Feasible and Stressful Trajectory Generation for Mobile Robots

Carl Hildebrandt, Sebastian Elbaum, Nicola Bezzo, Matthew B. Dwyer

Contact: hildebrandt.carl@virginia.edu

This work was funded in part by the NSF and U.S. Army Research Office Grant

Motivation

Mobile robots are **becoming more pervasive** in society

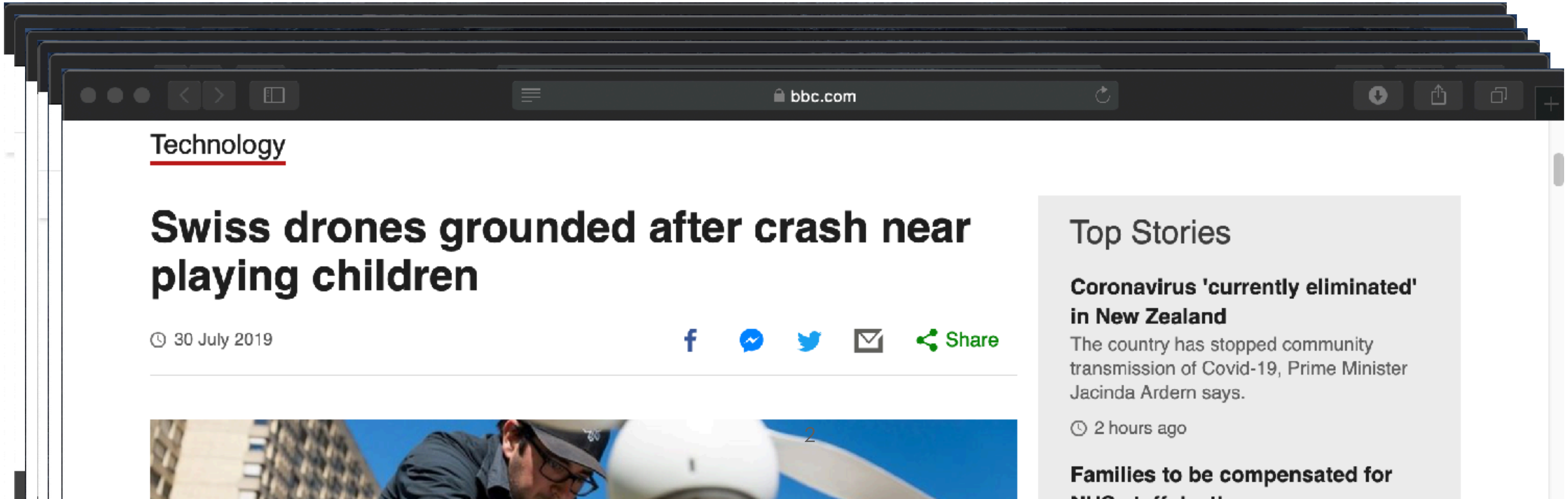
Autonomous Cars

Autonomous Drones

Autonomous Water Vehicles

Autonomous Space Vehicles

This has raised awareness of the **potential impact of faults** in such systems


bbc.com

Technology

Swiss drones grounded after crash near playing children

30 July 2019

Share

Top Stories

Coronavirus 'currently eliminated' in New Zealand

The country has stopped community transmission of Covid-19, Prime Minister Jacinda Ardern says.

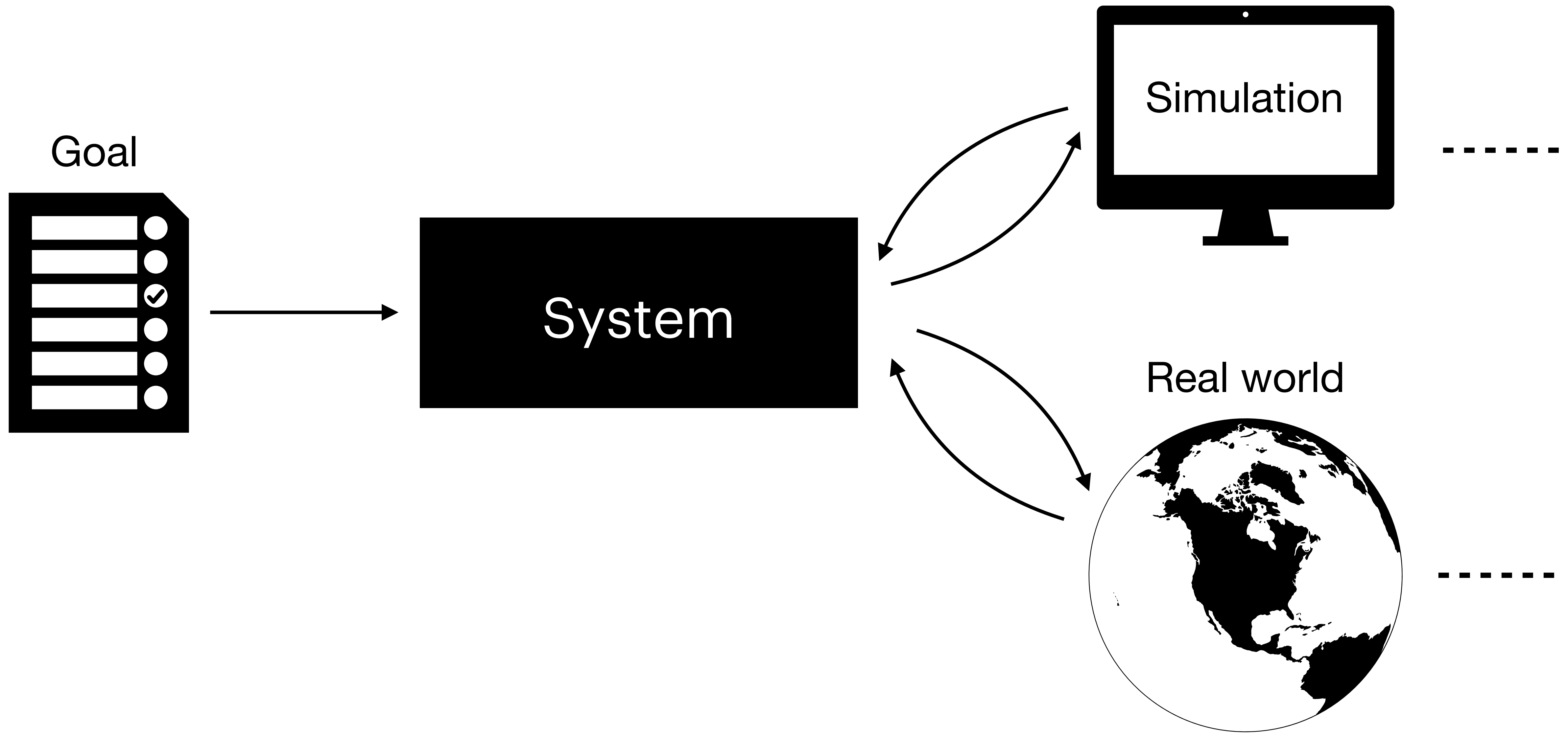
2 hours ago

Families to be compensated for

Motivation

Fully testing these systems is becoming incredibly important

End-to-end system testing



How are these goal trajectories generated?

Motivation

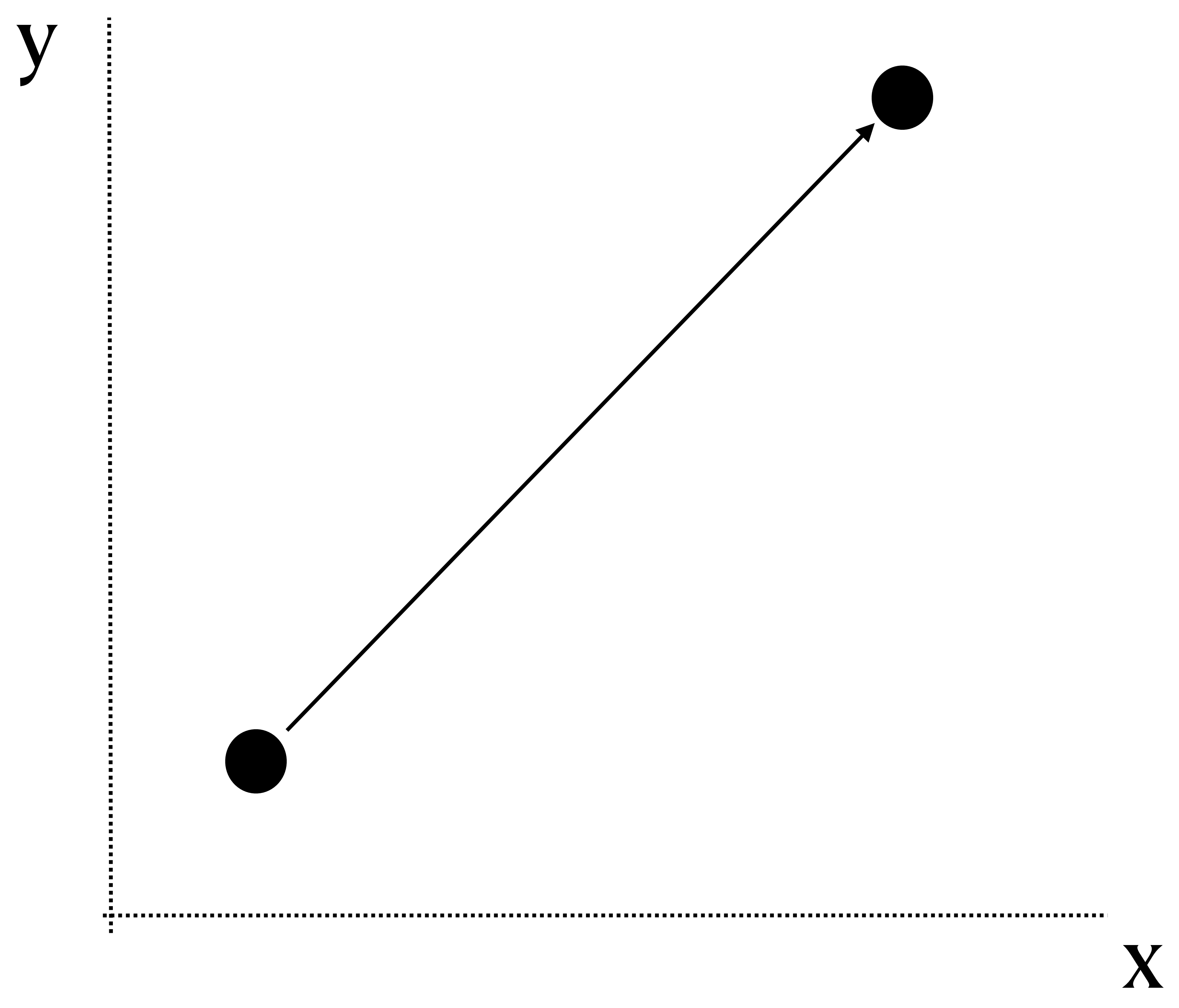
UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

For example, say you as a tester where given this empty space to test your vehicle

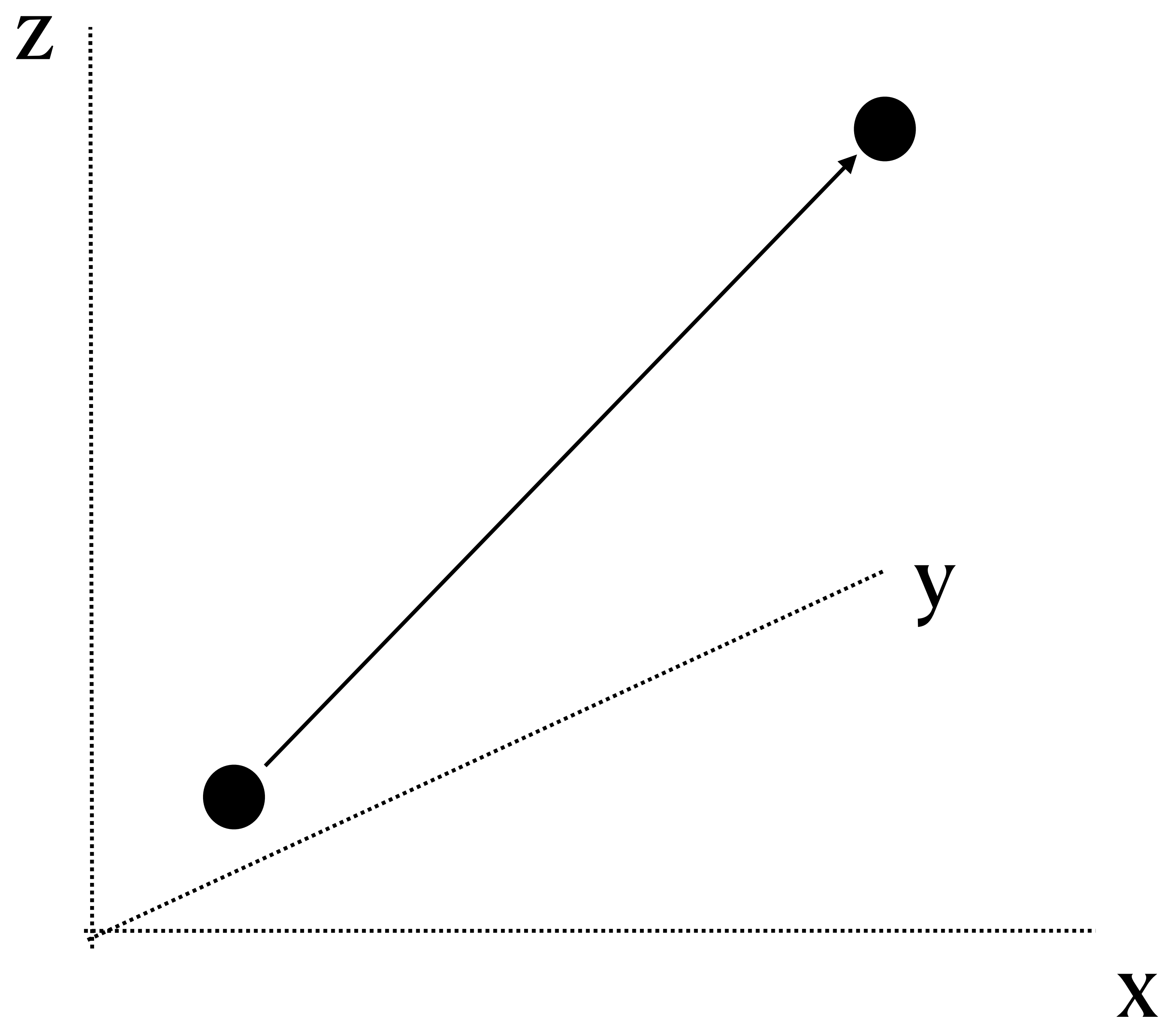
Motivation

System tests consist of executing a trajectory that resembles future deployment environments

Autonomous Car



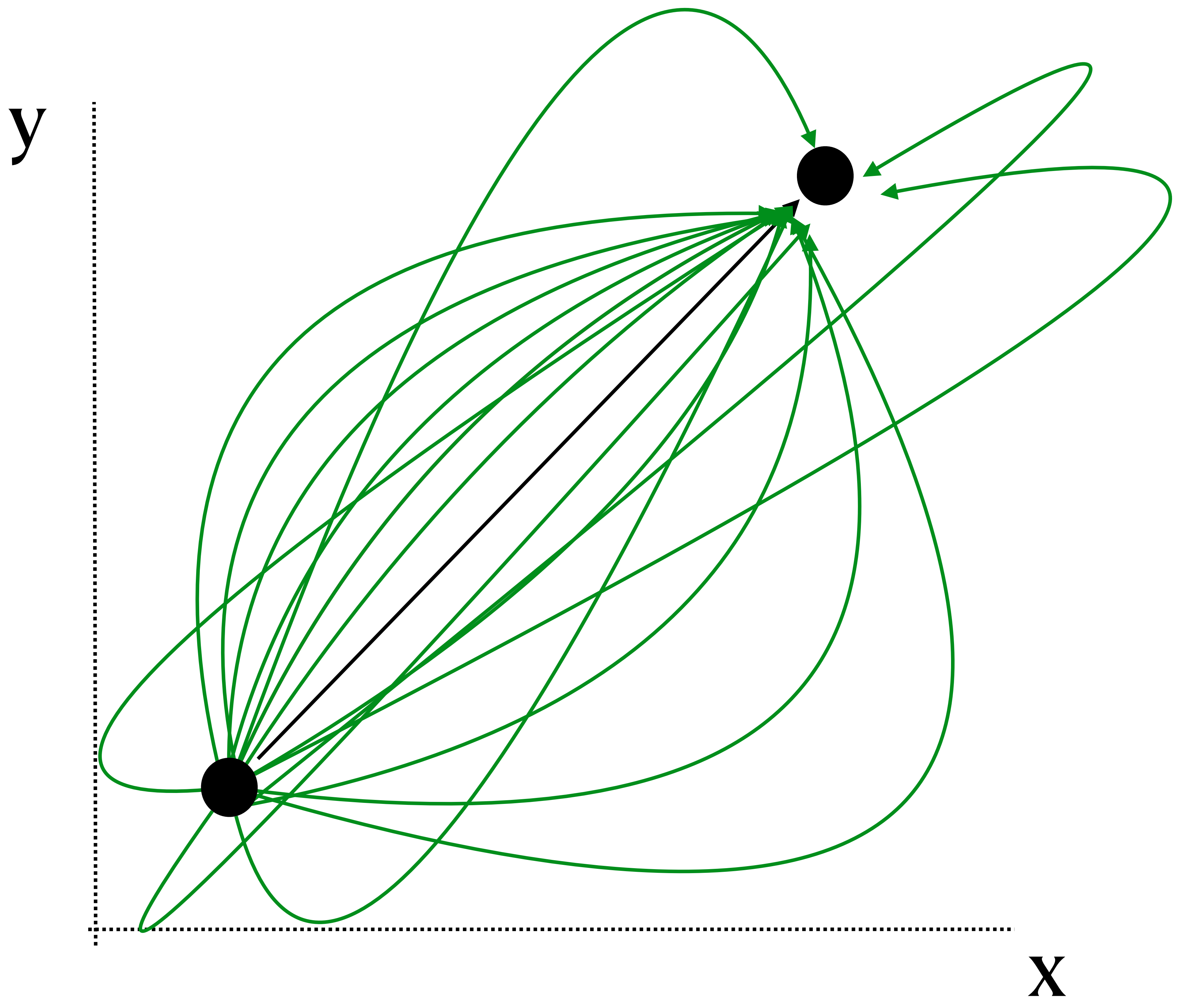
Autonomous Drone



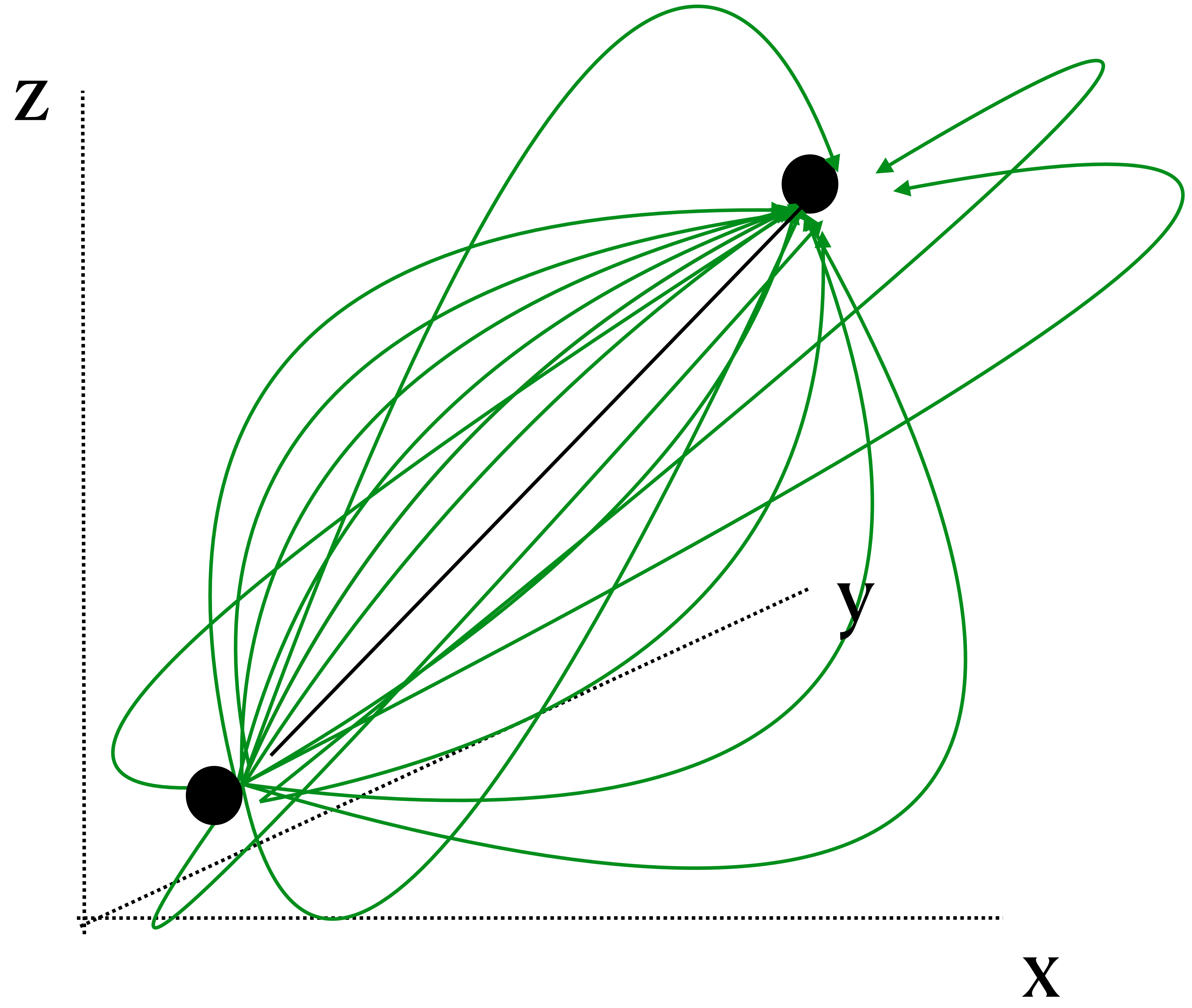
Motivation

However this results in a huge input space which needs to be considered

Autonomous Car



Autonomous Drone

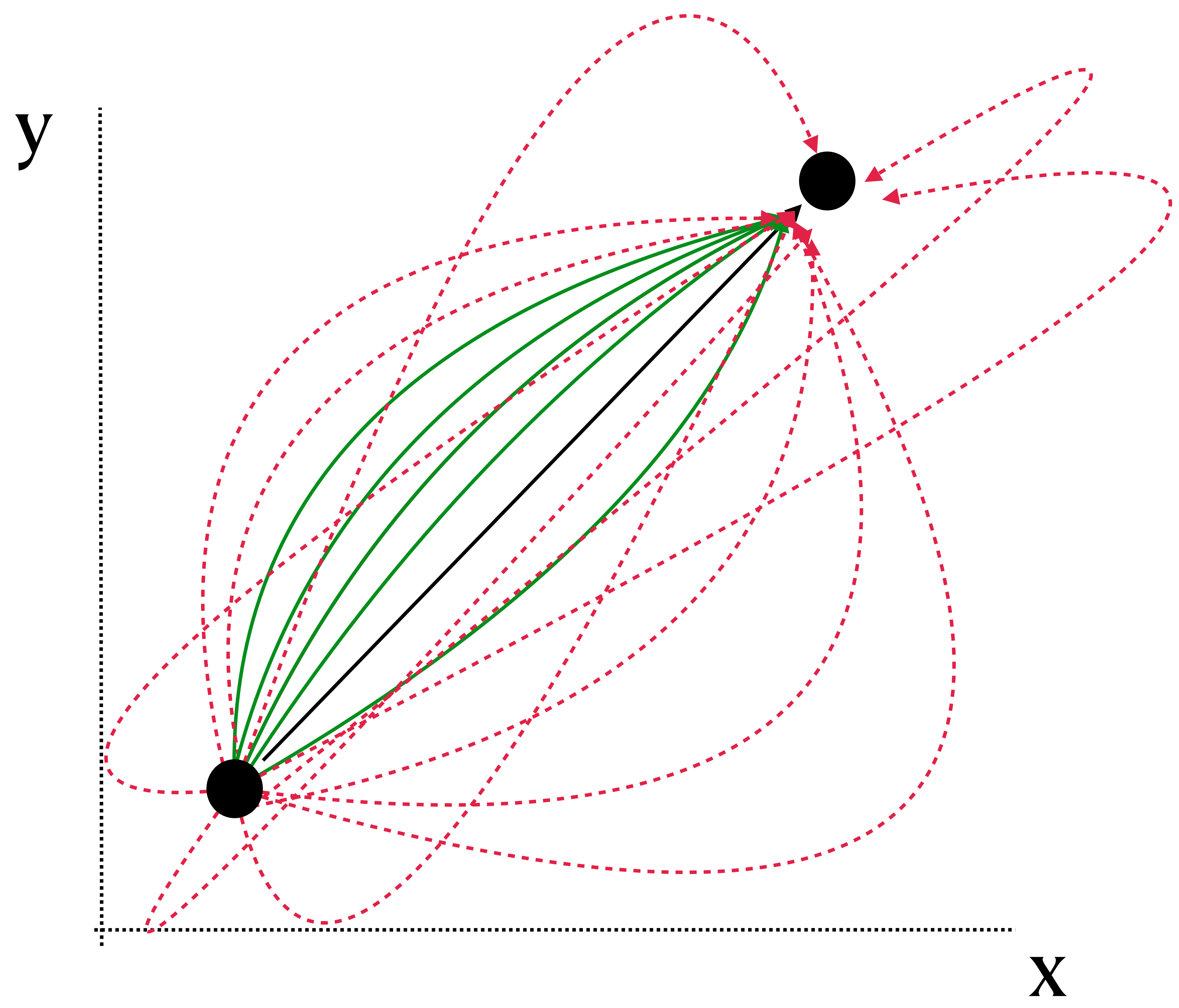


Motivation

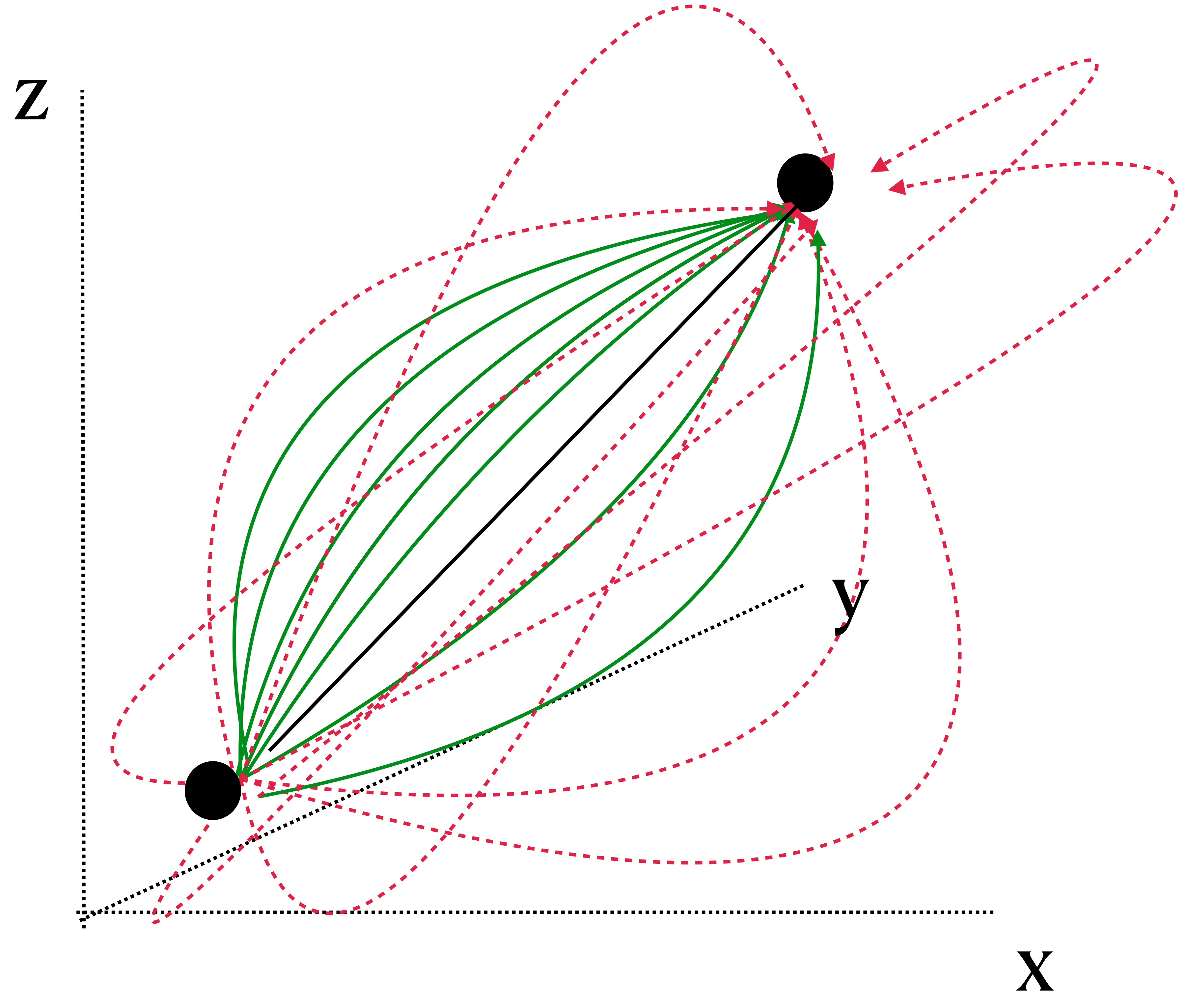
UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

