

*16-832 Spring'26*  
*Integrated Planning and Learning*

*Integrating learning into planning:  
speeding up planning*  
*Part 1*

*Maxim Likhachev*  
*Robotics Institute*  
*Carnegie Mellon University*

# Papers to cover

---

- *"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., 2022*
- *"Roadmaps with Gaps over Controllers: Achieving Efficiency in Planning under Dynamics" by Sivaramakrishnan et al., 2024*
- *"Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks" by Takahashi et al., 2019*
- *"Learning Local Heuristics for Search-Based Navigation Planning" by Veerapaneni et al., 2025*

# What?

---

- (Describes) a number of ways to speed up sampling-based motion planning (e.g., RRT, PRM, RRT\*) via incorporating Machine Learning

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Why?

- The run-time can be overly high for complex motion queries (e.g., spaces with narrow passageways, full-body motion planning, planning for systems with complex dynamics constraints)
- The solution quality can often be poor, even after post-processing

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Related Work

---

- N/A here since it is a survey paper

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Assumptions

---

- Reliance on the model used for planning

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

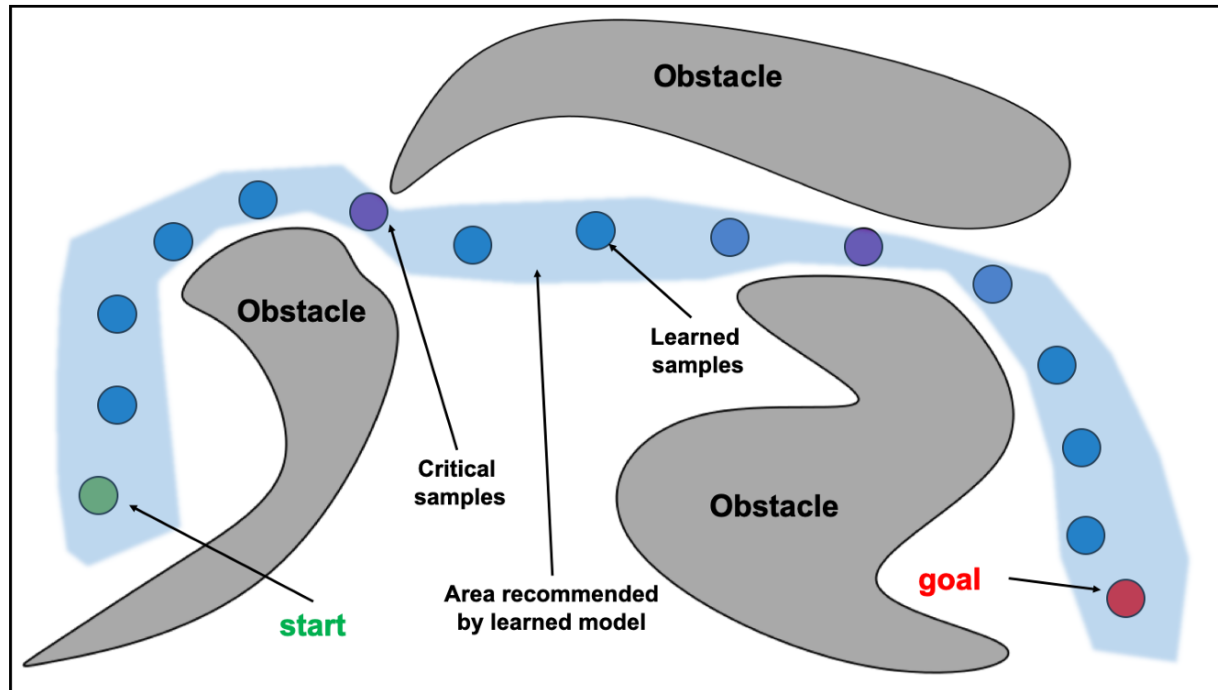
# How?

- Learning to speed up sample generation
- Learning to speed up collision detection
- Learning to speed up distance/cost computation
- Learning to speed up/improve quality of local planning
- Learning the “right” planner parameters/selection
- Learning a lower-dimensional planning representation

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up sample generation

- Goal: Improve quality and efficiency of sampling
  - Modeling the C-space
  - Biasing samples along likely shortest paths



**Figure 3.1:** Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $\mathbb{C}_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up sample generation

- Goal: Improve quality and efficiency of sampling
  - Modeling the C-space
    - Learn probability distributions that model  $C_{free}$  and  $C_{obs}$ . The estimator approximates  $P(\text{sample} \in C_{free})$ . Filters out samples unlikely to be valid. [Arslan and Tsiotras, 2015]
    - Learn models of local manifolds of valid configuration spaces. Use these local models for generating samples and local planning. [Kingston et al., 2019]
    - ...

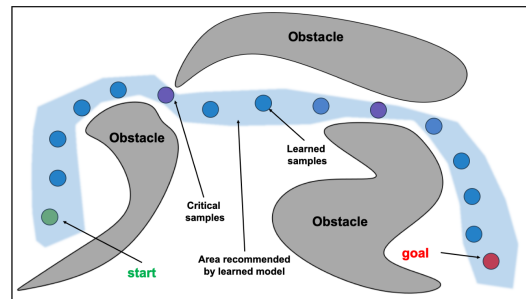


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up sample generation

- Goal: Improve quality and efficiency of sampling
  - biasing samples along likely shortest paths
    - Neural RRT\* [Wang et al., 2020]:
      - Generates large dataset of optimal paths using  $A^*$ .
      - Trains NN to output a probability density where for each cell, its probability mass represents the likelihood of containing a point along an optimal path.
      - The predicted probability distribution is used to generate samples to be used by an RRT\* planner.
      - To provide guarantees, a proportion of samples are generated using uniform sampling.

