

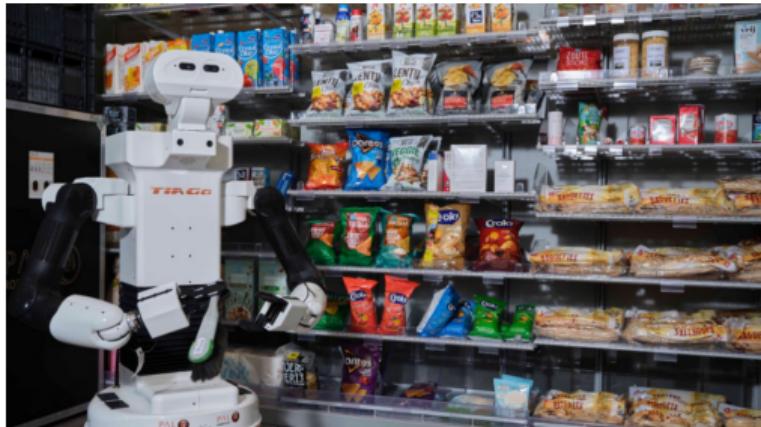
A Graph-Based Search Approach to Planning and Learning

G.S. Groote

Supervisors: Ir. C. Pezzato
Prof. Dr. Ir. M. Wisse
Dr. Ir. C.S. Smith

Delft University of Technology, The Netherlands

July 4, 2023



Challenges

- ① Controllers
- ② Different Environments
- ③ Many System Models of objects

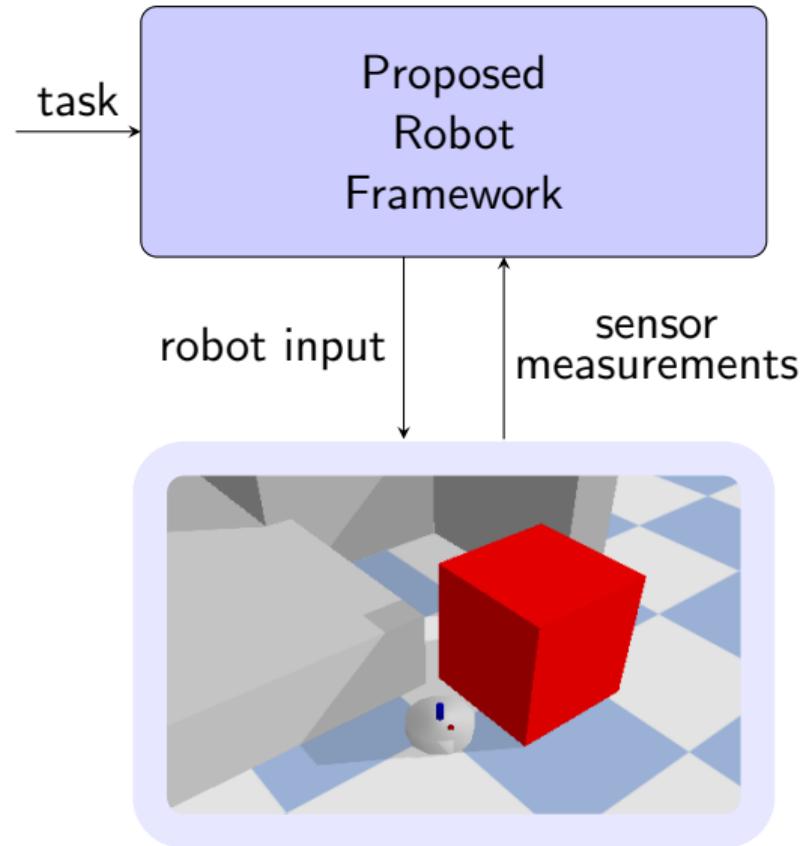
Table of Content

- ① Introduction
- ② Required Background
- ③ Proposed Method
- ④ Results
- ⑤ Conclusions