Many of these trajectories are infeasible for the given robot

Autonomous Car



Autonomous Drone



Motivation

University of Virginia | School of Engineering & Applied Science

Exploring typical trajectories is necessary to validate the behavior of mobile robots

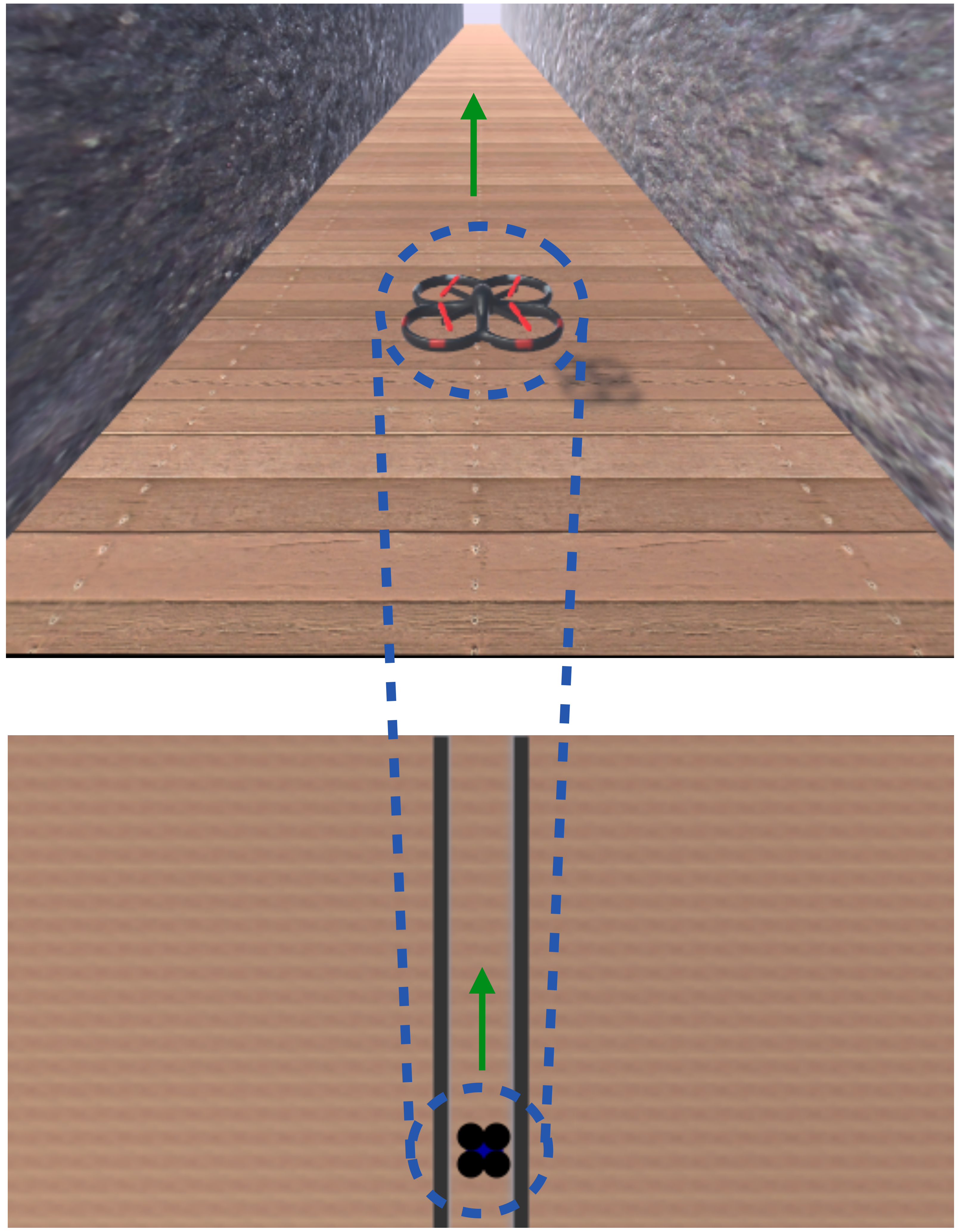
Motivation

Exploring typical trajectories is necessary to validate the behavior of mobile robots

May overlook faults that arise in the presence of stressful trajectories

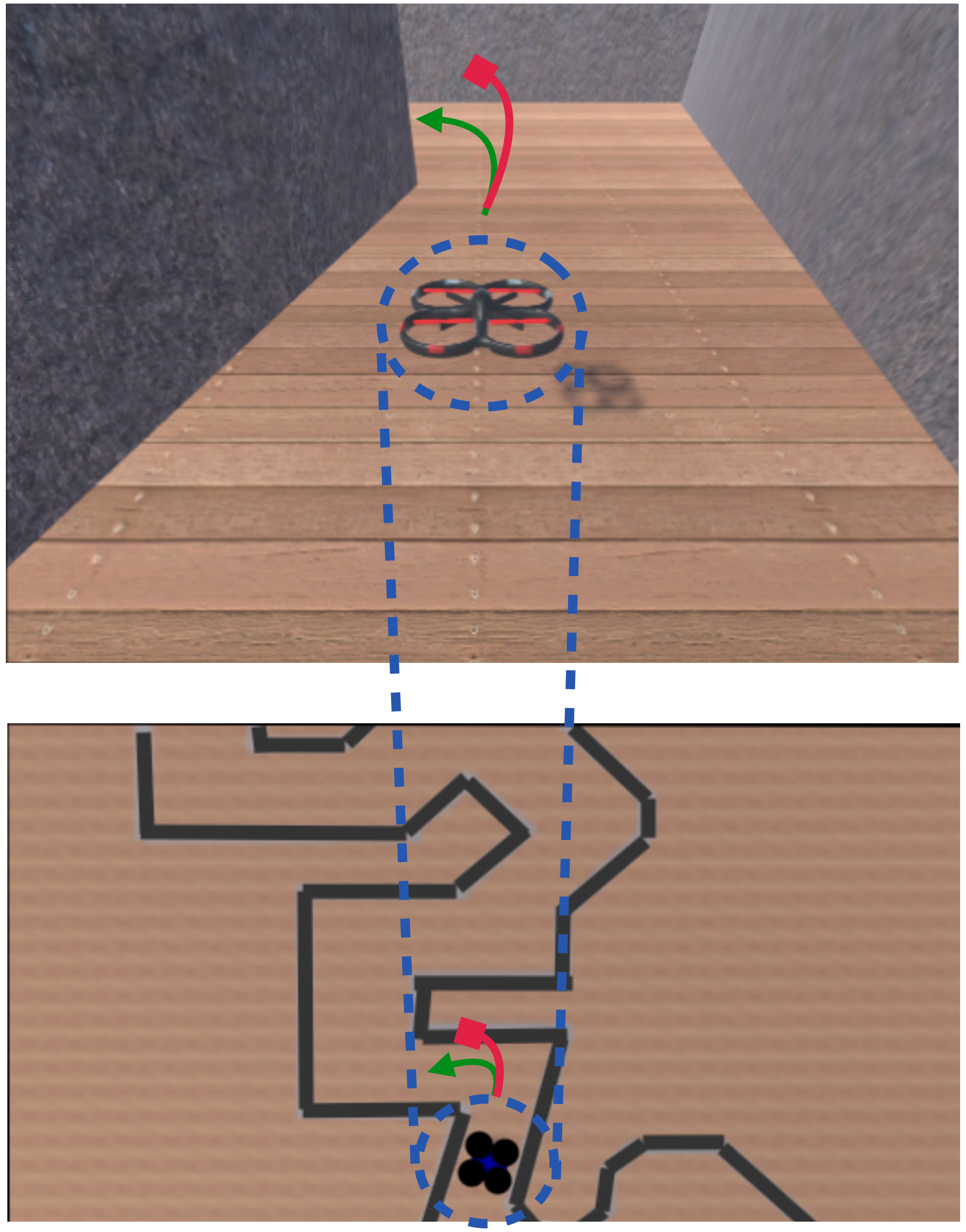
Motivation

May overlook faults that arise in the presence of stressful trajectories



Behind

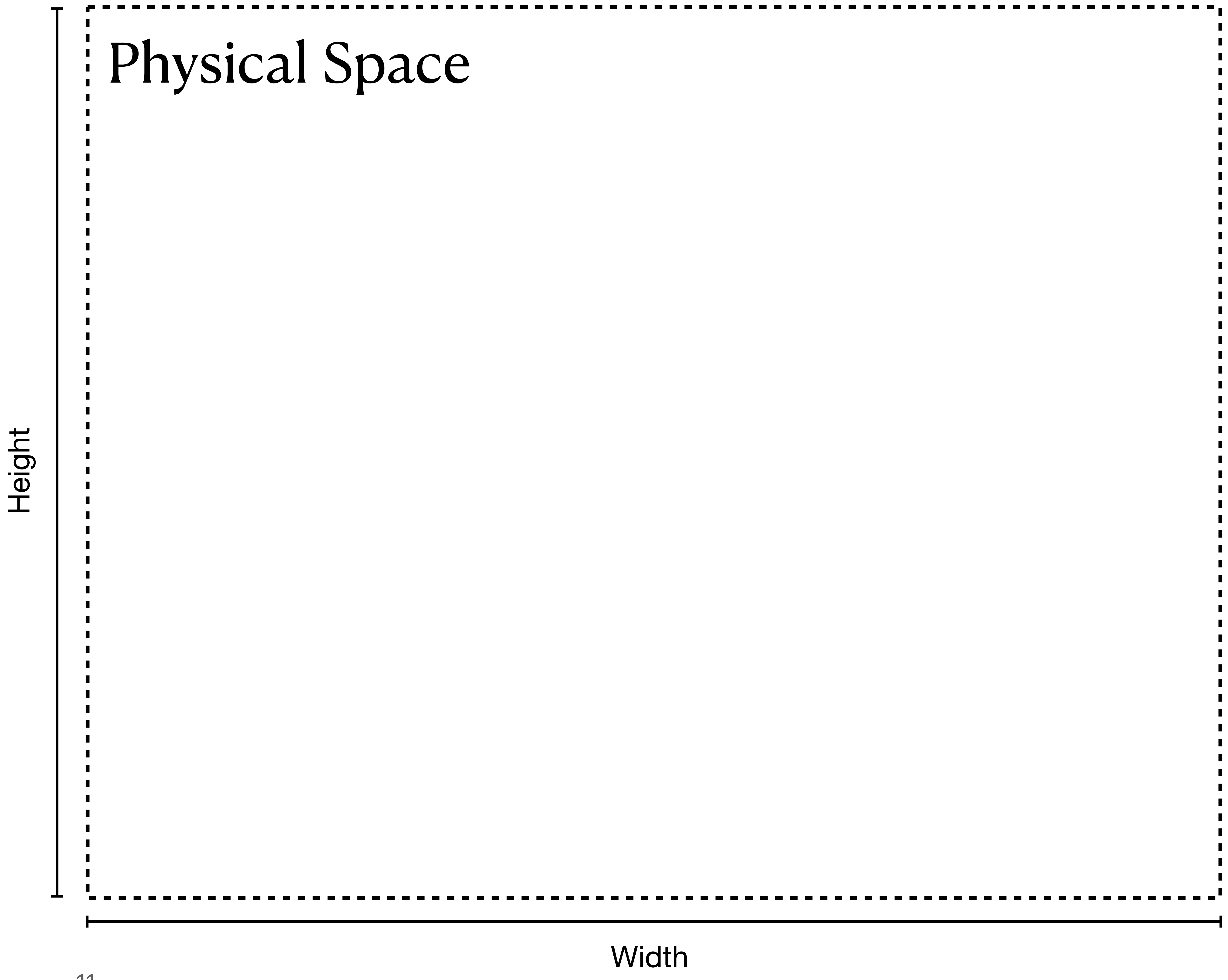
Birds Eye



Problem

Given:

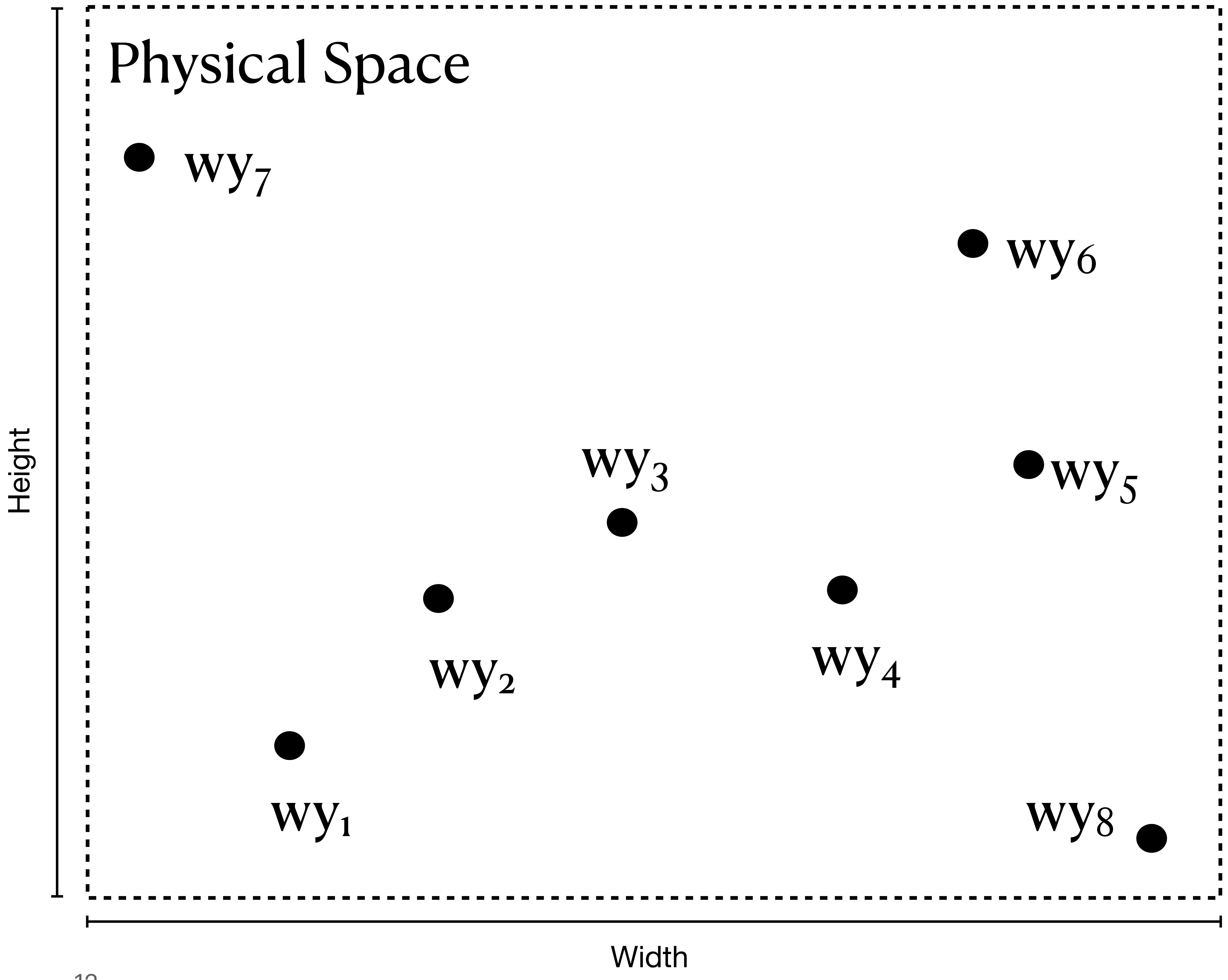
- Physical Space (W)



Problem

Given:

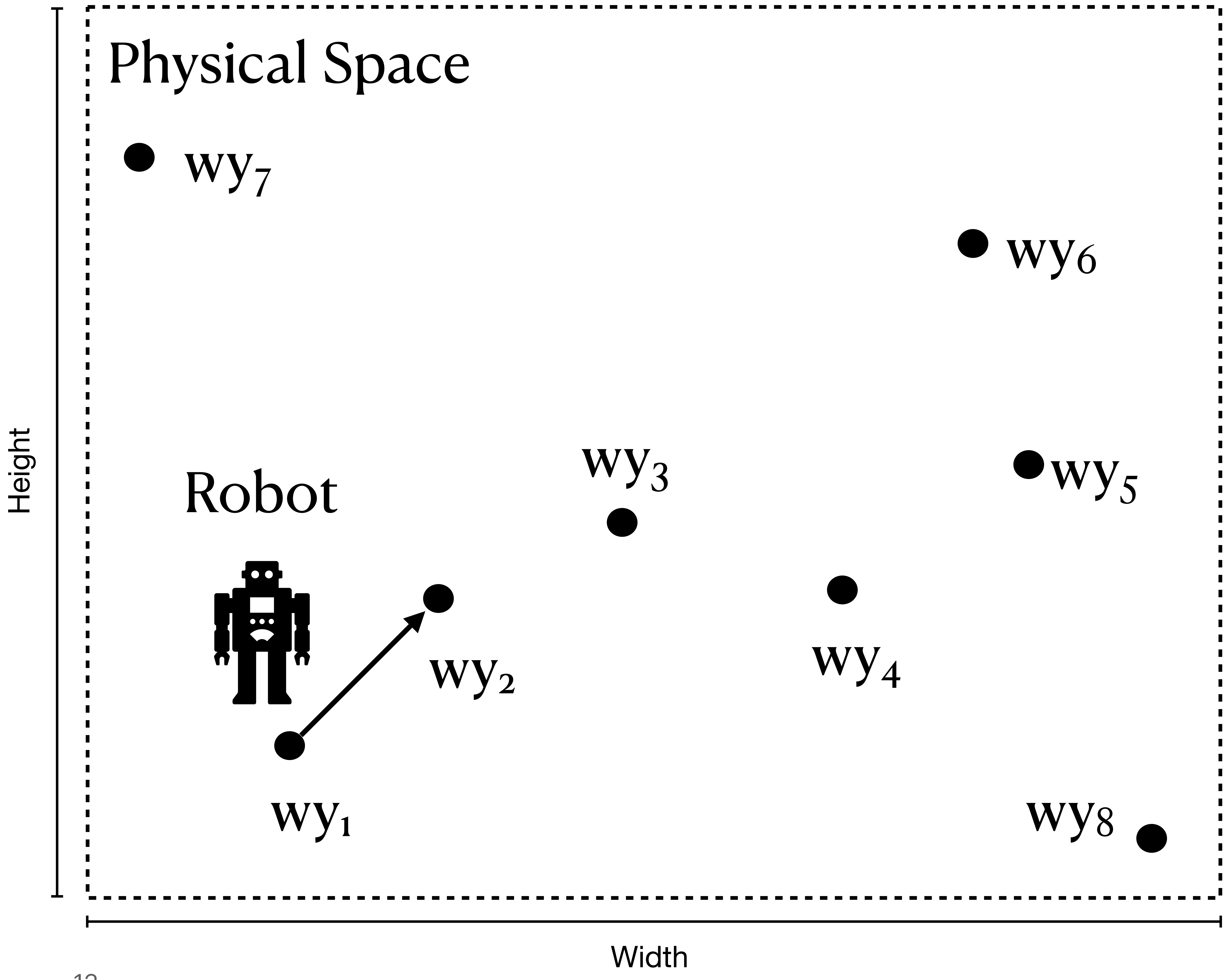
- Physical Space (W)
- $wy \in W$



Problem

Given:

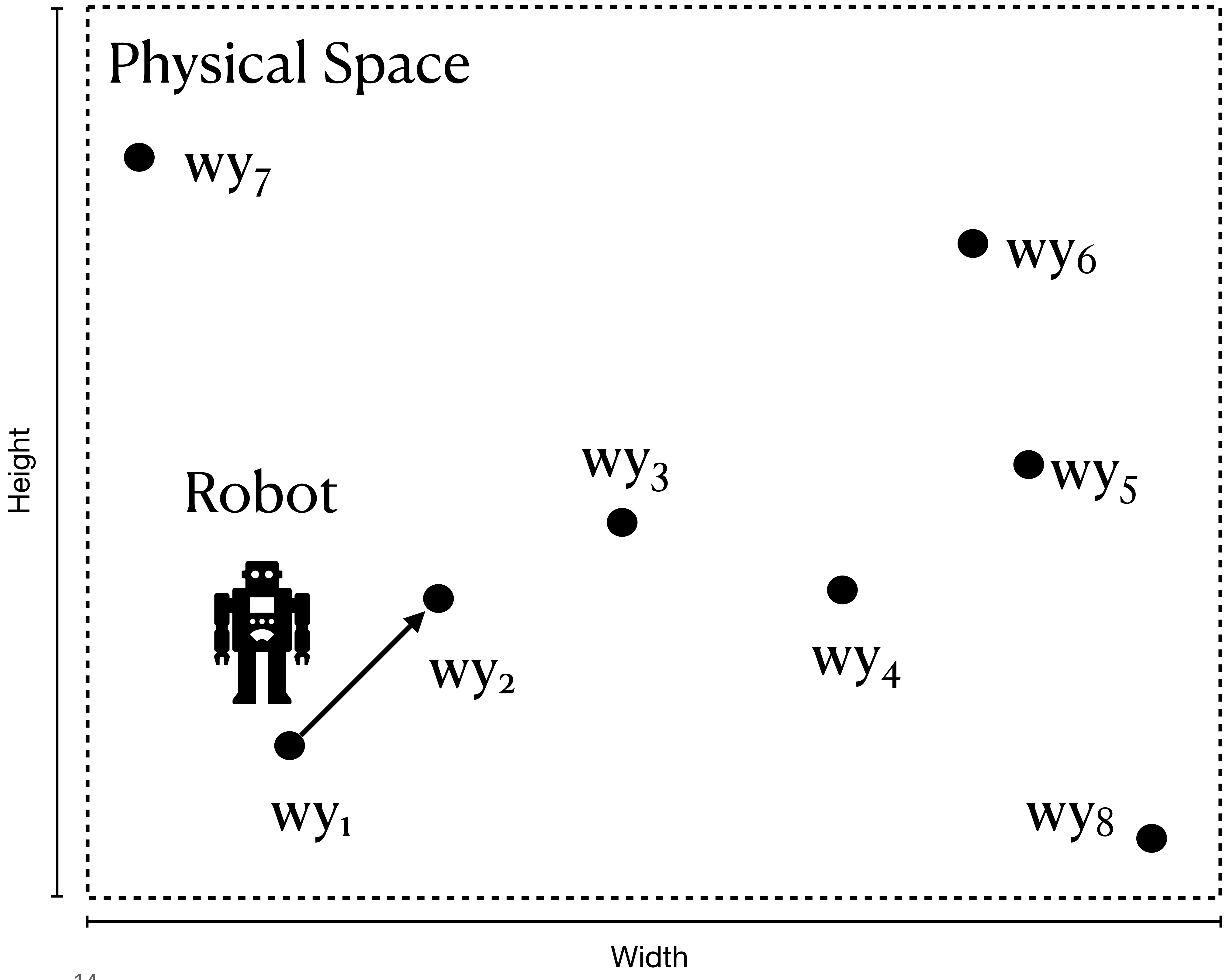
- Physical Space (W)
- wy ∈ W
- Robot (r) can traverse between waypoints such: valid(r) ⊆ W X W



Problem

Given:

- Physical Space (W)
- $wy \in W$
- Robot (r) can traverse between waypoints such: $valid(r) \subseteq W \times W$
- r arrives at a given wy_i with state s_i



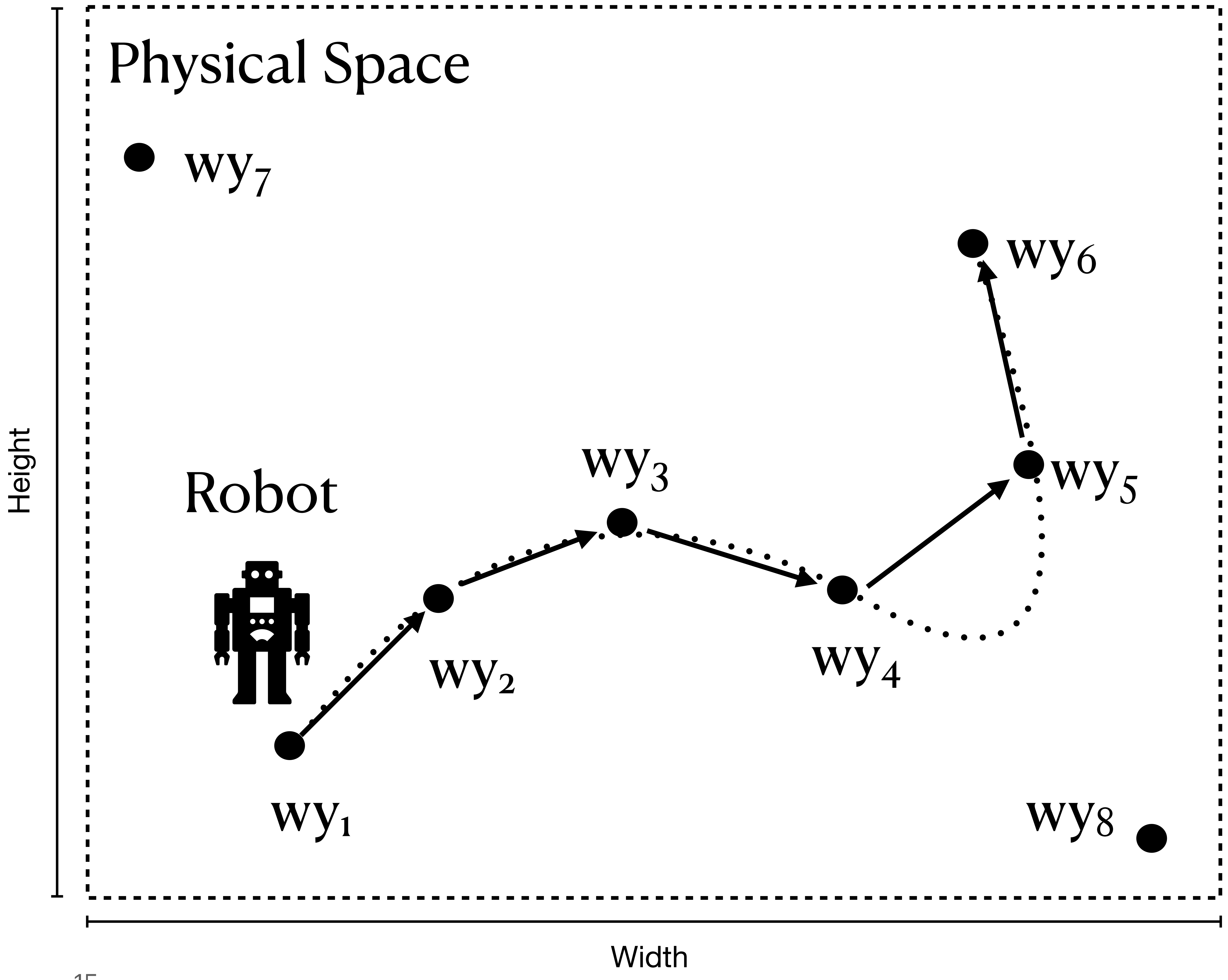
Problem

Given:

- Physical Space (W)
- wy $\in$ W
- Robot (r) can traverse between waypoints such: valid(r) $\subseteq$ W X W
- r arrives at a given wy_i with state s_i

We want:

- traj = <s_0,s_1, … s_N>



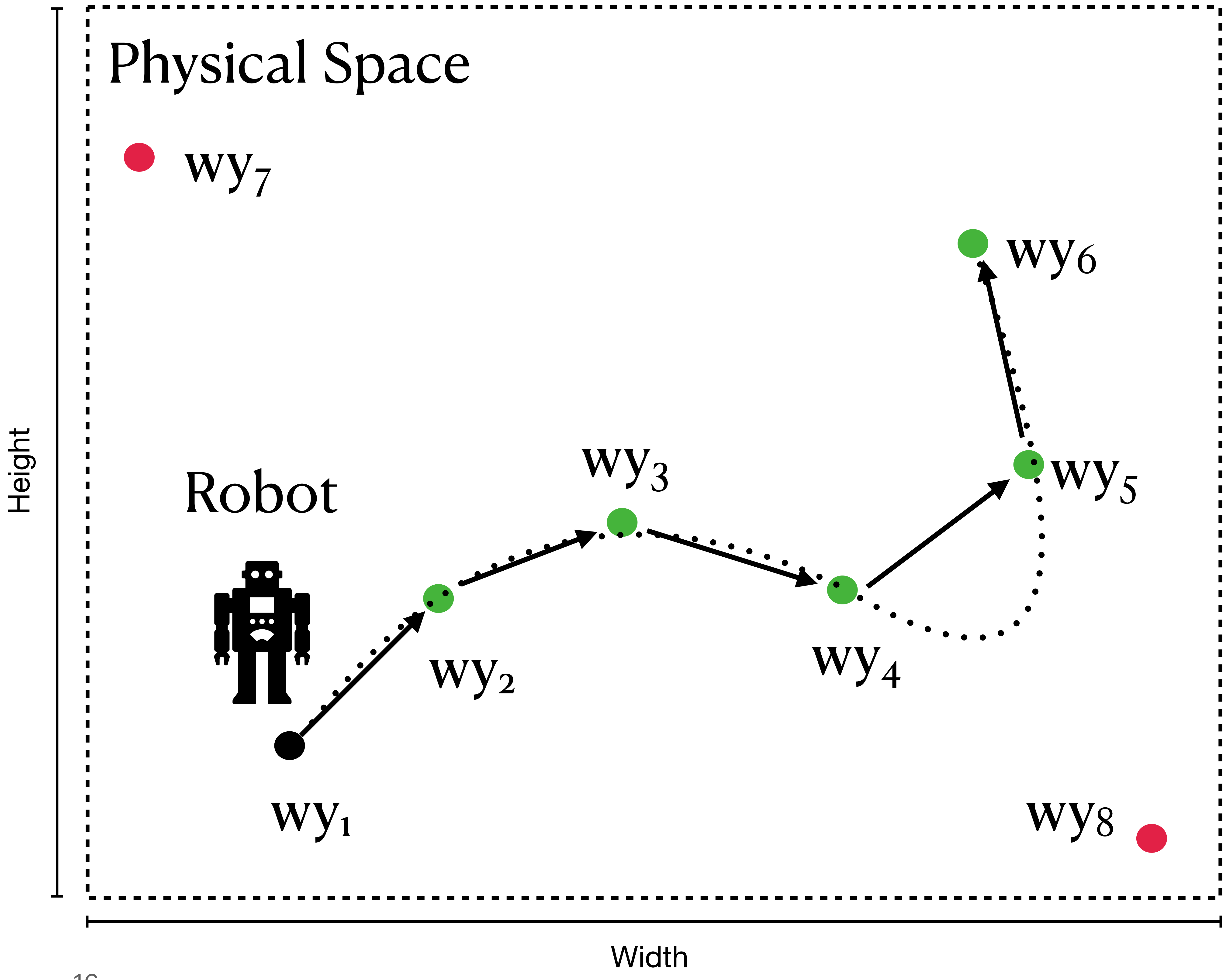
Problem

Given:

- Physical Space (W)
- $wy \in W$
- Robot (r) can traverse between waypoints such: $valid(r) \subseteq W \times W$
- r arrives at a given wy_i with state s_i

We want:

- $traj = <s_0, s_1, \ldots s_N>$
- **Feasible:** $traj_f = \{ traj \mid \forall 0 \leq i < n: traj[i], traj[i+1]) \in valid(r)\}$



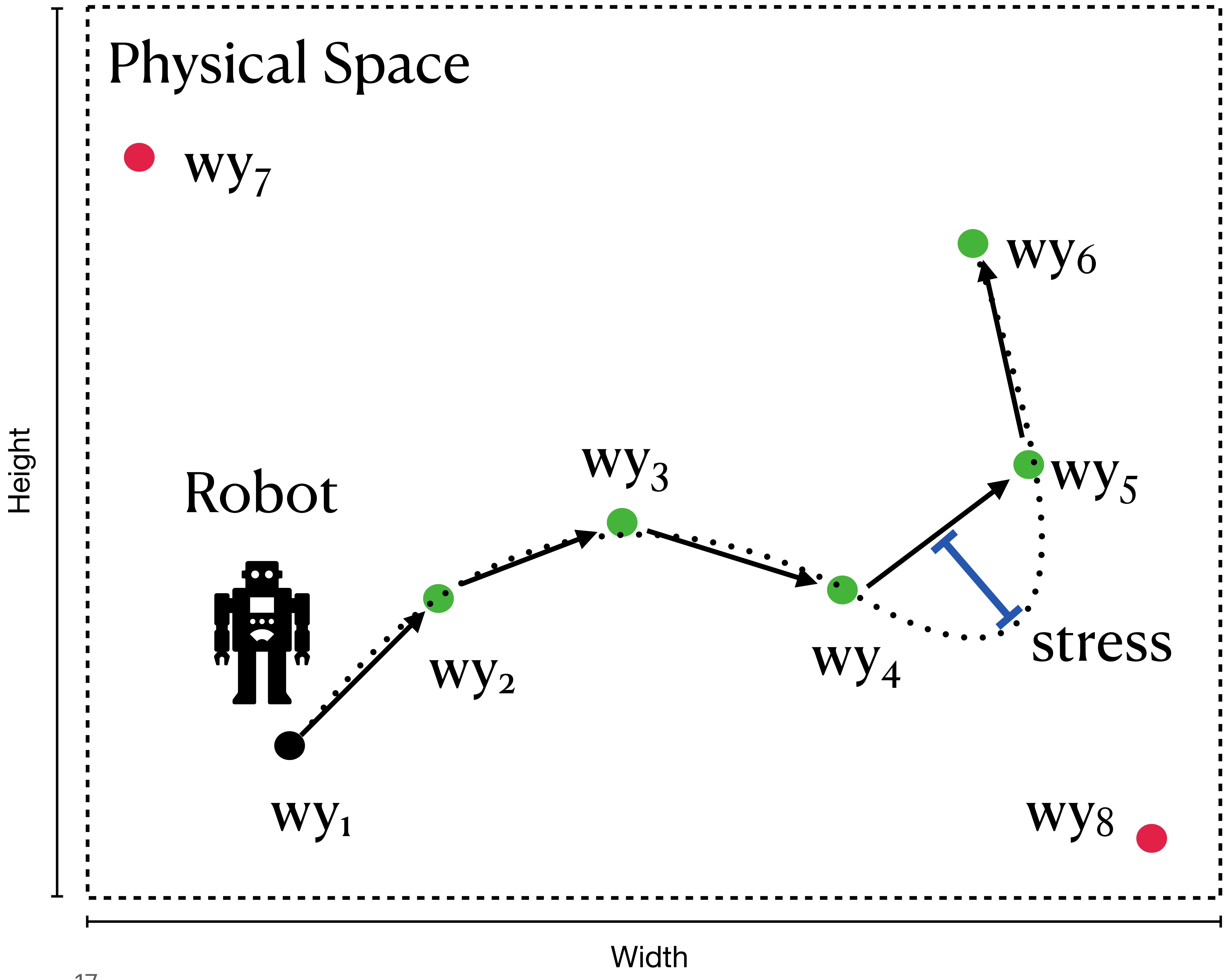
Problem

Given:

- Physical Space (W)
- wy $\in$ W
- Robot (r) can traverse between waypoints such: valid(r) $\subseteq$ W X W
- r arrives at a given wy_i with state s_i

We want:

- traj = $<s_0, s_1, \ldots s_N>$
- **Feasible:** $traj_f = \{ traj \mid \forall 0 \leq i < n: traj[i], traj[i+1]) \in valid(r)\}$
- *score*: W x W $\mapsto \mathbb{R}$ defines stress on r



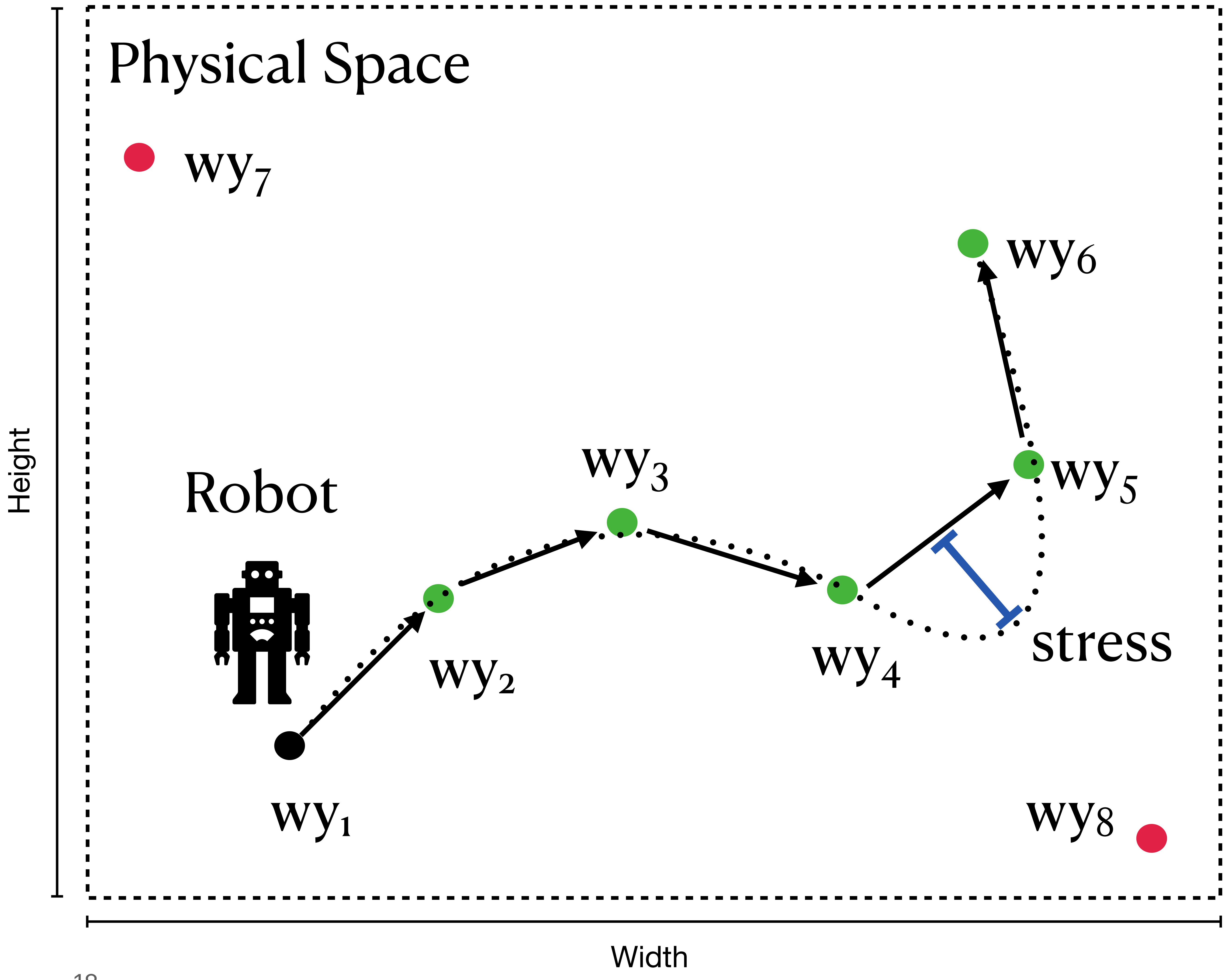
Problem

Given:

- Physical Space (W)
- $wy \in W$
- Robot (r) can traverse between waypoints such: $valid(r) \subseteq W \times W$
- r arrives at a given wy_i with state s_i

We want:

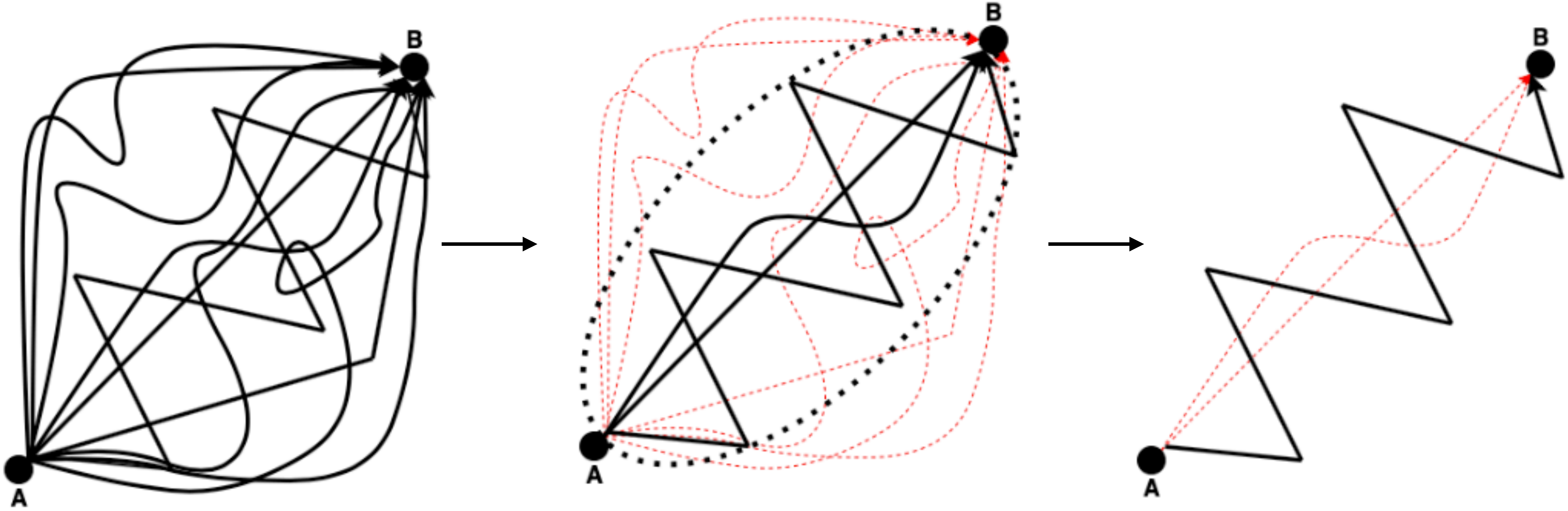
- $traj = <s_0, s_1, \ldots s_N>$
- **Feasible:** $traj_f = \{ traj \mid \forall 0 \le i < n: traj[i], traj[i+1]) \in valid(r)\}$
- *score*: $W \times W \mapsto \mathbb{R}$ defines stress on r
- **Stressful**: $traj_s \in traj_f$ such that $\forall\, traj \in Traj_f : score(traj) \le score(traj_s)$



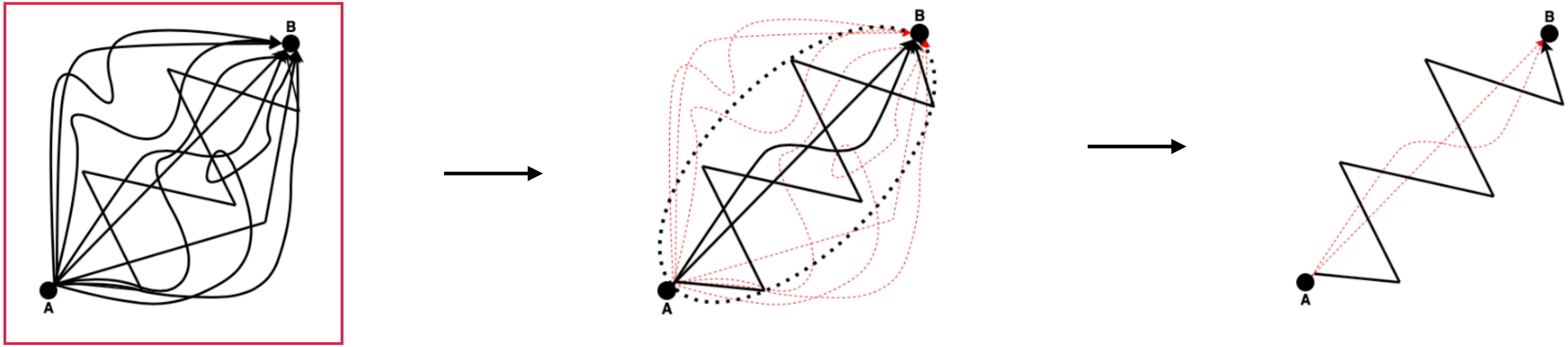
Conceptual Solution

Algorithmic solution is presented in the paper.

