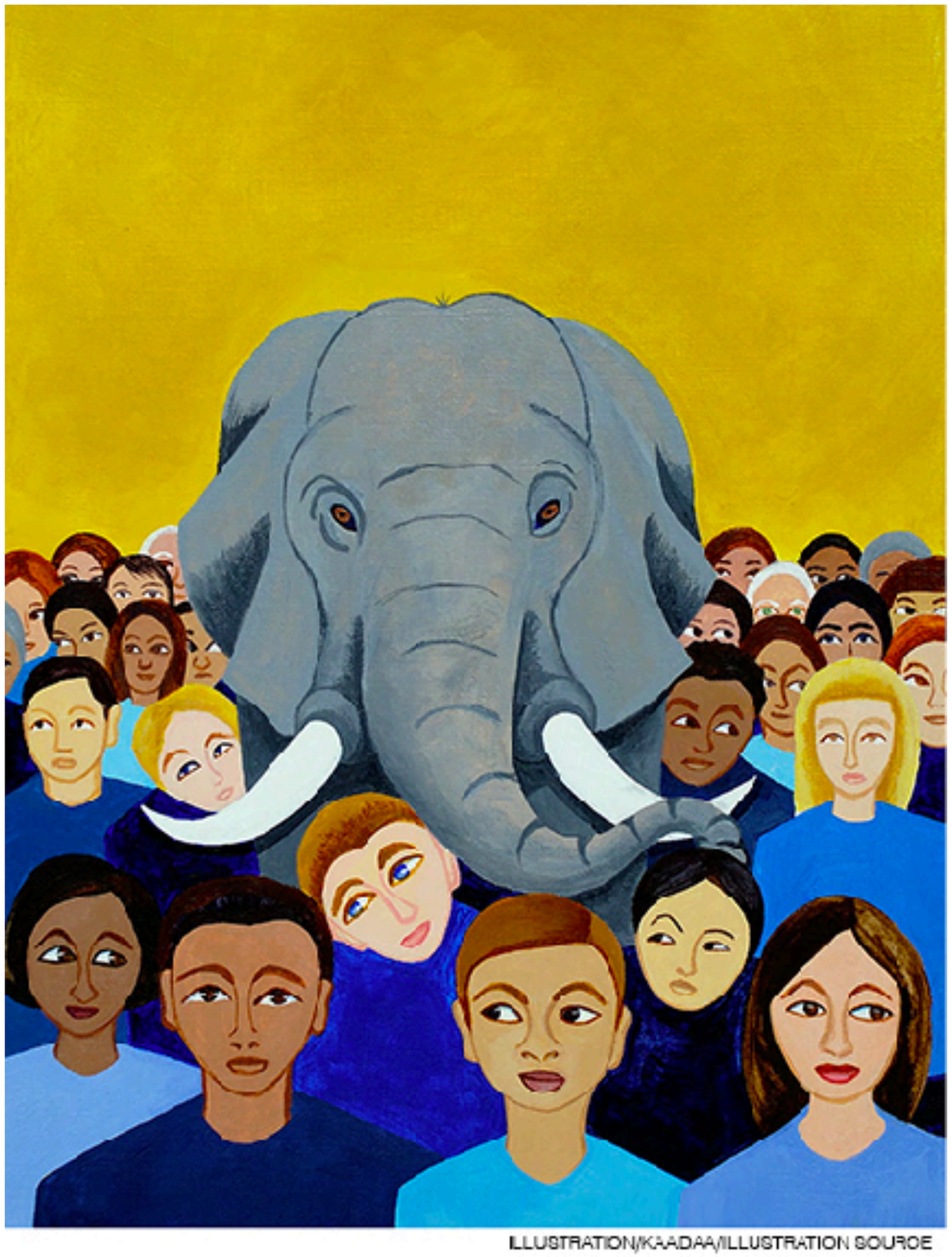
CSE 490R:

Mobile Robots

Instructor: Sanjiban Choudhury

TAs: Matthew Rockett, Gilwoo Lee,
Matt Schmittle

What did we learn ....



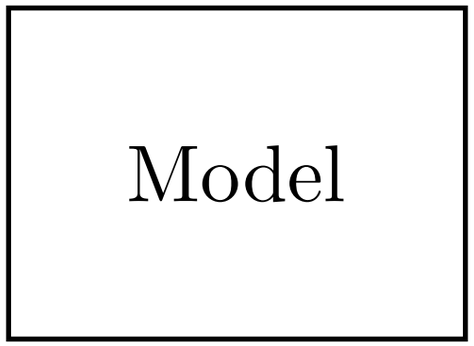
ILLUSTRATION/KAADAA/ILLUSTRATION SOURCE

... and why was there no learning?!

If machine learning is the process of building models about the world...

...then we were talking about learning the whole time

How do we build the model?



Three questions you should ask

1. What are we trying model?

2. What defines a good model?

3. What model should I use for my robot?

Bayes filter is a powerful tool

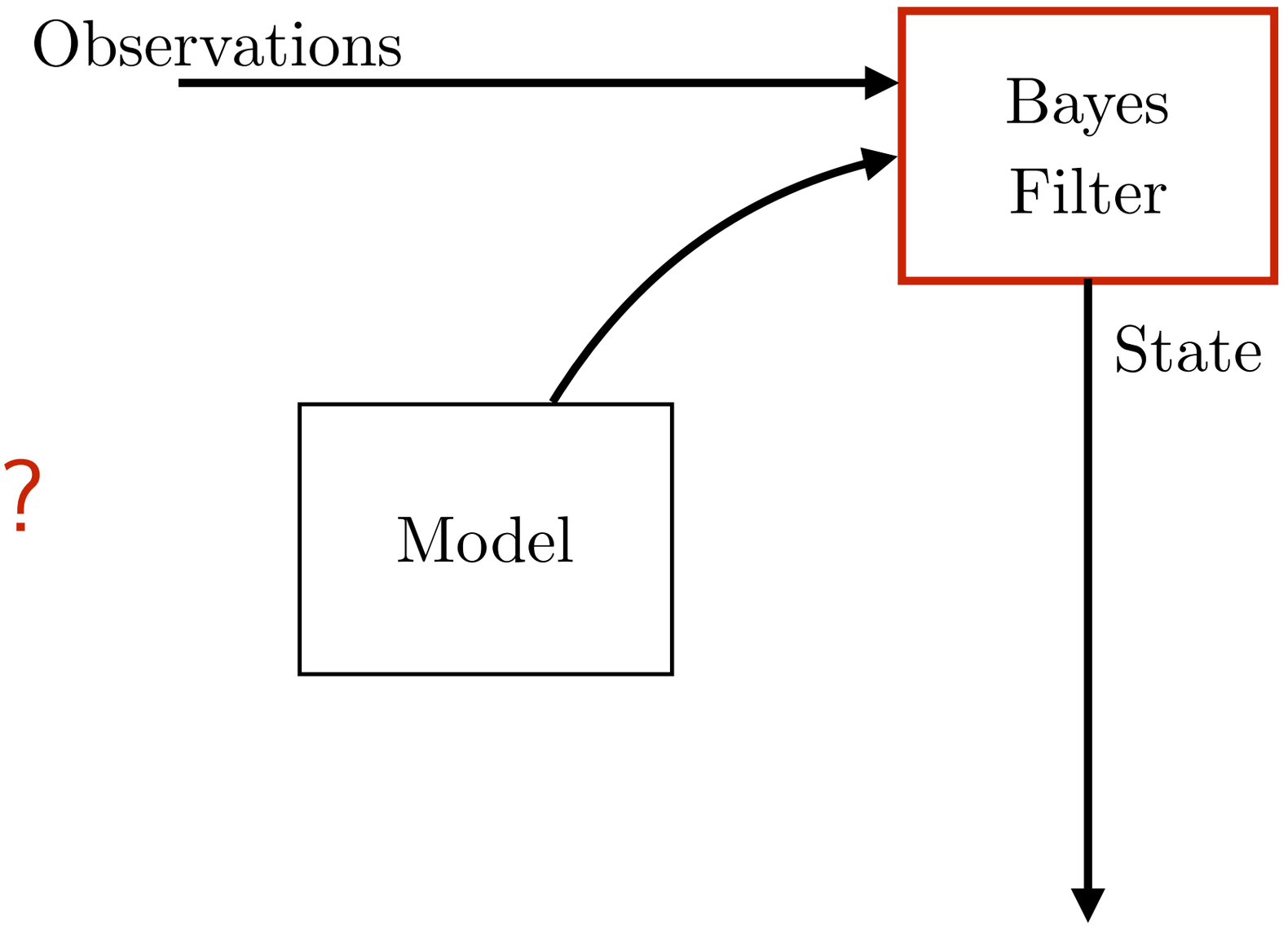
Localization

Mapping

SLAM

POMDP

How do we build the model?



Model predictive control (MPC)

$$\min_{u_{t+1},\ldots u_{t+H}} \sum_{k=t}^{t+H-1} J(x_k, u_{k+1})$$

(plan till horizon H) (Cost)

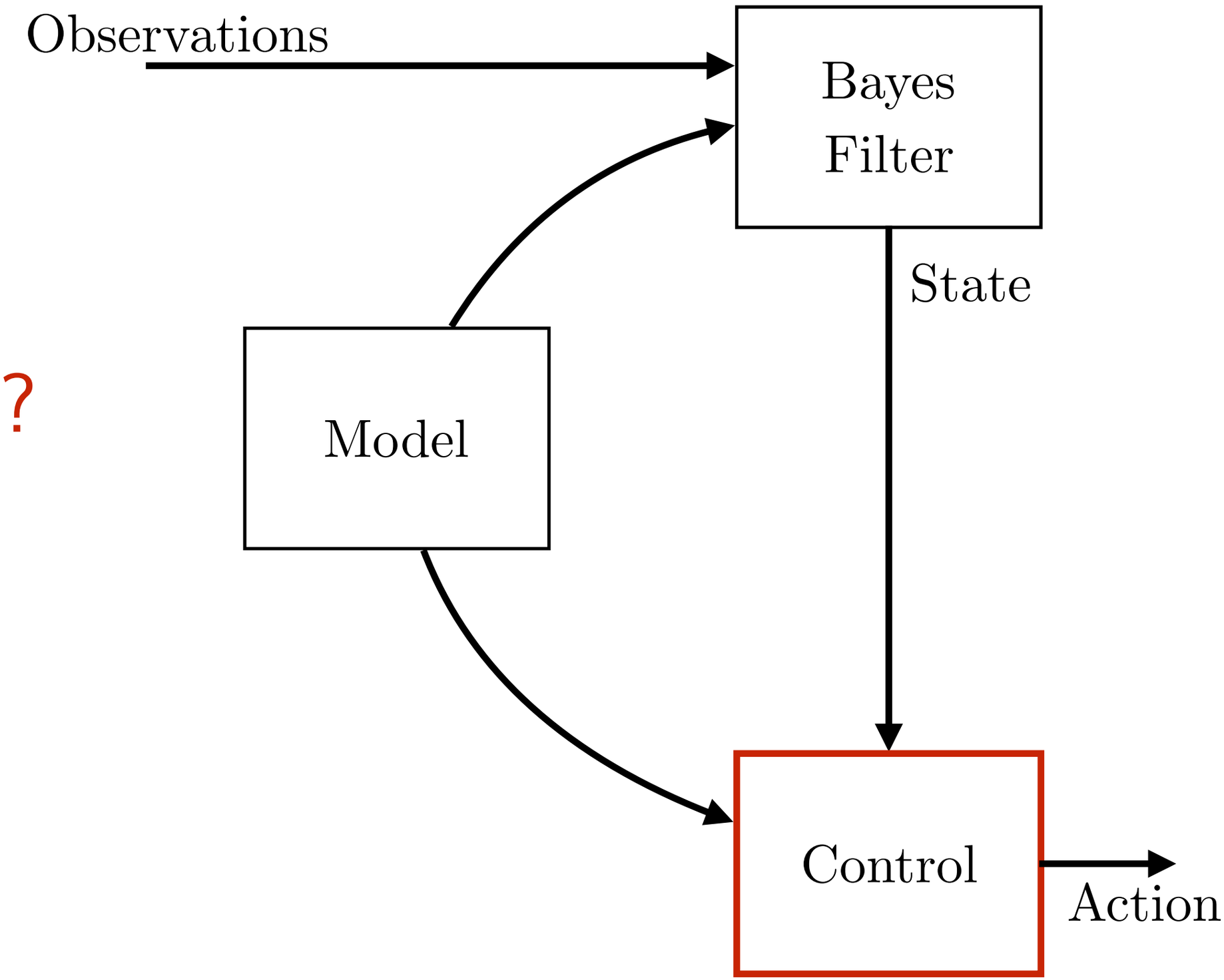
$$x_{k+1} = f(x_k, u_{k+1})$$

(Predict next state with dynamics)

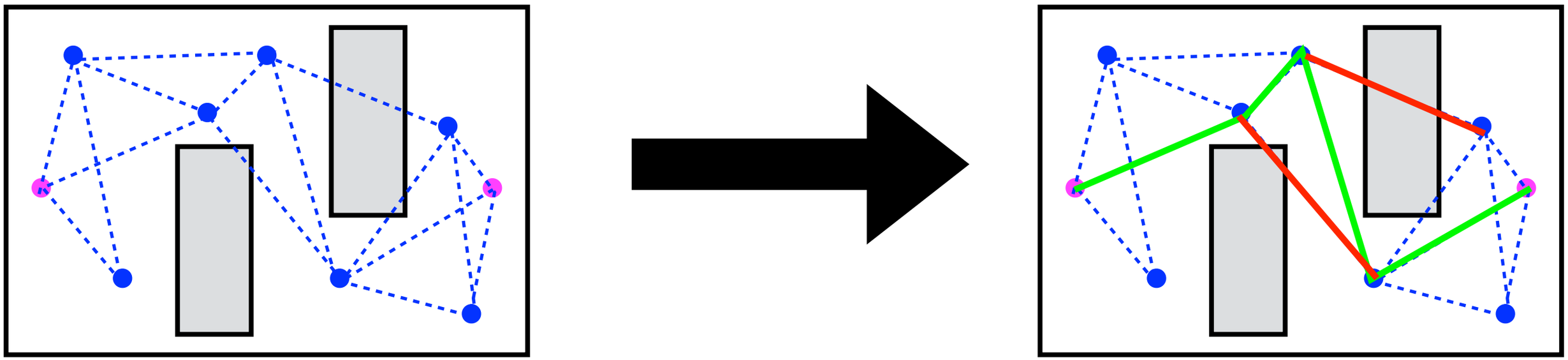
$$g(x_k, u_{k+1}) \leq 0$$

(Constraints)

How do we build the model?



General framework for motion planning

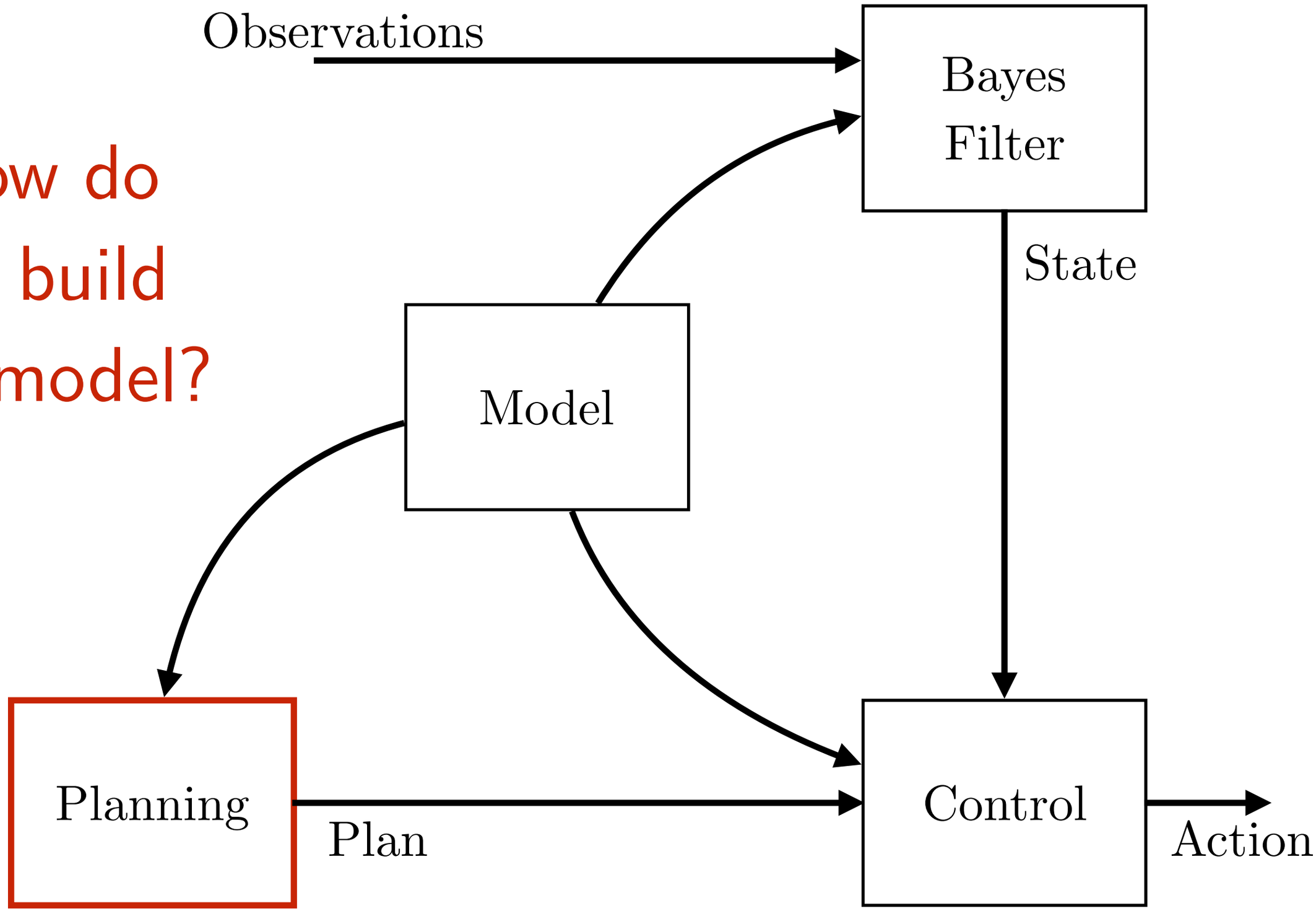


Create a graph

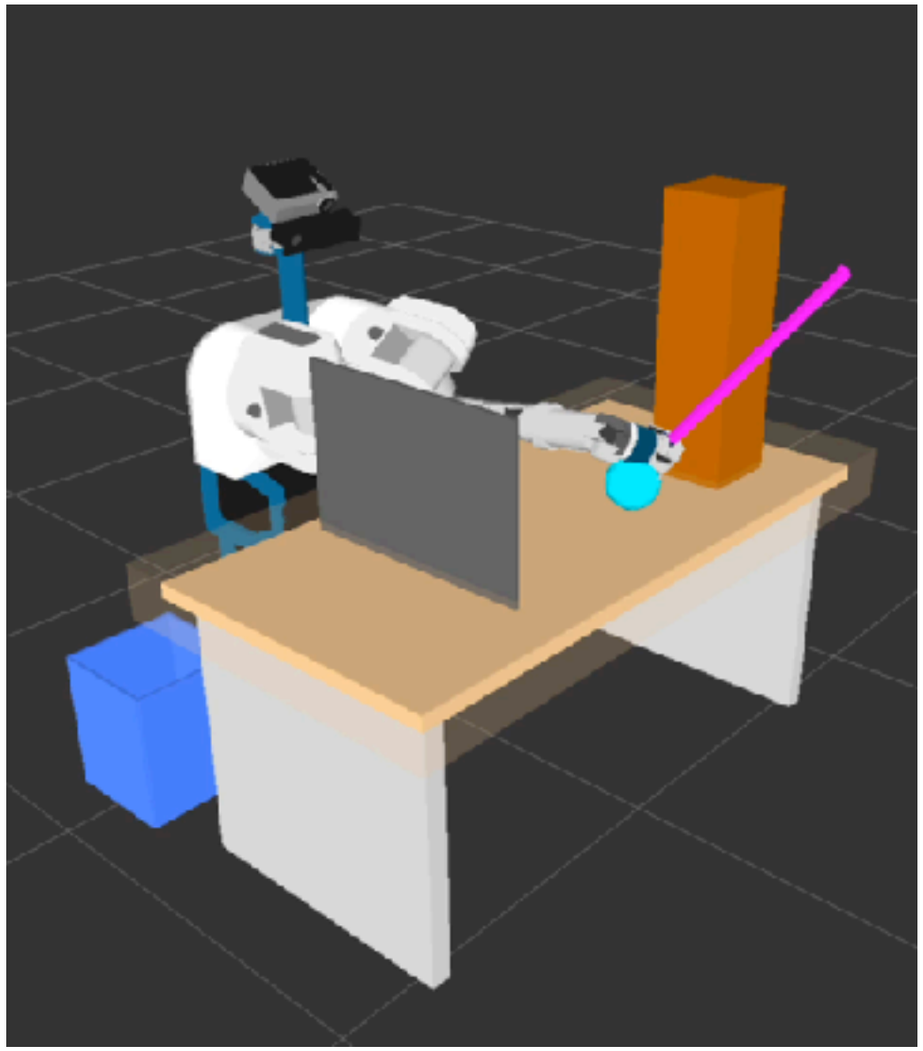
Search the graph

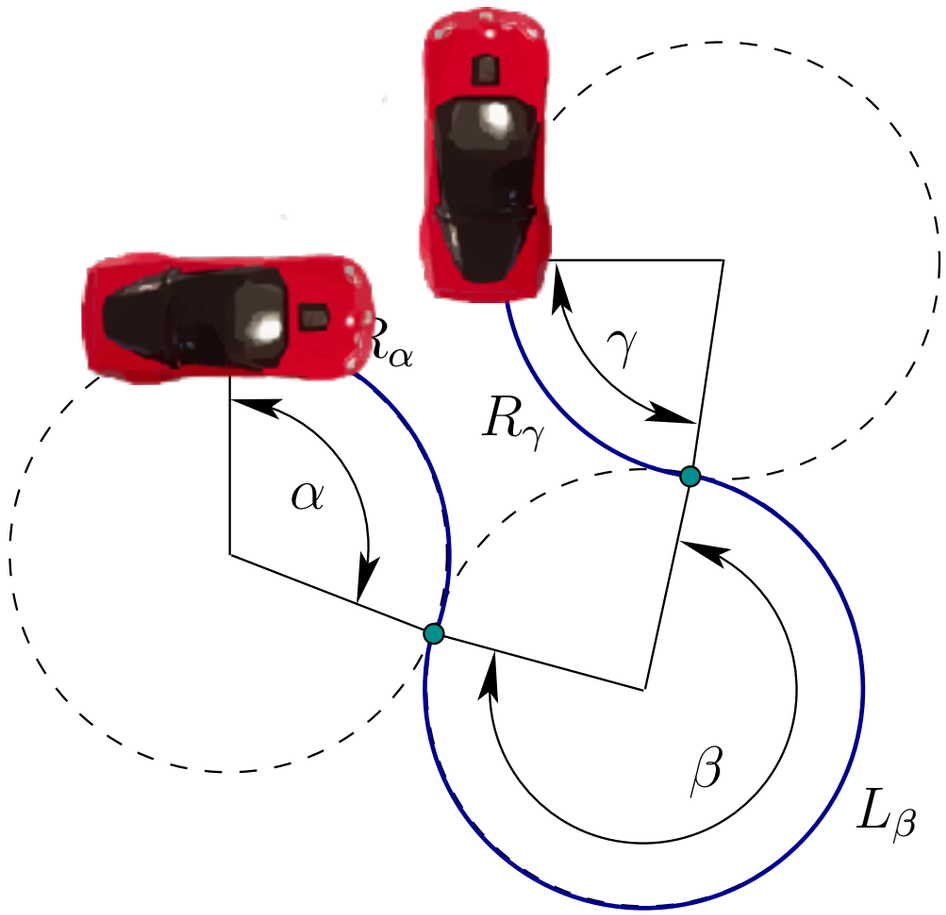
Interleave

How do we build the model?



