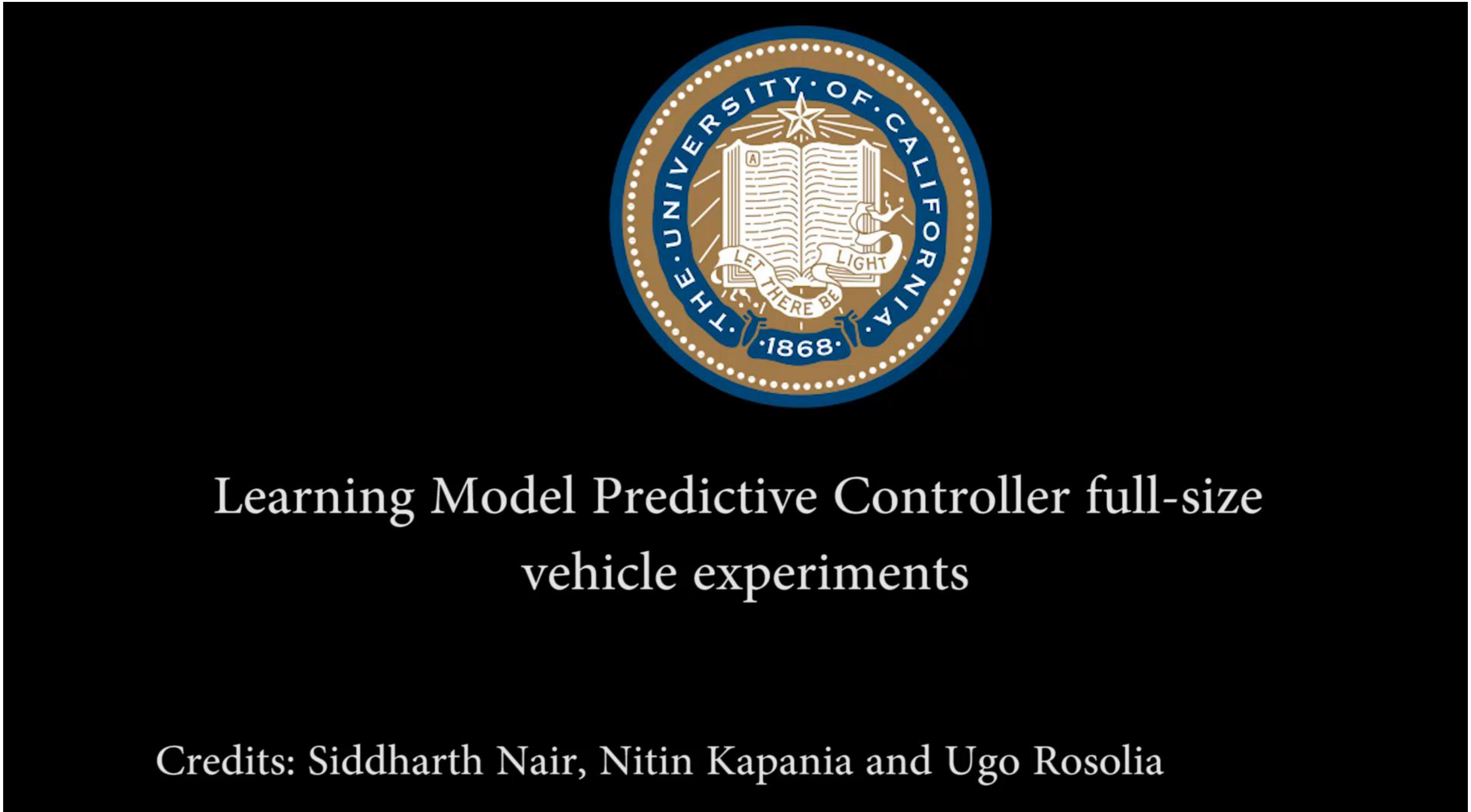


Vehicle Dynamics

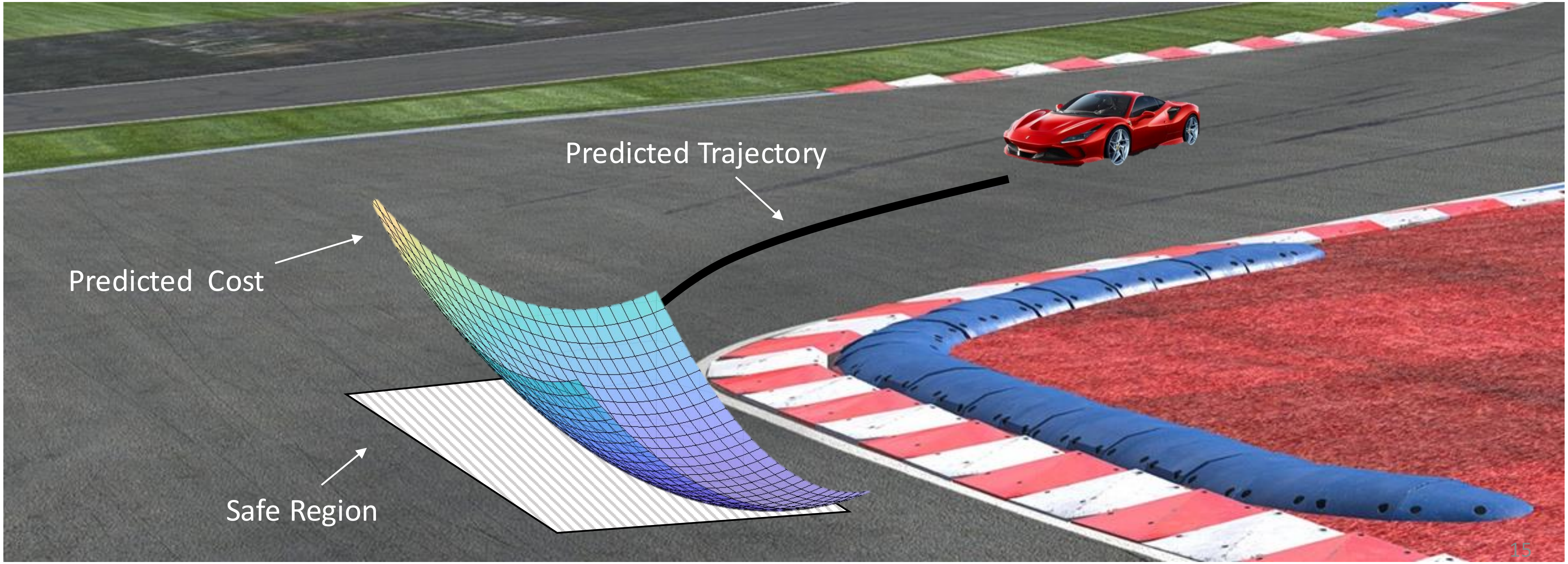
# Classic Approaches VS AI-based Strategies



# Today’s Example



# Lessons from Classical Approaches




- Predicted trajectory given by **Prediction Model**
- Safe region estimated by the **Safe Set**
- Predicted cost estimated by **Value Function**

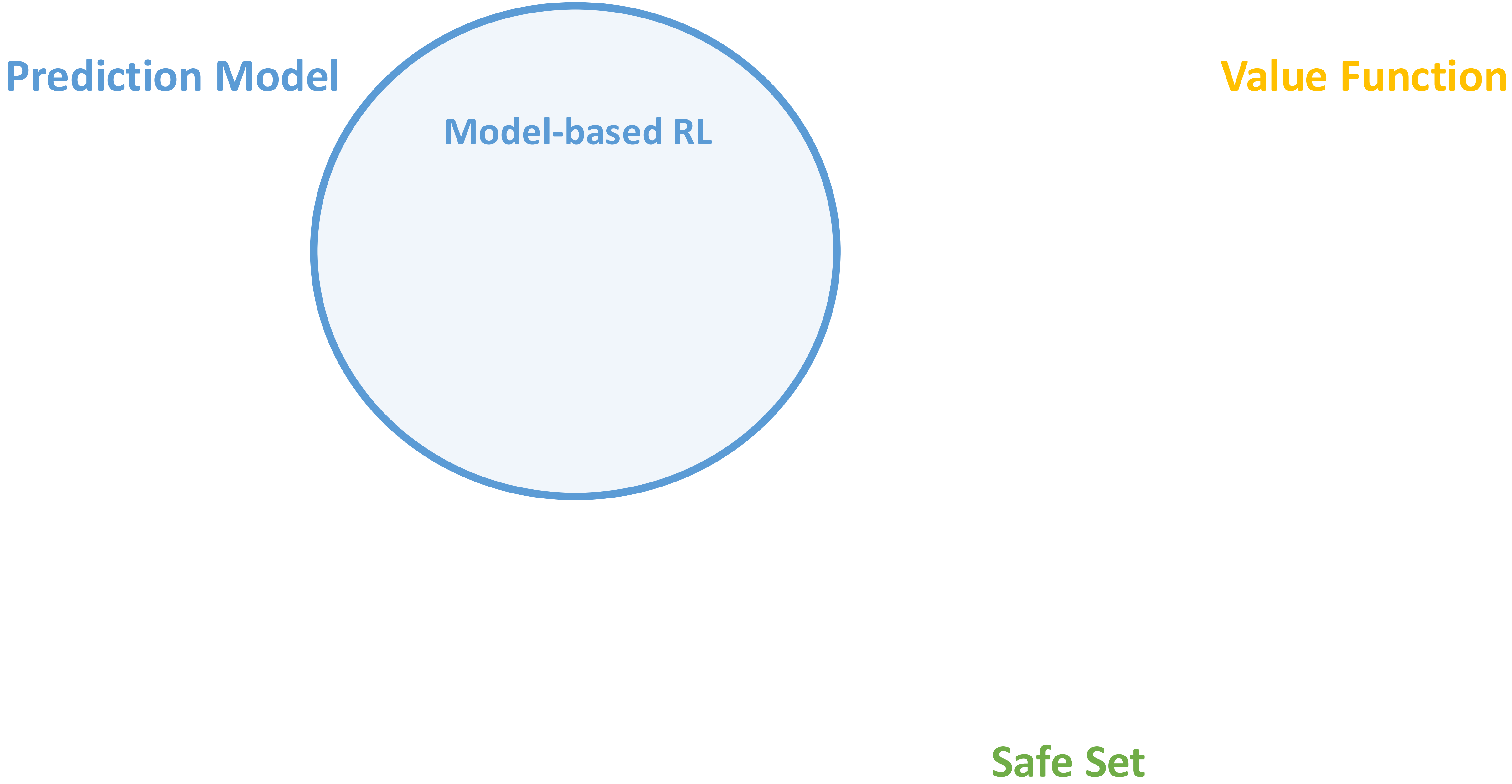
# Three key components to learn

**Prediction Model**

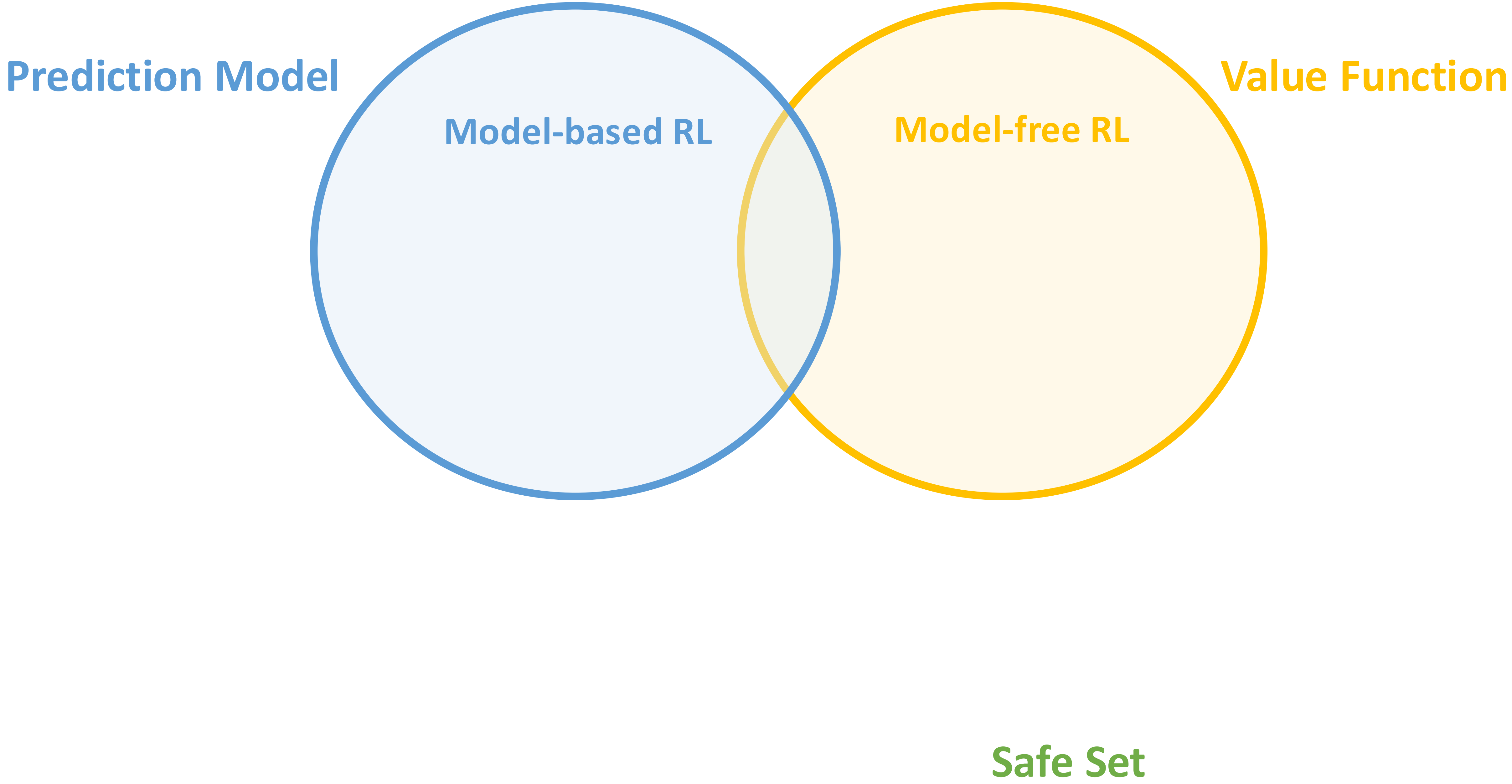
**Value Function**

**Safe Set**

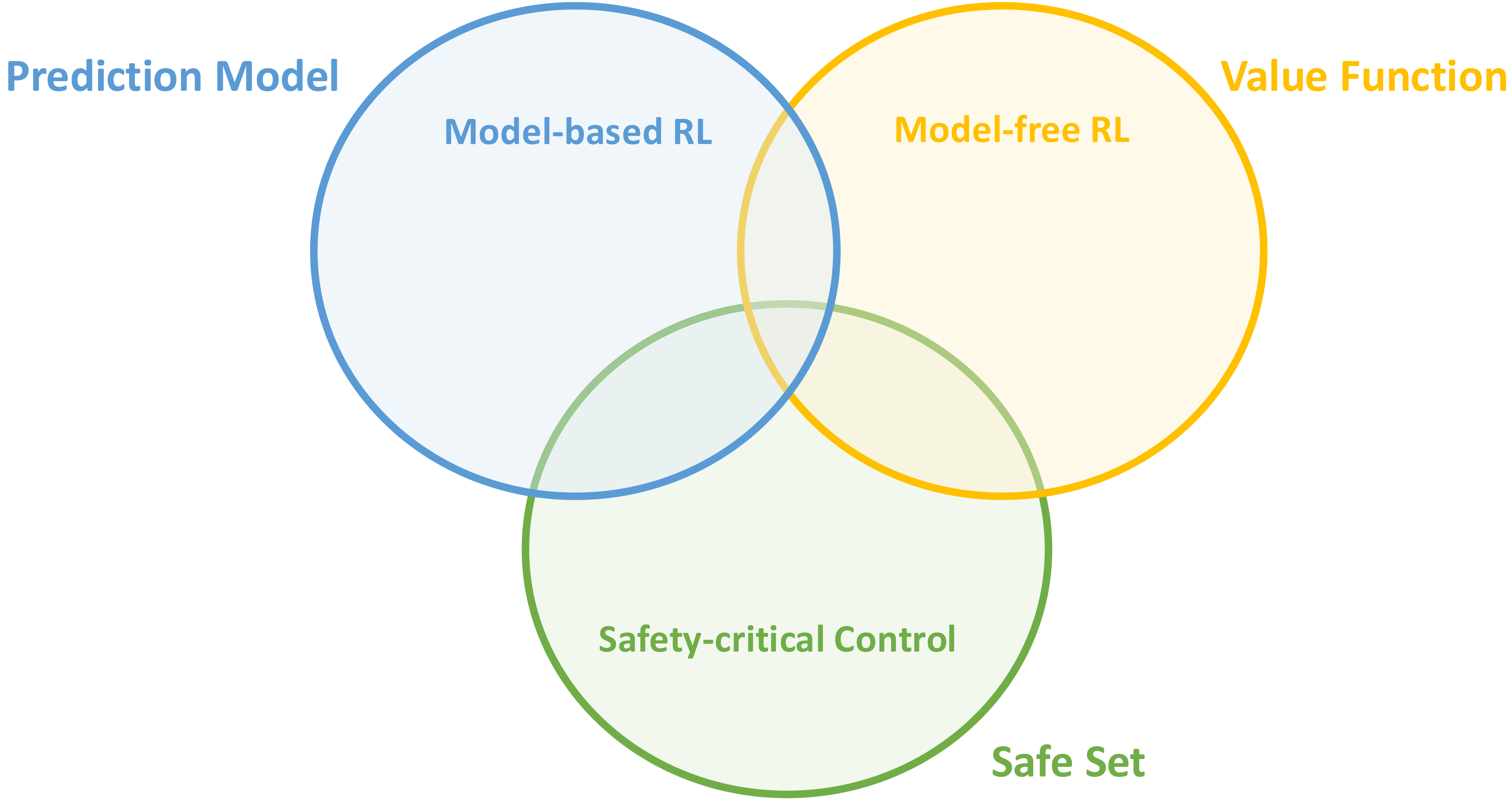
# Three key components to learn



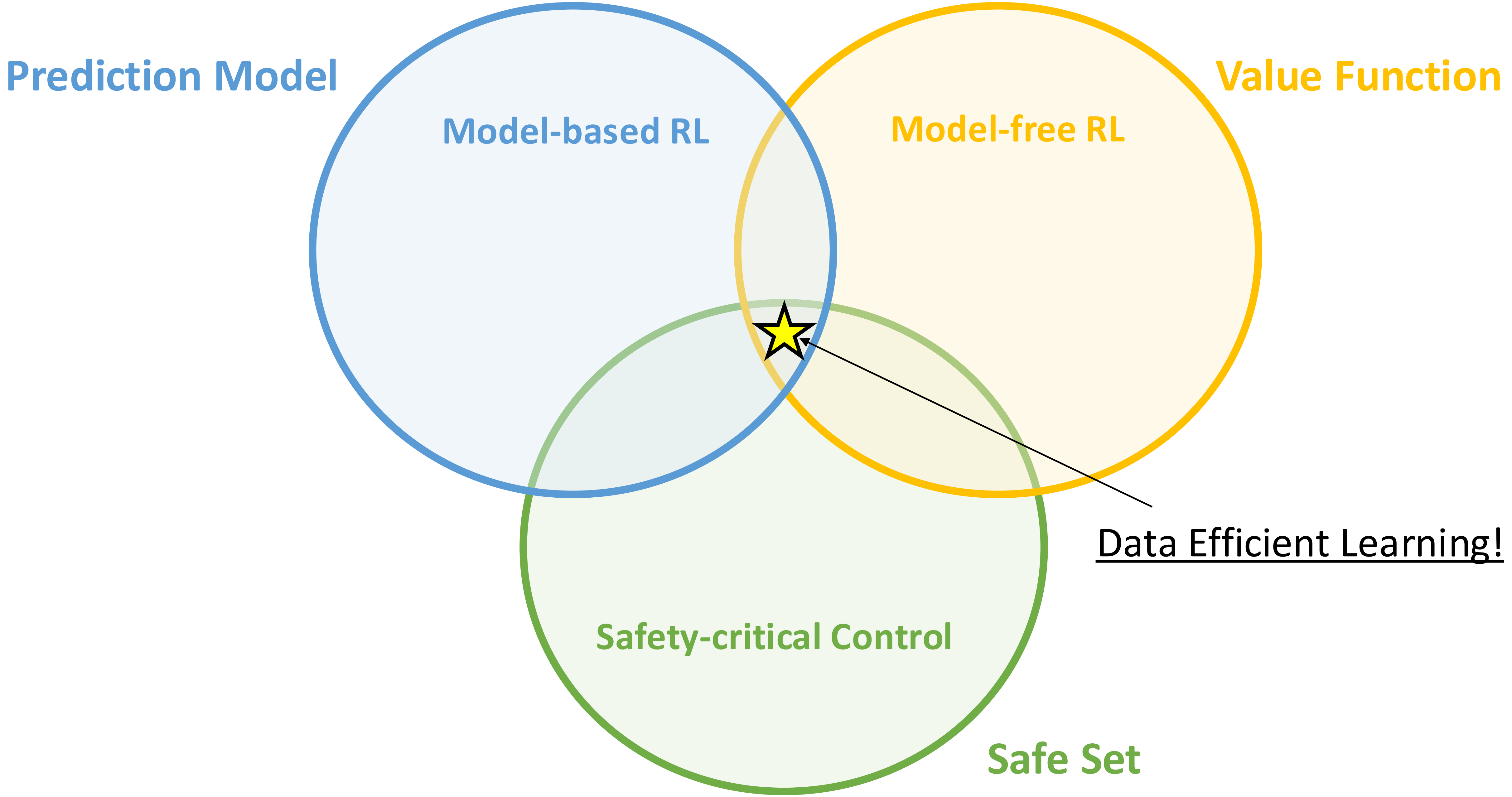
# Three key components to learn



# Three key components to learn



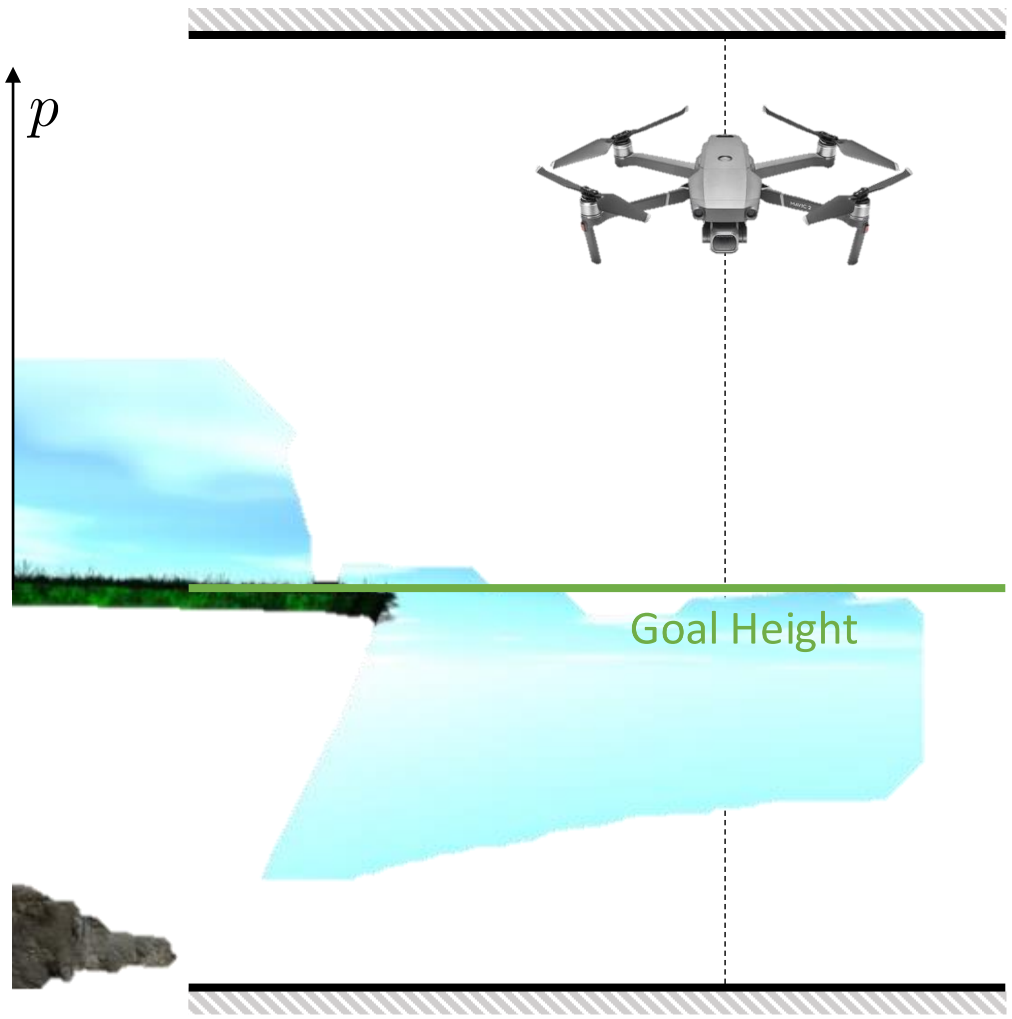
# Three key components to learn



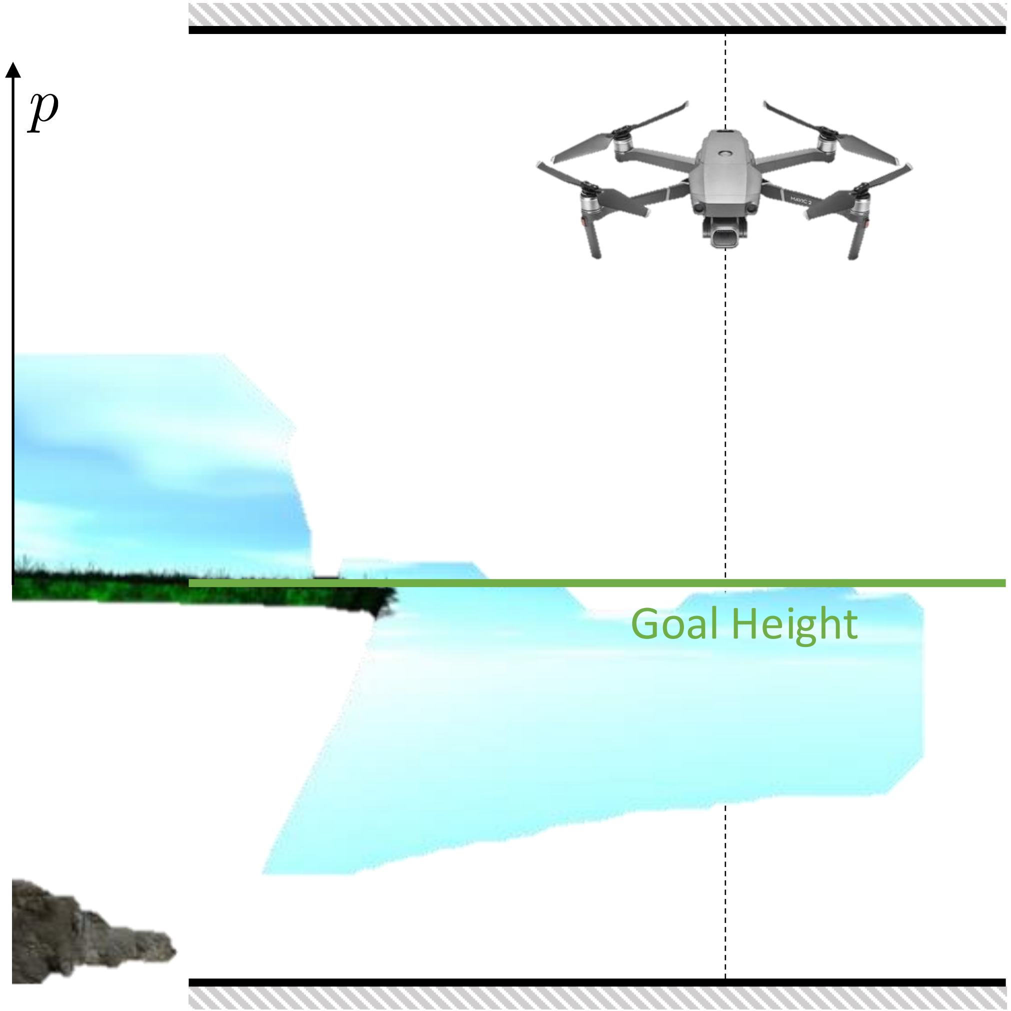
# Iterative Tasks

Iterative data collection and policy update

# Iterative Tasks – Drone Example



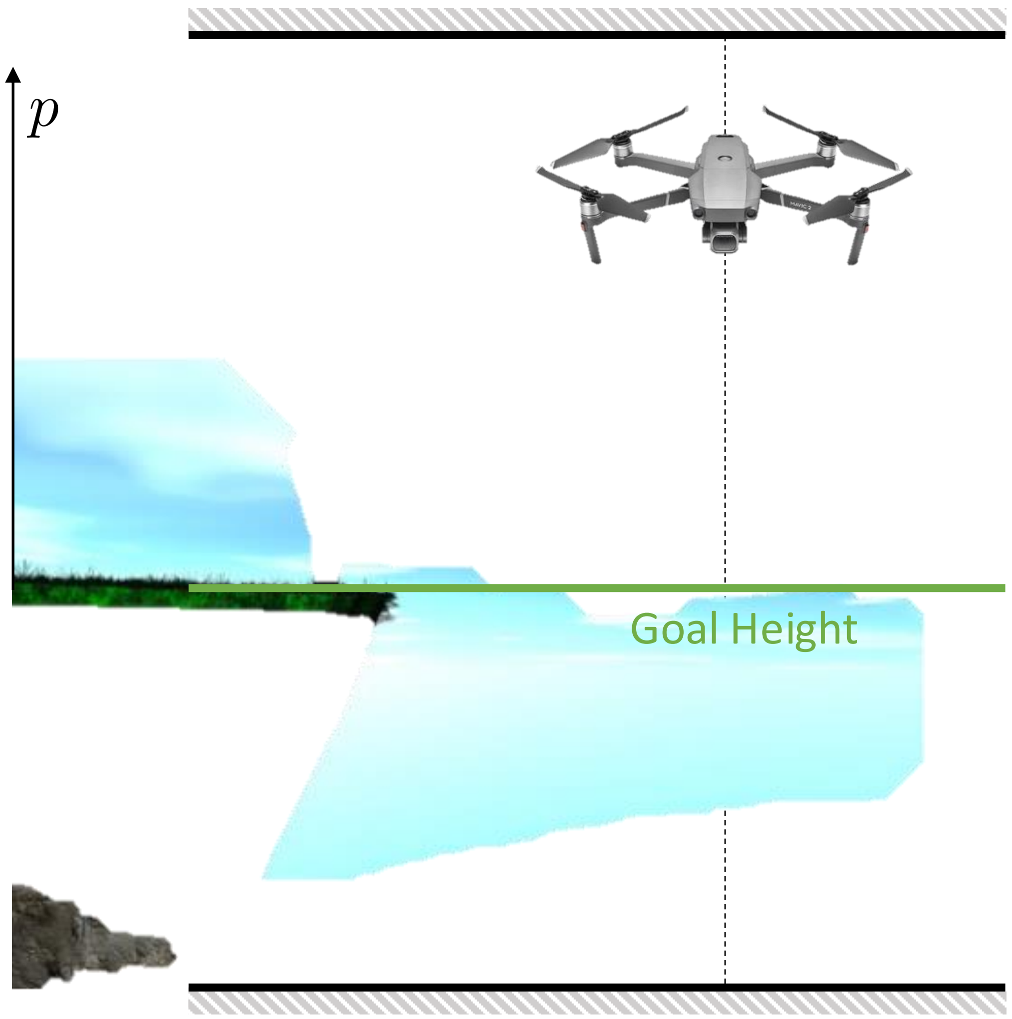
# Iterative Tasks – Drone Example




- State

$$x = \begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

# Iterative Tasks – Drone Example

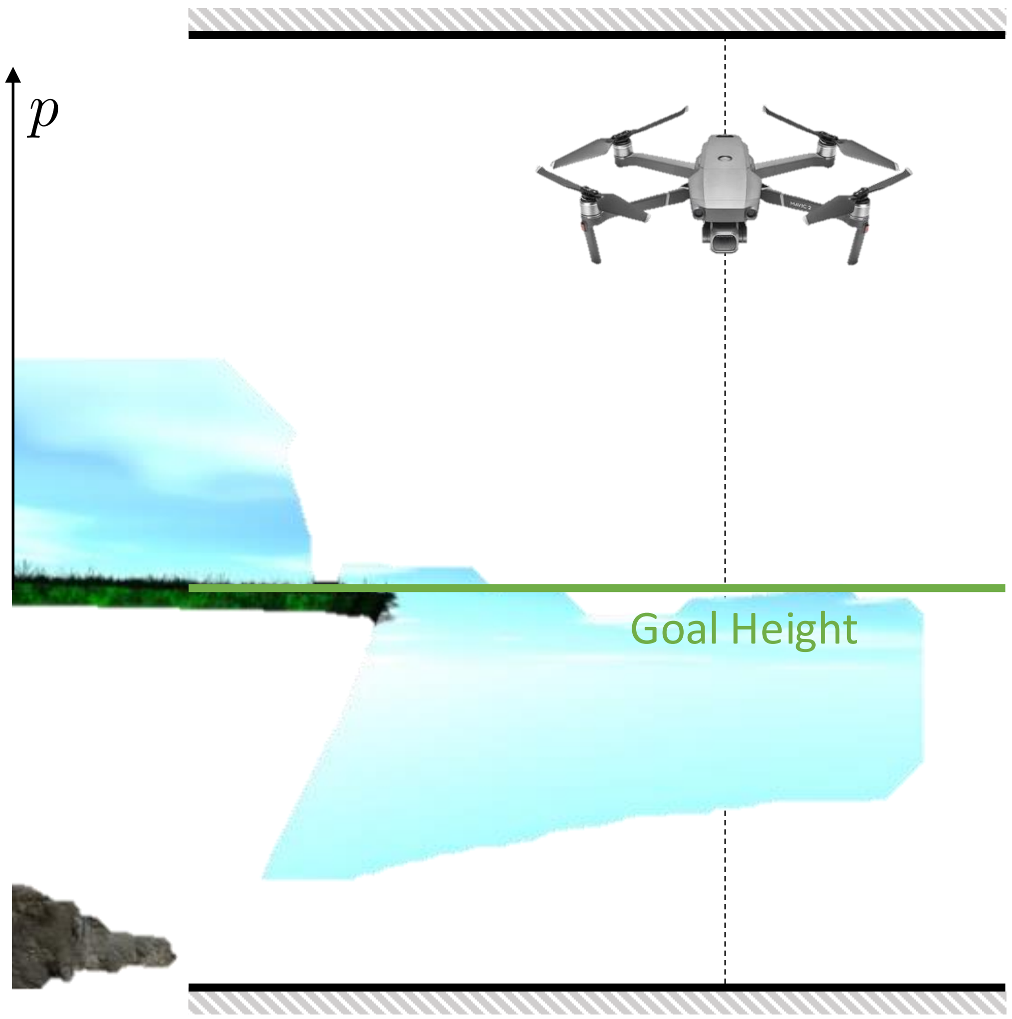



- State

$$x = \begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

- Input $u = a =$ acceleration

# Iterative Tasks – Drone Example



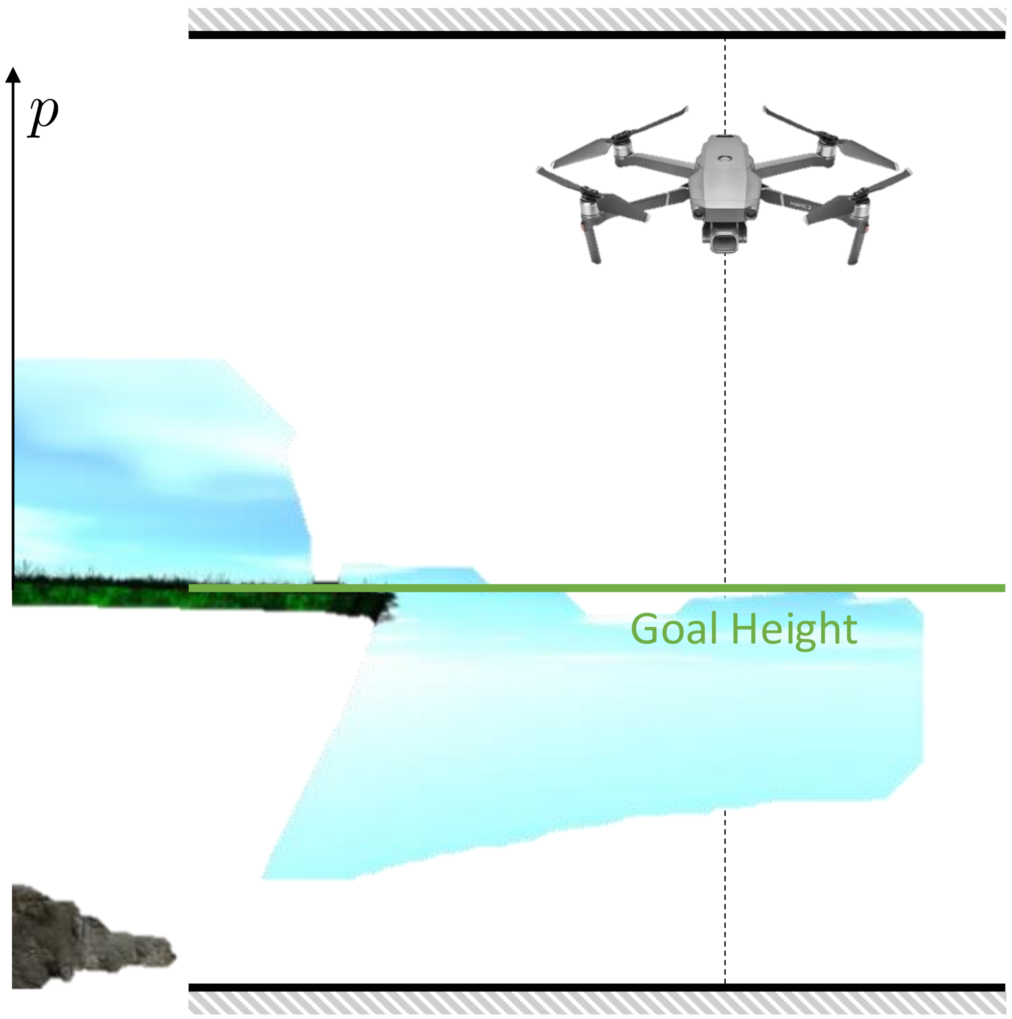

- State

$$x = \begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

- Input $u = a =$ acceleration
- Dynamics

$$\begin{bmatrix} p_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ a_k \end{bmatrix}$$