Goal: **Feasible** yet **stressful** trajectories

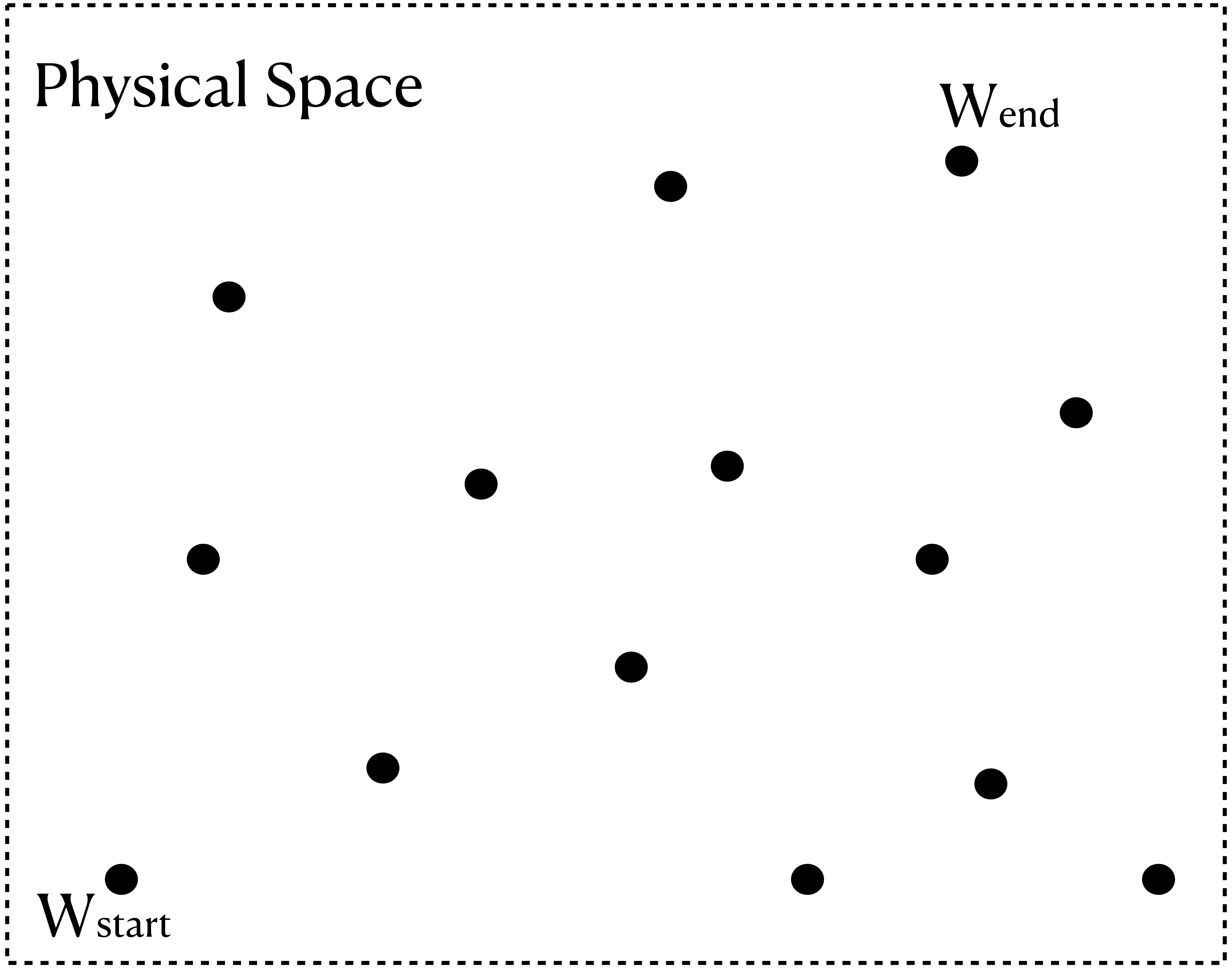


Conceptual Solution



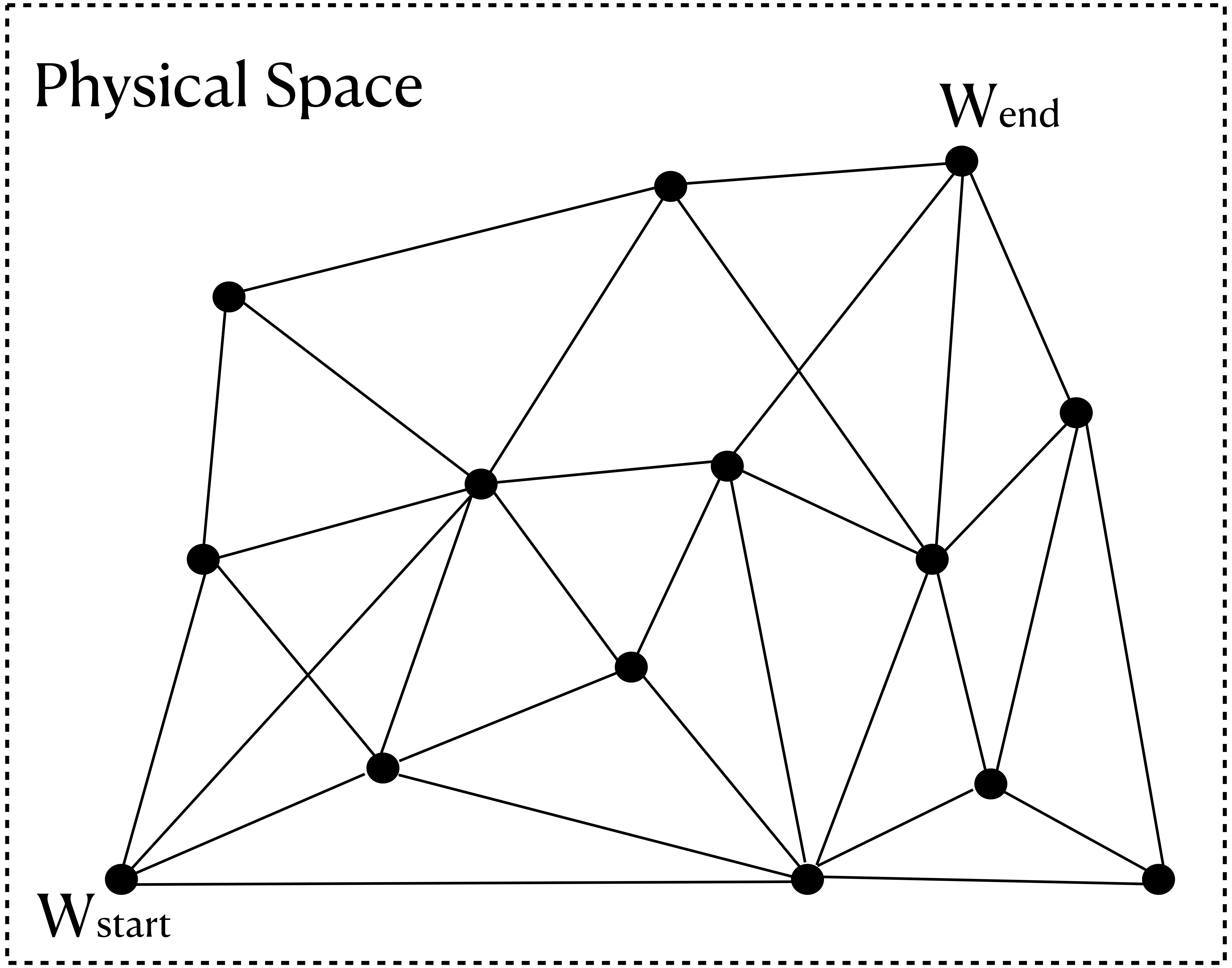
Generating Trajectories

Populate physical space with random waypoints



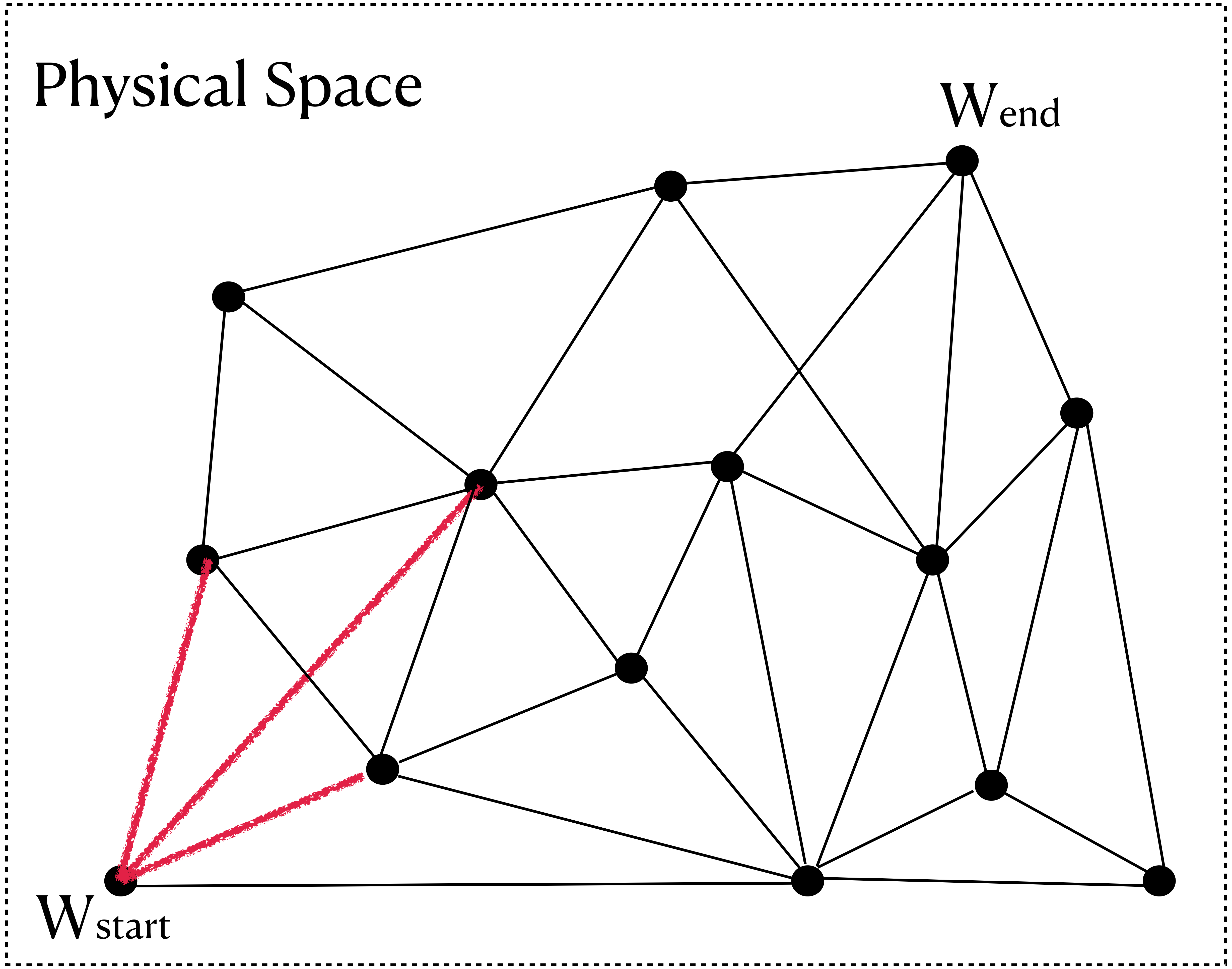
Generating Trajectories

Connect waypoints with edges



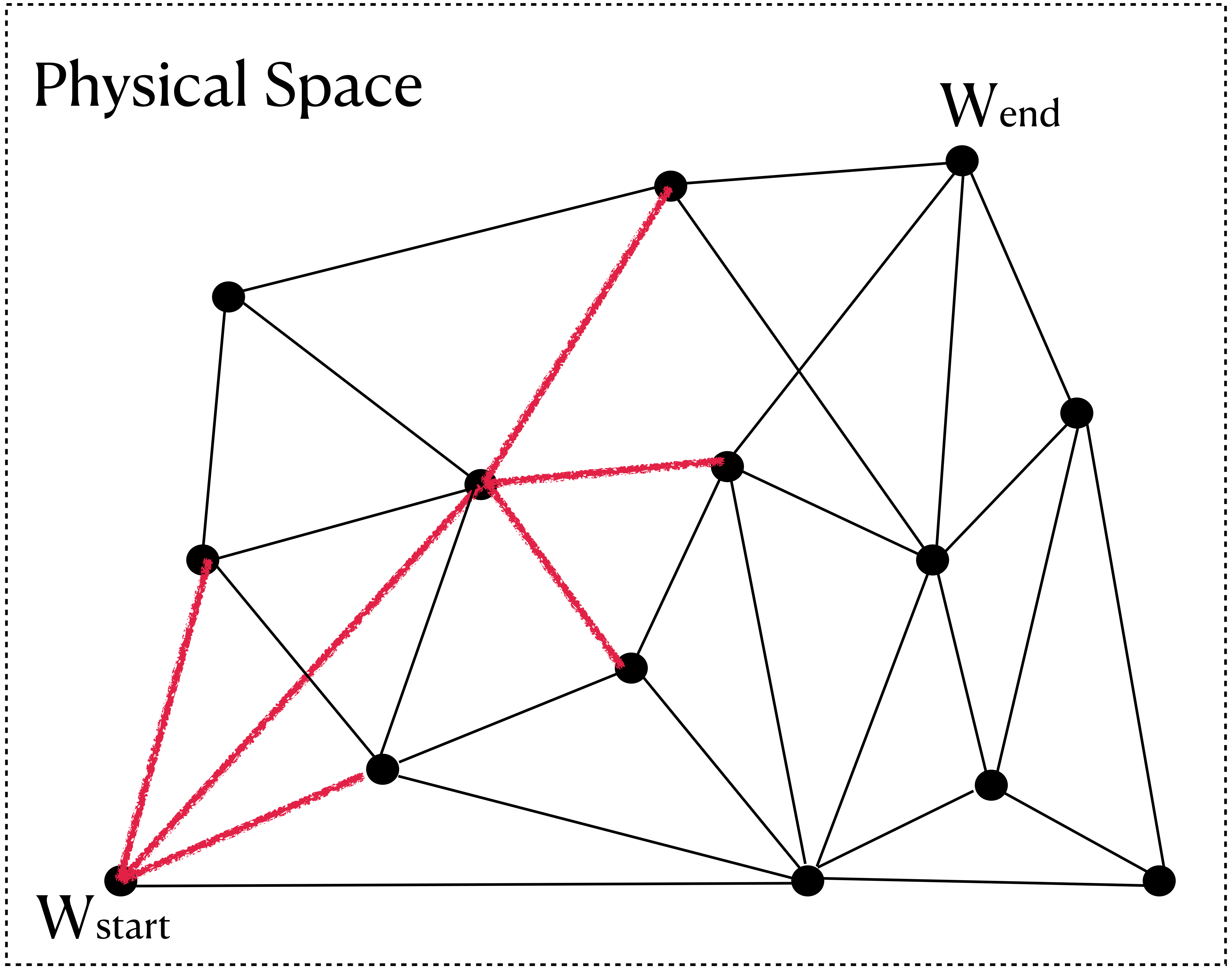
Generating Trajectories

Graph search problem



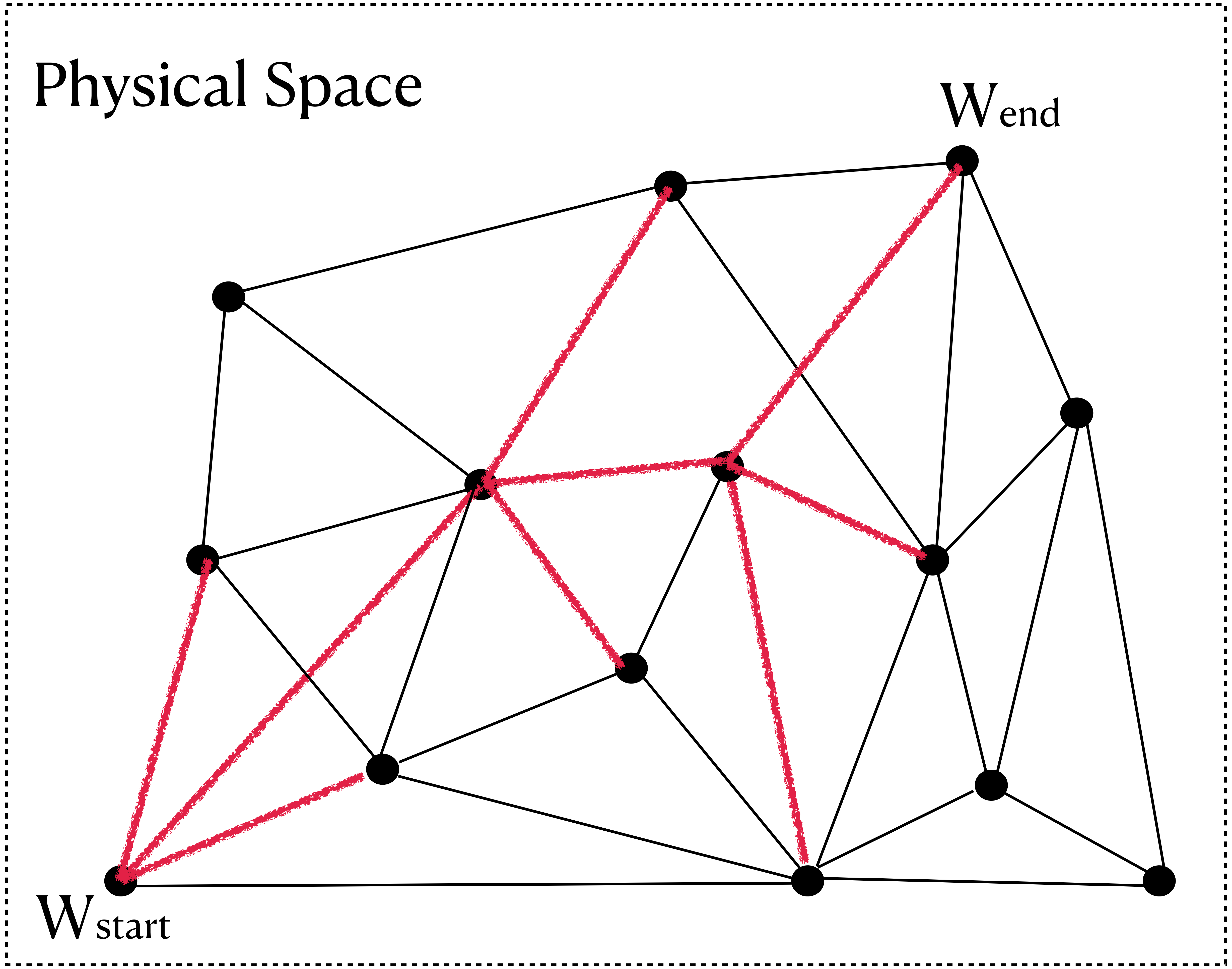
Generating Trajectories

Graph search problem



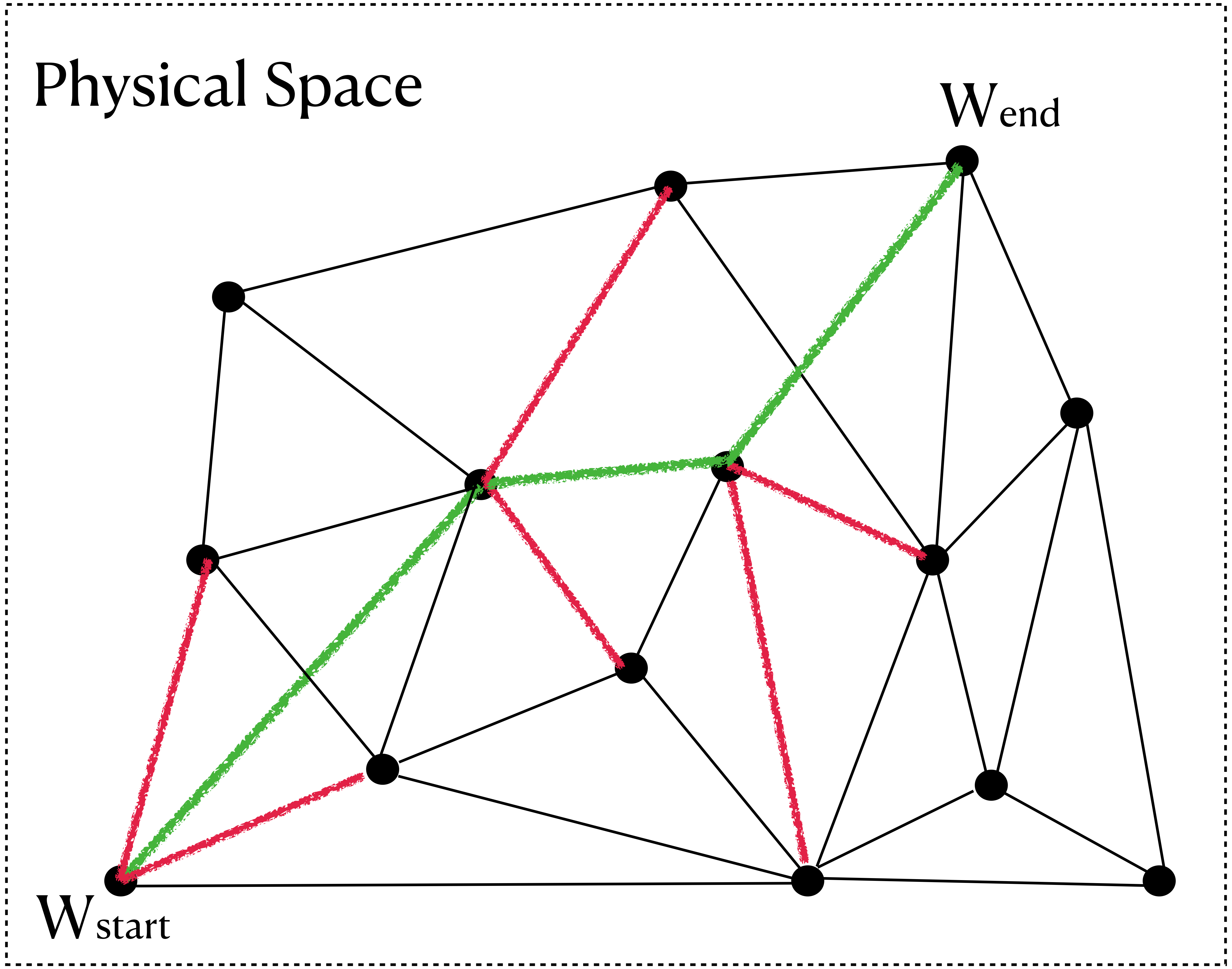
Generating Trajectories

Graph search problem

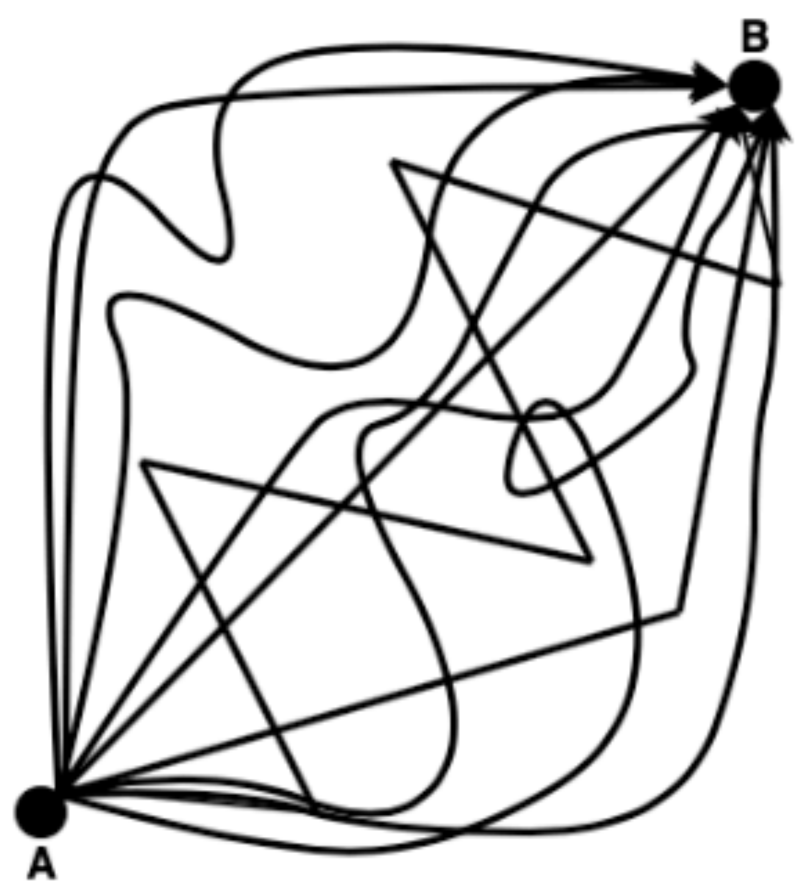


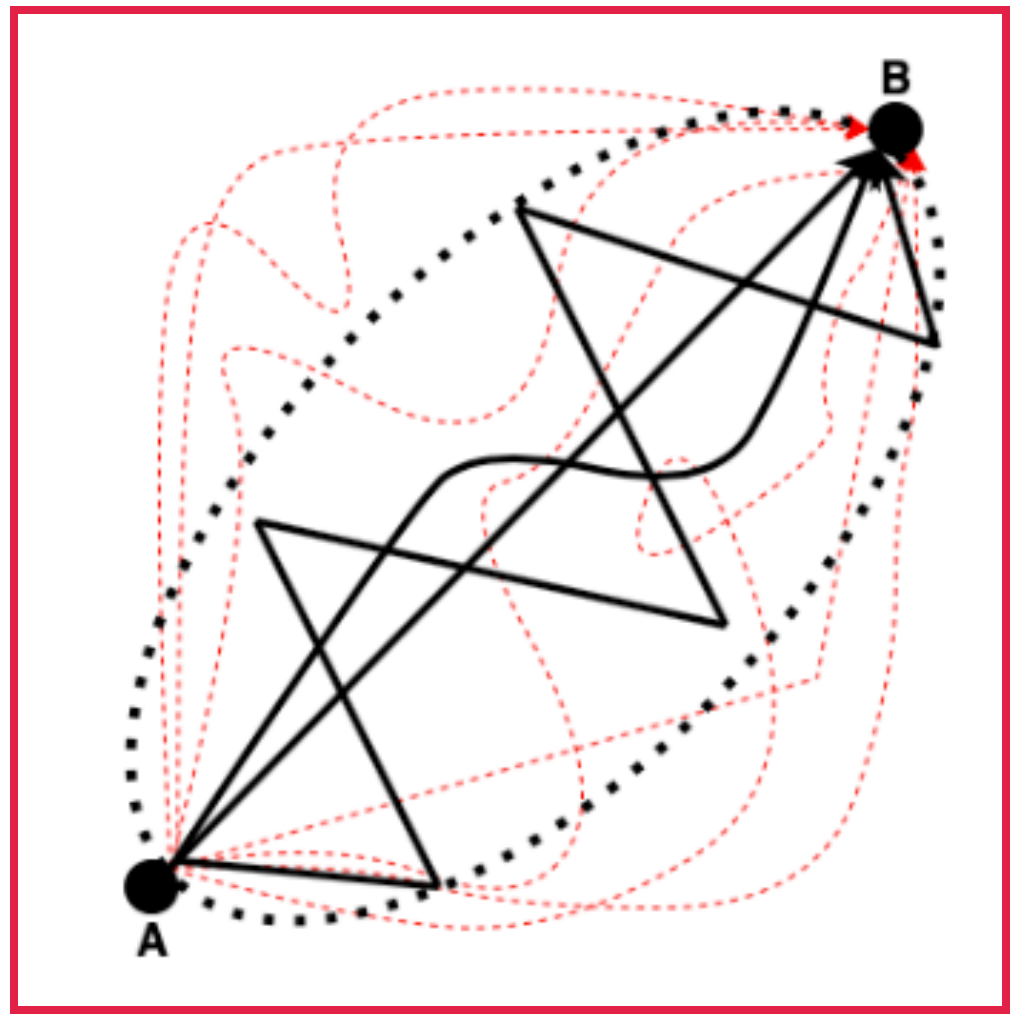
Generating Trajectories

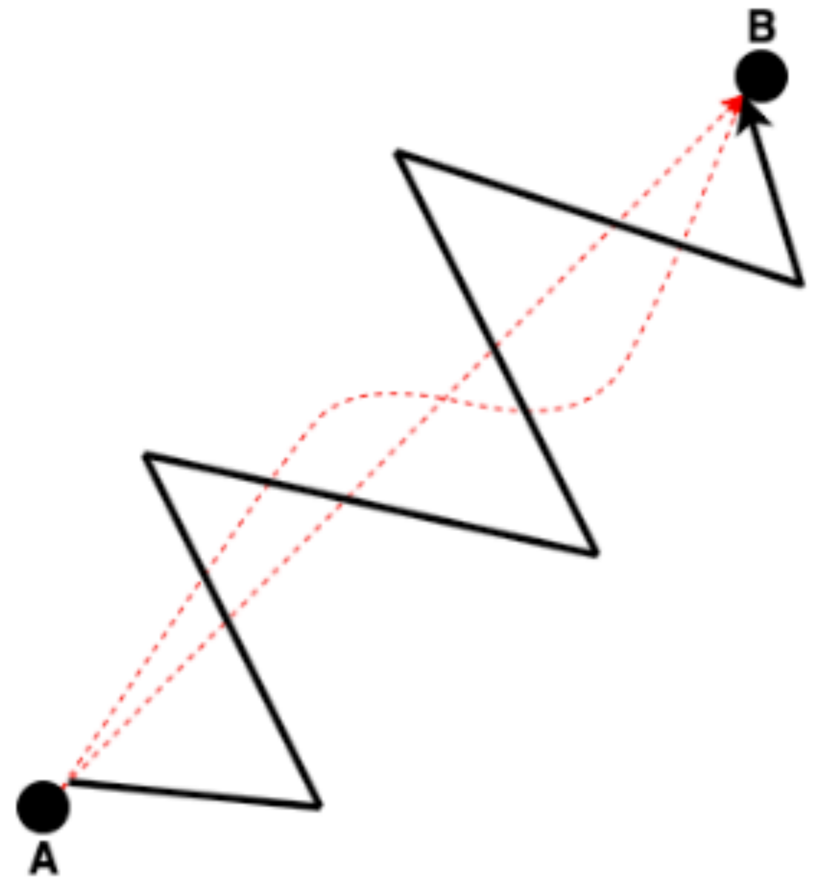
Graph search problem



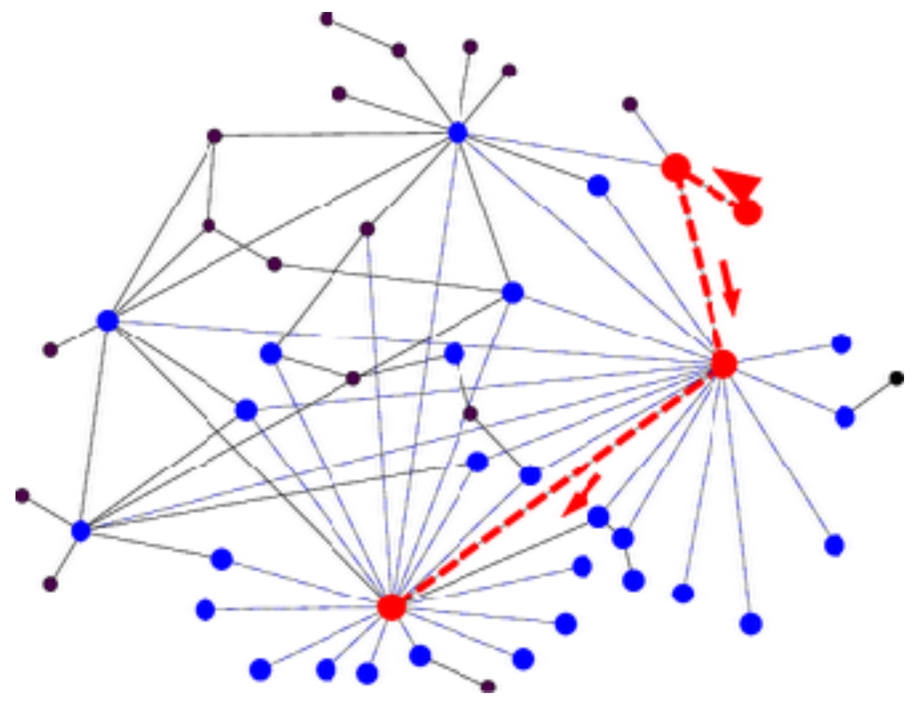
Solution







Search through world
looking for all trajectories

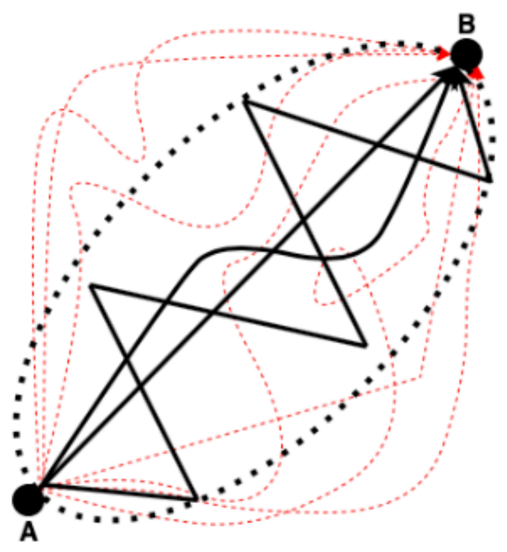


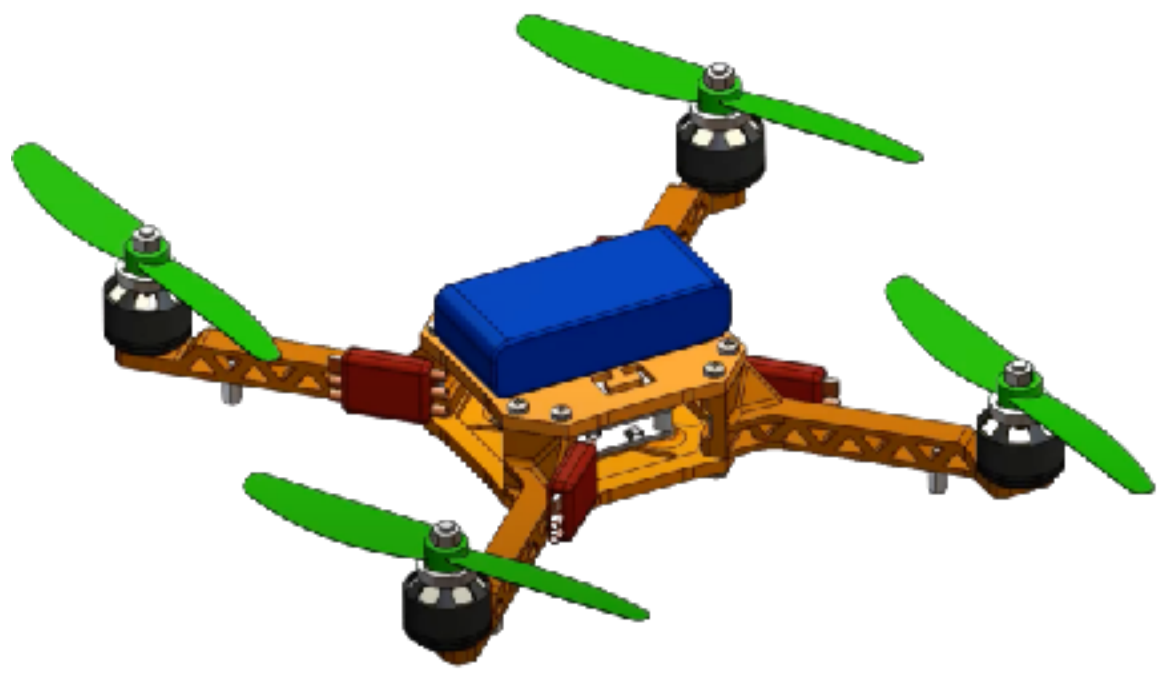
Graph search problem

Feasible Trajectories

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

How to select only **feasible** trajectories given the robot?





Feasible?
Infeasible?