What are the models?



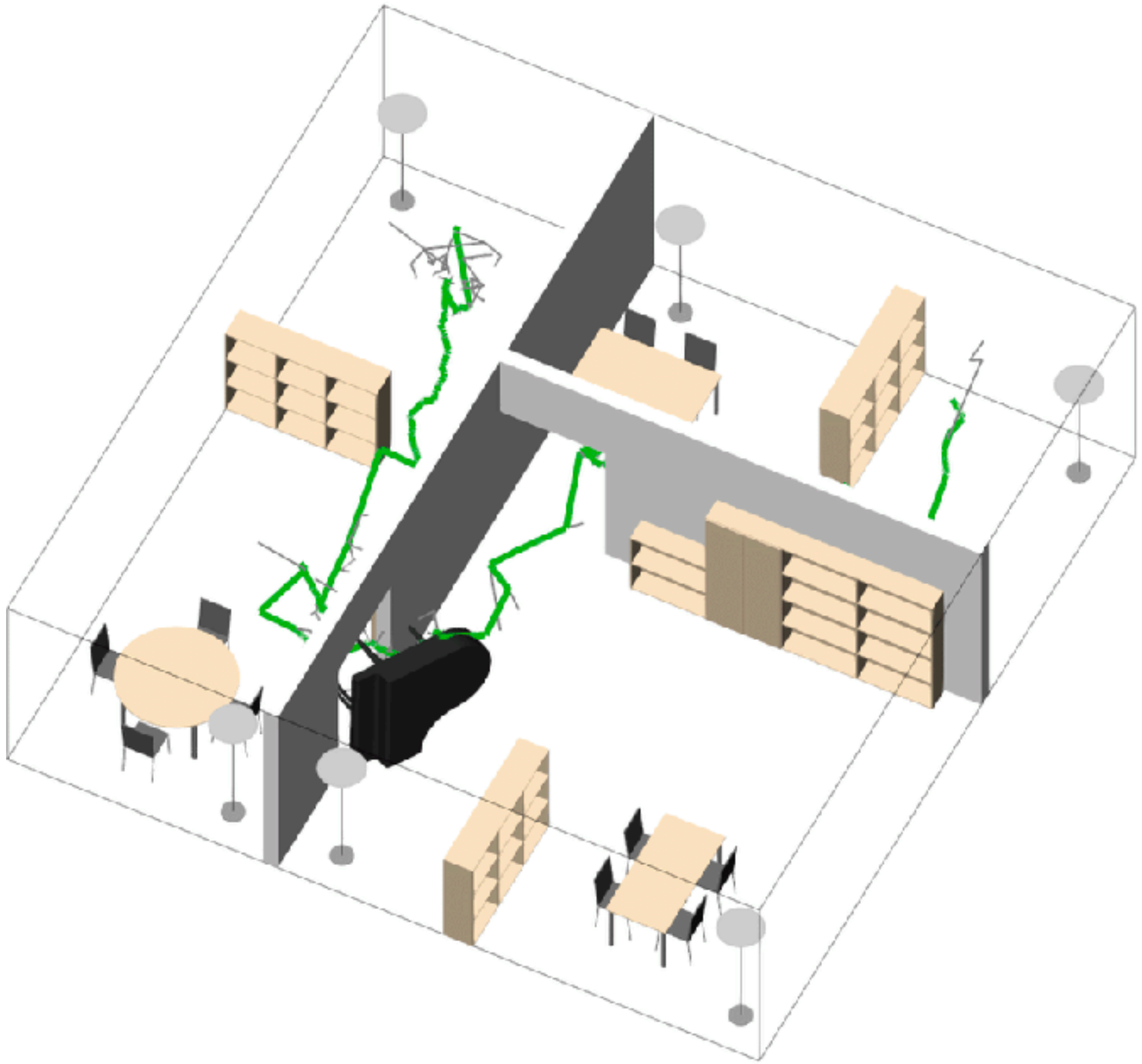


$R_\alpha L_\beta R_\gamma$

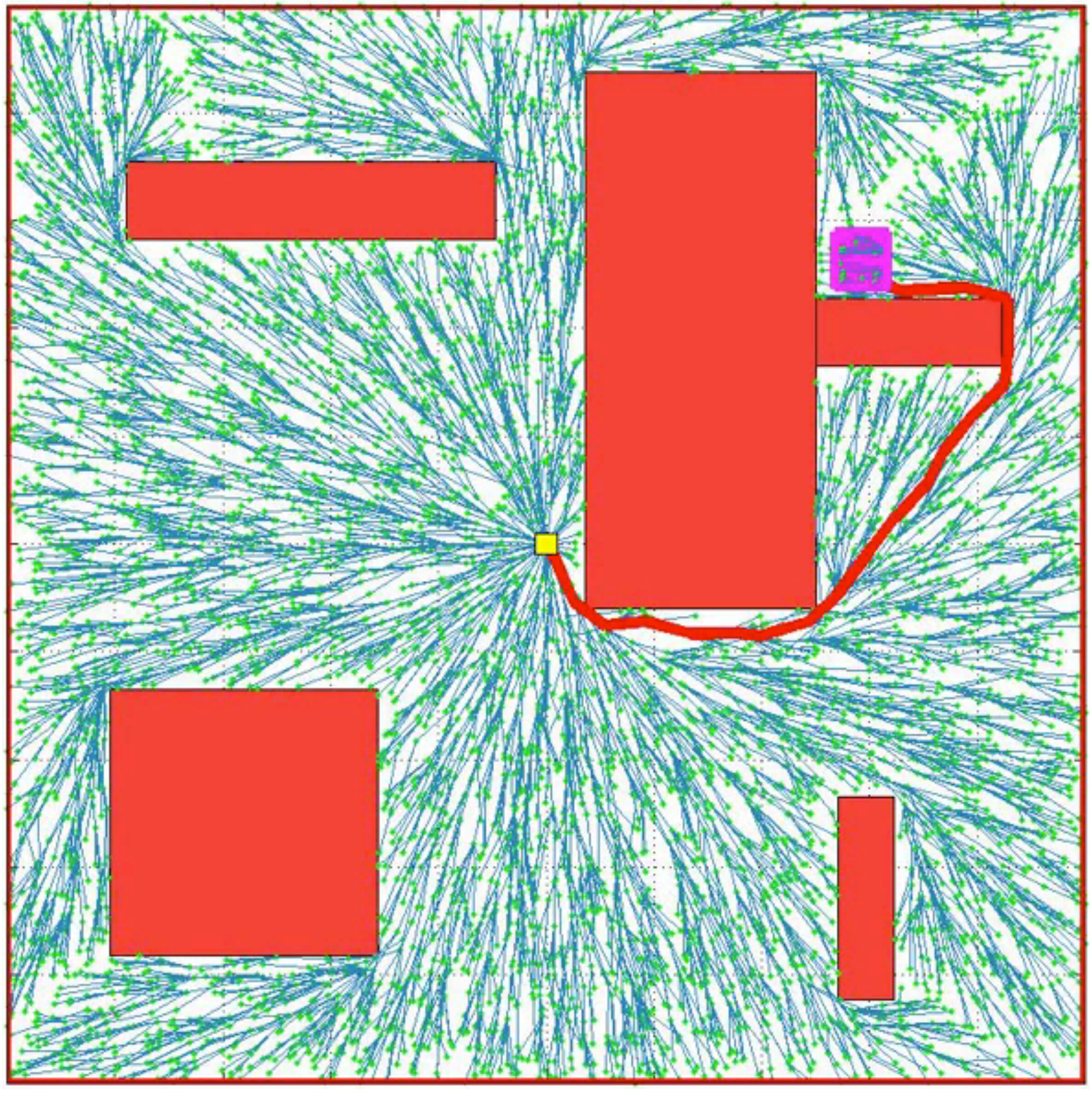
Are heuristics models?

Are graphs models?

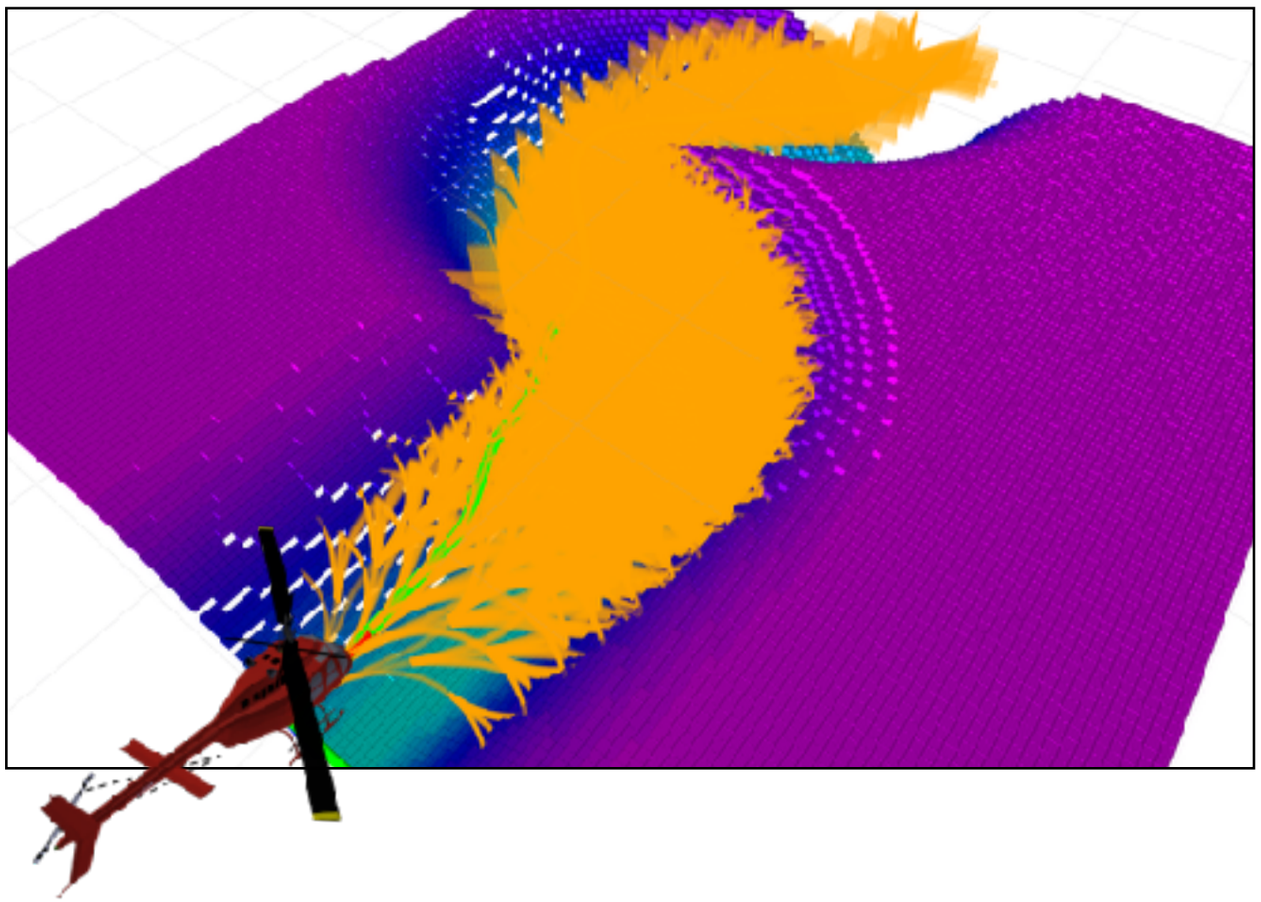
The Piano Mover's Problem



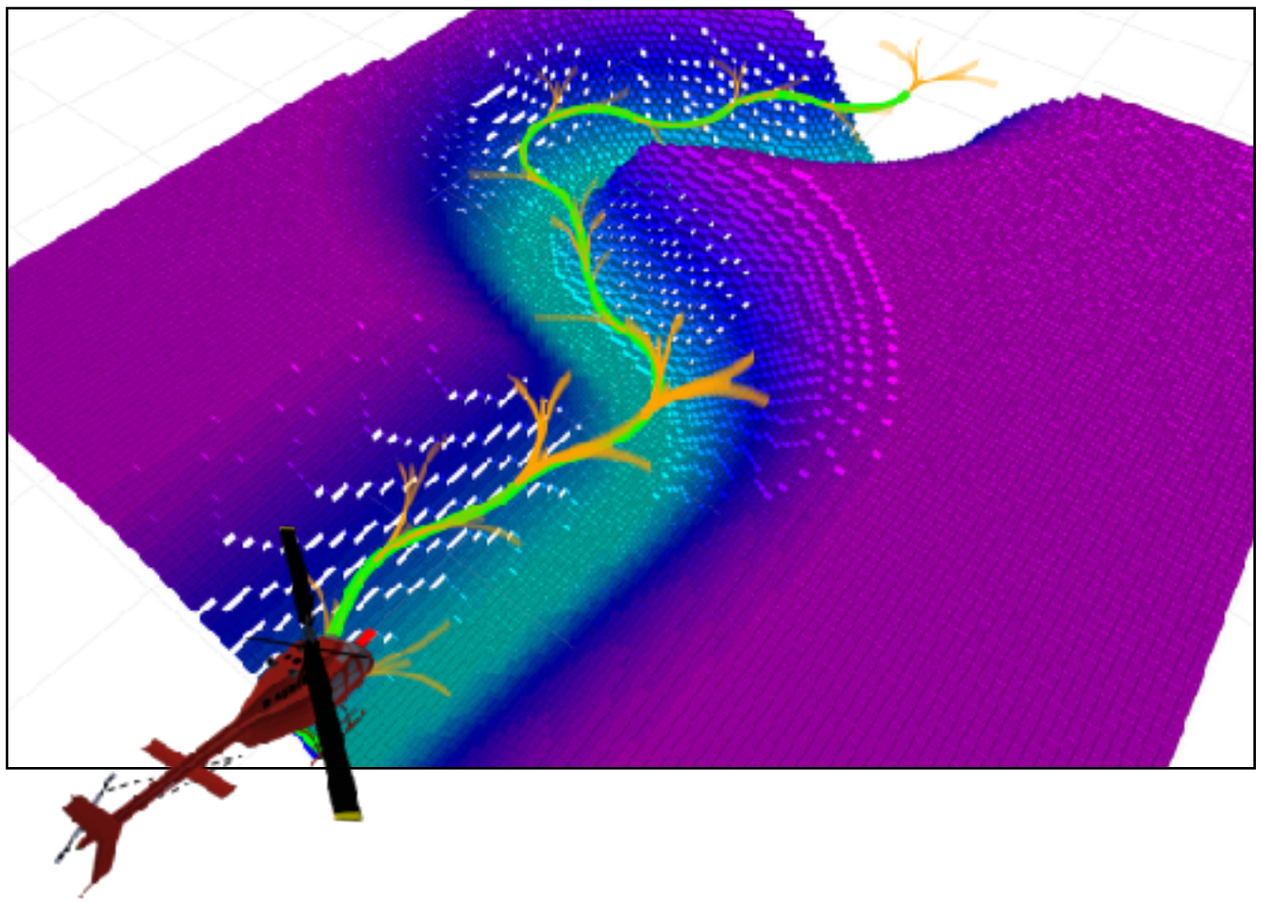
Problem: Planners search everywhere



… and I mean everywhere



Vanilla search



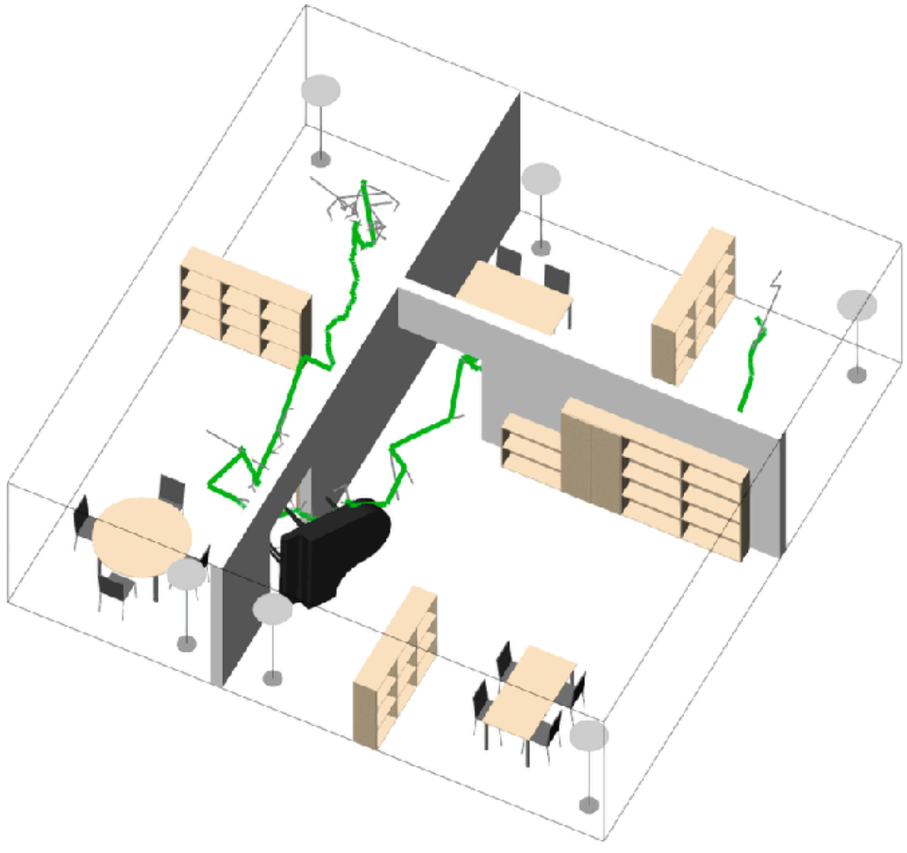
Search with common sense

Are heuristics models?

Are graphs models?

YES! They are models of
the spatial structure of the world

The Experienced Piano Movers' Problem



New Piano.
New House.
Same Mover.

How do we learn good models to improve real-time planning?

Is there ONE universal model for all scenarios?

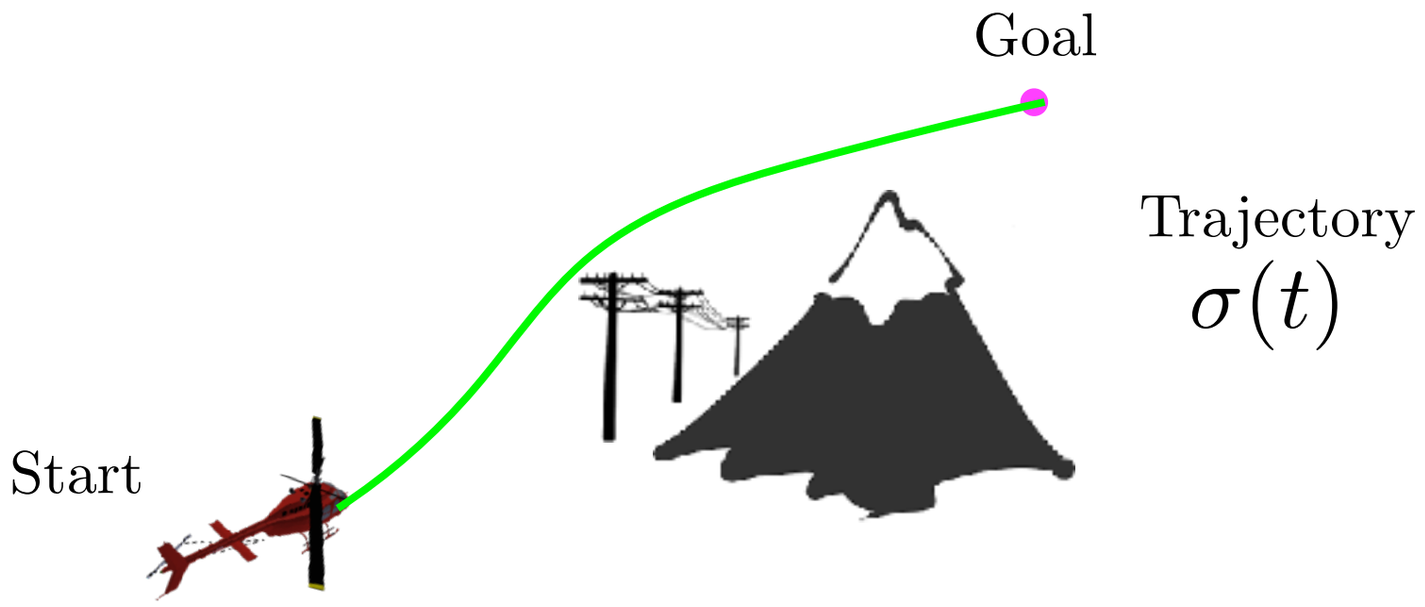
No! Depends on the environment

[AHS'14, ICRA'15, ICRA'16, ICRA'17]

Real-time performance of a planning strategy varies across problem instances

A planning module needs to adapt to the distribution of problems the robot encounters

Planning problem



$$\min_{\sigma \in \Sigma} \quad J(\sigma) \qquad \text{Minimize cost}$$

$$\begin{aligned} \text{s.t} \quad & \sigma(0) \in \Sigma_{\text{start}} \\ & \sigma(t_f) \in \Sigma_{\text{goal}} \end{aligned} \qquad \left| \quad \text{Start / Goal constraints} \right.$$

$$\begin{aligned} & \mathcal{F}(\sigma(t)) = 0 \\ & \mathcal{H}(\sigma(t)) \leq 0 \end{aligned} \qquad \left| \quad \text{Dynamics constraints} \right.$$

$$\sigma(t) \in \Sigma_{\text{valid}} \qquad \text{Obstacle constraints}$$

Performance of a planner

The performance is the cost of the solution $J(\sigma)$ computed by the planner in a finite time budget T

We use the following notation to represent this performance:

$$J(\Gamma, \mathcal{P}) \in [0, J_{\max}]$$

What do we want?