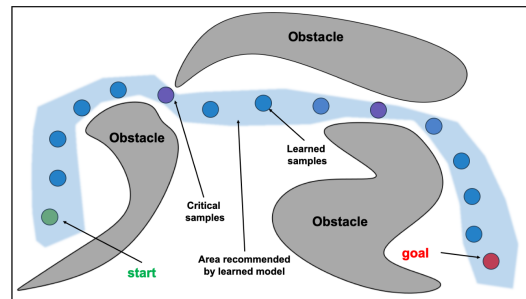


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up sample generation

- Goal: Improve quality and efficiency of sampling
  - Limitations/Concerns?

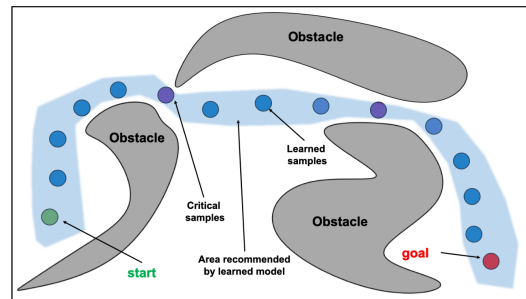


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up sample generation

- Goal: Improve quality and efficiency of sampling
  - Limitations/Concerns?
    - Relies heavily on the dataset and the representational power of the model
    - Conditioned on potentially lots of factors including the poses of all the objects
    - A uniform sampler may produce lower quality samples but is likely to be much faster

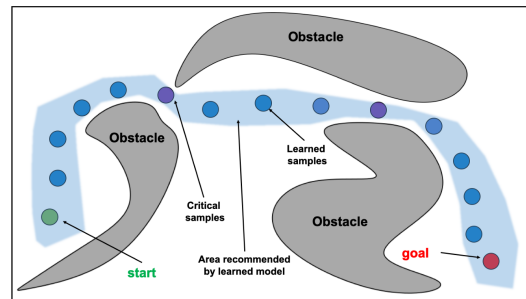


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up collision checking

- Goal: Speed up collision checking
  - Using a learned model in place of a collision detector
  - Determining the order in which to collision check nodes/edges

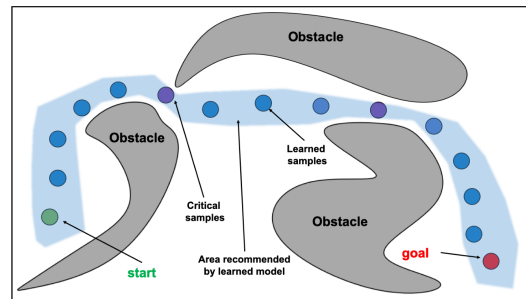


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up collision checking

- Goal: Speed up collision checking
  - Using a learned model in place of a collision detector
    - ClearanceNet-RRT [Kew et al., 2020]:
      - Uses NN (ClearanceNet) to predict distances to the nearest obstacle (obstacle clearance) conditioned on the poses of the robot and the obstacles in the environment
      - Constructs RRT tree with deferred “true” collision checking.
      - Once a path is found, performs “true” collision checking and fixes errors by performing gradient descent repair to move the path away from obstacles.

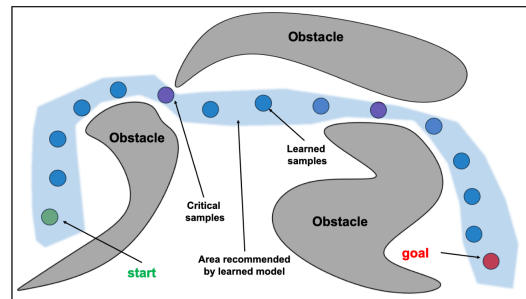


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up collision checking

- Goal: Speed up collision checking
  - Determining the order in which to collision check nodes/edges
    - Search Though Oracle Learning and Laziness (STROLL) [Bhardwaj et al., 2019):
      - Formulates the problem of edge selection for full collision checking as MDP
      - Performs Q-learning to learn an NN-based policy

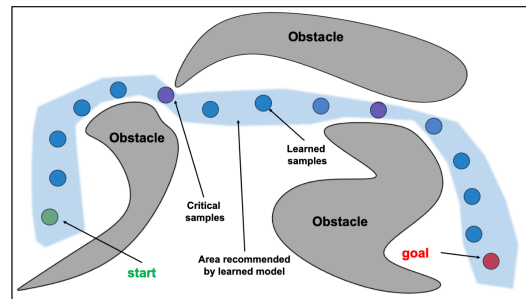


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up collision checking

- Goal: Speed up collision checking
  - Limitations/concerns?

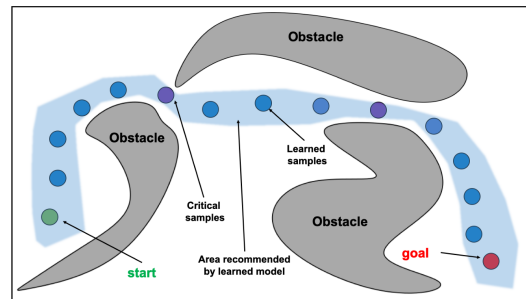


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up collision checking

- Goal: Speed up collision checking
  - Limitations/concerns
    - Assuming collision checking is much more expensive than inference, usually is a good idea
    - Needs to be done carefully to preserve theoretical guarantees on probabilistic completeness