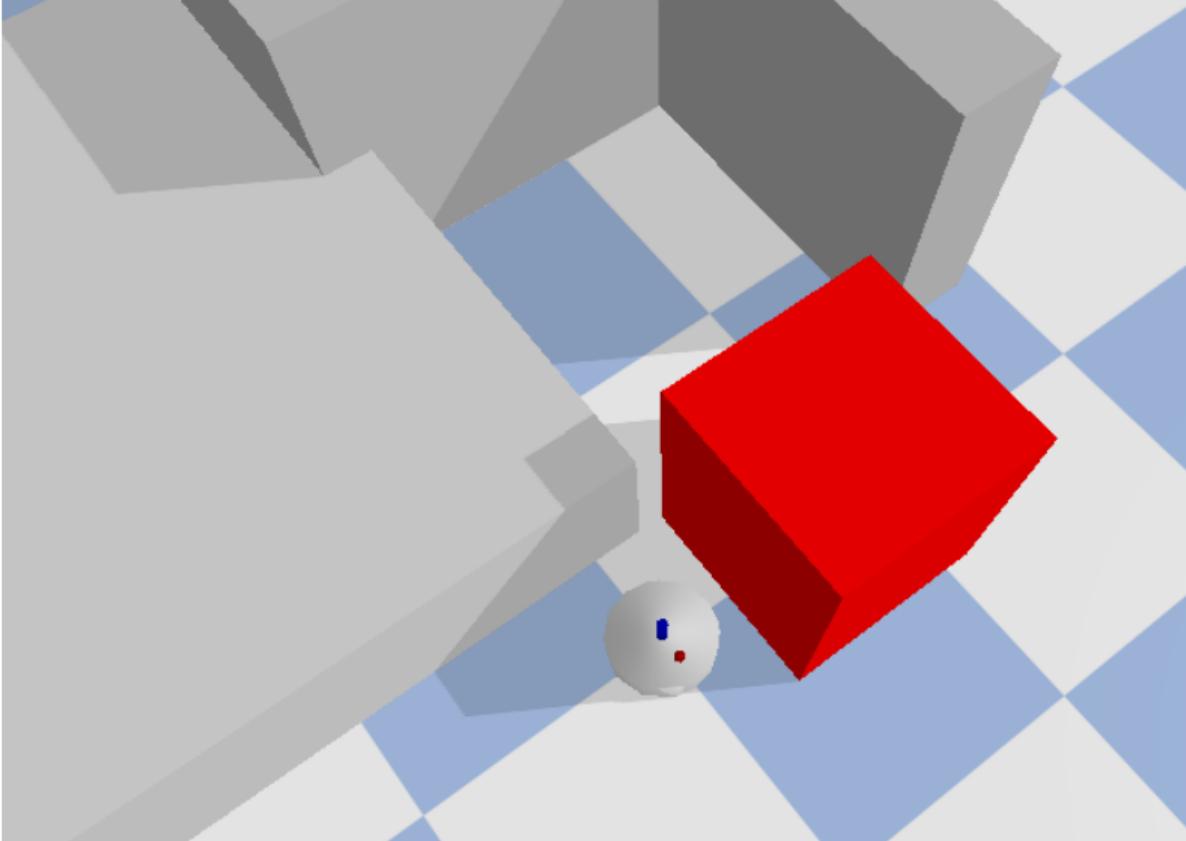
- Learn System Models
- Select best Available Controller

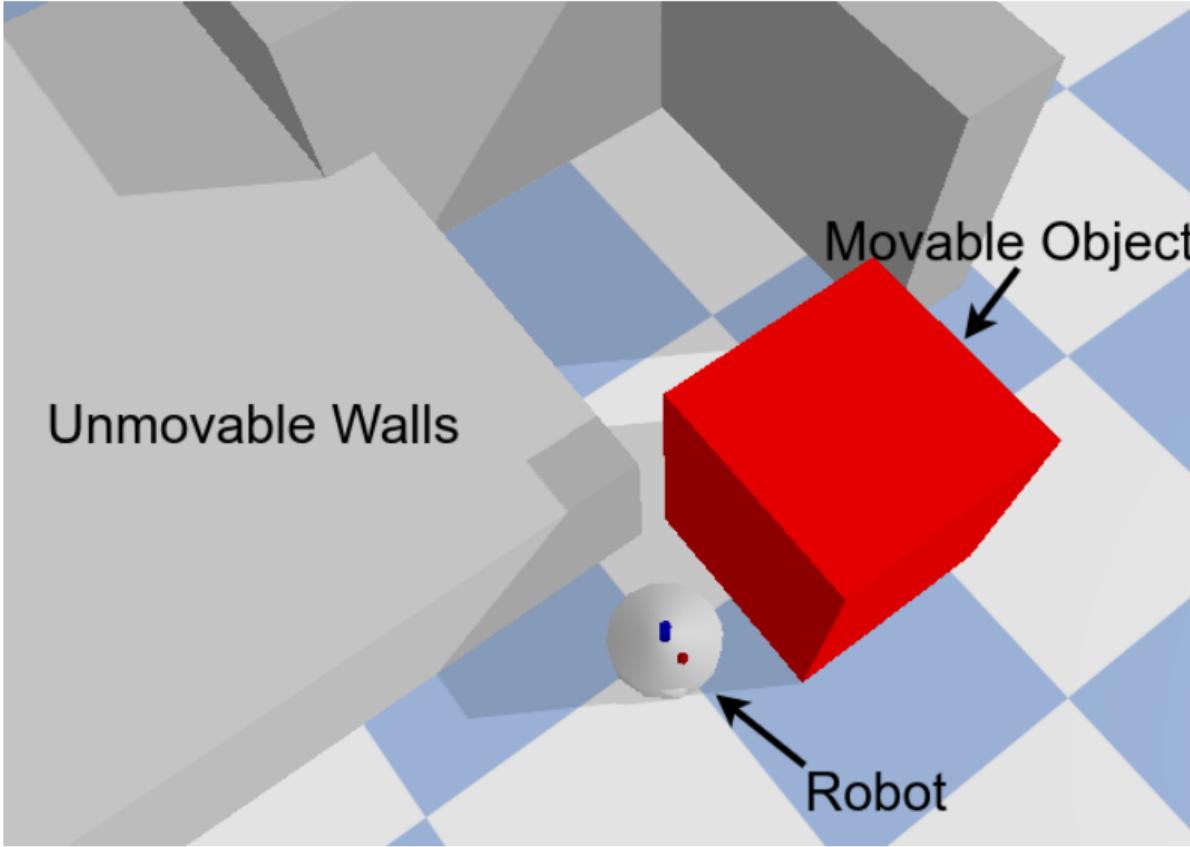
- Learn System Models
 - Select best Available Controller
-
- Navigation Among Movable Objects (NAMO)
 - Nonprehensile Pushing