Let $P(\Gamma)$ be the distribution of planning problem parameters
(distribution of start/goal, obstacles, dynamics)

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} \mathbb{E}_{P(\Gamma)}[J(\Gamma, \mathcal{P})]$$

(create a planner)

(distribution of planning problems)

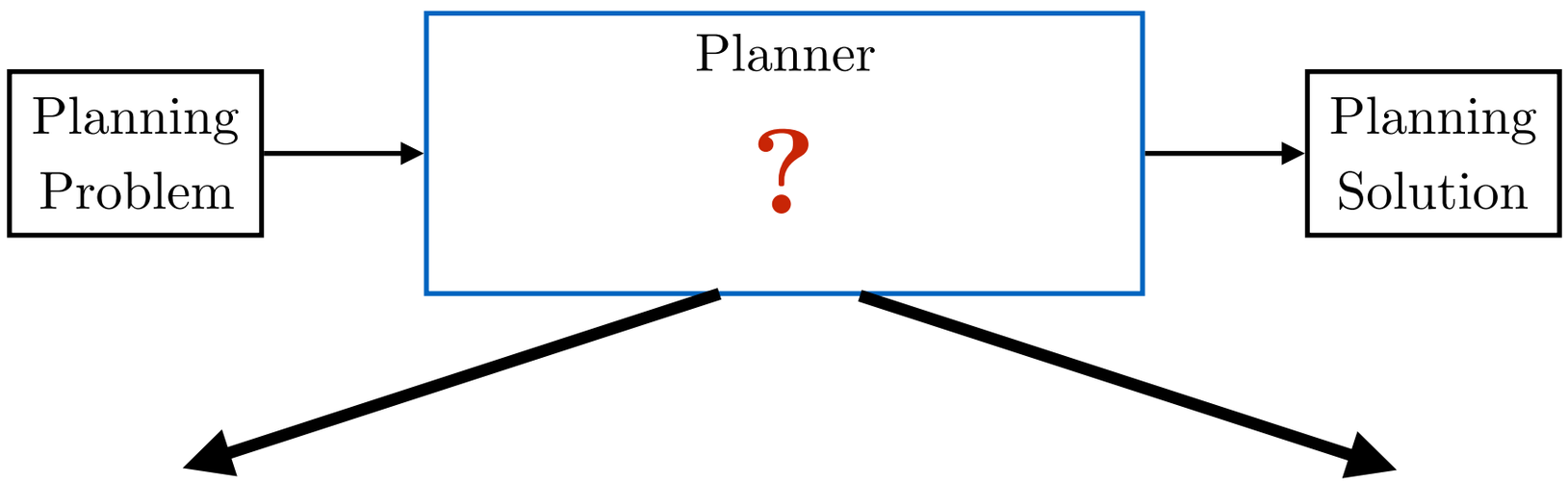
(cost of solution computed in real-time)

Problem:

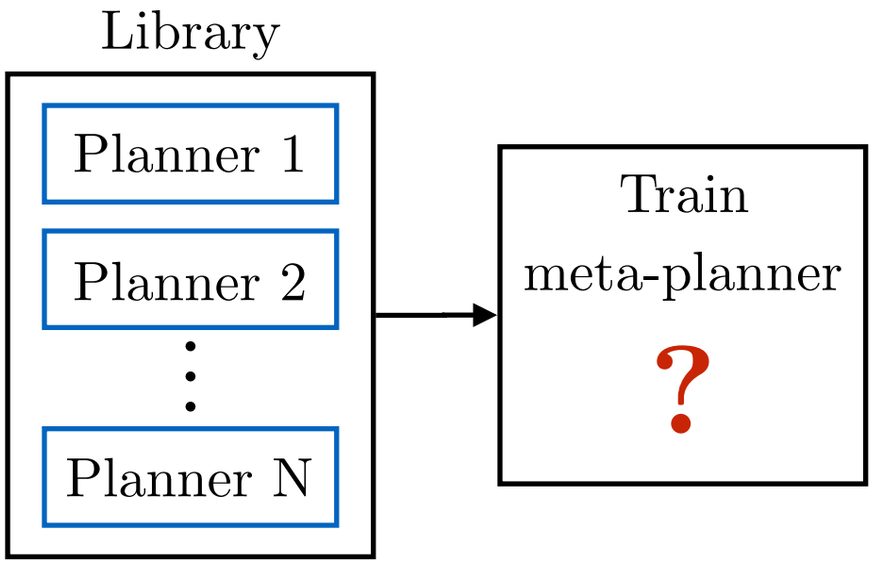
How do we tractably search over the space of all planners?

Approach outline

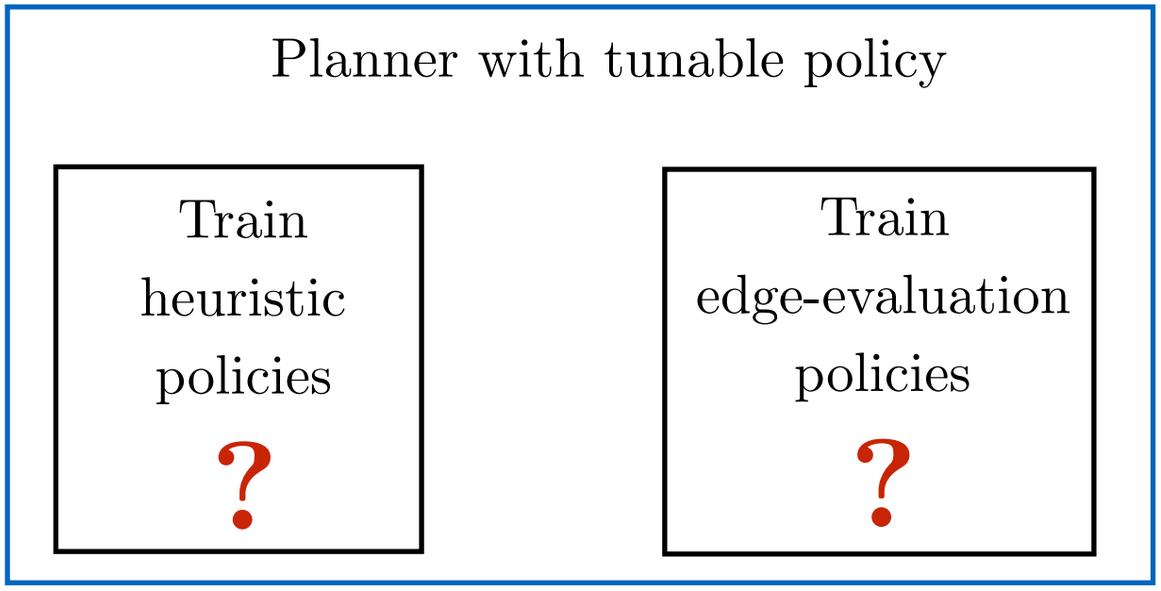
Q1. Framework to assemble real-time planners?



Q2. (Black-box adaptation)
Select planners from a library?



Q3. (White-box adaptation)
Directly learn planning policies?

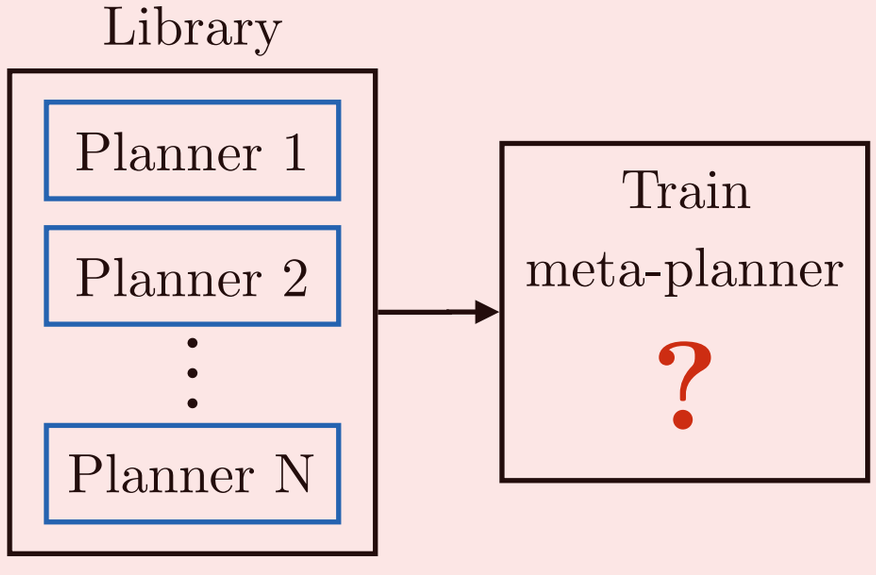


Approach outline

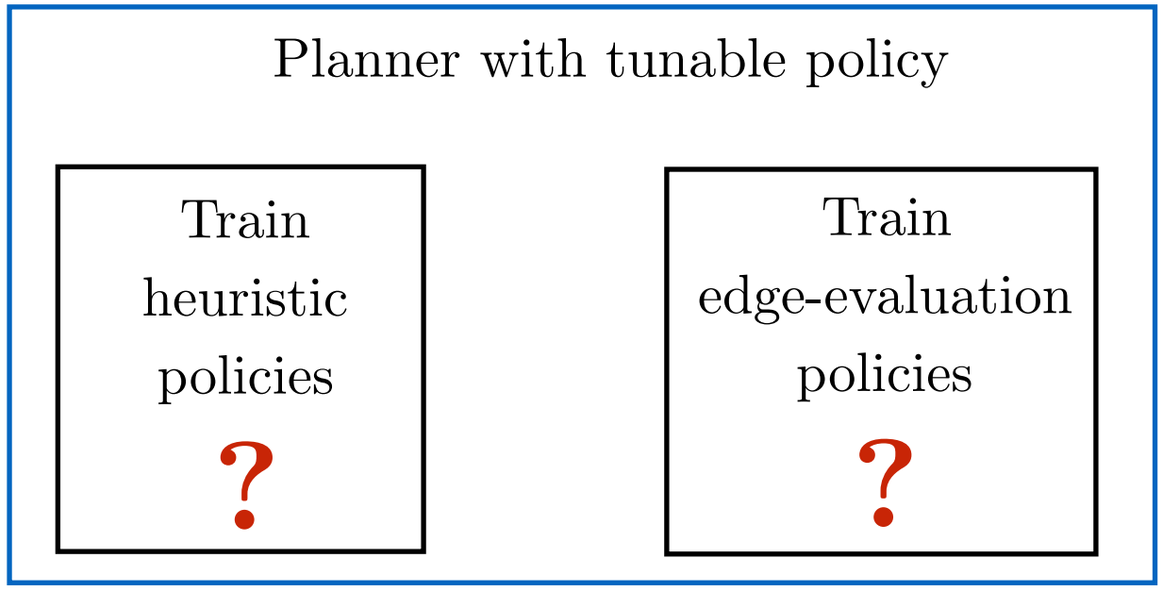
Q1. Framework to assemble real-time planners?

Expert planner with custom implicit graphs

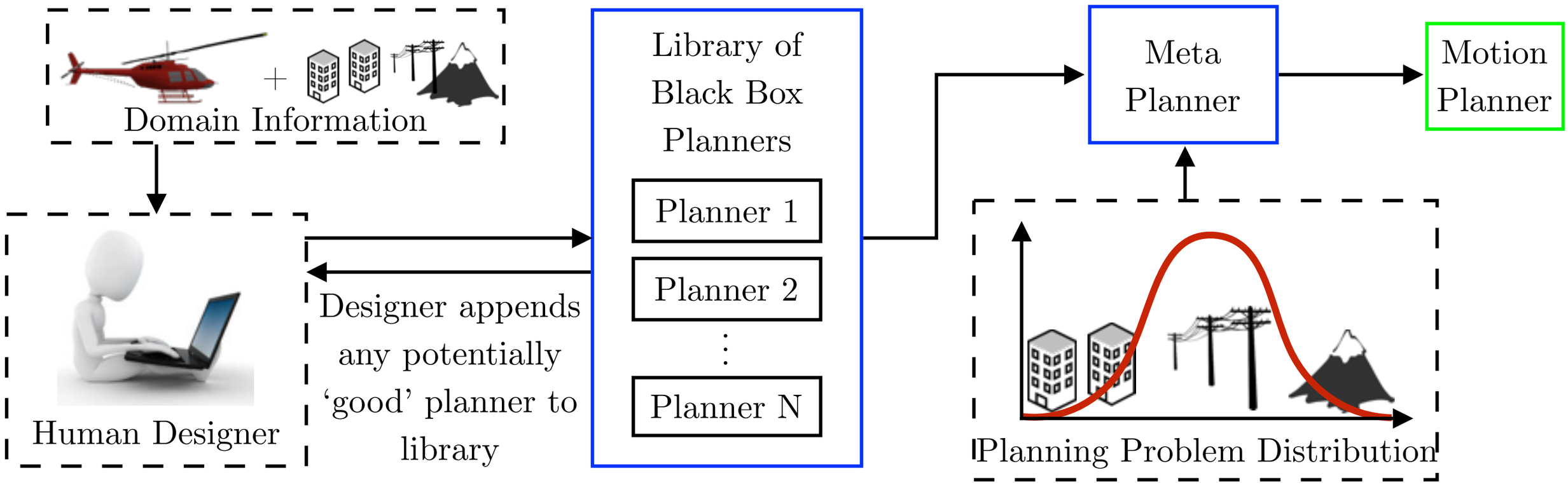
Q2. (Black-box adaptation)
Select planners from a library?



Q3. (White-box adaptation)
Directly learn planning policies?



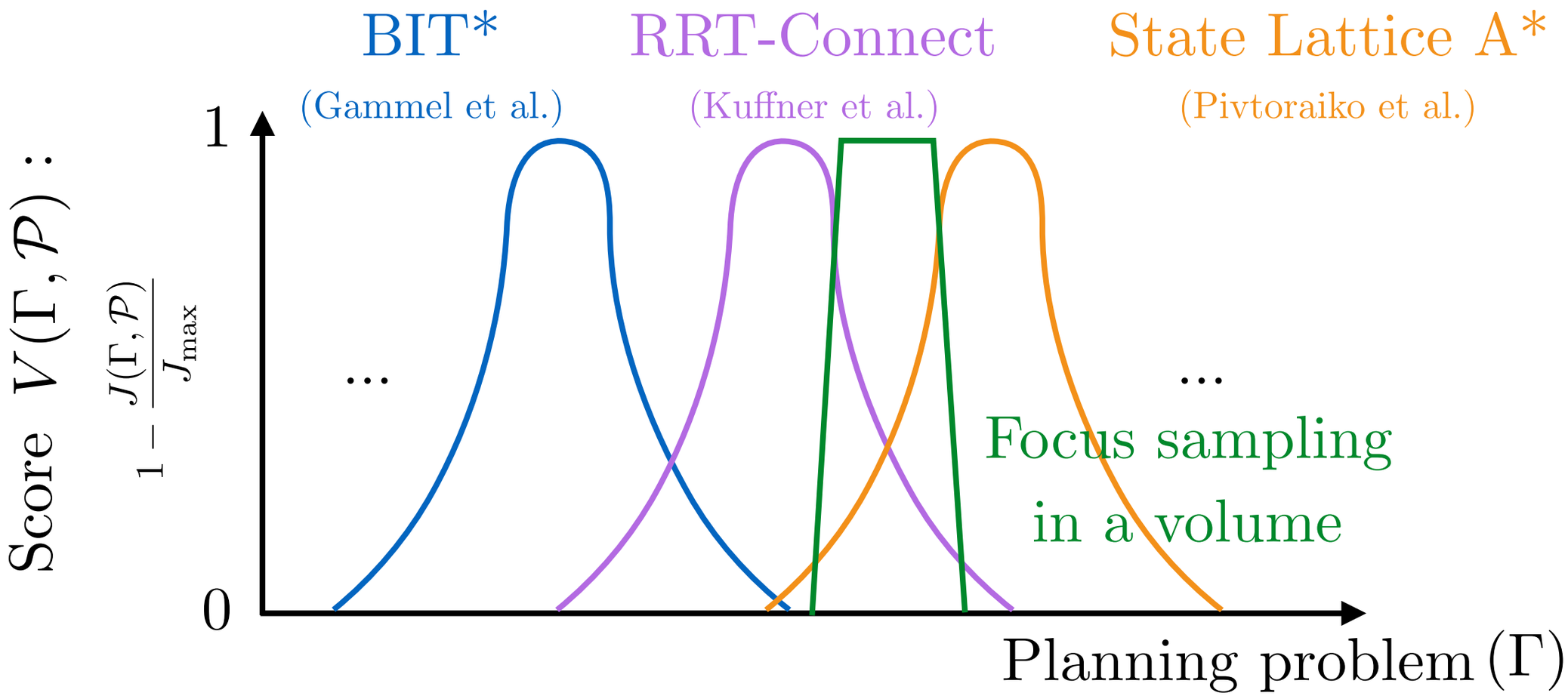
Black-box adaptive planning paradigm



Design a meta-planner that, given a planning problem distribution, adaptively selects planners from a library of black-box planners

Black-box implies planners are atomic operations

Library of diverse black-box planners



General purpose planners

Solves large number of problems,
small fraction has high score

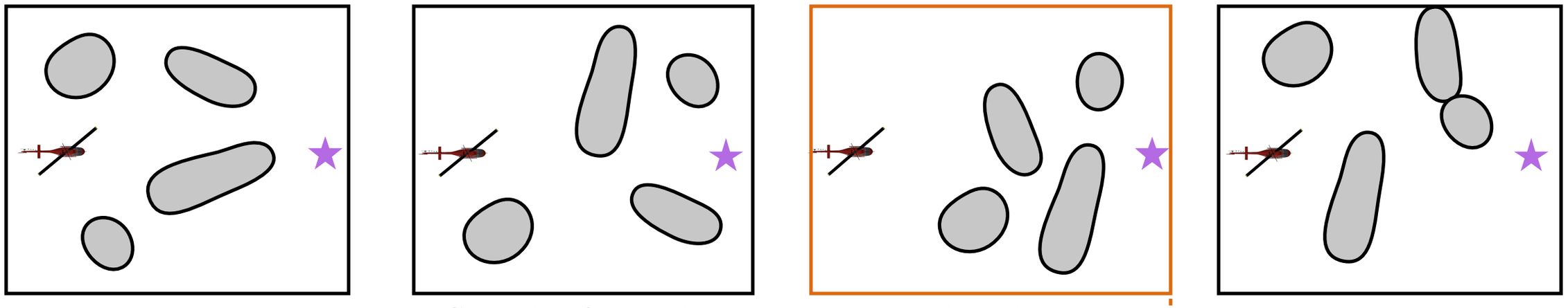
(not overfit on any problem)

Precision planners

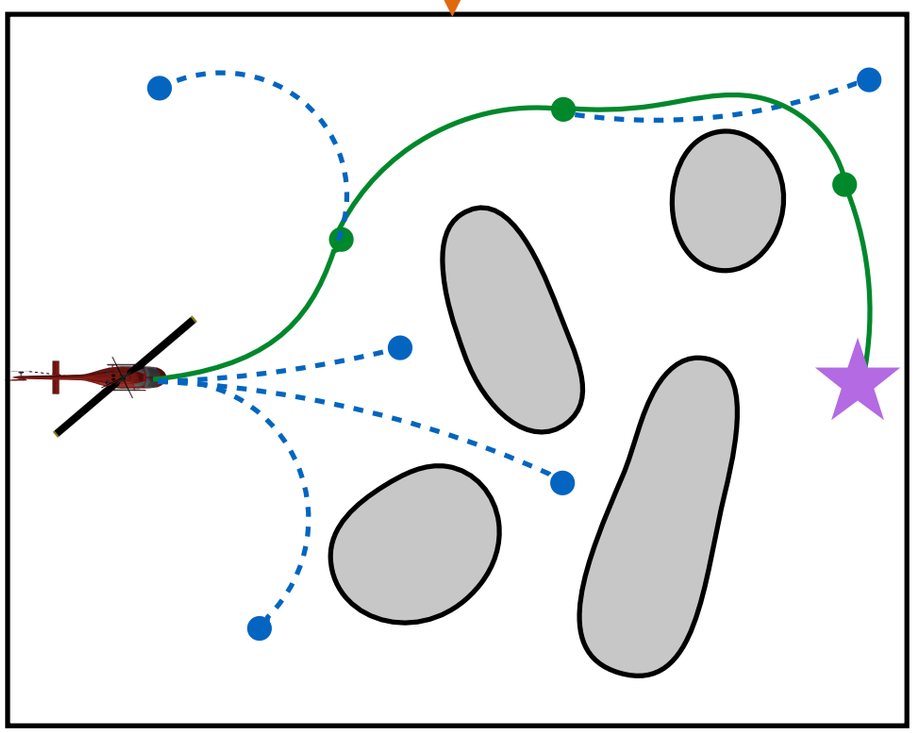
Solves small number of problems,
large fraction has high score

(designed to overfit)