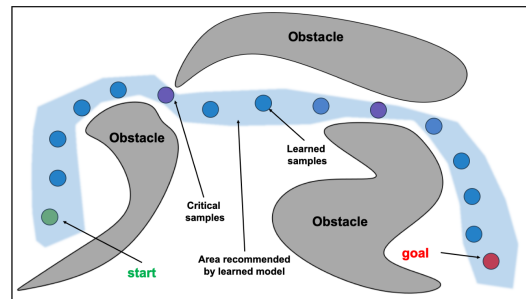


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up cost computation

- Goal: Improve the overall efficiency using learned cost proxies
  - Uses learned costs in the context of informed sampling-based motion planners [Gammell and Strub, 2021]
    - Uses learned costs as approximate cost-to-go values (aka heuristics) to prune “irrelevant” nodes/space exploration

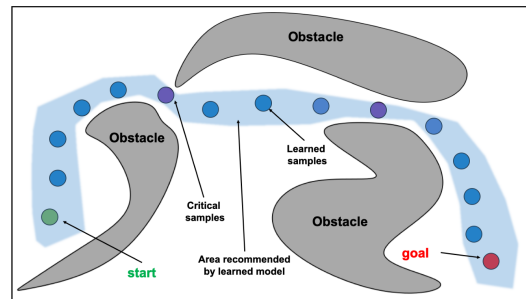


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up cost computation

- Goal: Improve the overall efficiency using learned cost proxies
  - Limitations/concerns?

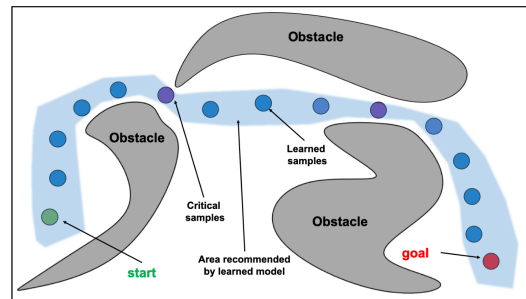


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning to speed up cost computation

- Goal: Improve the overall efficiency using learned cost proxies
  - Limitations/concerns?
    - Might be hard to get a training dataset for high-D problems
    - Have to be careful to guarantee solution quality guarantees

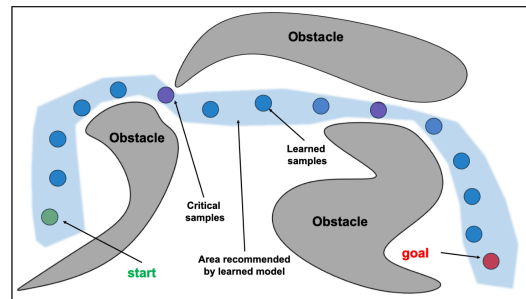


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning local planners

- Goal: Improve the speed and quality of local planners when planning with dynamic constraints
  - PRM-RL [Faust et al., 2018]:
    - Uses reinforcement learning to obtain a policy for point-to-point navigation with dynamics constraints.
    - During the construction of PRM, for each edge connection, it uses the learned policy to obtain paths between candidate neighbors. Neighbors are connected if the learned policy can consistently find a path between them.

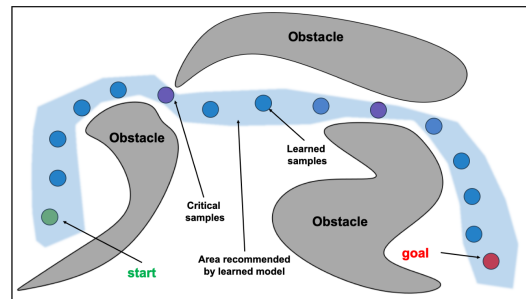


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning local planners

- Goal: Improve the speed and quality of local planners when planning with dynamic constraints
  - RL-RRT [Chiang et al., 2019]:
    - Uses an offline learned obstacle avoidance policy to perform local planning between two states during tree expansion, along with a learned reachability estimator to bias tree growth.

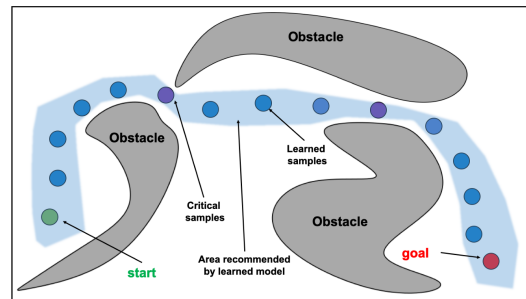


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning local planners

- Goal: Improve the speed and quality of local planners when planning with dynamic constraints
  - Learning-based kinodynamic RRT\* [Zheng and Tsiotras, 2021]:
    - Trains offline a controller to steer the system from one point to another based on a dataset of computed optimal trajectories
    - Also, learns cost-to-go metrics
    - Integrates both learned cost-to-go metrics (effectively, heuristics and edge cost estimates) and learned controllers within RRT\*

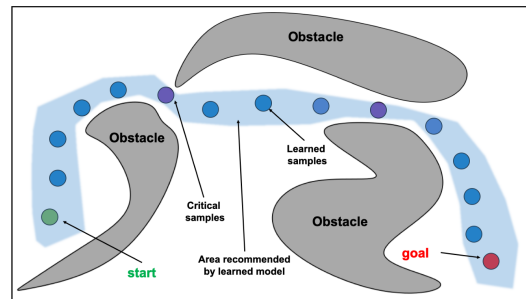


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning local planners

- Goal: Improve the speed and quality of local planners when planning with dynamic constraints
  - Limitations/concerns?