Physical Model

Kinematic and Dynamic Models

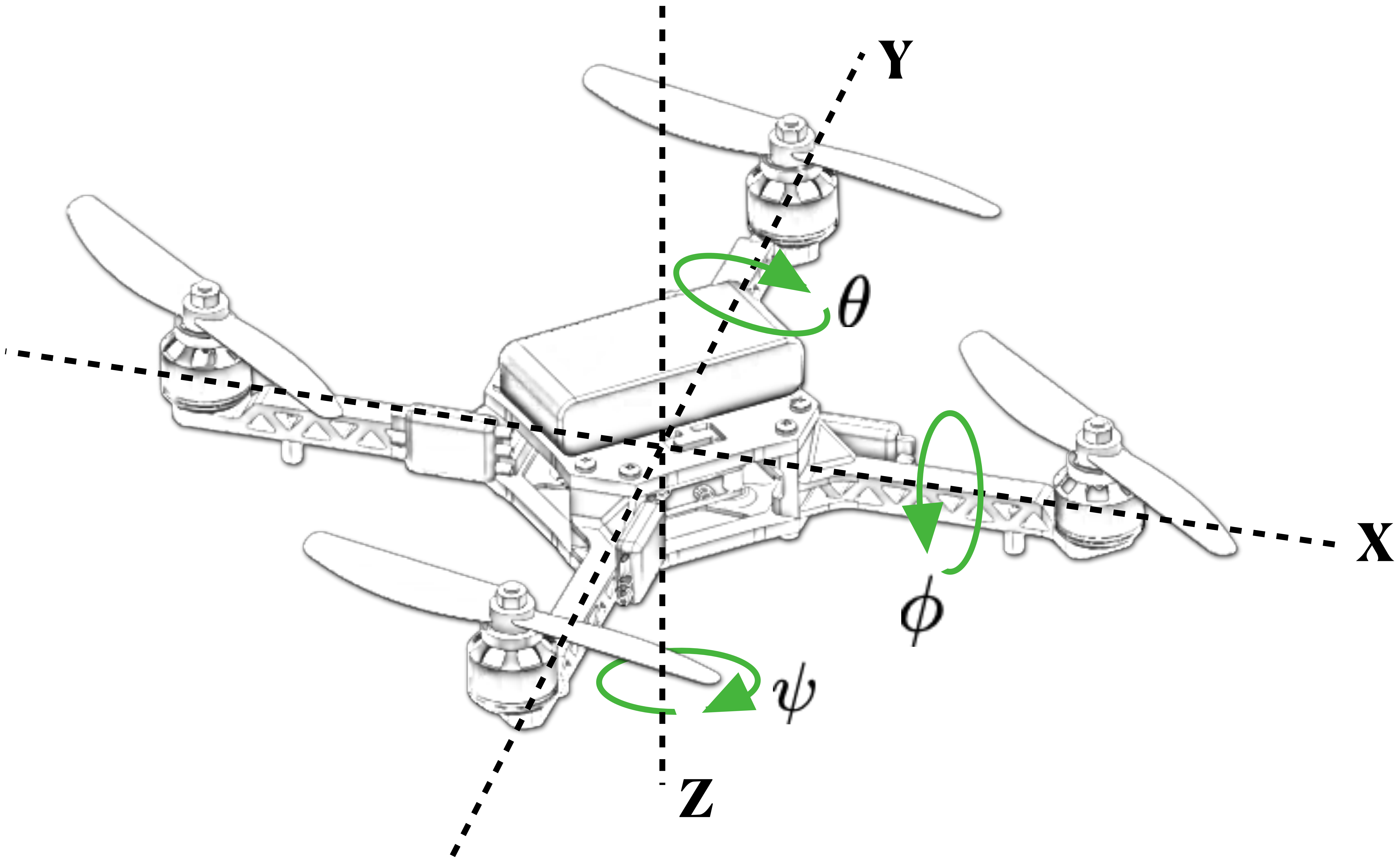
How to select only **feasible** trajectories given the robot?

Kinematic and Dynamic Models

$$\mathbf{s} = [x\ y\ z\ \phi\ \theta\ \psi\ v_x\ v_y\ v_z\ \omega_x\ \omega_y\ \omega_z]^T$$

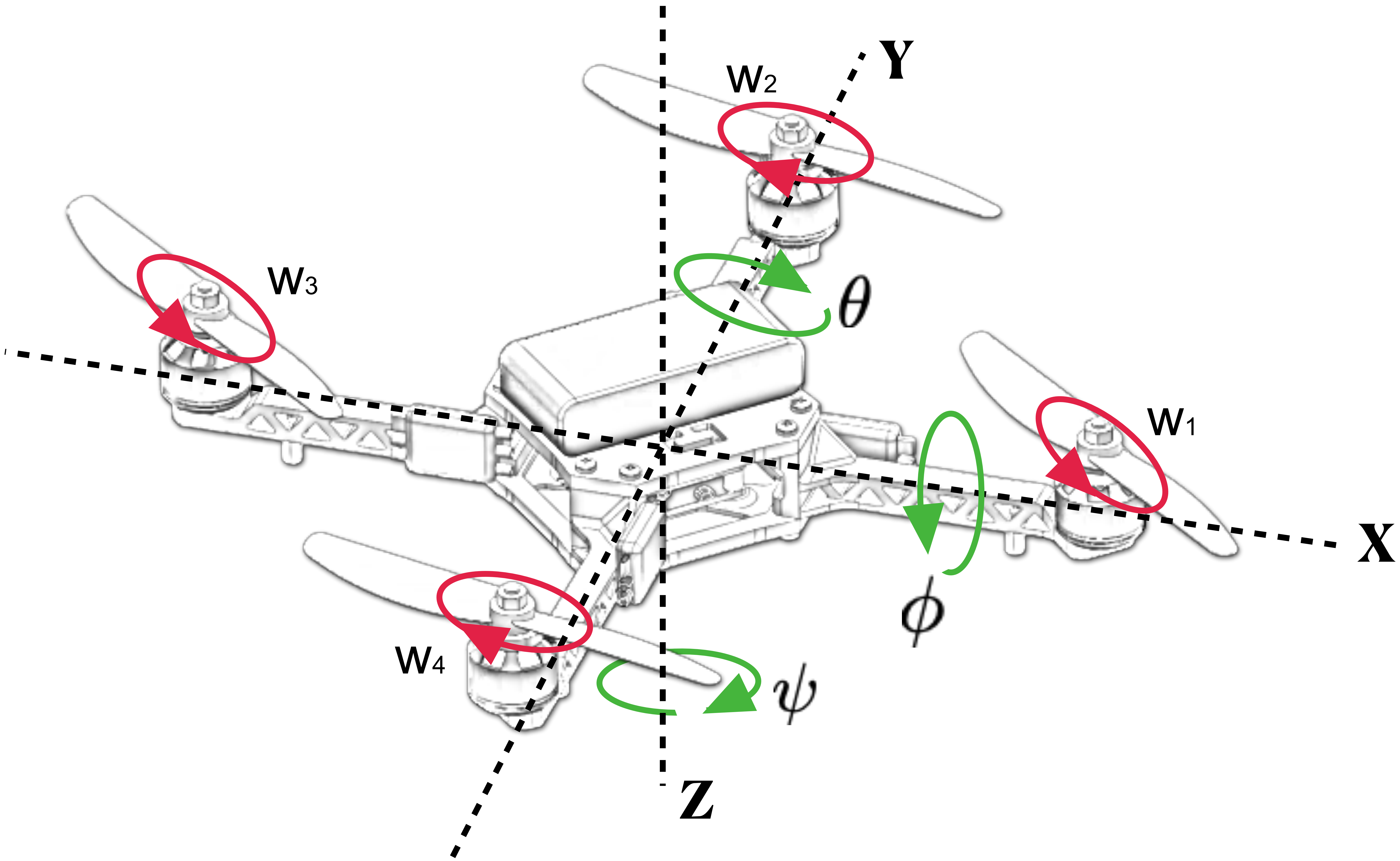
Kinematic and Dynamic Models

$$\mathbf{s} = \left[\boxed{x\ y\ z}\ \boxed{\phi\ \theta\ \psi}\ \boxed{v_x\ v_y\ v_z}\ \boxed{\omega_x\ \omega_y\ \omega_z}\right]^T$$



Kinematic and Dynamic Models

A quadrotor is controlled by changing the velocity of the propellers.



Kinematic and Dynamic Models

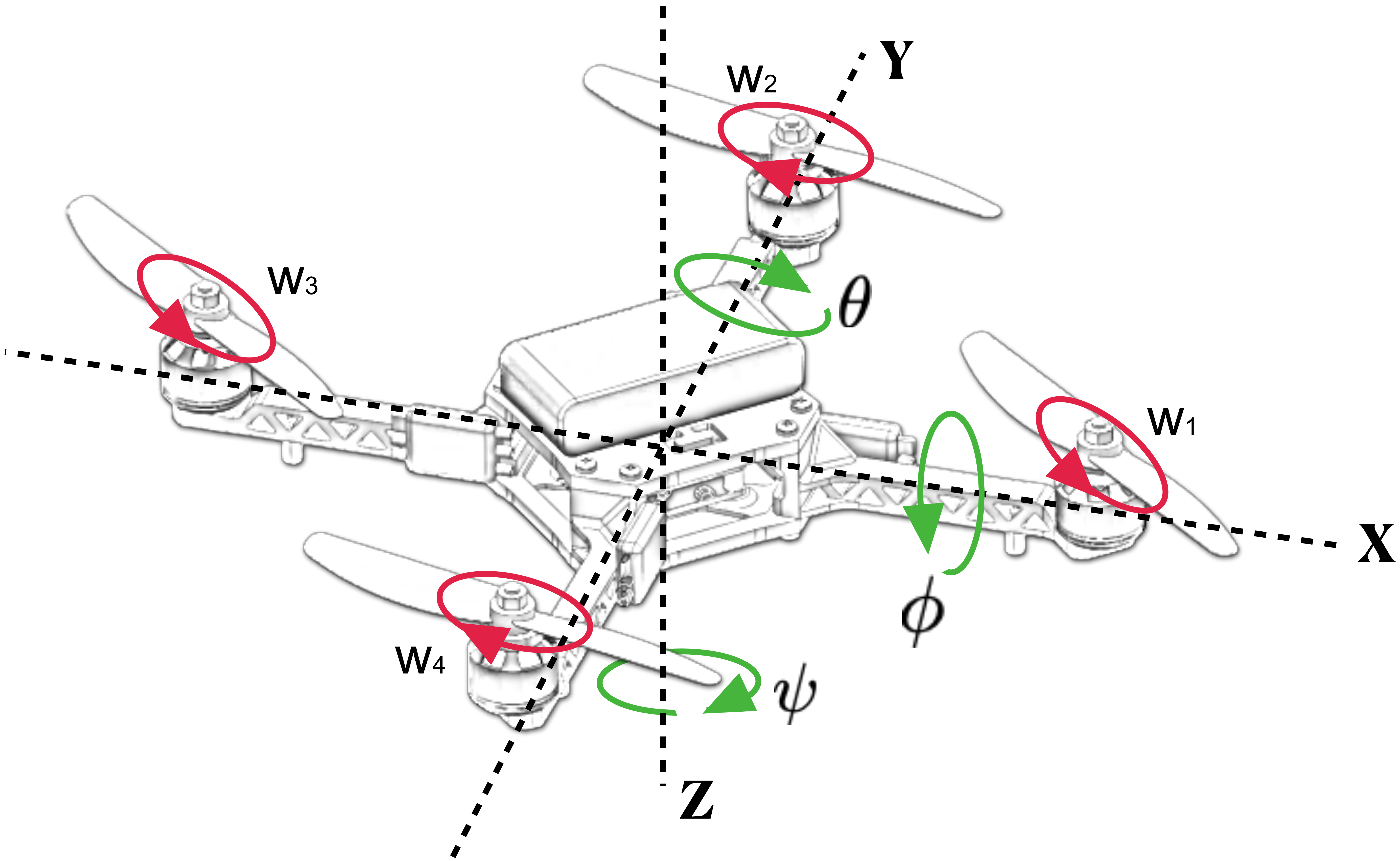
$$\mathbf{s} = [x \; y \; z \; \phi \; \theta \; \psi \; v_x \; v_y \; v_z \; \omega_x \; \omega_y \; \omega_z]^T$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} F \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & dk_f & 0 & -dk_f \\ -dk_f & 0 & dk_f & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix}$$

$$\begin{bmatrix} \dot{\omega_x} \\ \dot{\omega_y} \\ \dot{\omega_z} \end{bmatrix} = \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}} \omega_y \omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}} \omega_x \omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}} \omega_x \omega_y \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

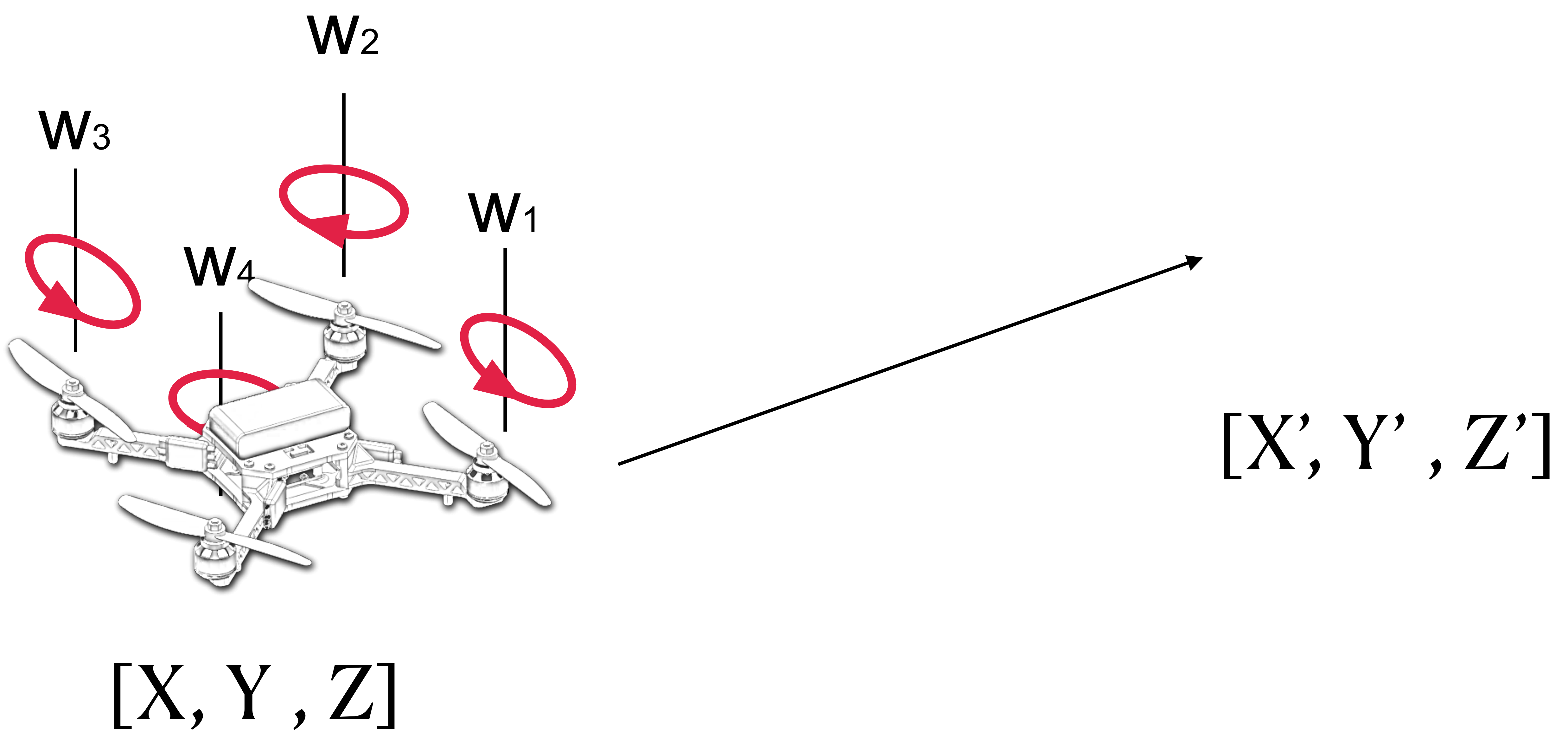
$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} \begin{bmatrix} cos(\phi)cos(\psi)sin(\theta) + sin(\phi)sin(\psi) \\ cos(\phi)sin(\theta)sin(\psi) + cos(\psi)sin(\phi) \\ sin(\theta)sin(\phi) \end{bmatrix} u_1$$



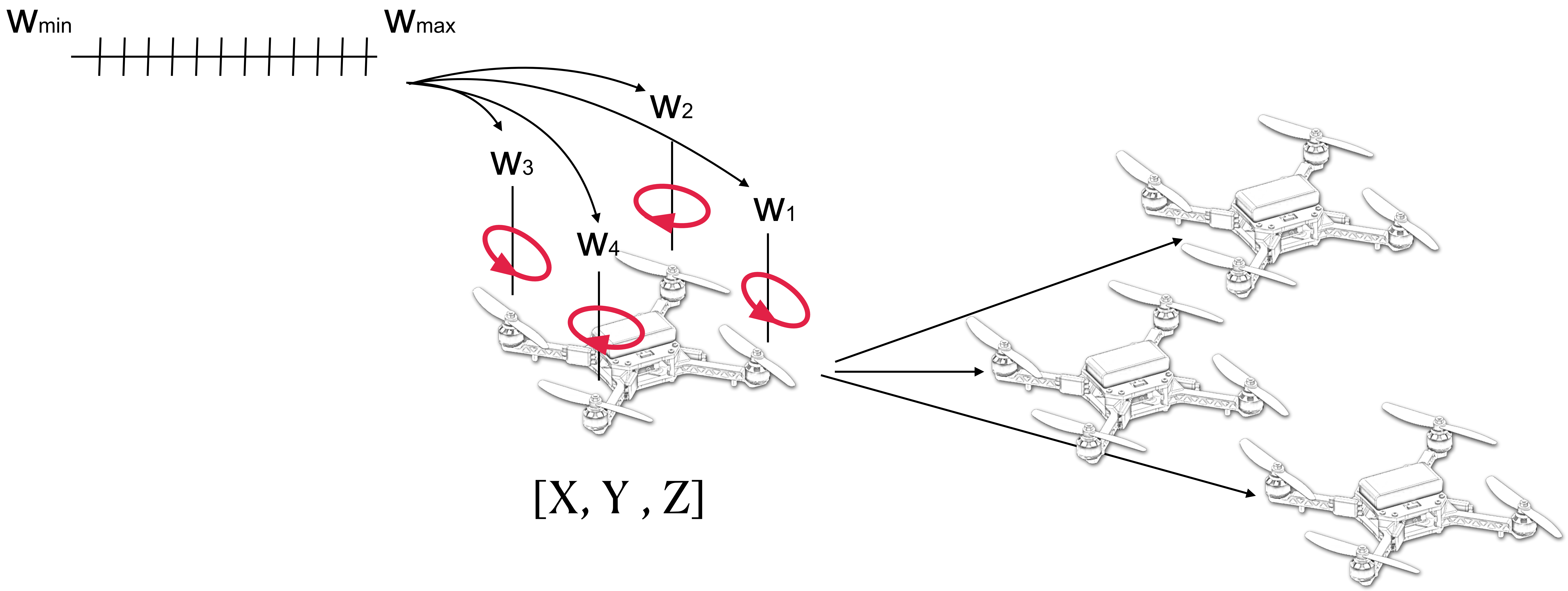
Kinematic and Dynamic Models

Using KD models we can compute the robots new position based on some input



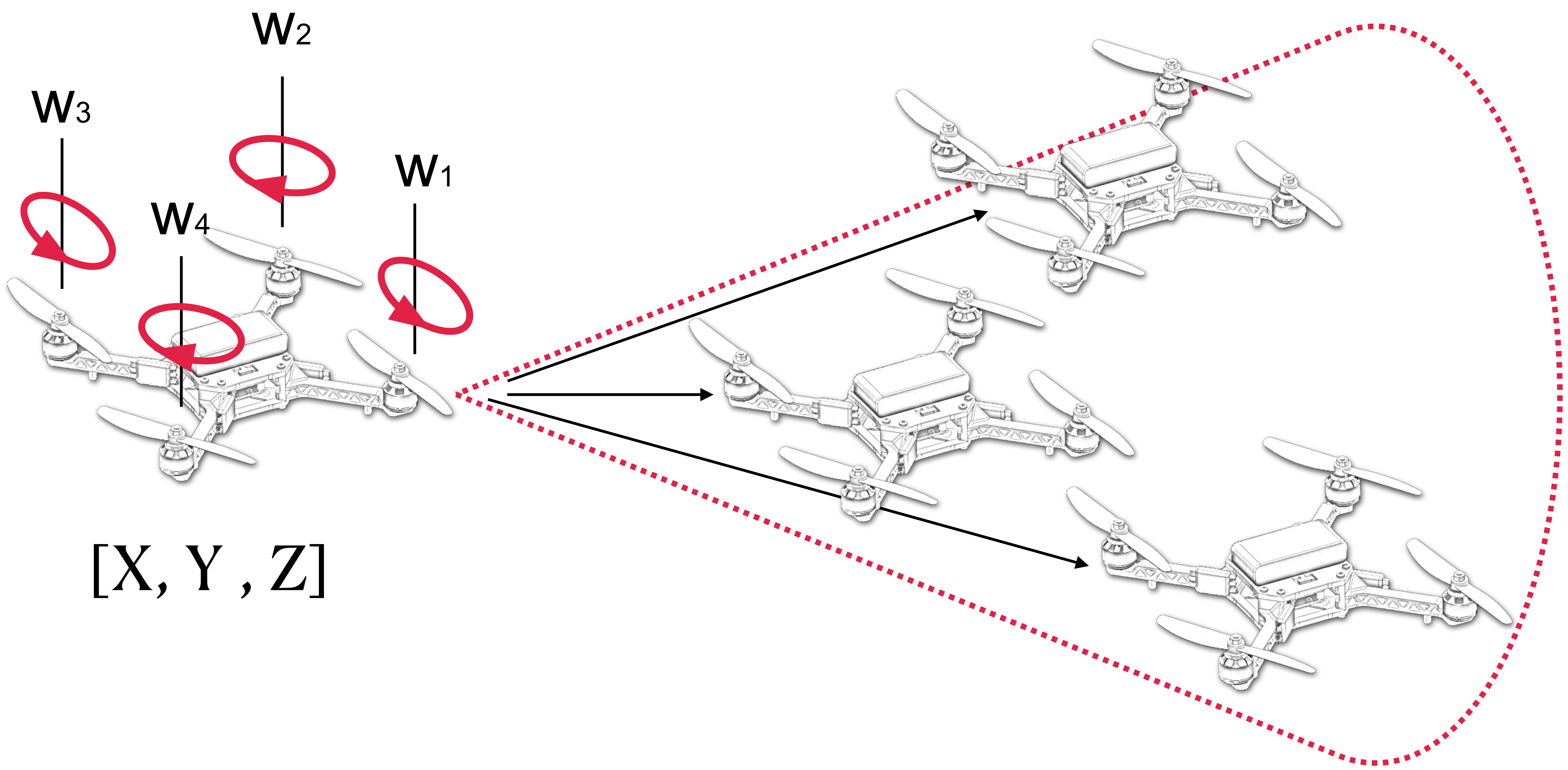
Reachable Sets

We can apply all permutations of input to determine all feasible future locations.



Reachable Sets

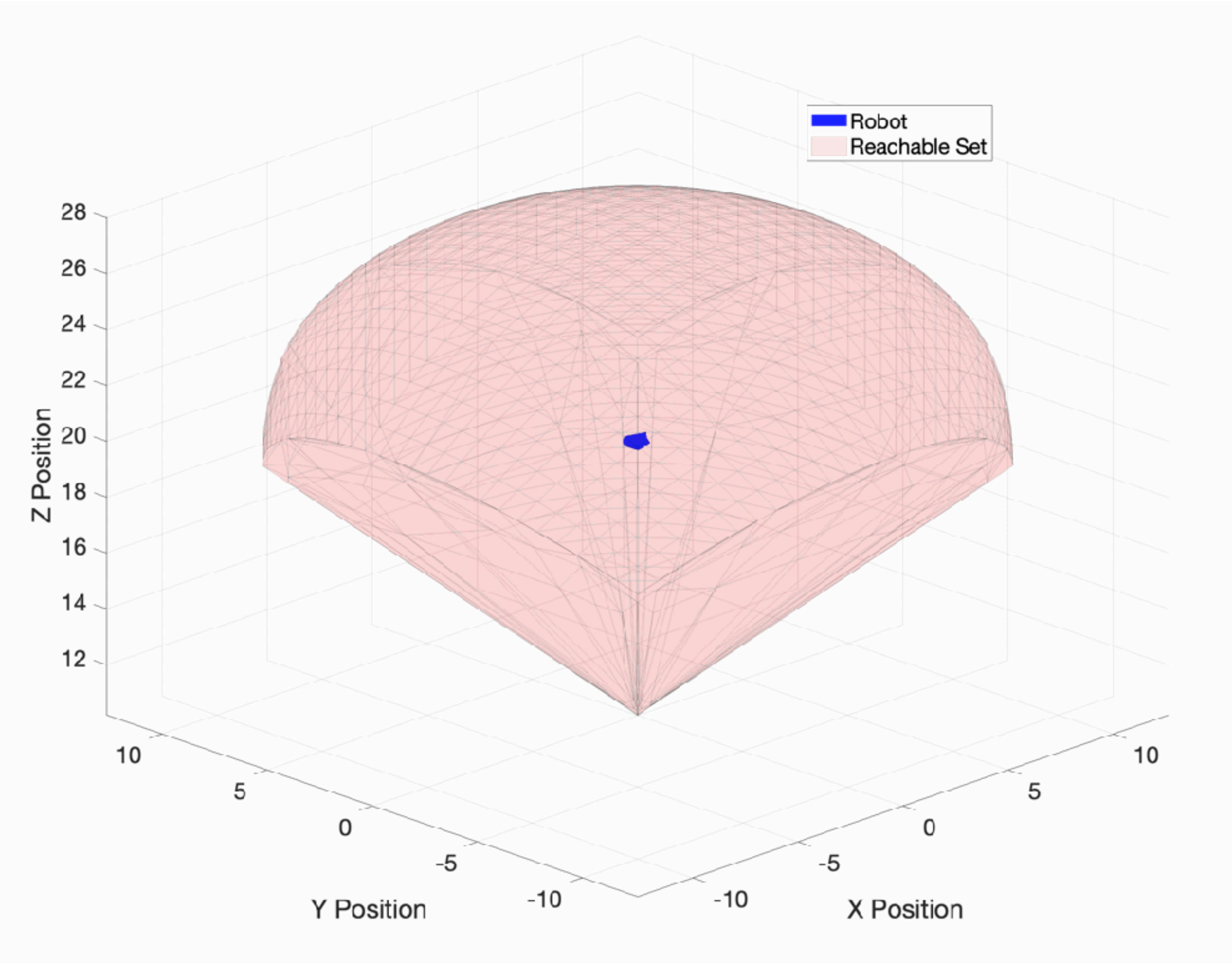
The area or volume covered by all future states is called the reachable set.