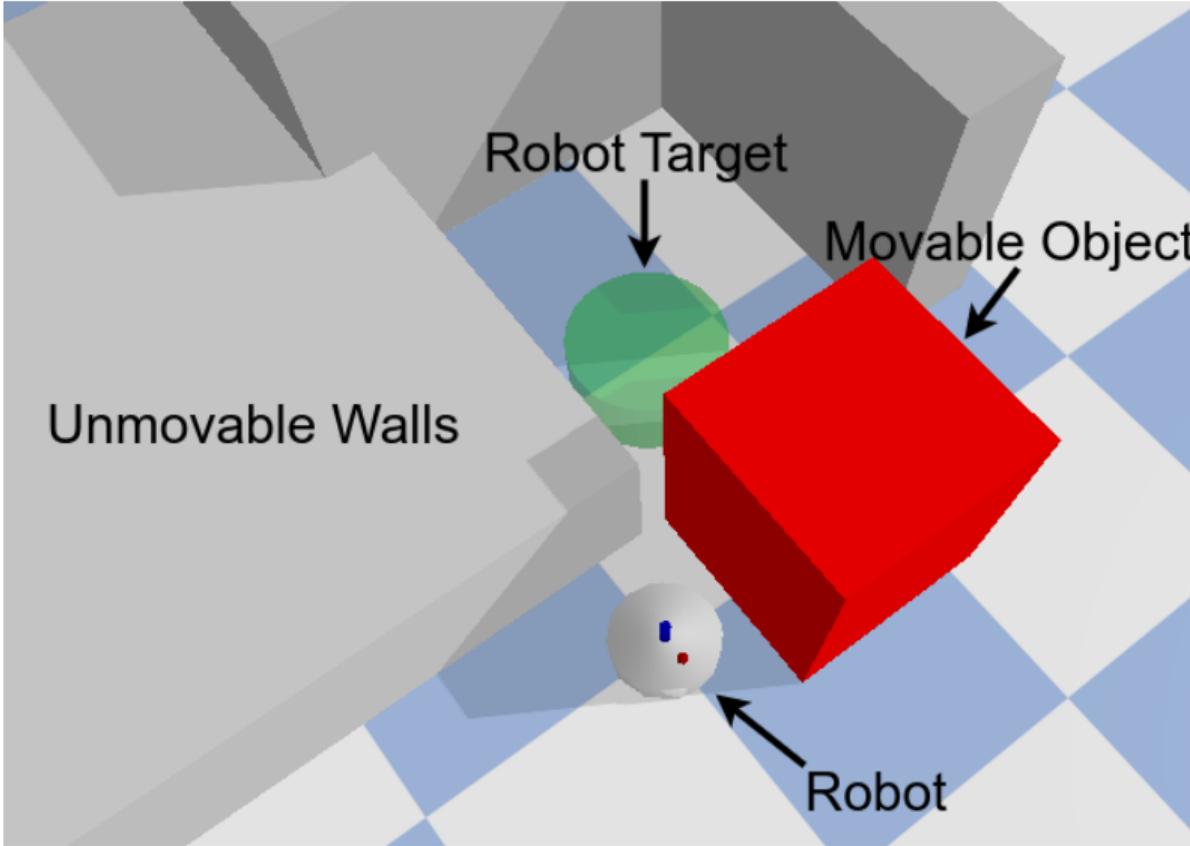
Intro



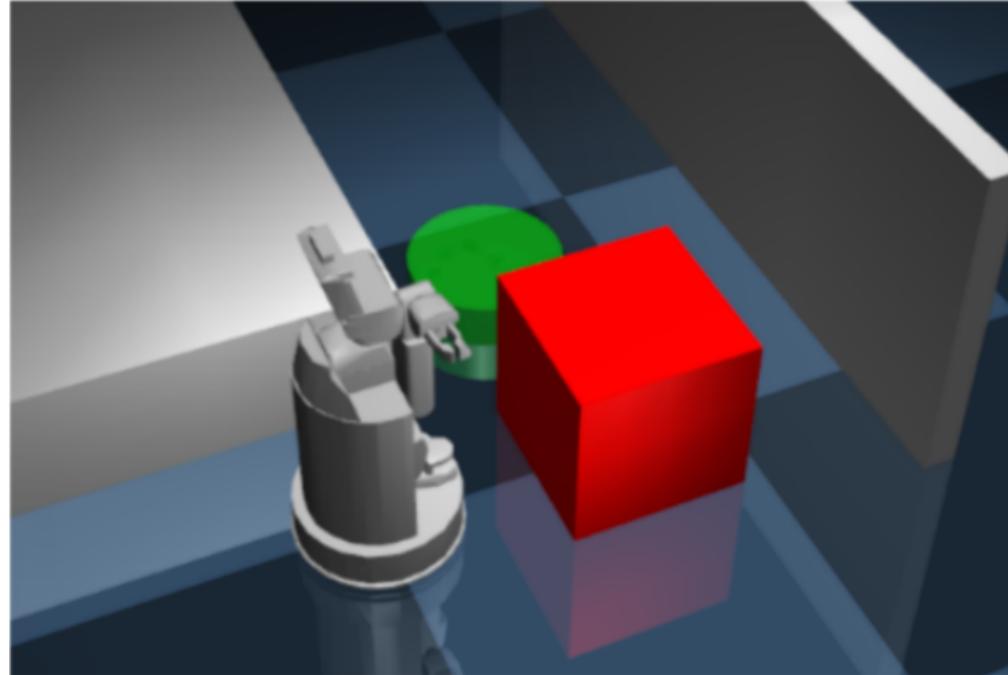
Intro







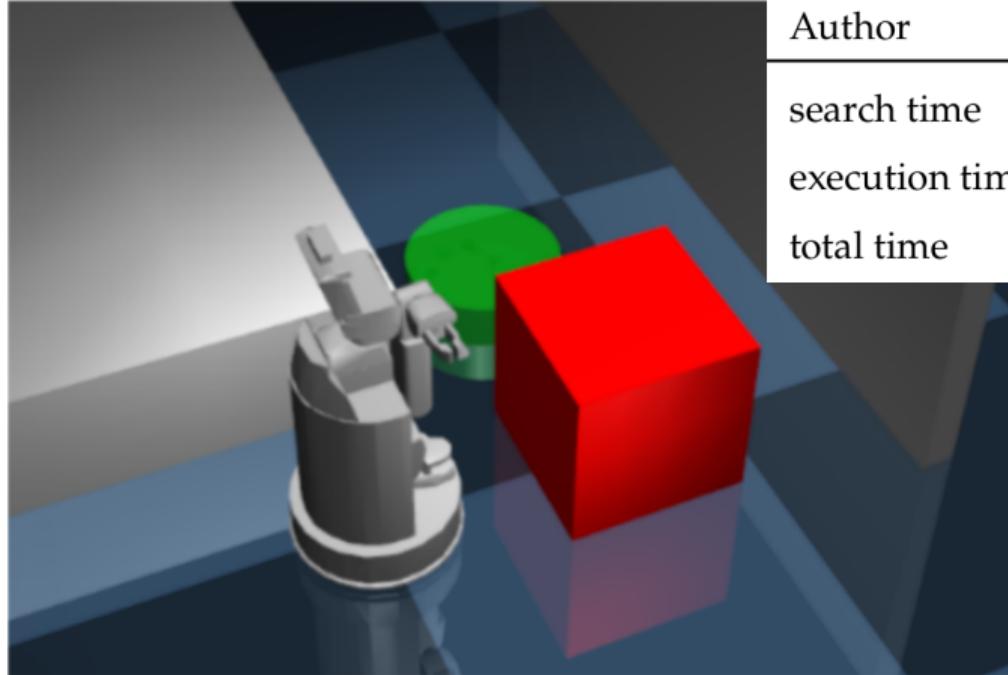
Intro



1

¹Maozhen Wang et al. (Oct. 24, 2020). "Affordance-Based Mobile Robot Navigation Among Movable Obstacles". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, NV, USA: IEEE, pp. 2734–2740. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341337

Intro

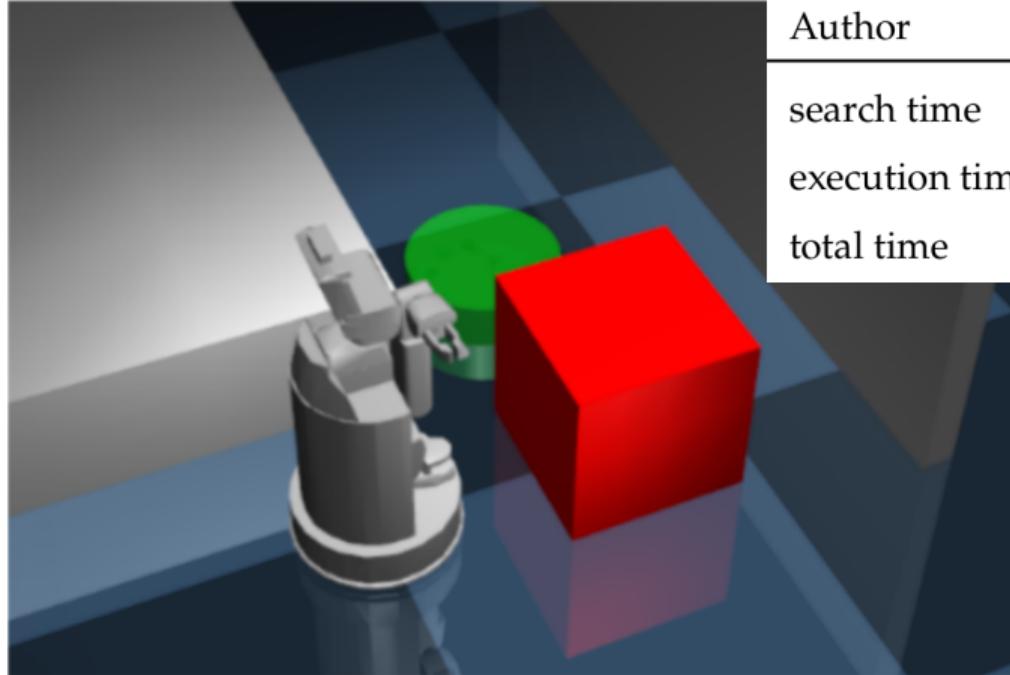


1

Author	Wang et al.
search time	109 sec
execution time	67 sec
total time	176 sec

¹Maozhen Wang et al. (Oct. 24, 2020). "Affordance-Based Mobile Robot Navigation Among Movable Obstacles". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, NV, USA: IEEE, pp. 2734–2740. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341337

Intro



1

Author	Wang et al.	Groote
search time	109 sec	?
execution time	67 sec	?
total time	176 sec	?

¹Maozhen Wang et al. (Oct. 24, 2020). "Affordance-Based Mobile Robot Navigation Among Movable Obstacles". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, NV, USA: IEEE, pp. 2734–2740. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341337

Research Question:

How to learn system models and how to select the best available control method during task execution?