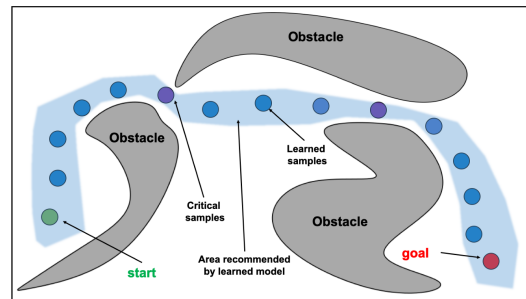


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning local planners

- Goal: Improve the speed and quality of local planners when planning with dynamic constraints
  - Limitations/concerns?
    - Requires a potentially large dataset to get good performance on complex tasks
    - Need to be careful to maintain theoretical guarantees on asymptotic completeness and (in case of RRT\*) optimality

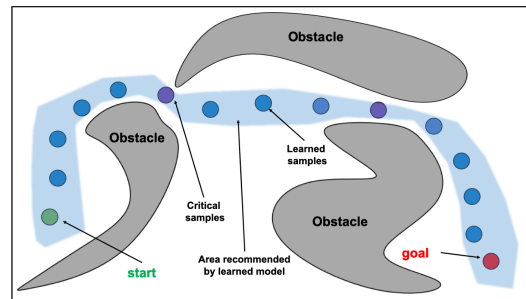


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning planner parameters/selection

- Goal: Improve the chances that we apply the “right” planner or configure it best
  - Planner Ensemble [Choudhury et al., 2015]:
    - Based on offline generated data that captures planner performance, learns a mapping from a given scenario to probability of solving it for a given planner (including both sampling-based planners and trajectory optimization methods)
    - Online, selects the ensemble of planners that has the maximum likelihood of finding a feasible solution.

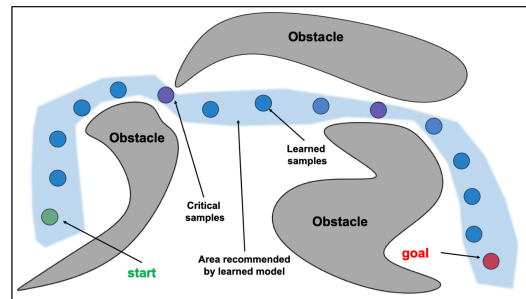


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning planner parameters/selection

- Goal: Improve the chances that we apply the “right” planner or configure it best
  - “When to stop planning problem” [Sun et al., 2021]:
    - Learns the relationship between solution quality and planning time
    - Online uses to determine when to stop improving the solution
    - Data generated by running an anytime planner on training problems for a long enough time to obtain an optimal solution with a high probability

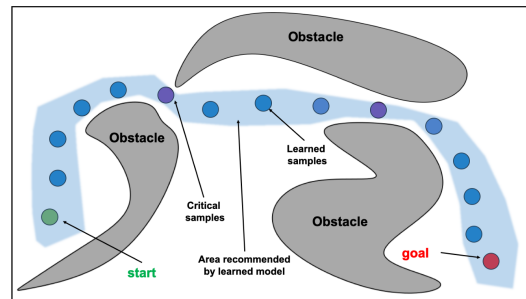


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning planner parameters/selection

- Goal: Improve the chances that we apply the “right” planner or configure it best
  - Limitations/concerns?

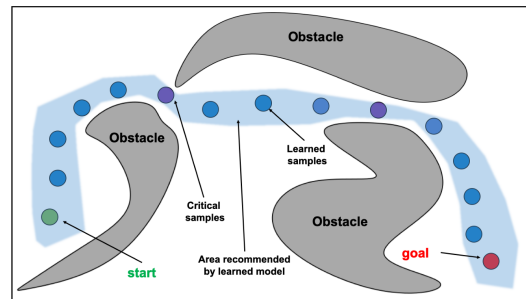


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning planner parameters/selection

- Goal: Improve the chances that we apply the “right” planner or configure it best
  - Limitations/concerns?
    - The learned distribution needs to be conditioned on features extracted from the environment.
    - These features have to be either hand-crafted or learned. If learned, then it is likely to be a hard learning problem.

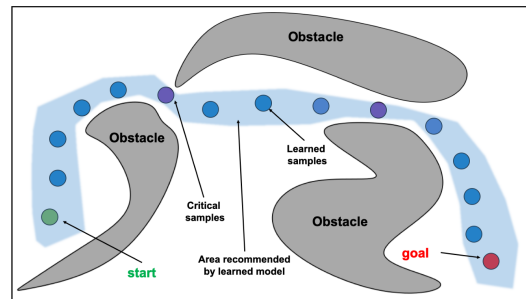


Figure 3.1: Given a planning problem: start (red), goal (green) and obstacles (grey), a learned sampling strategy predicts samples inside the blue region, characterized by parts of  $C_{free}$  that lie along a solution path. Some of these samples (purple) lie inside critical regions, such as narrow passages.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahon et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning low-D planning representation

- Goal: Reduce the search space, and thereby speed up planning
  - Learned Latent RRT (L2RRT) [Ichter and Pavone, 2019]:
    - Learns a latent space via an autoencoder
    - Constructs a tree directly in the latent space by propagating randomly sampled controls in it using a learned latent dynamics model
    - Uses learned collision checking to check for collisions in the latent space.
    - The collision checking network outputs the probability of a latent state being in collision or not, and L2RRT makes use of a pre-defined confidence threshold to decide whether to trust the learned collision checker or not

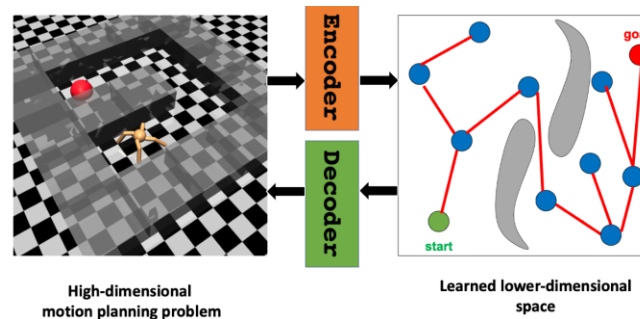


Figure 4.3: Example of using learned lower dimensional space. A simulated quadruped robot (left) with a 111-dimensional state space and 8-dimensional control space must navigate the maze to reach the goal (red circle). It may be practically infeasible to apply a SBMP directly to this problem, so a learned *encoder* model transforms the problem into a lower-dimensional space (right) where a SBMP can be applied to find a solution. Once a solution has been found, it can be applied directly to the high-dimensional problem via a learned *decoder*.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning low-D planning representation

- Goal: Reduce the search space, and thereby speed up planning
  - Limitations/concerns?