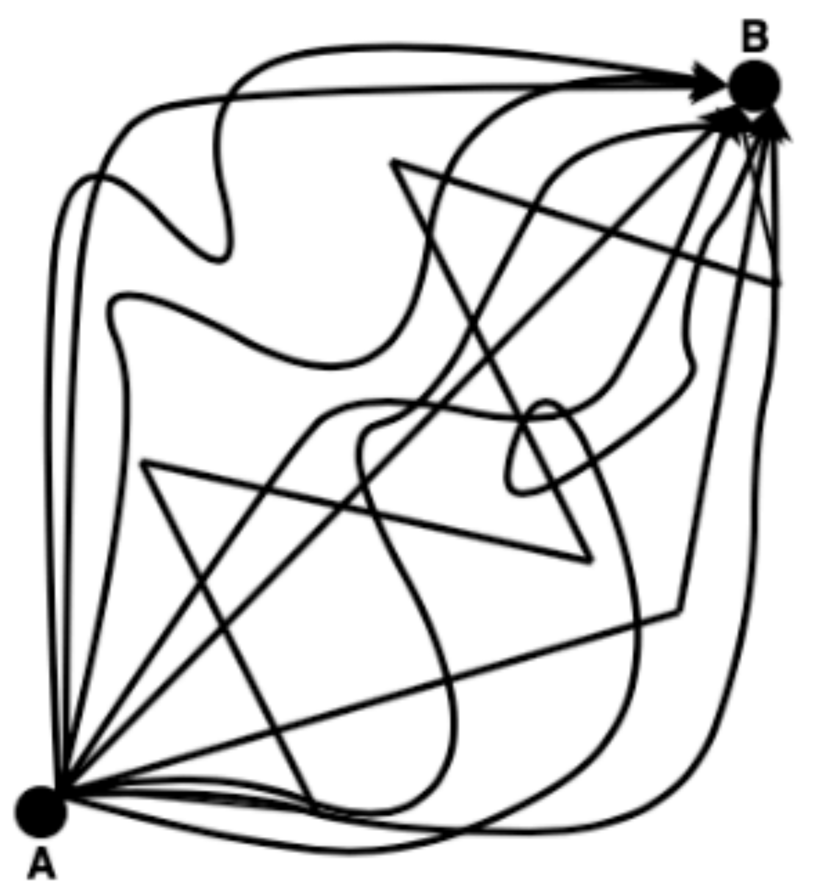
[X, Y , Z]

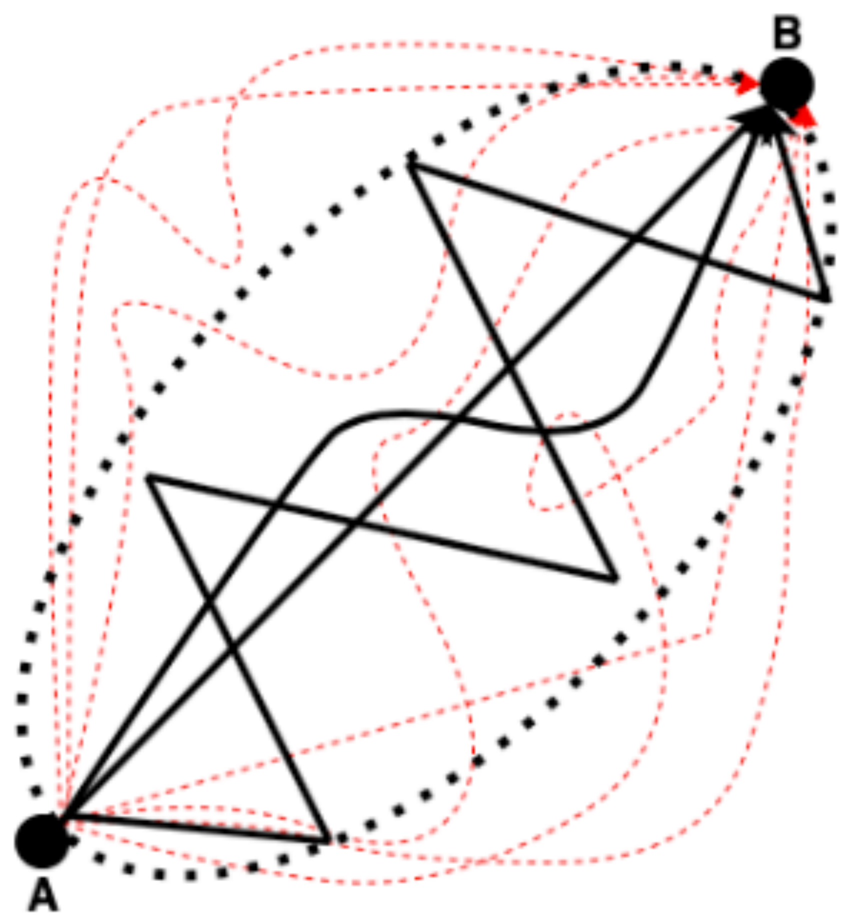
Reachable Sets

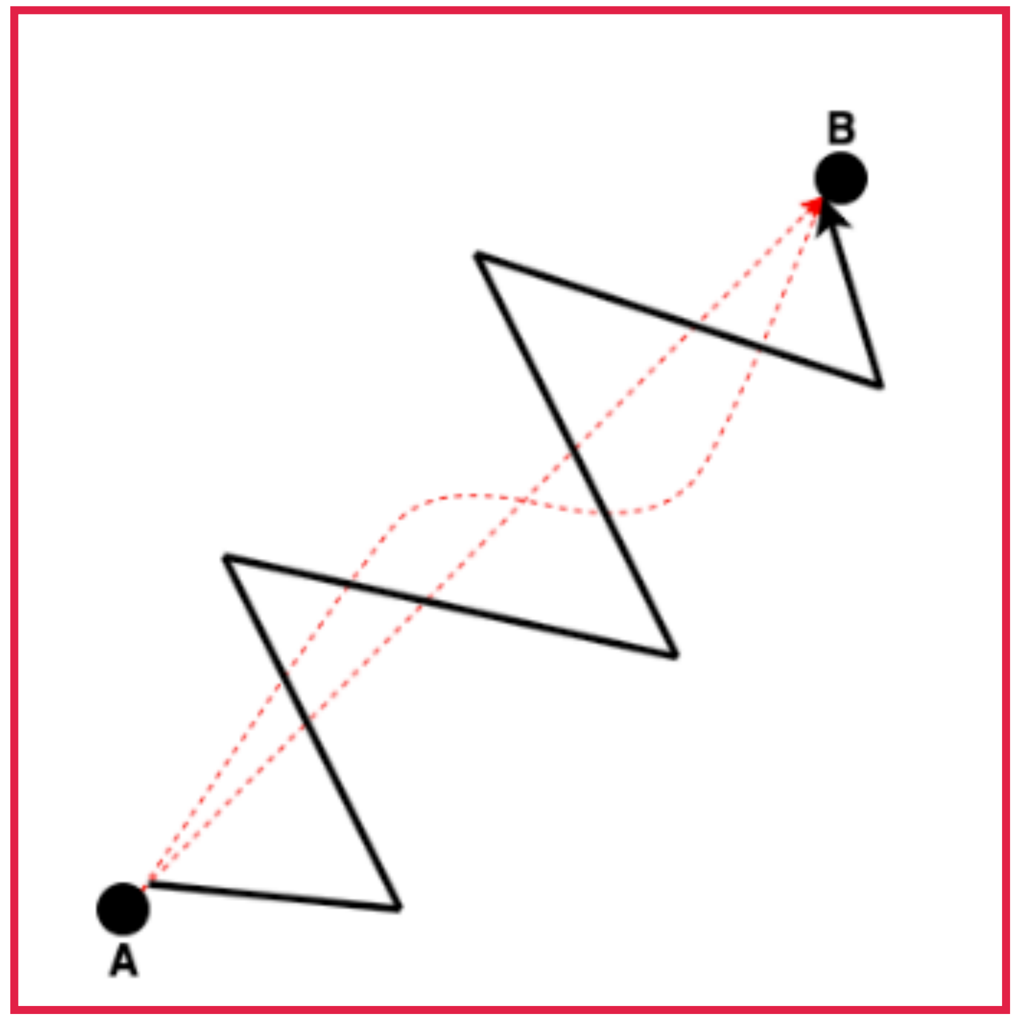
Reachable set of a quadrotor



Generating Trajectories

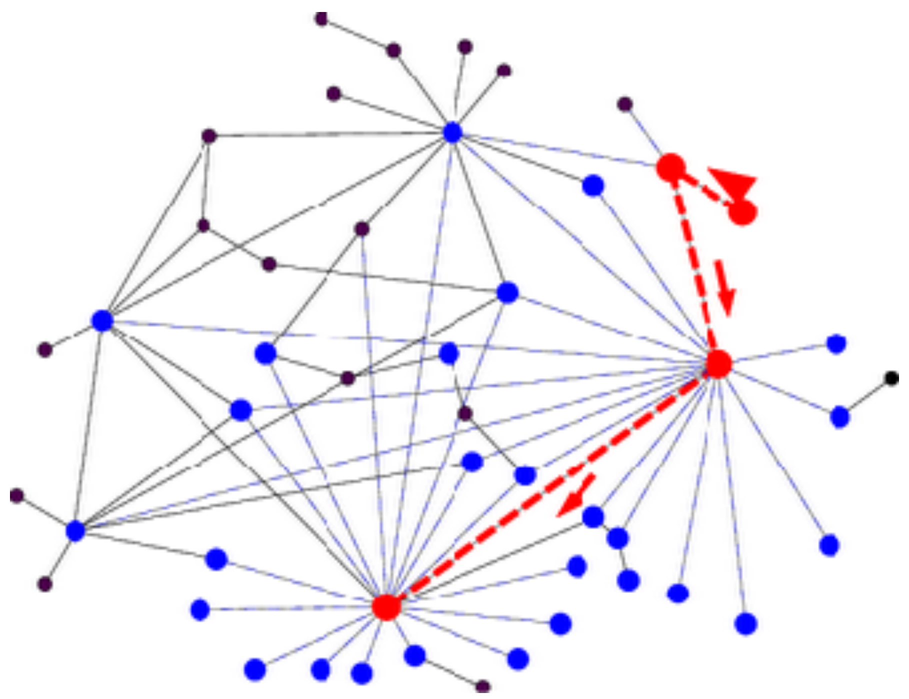




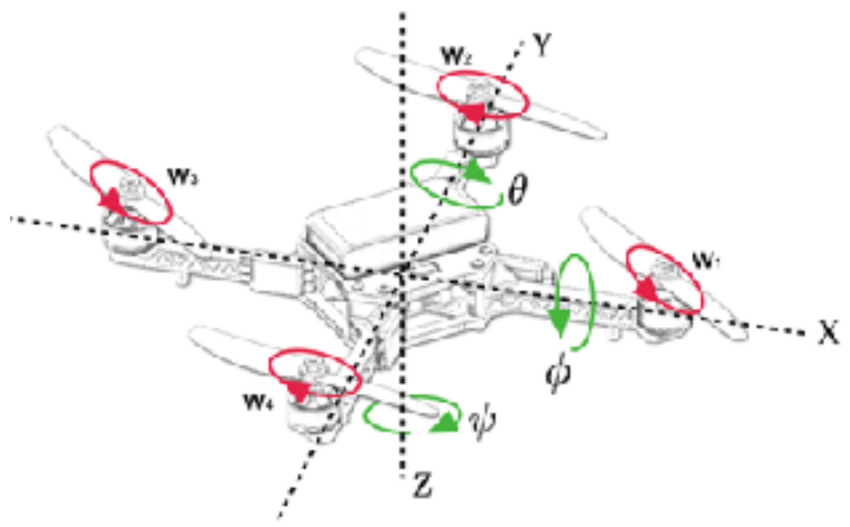


Search through world looking for all trajectories

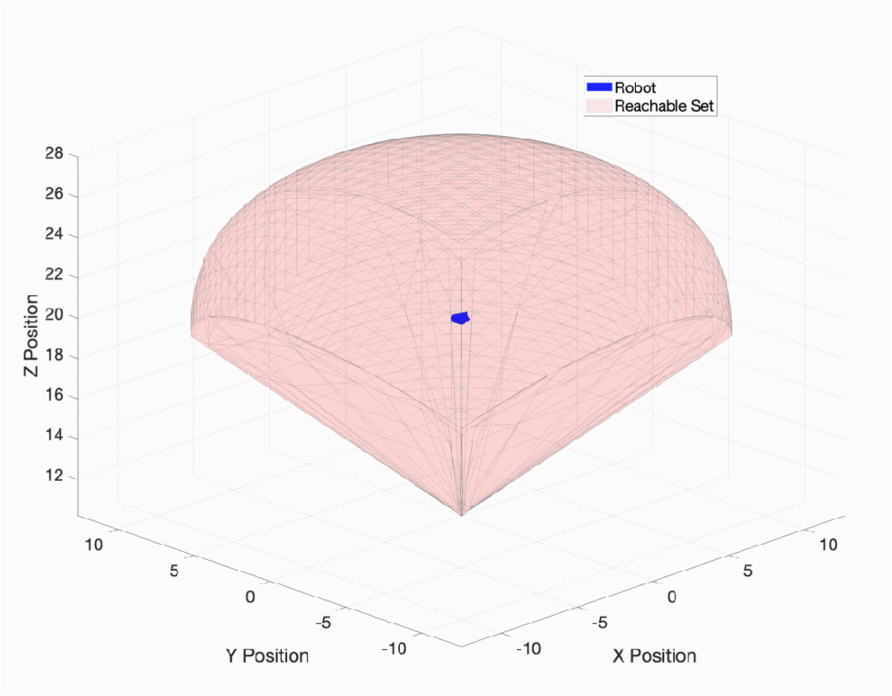
Use kinematic model and reachable set to find feasible trajectories



Graph search problem



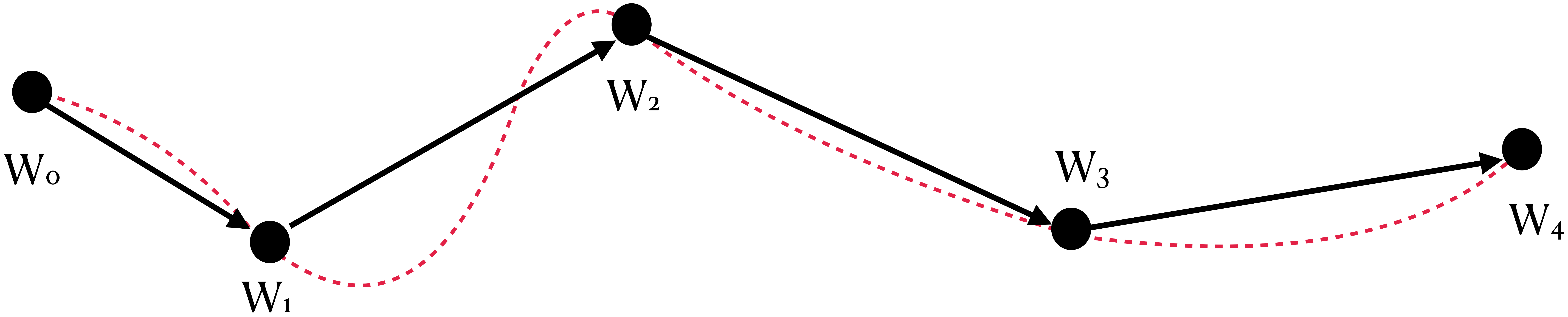
Kinematic and Dynamic Models



Reachable Set

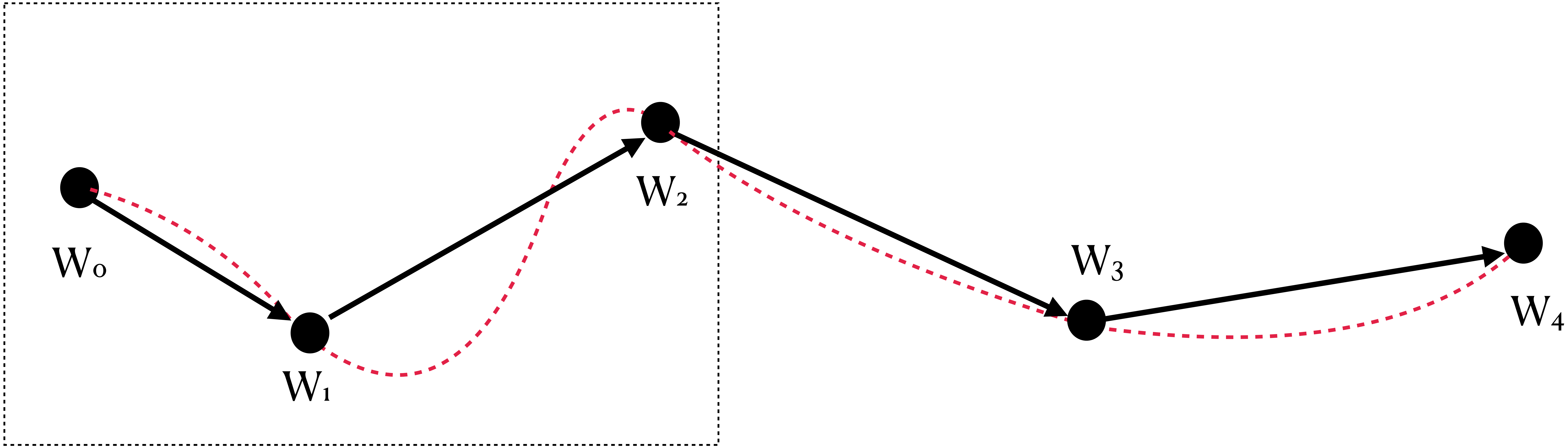
Stress Metrics

How to select trajectories which will **induce stress** in the robot:



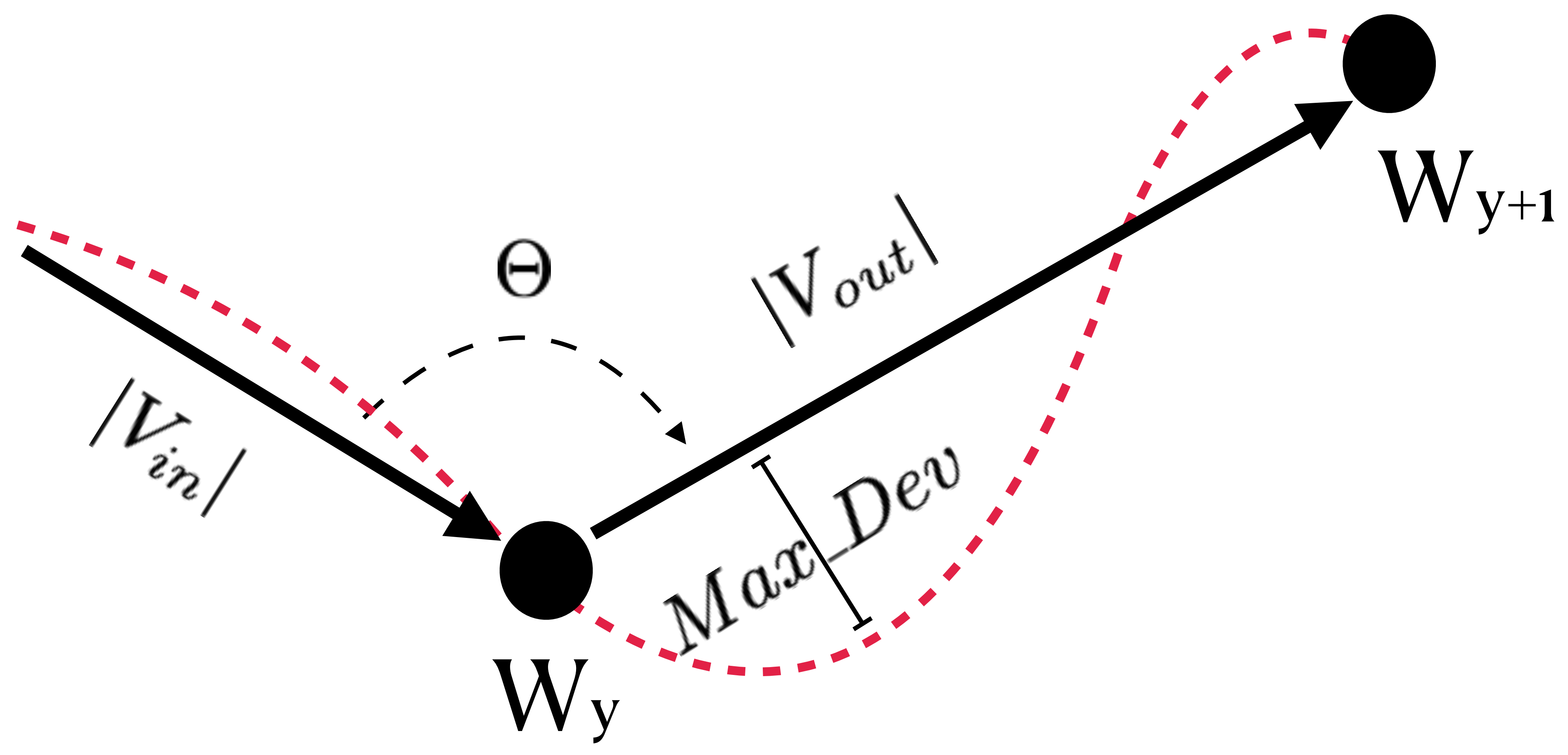
Stress Metrics

Focusing on a single segment



Stress Metrics

Assume we were interested in the maximum deviation



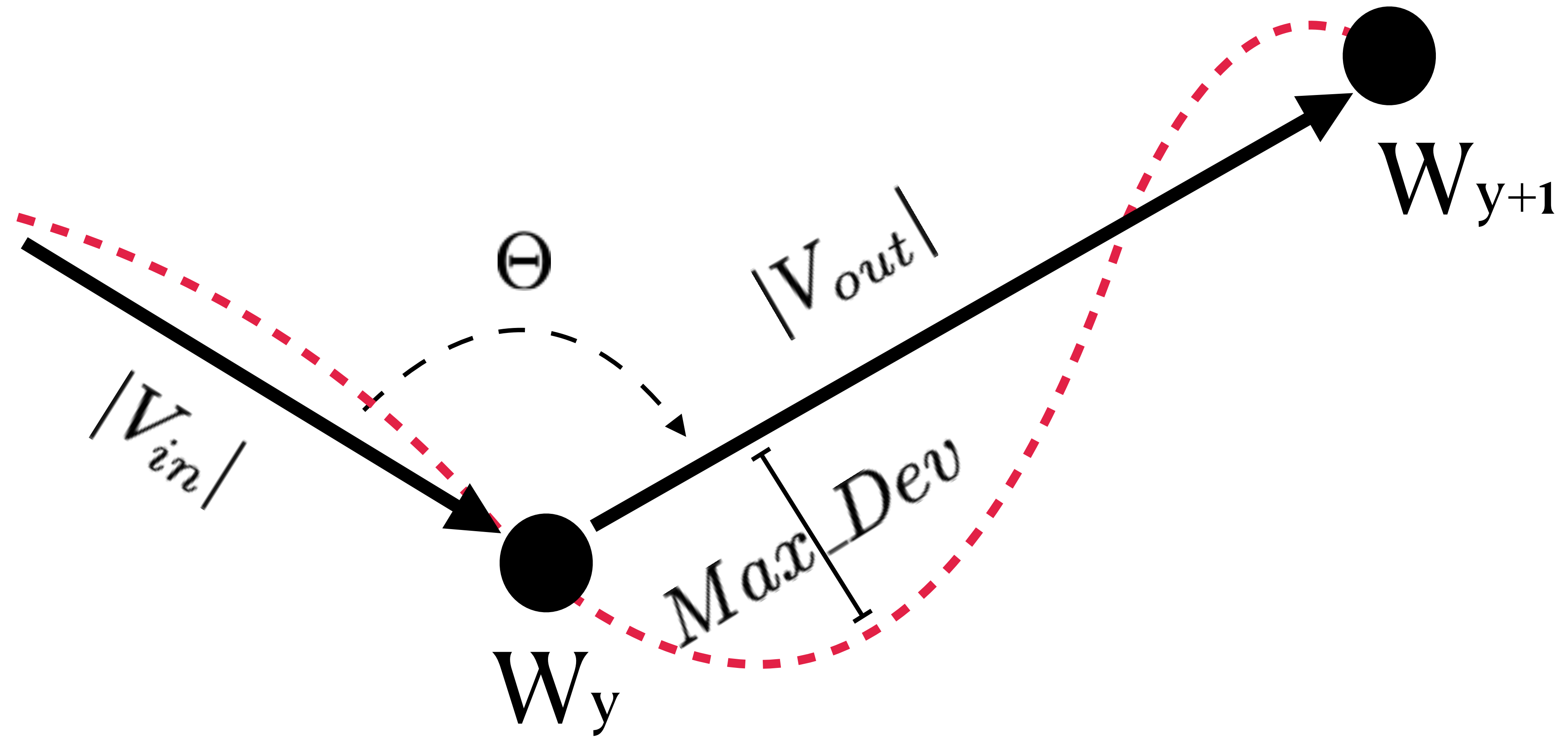
Stress Metrics

score

$$f(V_{in}, V_{out}, \Theta) = Max\_Dev$$

Handcrafted by experts

Learned

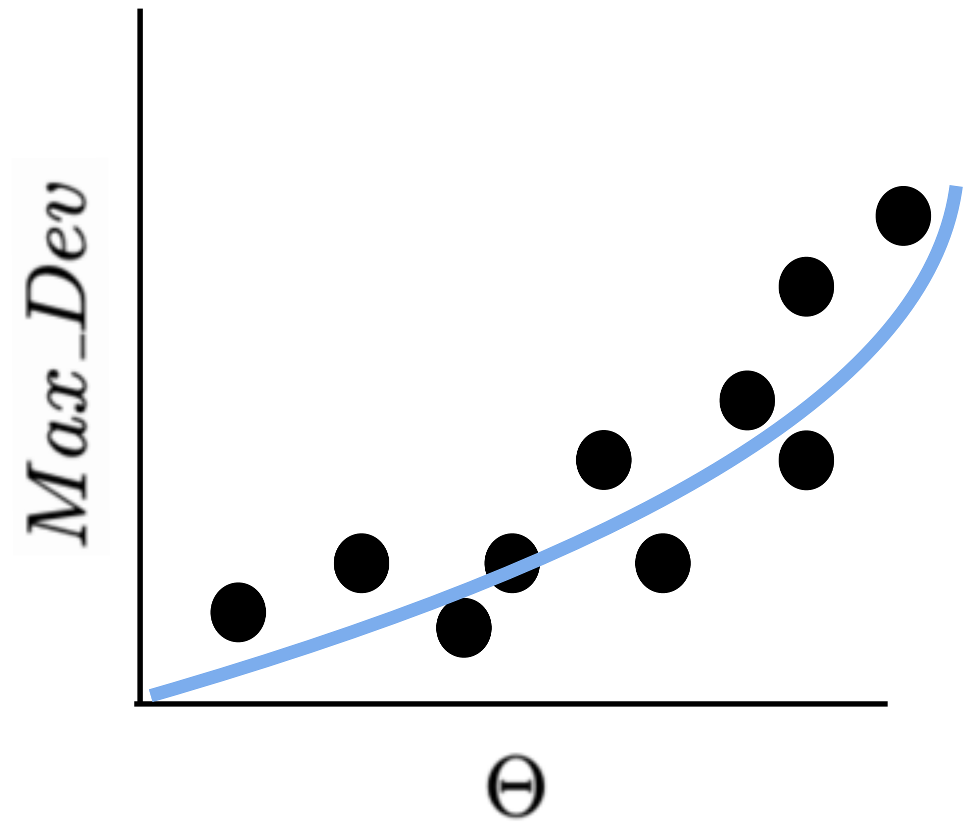


Stress Metrics

How could we learn this function?