Assumptions

① Closed-World

Assumptions

- ① Closed-World**
- ② Perfect Object Sensor**

Assumptions

- ① Closed-World**
- ② Perfect Object Sensor**
- ③ Tasks are Commutative**

Required Background

Set of Available Controllers:

$$action_1 : \{(controller_1^{action1}, model_1^{action1}), \dots\}$$

$$action_2 : \{(controller_1^{action2}, model_1^{action2}), \dots\}$$

⋮

$$action_n : \{(controller_1^{actionN}, model_1^{actionN}), \dots\}$$

Required Background

System Models

- ① LTI drive model

Required Background

System Models

- ① LTI drive model
- ② LTI push model

Required Background

System Models

- ① LTI drive model
- ② LTI push model
- ③ Nonlinear push model

Required Background

System Models

- ① LTI drive model
- ② LTI push model
- ③ Nonlinear push model

System Model Structure

$$x(k+1) = f(x(k), u(k))$$

Required Background

Control Methods

- ① Model Predictive Control (MPC)
- ② Model Predictive Path Integral (MPPI) control

Required Background

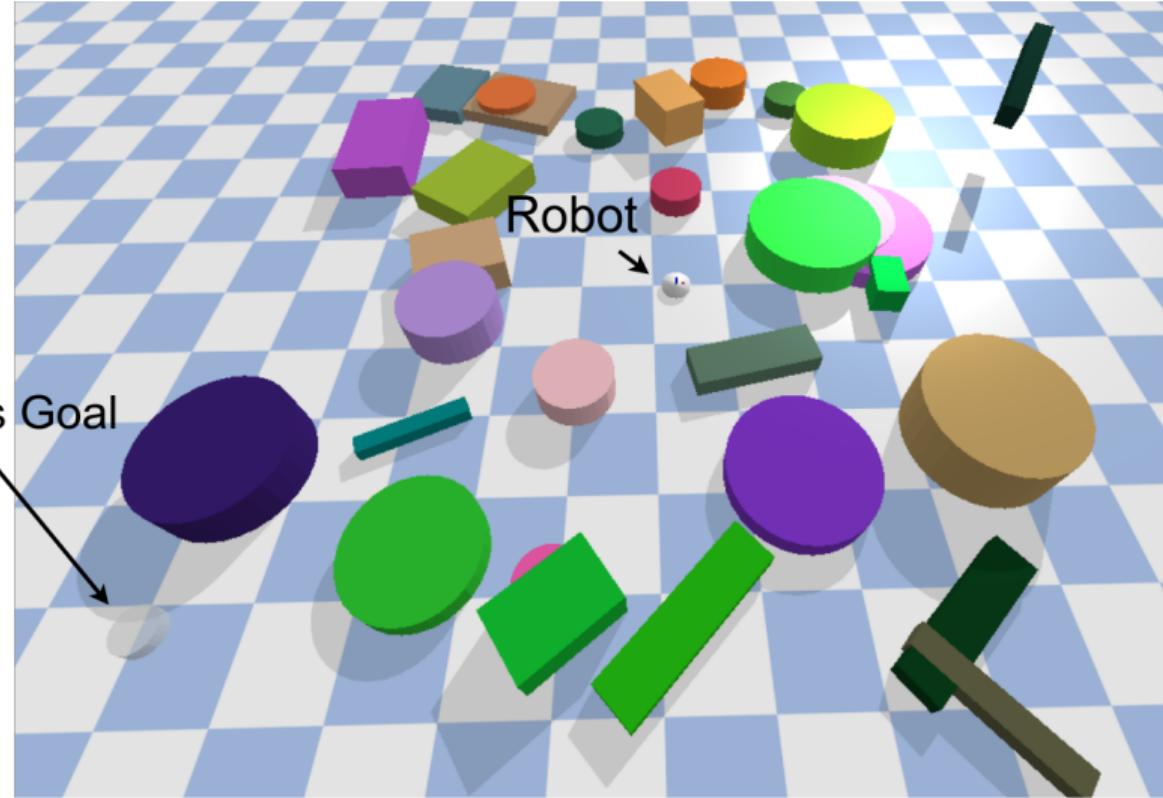
Driving

- ① (MPC, *Iti-drive-model*)
- ② (MPPI, *Iti-drive-model*)