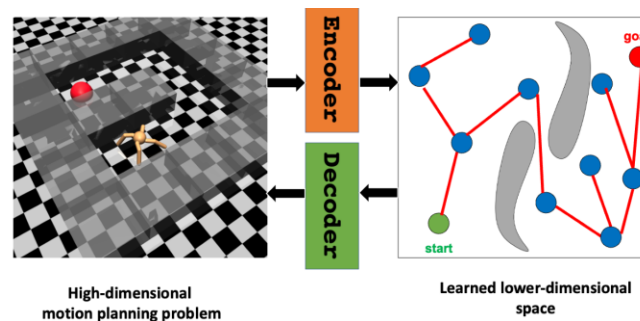


Figure 4.3: Example of using learned lower dimensional space. A simulated quadruped robot (left) with a 111-dimensional state space and 8-dimensional control space must navigate the maze to reach the goal (red circle). It may be practically infeasible to apply a SBMP directly to this problem, so a learned *encoder* model transforms the problem into a lower-dimensional space (right) where a SBMP can be applied to find a solution. Once a solution has been found, it can be applied directly to the high-dimensional problem via a learned *decoder*.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Learning low-D planning representation

- Goal: Reduce the search space, and thereby speed up planning
  - Limitations/concerns?
    - The required size of the dataset
    - Generalization capabilities
    - Theoretical guarantees on completeness/path quality

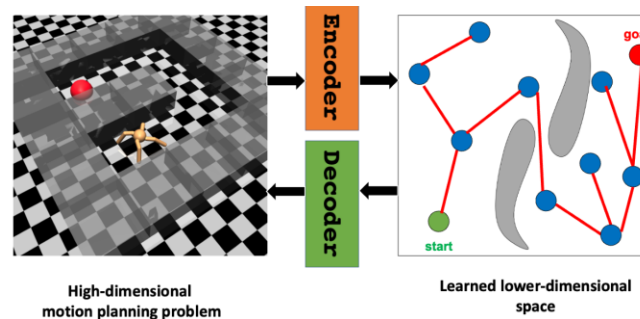


Figure 4.3: Example of using learned lower dimensional space. A simulated quadruped robot (left) with a 111-dimensional state space and 8-dimensional control space must navigate the maze to reach the goal (red circle). It may be practically infeasible to apply a SBMP directly to this problem, so a learned *encoder* model transforms the problem into a lower-dimensional space (right) where a SBMP can be applied to find a solution. Once a solution has been found, it can be applied directly to the high-dimensional problem via a learned *decoder*.

*"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., Foundations and Trends® in Robotics, Volume 9, Issue 4, 2022*

# Papers to cover

- *"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., 2022*
- *"Roadmaps with Gaps over Controllers: Achieving Efficiency in Planning under Dynamics" by Sivaramakrishnan et al., 2024*
- *"Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks" by Takahashi et al., 2019*
- *"Learning Local Heuristics for Search-Based Navigation Planning" by Veerapaneni et al., 2025*

# What?

---

- Learning a heuristic function for A\* search in the context of planning for  $\langle x, y, \theta \rangle$  navigation

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# Why?

- Search-based planning for  $\langle x, y, \theta \rangle$  problem in large complex spaces can be time-consuming, especially when going for a close-to-optimal solution.
- The efficiency of search-based planning is heavily dependent on how informative heuristics are.

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# Related Work

- Most related according to the paper:
  - Imitation learning with NN learn a heuristic policy [Bhardwaj, Choudhury, and Scherer, 2017].
  - “Tested on simple domains but relatively high dimensional states with the fixed start and goal. Once start positions, goal positions, and environments are randomized, local features may not be able to guide the planner to a goal efficiently, as discussed in (Dhiman et al. 2018). On the contrary, we learn a heuristic function defined over the entire environment given the goal configuration and the map to avoid the misleading problem.”
- Proposed work helps generalize to multiple environments
  - Also, can handle environments where a path does not exist

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. “Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks,” Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# Assumptions

---

- Reliance on the model used for planning

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# How?