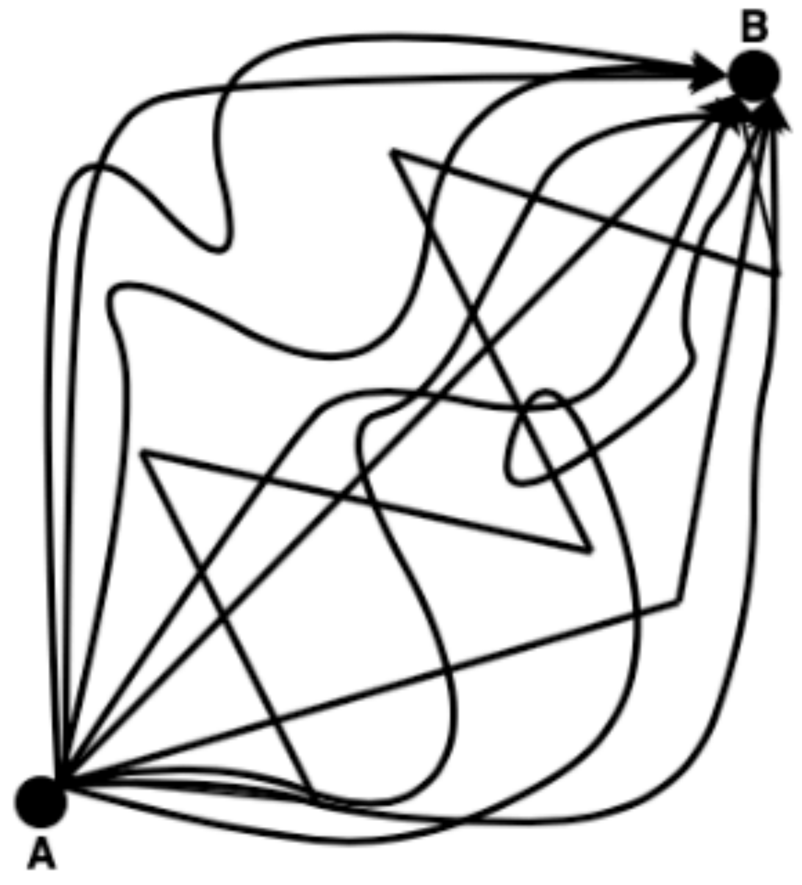
$$f(V_{in}, V_{out}, \Theta) = Max\_Dev$$

Generate Random Feasible Trajectories → Execute on Trajectories →

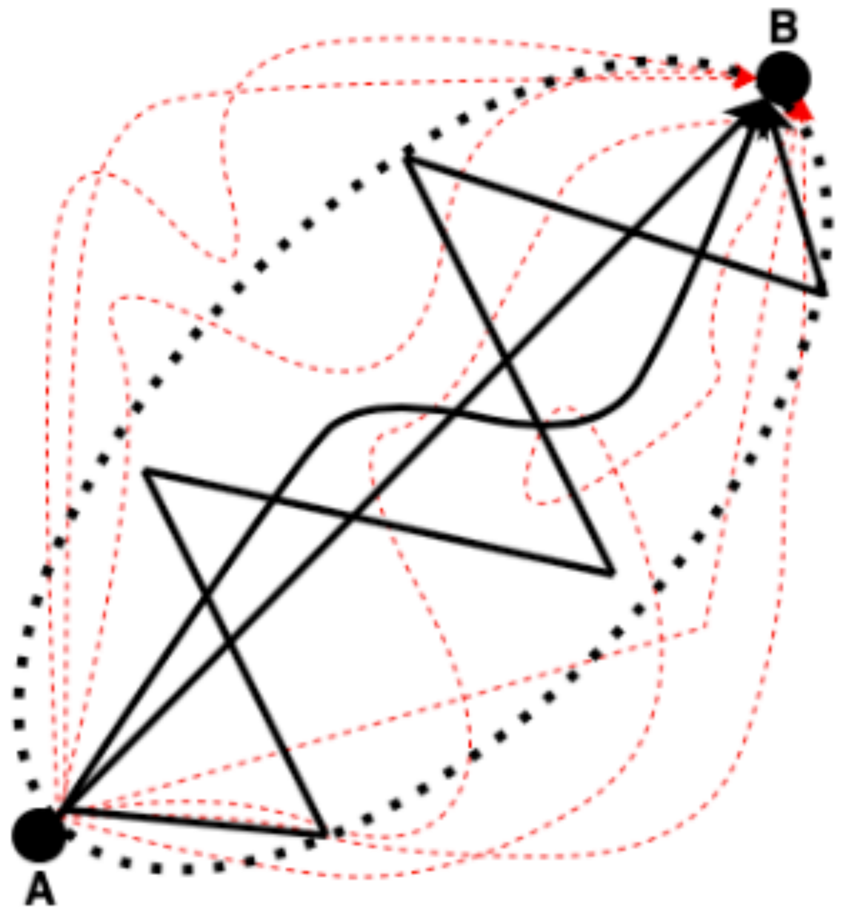


→ Generate New Stressful Trajectories

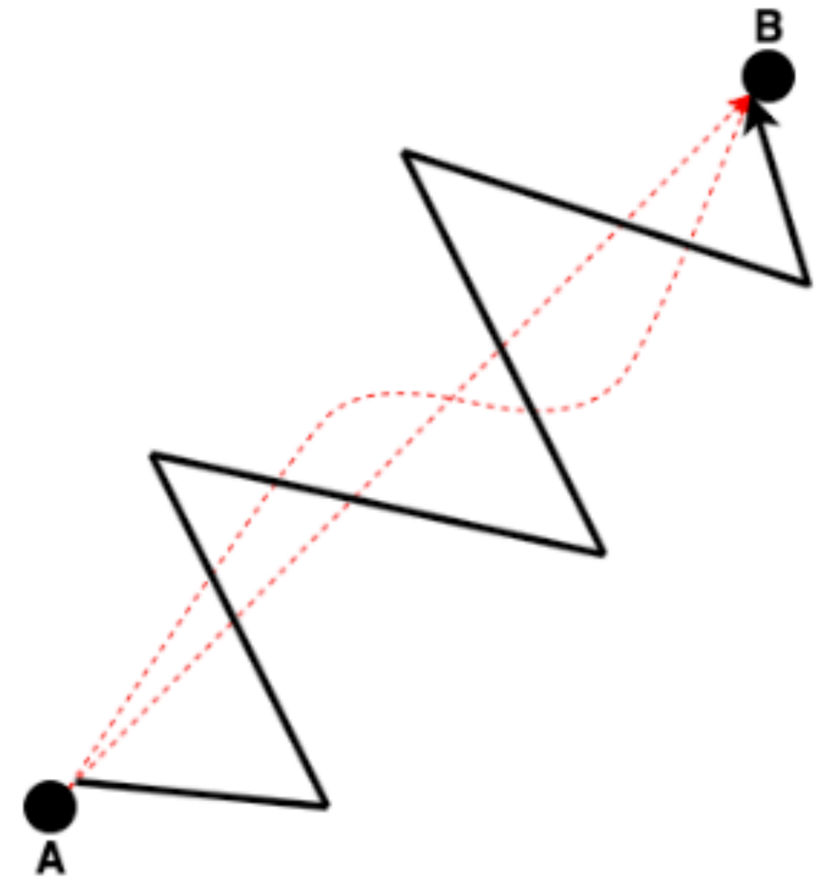
Generating Trajectories



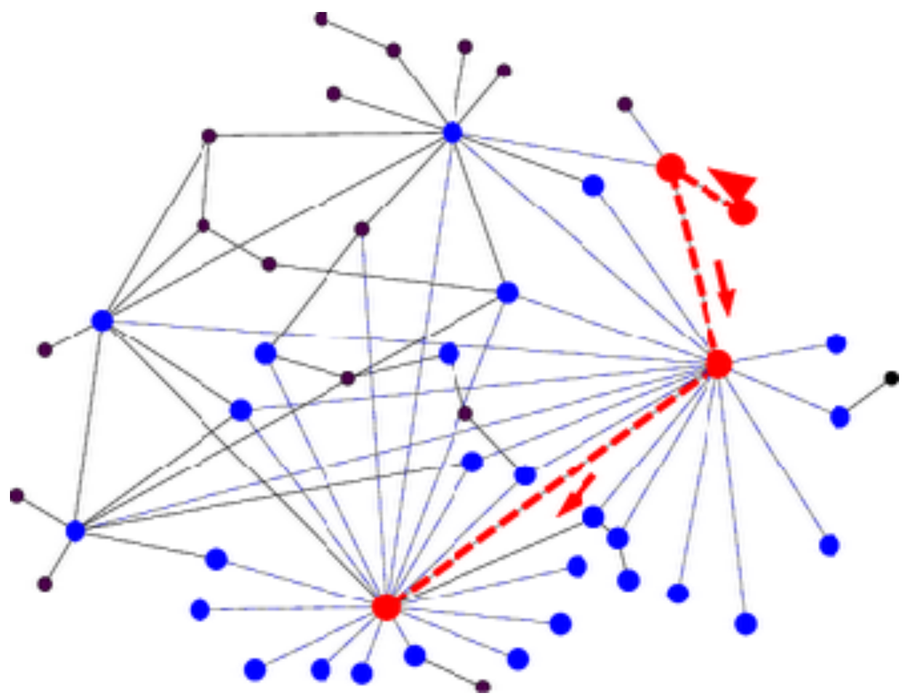
Search through world looking for all trajectories



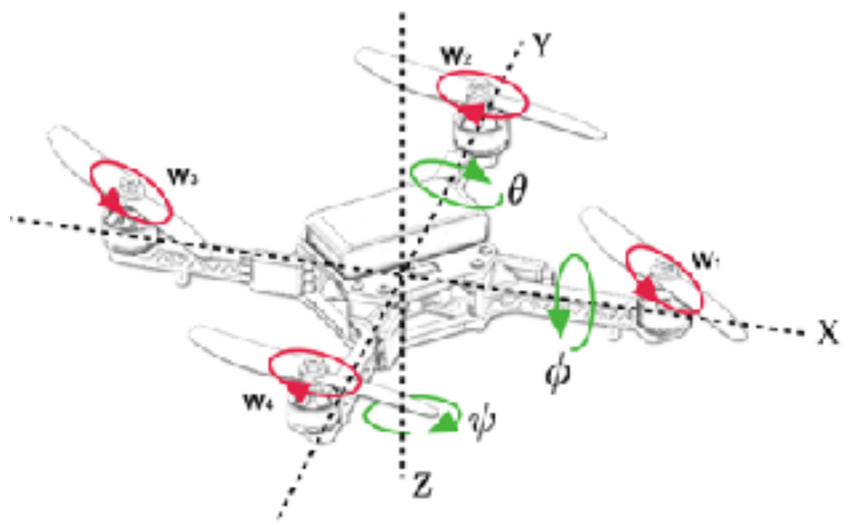
Use kinematic model and reachable set to find feasible trajectories



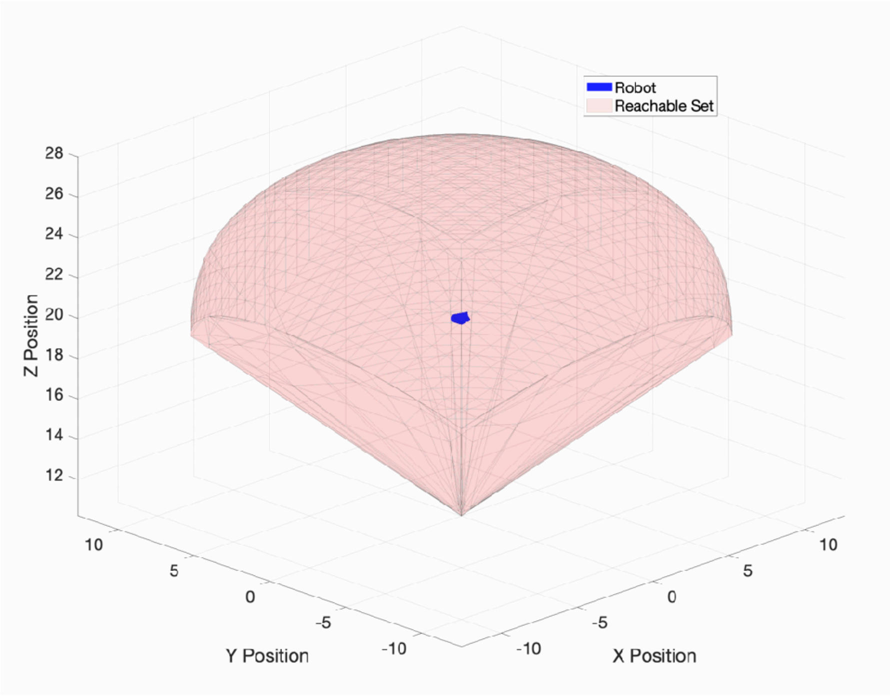
Using parametrizable scoring model, score trajectories based on predicted stress.



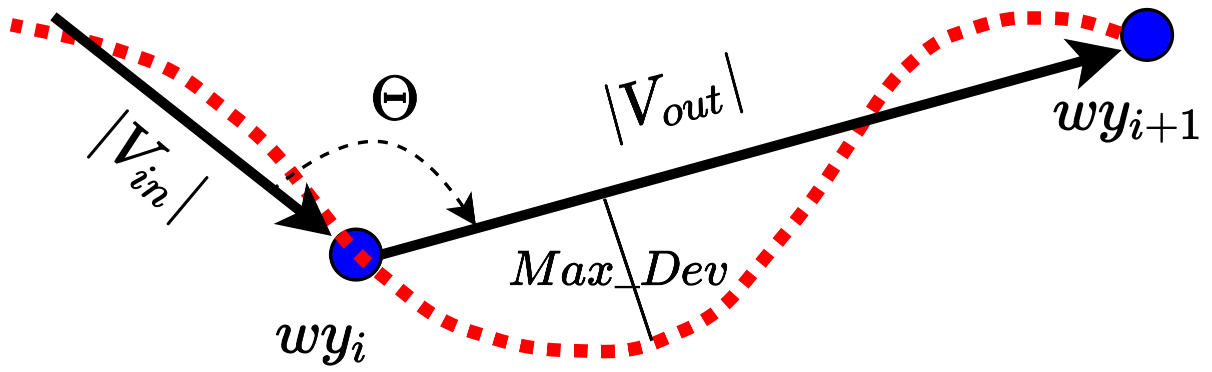
Graph search problem



Kinematic and Dynamic Models



Reachable Set



Scoring Model

Generating Trajectories

Algorithm 1: Trajectory Generation Manager

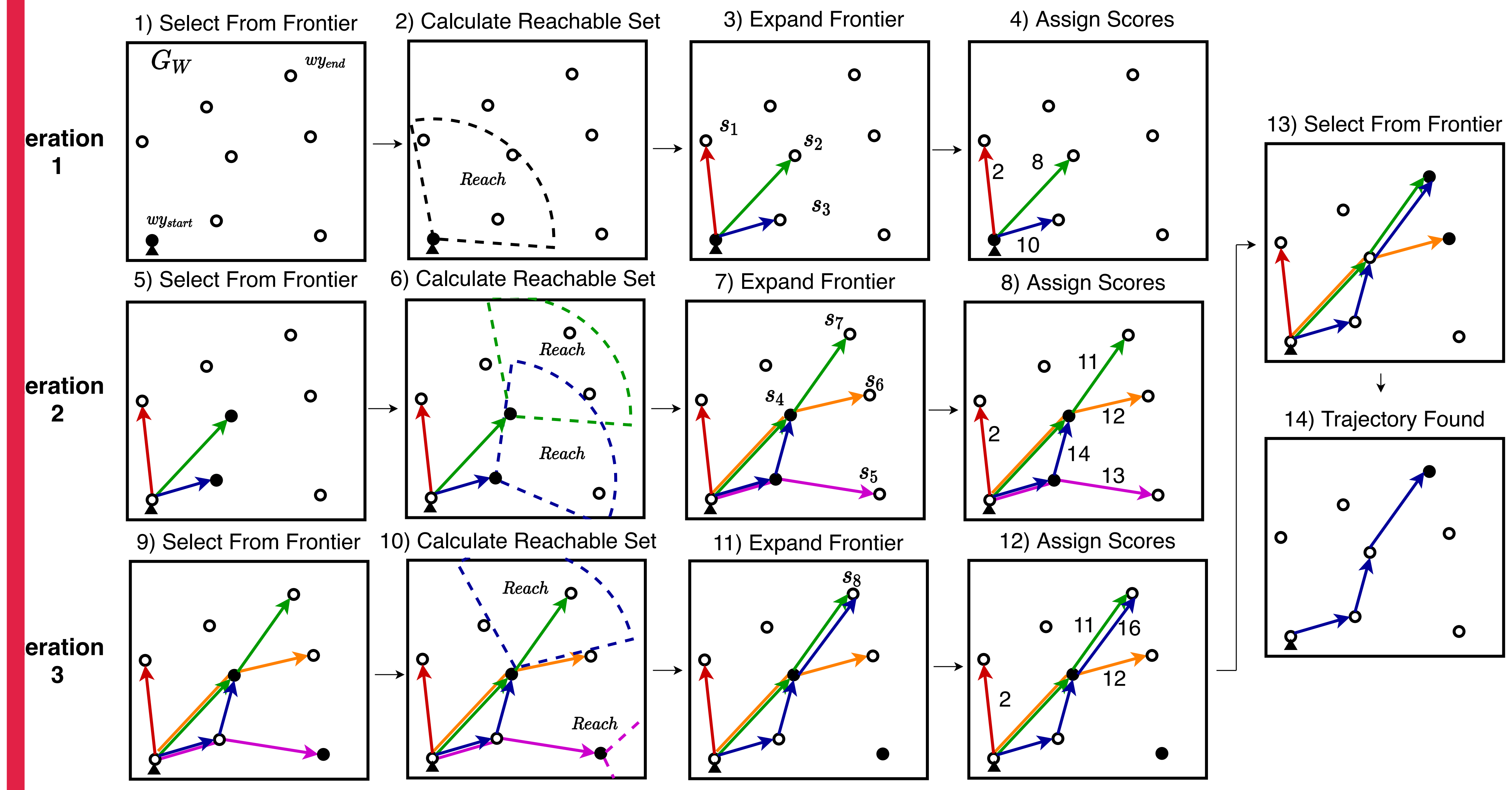
```
Input  : W, N_wy, wy_start, wy_end, N_traj, Limit, KD, Res,
         Width, ScoringModel
Output : Traj_s
Traj_s = ∅
while time < Limit do
    Wy = randomWpS
    G_W = graph(wy_star
    s_start = estimateRol
    traj_init = {s_start}
    Frontier = {(traj_init:
    Traj_c = ∅
    while Traj_c == ∅ a
        Frontier', Traj_c
        Frontier, Res, '
        Frontier = Fron
    end
    Traj_s = Traj_s ∪ Tra
end
return Traj_s
```

Algorithm 2: Explore Frontier

```
Function exploreFrontier(G_W, wy_end, KD, Frontier, Res,
    Width, N_traj, ScoringModel)
    Traj_c = ∅
    Frontier' = ∅
    SortedFrontier = sort(Frontier.scores)
    for i = 0; i < Width; i++ do
        // Select From Frontier
        traj = SortedFrontier[i].traj
        Frontier = Frontier ∩ not traj
        if |traj| == N_traj, and traj[N_traj].position == wy_end
         then
            Traj_c = Traj_c ∪ traj
        end
        if |traj| < N_traj then
            last_s = traj[last].state
            // Calculate Reachable Set
            Reach = calculateReachSet(last_s, KD, Res)
            for wy in (G_W ∩ Reach)) do
                new_s = estimateRobotState(last_s, wy)
                traj_n = traj ∪ new_s
                // Expand Frontier
                Frontier' = Frontier'
            end
        end
        // Assign Scores
        Frontier' = assignScores(Fro
    end
    return Frontier', Traj_c
```

Algorithm 3: Assign Scores

```
Function assignScores(Frontier, ScoringModel)
    for traj in Frontier do
        score = 0
        for each pairOfStates in traj do
            score += scoringModel(pairOfStates)
        end
        traj.score = score
    end
    return Frontier
```



Tool

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

https://hildebrandt-carl.github.io/RobotTestGenerationArtifact/

Evaluation

Our study aimed to answer two questions:

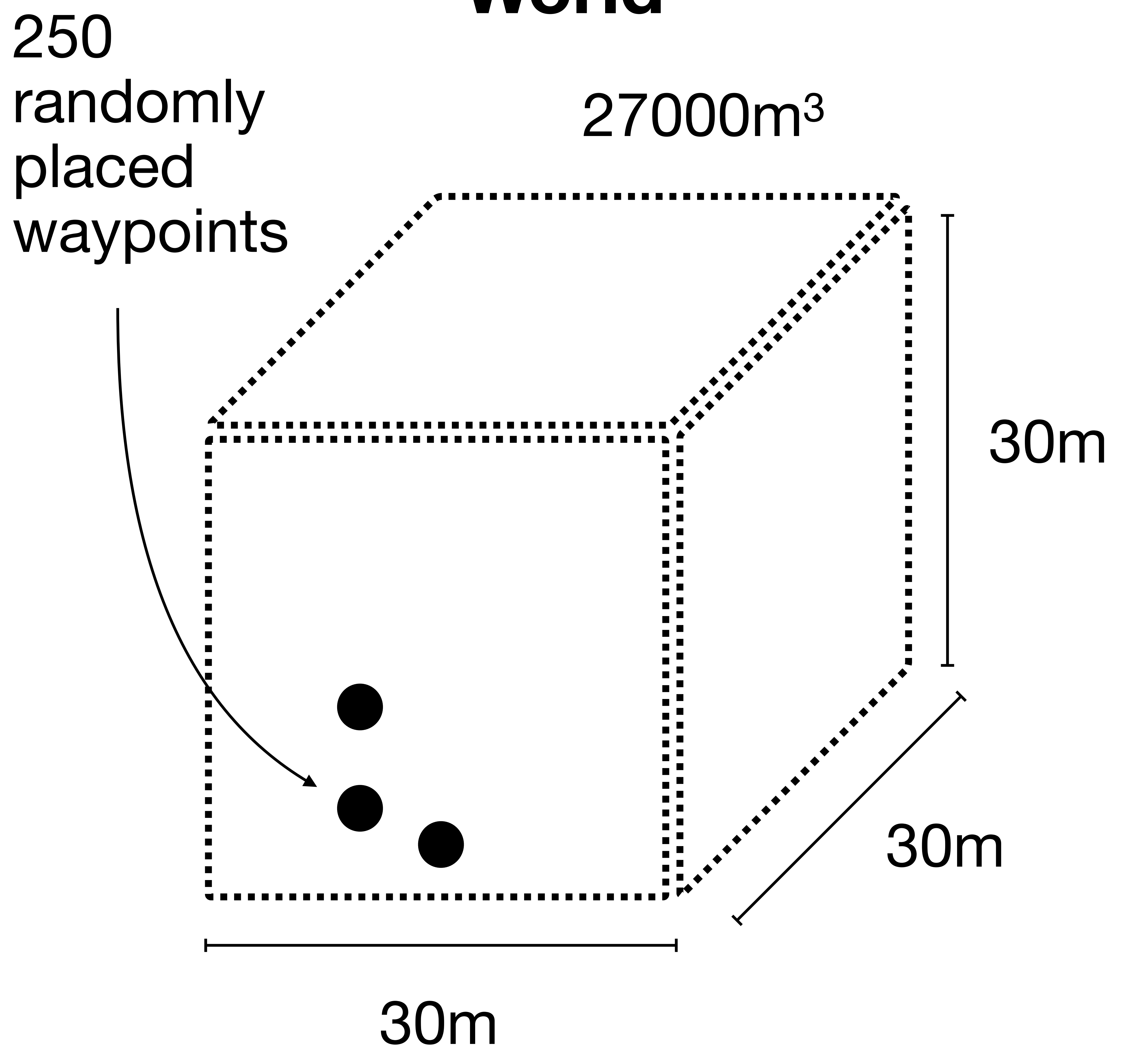
RQ1) Does the introduction of the kinematic and dynamic models improve the ability to generate feasible and valid trajectories?

RQ2) Does the introduction of a scoring model improve the ability to generate stressful trajectories?

Evaluation Setup

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

World



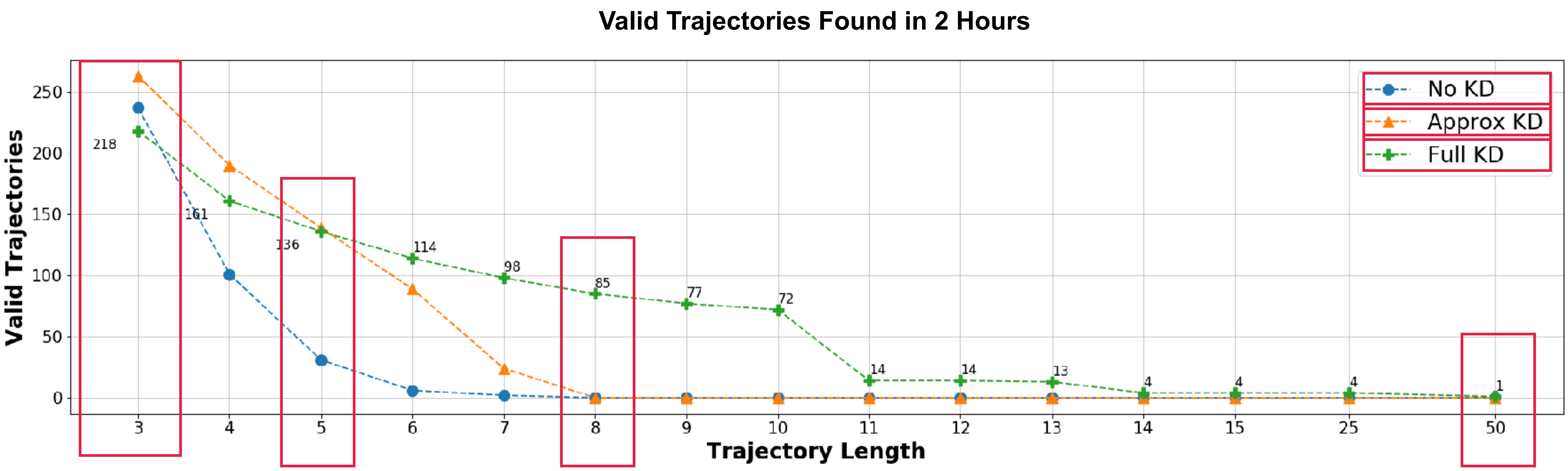
Robots

Robot Hardware	Robot Software	Execution
Flightgoggles Quadrotor[23]	Unstable Waypoint Controller[66]	Simulation
	Stable Waypoint Controller[66]	Simulation
	Fixed Velocity Controller	Simulation
	Minimum Snap Controller[42]	Simulation
Parrot Anafi Quadrotor [48]	Waypoint Controller[50]	Simulation
		Real World

Mit Drone Figure) http://news.mit.edu/2018/virtual-reality-testing-ground-drones-0517
Parrot Drone Figure) https://www.parrot.com/us/

RQ1 Answer

RQ1) Does the introduction of the kinematic and dynamic models improve the ability to generate feasible and valid trajectories?



RQ1 Answer

Takeaway: Using the kinematic and dynamic models improves the ability to find physically feasible trajectories.