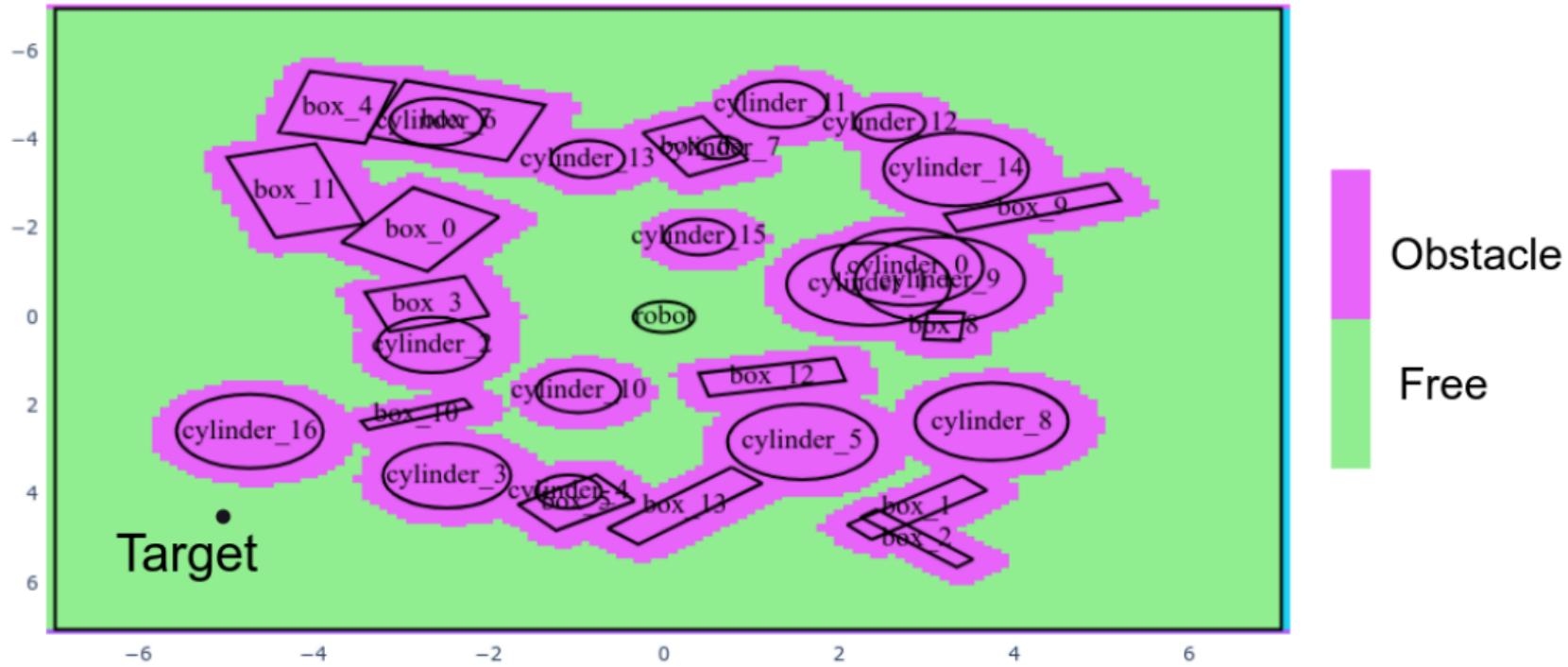
Pushing

- ① (MPPI, *Iti-push-model*)
- ② (MPPI, *nonlinear-push-model*)

Required Background



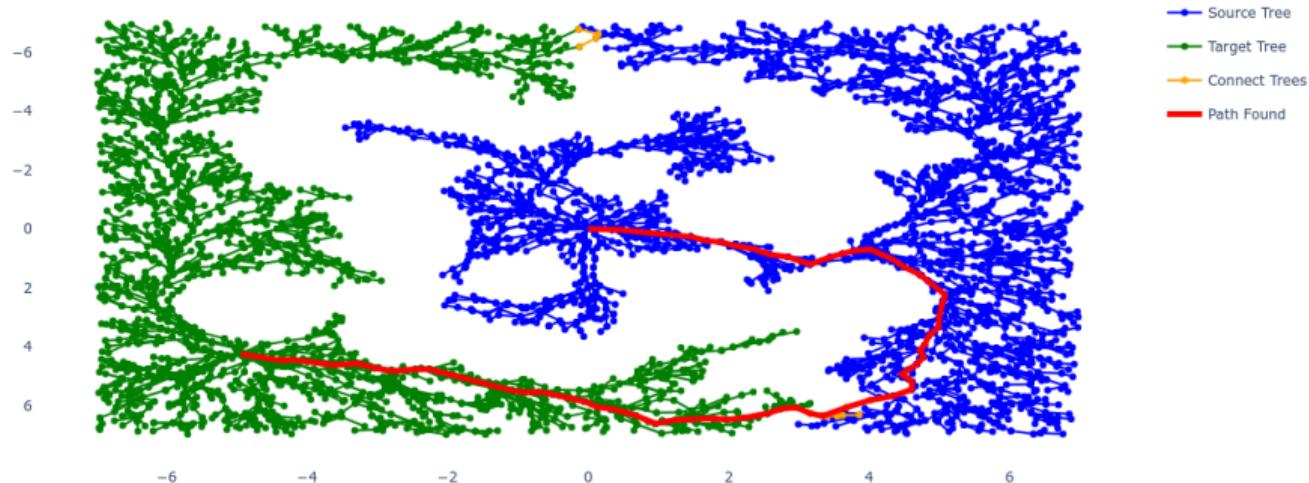
Required Background: Path Estimation



Required Background: Path Planning

Required Background: Path Planning

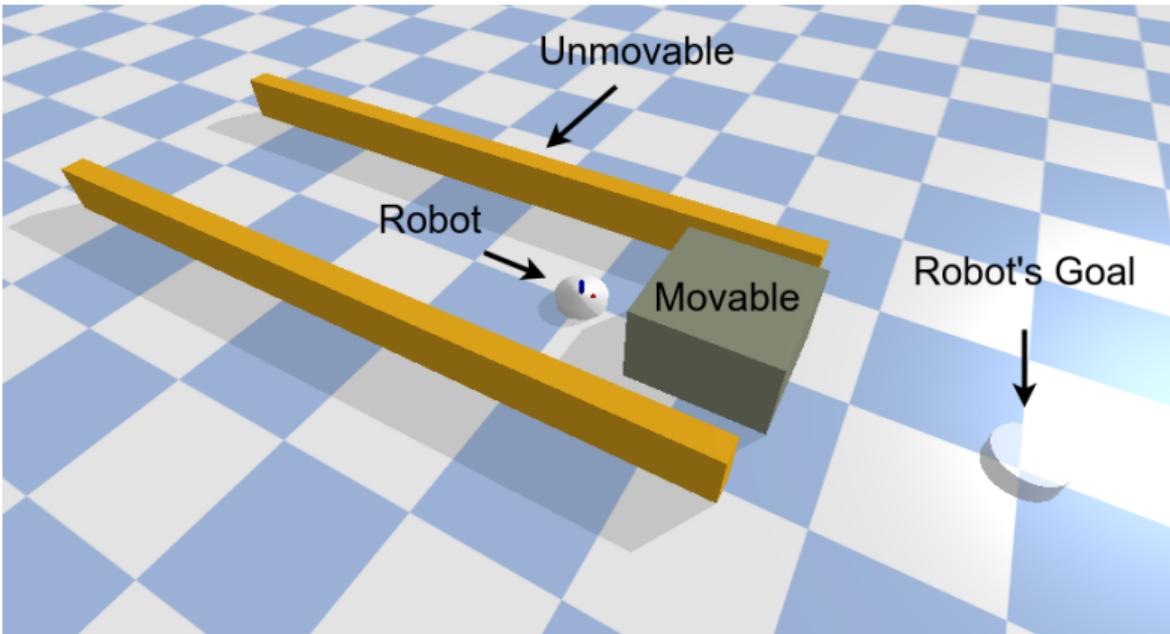
Double-tree Rapidly exploring Random Tree (RRT*) Algorithm ²:



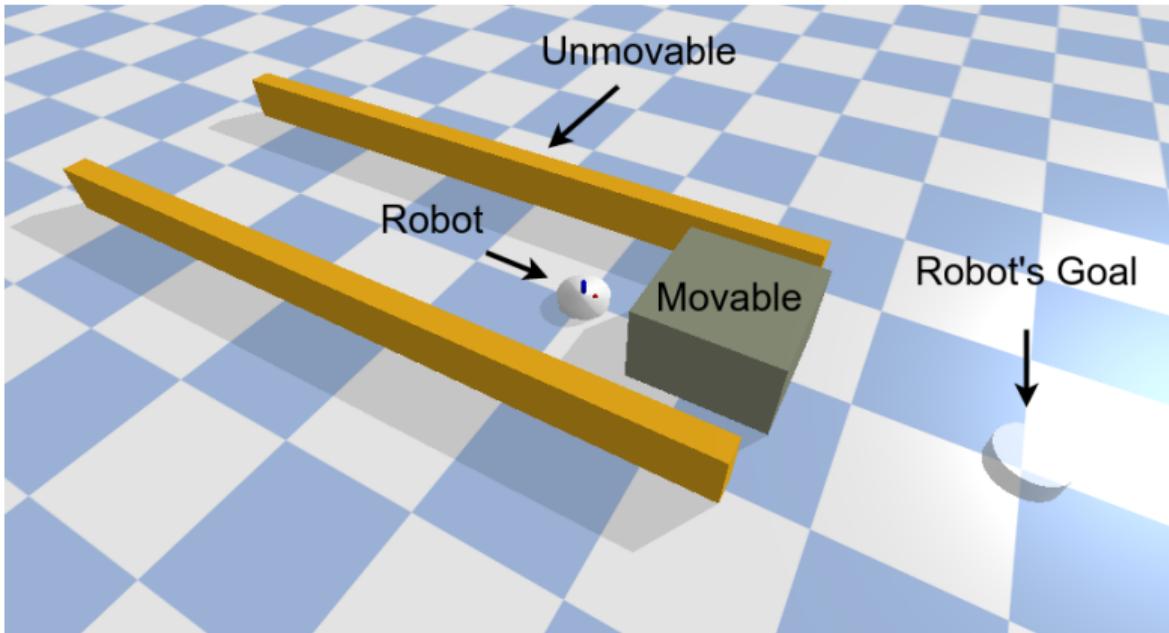
$$Cost_{path} = \sum_{i=1}^{n-1} Distance(c_i, c_{i+1})$$

²Long Chen et al. (Dec. 2018). "A Fast and Efficient Double-Tree RRT*-Like Sampling-Based Planner Applying on Mobile Robotic Systems". In: *IEEE/ASME Transactions on Mechatronics* 23.6, pp. 2568–2578. ISSN: 1083-4435, 1941-014X. DOI: 10.1109/TMECH.2018.2821767