- Overall architecture

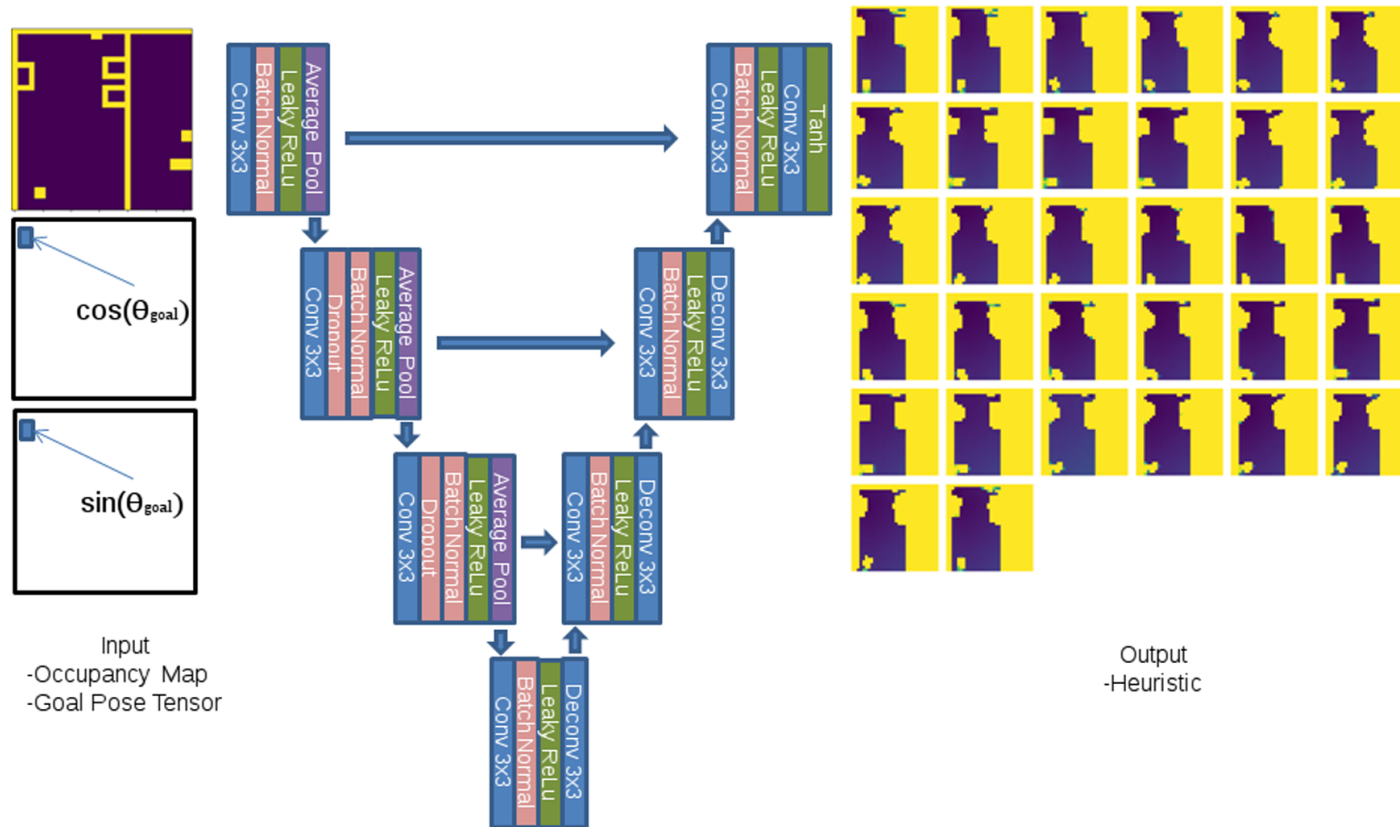


Figure 2: Network architecture used in our experiments (continuous domain).

Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# How?

- Input into NN:
  - an occupancy map (2D matrix) and a goal tensor  $G$  which is two 2-D matrices:

$$G(0, i, j) = \begin{cases} \cos(\theta_{goal}), & \text{if } i = i_{goal} \text{ and } j = j_{goal} \\ 0, & \text{otherwise} \end{cases}$$

$$G(1, i, j) = \begin{cases} \sin(\theta_{goal}), & \text{if } i = i_{goal} \text{ and } j = j_{goal} \\ 0, & \text{otherwise} \end{cases}$$

- Output from NN:
  - tensor of  $W \times H \times O$  where  $W$  and  $H$  is the number of columns and rows in the occupancy map, respectively, and  $O$  is the number of orientations

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# How?

- Loss function during training:
  - Simplest form:

$$MAE(h, h^*) = \frac{1}{N} \sum_{i=0}^{N-1} |h(i) - h^*(i)|$$

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# How?

- Loss function during training:
  - Simplest form:

$$MAE(h, h^*) = \frac{1}{N} \sum_{i=0}^{N-1} |h(i) - h^*(i)|$$

- A bit more complex form that “favors” admissibility:

$$\begin{aligned} Loss_{piece}(h, h^*) = \frac{1}{N} \left( \right. \\ & \alpha_1 \sum_{i=0}^{N-1} |h(i) - h^*(i)| [h(i) < h_{min}(i)] \\ & + \sum_{i=0}^{N-1} |h(i) - h^*(i)| [h_{min}(i) \leq h(i) \leq h^*] \\ & \left. + \alpha_2 \sum_{i=0}^{N-1} |h(i) - h^*(i)| [h^*(i) < h(i)] \right) \end{aligned}$$

Takahashi, T., Sun, H., Tian, D., & Wang, Y. “Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks,” *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# How?

- Loss function during training:
  - Better “informed” heuristic doesn’t imply less search efforts
  - “Gradient” in heuristics is often more valuable (and is always more valuable in the context of sub-optimal search)

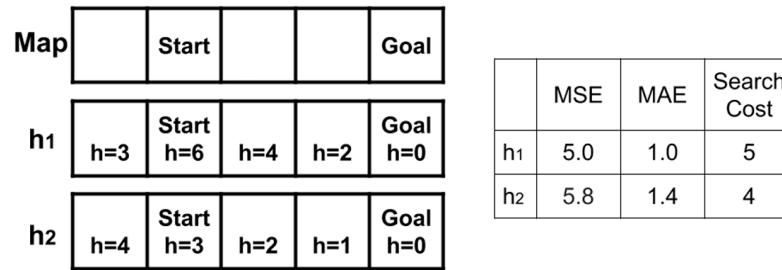


Figure 3: Equation 2 does not satisfy in this example. We have a 1D map that contains five nodes. A robot can move either left or right. The cost of an action is 2.  $h_1$  and  $h_2$  represent some learned heuristics.  $h$  in each node represents a value of the heuristic function. Mean squared error (Equation 3) and mean absolute error (Equation 4) between the heuristic and the optimal heuristic are shown in the table on the right. The search cost is the number of expanded nodes when we apply the  $A^*$  algorithm.  $h_2$  has more errors than  $h_1$ , but  $h_2$  has a lower search cost.