# Iterative Tasks – Drone Example




- State

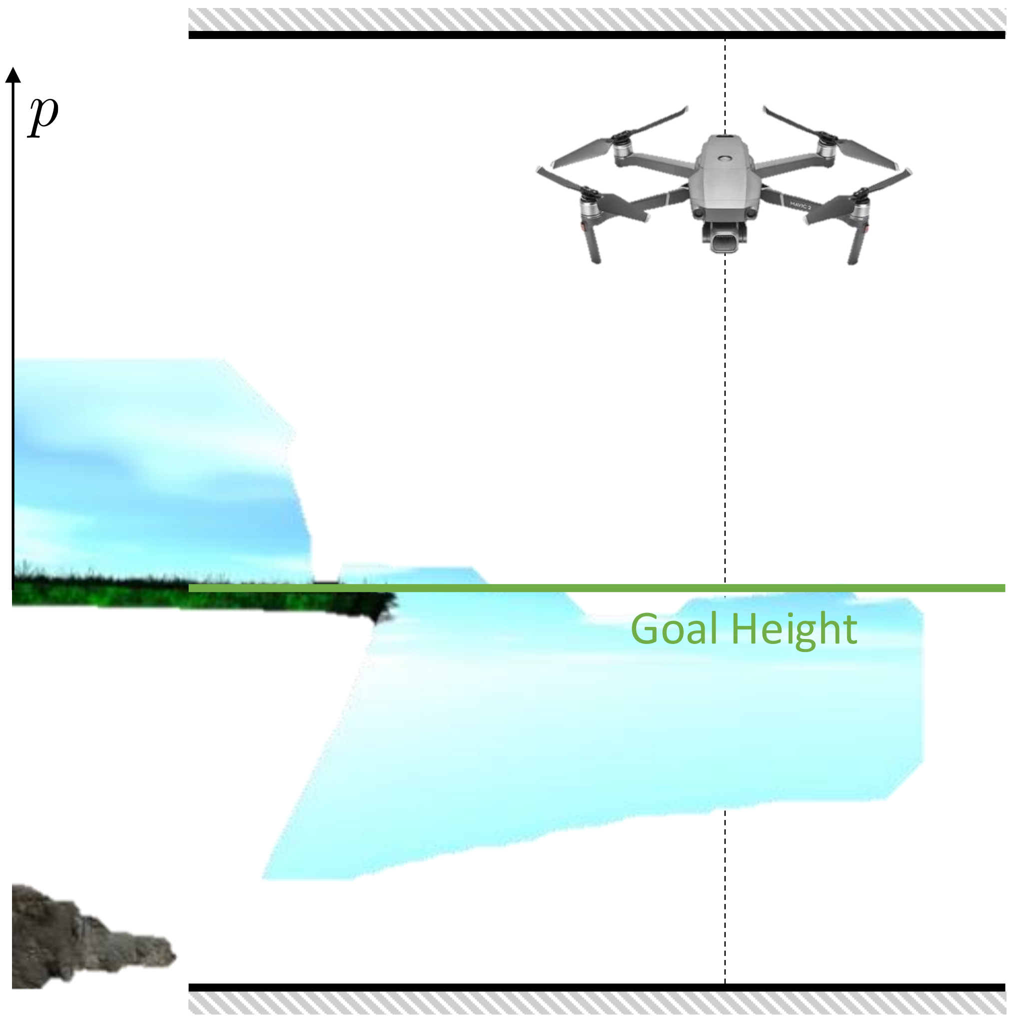
$$x = \begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

- Input $u = a =$ acceleration
- Dynamics

$$\begin{bmatrix} p_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ a_k \end{bmatrix}$$

- Cost $x_k^\top Q x_k + u_k^\top R u_k$

# Iterative Tasks – Drone Example




- State

$$x = \begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

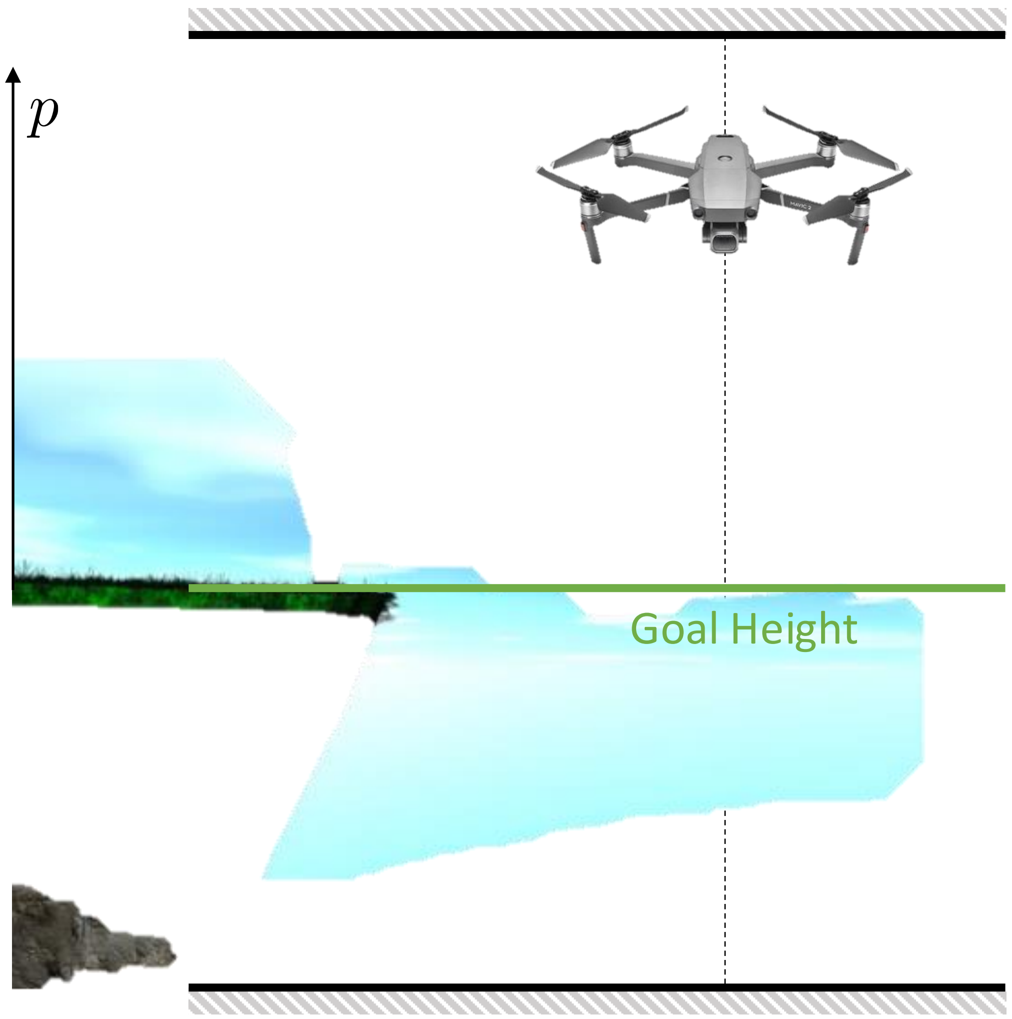
- Input $u = a =$ acceleration
- Dynamics

$$\begin{bmatrix} p_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ a_k \end{bmatrix}$$

- Cost $x_k^\top Q x_k + u_k^\top R u_k$
- Constraints

$$\begin{bmatrix} -5 \\ -5 \\ -0.5 \end{bmatrix} \leq \begin{bmatrix} p_k \\ v_k \\ a_k \end{bmatrix} \leq \begin{bmatrix} 5 \\ 5 \\ 0.5 \end{bmatrix}$$

# Iterative Tasks – Drone Example




- State

$$x = \begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

- Input $u = a =$ acceleration
- Dynamics

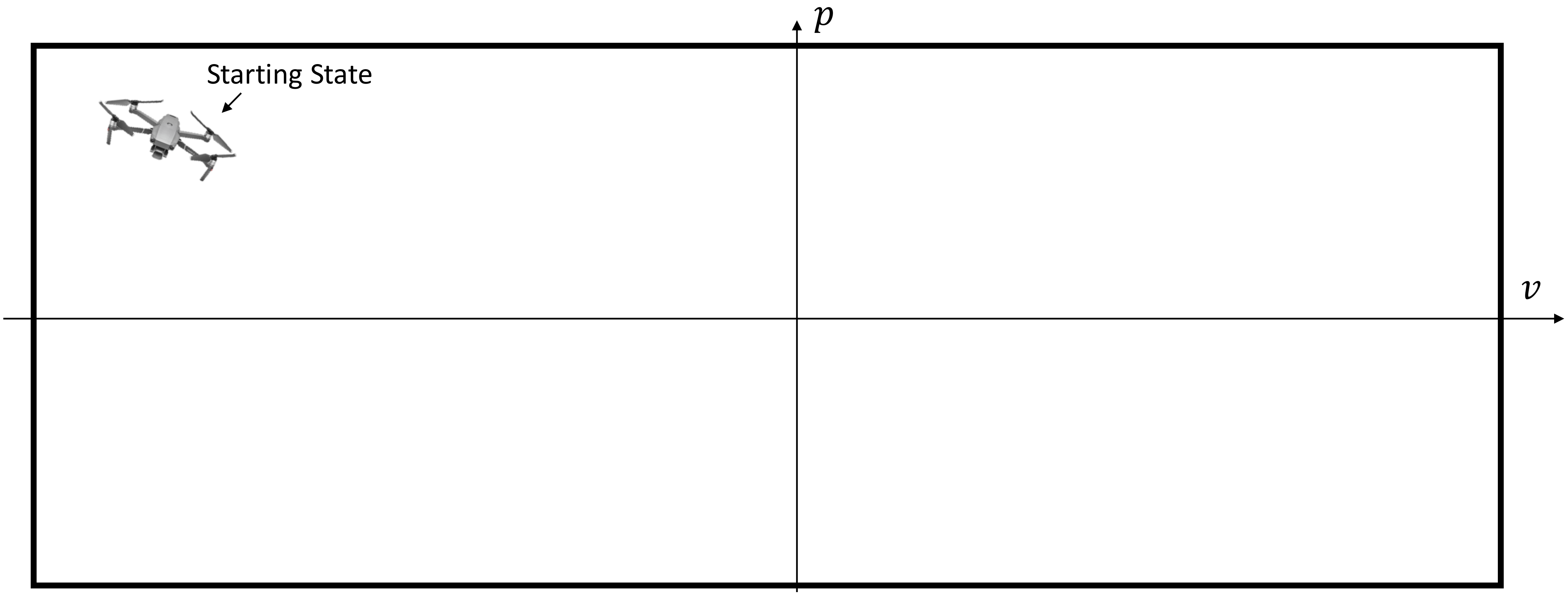
$$\begin{bmatrix} p_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ a_k \end{bmatrix}$$

- Cost $x_k^\top Q x_k + u_k^\top R u_k$
- Constraints

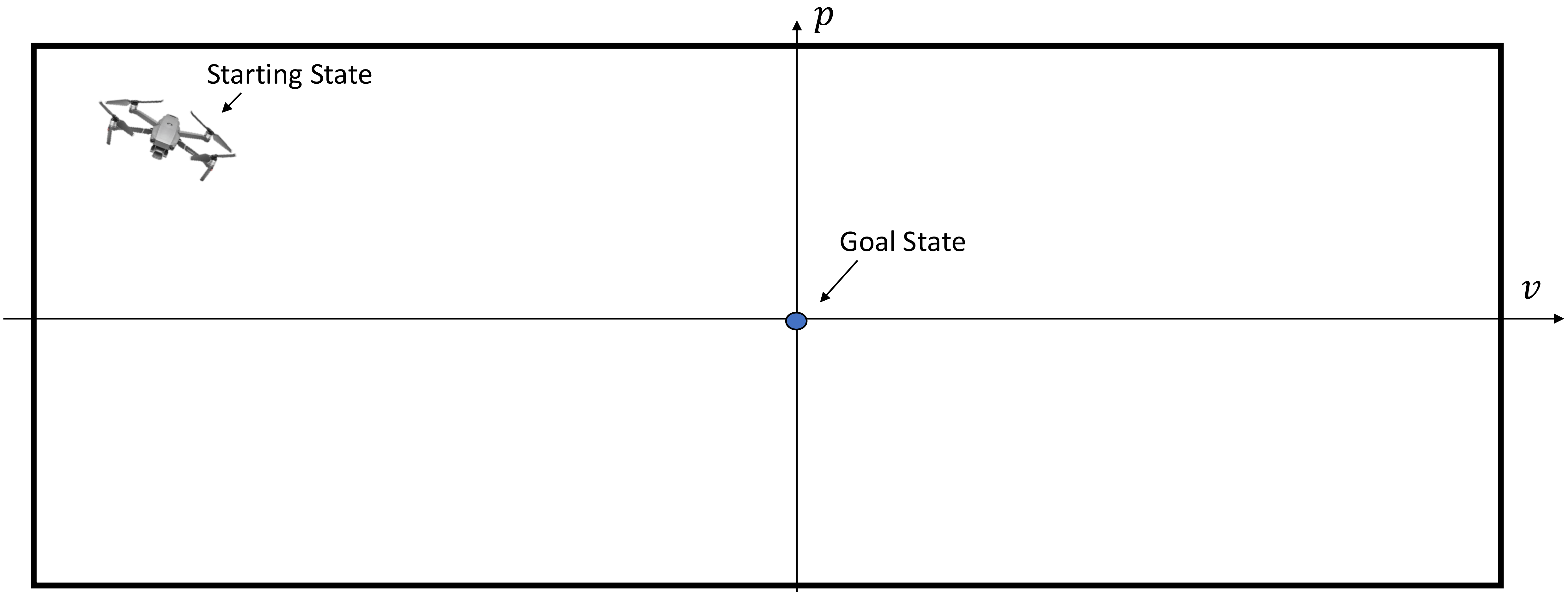
$$\begin{bmatrix} -5 \\ -5 \\ \boxed{-0.5} \end{bmatrix} \leq \begin{bmatrix} p_k \\ v_k \\ \boxed{a_k} \end{bmatrix} \leq \begin{bmatrix} 5 \\ 5 \\ \boxed{0.5} \end{bmatrix}$$

Limited actuation!

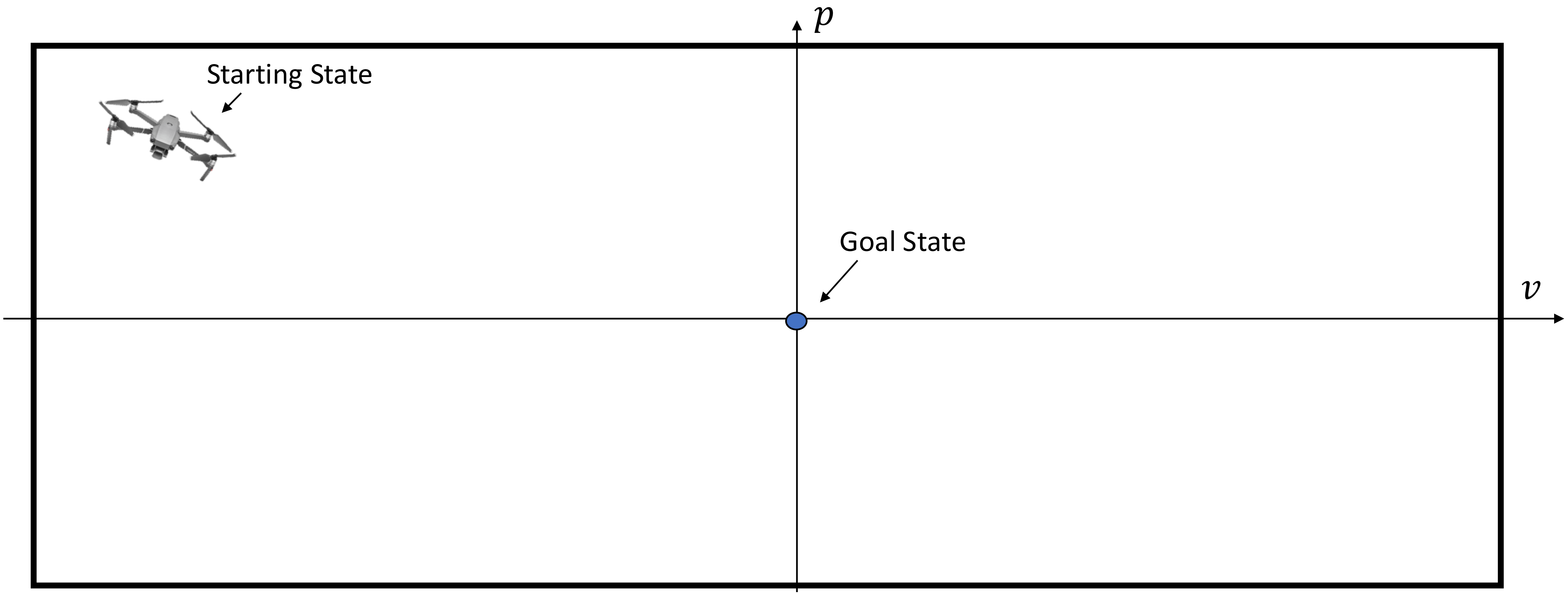
# Iterative Tasks – Drone Example



# Iterative Tasks – Drone Example

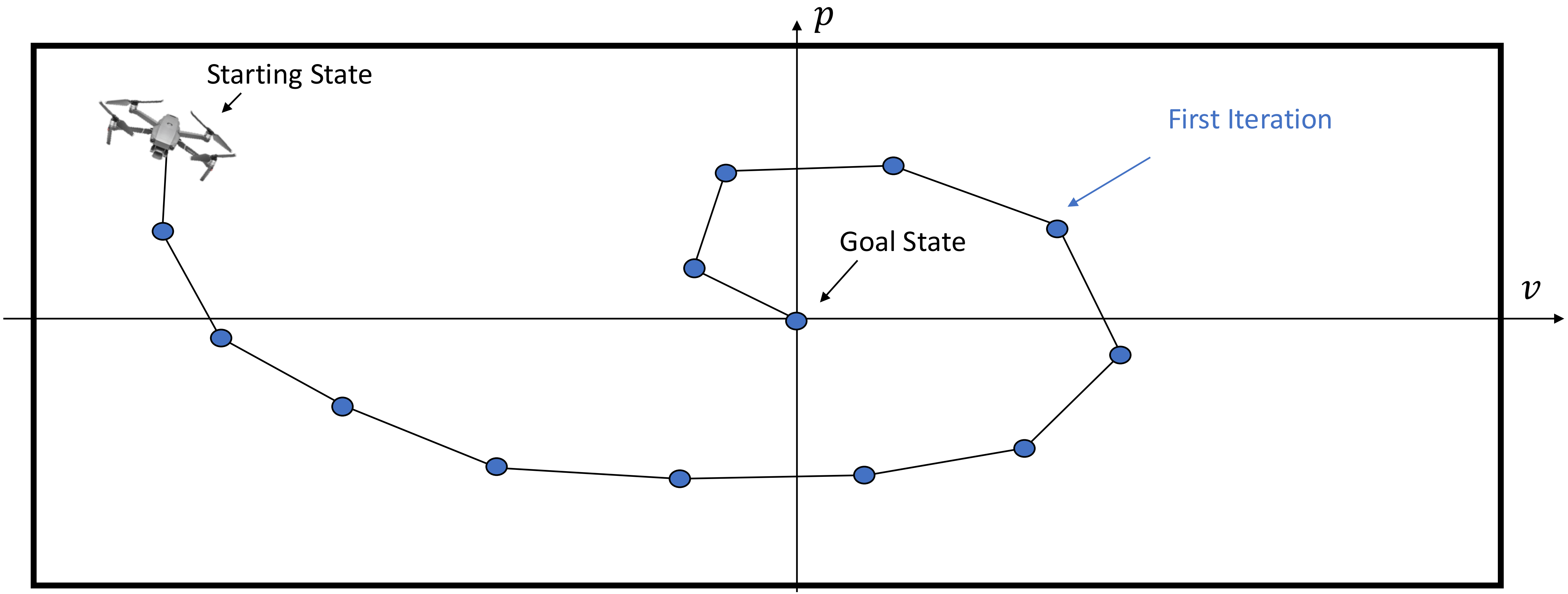


# Iterative Tasks – Drone Example



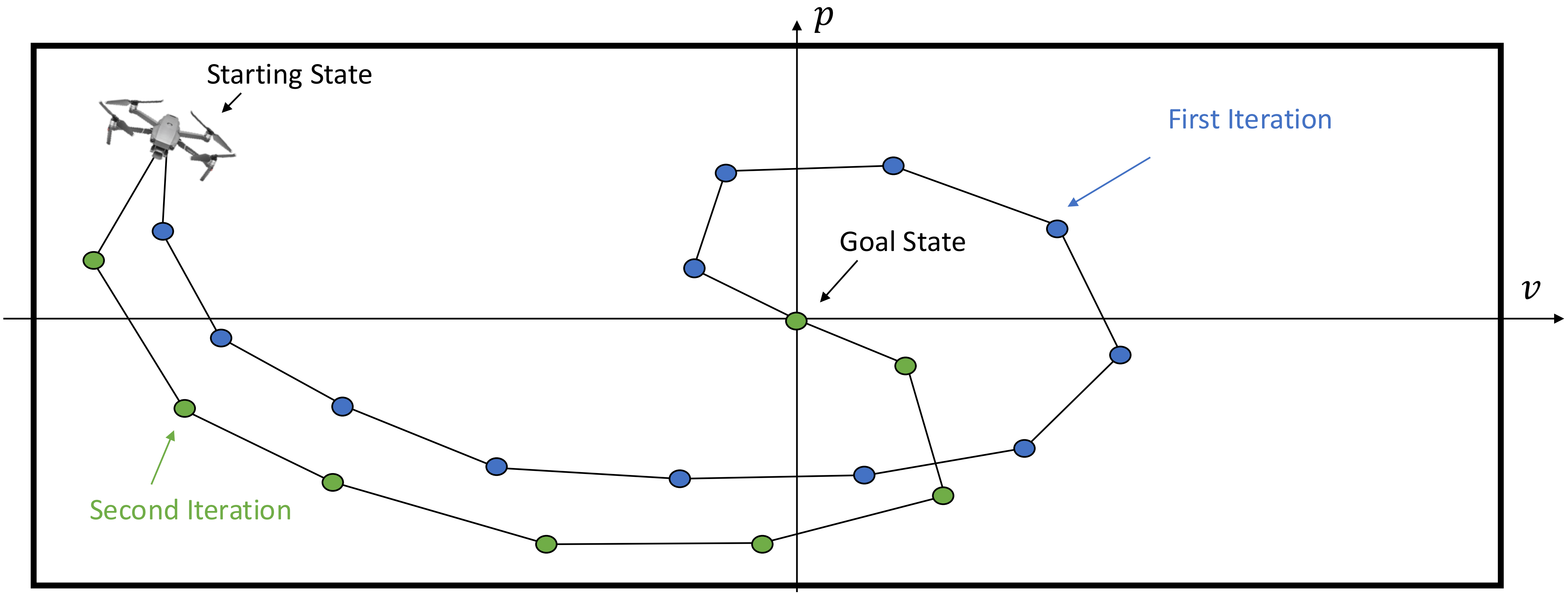

- Iteration = one execution of the task

# Iterative Tasks – Drone Example



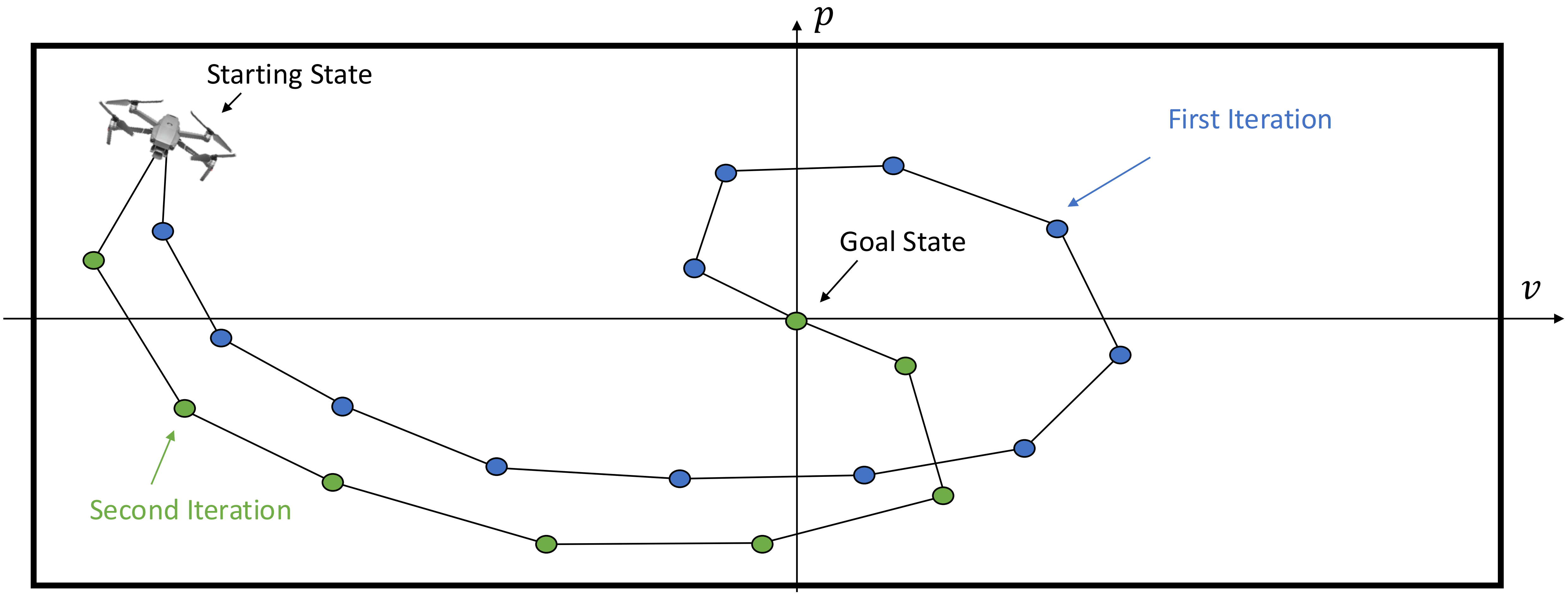

- Iteration = one execution of the task

# Iterative Tasks – Drone Example




- Iteration = one execution of the task

# Iterative Tasks – Drone Example



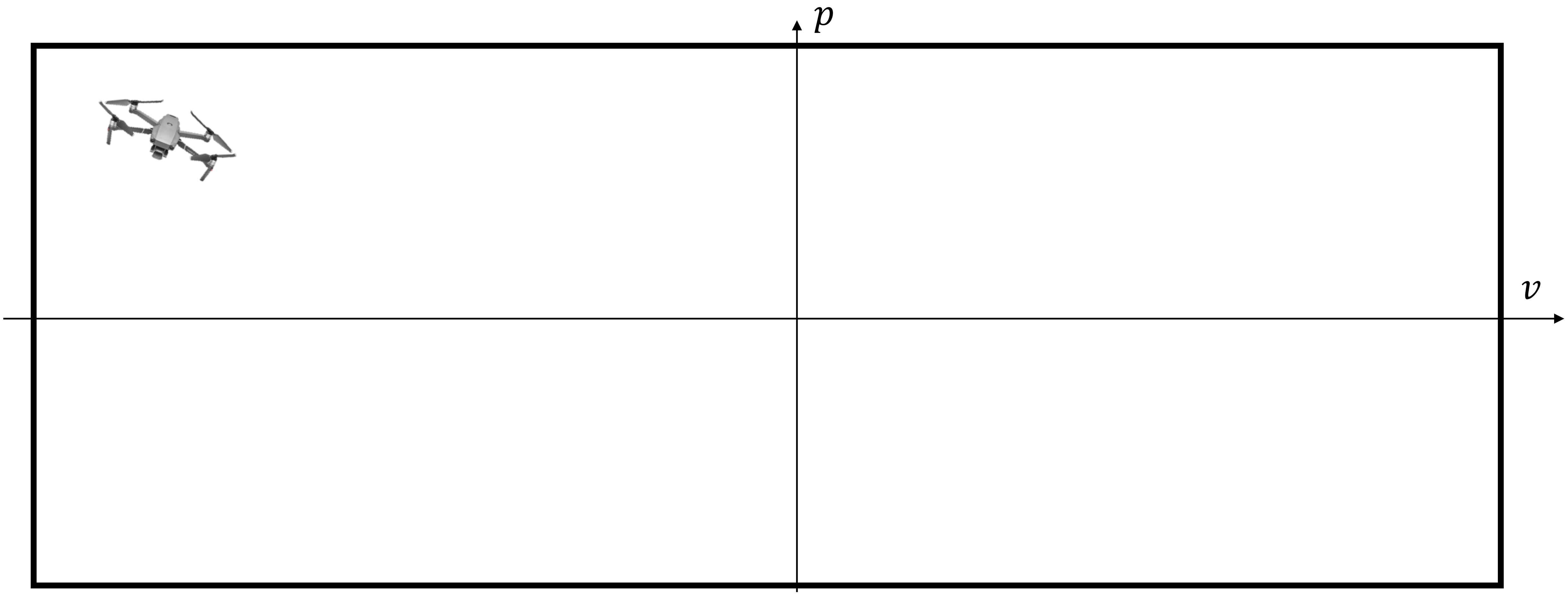

- Iteration = one execution of the task
- Objective: Drive the drone optimally from the starting state to the goal state

# Learning Model Predictive Control (LMPC)

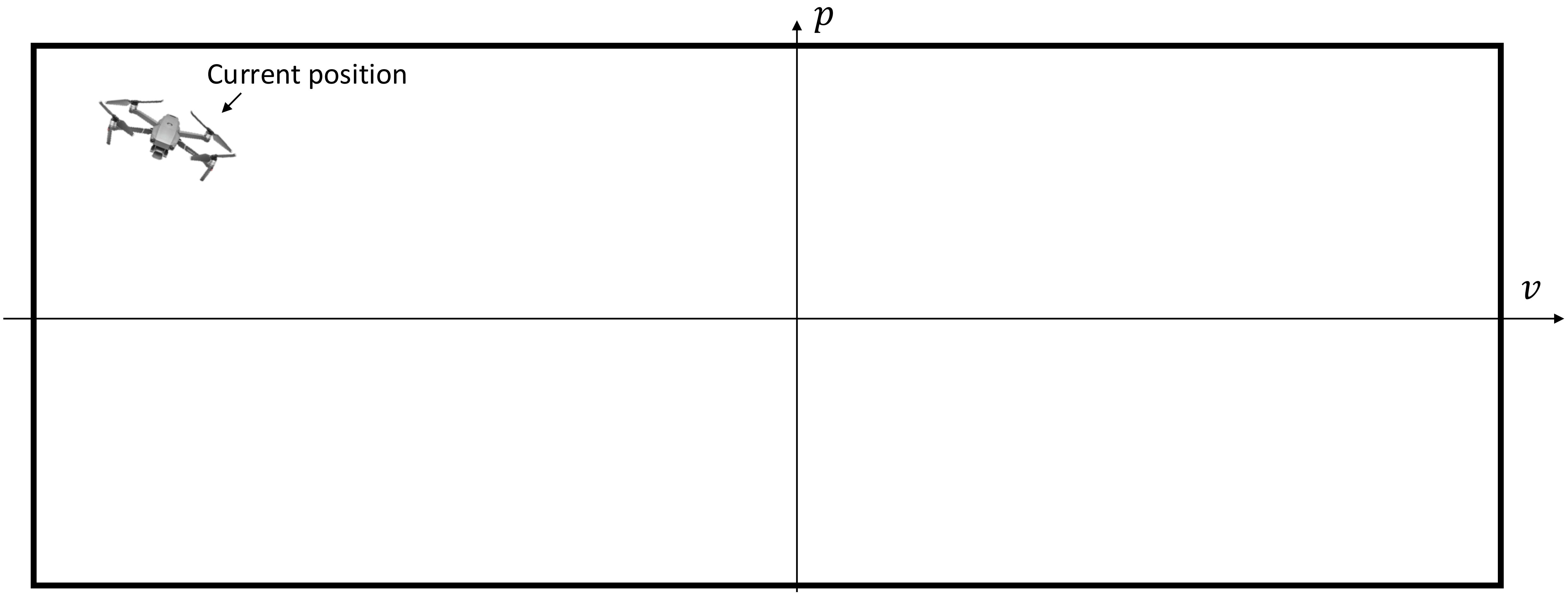
Exploit historical data

# Learning Model Predictive Control (LMPC) – Key Idea

# Learning Model Predictive Control (LMPC) – Key Idea



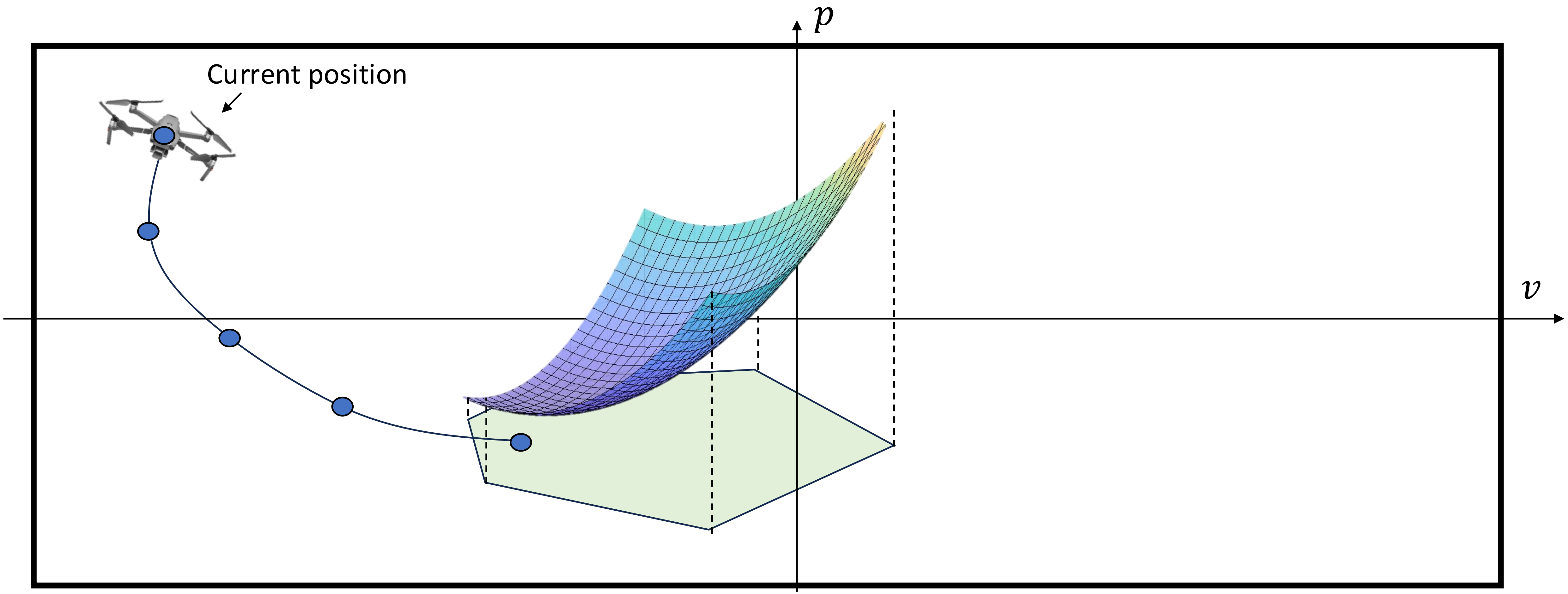
# Learning Model Predictive Control (LMPC) – Key Idea