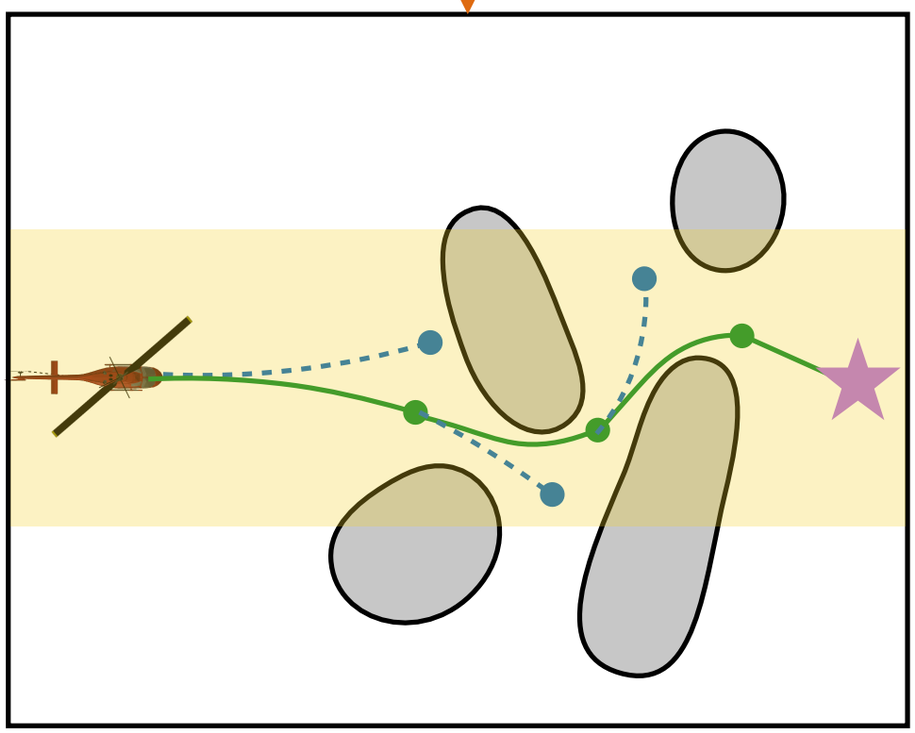
General purpose vs Precision planner



Samples from planning problem distribution

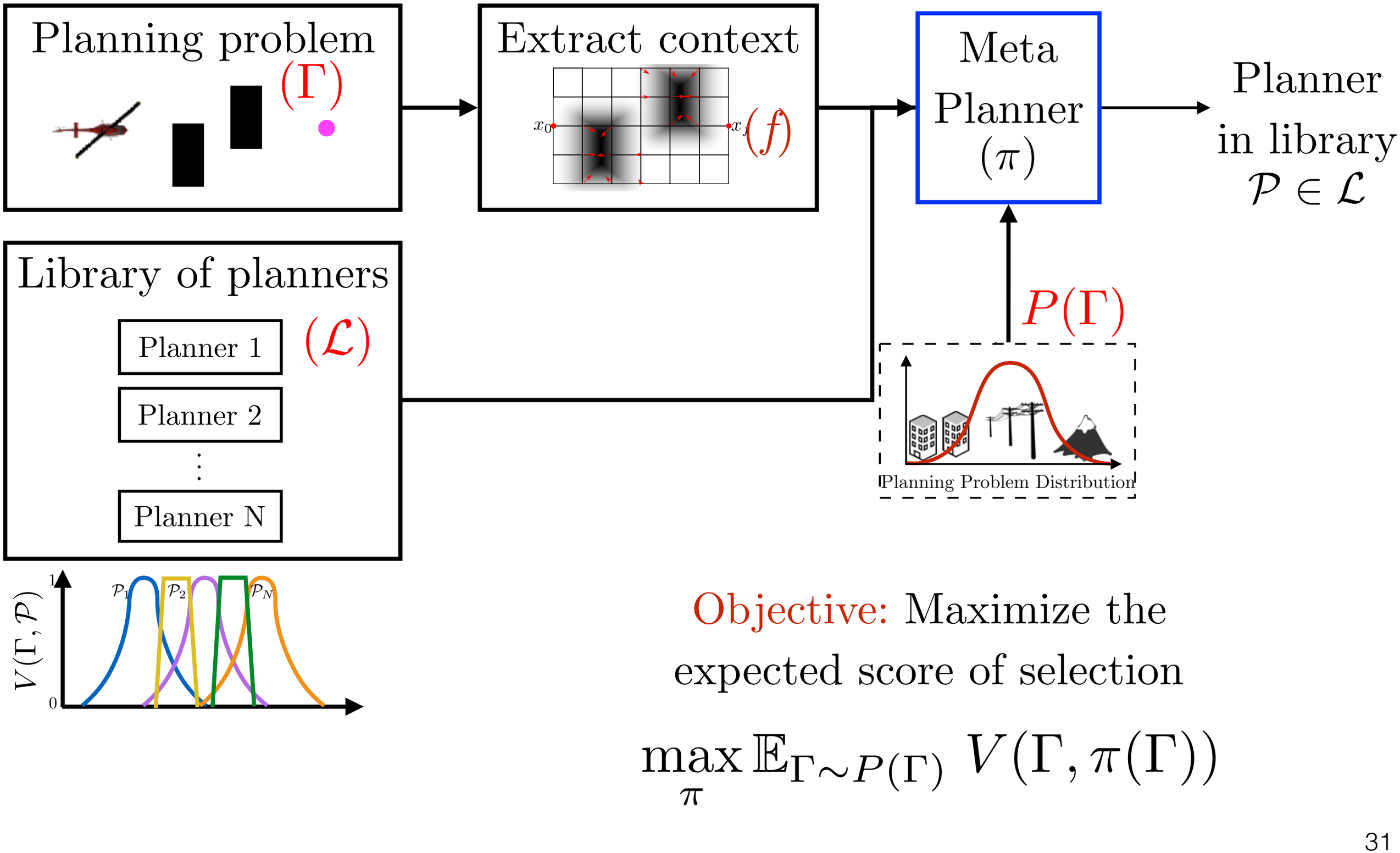


General purpose planner samples everywhere, finds a high cost solution



Precision planner focuses sampling, finds a low cost solution

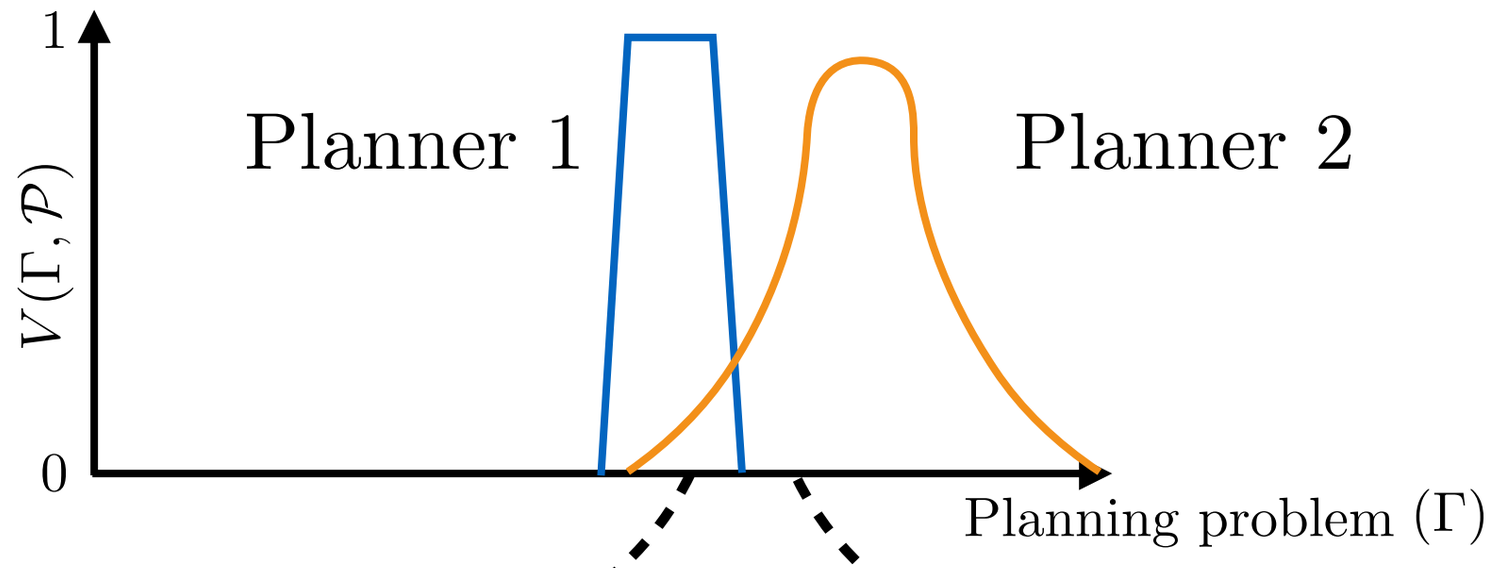
Meta-planner classifies a problem to a planner

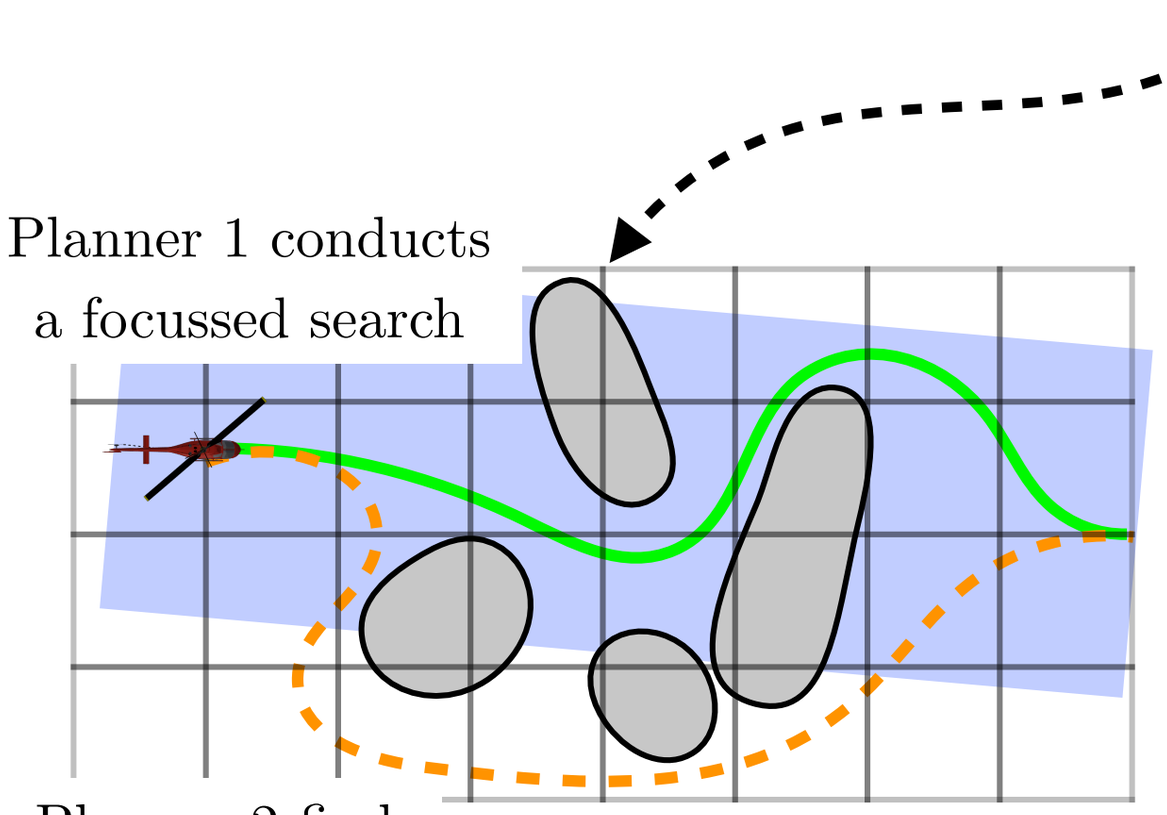


Objective: Maximize the expected score of selection

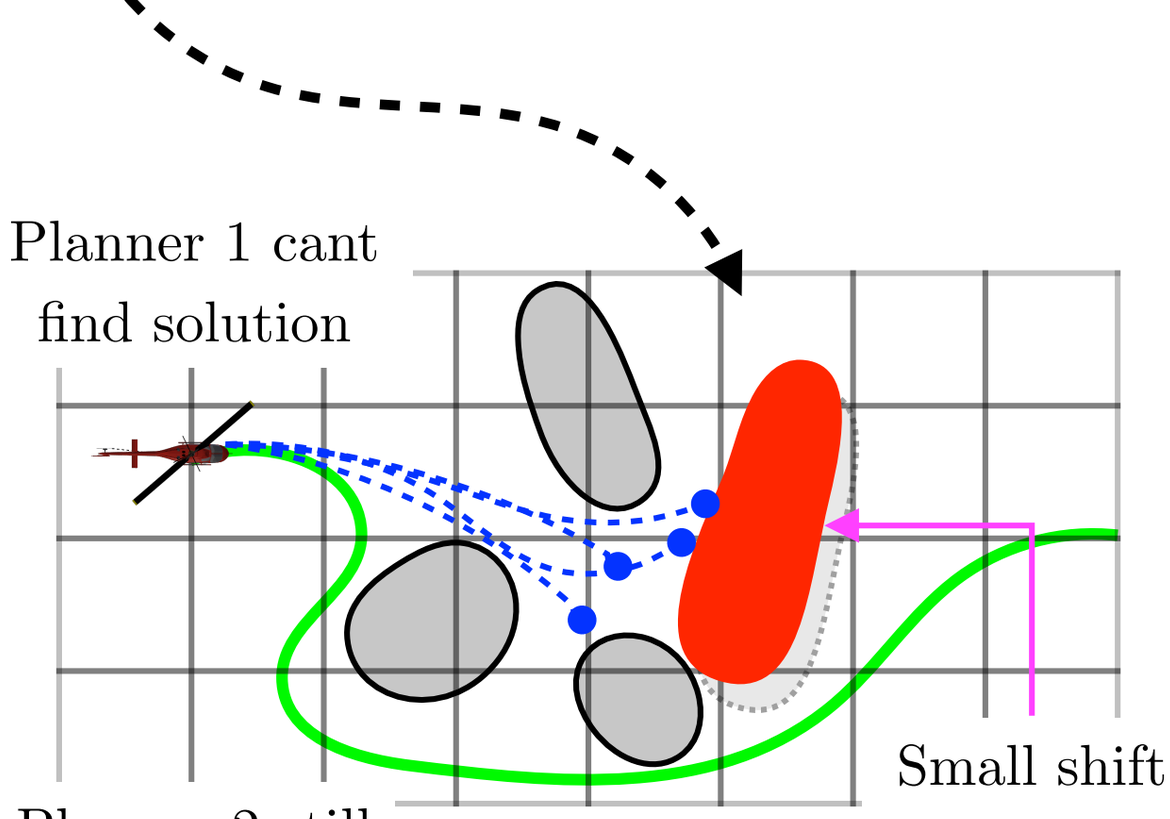
$$\max_{\pi} \mathbb{E}_{\Gamma \sim P(\Gamma)} V(\Gamma, \pi(\Gamma))$$

Difficulty in classifying precision planners





Planner 1 = 1.0, Planner 2 = 0.3

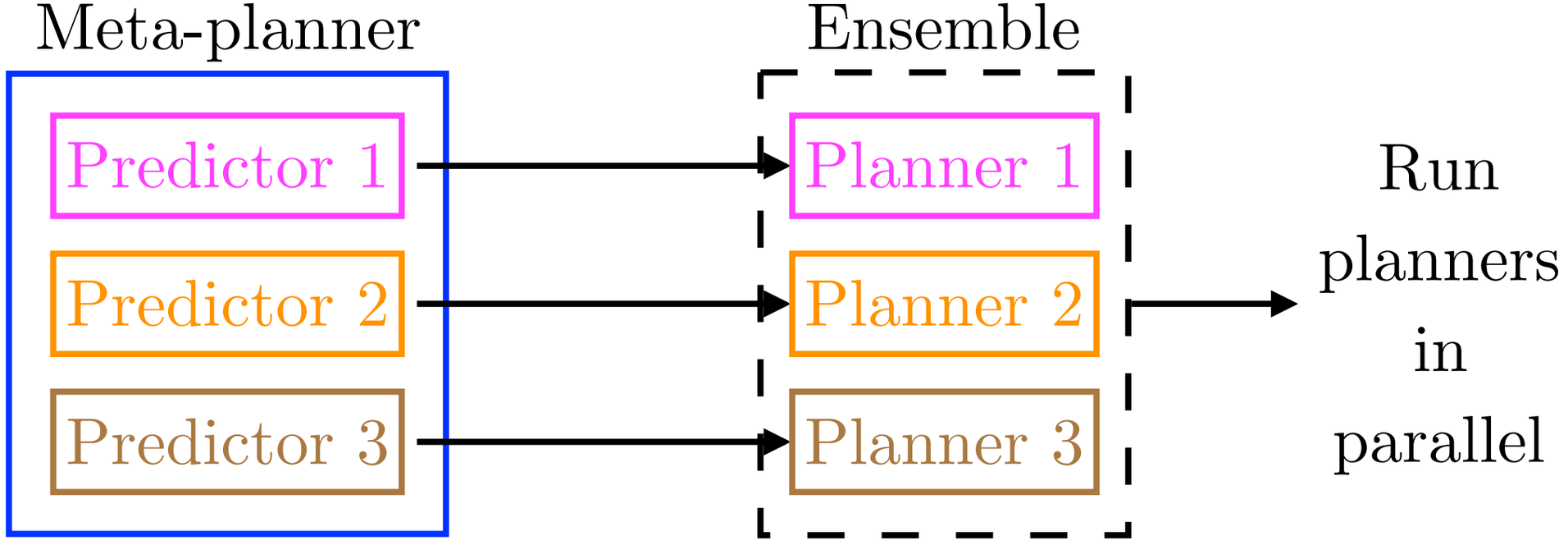


Planner 1 = 0.0, Planner 2 = 0.6

Key Idea: Predict an ensemble of planners

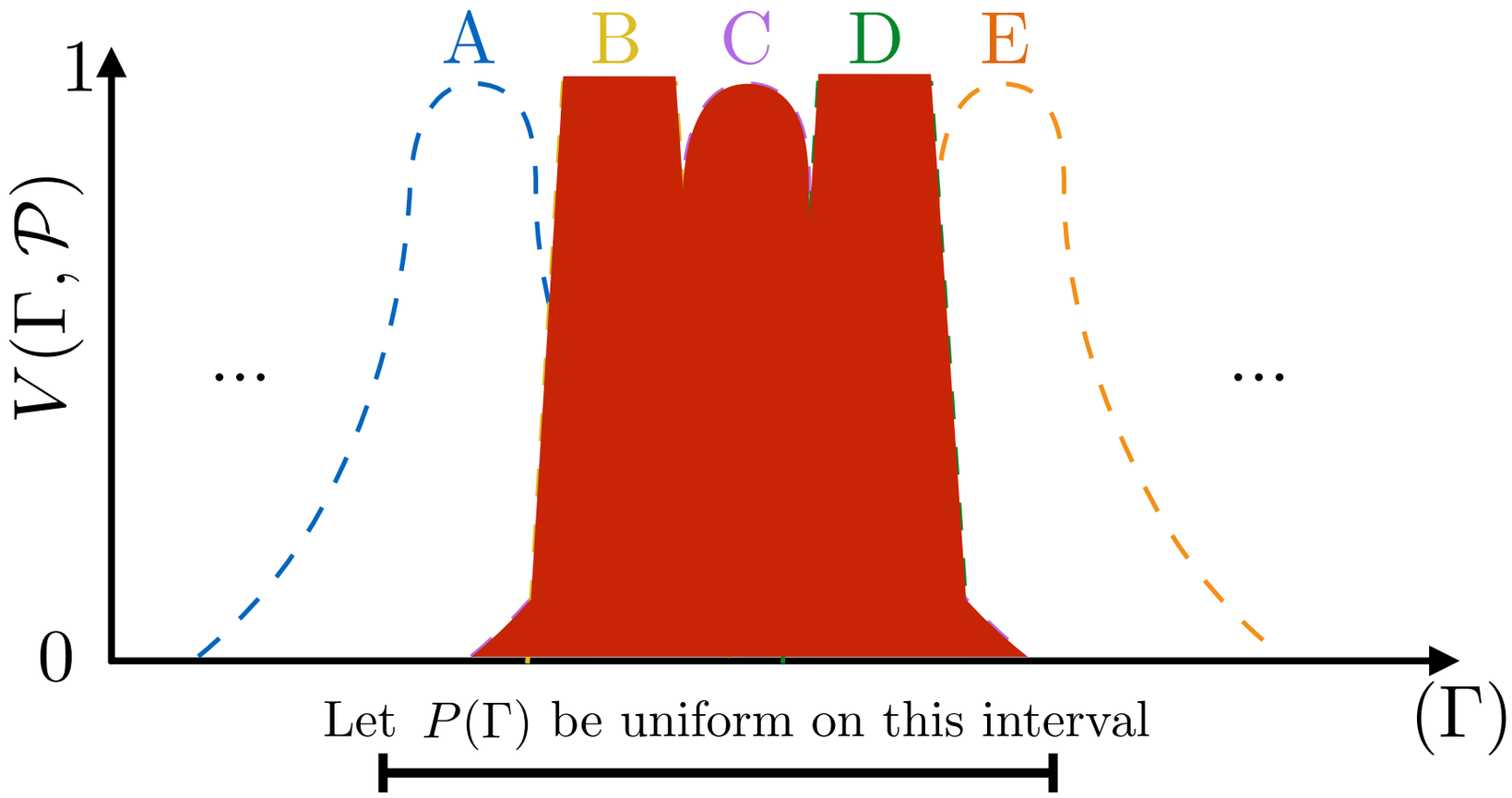
Since predicting a single planner is difficult,

hedge our bets by predicting an ensemble of planners



Simple setting: Select a static ensemble of size 3

$$\max_{\mathcal{P}_1,\mathcal{P}_2,\mathcal{P}_3} \mathbb{E}_{\Gamma\sim P(\Gamma)} \max(V(\Gamma,\mathcal{P}_1), V(\Gamma,\mathcal{P}_2), V(\Gamma,\mathcal{P}_3))$$



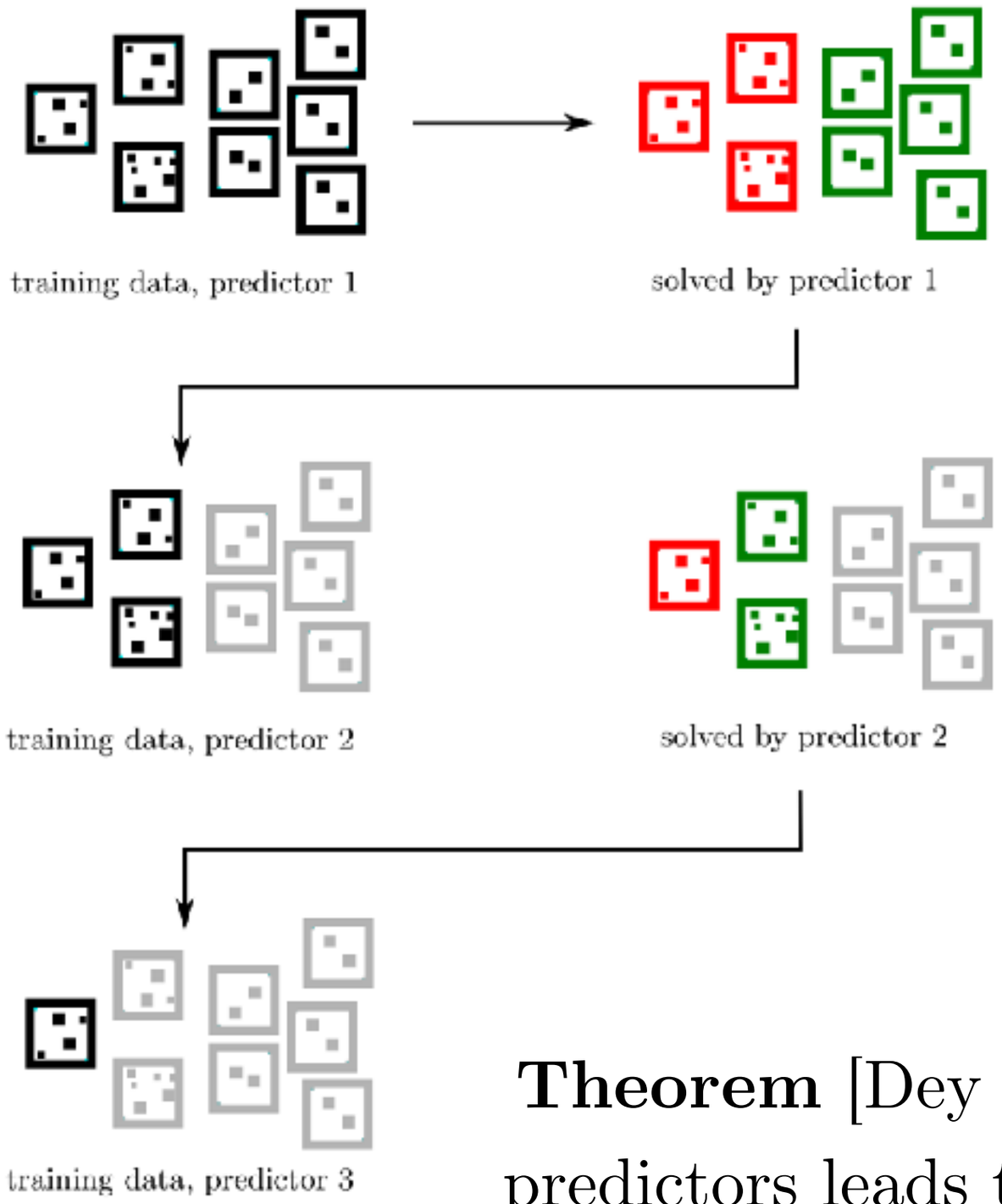
Selected Ensemble ($\mathcal{E}$)
Planner C
Planner B
Planner D