- Composite loss function that combines “informativeness” and “gradient”:

$$Loss = Loss_1 + \alpha Loss_{grad}$$

$MAE Loss$   $\swarrow$   $\nwarrow$  “difference in policy actions” loss

Takahashi, T., Sun, H., Tian, D., & Wang, Y. “Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks,” *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# How?

---

- Training:
  - Generate random environments, with random start and goal
  - Run backward Dijkstra's from goal to get perfect heuristics

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# How?

- Results:

Loss functions	Complexity Evaluation								
	Comparison against baselines 130 test tasks $N$ : The number of expanded Nodes $N_b$ : Baseline, $N$ : Learned				Ratio of the number of expanded nodes $q$ th percentile Ratio = $\frac{N}{N_{Euclidean}}$				
	Euclidean		Scaled Euclidean		25th	Median	75th	90th	95th
	$N_b < N$	$N_b > N$	$N_b < N$	$N_b > N$					
MSE	76	44	102	15	0.650	1.939	4.348	8.256	12.380
MSE, $loss_{grad}$	60	62	90	32	0.423	1.014	2.397	6.314	9.527
MAE	34	<b>87</b>	69	51	0.145	0.383	1.095	2.351	5.272
MAE, $loss_{grad}$	21	<b>99</b>	53	67	0.084	0.179	0.859	1.456	4.292
Piecewise	13	<b>107</b>	36	<b>83</b>	0.035	0.081	0.248	0.770	1.082
Piecewise, $loss_{grad}$	16	<b>105</b>	37	<b>83</b>	0.046	0.109	0.307	0.908	1.656
Baseline (Euclidean)	NA	NA	87	42	1.00	1.00	1.00	1.00	1.00
Baseline (Scaled Euclidean)	42	87	NA	NA	0.122	0.265	0.641	0.888	1.500

Table 2: Results of experiments in the continuous domain. The numbers in the comparison against baselines show the number of tasks that are better than the other method. For example,  $N_{base} < N$  shows the number of tasks where the baseline has fewer expanded nodes than that of the learned heuristic. The bold numbers indicate that the learned heuristic is statistically better than the baseline ( $\chi^2$  test,  $p < 0.05$ ). Outliers dominate the mean and the variance. Instead, we show the  $q$  th percentile.

*\*scaled euclidean: inflated heuristics with inflation factor=1.5*

Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# Limitations?

---

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# Limitations?

---

- Guarantees on solution quality
- Support for higher-D?
- Philosophically: doesn't seem to be scalable w.r.t. size/complexity of the environment

*Takahashi, T., Sun, H., Tian, D., & Wang, Y. "Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2019*

# Papers to cover

---

- *"A Survey on the Integration of Machine Learning with Sampling-based Motion Planning" by McMahan et al., 2022*
- *"Roadmaps with Gaps over Controllers: Achieving Efficiency in Planning under Dynamics" by Sivaramakrishnan et al., 2024*
- *"Learning Heuristic Functions for Mobile Robot Path Planning Using Deep Neural Networks" by Takahashi et al., 2019*
- *"Learning Local Heuristics for Search-Based Navigation Planning" by Veerapaneni et al., 2025*

# What?

- Learn local heuristics (heuristics within a small window centered around the state in question): “cost of escaping local region”

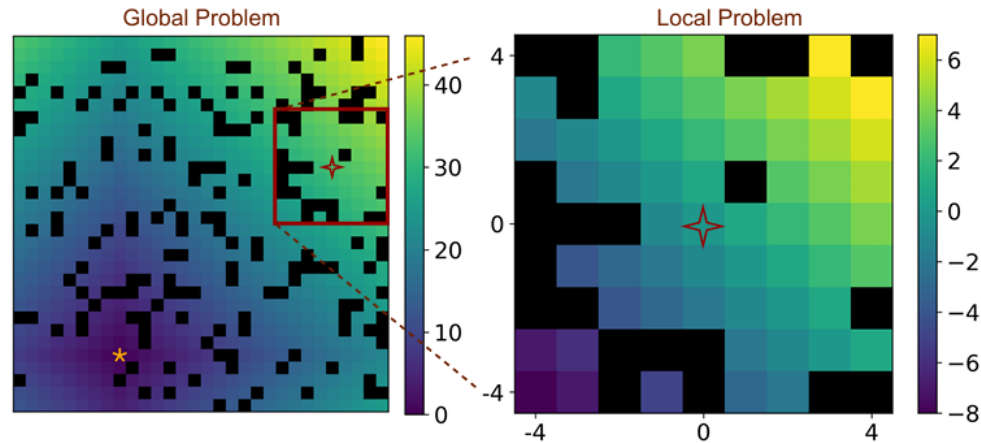


Figure 1: Left: Estimating a *global* cost-to-go heuristic from the red star state  $s$  to the orange goal requires reasoning over a large region. Right: We define and learn a *local* heuristic centered at  $s$  which reasons about vehicle dynamics and obstacles in a local region, that we then combine with the global heuristic. This significantly smaller problem eases the learning progress, enables generalization, and provides significant improvements in some domains.