Algorithm steps:

- Get current state

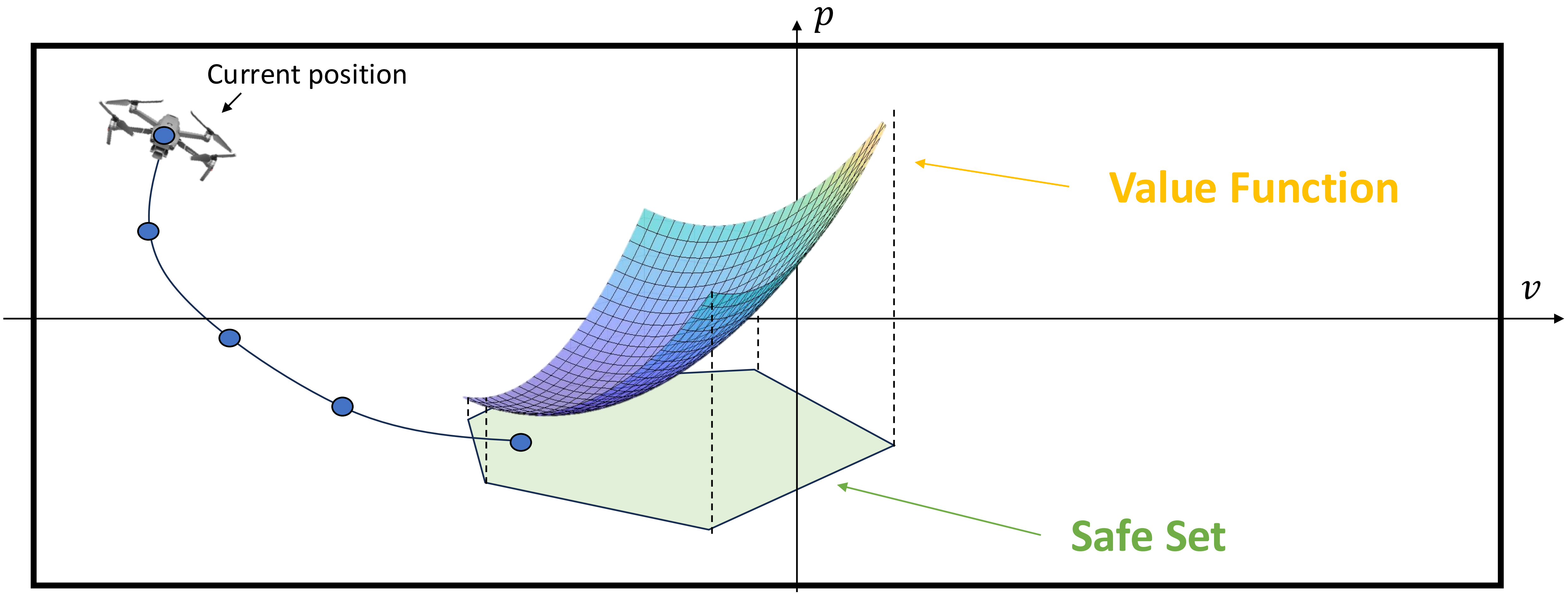
# Learning Model Predictive Control (LMPC) – Key Idea




Algorithm steps:

- Get current state
- Plan a trajectory

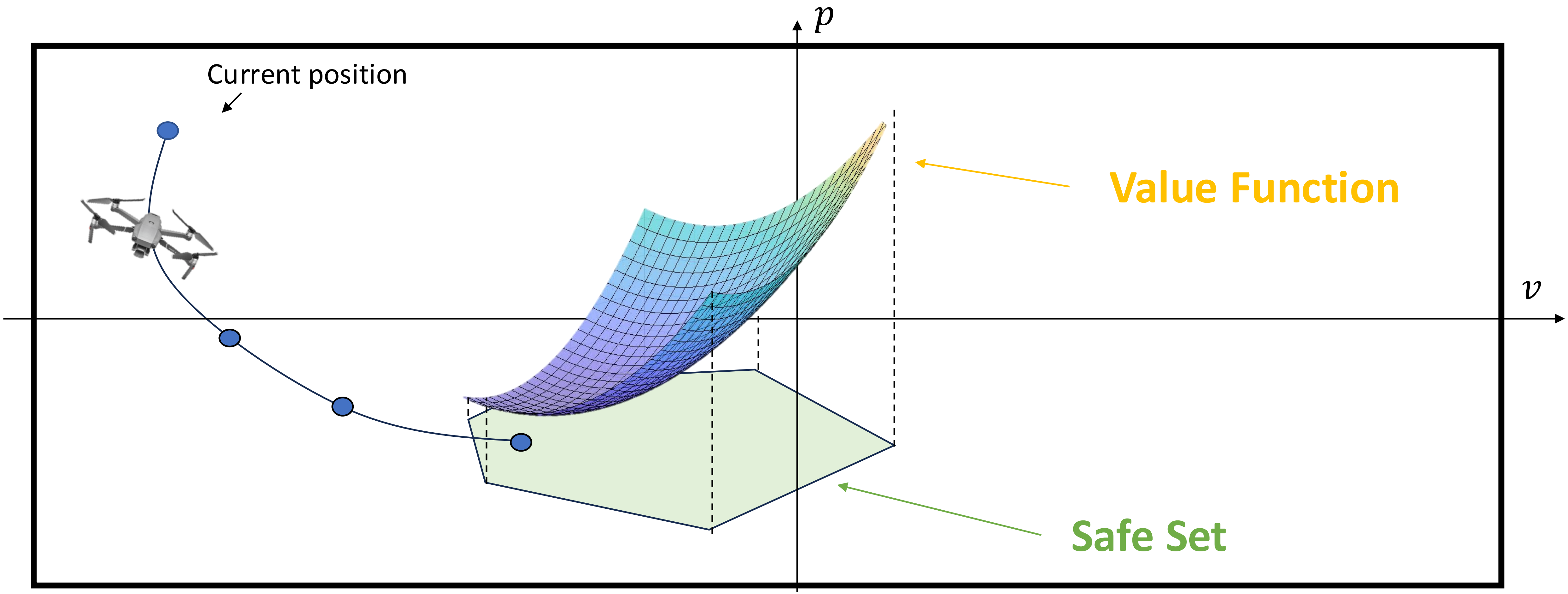
# Learning Model Predictive Control (LMPC) – Key Idea




Algorithm steps:

- Get current state
- Plan a trajectory

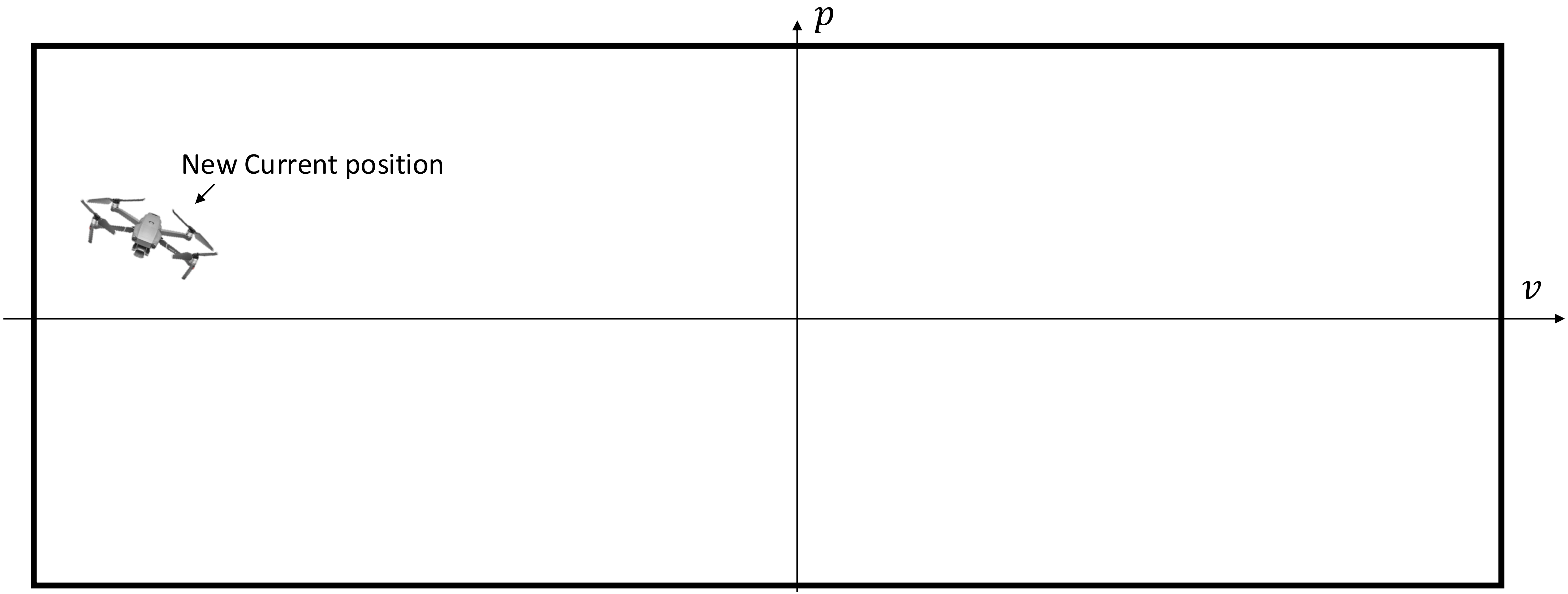
# Learning Model Predictive Control (LMPC) – Key Idea




Algorithm steps:

- Get current state
- Plan a trajectory
- Execute the action

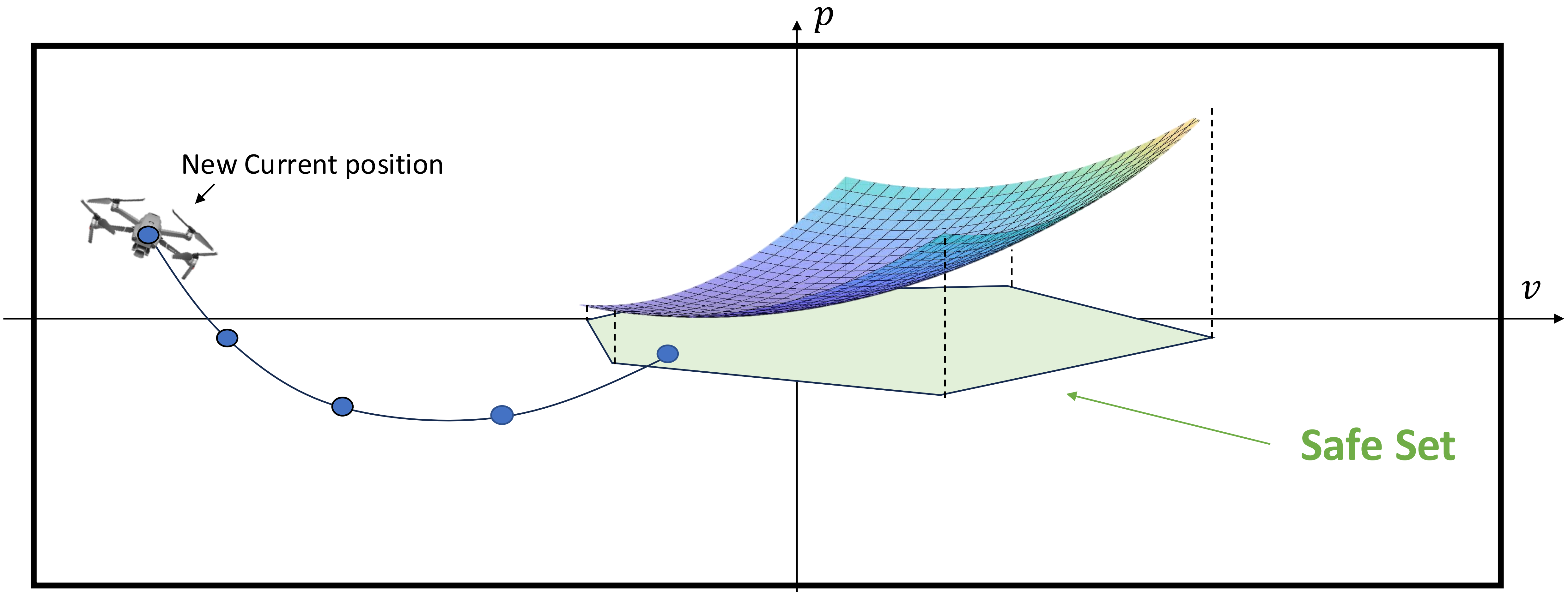
# Learning Model Predictive Control (LMPC) – Key Idea




Algorithm steps:

- Get current state
- Plan a trajectory
- Execute the action

# Learning Model Predictive Control (LMPC) – Key Idea



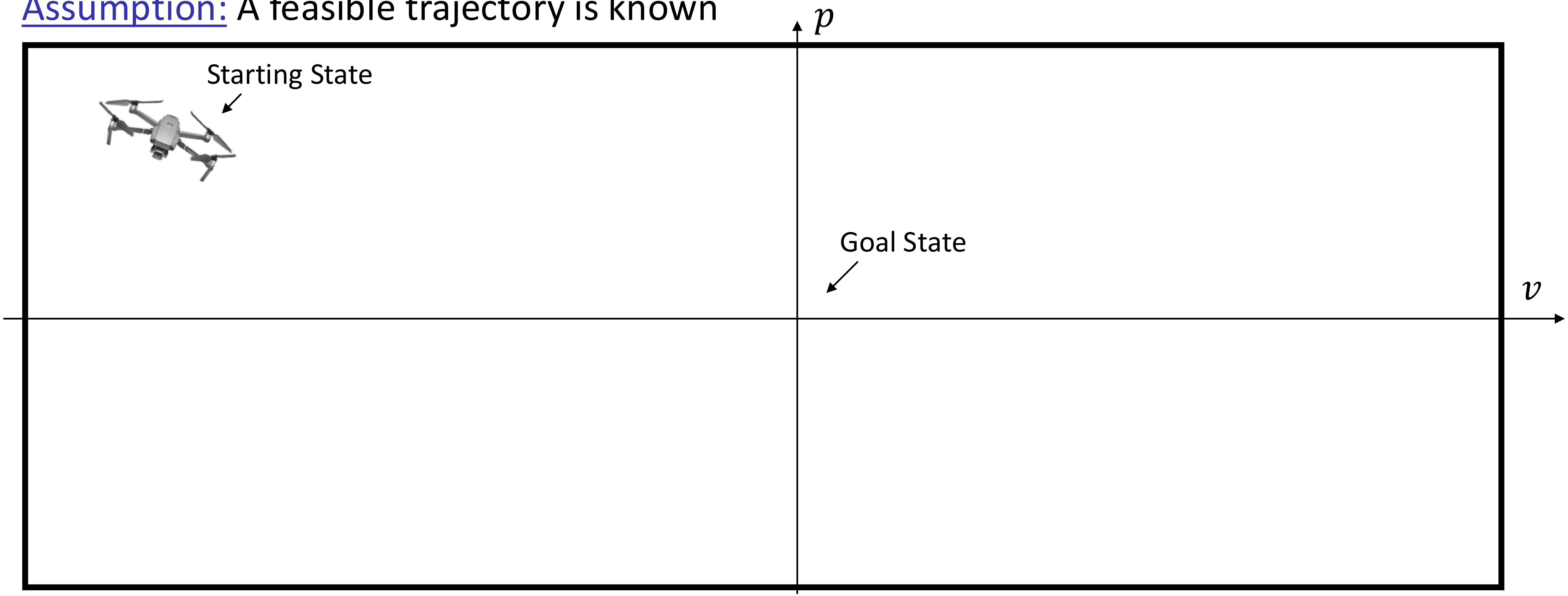

Algorithm steps:

- Get current state
- Plan a trajectory
- Execute the action

# Learning the Safe Set

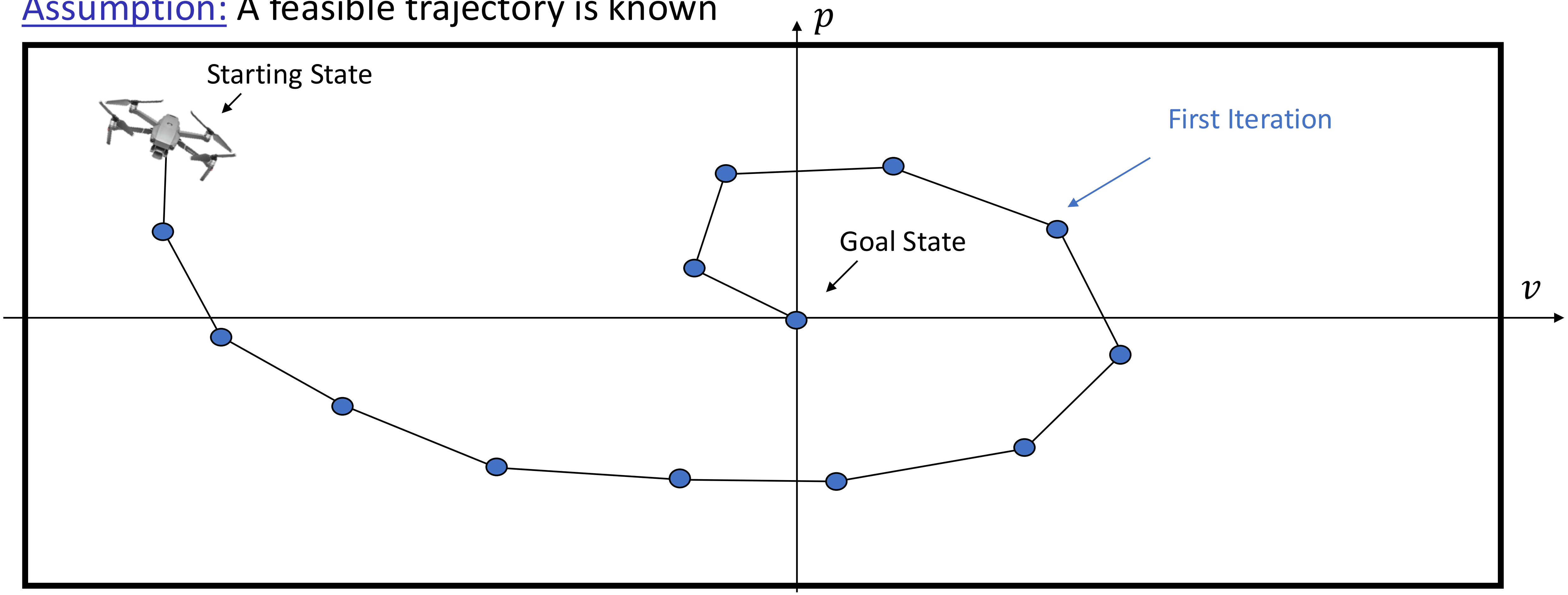
# Iteration 1

Assumption: A feasible trajectory is known



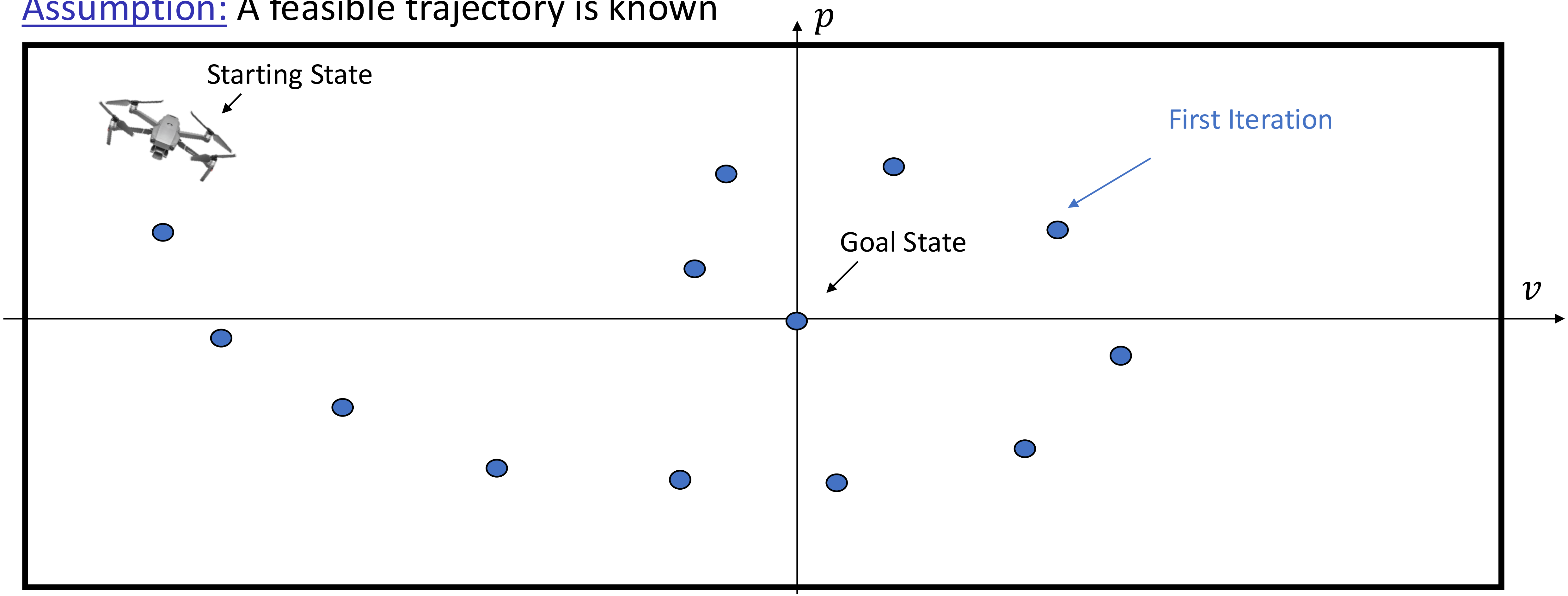
# Iteration 1

Assumption: A feasible trajectory is known



# Iteration 1

Assumption: A feasible trajectory is known

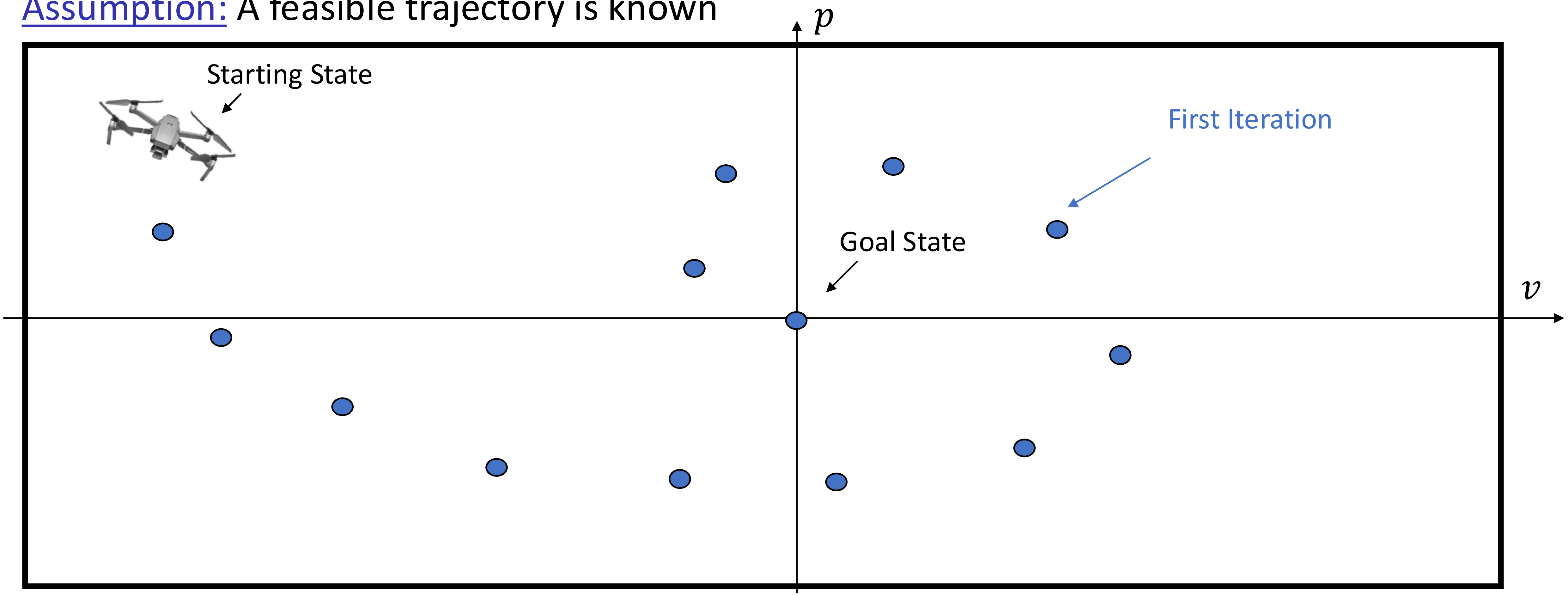



**Definition: Sampled Safe Set**

$$\mathcal{SS}^1 = \{\text{Stored Data}\}$$

# Iteration 1

Assumption: A feasible trajectory is known



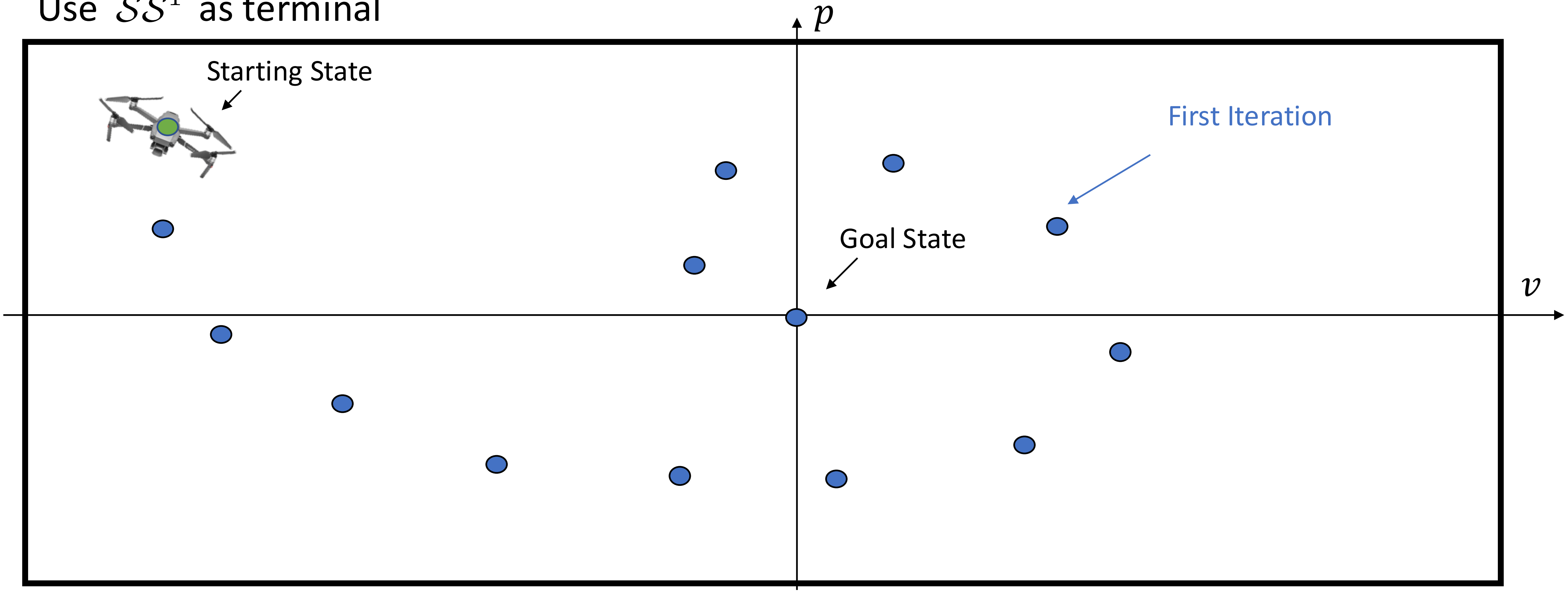

**Definition: Sampled Safe Set**

$$\mathcal{SS}^1 = \{\text{Stored Data}\}$$

Set of states from which the task can be completed!