Algorithm: Greedily select planners that maximize gain in score

Theorem: Greedy maximization of monotone submodular function is near-optimal

$$\mathbb{E}_{\Gamma\sim P(\Gamma)} V(\mathcal{E}_{\text{greedy}}) \geq \left(1-\frac{1}{e}\right) \mathbb{E}_{\Gamma\sim P(\Gamma)} V(\mathcal{E}^*)$$ (Krause et al.'12)

Greedily stack cost-sensitive classifiers [ICRA'15,'16]



Train predictor 1 to classify on all problems

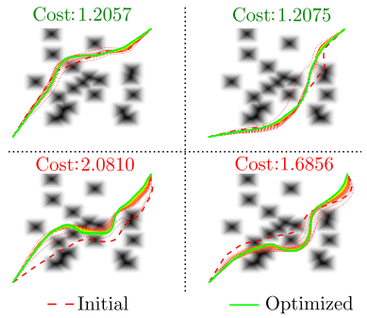
Train predictor 2 to on problems predictor 1 fails to solve

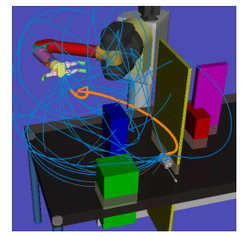
Deeper predictors focus on unsolved (hard) problems

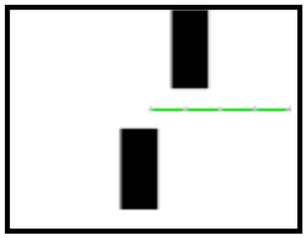
Theorem [Dey et al.'13] Greedily training predictors leads to a near-optimal sequence

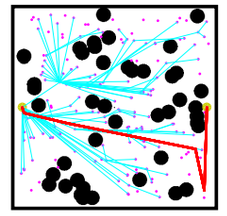
List prediction significantly reduces empirical risk

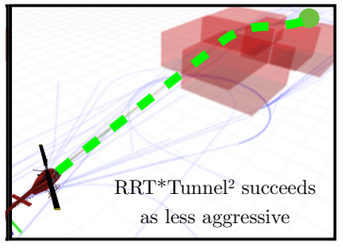
Learner	Single Element	Ensemble (size: 3)	
hinge-loss + linear	0.1073	0.0715	2D traj opt seed prediction
square-loss + linear	0.1106	0.0721	
hinge-loss + linear	15.454	3.6085	7D traj opt seed prediction
square-loss + linear	13.013	3.6799	
hinge-loss + linear	0.0976	0.0325	4D arm planner heuristic prediction
square-loss + linear	0.0933	0.0360	
hinge-loss + linear	0.2222	0.0222	2D geometric planner prediction
square-loss + linear	0.2281	0.0281	
hinge-loss + linear	0.104	0.035	Helicopter planner prediction
square-loss + linear	0.120	0.055	
random forest	0.101	0.021	



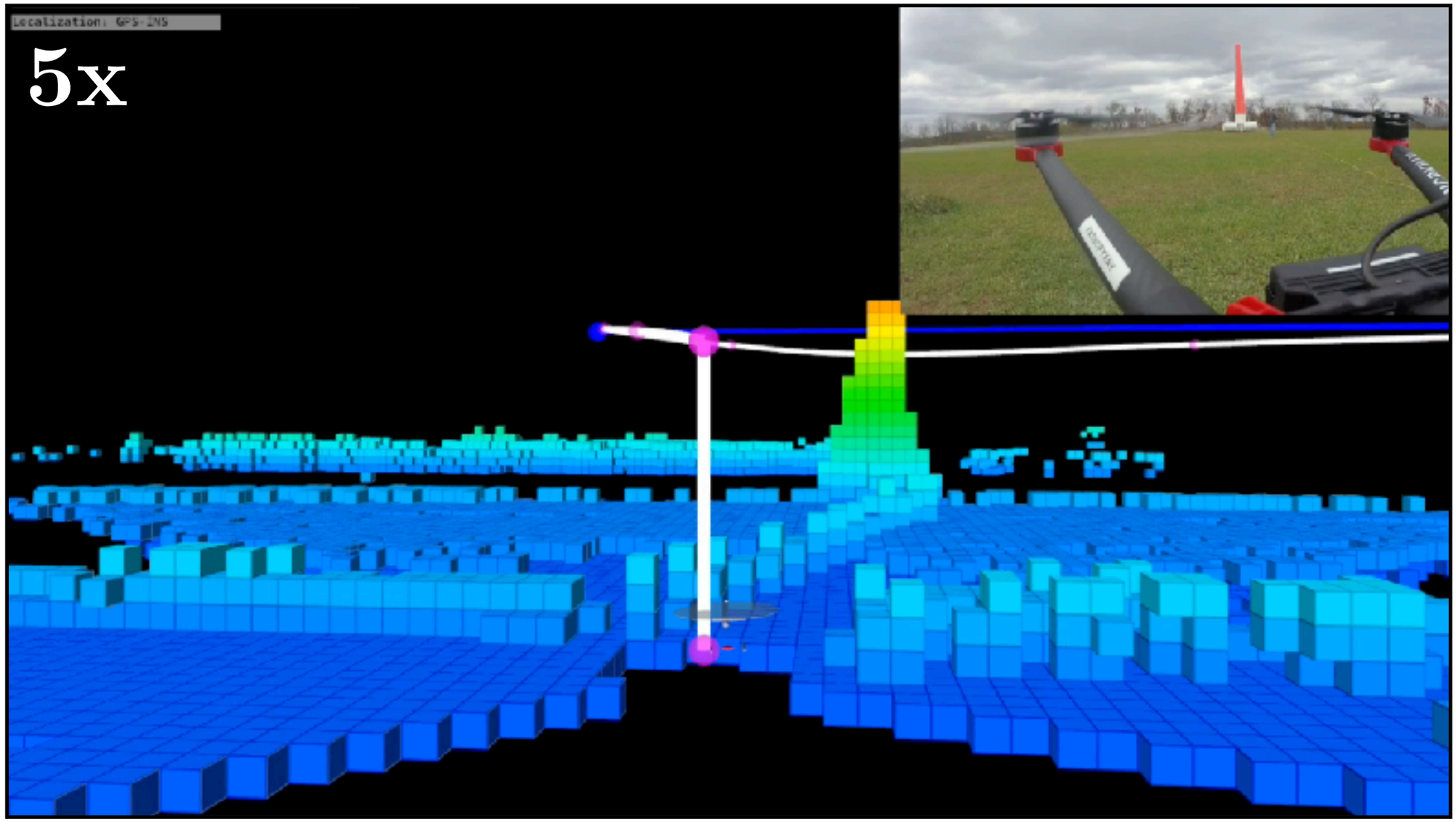


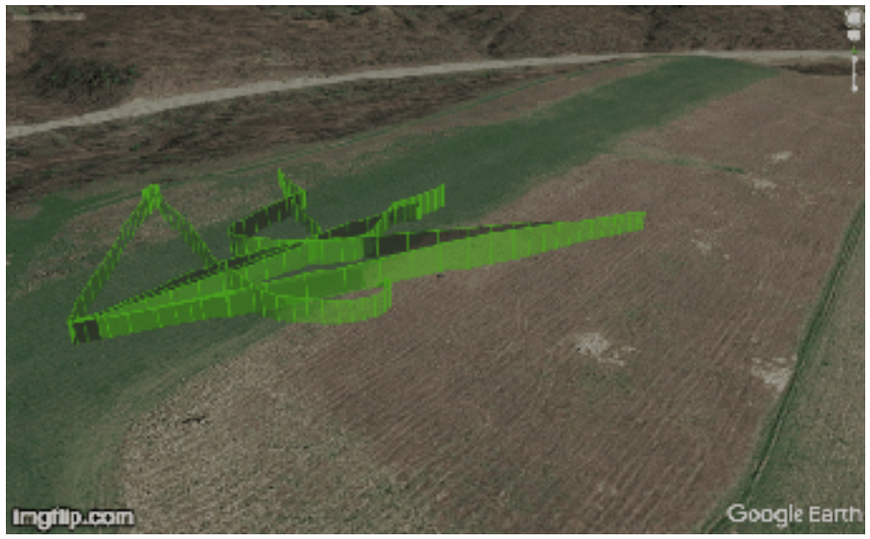






Application: Large UAV flying long durations

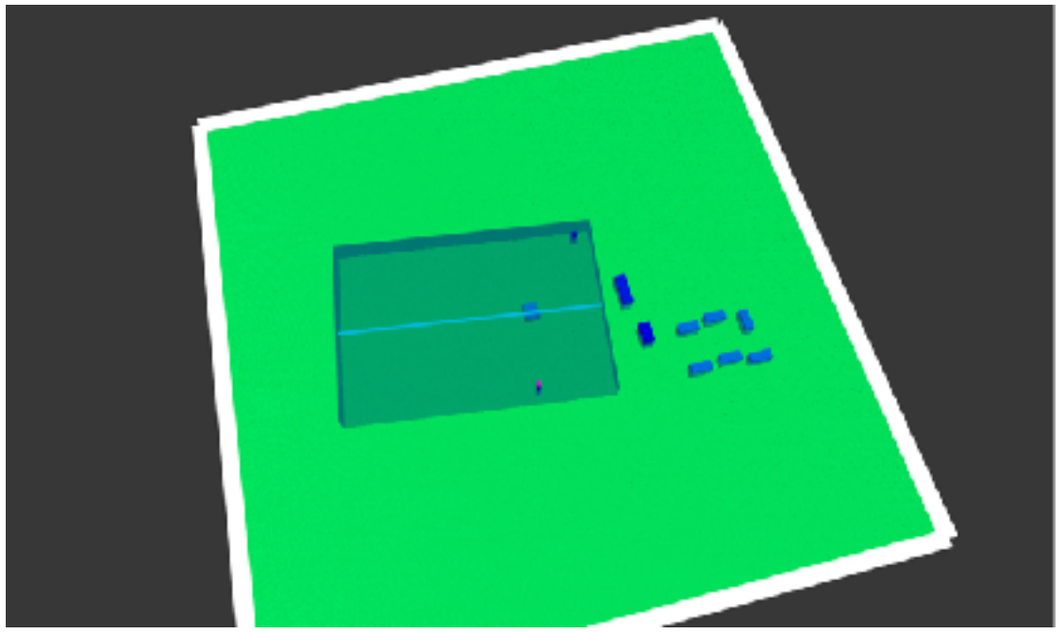




Total Time	Total Distance	Top Speed	Missions
151.9 min	15.833 km	10 m/s	45

Success: 95.56% (compared to 64% for baseline)

Training in simulation



(Monte-carlo sampling of planning problems)

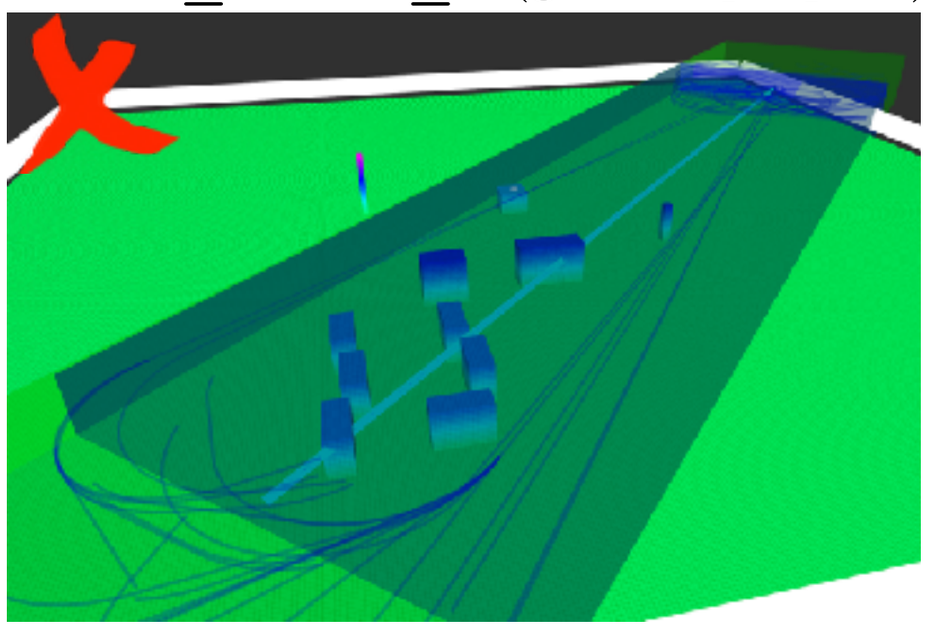
Planner library

General purpose planners	rrtstar_tunnel_1	
	rrtstar_tunnel_2	
	bitstar_1	
	informed_rrtstar_1	
Precision planners	single_detour_1	
	single_detour_2	
	double_detour_1	
	double_detour_2	

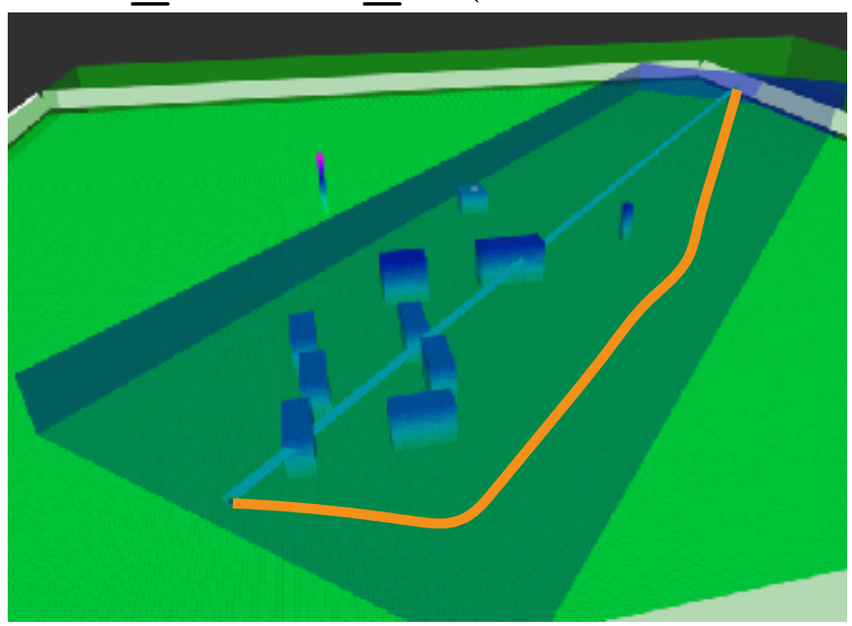
Training performance

Data (Train / Test)	1000 (70 / 30)
Number of planners	8
Feature dim	10
Risk (ensemble: 1)	0.20
Risk (ensemble: 2)	0.08

rrtstar_tunnel_1 (general purpose)

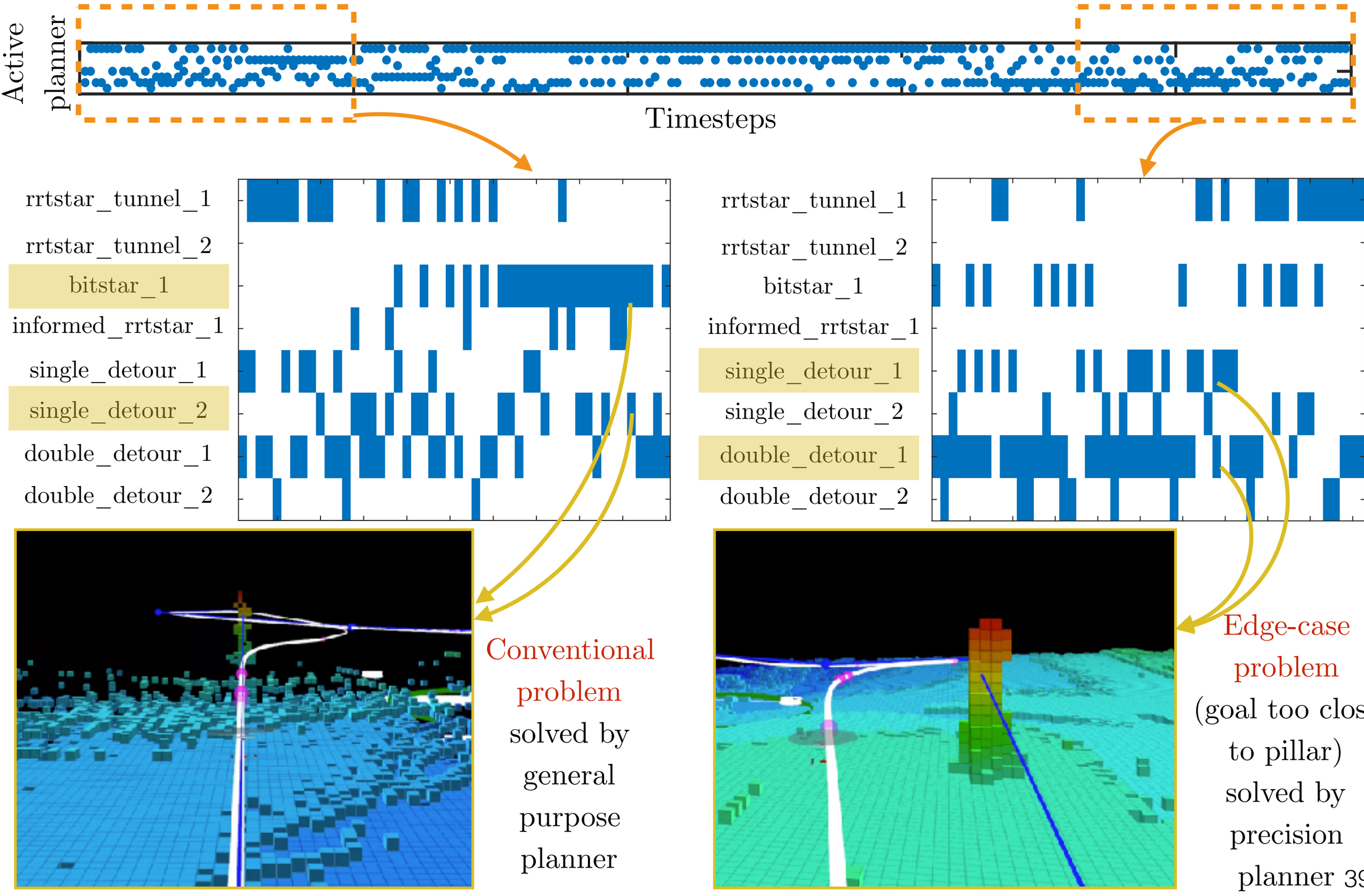


double_detour_1 (precision planner)



Edge-cases which only a precision planner solves

Analysis of test time performance

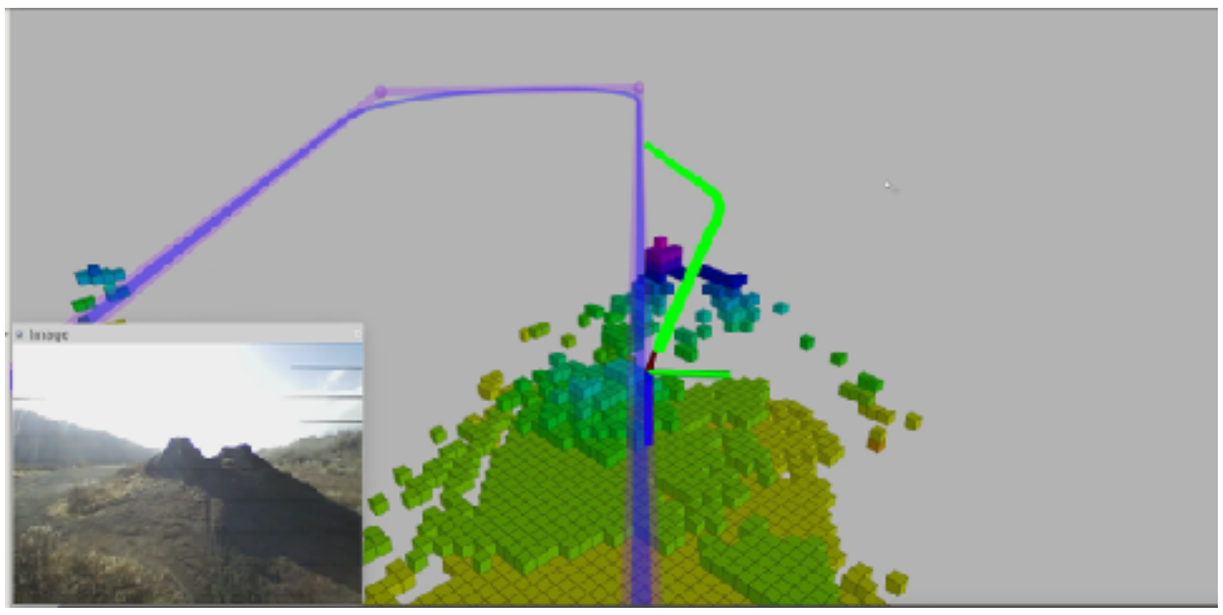


Conventional problem solved by general purpose planner

Edge-case problem (goal too close to pillar) solved by precision planner

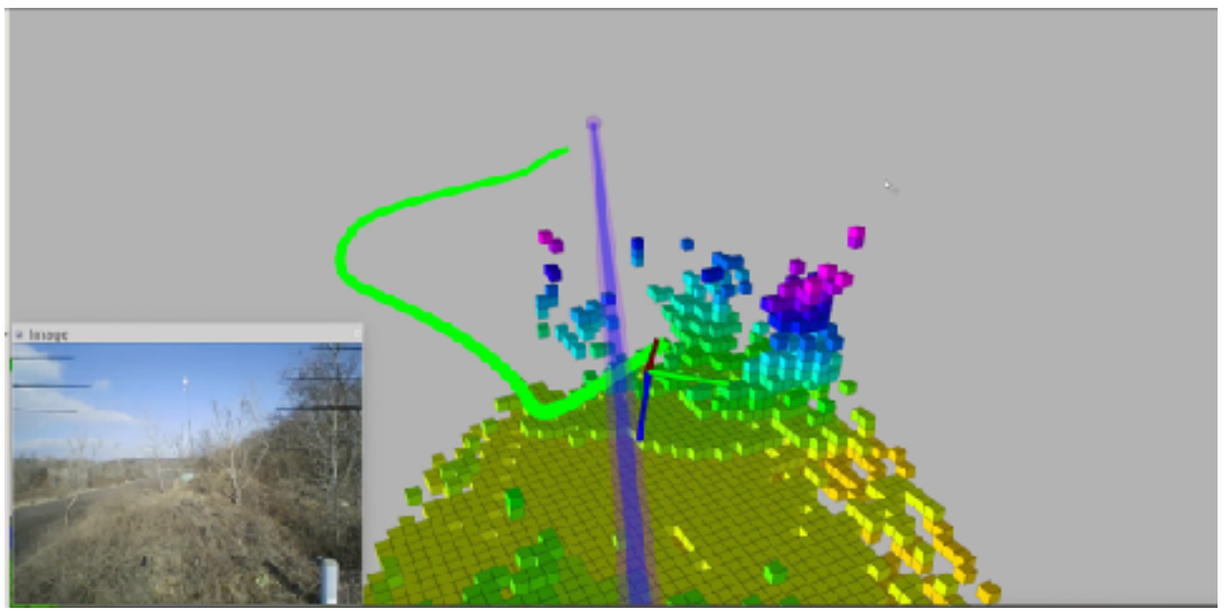
Platform 2: Small UAV flying in diverse scenarios