R. Veerapaneni, M. S. Saleem, and M. Likhachev, “Learning local heuristics for search-based navigation planning,” *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2023

# Why?

---

- To avoid the combinatorial sample complexity of learning

*R. Veerapaneni, M. S. Saleem, and M. Likhachev, "Learning local heuristics for search-based navigation planning," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2023*

# How?

- Computing global heuristics using a local region

Mathematically, given a state  $s = (x, y, \Omega)$  with position  $x, y$  and other state parameters  $\Omega$  (e.g. heading, velocity), we define a local region  $LR(s)$  to contain the states within a window of  $K$ , i.e.  $LR(s) = \{s' \mid K \geq |s.x - s'.x|, K \geq |s.y - s'.y|\}$ . Let  $LRB(s)$  be the border of this region, i.e.  $\{s' \mid K = |s.x - s'.x| \vee K = |s.y - s'.y|\}$ . Conceptually, assuming unit length actions, any path from  $s$  to  $s_g$  must contain a state in  $LRB(s)$ , or directly reach the goal in the local region  $LR(s)$ . If neither are possible from  $s$ , then  $s$  cannot leave  $LR(s)$ , is in a dead end, and should have an infinite heuristic value. Thus our objective value  $h_{gk}(s)$  is

$$h_{gk}(s) = \min_{s'} \begin{cases} c(s, s') + h_g(s'), & s' \in LRB(s) \\ c(s, s') + 0, & s' = s_g \in LR(s) \\ \infty, & \text{otherwise} \end{cases} \quad (1)$$

*R. Veerapaneni, M. S. Saleem, and M. Likhachev, "Learning local heuristics for search-based navigation planning," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2023*

# How?

- Local heuristics that needs to be learned:

$$h_k(s) = \min_{s'} \begin{cases} (c(s, s') + h_g(s')) - h_g(s), & s' \in LRB(s) \\ (c(s, s') + 0 - h_g(s)), & s' = s_g \in LR(s) \\ \infty, & \text{otherwise} \end{cases} \quad (2)$$

Therefore instead of passing  $h_g$  into the NN, we only need the relative information  $h_g(s') - h_g(s) \in LR(s)$ .

- Global heuristics for  $s$  becomes:  $h_{gk}(s) = h_g(s) + h_k(s)$

# Results

- **NN input:** **Local state input:** We input  $LR(s)$  as a 2 channel  $2K + 1$  by  $2K + 1$  image centered at  $(\text{floor}(x), \text{floor}(y))$ . The first channel is the binary obstacle map, the second the local invariant heuristic  $h_g(s') - h_g(s)$ . We additionally input the local invariant state containing  $(x - \text{floor}(x), y - \text{floor}(y), \theta, v)$ .

Map Type	Split	Method	Reduction in nodes expanded			
			$w2$	$w8$	$w32$	$w128$
random20	Train	A* w/TL	6.76	10.88	12.78	14.7
		LoHA*	3.53	7.92	10.33	11.6
	Test	A* w/TL	6.6	10.42	14.45	15.75
		LoHA*	3.57	6.94	10.46	12.67
random30	Train	A* w/TL	12.21	26.3	40.38	44.02
		LoHA*	2.16	12.07	18.08	20.51
	Test	A* w/TL	10.36	28.58	43.57	44.3
		LoHA*	1.68	7.71	13.59	16.55
Denver_256	Train	A* w/TL	2.43	6.45	5.92	7.13
		LoHA*	1.22	5.15	3.98	6.37
	Test	A* w/TL	4.54	16.37	30.73	29.21
		LoHA*	1.43	8.43	28.16	30.73

Table 1: LoHA\* Results — We report the median multiplicative reduction in nodes compared to weighted A\*. We see that LoHA\* is able to get larger reductions as the weight  $w$  increases, and that we are able to effectively generalize to different maps.

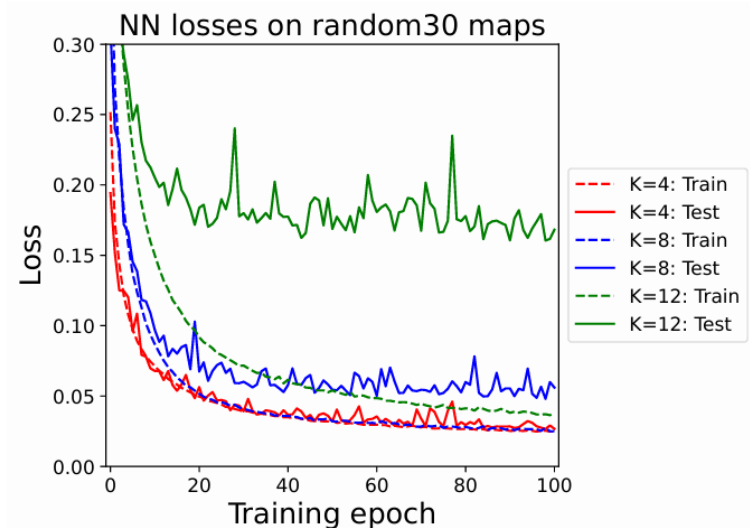


Figure 3: The y-axis is the log relative loss objective; a loss of 0.2 roughly translates to  $\geq 50\%$  absolute relative error, 0.1 to  $\geq 35\%$ . As  $K$  increases, the neural network struggles to generalize to the test maps. This supports our motivation that learning a local heuristic eases the learning problem and improves generalization.

R. Veerapaneni, M. S. Saleem, and M. Likhachev, “Learning local heuristics for search-based navigation planning,” *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2023

# Limitations

---

- Guarantees on solution quality
- Pretty specific to navigation problem

*R. Veerapaneni, M. S. Saleem, and M. Likhachev, "Learning local heuristics for search-based navigation planning," Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2023*