# Iteration 2, Step 0

Use $\mathcal{SS}^1$ as terminal

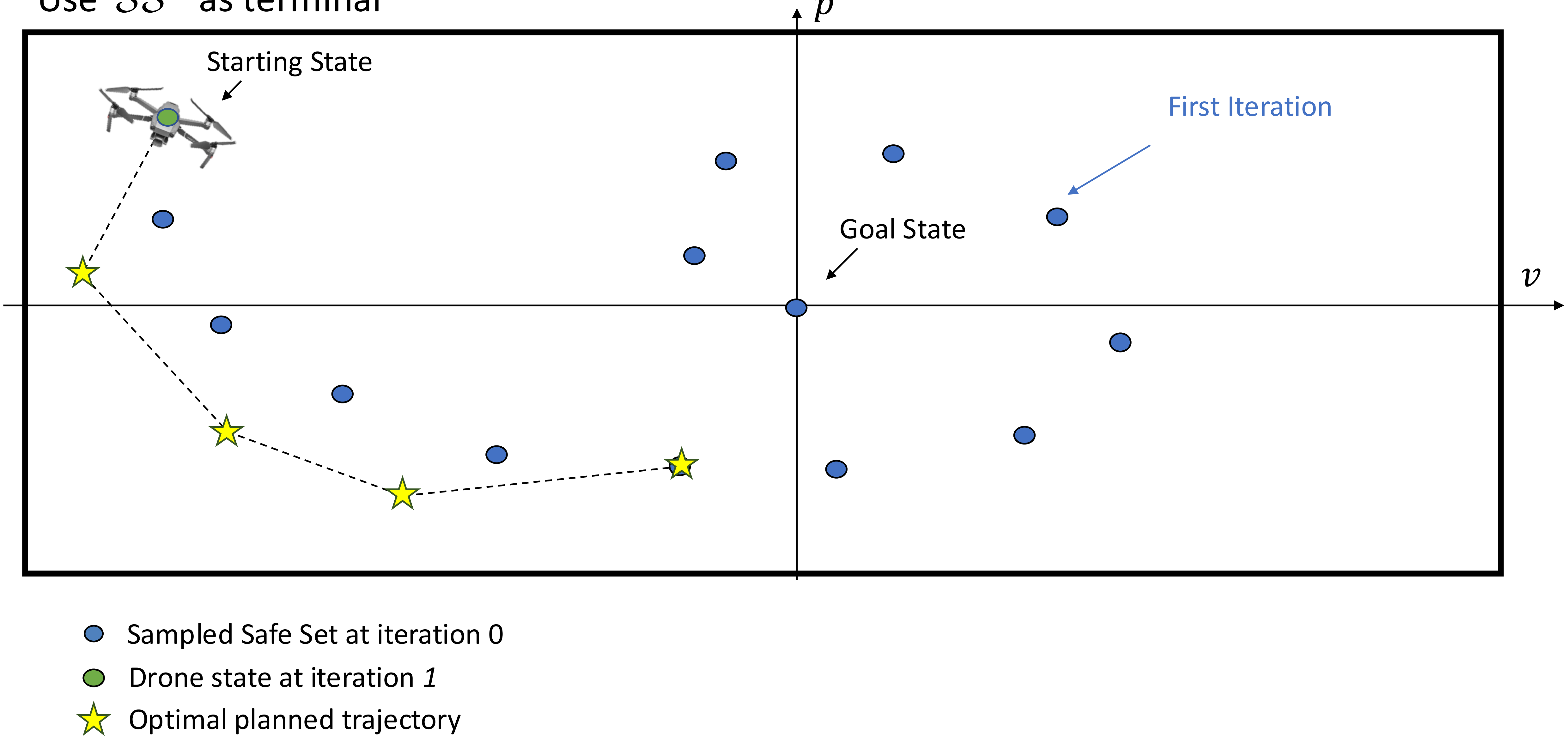



Sampled Safe Set at iteration 0

Drone state at iteration *1*

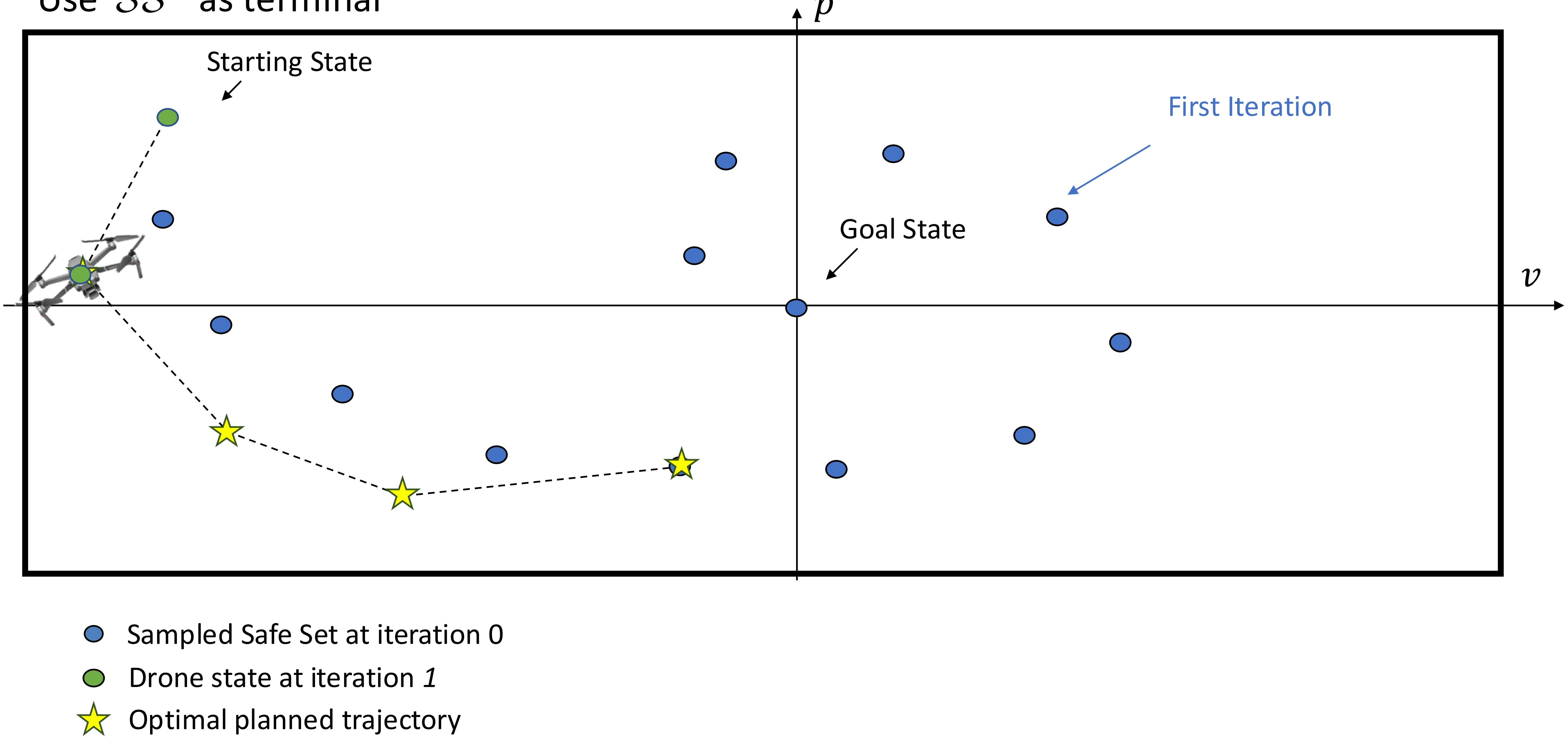
# Iteration 2, Step 0

Use $\mathcal{SS}^1$ as terminal



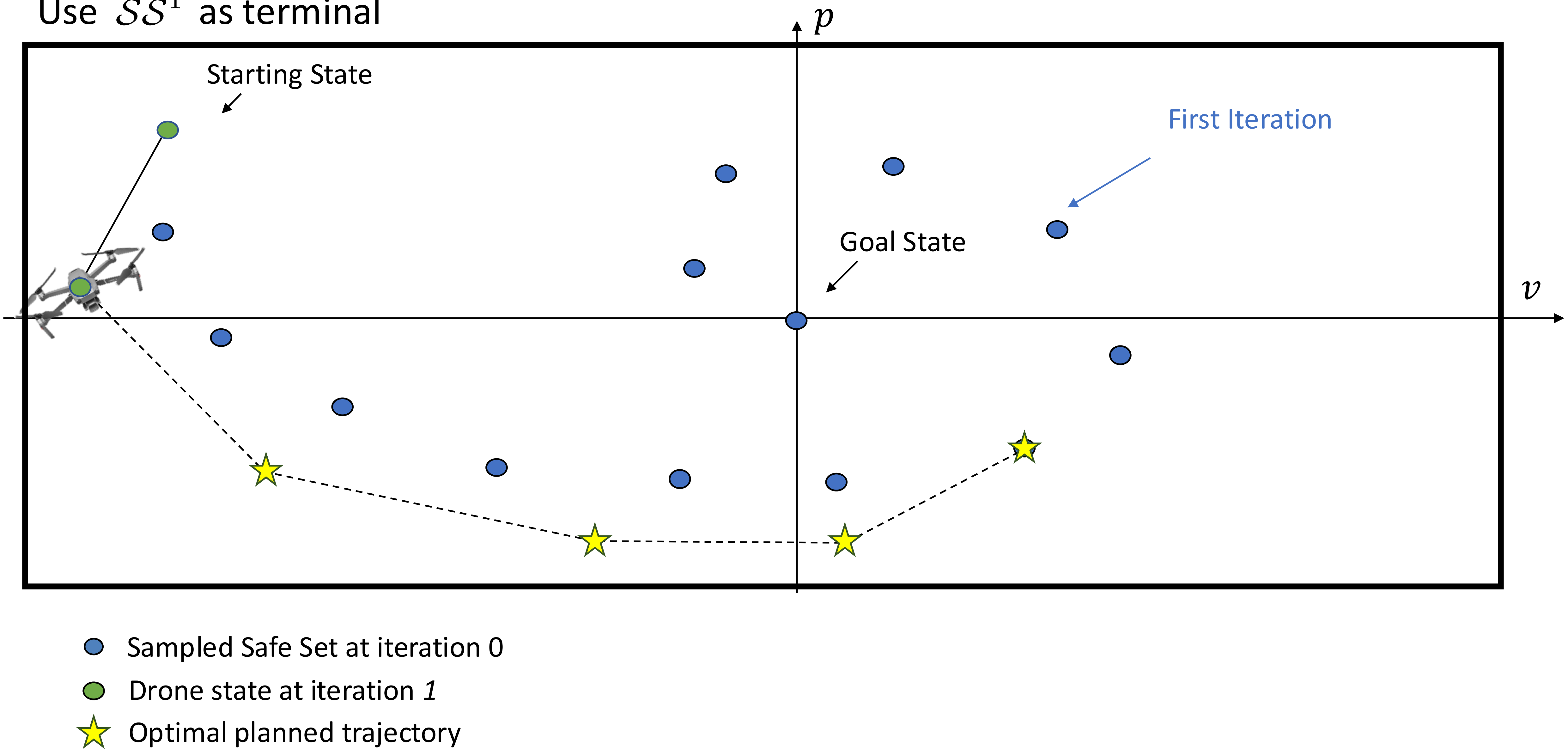
# Iteration 2, Step 1

Use $\mathcal{SS}^1$ as terminal



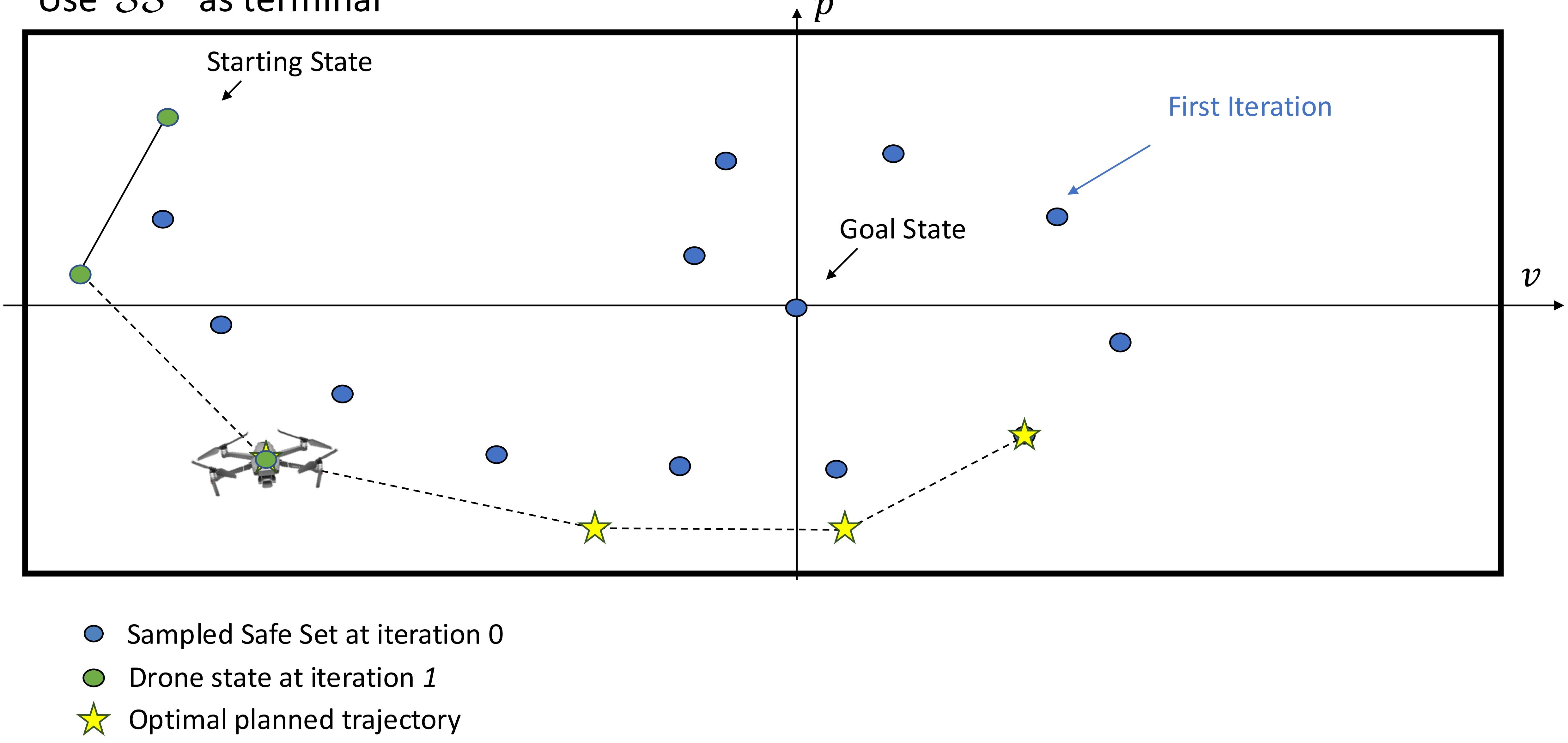
# Iteration 2, Step 1

Use $\mathcal{SS}^1$ as terminal



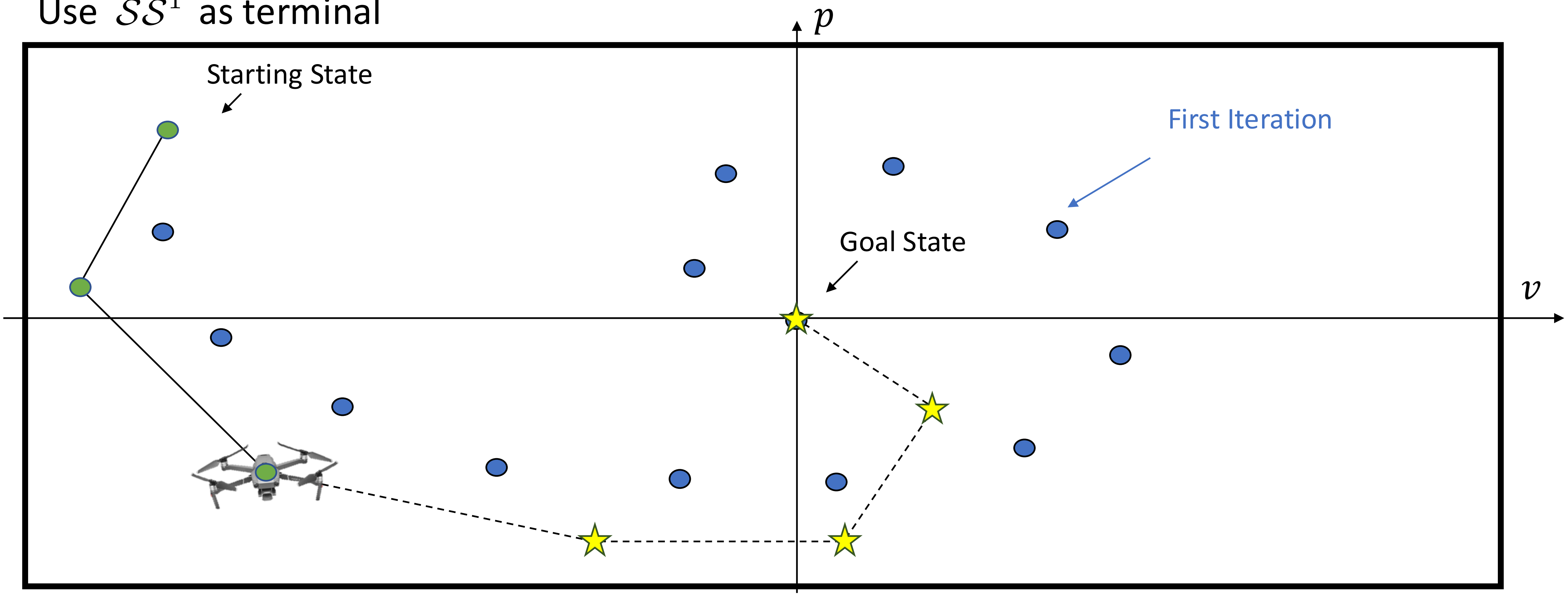
# Iteration 2, Step 2

Use $\mathcal{SS}^1$ as terminal



# Iteration 2, Step 2

Use $\mathcal{SS}^1$ as terminal



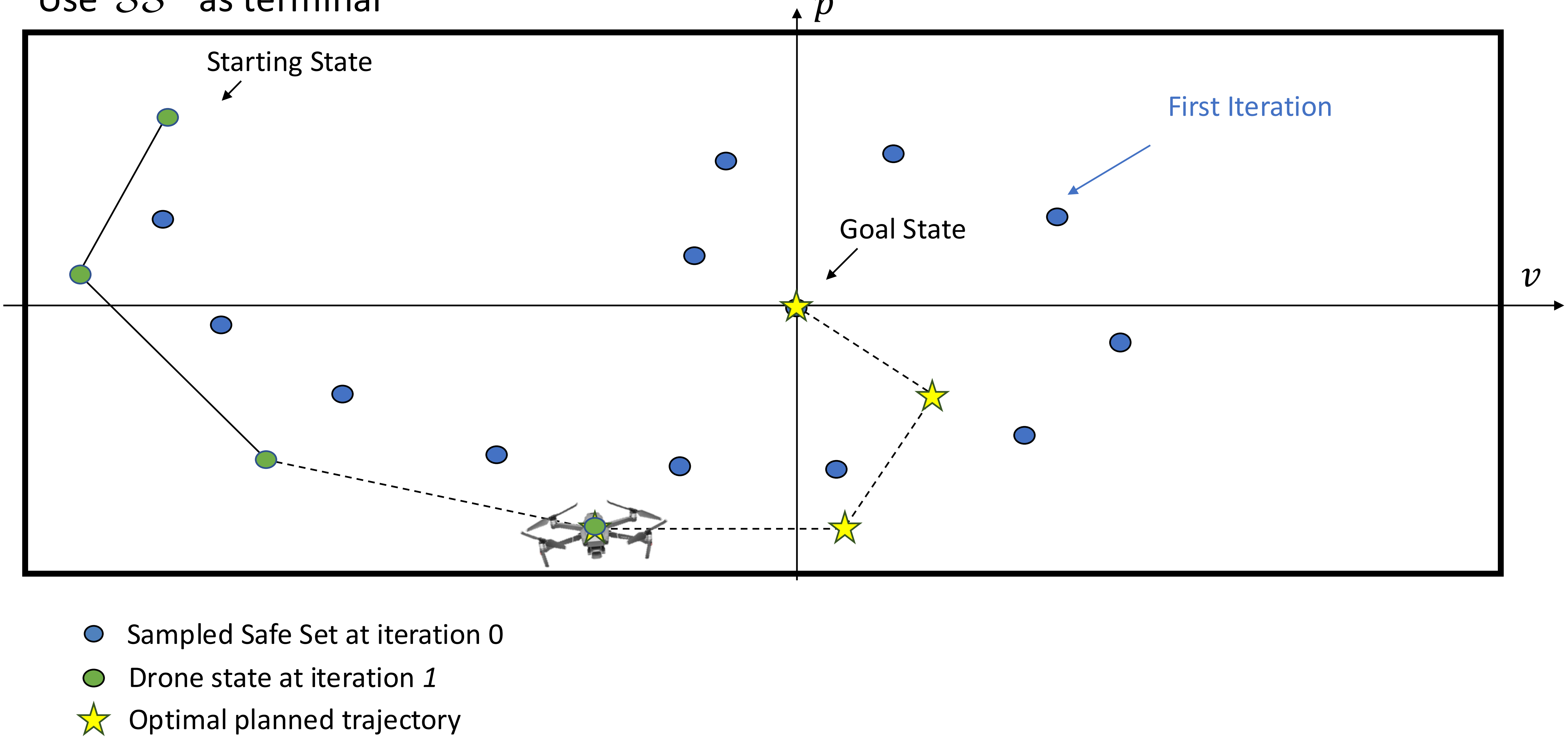

Sampled Safe Set at iteration 0

Drone state at iteration *1*

Optimal planned trajectory

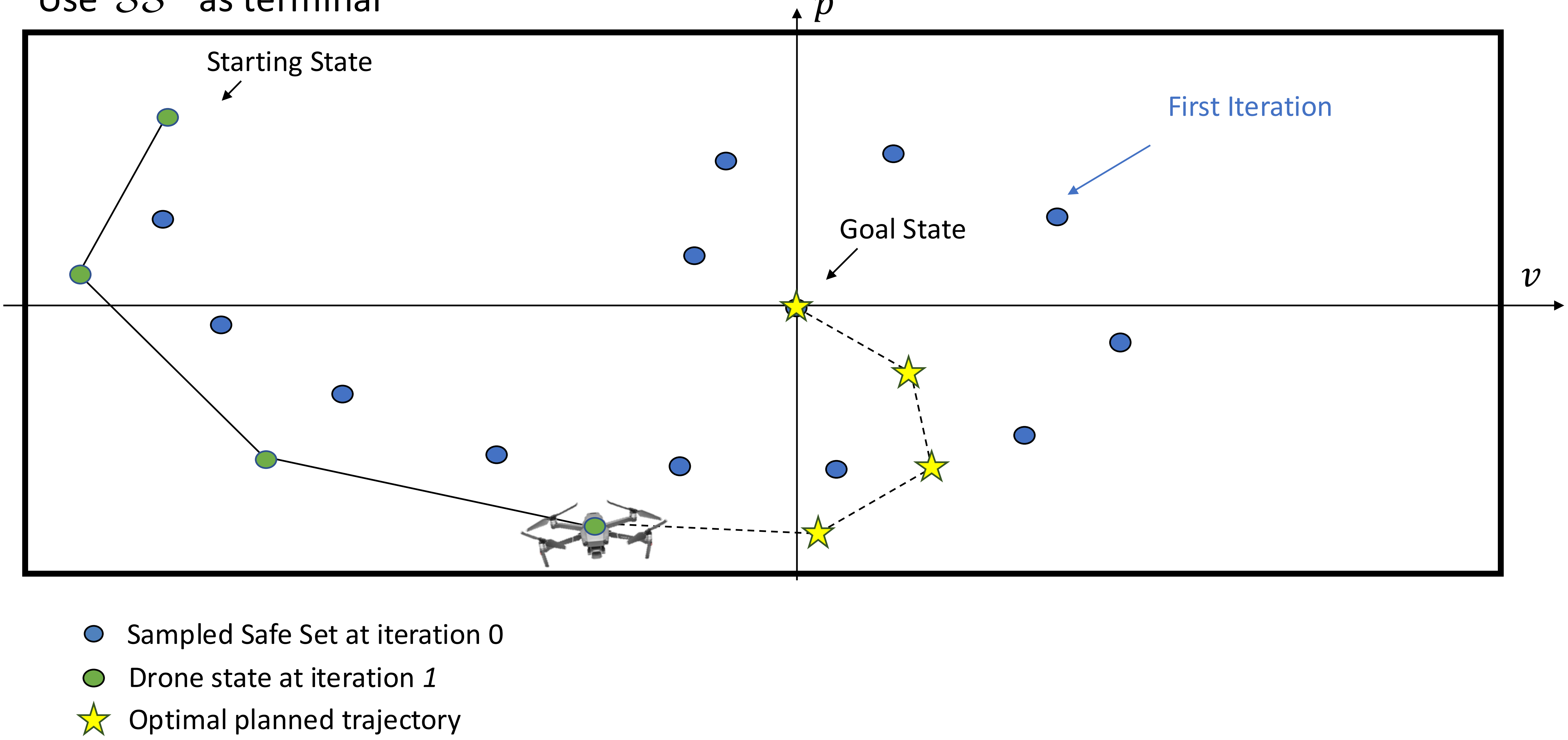
# Iteration 2, Step 3

Use $\mathcal{SS}^1$ as terminal



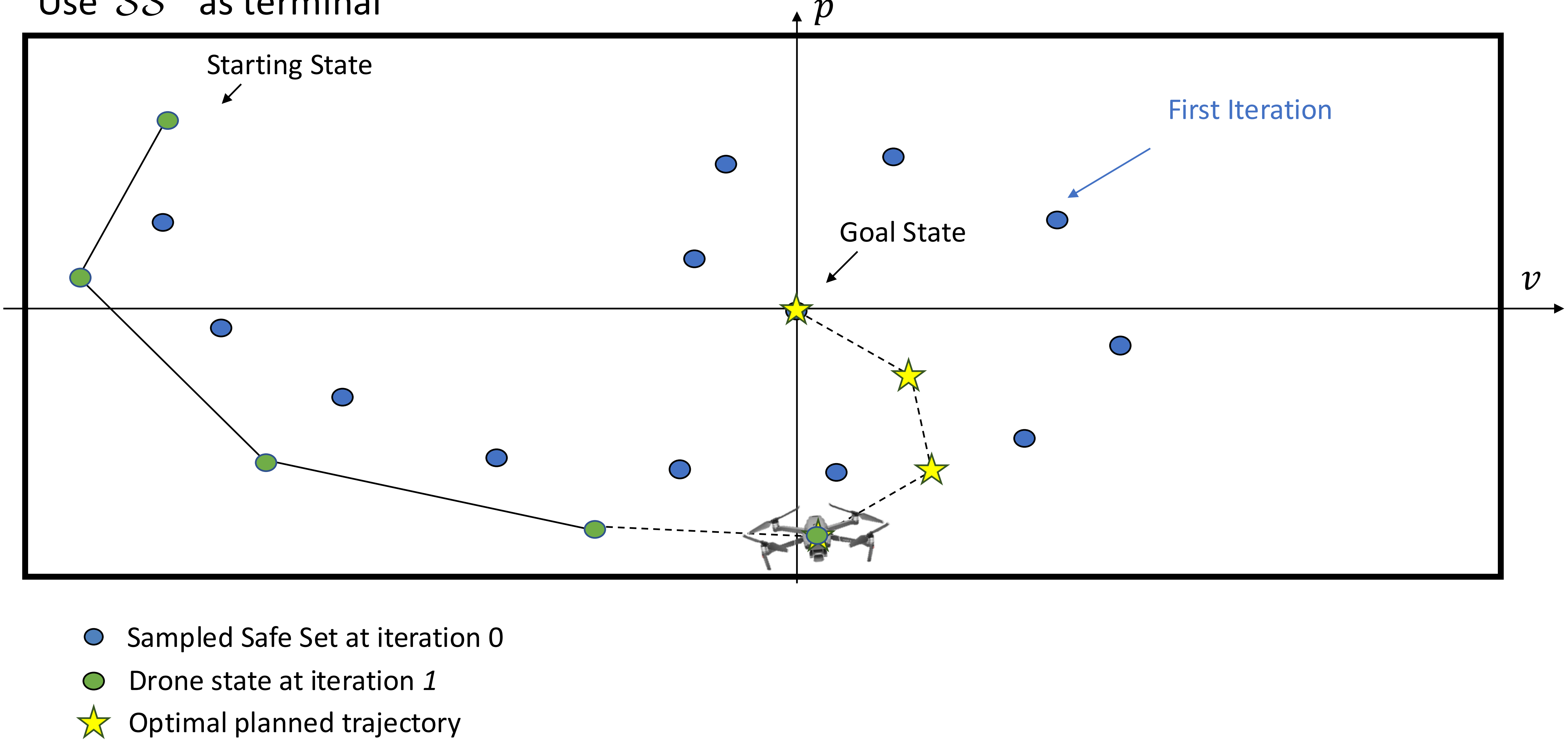
# Iteration 2, Step 3

Use $\mathcal{SS}^1$ as terminal



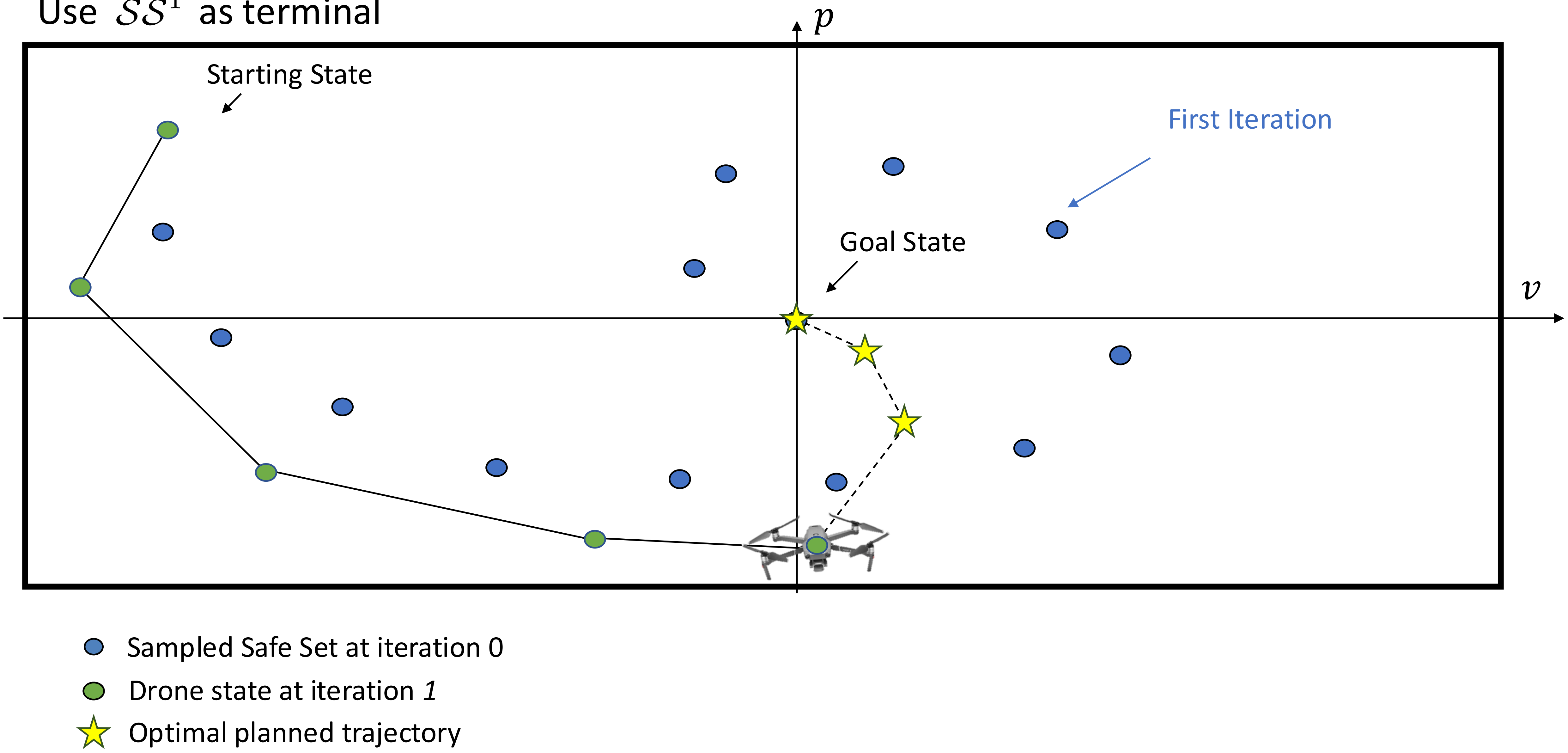
# Iteration 2, Step 4

Use $\mathcal{SS}^1$ as terminal



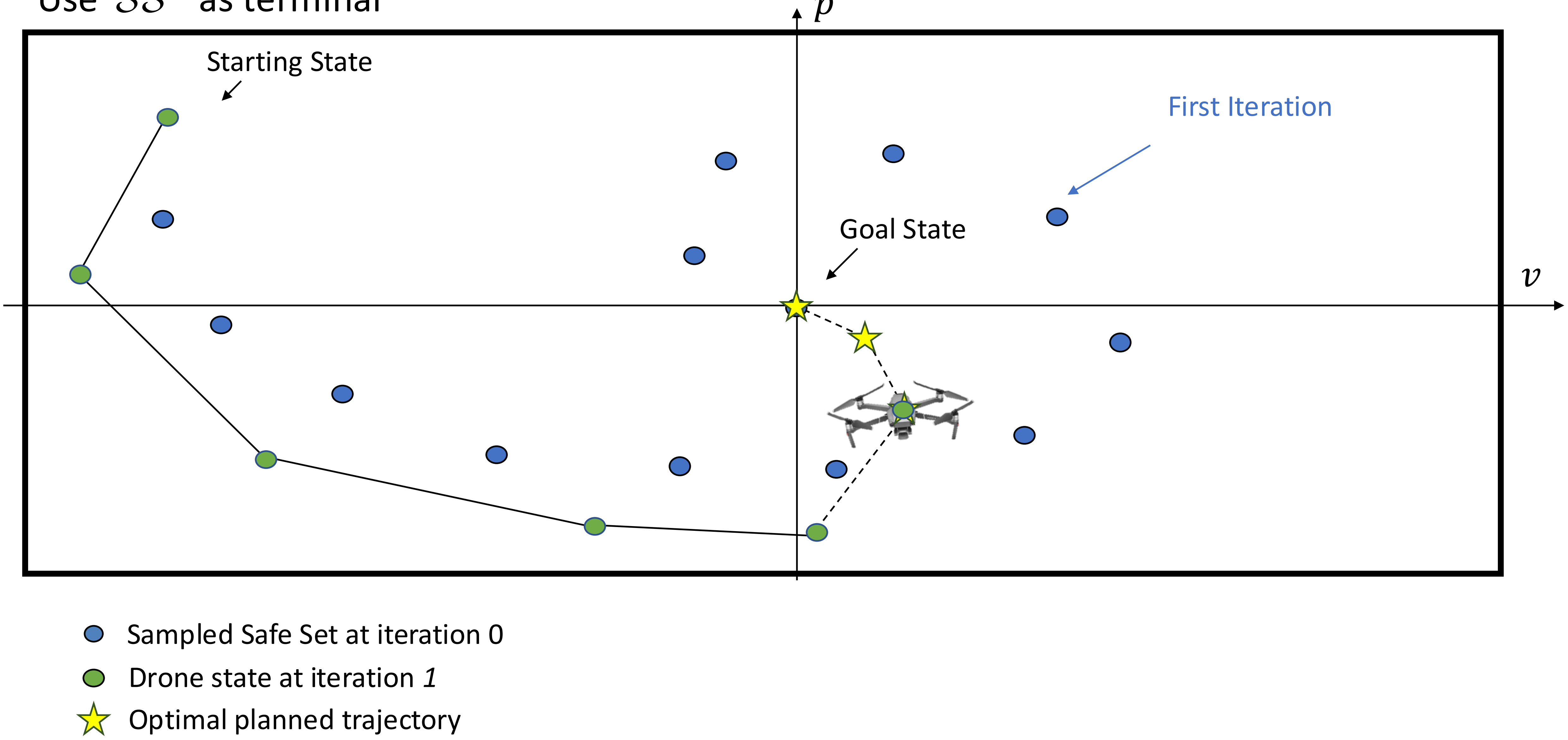
# Iteration 2, Step 4

Use $\mathcal{SS}^1$ as terminal



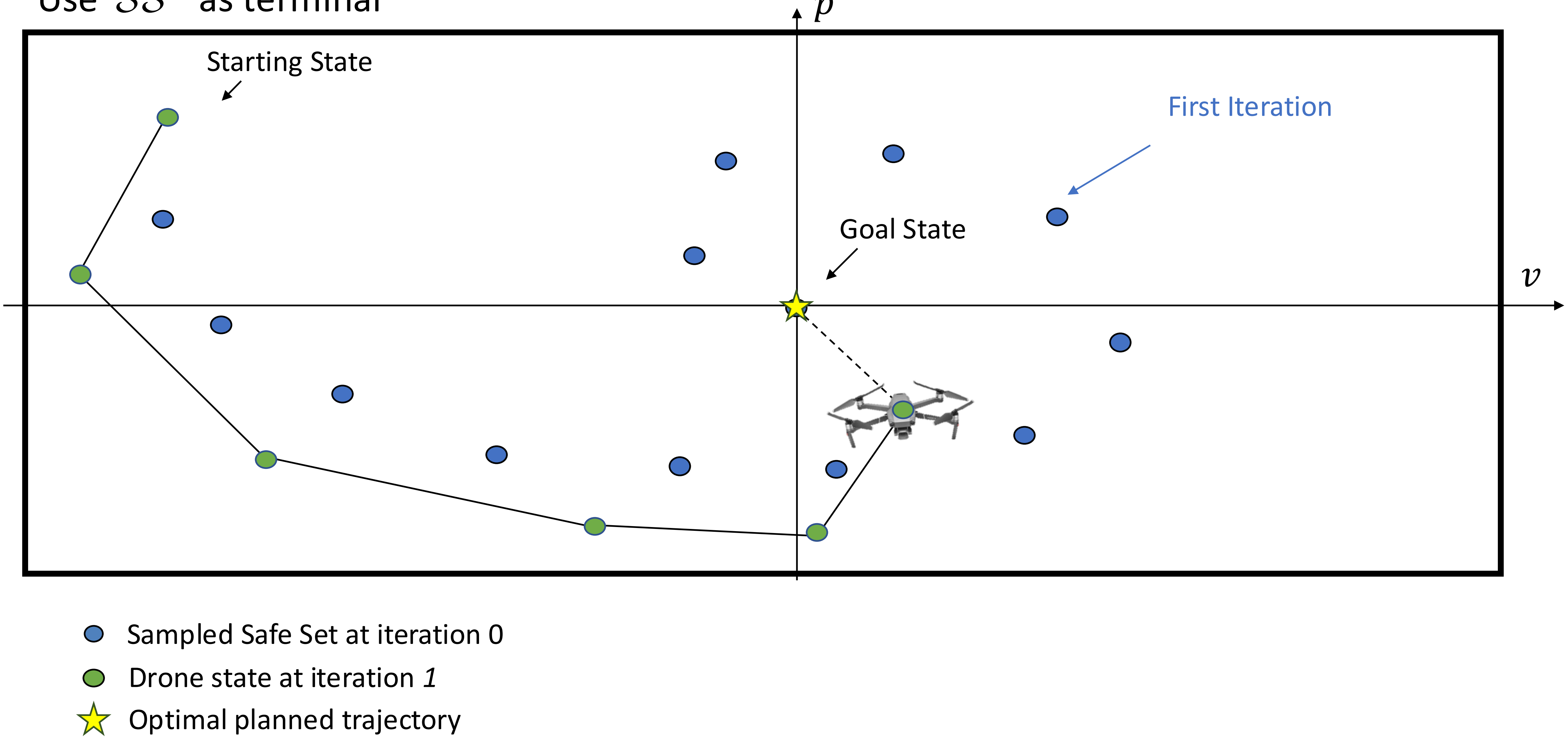
# Iteration 2, Step 5

Use $\mathcal{SS}^1$ as terminal



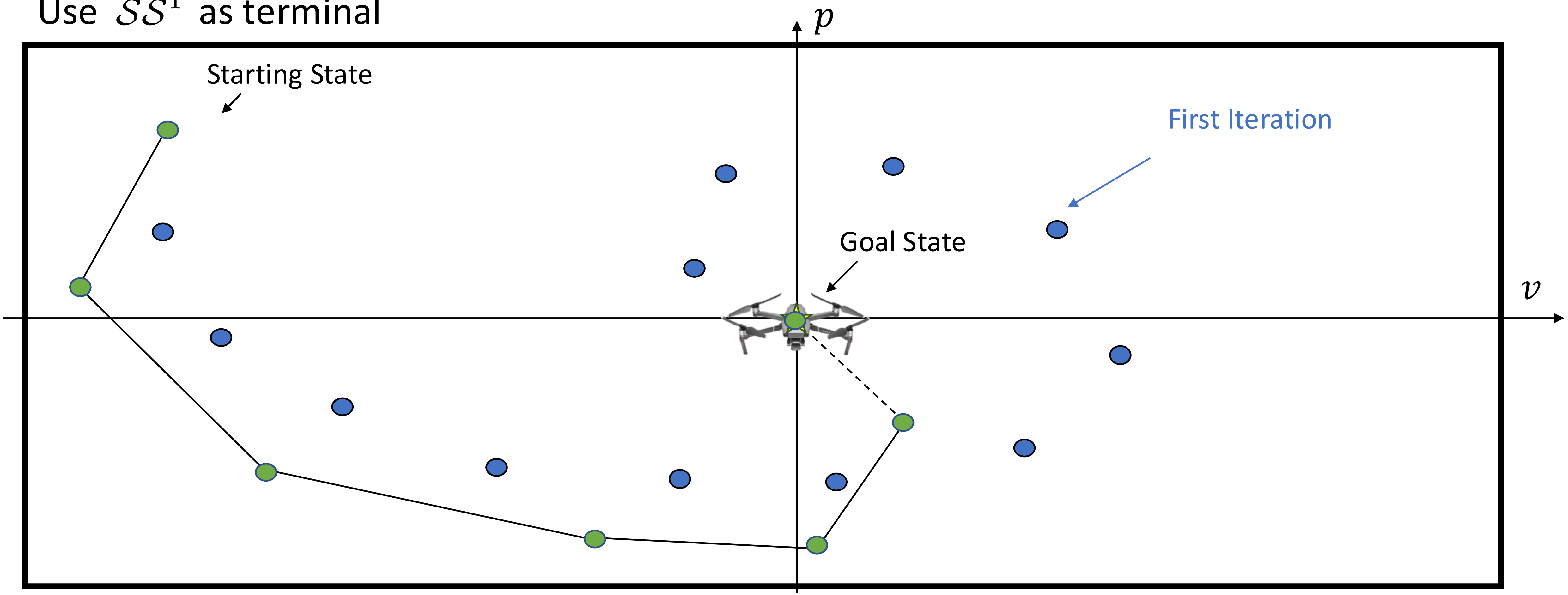
# Iteration 2, Step 5

Use $\mathcal{SS}^1$ as terminal



# Iteration 2, Step 5

Use $\mathcal{SS}^1$ as terminal



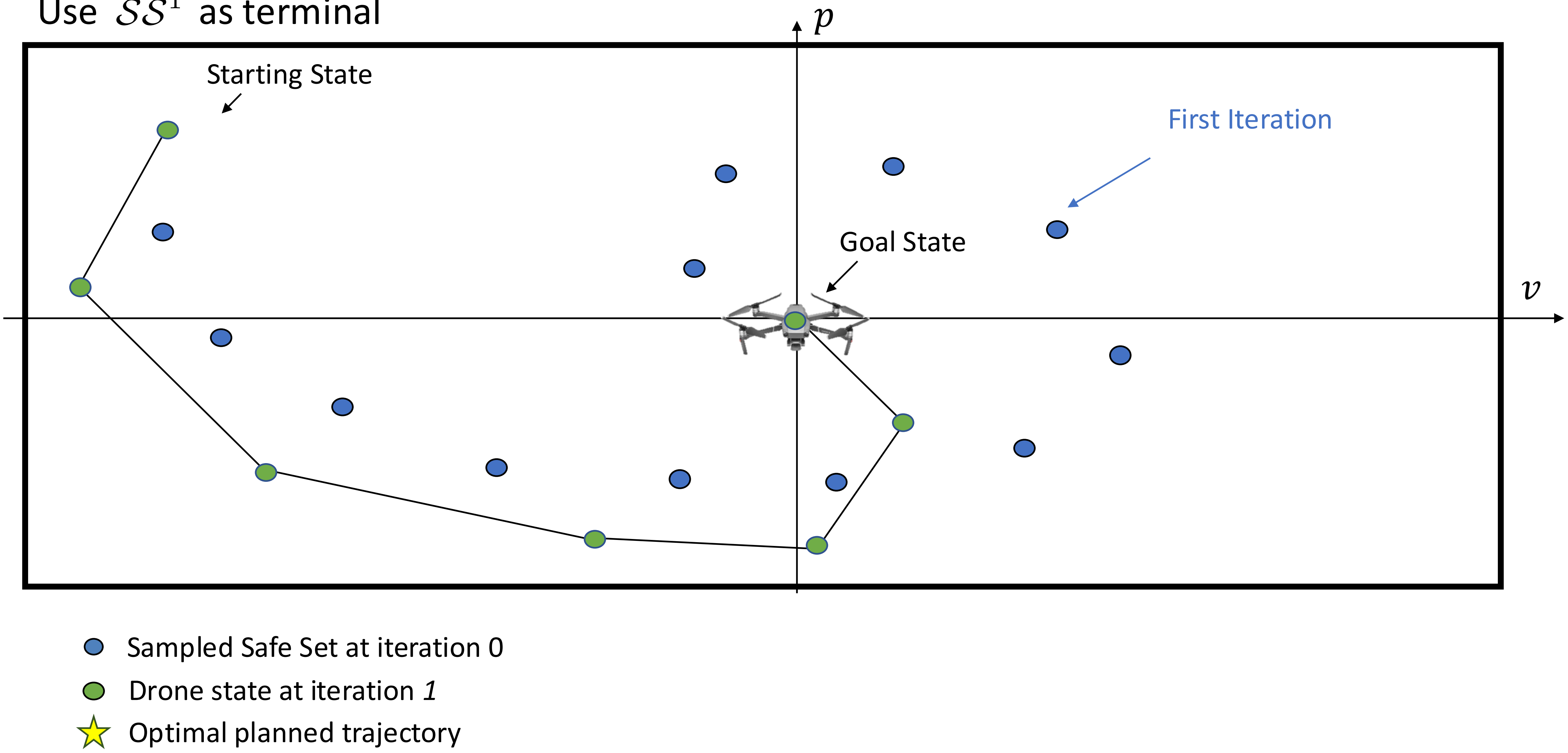

Sampled Safe Set at iteration 0

Drone state at iteration *1*

Optimal planned trajectory