(Choudhury, Maeta, Dugar, MacAllister, Scherer)



(Higher speed, curvature constraints, more clearance)

Ensemble: double_detour_1, rrtstar_tunnel_1



(Lower speed, unconstrained, less clearance)

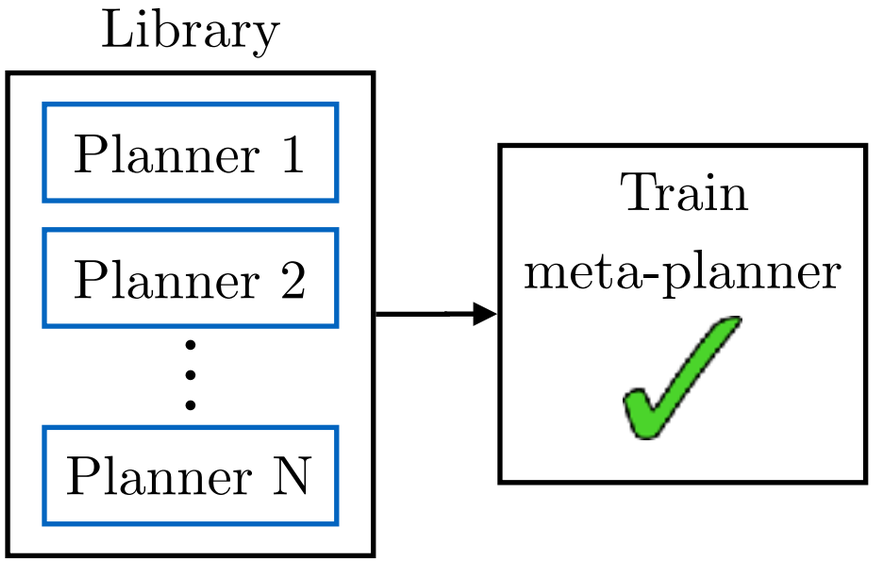
Ensemble: astar_3, bitstar_3

Approach outline

Q1. Framework to assemble real-time planners?

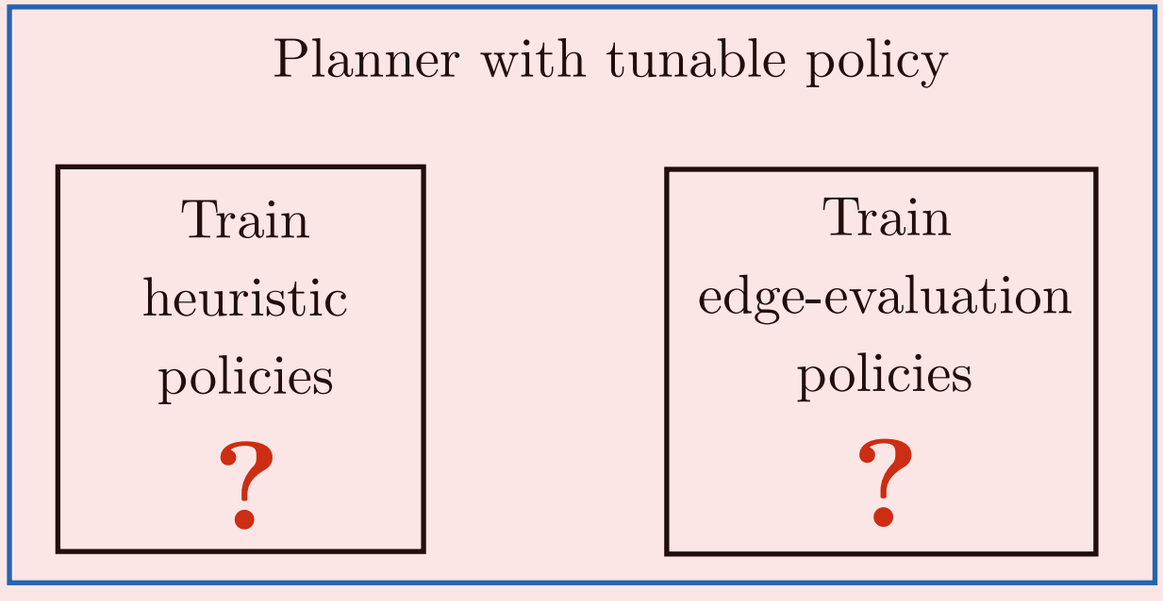
Expert planner with custom implicit graphs

Q2. (Black-box adaptation)
Select planners from a library?

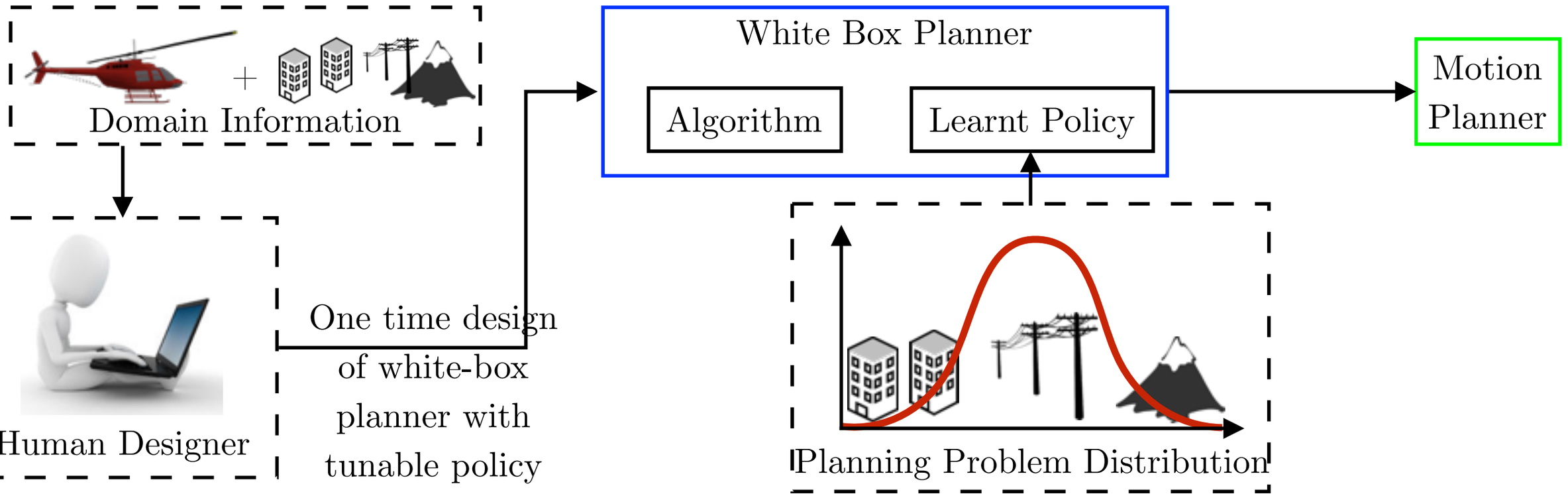


Greedily train an ensemble

Q3. (White-box adaptation)
Directly learn planning policies?



White-box adaptive planning paradigm



How can we train a planning policy?

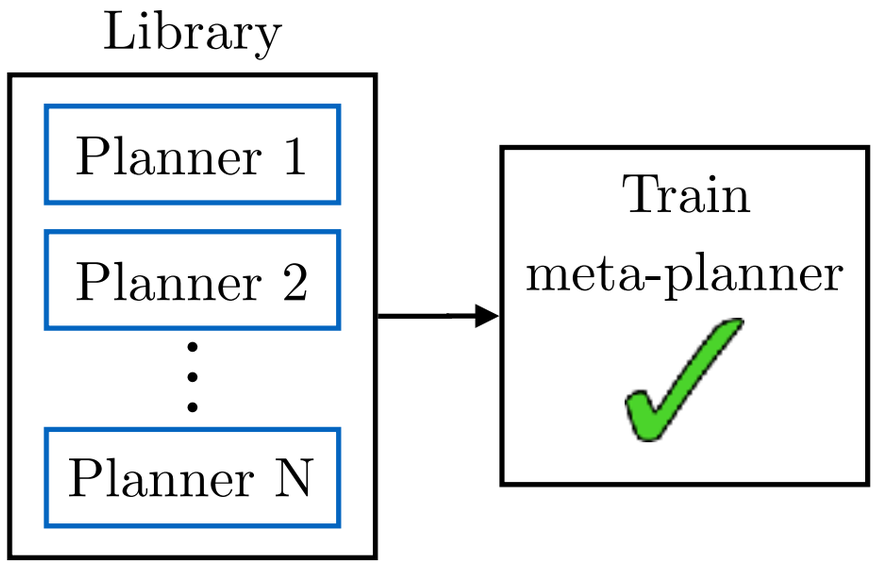
(E.g. a heuristic policy, a sampling policy, a collision-checking policy ..)

Approach outline

Q1. Framework to assemble real-time planners?

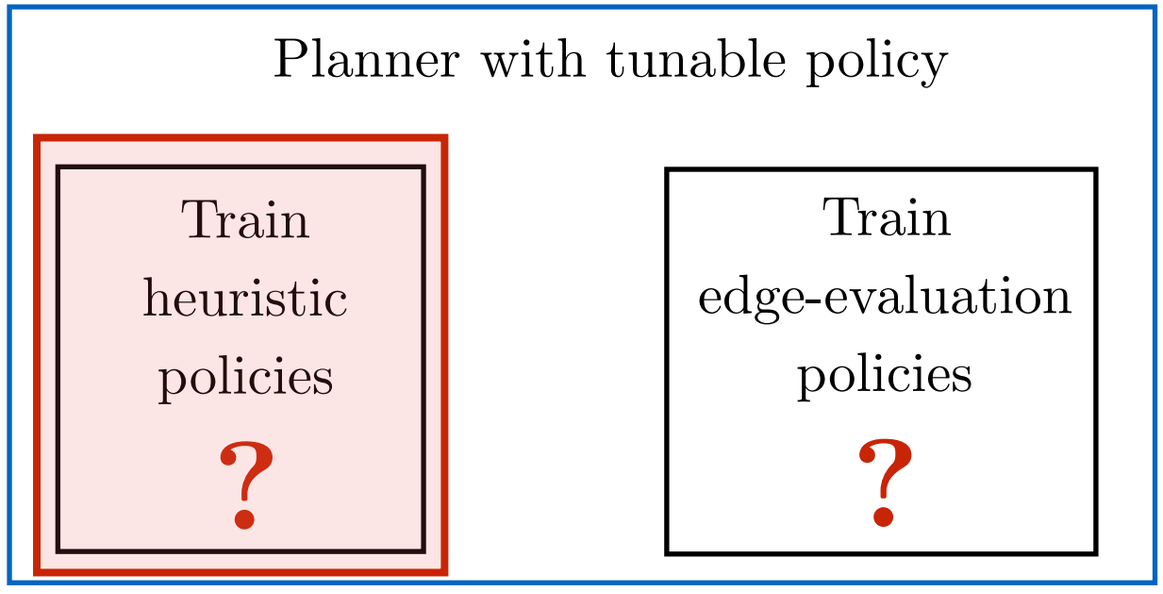
Expert planner with custom implicit graphs

Q2. (Black-box adaptation)
Select planners from a library?

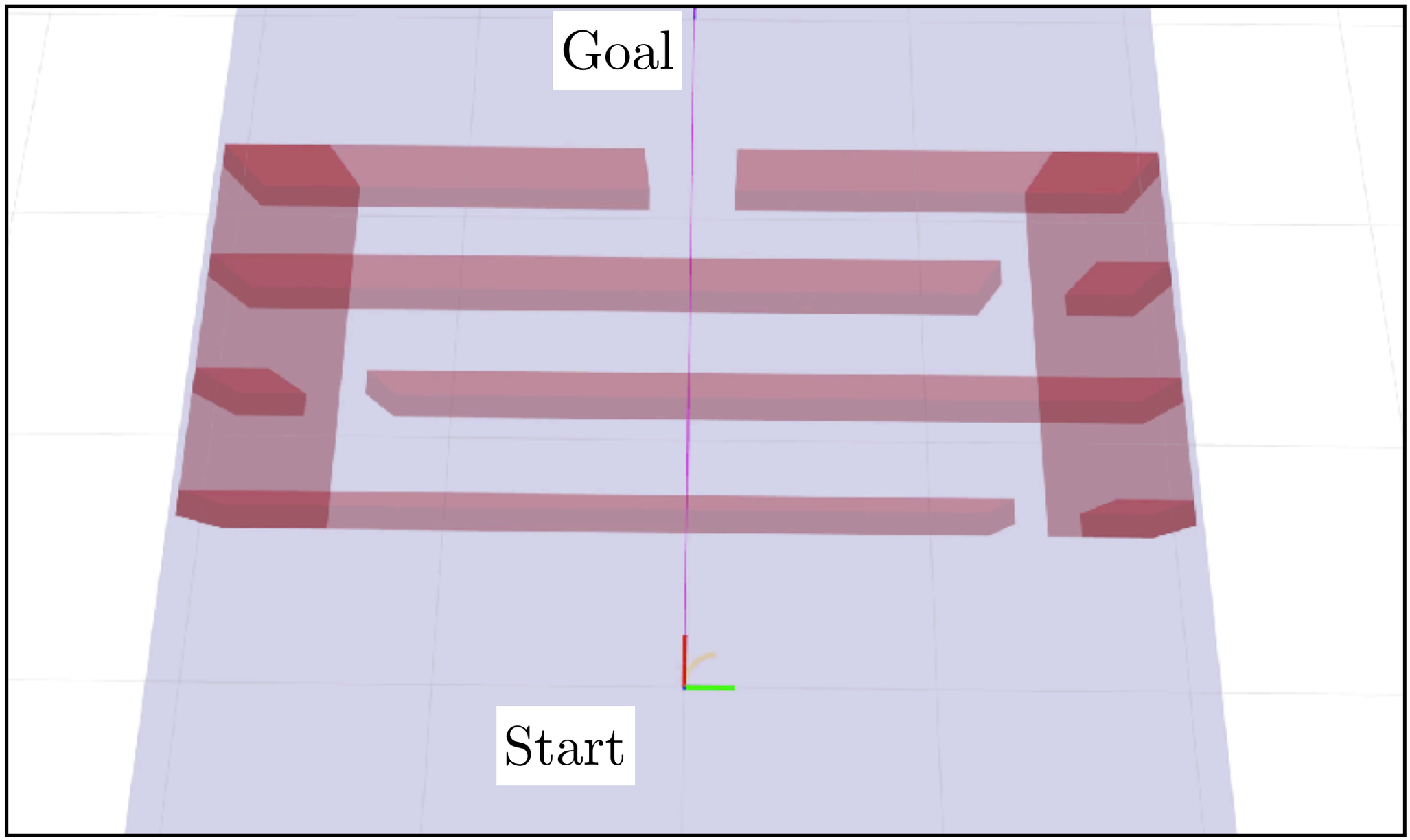


Greedily train an ensemble

Q3. (White-box adaptation)
Directly learn planning policies?



Heuristic policies in search based planning



Objective: Guide a search tree from start to goal to find a feasible path

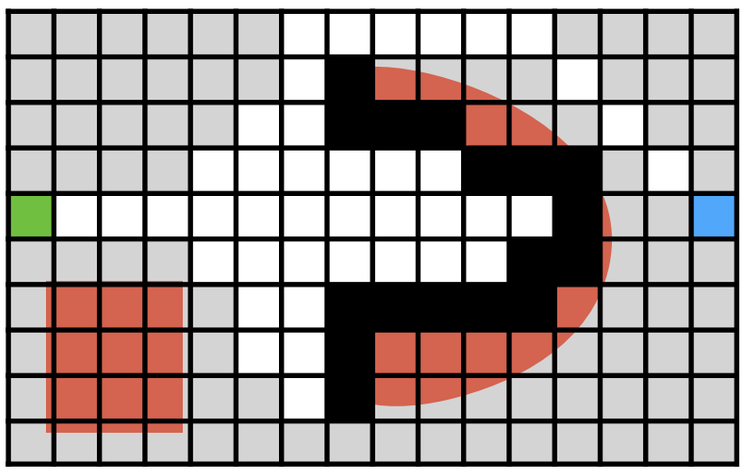
Why does a heuristic need to be adaptive?

Prior work has mainly focussed on bounding solution quality
by defining heuristics as distance estimate of cost-to-go (Pohl'70, Pearl'84)

Compute estimates using relaxation-based (Likhachev et al.'09, Dolgov et al.'08)
or learning-based (Xu et al.'07, Garrett et al.'16, Aine et al.'15, Paden et al.'16) approaches

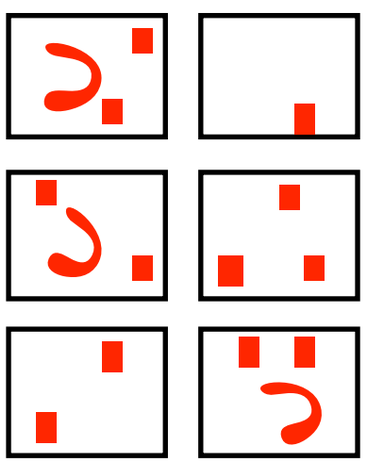
Problem: Small estimation error leads to excessive expansions

Inflated Euclidean Heuristic

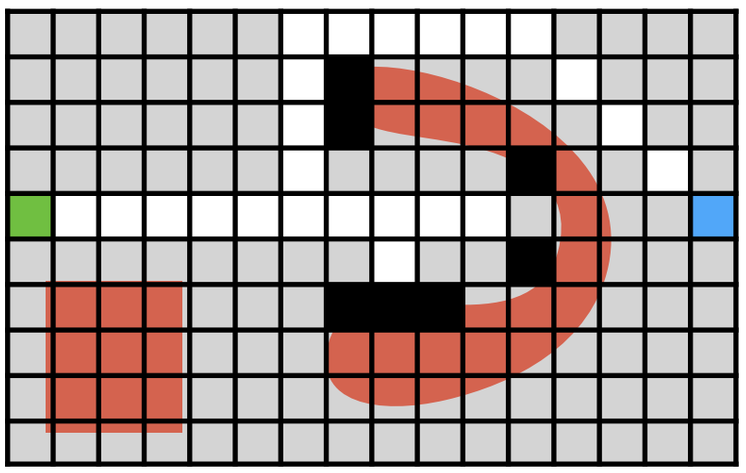


Heuristic gets trapped
in 'bug trap' due to greediness

Learnt Heuristic Policy

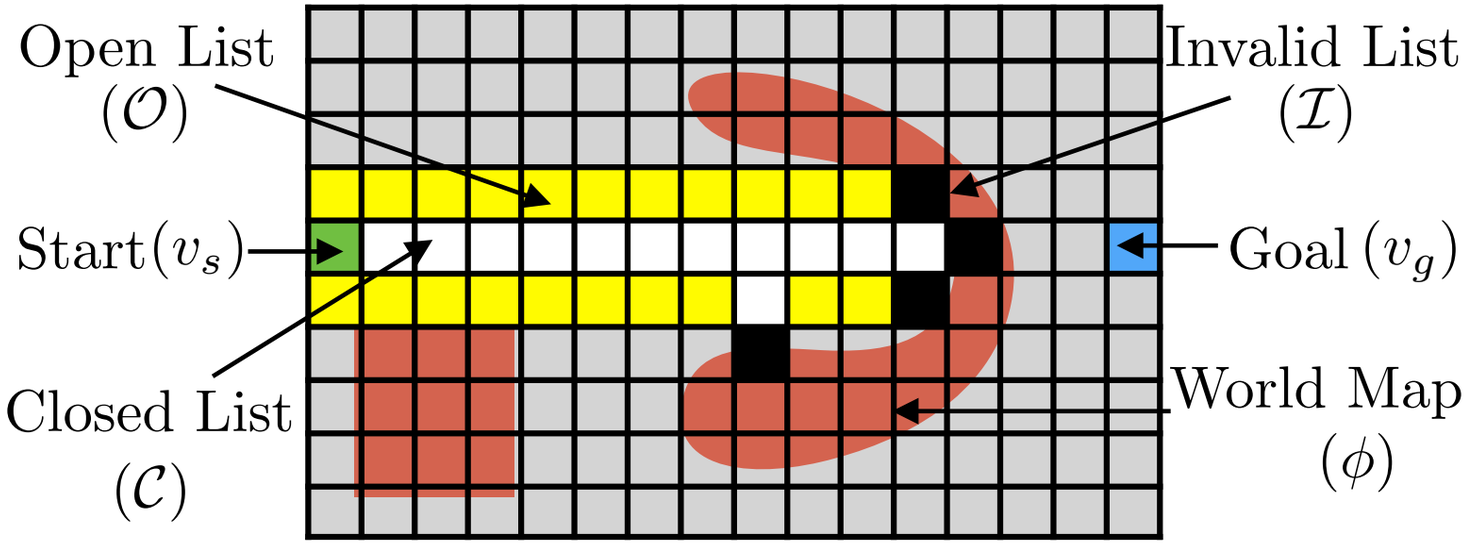


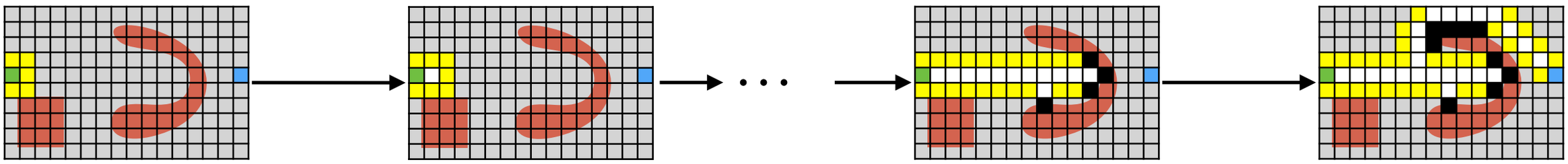
Worlds with
'bug traps'