RQ2 Answer

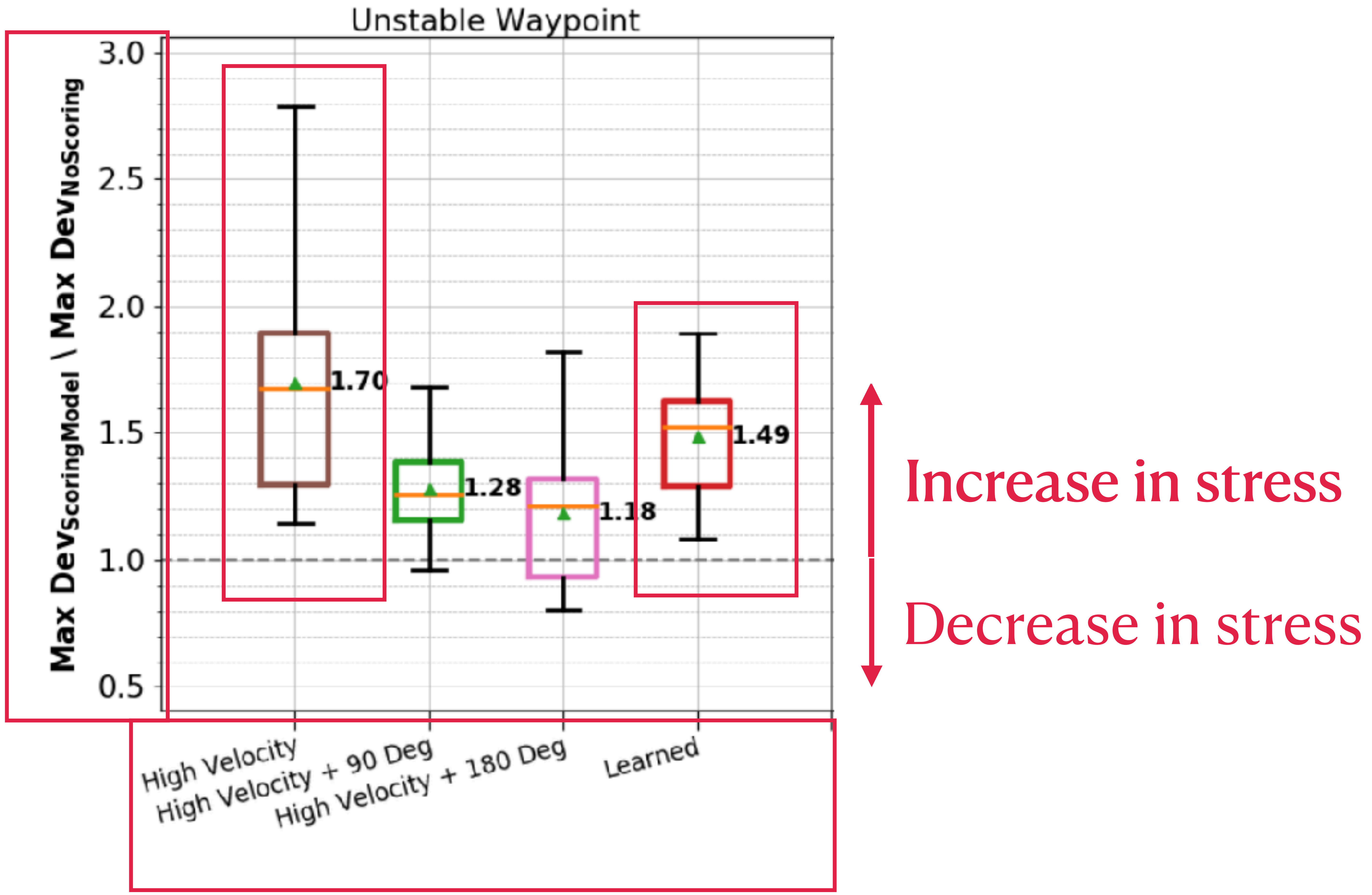
RQ2) Does the introduction of a scoring model improve the ability to generate stressful trajectories?

High Velocity	Assigns high scores to trajectories with high velocities.
High Velocity + 90 Deg	Assigns high scores to trajectories with high velocities and include 90 degree turns.
High Velocity + 180 Deg	Assigns high scores to trajectories with high velocities and include 180 degree turns
Learned	Learns a scoring model based on the execution of prior trajectories

RQ2 Answer

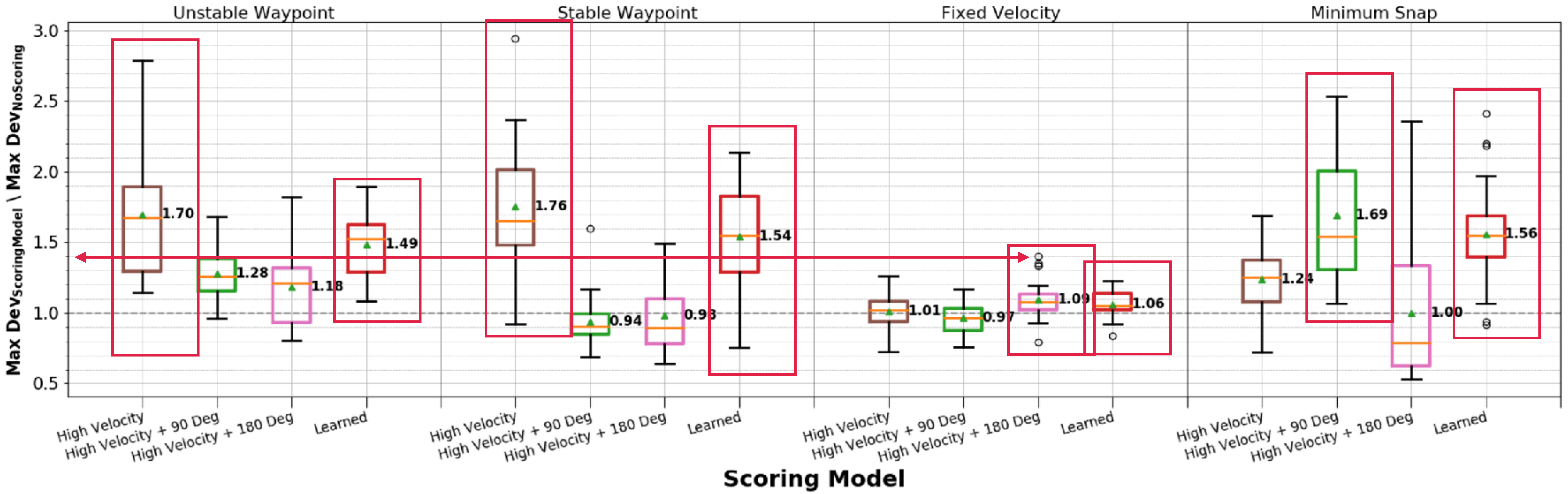
UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

RQ2) Does the introduction of a scoring model improve the ability to generate stressful trajectories?



RQ2 Answer

RQ2) Does the introduction of a scoring model improve the ability to generate stressful trajectories?



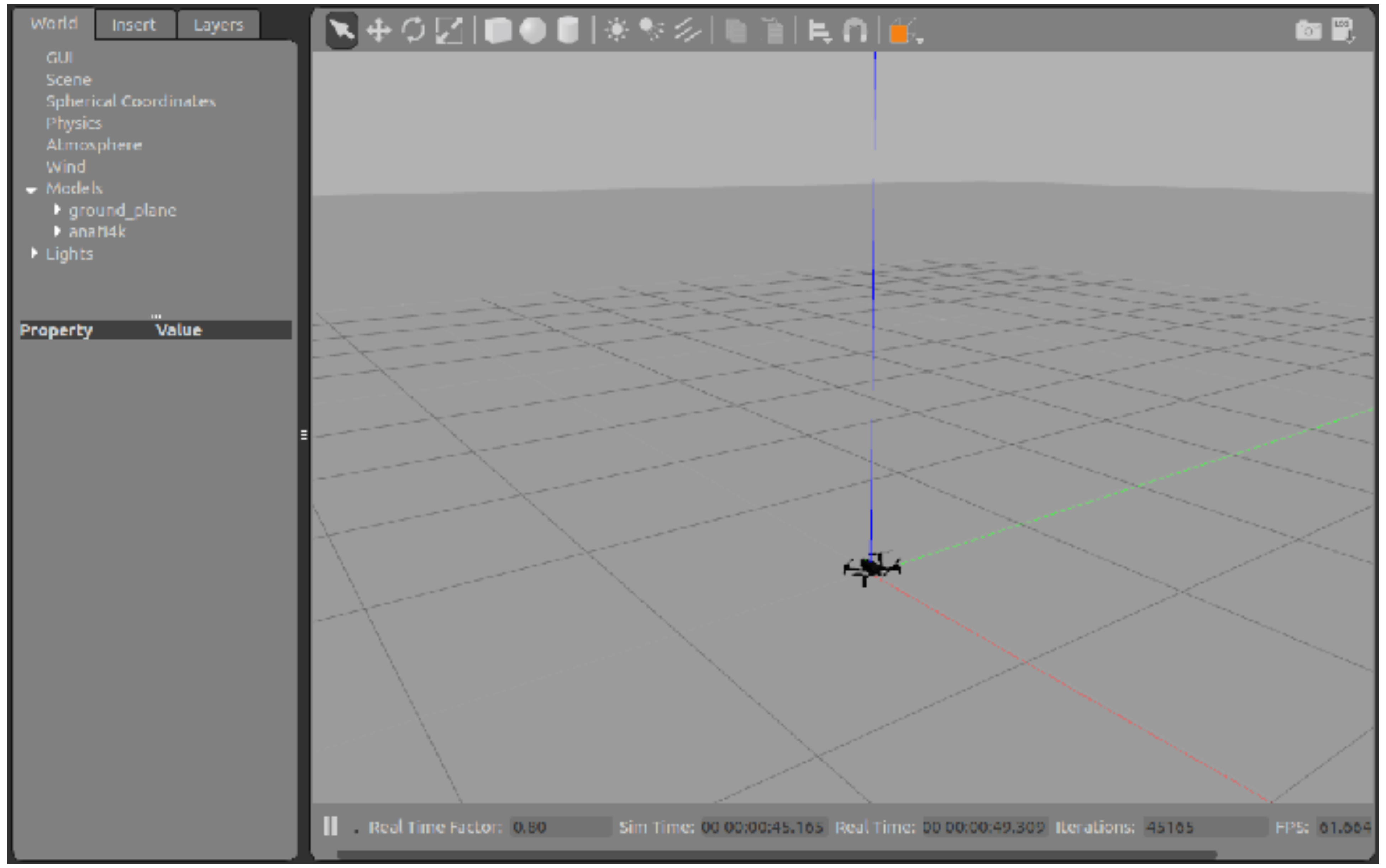
RQ2 Answer

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE

Takeaway: Introducing both handcrafted and learned scoring model into trajectory generation produces test **that on average are 55.9% and 41.3% more stressful** than trajectories without a scoring model respectively.

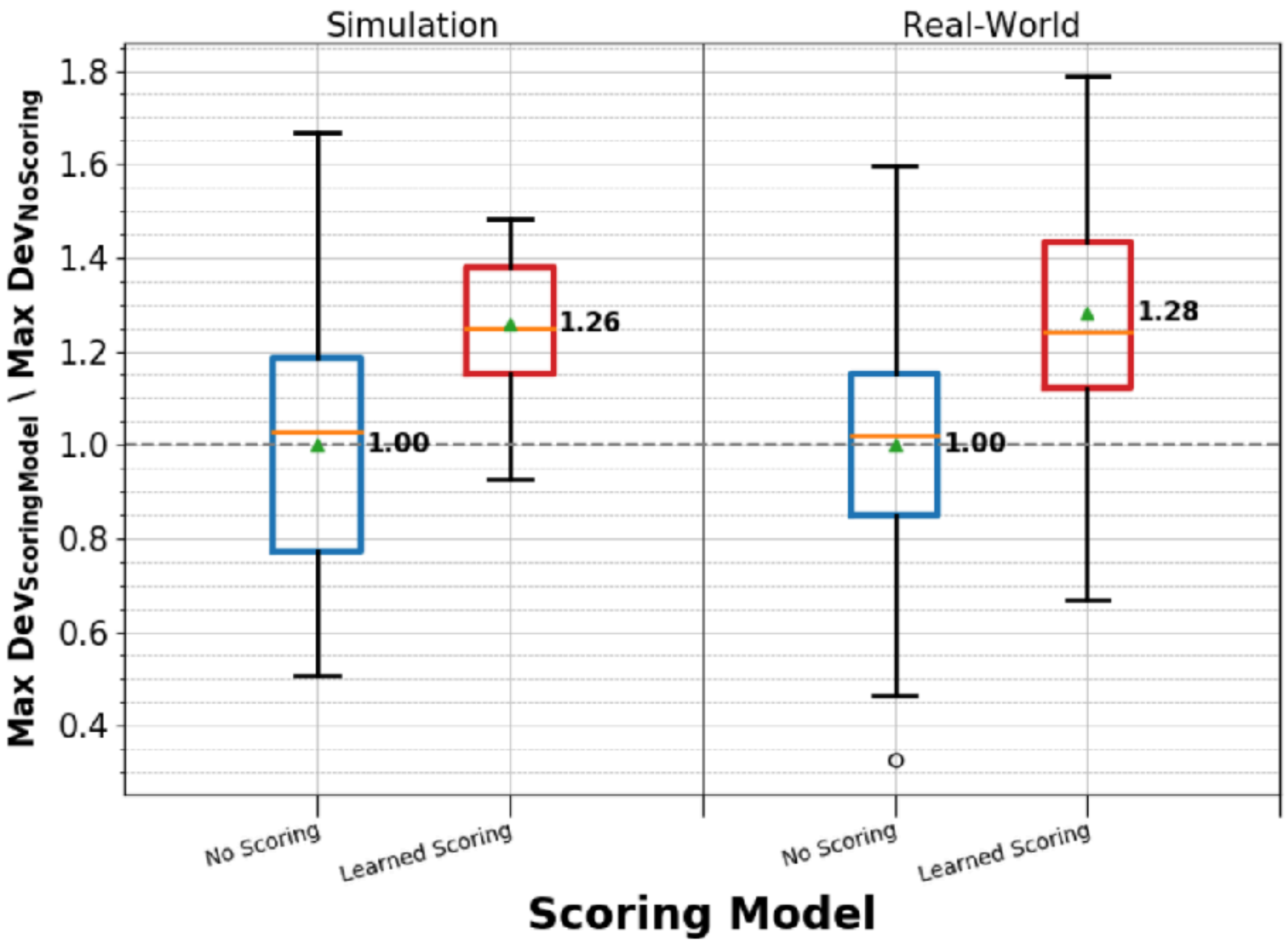
Study

Performed a study on a commercial quadrotor in the real world.



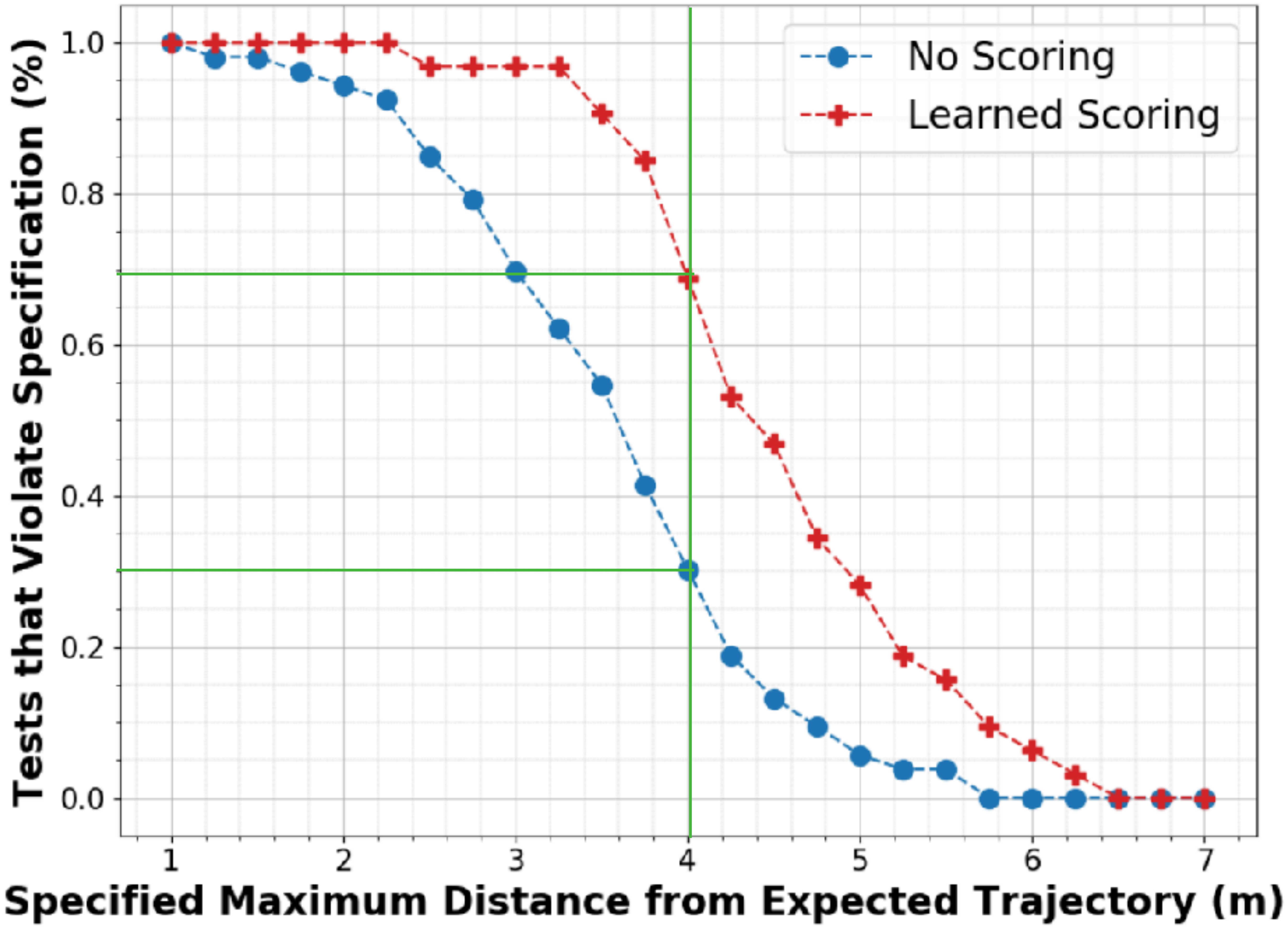
Study

Performed a study on a commercial quadrotor in the real world.



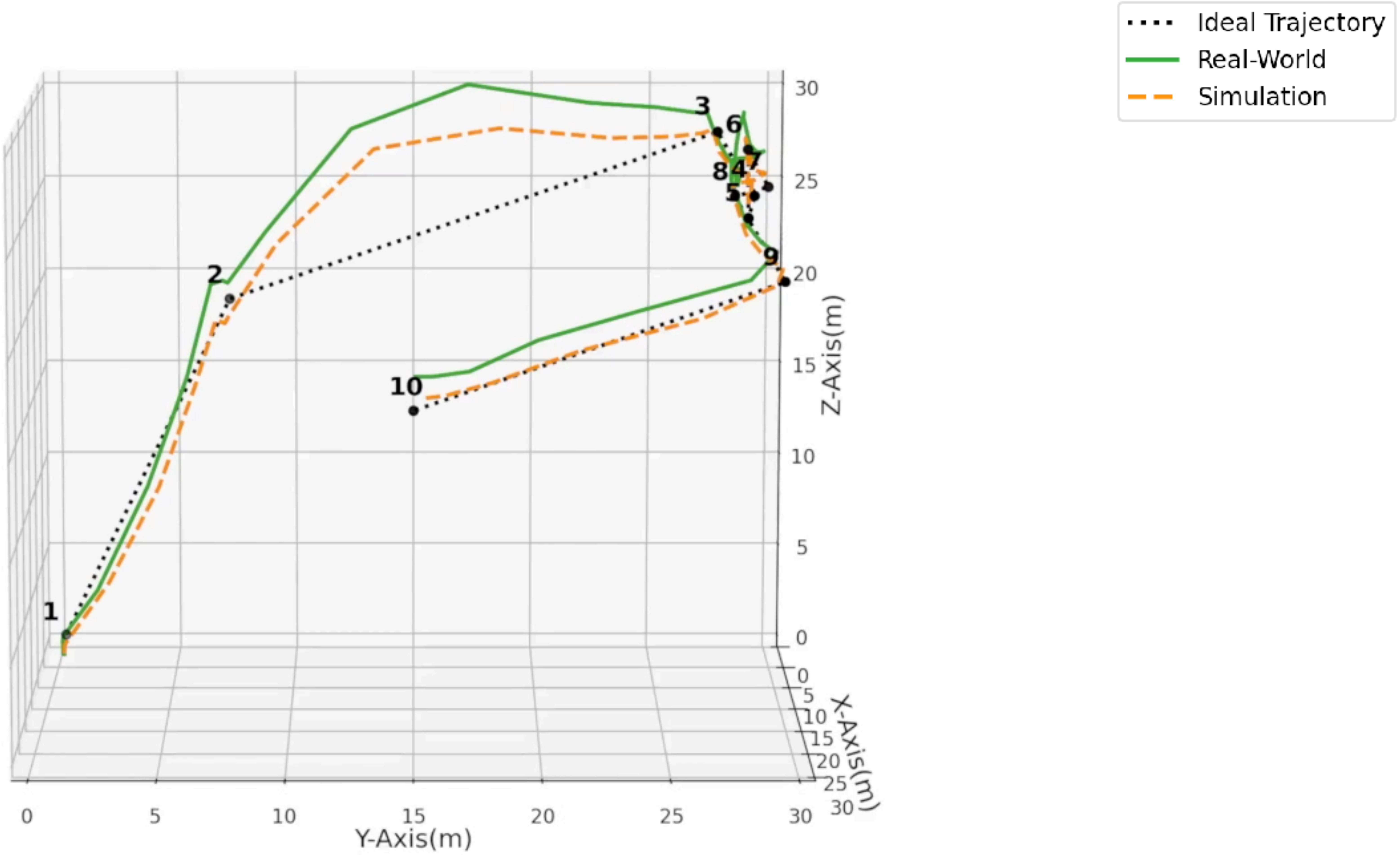
Study

Performed a study on a commercial quadrotor in the real world.



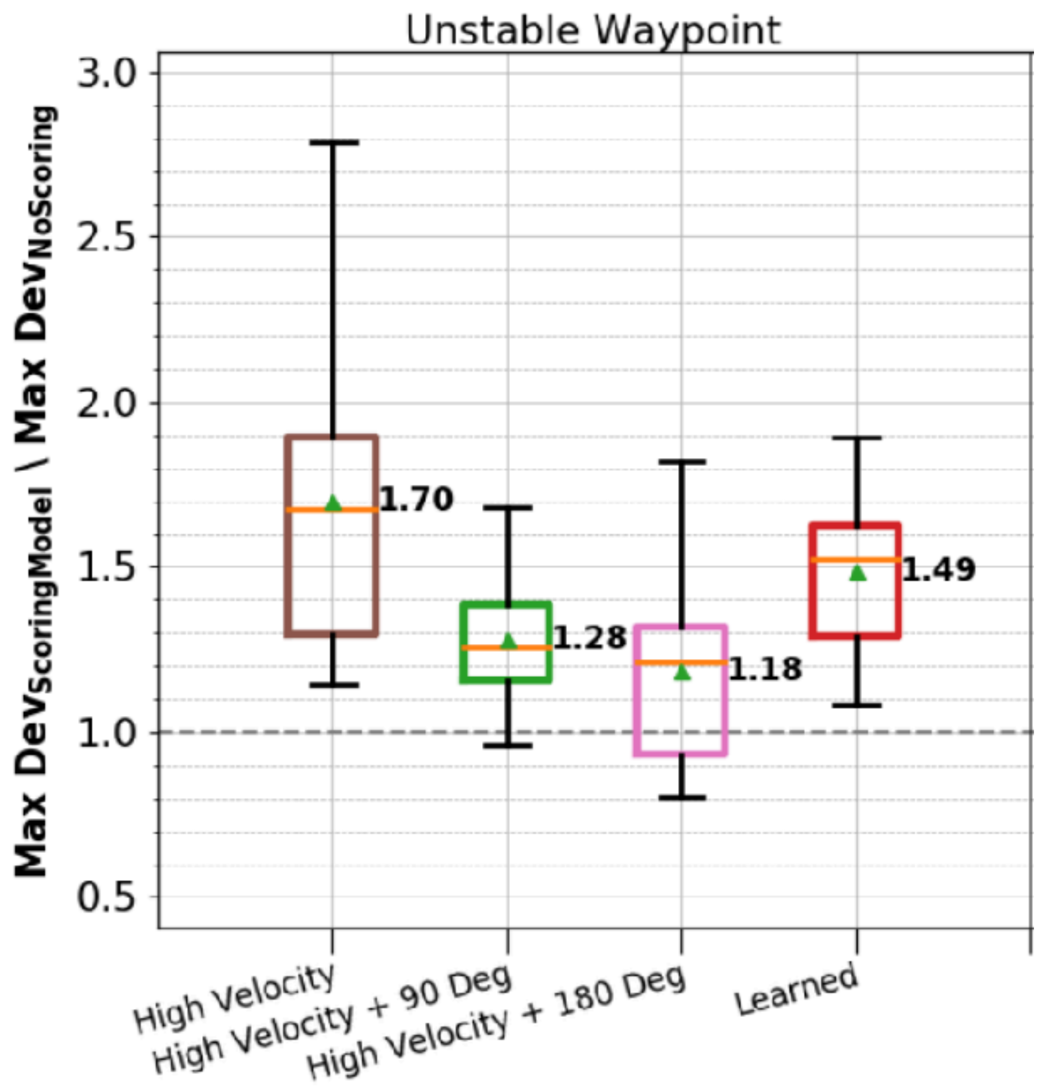
Study

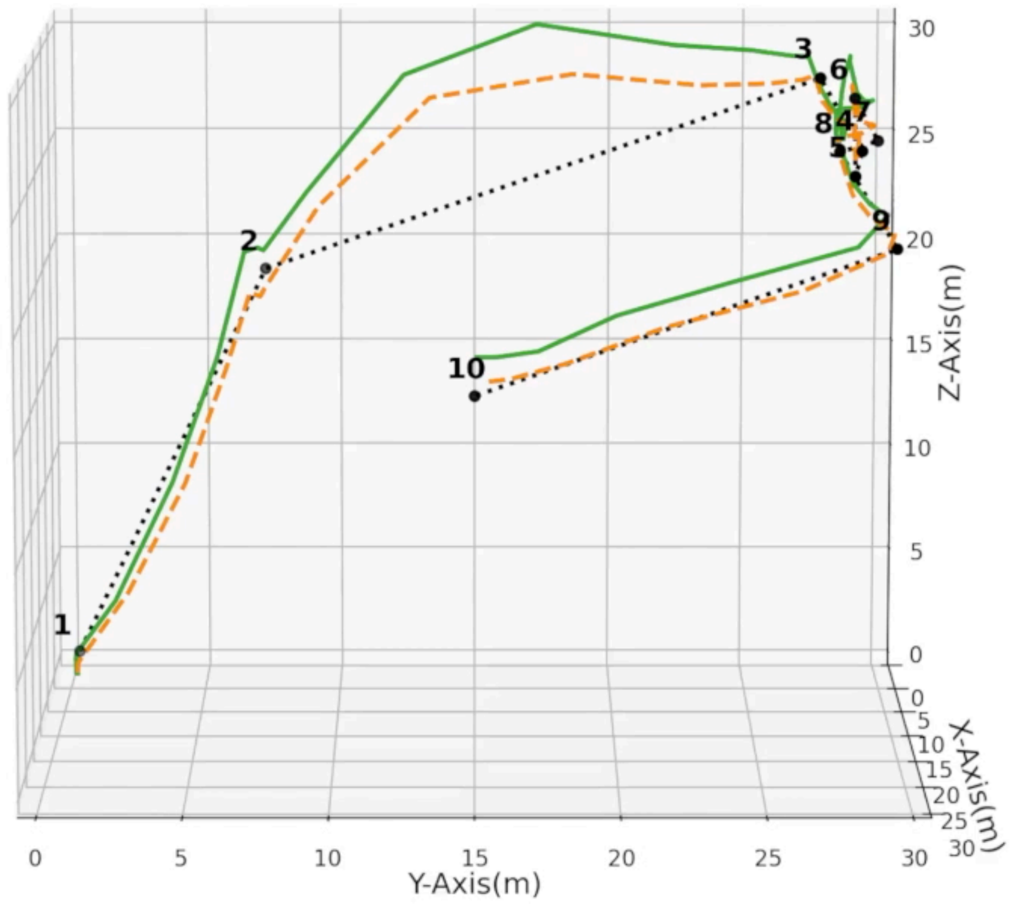
Performed a study on a commercial quadrotor in the real world.



Conclusion

Takeaway: We have introduced a **novel approach for the automatic generation of feasible and stressful trajectories** for mobile robots. The approach was able to generate **valid trajectories** that caused a **mean increase of stress of up to 76%**.





Contact: hildebrandt.carl@virginia.edu

This work was funded in part by the NSF and U.S. Army Research Office Grant