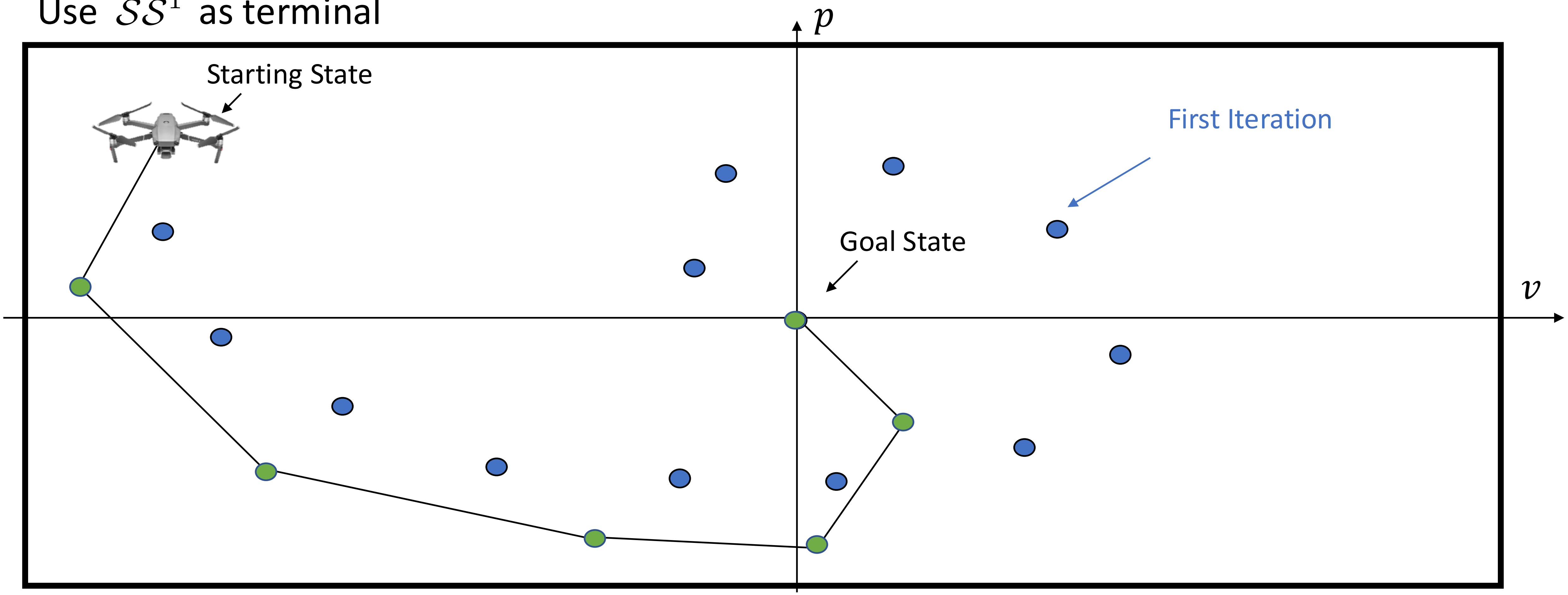
# Iteration 2, Step 5

Use $\mathcal{SS}^1$ as terminal



# Iteration 2, Step 5

Use $\mathcal{SS}^1$ as terminal



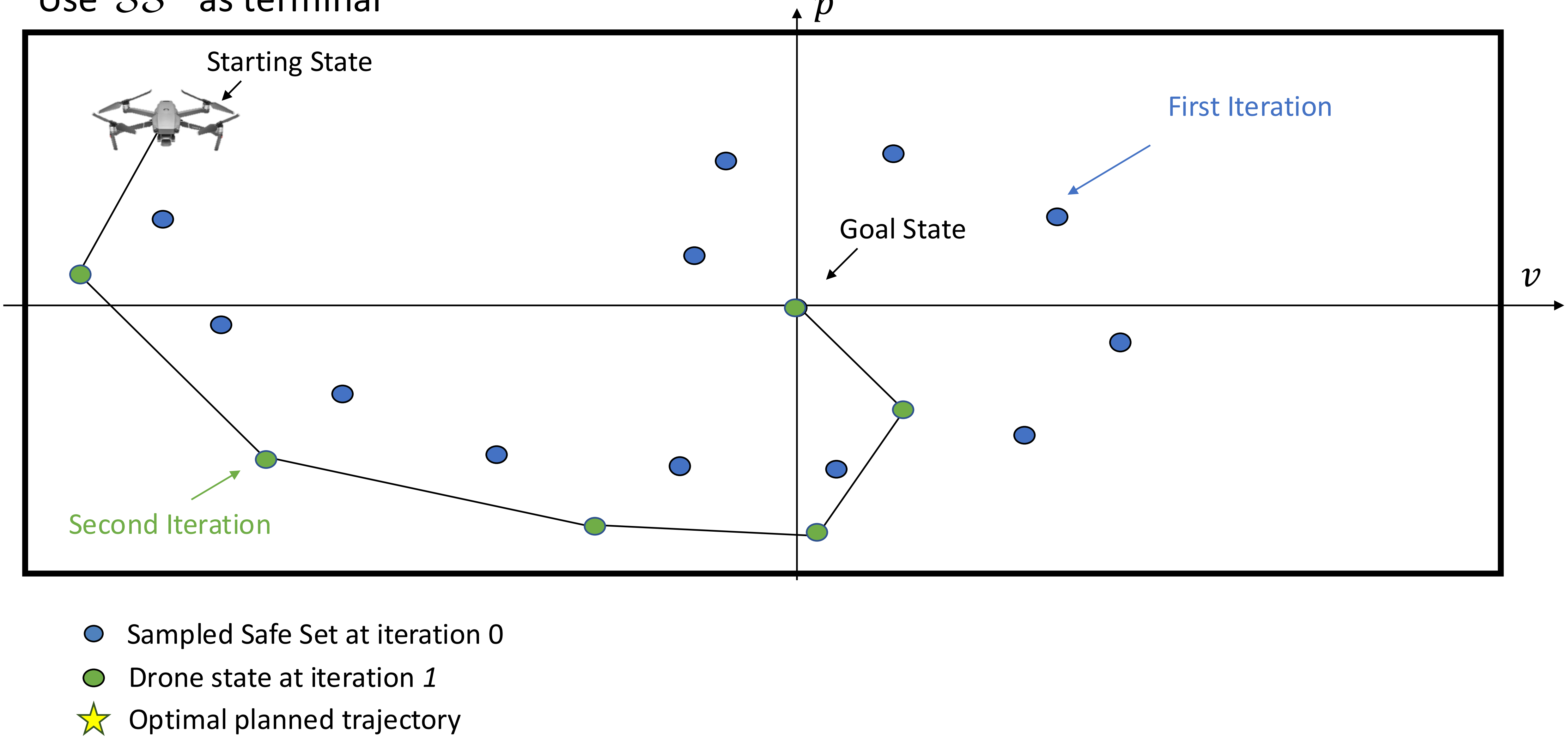

Sampled Safe Set at iteration 0

Drone state at iteration *1*

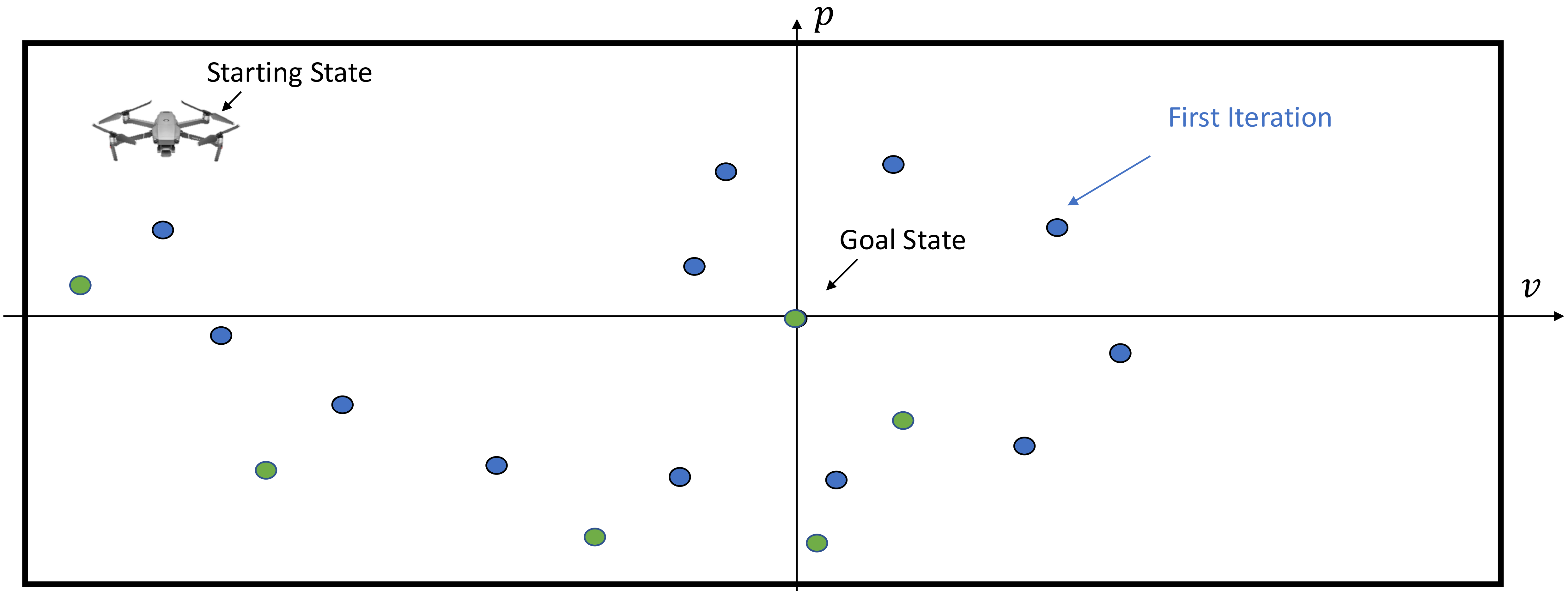
Optimal planned trajectory

# Iteration 2, Step 5

Use $\mathcal{SS}^1$ as terminal



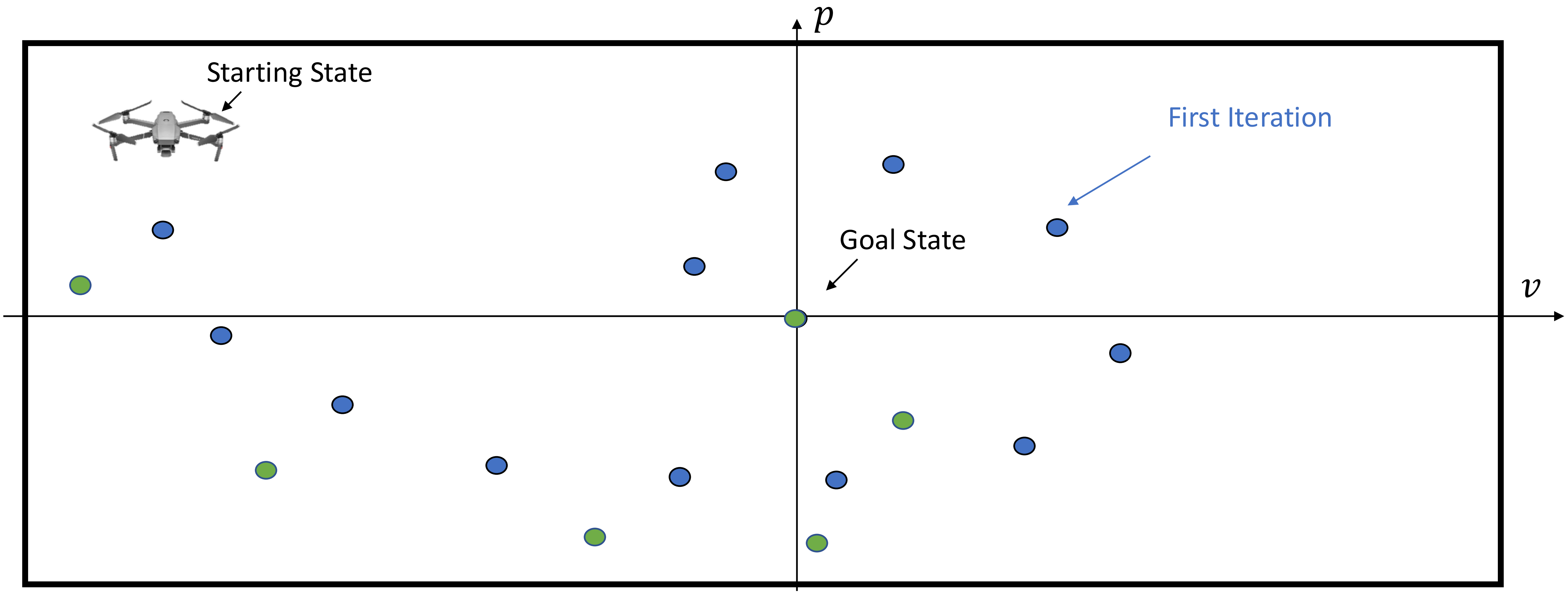
# Iteration 3




**Definition: Sampled Safe Set**

$$\mathcal{SS}^j = \{\text{Stored Data at all iterations}\}$$

# Iteration 3

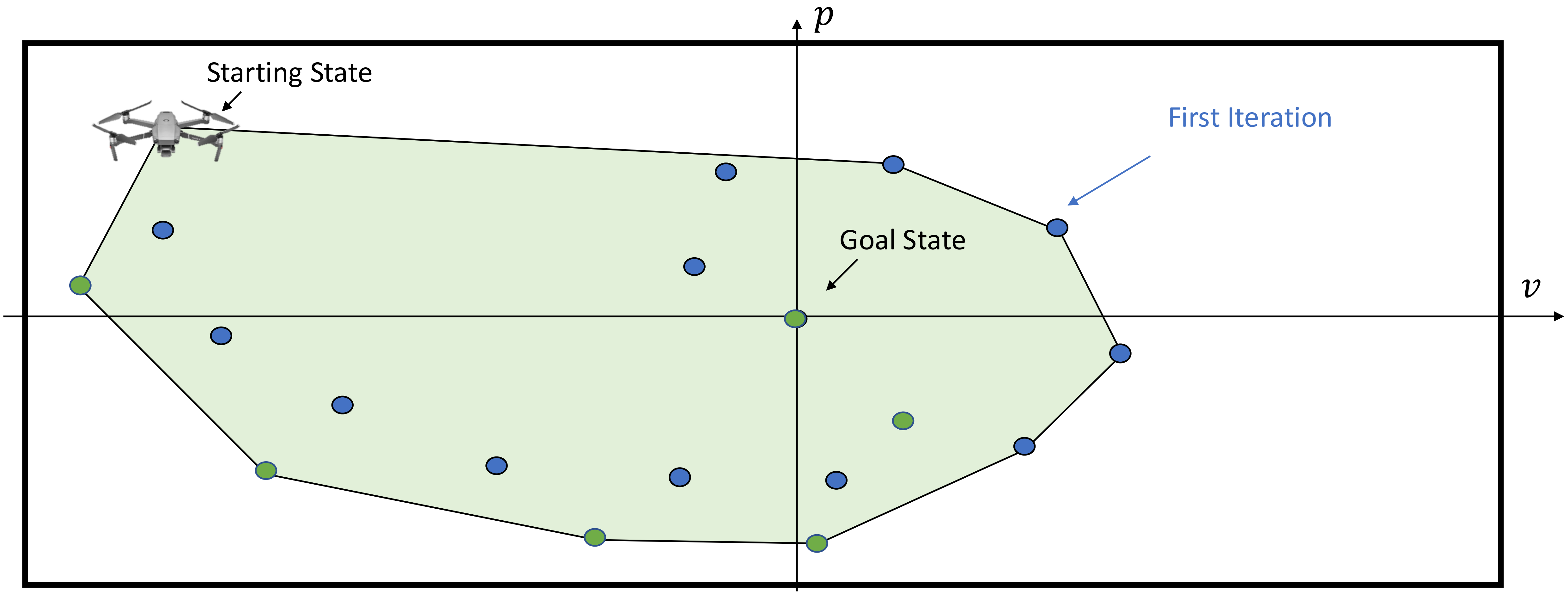



**Definition: Sampled Safe Set**

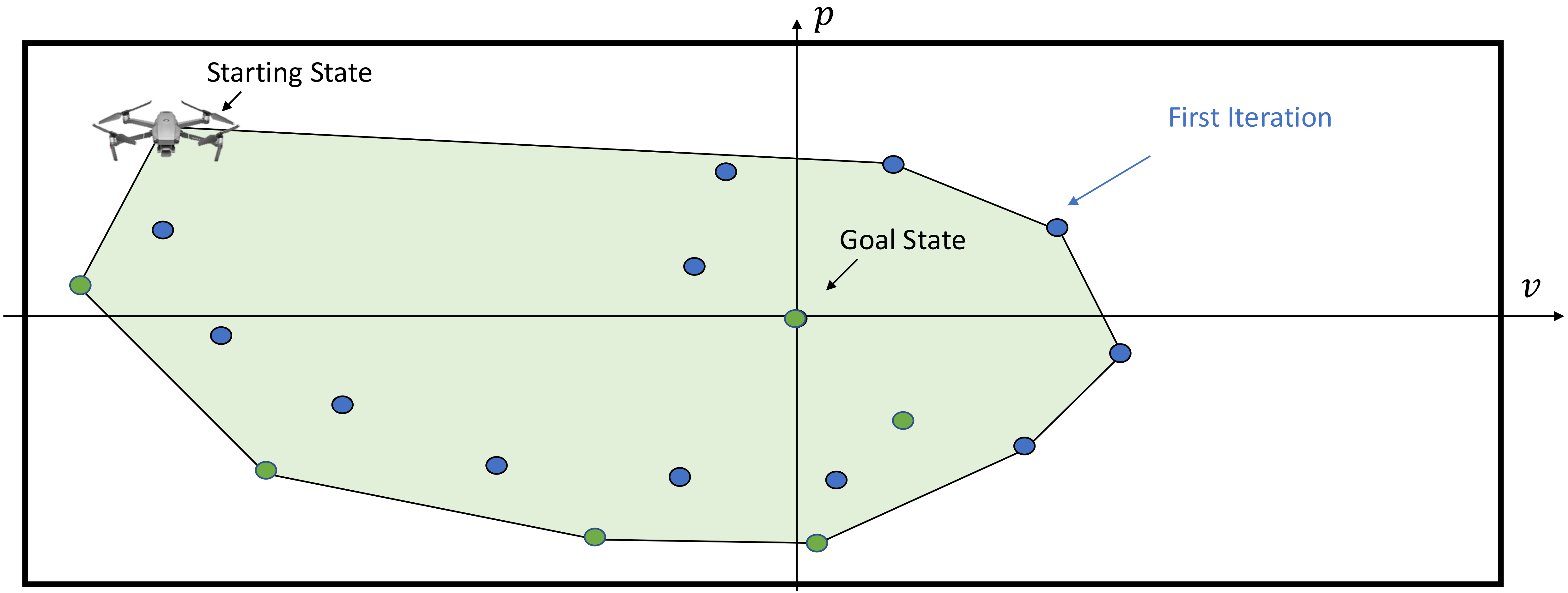
$$\mathcal{SS}^j = \{\text{Stored Data at all iterations}\}$$

Set of states from which the task can be completed!

# Iteration 3



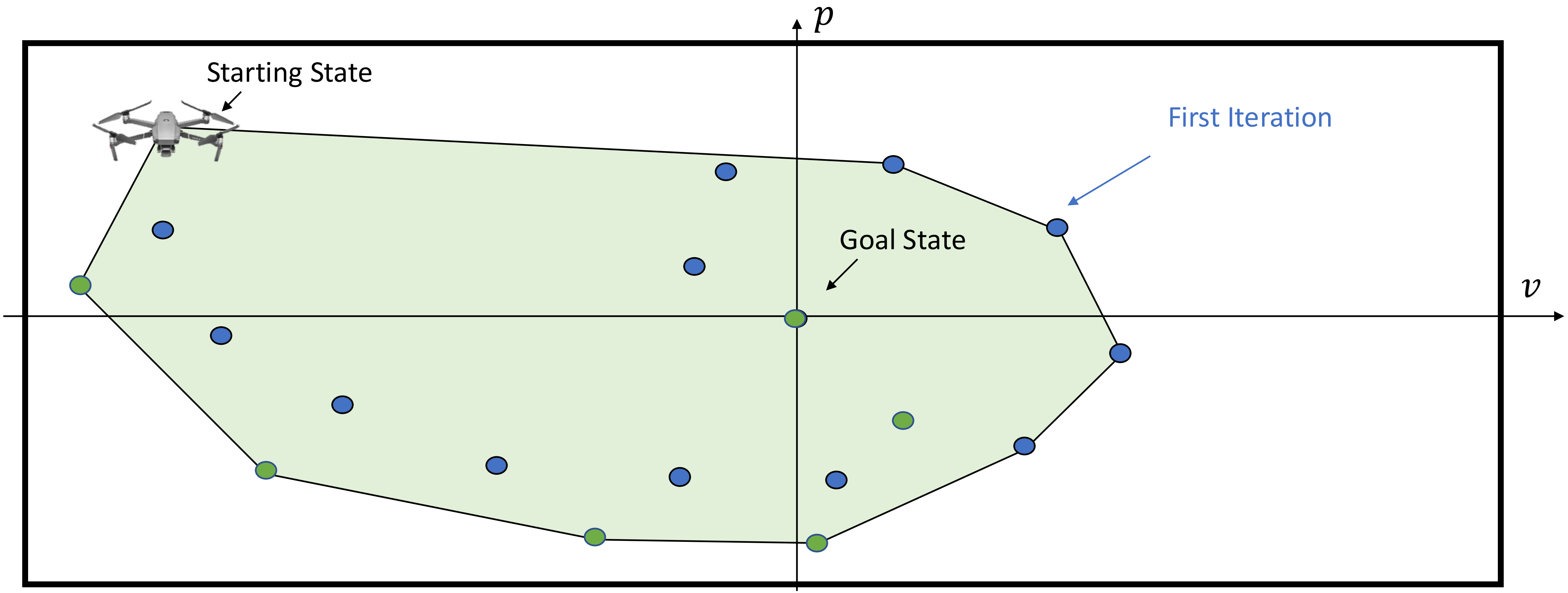
# Iteration 3




**Definition: Convex Safe Set**

$$\mathcal{CS}^j = \mathrm{Conv}(\{\text{Stored Data at all iterations}\})$$

# Iteration 3

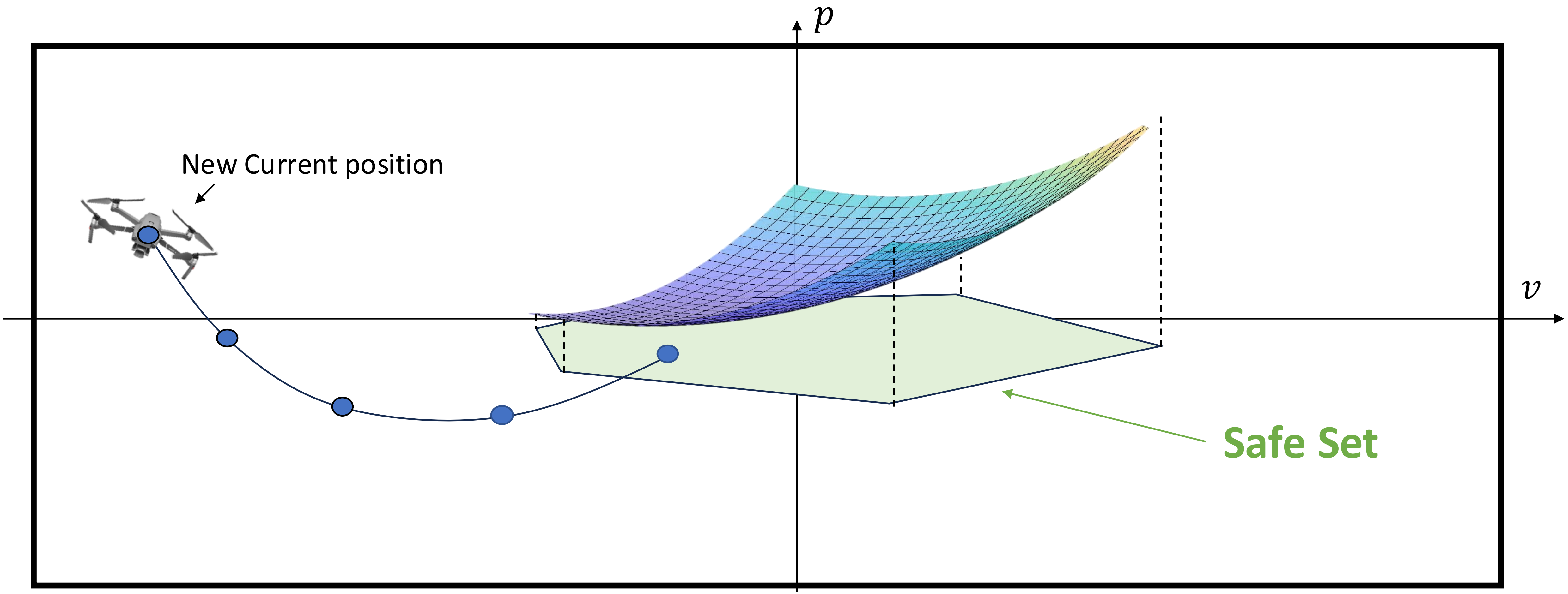



**Definition: Convex Safe Set**

$$\mathcal{CS}^j = \text{Conv}(\{\text{Stored Data at all iterations}\})$$

Set of states from which the task can be completed!

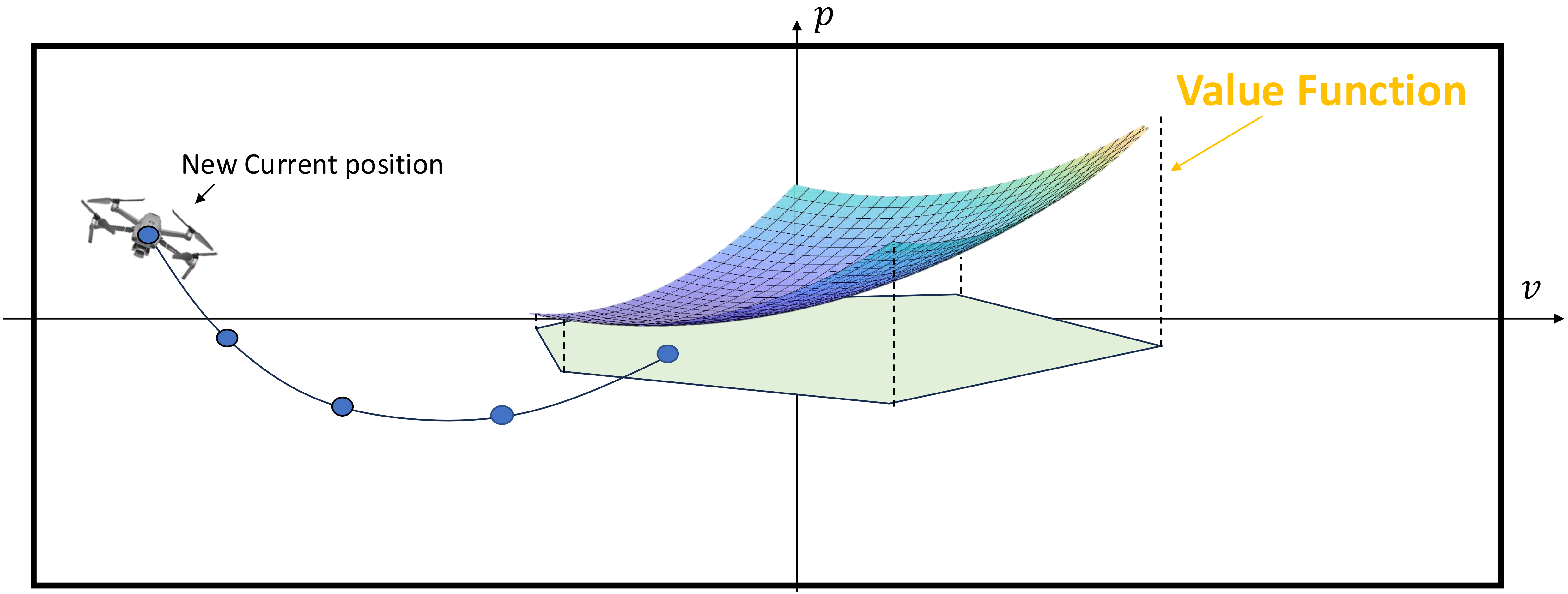
# Learning Model Predictive Control (LMPC) – Key Idea




Algorithm steps:

- Get current state
- Plan a trajectory
- Execute the action

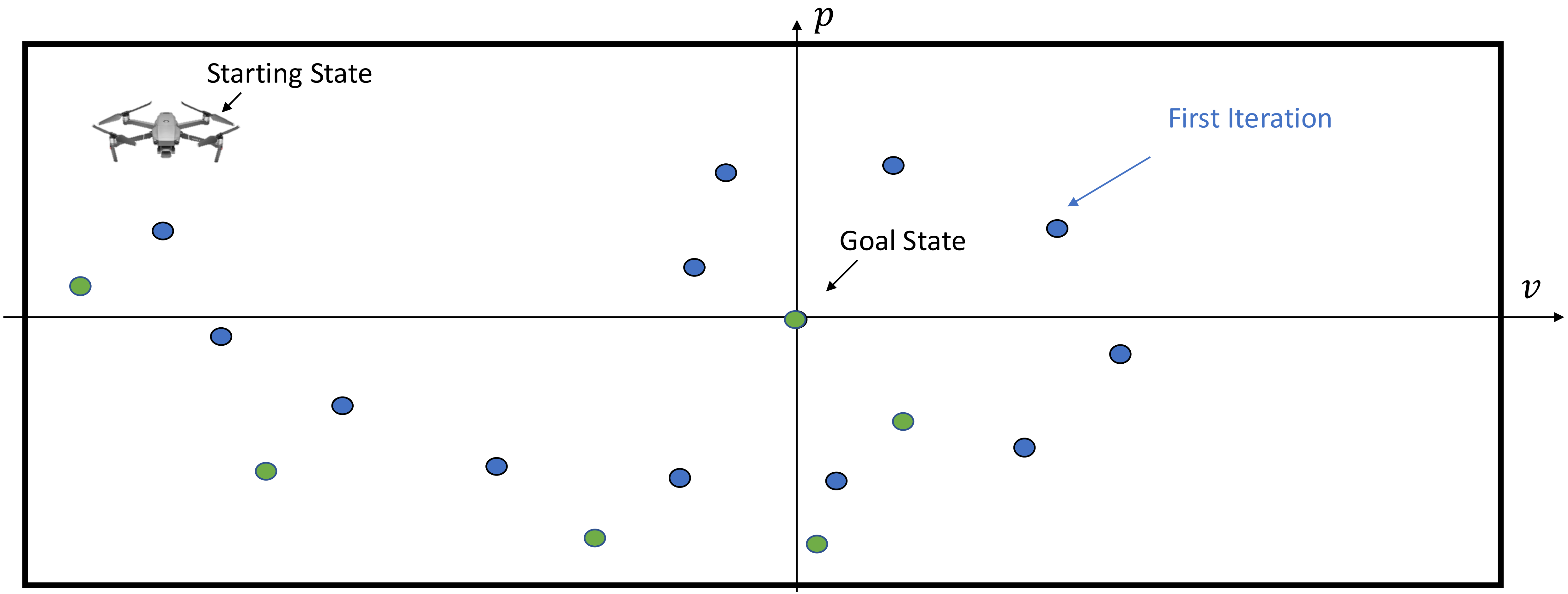
# Learning Model Predictive Control (LMPC) – Key Idea



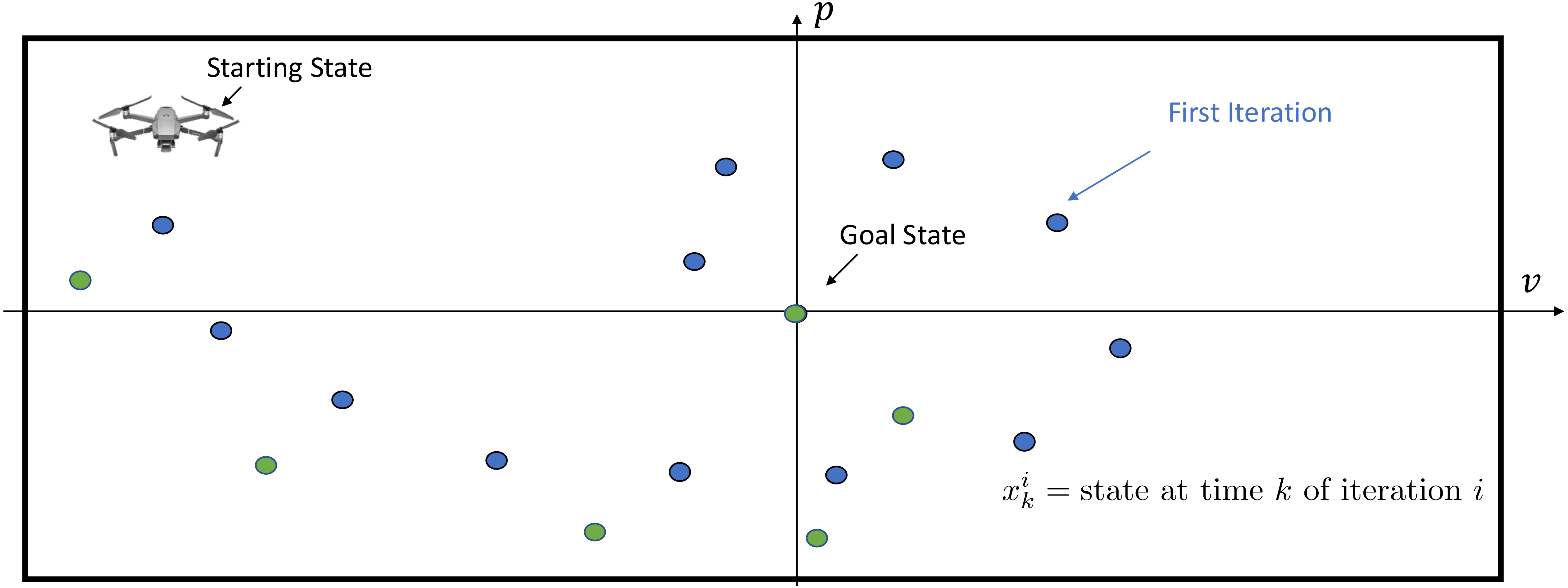

Algorithm steps:

- Get current state
- Plan a trajectory
- Execute the action

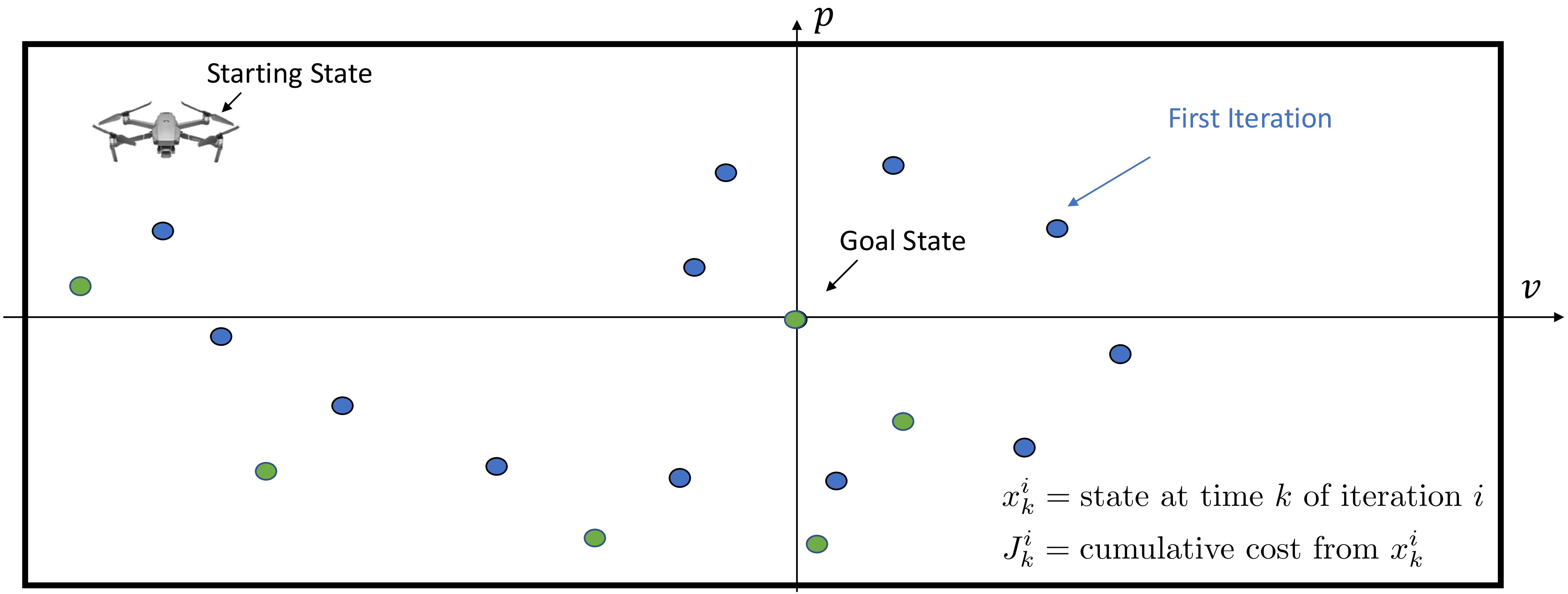
# Value Function Estimation



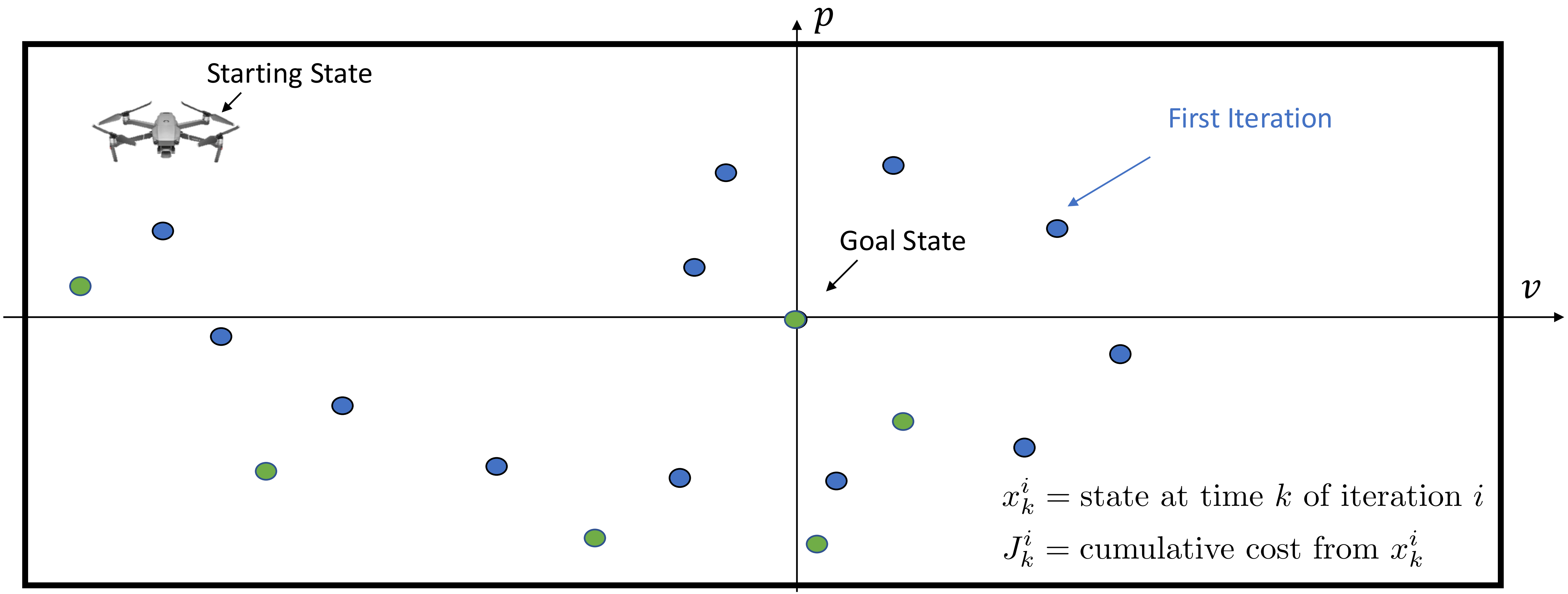
# Value Function Estimation



# Value Function Estimation



# Value Function Estimation



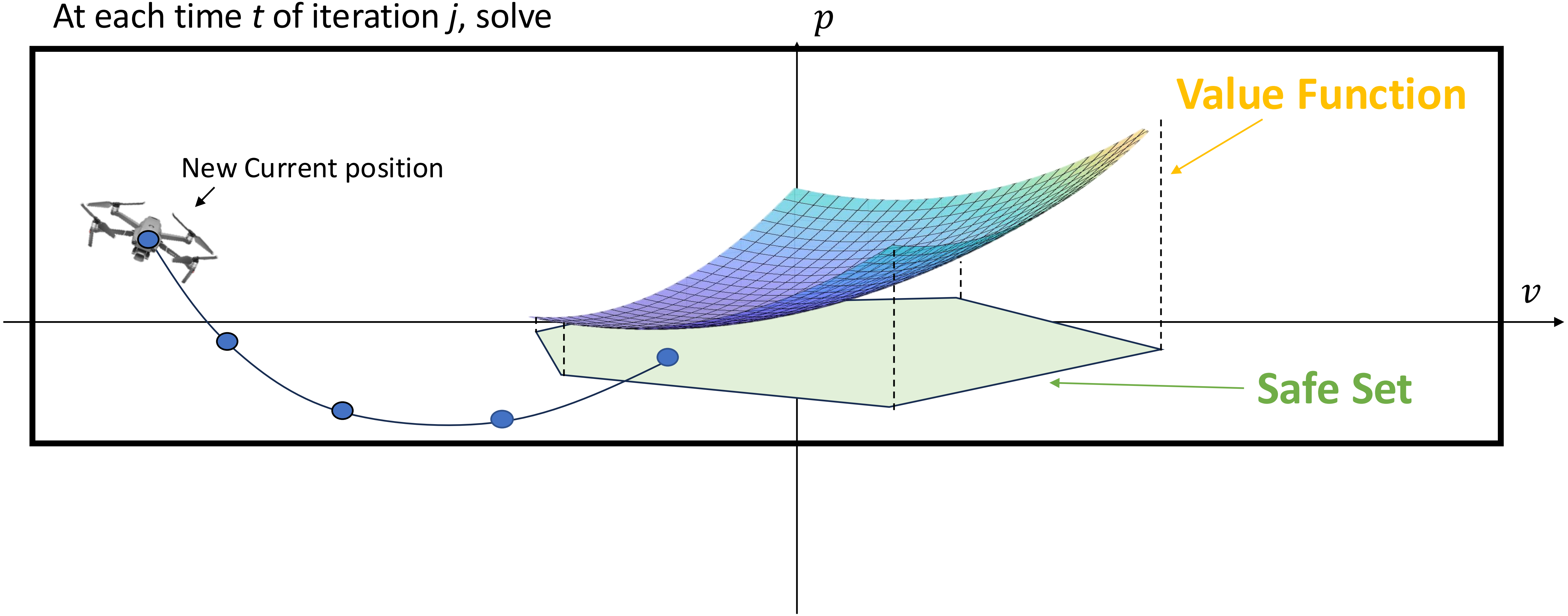

**Value Function**

$$V^j(x) = \text{Interpolation of the set of data } \{\{(J_k^i, x_k^i)\}_{i=0}^{j}\}_{k=0}^{K}$$

# LMPC Summary

# LMPC Summary

At each time $t$ of iteration $j$, solve



# LMPC Summary

At each time *t* of iteration *j*, solve

$$
\begin{aligned}
J(x(t)) = \min_{u_0,\ldots,u_{N-1}} \quad & \sum_{k=0}^{N-1} \left( x_k^\top Q x_k + u_k^\top R u_k \right) + V^{j-1}(x_N) \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k), \\
& x_0 = x(t), \\
& x_k \in \mathcal{X}, \ u_k \in \mathcal{U}, \\
& x_N \in \ \mathcal{SS}^{j-1}, \\
& \forall k \in [0, \cdots, N-1]
\end{aligned}
$$

# LMPC Summary

At each time *t* of iteration *j*, solve

$$
\begin{aligned}
J(x(t)) = \min_{u_0,\dots,u_{N-1}} \quad & \sum_{k=0}^{N-1} \left(x_k^\top Q x_k + u_k^\top R u_k\right) + V^{j-1}(x_N) \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k), \\
& x_0 = x(t), \\
& x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \\
& x_N \in \ \mathcal{SS}^{j-1}, \\
& \forall k \in [0, \cdots, N-1]
\end{aligned}
$$

**Guarantees for constrained (linear) systems [1,2]**

The properties of the (convex) safe set and (convex) V-function allows us to guarantee:

- Safety: constraint satisfaction at iteration *j* → satisfaction at iteration *j+1*
- Non-decreasing Performance: closed-loop cost at iteration *j* >= closed-loop cost at iteration *j+1*
- Performance Improvement: closed-loop cost strictly deceasing at each iteration (LICQ required)
- (Global) optimality: steady state trajectory is optimal for the original problem (LICQ required)


[1] U. Rosolia, F. Borrelli. “Learning model predictive control for iterative tasks. a data-driven control framework.” *IEEE Transactions on Automatic Control* (2018).

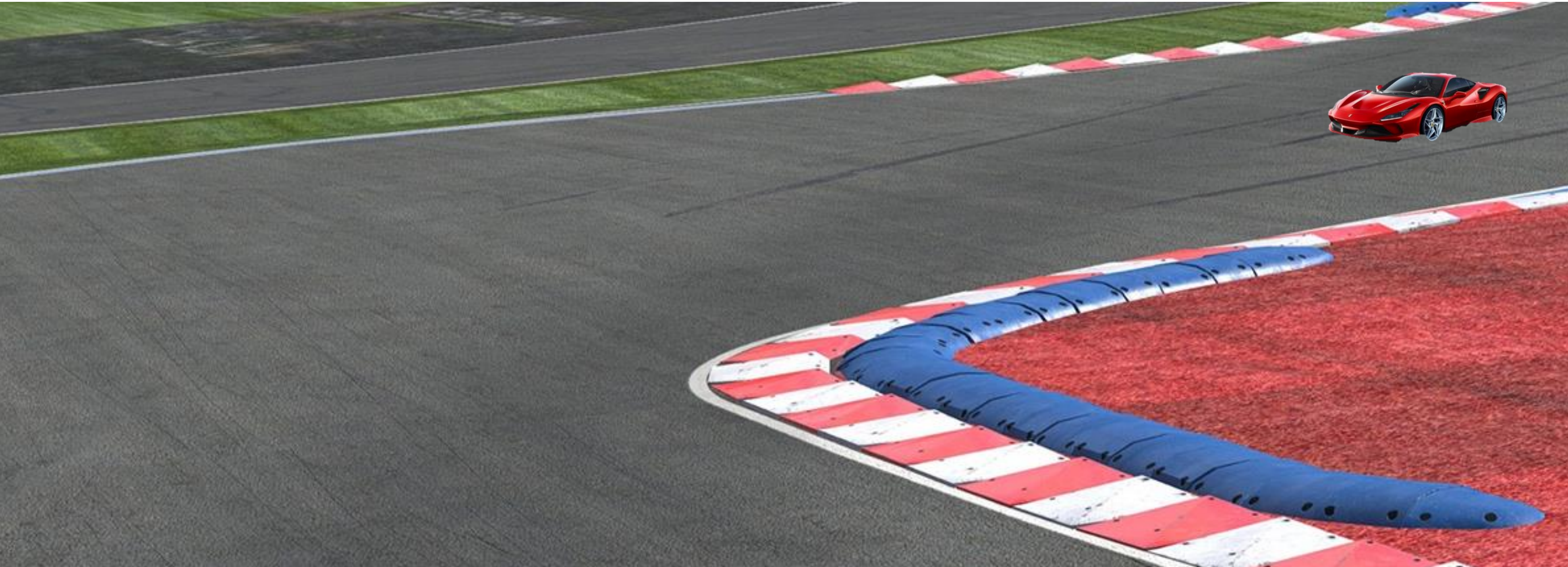
[2] U. Rosolia, F. Borrelli. “Learning model predictive control for iterative tasks: A computationally efficient approach for linear system.” *IFAC-PapersOnLine* (2017)

[3] U. Rosolia, Y. Lian, E. Maddalena, G. Ferrari-Trecate, and C. N. Jones. "On the Optimality and Convergence Properties of the Iterative Learning Model Predictive Controller." IEEE Transactions on Automatic Control (2022).

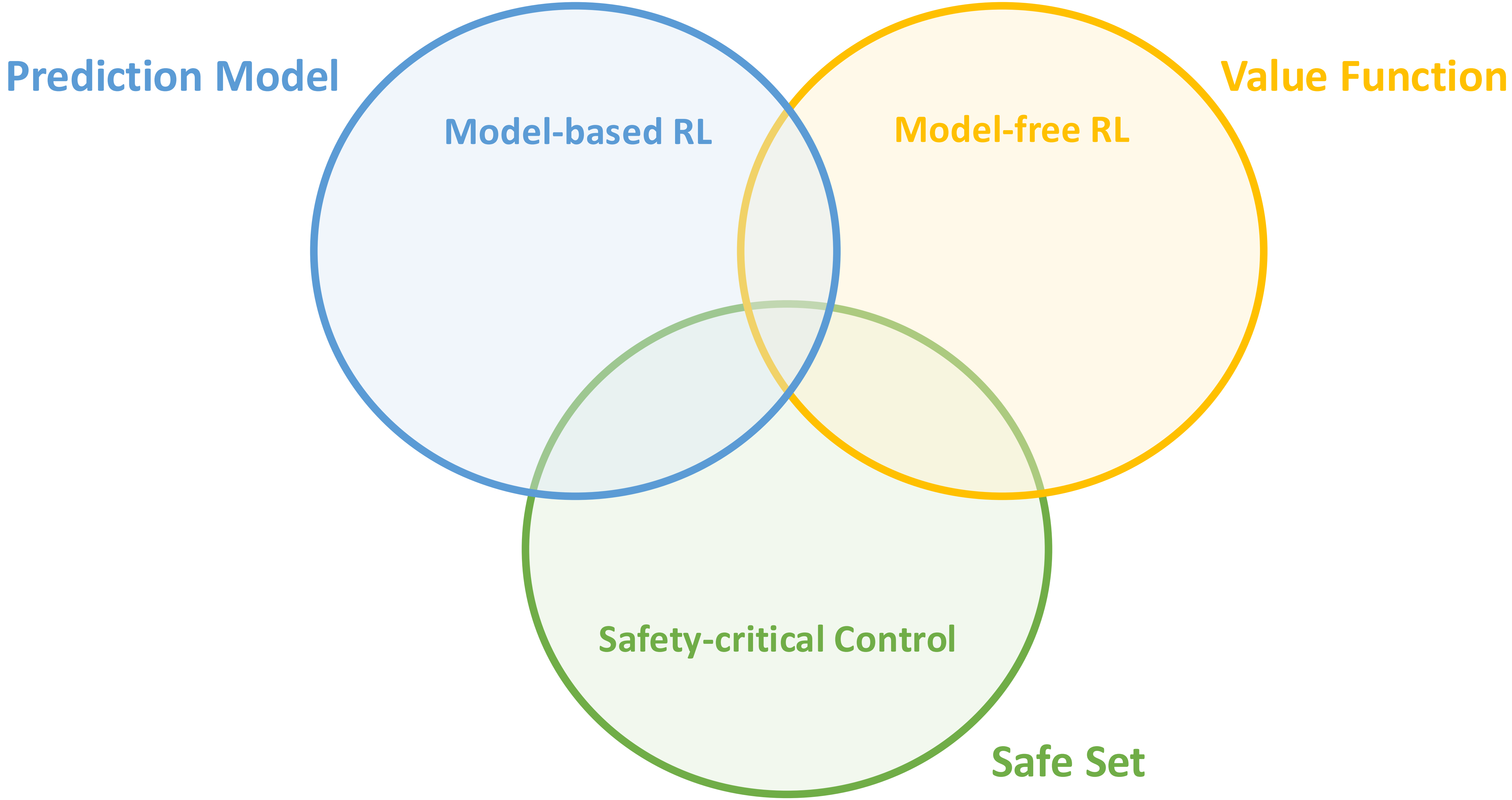
# Autonomous Racing

Goal: Minimize lap time

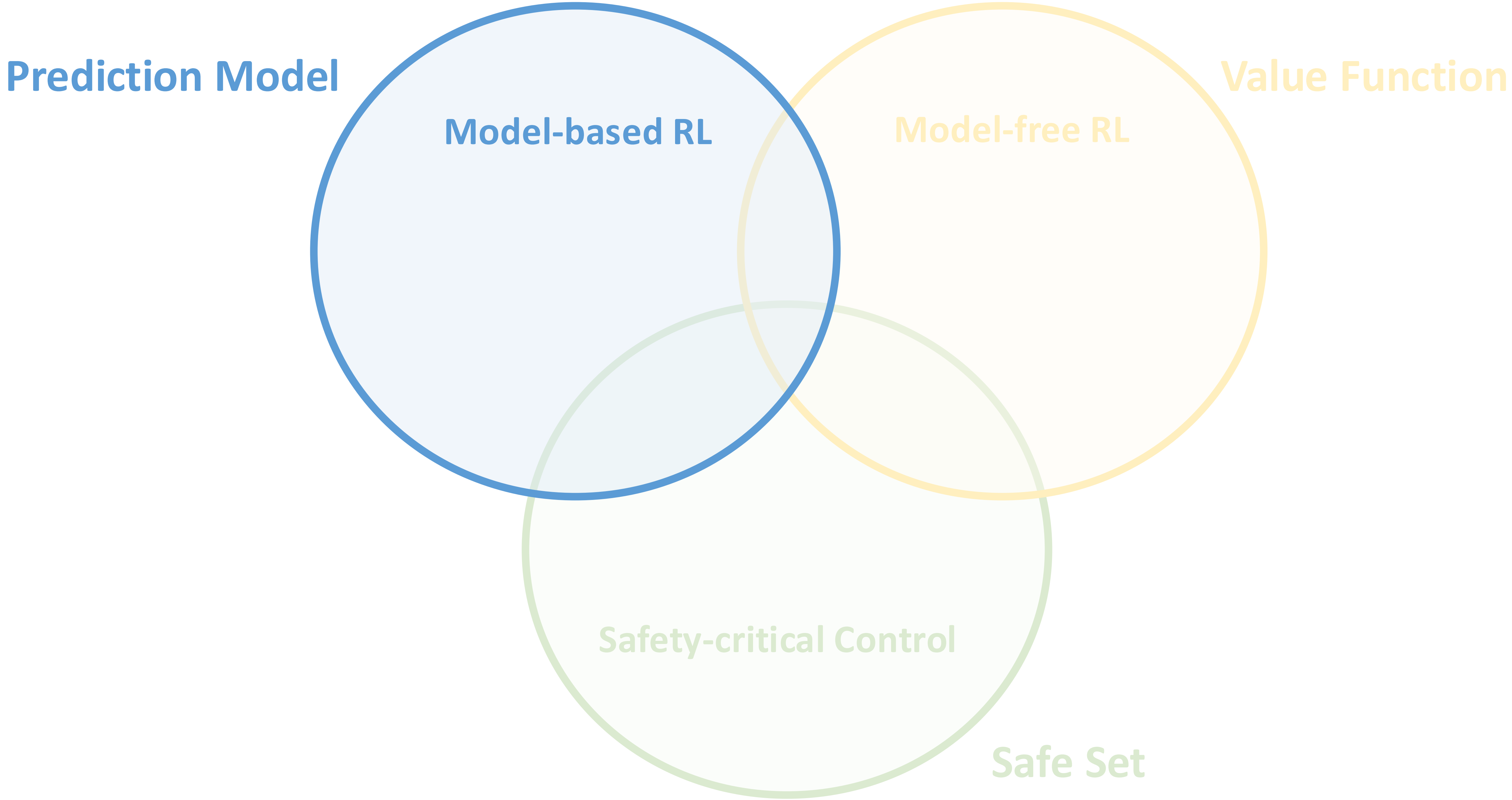
Requirement: Guarantee safety



# Three key components to learn



# Three key components to learn



# Hyundai California Proving Ground

# Hyundai California Proving Ground

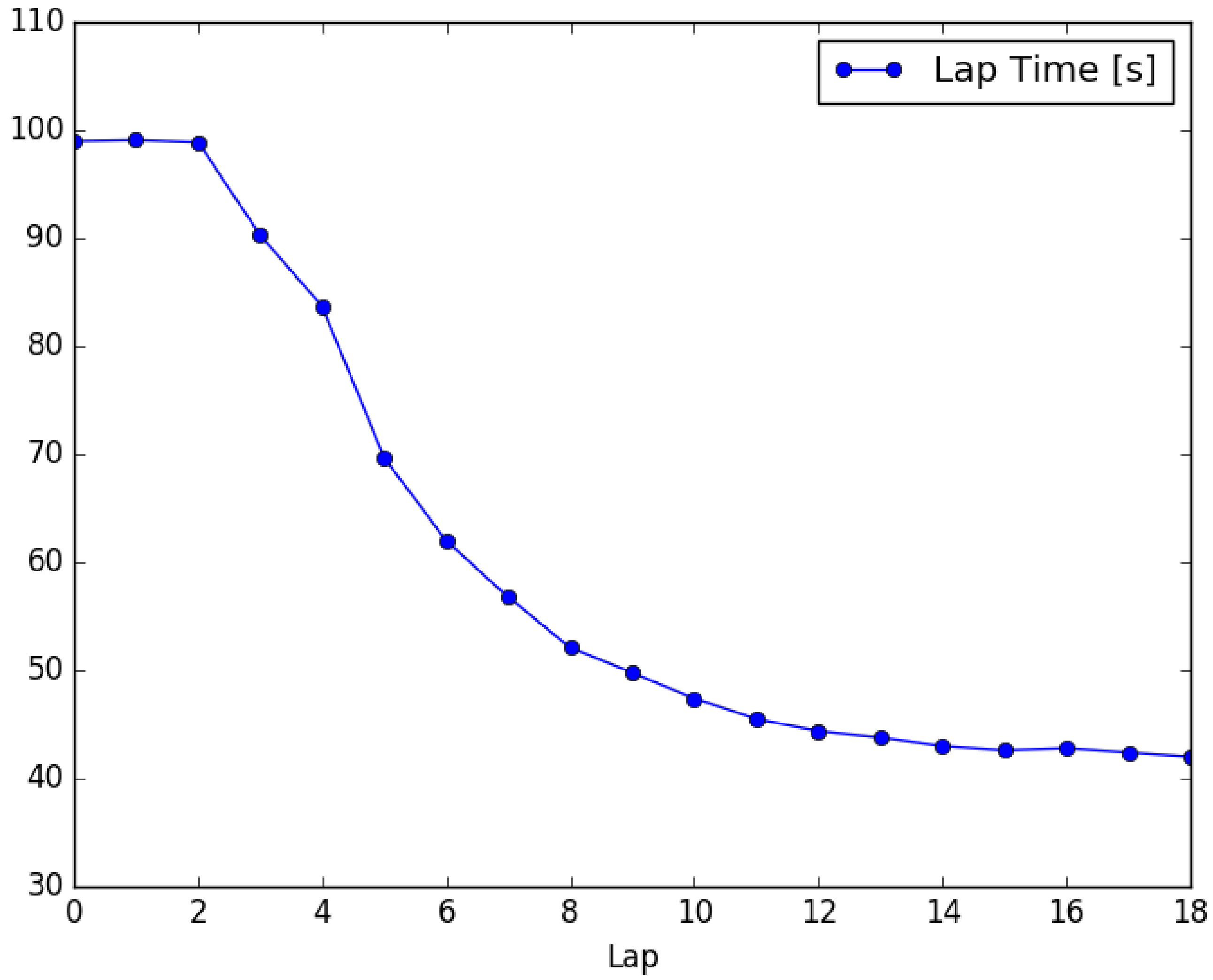
Starting Line

Finish Line

# Learning Model Predictive Controller full-size vehicle experiments

Credits: Siddharth Nair, Nitin Kapania and Ugo Rosolia

# Lap Time



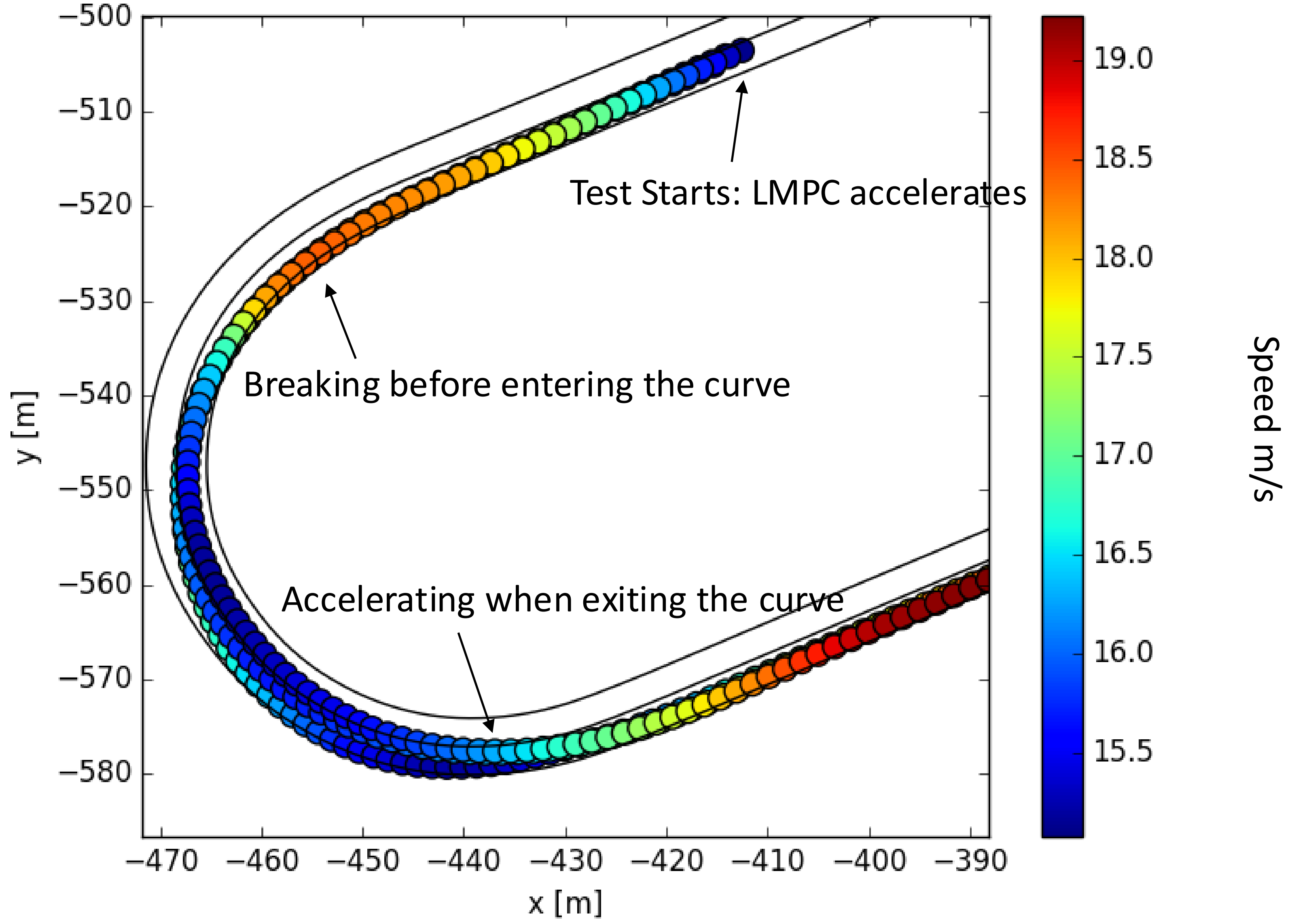

The control policy is constructed using ~1k data points (last 2 laps)

The control action is computed using ~100 data points

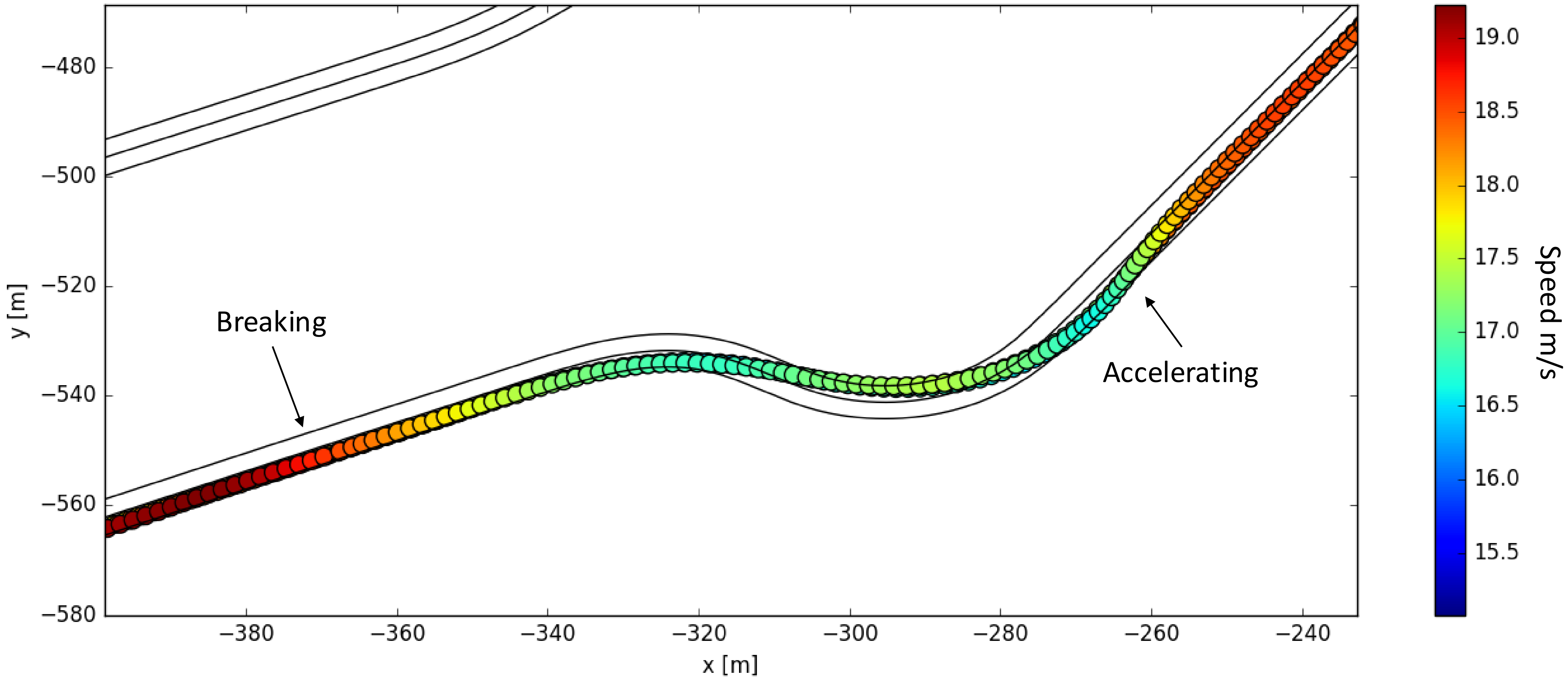
# Learning Model Predictive Controller full-size vehicle experiments

Credits: Siddharth Nair, Nitin Kapania and Ugo Rosolia

# Velocity Profile at Convergence (Curve 1)



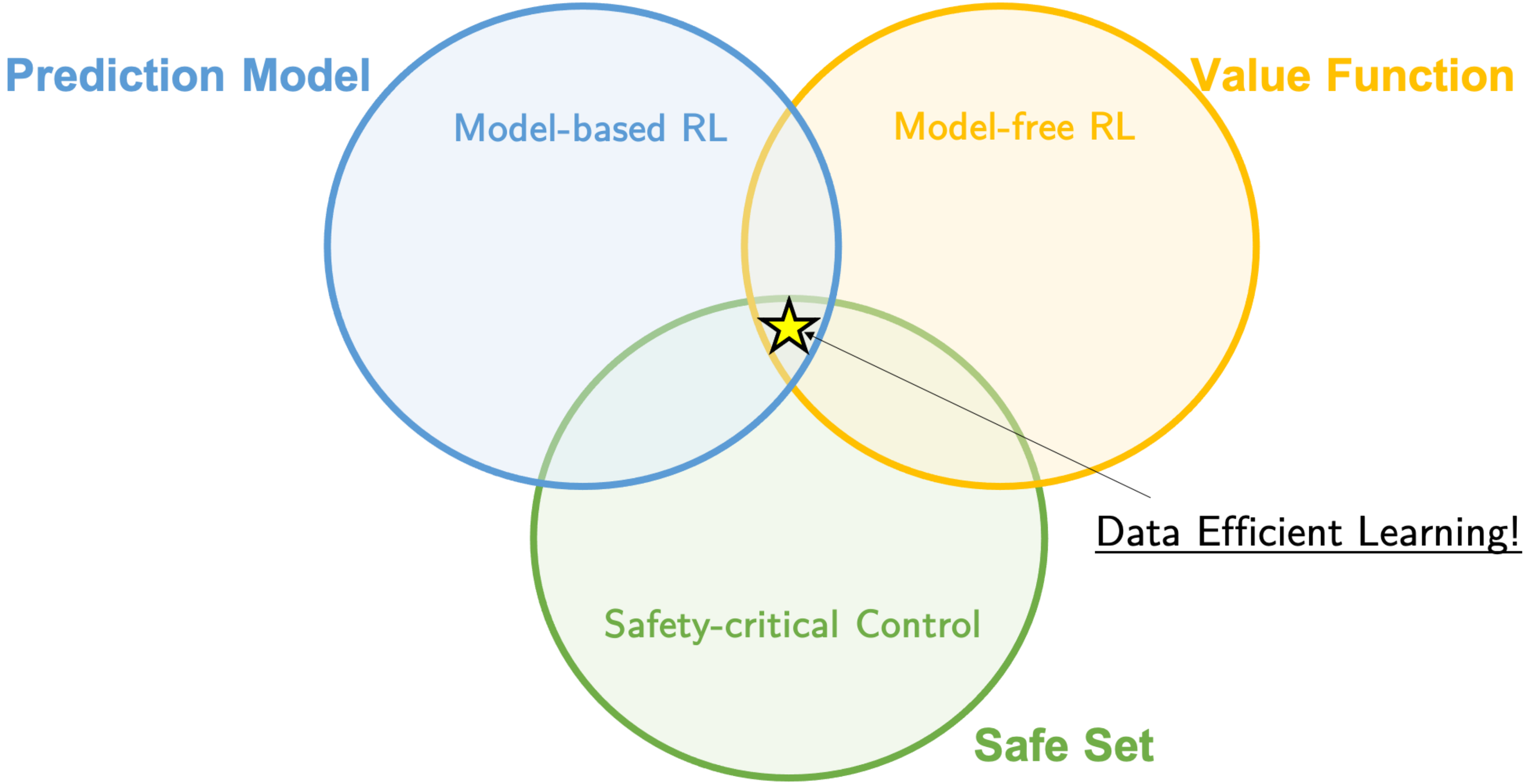
# Velocity Profile at Convergence (Chicane)