Heuristic does not get trapped,
searches along periphery

Problem: Minimize expansions in BFS

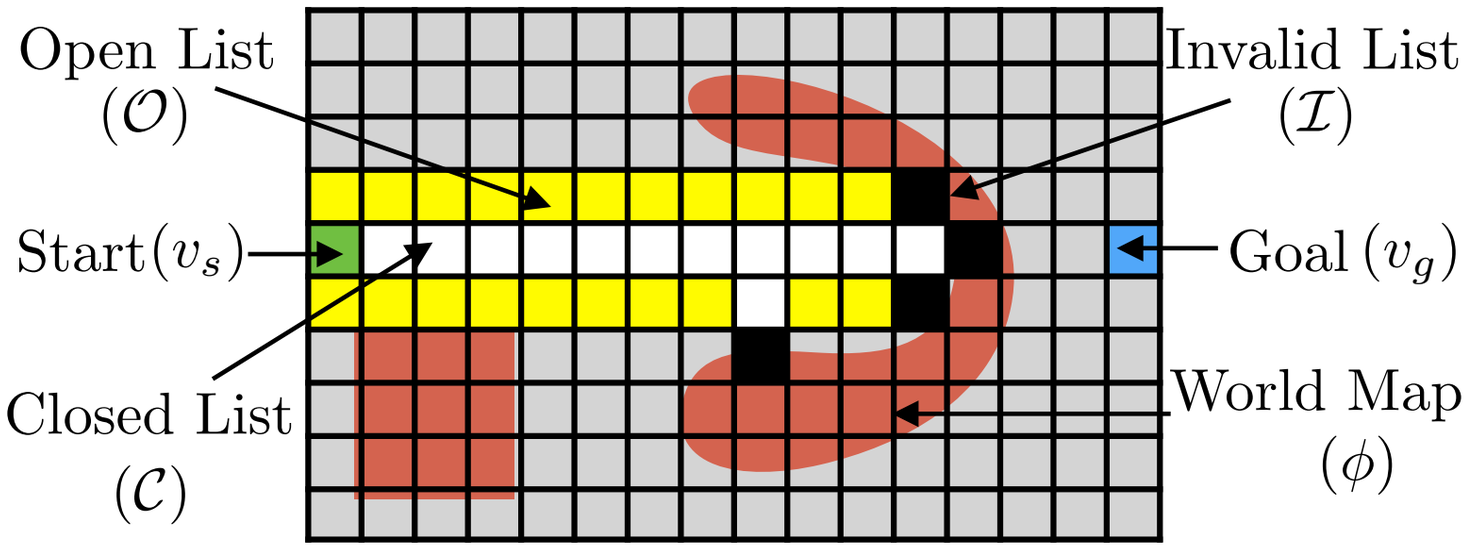




Algorithm `BestFirstSearch`(v_s, v_g, ϕ)

1: $\mathcal{O} \leftarrow v_s,\ \mathcal{C} \leftarrow \emptyset,\ \mathcal{I} \leftarrow \emptyset$
2: **while** $v_g \notin \mathcal{O}$ **do** [Goal not found]
3: $v \leftarrow$ `Select`$(\mathcal{O})$ [Select node to expand]
4: $(\mathcal{V}_{\text{succ}}, \mathcal{E}_{\text{inv}}) \leftarrow$ `Expand`(v, ϕ) [Expand node]
5: $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{V}_{\text{succ}},\ \mathcal{C} \leftarrow \mathcal{C} \cup v,\ \mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{E}_{\text{inv}}$ [Update list]
6: **Return** `Path` (v_s, v_g)

Problem: Minimize expansions in BFS



Compute a heuristic policy that maps search state to node to expand

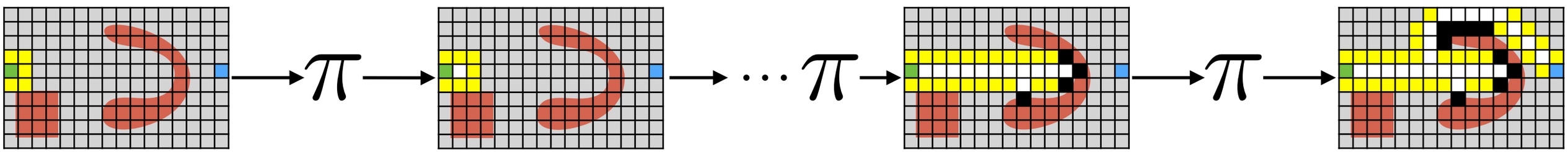
$$\pi(\mathcal{O}, \mathcal{C}, \mathcal{I}) \rightarrow v \in \mathcal{O}$$

Objective: $\min_{\pi} \mathbb{E}_{\phi \sim P(\phi)} \; c(\pi, \phi)$

Distribution over world maps

Number of BFS expansions

Sequential decision making under uncertainty



Cast as a Partially Observable Markov Decision Process (POMDP)

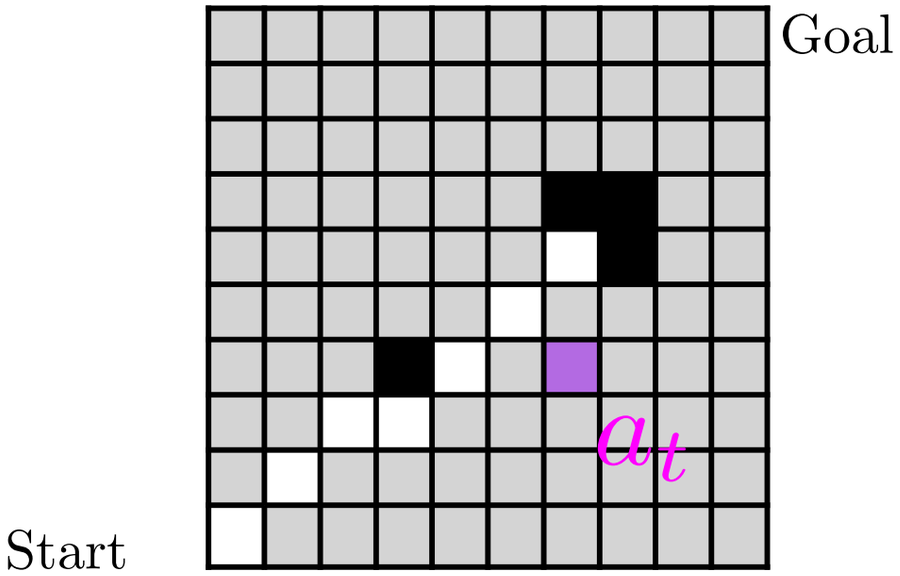
State	$s_t : \{\mathcal{O}_t, \phi\}$	Cost	$\begin{cases} 0 & \text{if } v_g \in \mathcal{O}_t \\ 1 & \text{otherwise} \end{cases}$	Observation	$o_t : \{\mathcal{V}_{\text{succ}}, \mathcal{E}_{\text{inv}}\}$
Action	$a_t : v \in \mathcal{O}_t$			History	$\psi_t : \{\mathcal{O}_t, \mathcal{C}_t, \mathcal{I}_t\}$

POMDP solvers such as POMCP (Silver et al.'10) or DESPOT (Somani et al.'13) require a lot of online effort - we want cheap policies!

Model-free approaches such as Q-learning (Watkins et al.'92) or REINFORCE (Williams et al.'92) are sample inefficient

Key Idea: Imitate a clairvoyant oracle

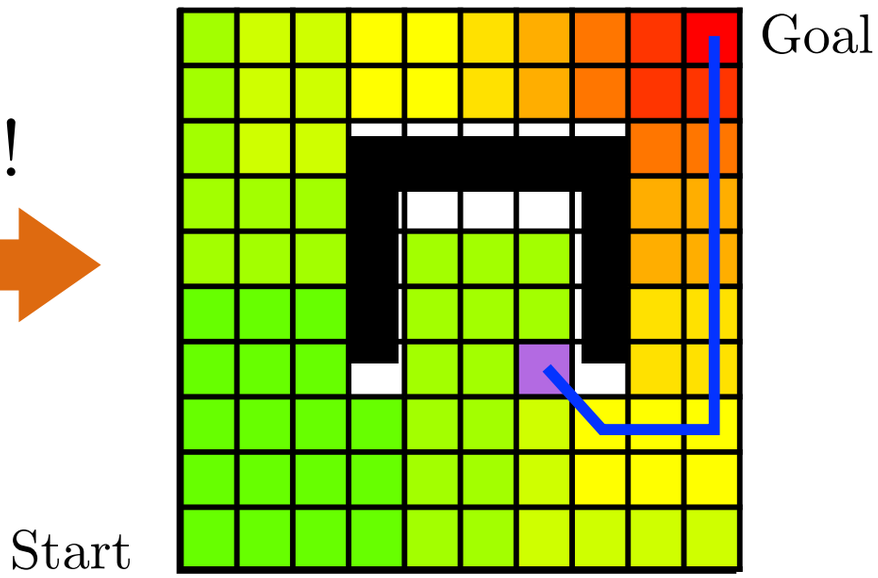
Learner (Q-estimator)



Estimates action
value from history

Imitate!

Oracle (Backwards Dijkstra)

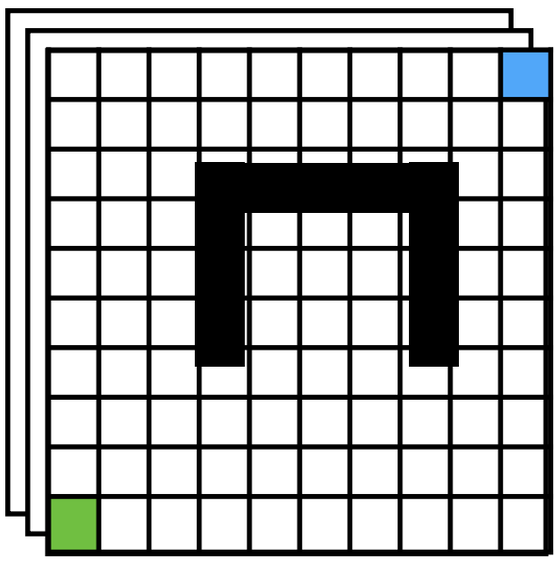


Solves full problem to
get true expansions-to-go

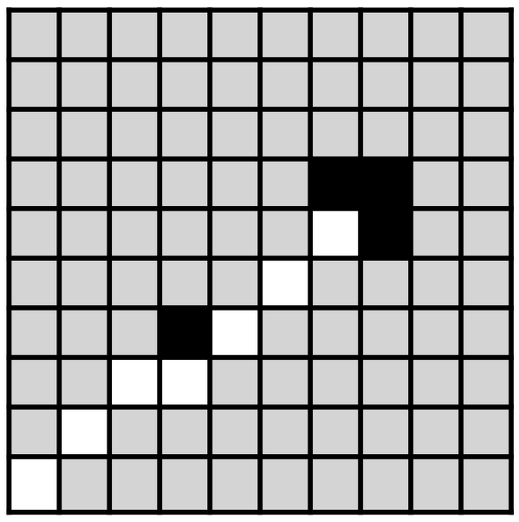
Oracle is "clairvoyant" as it can process the whole world map

We have shown this to be an effective strategy applicable to other POMDP problems such as information gathering [ICRA'16, RSS'17]

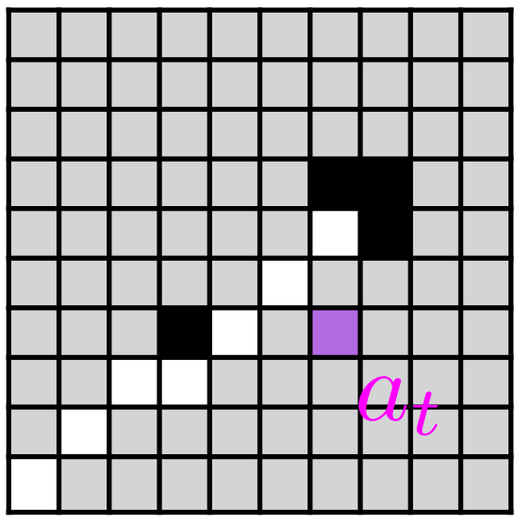
SAIL: Search as Imitation Learning [CoRL'17]



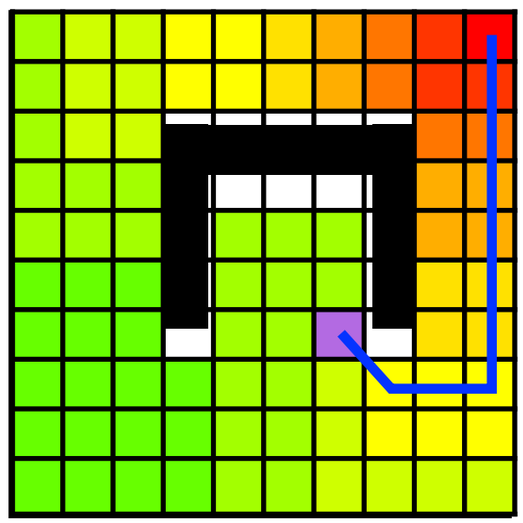
Sample a world from database $P(\phi)$



Roll-in a mixture of learner + oracle to get search state $\psi_t : \{\mathcal{O}_t, \mathcal{C}_t, \mathcal{I}_t\}$



Execute random action and extract features $\{\psi_t, a_t\} \rightarrow f_t$



Query oracle to get value $Q^{\pi_{\text{OR}}}$

Aggregate data $\{f_t, Q^{\pi_{\text{OR}}}\}$

Update learner

Using no-regret analysis (Ross and Bagnell 2014)
we can bound learner performance [RSS'17]

SAIL outperforms baselines on several datasets

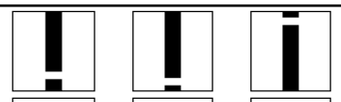
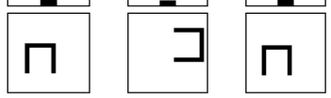
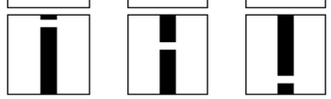
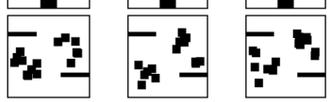
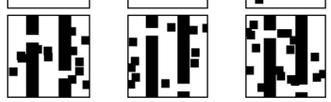
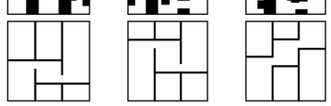
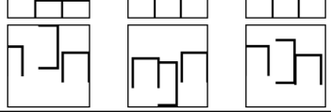
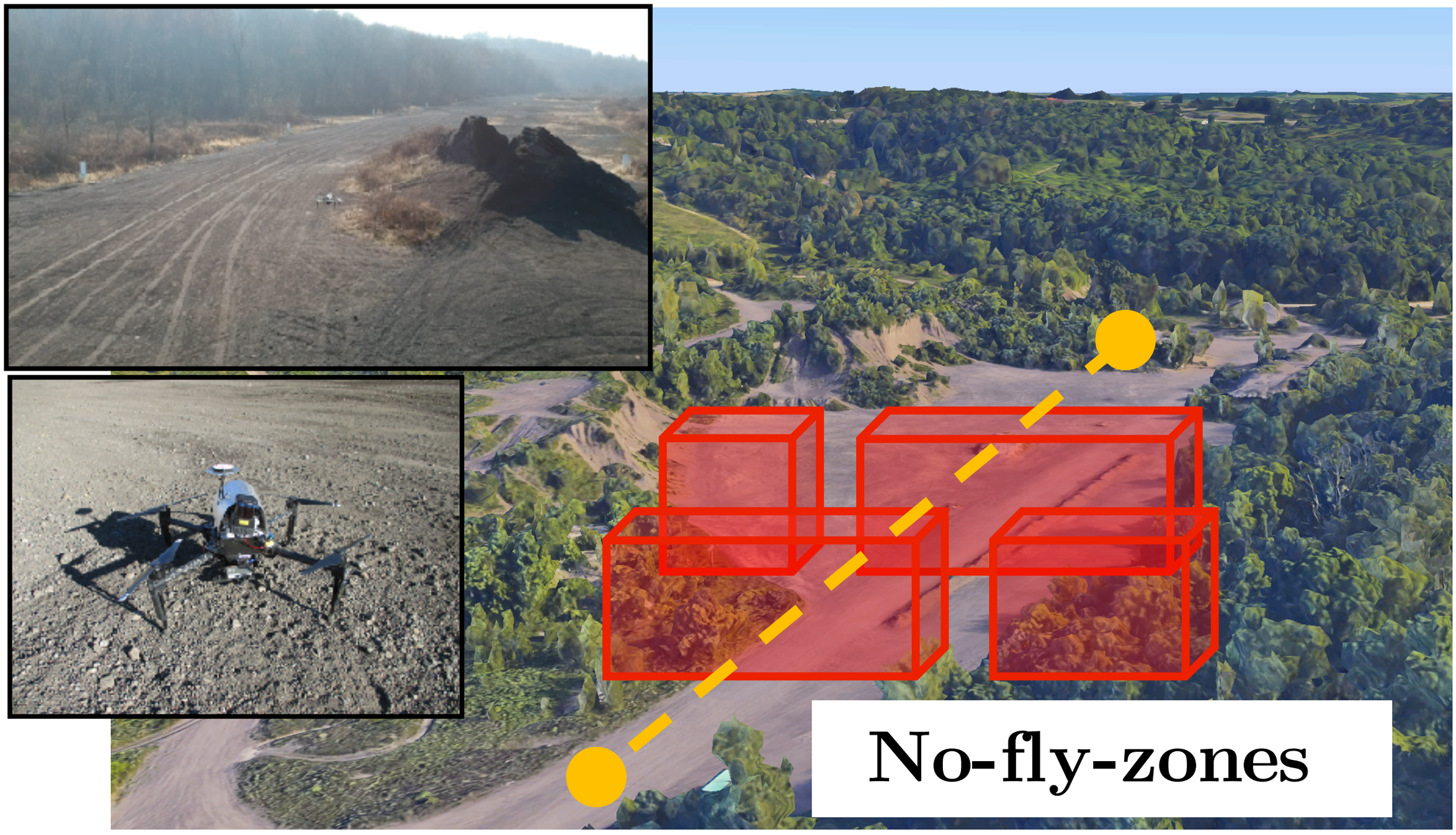
		Learning			Non-learning approaches			
Dataset	**Sample Worlds**	**SAIL**	**SL**	**CEM**	**QL**	$h_{\mathbf{EUC}}$	$h_{\mathbf{MAN}}$	**A***	**MHA***
Alternating Gaps		**0.039**	0.432	0.042	1.000	1.000	1.000	1.000	1.000
Single Bugtrap		0.158	0.214	**0.057**	1.000	0.184	0.192	1.000	0.286
Shifting Gaps		**0.104**	0.464	1.000	1.000	0.506	0.589	1.000	0.804
Forest		**0.036**	0.043	0.048	0.121	0.041	0.043	1.000	0.075
Bugtrap+Forest		**0.147**	0.384	0.182	1.000	0.410	0.337	1.000	0.467
Gaps+Forest		**0.221**	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Mazes		**0.103**	0.238	0.479	0.399	0.185	0.171	1.000	0.279
Multiple Bugtraps		**0.479**	0.480	1.000	0.835	0.648	0.617	1.000	0.876

Table shows average normalized expansions

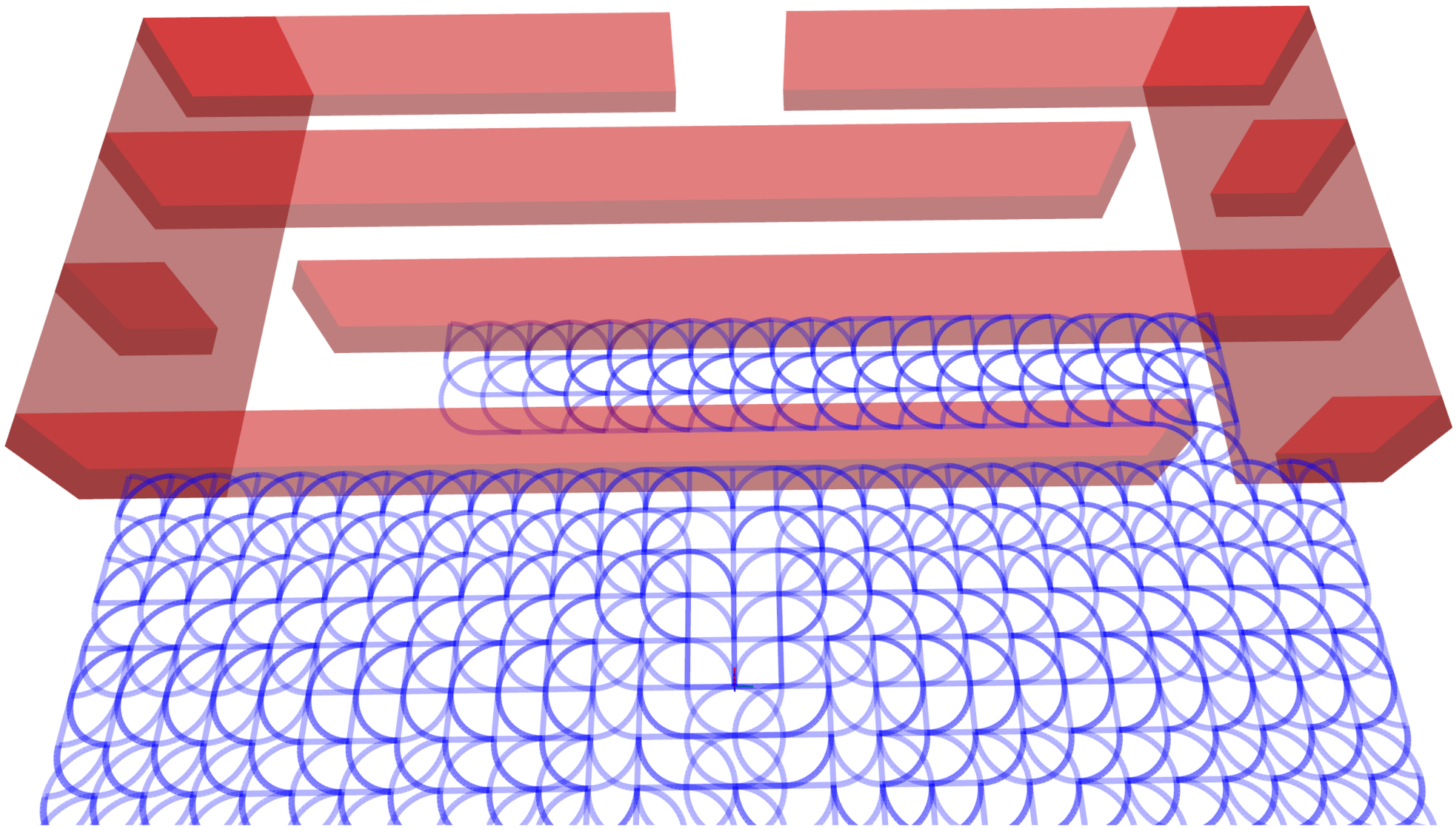
Applying SAIL on real world problems

High speed no-fly-zone avoidance requires

real-time nonholonomic path planning



A* with Dubins heuristic times out

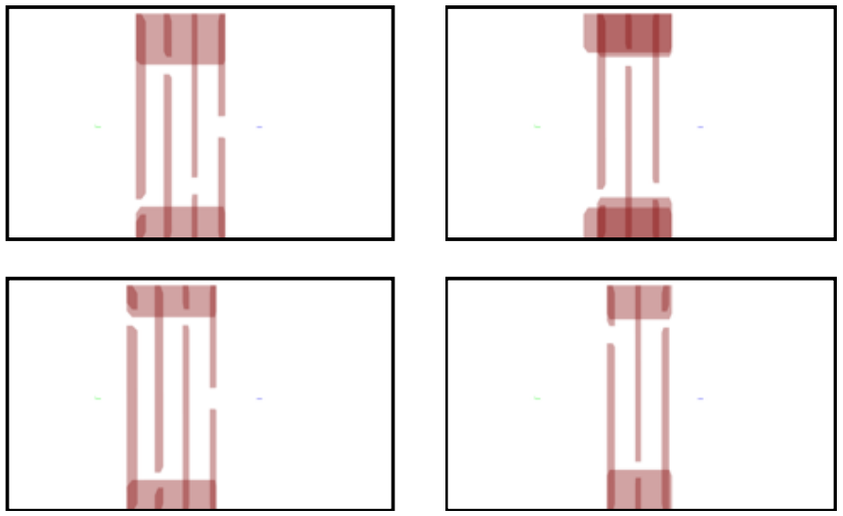


Expands 1910 states in time budget (1000 ms)