# Learning Model Predictictive Control for Autonomous Racing

# The key components

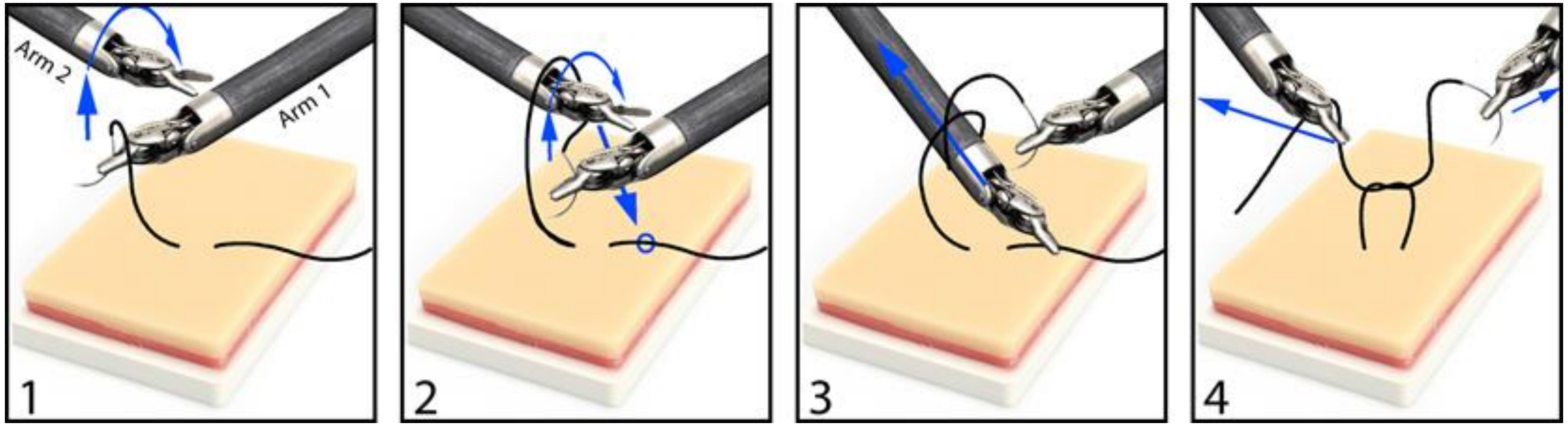
- Predicted trajectory given by **prediction model**
- Predicted cost estimated by **value function**
- Safe region estimated by the **safe set**

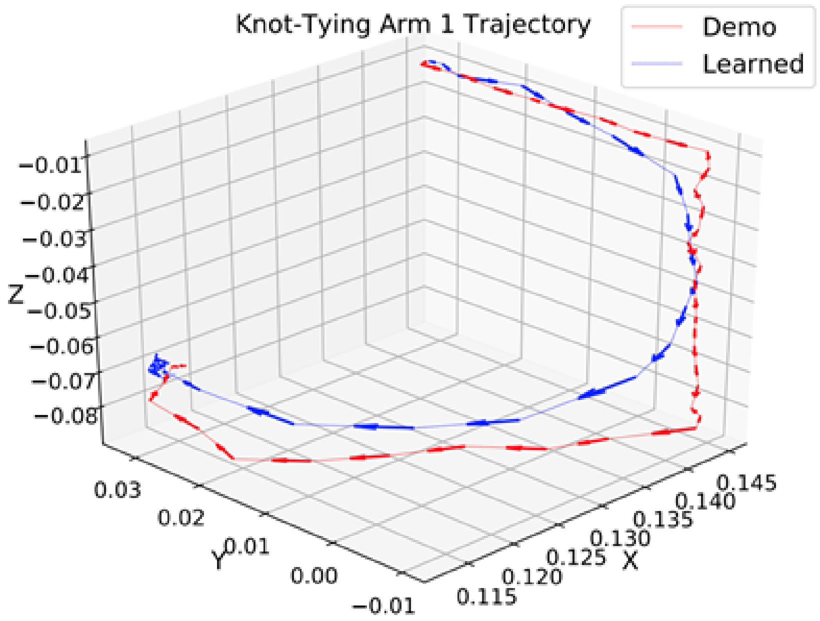


# Terminal Components via DNN


Brijen


Ashwin

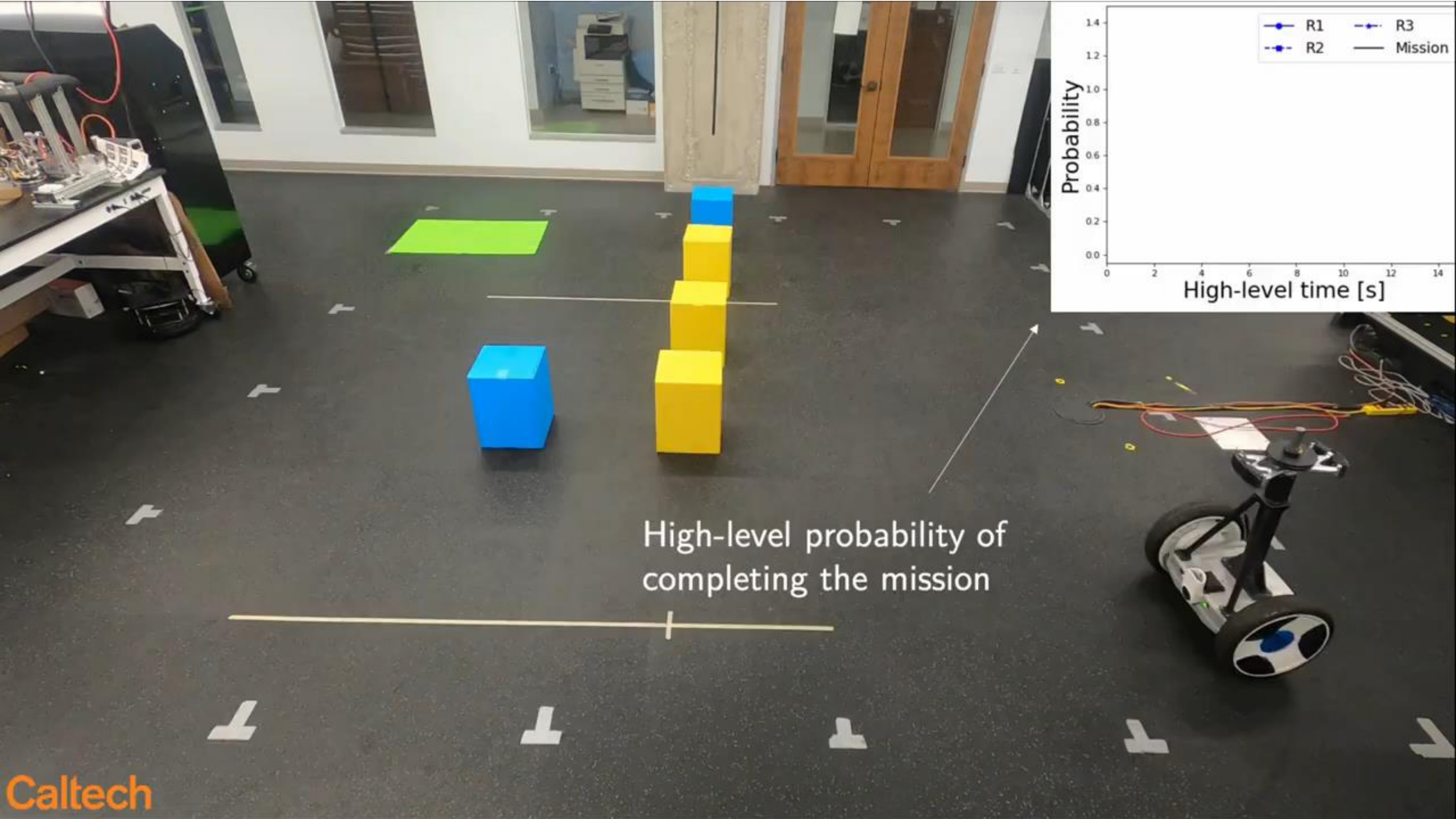



- Safe Set constructed using non-parametric estimation
- Model ensemble and input sampling strategies for MPC
- Knot tying task on real surgical robot with inefficient demos (red)
- Constraints: stay within 1 cm tube of reference trajectory
- SAVED successfully smooths + optimizes demos

"Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks.", B. Thananjeyan*, A. Balakrishna*, U. Rosolia, F. Li, R. McAllister, J. E. Gonzalez, S. Levine, F. Borrelli, K. Goldberg *IEEE Robotics and Automation Letters (RA-L)* (2020)

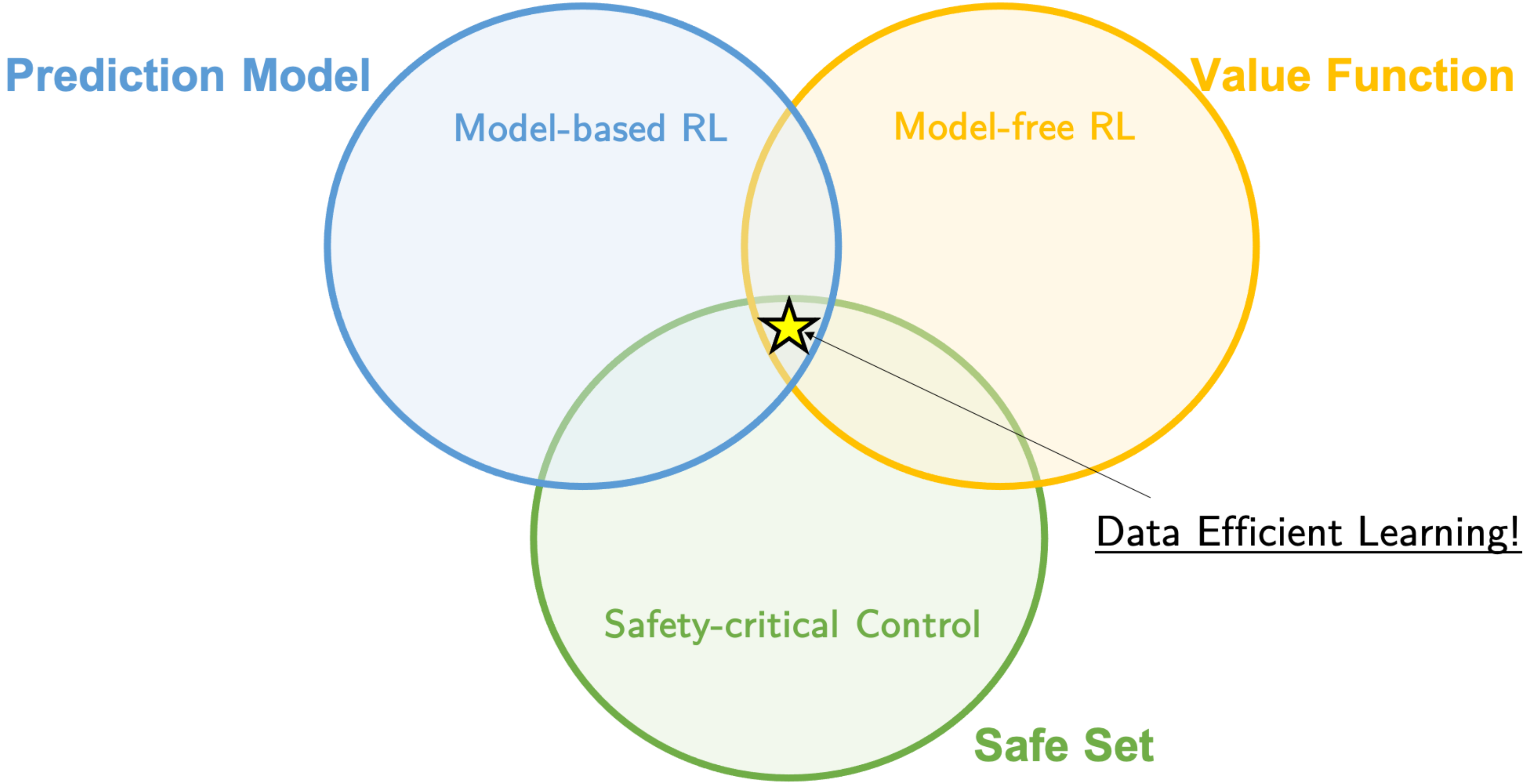
*= equal contribution



R1
R3
R2
Mission
Probability
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
High-level time [s]
High-level probability of completing the mission
Caltech

# The key components

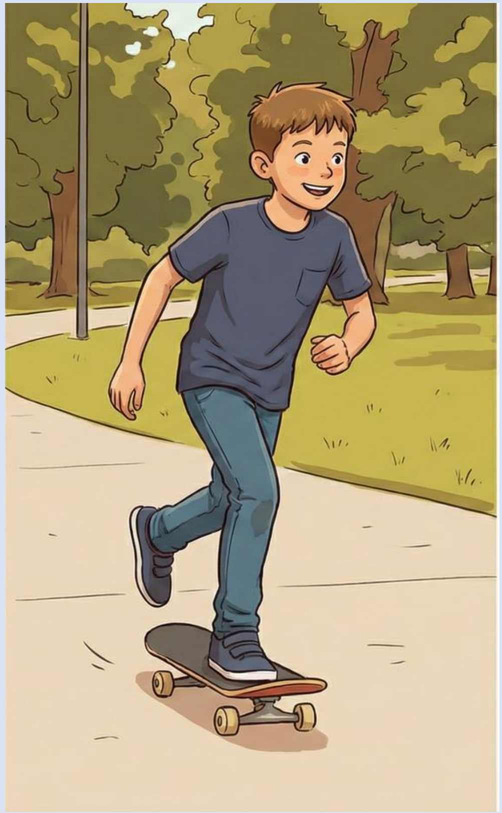
- Predicted trajectory given by **prediction model**
- Predicted cost estimated by **value function**
- Safe region estimated by the **safe set**



# The three phases of learning

# The three phases of learning

Skill improvement



# The three phases of learning

Skill acquisition

Skill improvement

# The three phases of learning

Skill acquisition

Skill improvement

Skill automation

# The three phases of learning

Skill acquisition

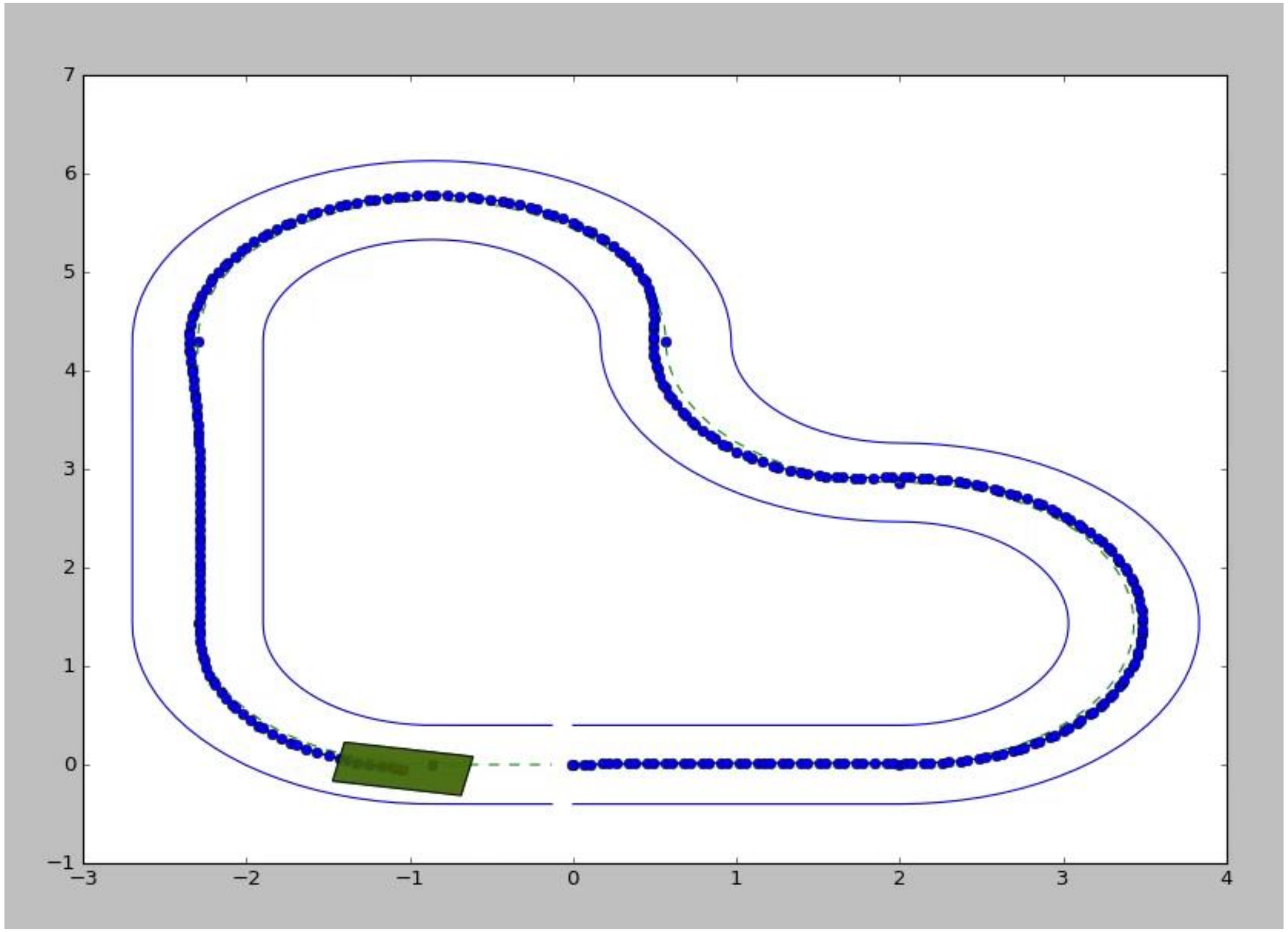
Skill improvement

Skill automation

# **Do you need the safe set?** – Yes

## LMPC without Invariant Set

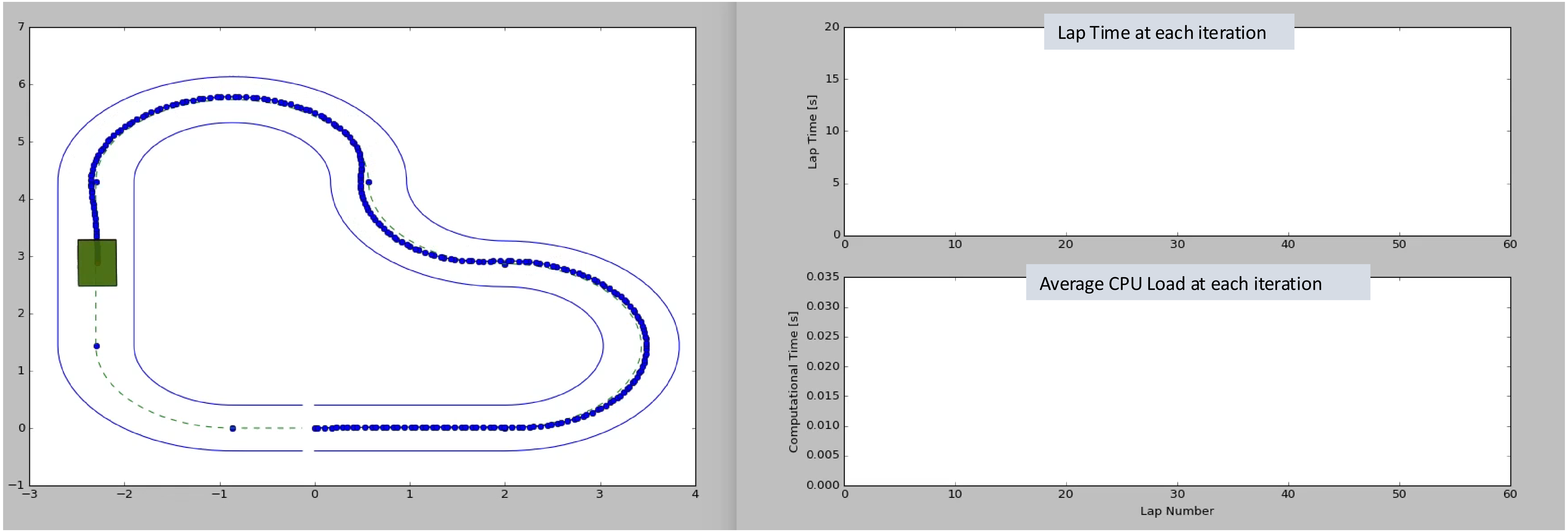
The controller extrapolates the V-function on the Vx dimension



# Do you need to Predict to Learn? Yes

When the LMPC horizon is N = 1 the controller

- solves the Bellman equation using the V-function as value function approximation
- does not explore the state space as it cannot plan outside the safe set



# The three phases of learning

Skill acquisition

Skill improvement

Skill automation

# The three phases of learning

Skill acquisition

Skill improvement

Skill automation

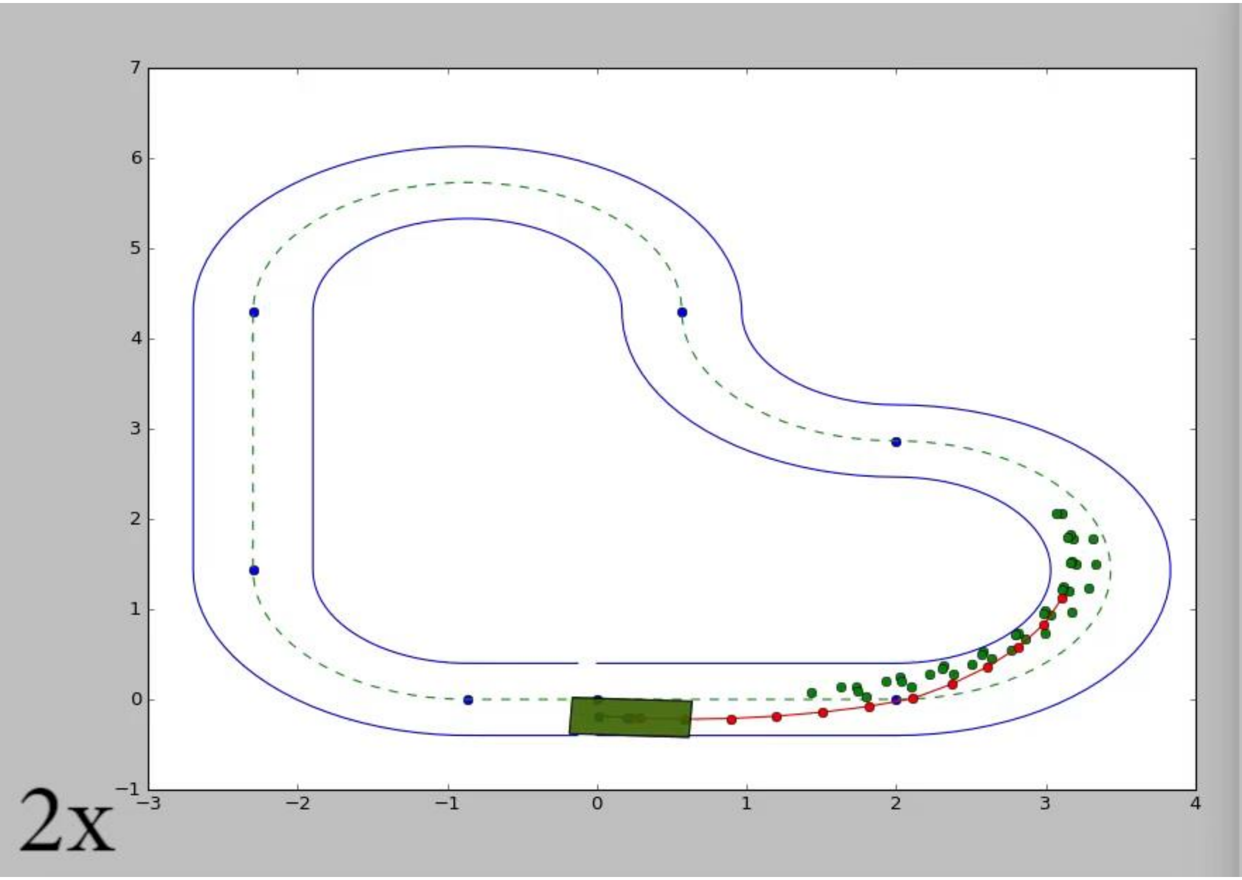
# The three phases of learning

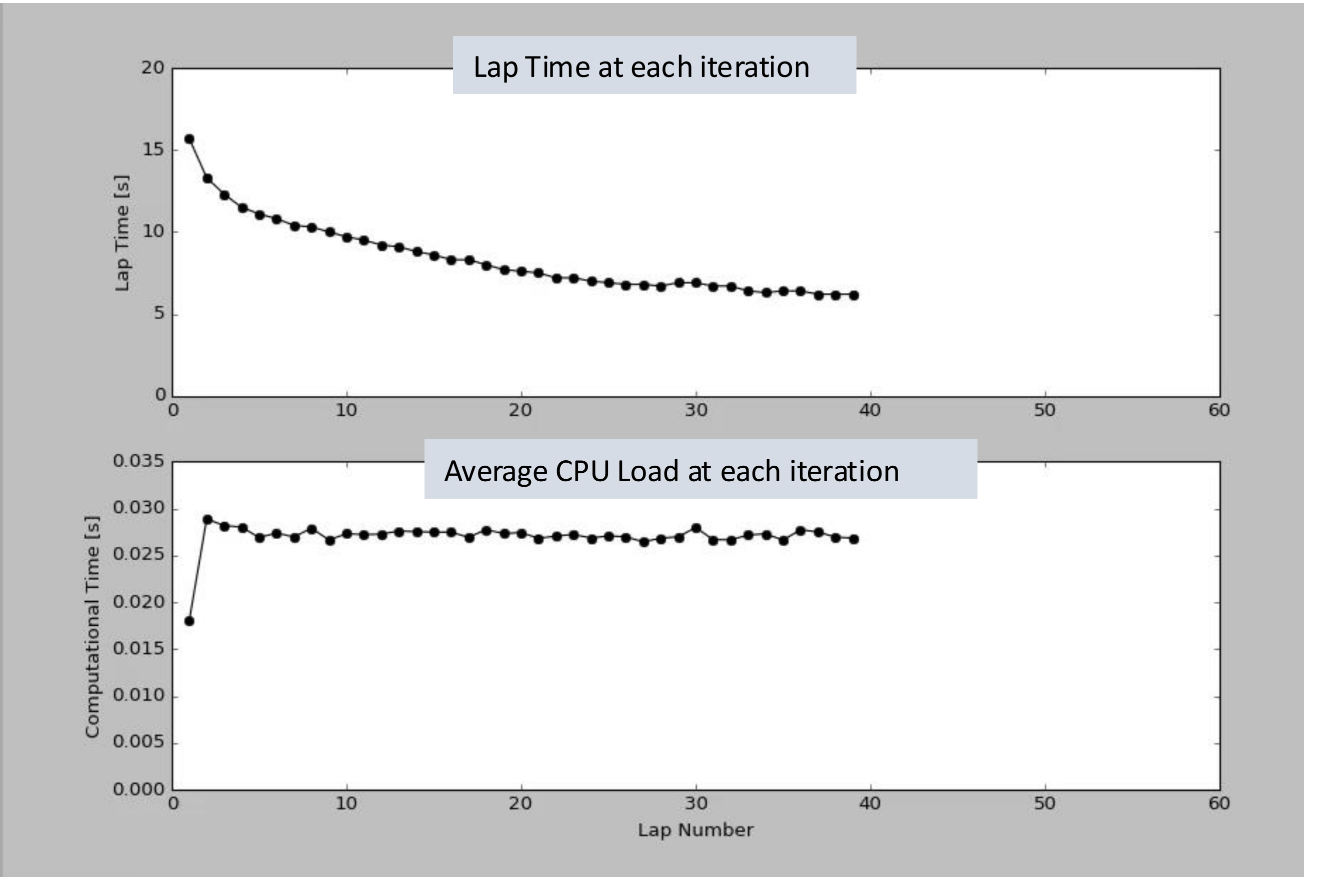
Skill acquisition

Skill improvement

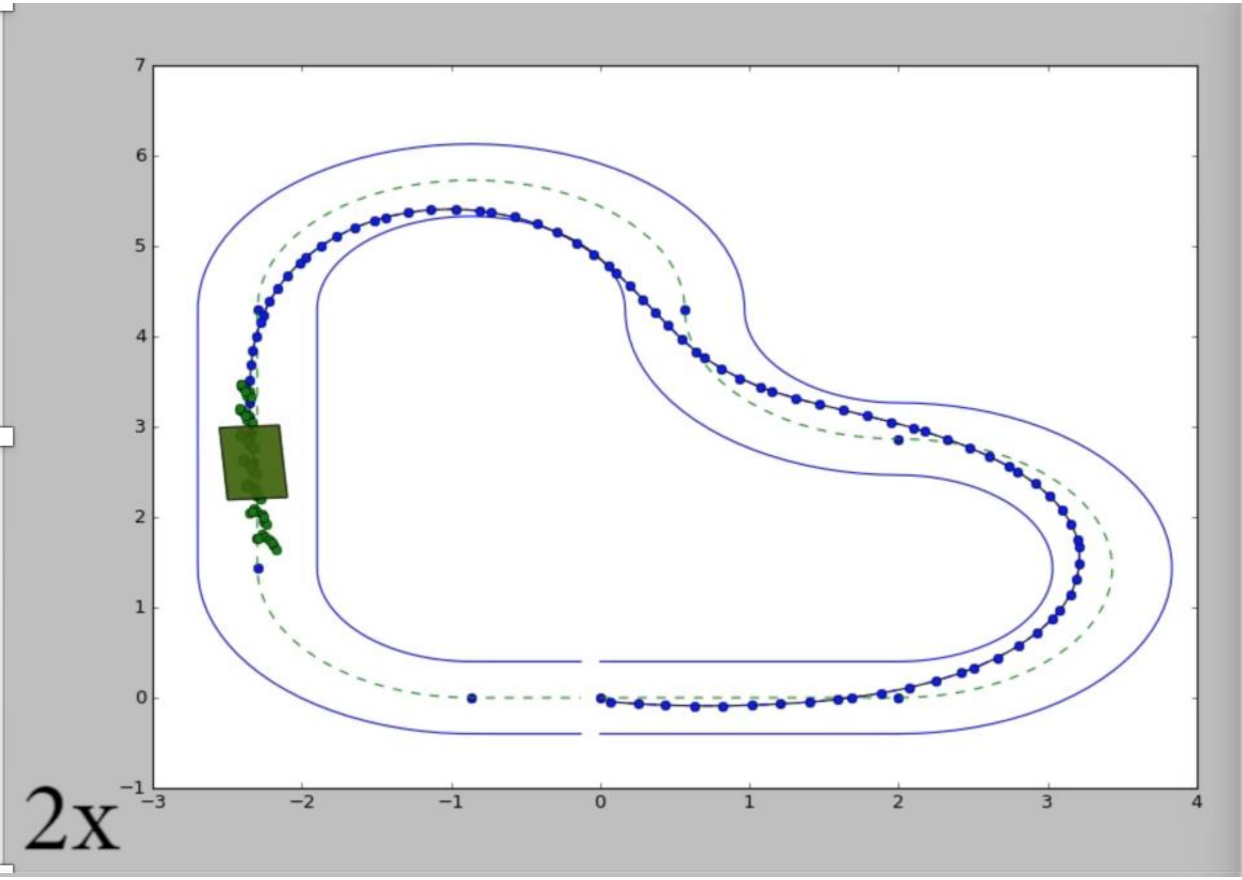
Skill automation

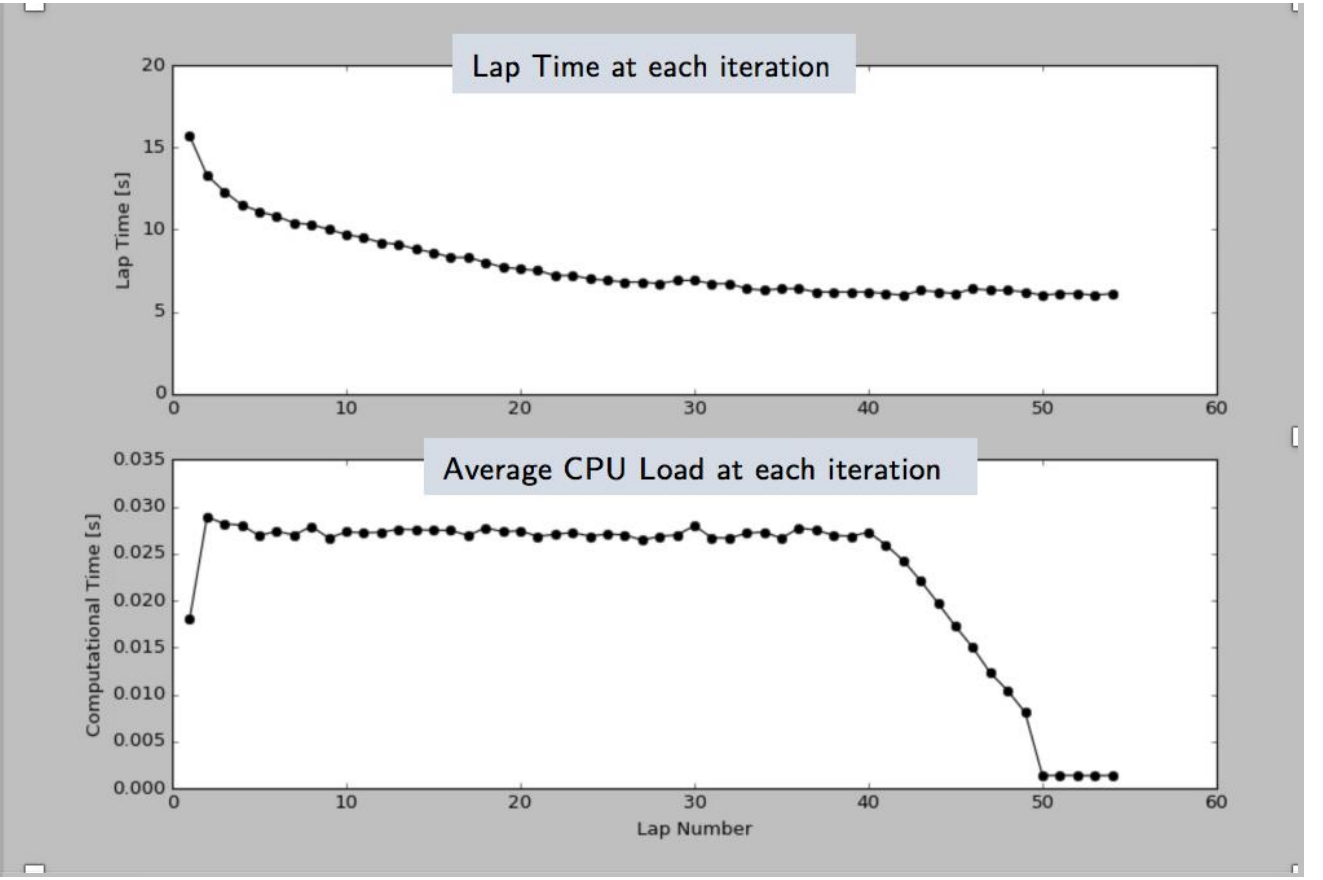
# Do you need to Predict at Convergence? No



# Do you need to Predict at Convergence? No

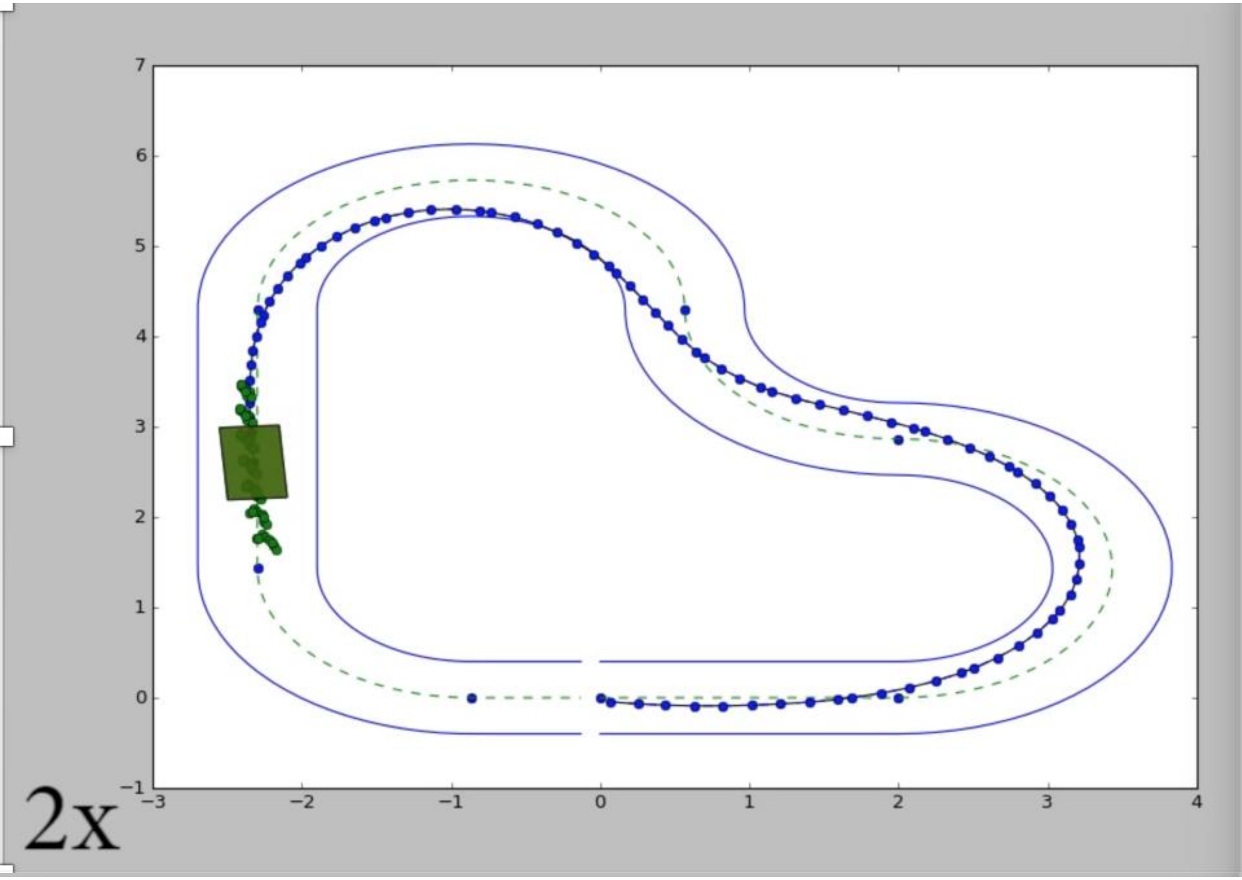


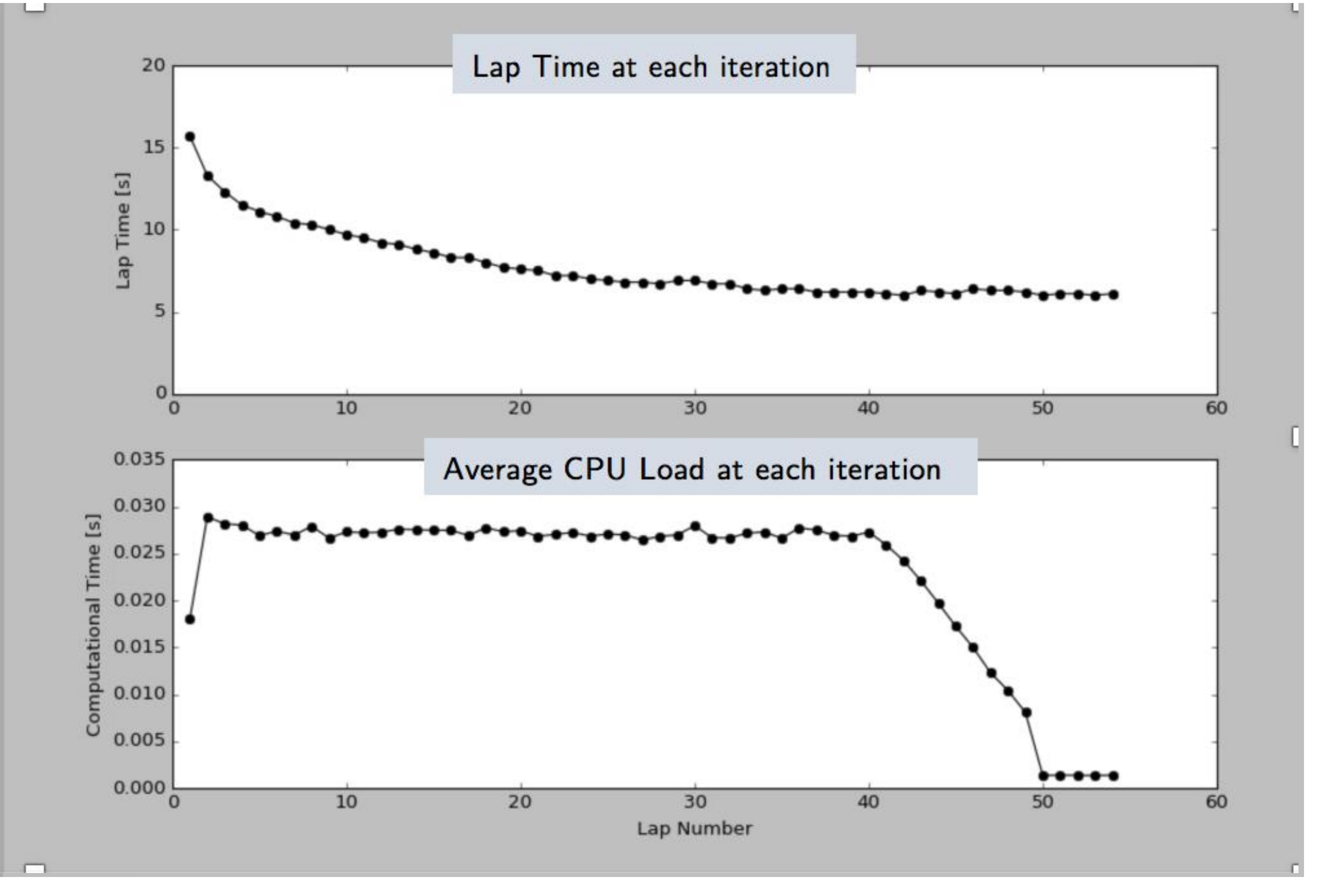




## Value Function Approximation

$$
\begin{aligned}
[\lambda_0^{0,*}, \ldots, \lambda_i^{j,*}] = \arg\min_{\lambda_i^j \in [0,1]} \quad & \sum_i \sum_j J_i^j \lambda_i^j \\
\text{s.t} \quad & \sum_i \sum_j x_i^j \lambda_i^j = x(t), \\
& \sum_i \sum_j \lambda_i^j = 1
\end{aligned}
$$

# Do you need to Predict at Convergence? No



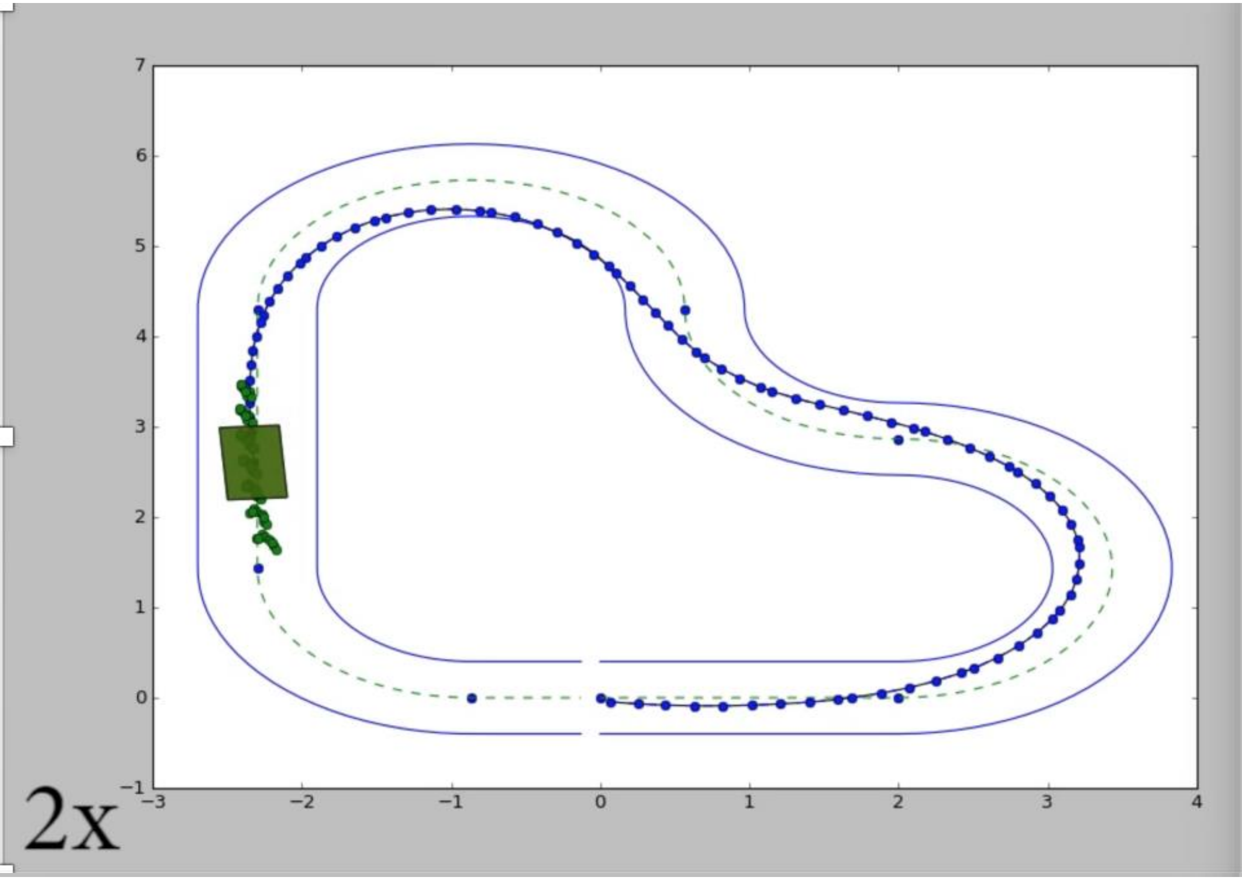

## Value Function Approximation

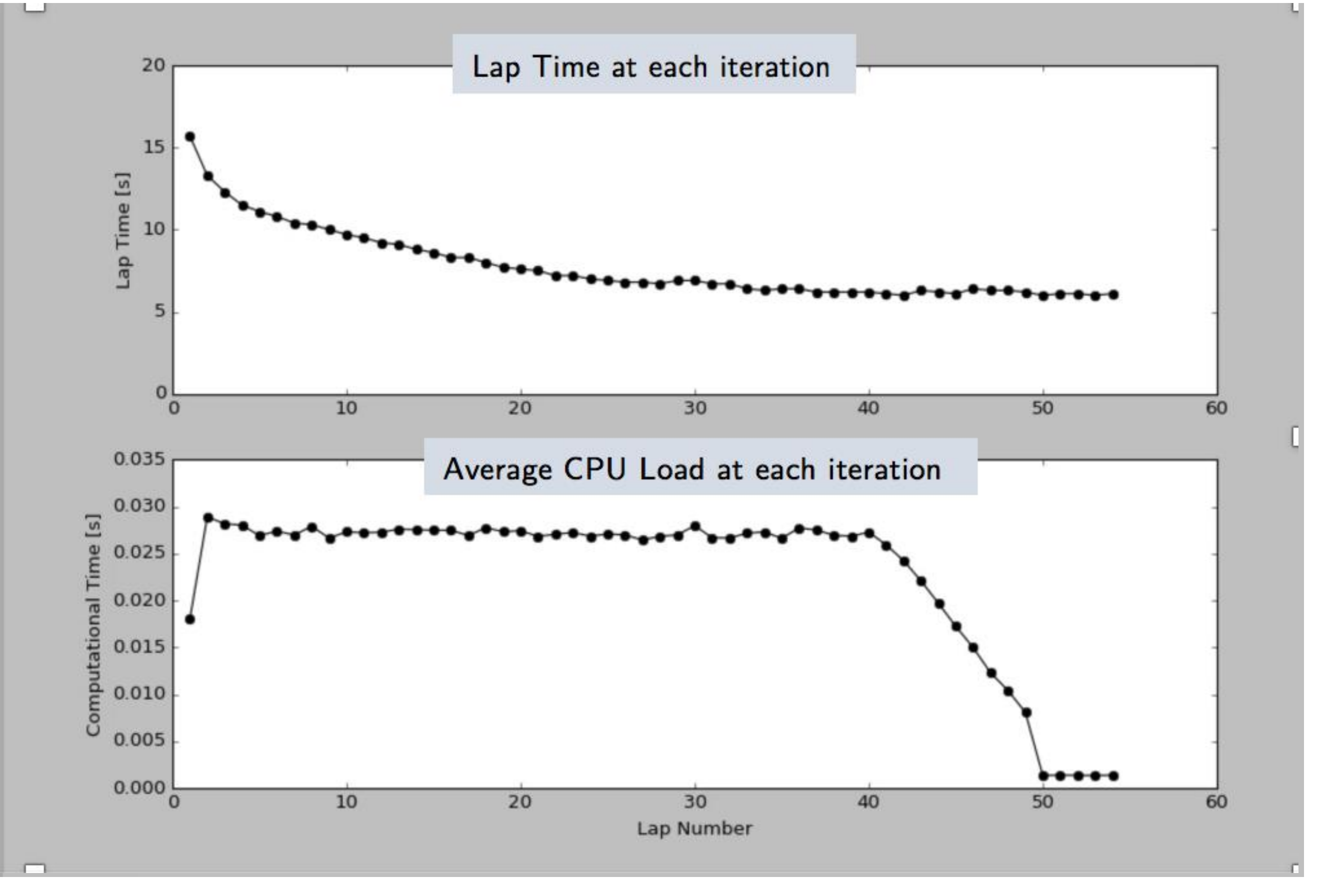
$$
\begin{aligned}
[\lambda_0^{0,*}, \ldots, \lambda_i^{j,*}] = \arg\min_{\lambda_i^j \in [0,1]} \quad & \textstyle\sum_i \sum_j J_i^j \lambda_i^j \\
\text{s.t} \quad & \textstyle\sum_i \sum_j x_i^j \lambda_i^j = x(t), \\
& \textstyle\sum_i \sum_j \lambda_i^j = 1
\end{aligned}
$$

## Control Policy

$$\pi\big(x(t)\big) = \textstyle\sum_i \sum_j u_i^j \lambda_i^{j,*}$$

# Do you need to Predict at Convergence? No




## Value Function Approximation

$$
\begin{aligned}
[\lambda_0^{0,*},\dots,\lambda_i^{j,*}] = \arg\min_{\lambda_i^j\in[0,1]} \quad & \sum_i\sum_j J_i^j\lambda_i^j \\
\text{s.t} \quad & \sum_i\sum_j x_i^j\lambda_i^j = x(t), \\
& \sum_i\sum_j \lambda_i^j = 1
\end{aligned}
$$

## Control Policy

Stored Data

$$\pi\big(x(t)\big) = \sum_i\sum_j u_i^j\lambda_i^{j,*}$$

# The three phases of learning

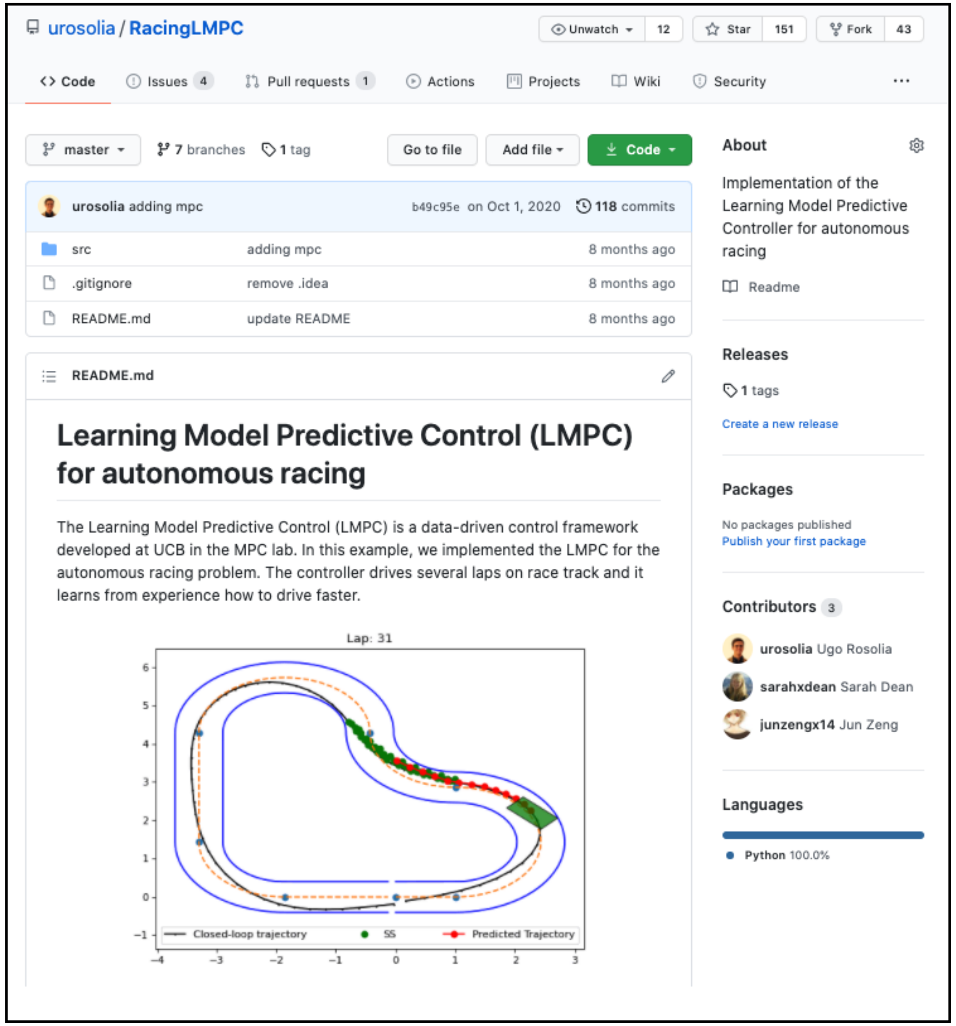
Skill acquisition

Skill improvement

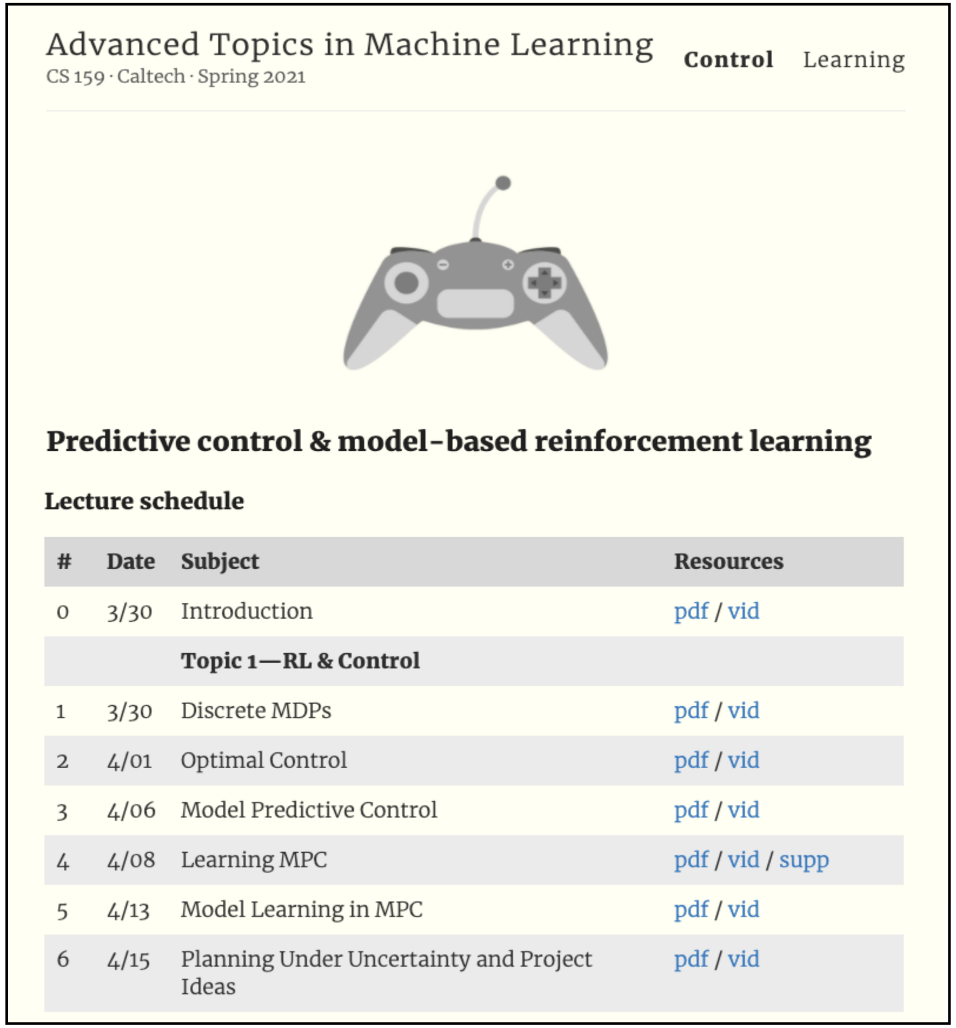
Skill automation

# Thanks! Questions?

## Code available online




## Course material online



Advanced Topics in Machine Learning

CS 159 · Caltech · Spring 2021

**Control** Learning

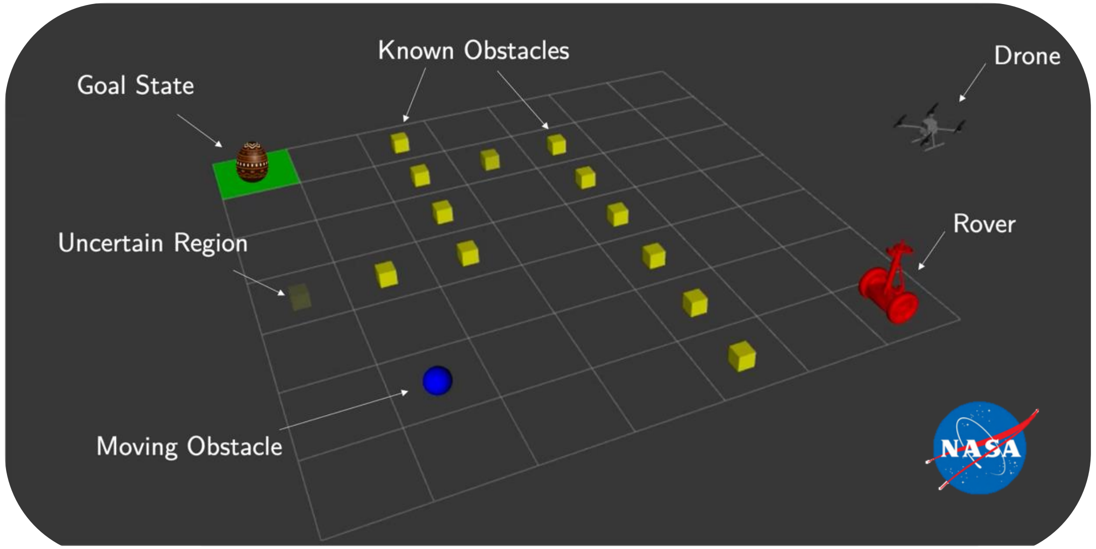
**Predictive control & model-based reinforcement learning**

**Lecture schedule**

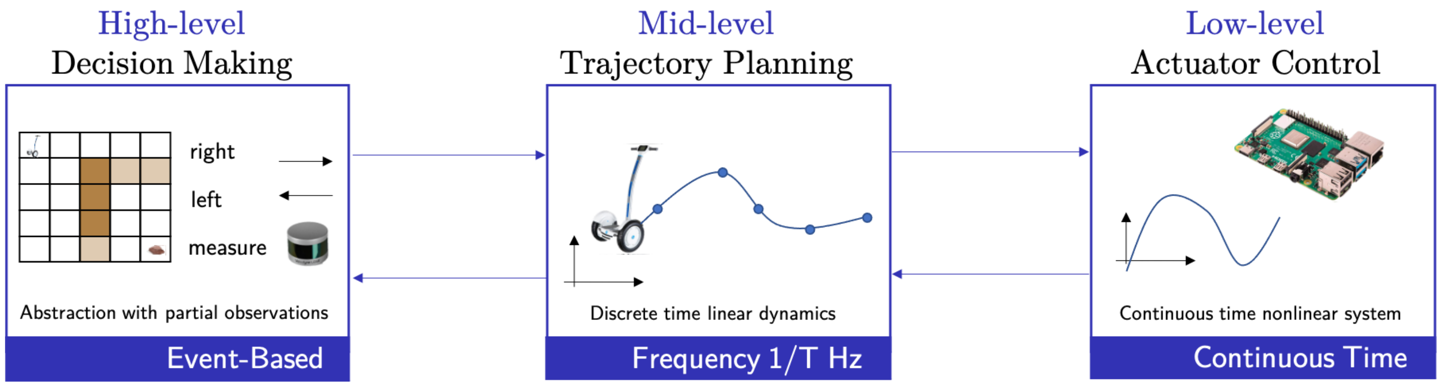
| # | Date | Subject | Resources |
|---|---|---|---|
| 0 | 3/30 | Introduction | pdf / vid |
| | | **Topic 1—RL & Control** | |
| 1 | 3/30 | Discrete MDPs | pdf / vid |
| 2 | 4/01 | Optimal Control | pdf / vid |
| 3 | 4/06 | Model Predictive Control | pdf / vid |
| 4 | 4/08 | Learning MPC | pdf / vid / supp |
| 5 | 4/13 | Model Learning in MPC | pdf / vid |
| 6 | 4/15 | Planning Under Uncertainty and Project Ideas | pdf / vid |

# What is next?

- Partial Observability

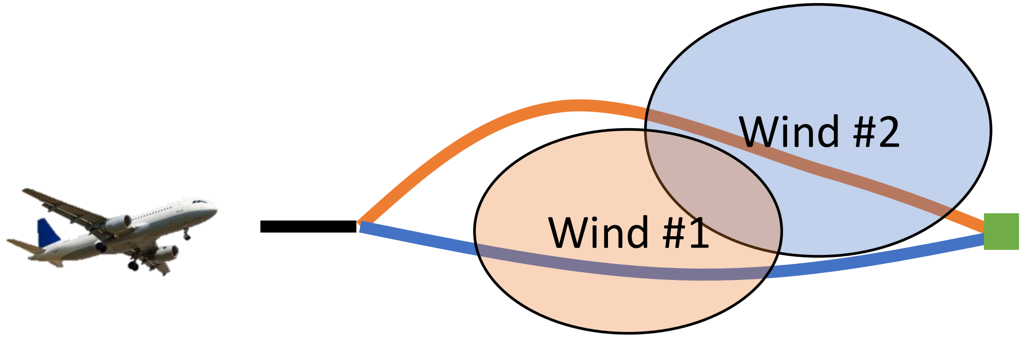



- Multi-agent systems




- Hierarchy + Learning

- Optimize over strategies, not trajectories



# System ID in Autonomous Racing

- Nonlinear Dynamical System,

$$\begin{aligned}
\ddot{x} &= \dot{y}\dot{\psi} + \frac{1}{m}\sum_i F_{x_i} \\
\ddot{y} &= -\dot{x}\dot{\psi} + \frac{1}{m}\sum_i F_{y_i} \\
\ddot{\psi} &= \frac{1}{I_z}(a(F_{y_{1,2}}) - b(F_{y_{2,3}}) + c(-F_{x_{1,3}} + F_{x_{2,4}}) \\
\dot{X} &= \dot{x}\cos\psi - \dot{y}\sin\psi, \quad \dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi
\end{aligned}$$

# System ID in Autonomous Racing

- Nonlinear Dynamical System,

$$\begin{aligned}
\ddot{x} &= \dot{y}\dot{\psi} + \frac{1}{m}\sum_i F_{x_i} \\
\ddot{y} &= -\dot{x}\dot{\psi} + \frac{1}{m}\sum_i F_{y_i} \\
\ddot{\psi} &= \frac{1}{I_z}(a(F_{y_{1,2}}) - b(F_{y_{2,3}}) + c(-F_{x_{1,3}} + F_{x_{2,4}}) \\
\boxed{\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi, \quad \dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi}
\end{aligned}$$

Kinematic Equations

# System ID in Autonomous Racing

- Nonlinear Dynamical System,

$$\begin{aligned}
\ddot{x} &= \dot{y}\dot{\psi} + \frac{1}{m}\sum_i F_{x_i} \\
\ddot{y} &= -\dot{x}\dot{\psi} + \frac{1}{m}\sum_i F_{y_i} \\
\ddot{\psi} &= \frac{1}{I_z}(a(F_{y_{1,2}}) - b(F_{y_{2,3}}) + c(-F_{x_{1,3}} + F_{x_{2,4}}) \\
\dot{X} &= \dot{x}\cos\psi - \dot{y}\sin\psi, \quad \dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi
\end{aligned}$$

Kinematic Equations

- Identifying the Dynamical System

$$x^j_{k+1|t} = \begin{bmatrix} \dot{x}_{k+1|t} \\ \dot{y}_{k+1|t} \\ \ddot{\psi} \\ \psi_{k+1|t} \\ X_{k+1|t} \\ Y_{k+1|t} \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \end{bmatrix} x^j_{k|t} + \begin{bmatrix} \\ \\ \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \end{bmatrix} \begin{bmatrix} u^j_{k|t} \\ 1 \end{bmatrix}$$

Linearization around predicted trajectory

# System ID in Autonomous Racing

- Nonlinear Dynamical System,

$$
\begin{aligned}
\ddot{x} &= \dot{y}\dot{\psi} + \frac{1}{m}\sum_i F_{x_i} \\
\ddot{y} &= -\dot{x}\dot{\psi} + \frac{1}{m}\sum_i F_{y_i} \\
\ddot{\psi} &= \frac{1}{I_z}(a(F_{y_{1,2}}) - b(F_{y_{2,3}}) + c(-F_{x_{1,3}} + F_{x_{2,4}}))
\end{aligned}
$$

Dynamic Equations

$$
\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi, \quad \dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi
$$

Kinematic Equations

- Identifying the Dynamical System

$$
x^j_{k+1|t} = \begin{bmatrix} \dot{x}_{k+1|t} \\ \dot{y}_{k+1|t} \\ \ddot{\psi} \\ \psi_{k+1|t} \\ X_{k+1|t} \\ Y_{k+1|t} \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \end{bmatrix} x^j_{k|t} + \begin{bmatrix} \\ \\ \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \end{bmatrix} \begin{bmatrix} u^j_{k|t} \\ 1 \end{bmatrix}
$$

Linearization around predicted trajectory

# System ID in Autonomous Racing

- Nonlinear Dynamical System,

$$\begin{aligned}
\ddot{x} &= \dot{y}\dot{\psi} + \frac{1}{m}\textstyle\sum_i F_{x_i} \\
\ddot{y} &= -\dot{x}\dot{\psi} + \frac{1}{m}\textstyle\sum_i F_{y_i} \\
\ddot{\psi} &= \frac{1}{I_z}(a(F_{y_{1,2}}) - b(F_{y_{2,3}}) + c(-F_{x_{1,3}} + F_{x_{2,4}})
\end{aligned}$$

Dynamic Equations

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi, \quad \dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi$$

Kinematic Equations

- Identifying the Dynamical System

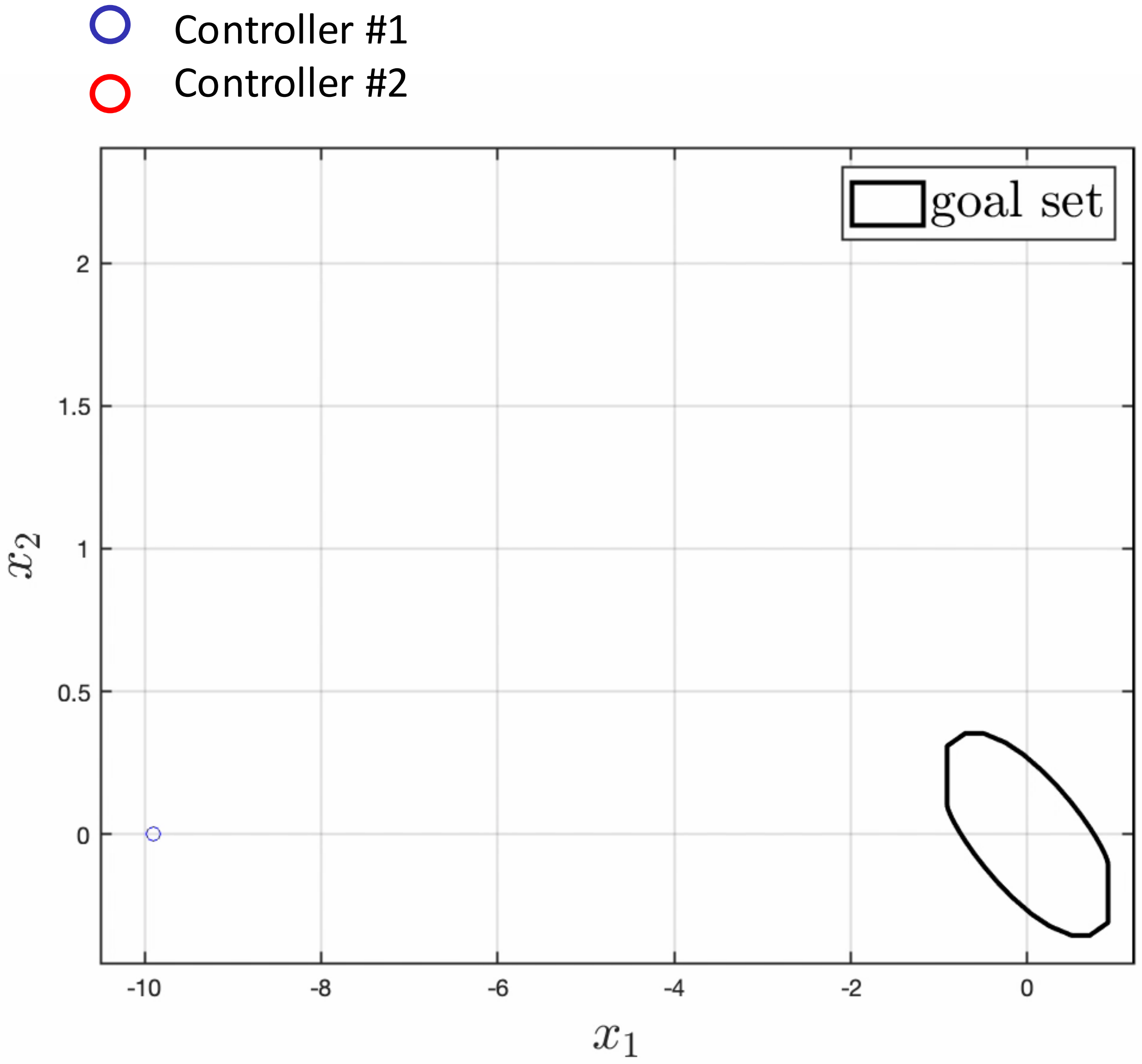
Local Linear Regression

$$x^j_{k+1|t} = \begin{bmatrix} \dot{x}_{k+1|t} \\ \dot{y}_{k+1|t} \\ \ddot{\psi} \\ \psi_{k+1|t} \\ X_{k+1|t} \\ Y_{k+1|t} \end{bmatrix} = \left[ \begin{array}{c} \boxed{\operatorname*{argmin}_{\Lambda_y} \sum_{i,s} K(x^j_{k|t} - x^i_s)||\Lambda_y \begin{bmatrix} x^i_s \\ u^i_s \\ 1 \end{bmatrix} - y^i_{s+1}||, \forall y \in \{\dot{x}, \dot{y}, \ddot{\psi}\}} \\ \boxed{\begin{array}{c} \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \end{array}} \; x^j_{k|t} + \boxed{\begin{array}{c} \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \\ \text{Linearized Kinematics} \end{array}} \end{array} \right] \begin{bmatrix} u^j_{k|t} \\ 1 \end{bmatrix}$$

Linearization around predicted trajectory

# Safe Sets and Value Functions Estimation via Sampling

# Safe Sets and Value Functions Estimation via Sampling




- Collect several trajectories with the same controller
- **Safe sets** computed as before using multiple trajectories
- **Value functions** estimate either the mean or worst-case cost
- All statement hold with some probability that is proportional to the amount of data


U. Rosolia, and F. Borrelli. "Sample-based learning model predictive control for linear uncertain systems." In 2019 IEEE 58th Conference on Decision and Control (CDC). IEEE, 2019.
U. Rosolia, X. Zhang, and F. Borrelli. "Robust learning model predictive control for linear systems performing iterative tasks." IEEE Transactions on Automatic Control (2021).

# Comparison with Approximate DP (aka RL)

- Some references:
  - Bertsekas paper connecting MPC and ADP [1], books on RL and OC [2,3]
  - Lewis and Vrabie survey [4]
  - Recht survey [5]

- Learning MPC highlights
  - Continuous state and action formulation
  - Constraints satisfaction
  - V-function constructed locally based on cost/model driven exploration
  - V-function at stored state is “exact” and upperbounds at intermediate points


[1] D. Bertsekas, “Dynamic programming and suboptimal control: A survey from ADP to MPC.” European Journal of Control 11.4-5 (2005)
[2] D. Bertsekas, “Reinforcement learning and optimal control.” Athena Scientific, 2019.
[3] D. Bertsekas, “Distributed Reinforcement Learning” http://web.mit.edu/dimitrib/www/RL_2_Rollout_&_PI.pdf
[4] F. Lewis, Frank, and D. Vrabie. "Reinforcement learning and adaptive dynamic programming for feedback control." IEEE circuits and systems magazine 9.3 (2009)
[5] R. Benjamin. "A tour of reinforcement learning: The view from continuous control." Annual Review of Control, Robotics, and Autonomous Systems 2 (2019)