Dubins distance is a poor estimate of expansions-to-go

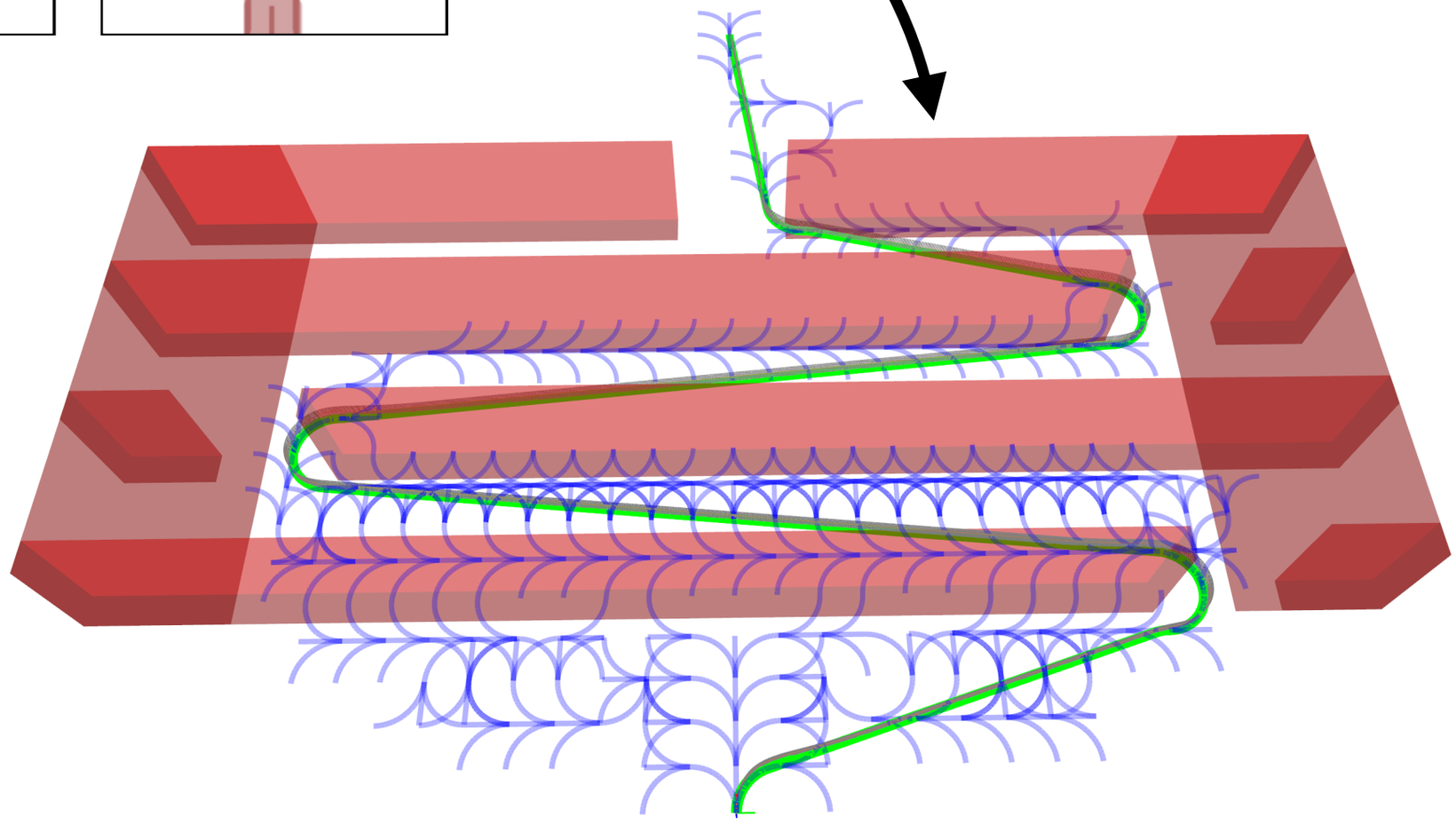
SAIL learns to follow maze wall

Train a policy on maze worlds



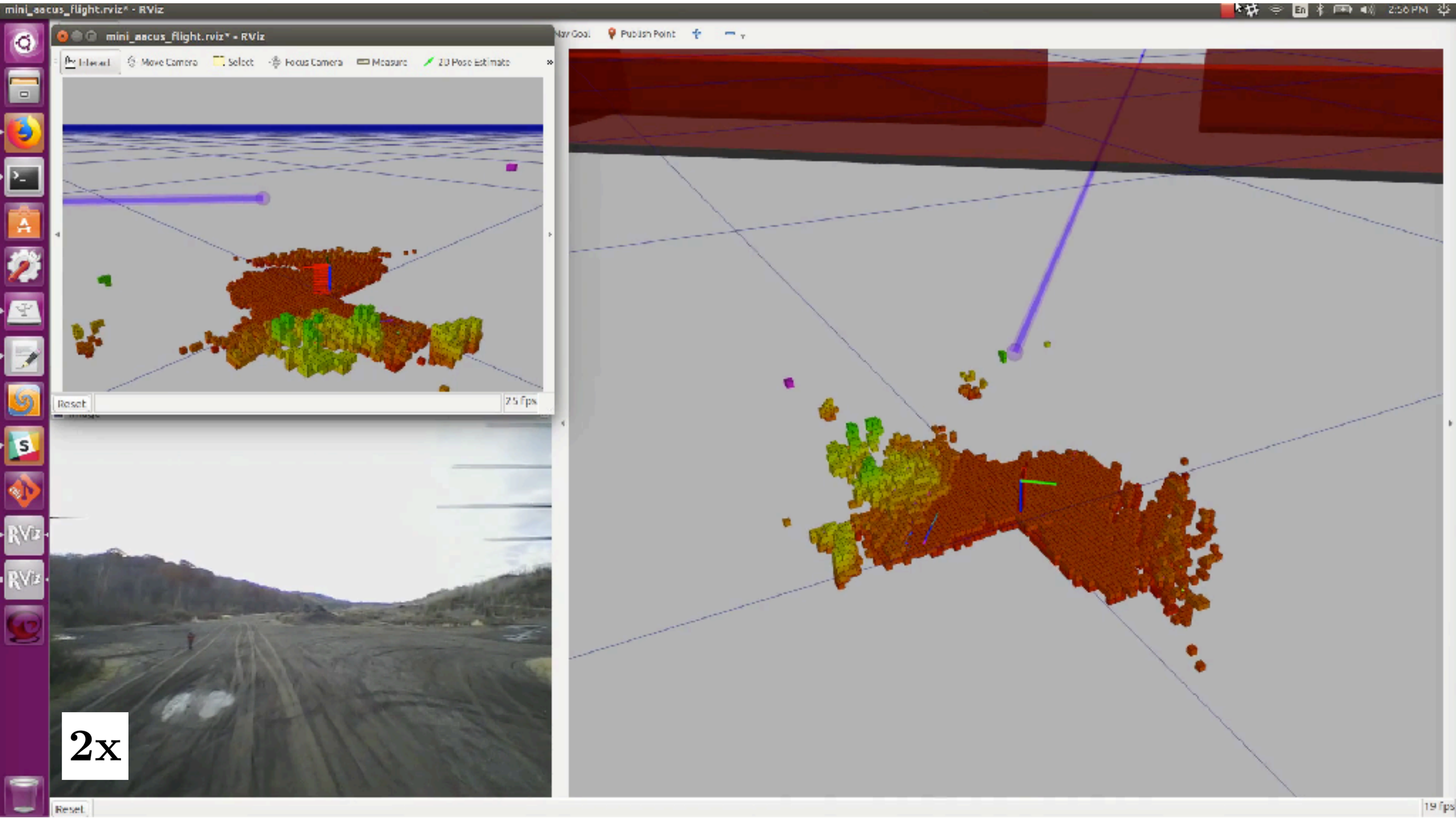
Learnt policy expands

180 states in 120ms

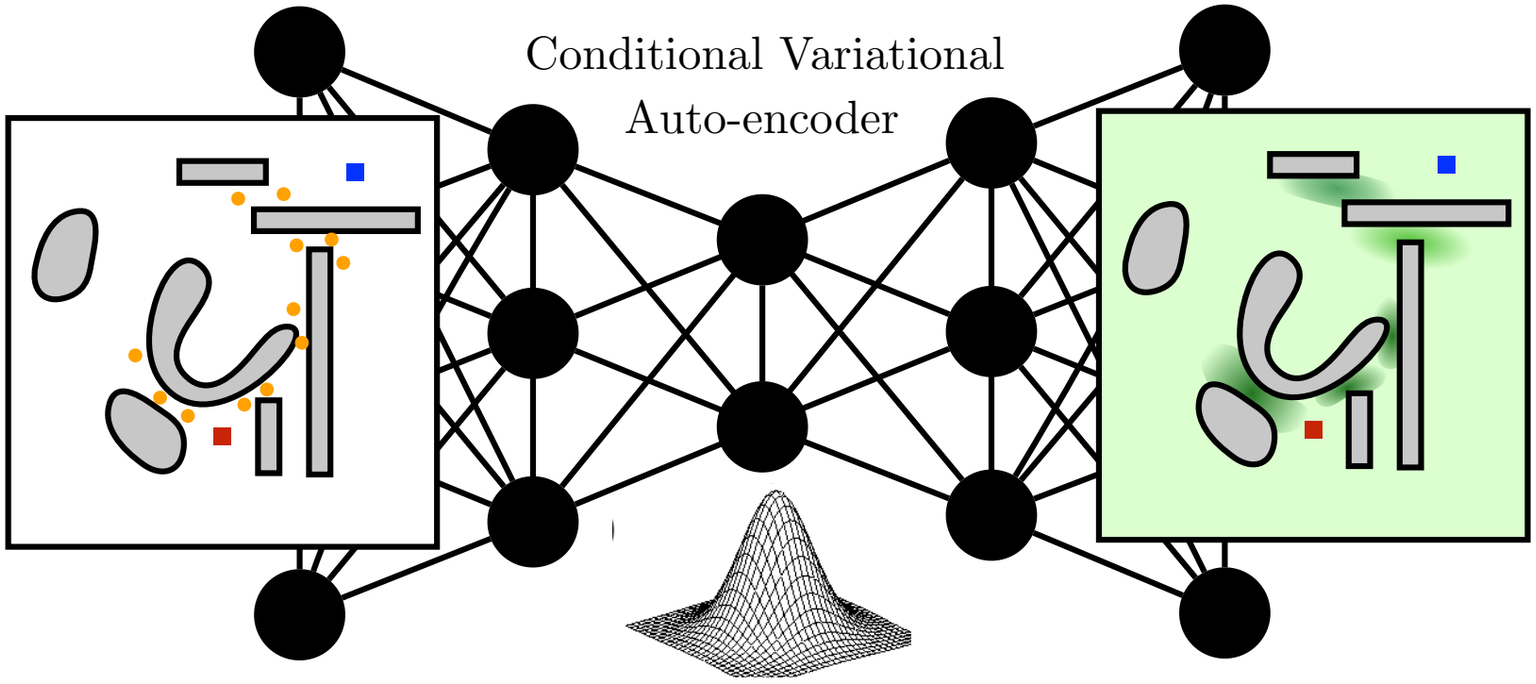


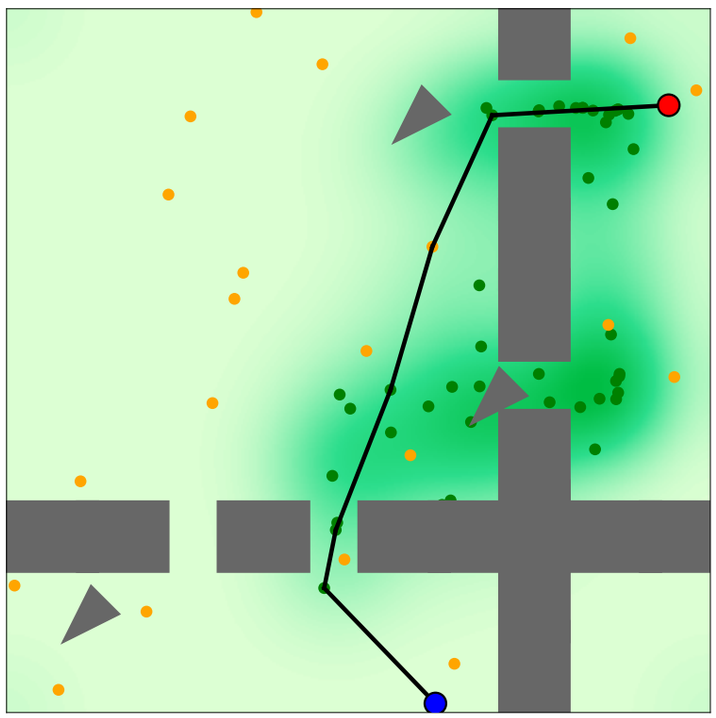
Flight test with onboard execution

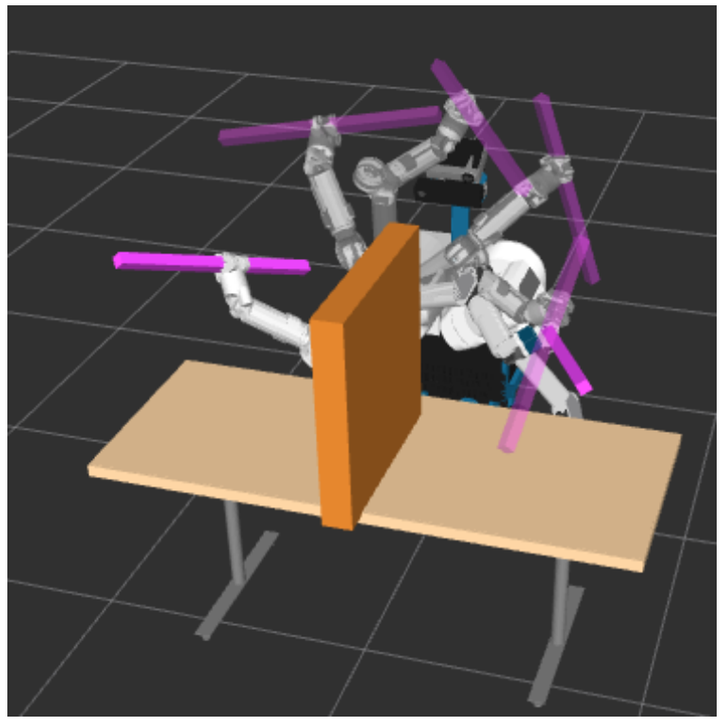
(Choudhury, Bhardwaj, Maeta, Scherer)

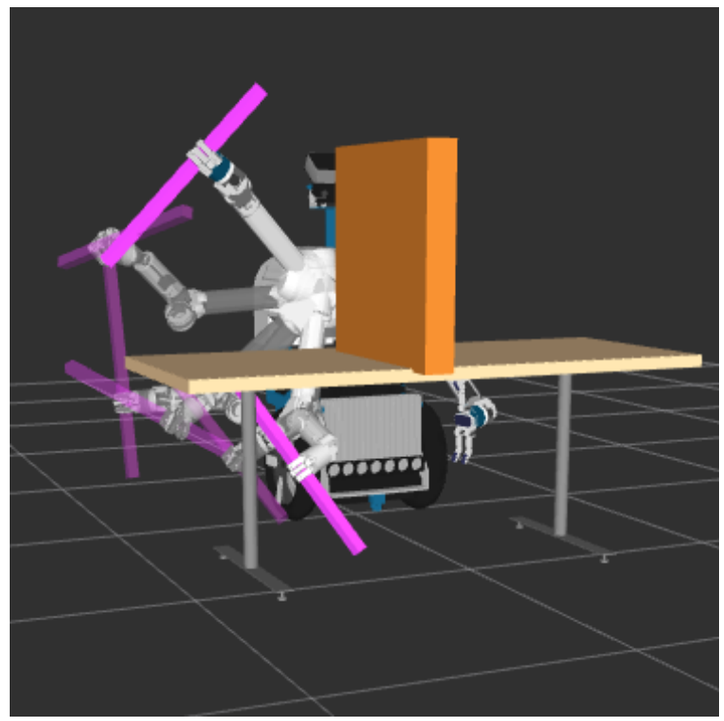


Can we directly learn graphs?









[In submission, IROS '19]

White-box or black-box

White-box framework

Pro:

Efficient single planner

Minimal human involvement

Con:

Powerful policy class

Restricted to underlying graph

Black-box framework

Con:

Explicit context extraction

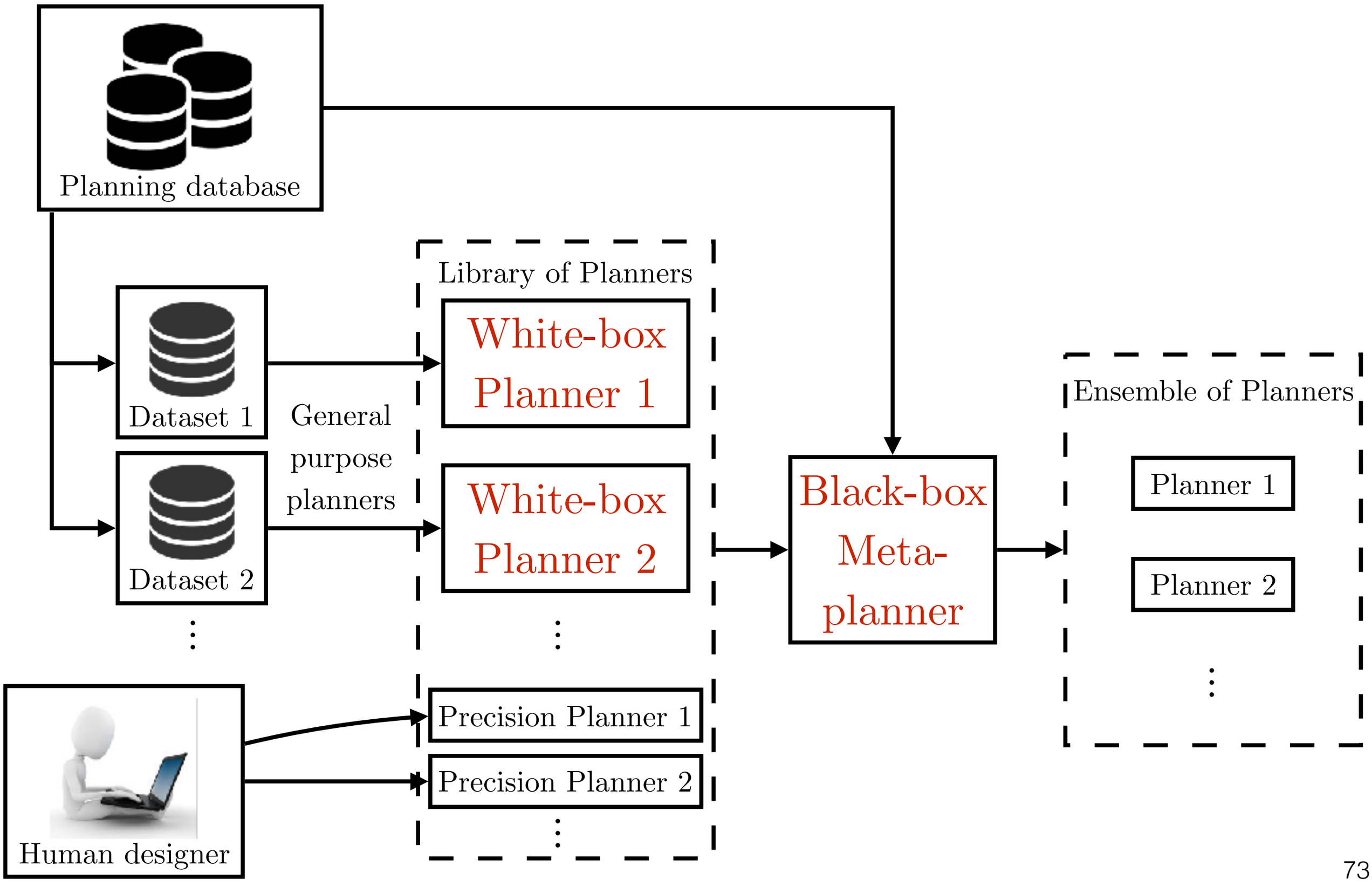
Human to design library

Pro:

Handle edge-cases via ensemble

Arbitrary precision planners

Unified framework for adaptive planners



Thank you!

Acknowledgements