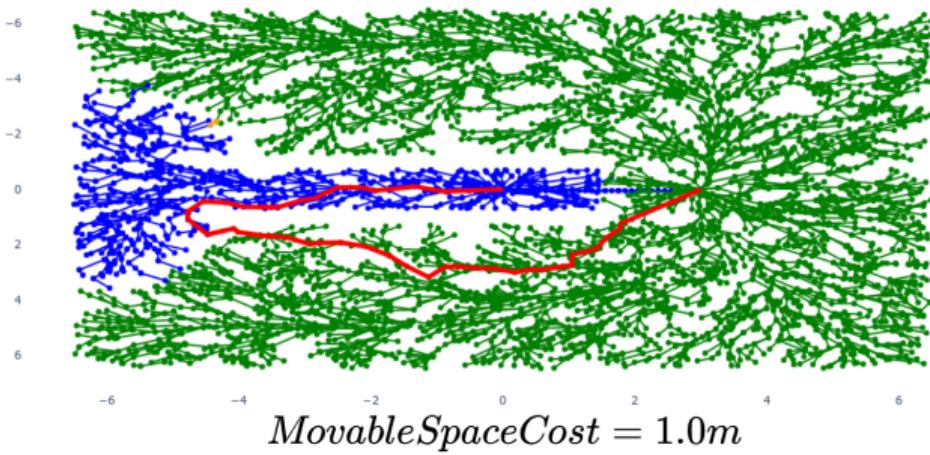
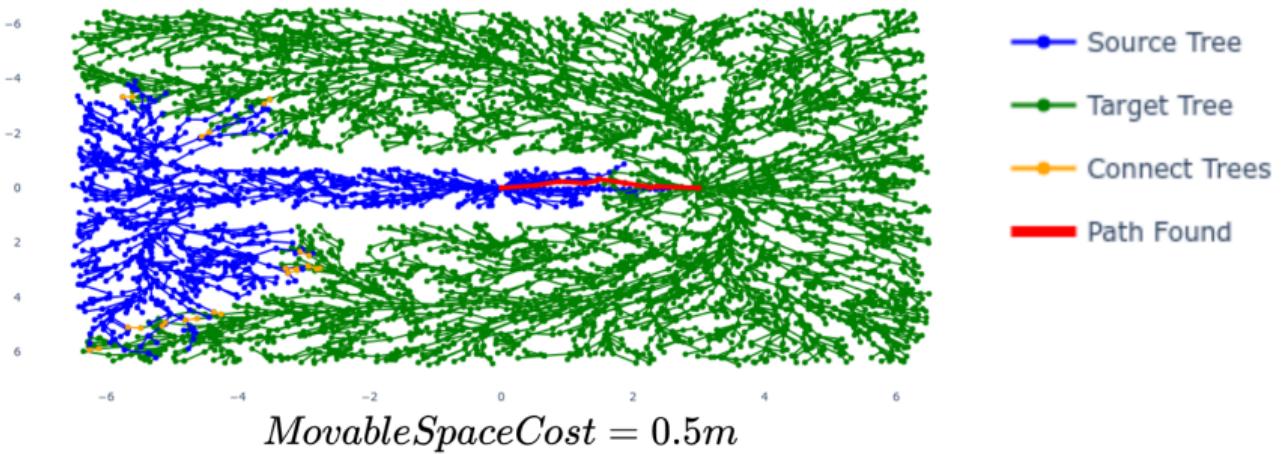
Proposed Method



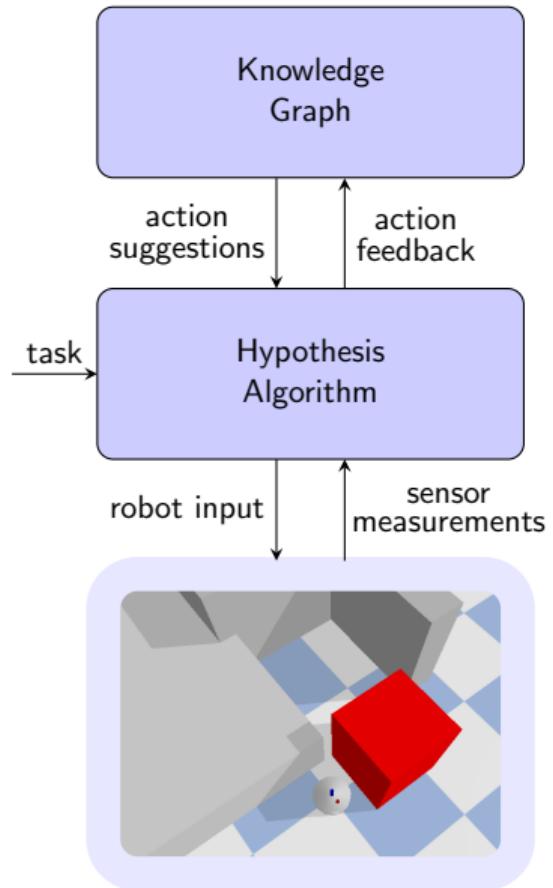
Proposed Method



$$Cost_{path} = \sum_{i=1}^{n-1} Distance(c_i, c_{i+1}) + MovableSpaceCost + UnknownSpaceCost$$

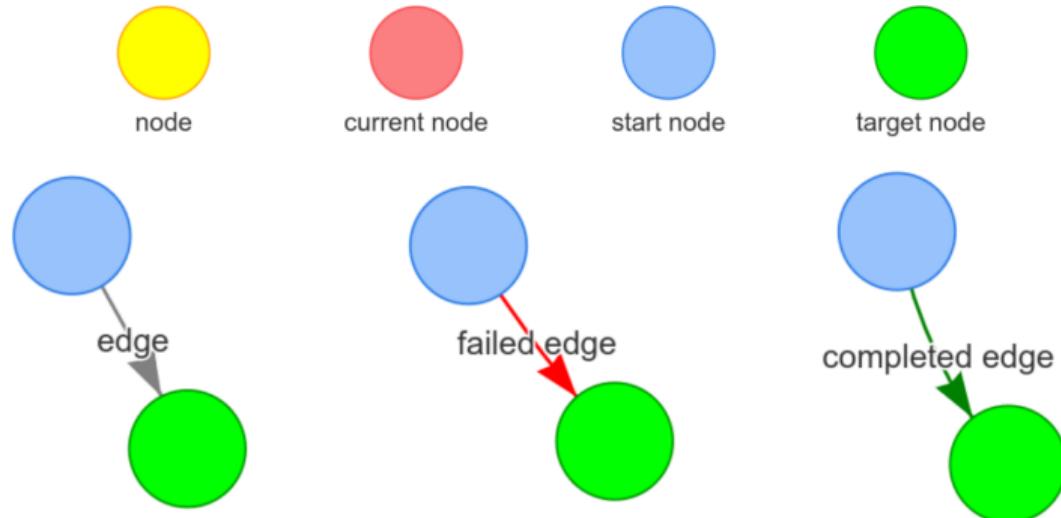


Proposed Method



Proposed Method

$$G^{hypothesis} = \langle V_H, E_H \rangle$$



Proposed Method: H-Graph

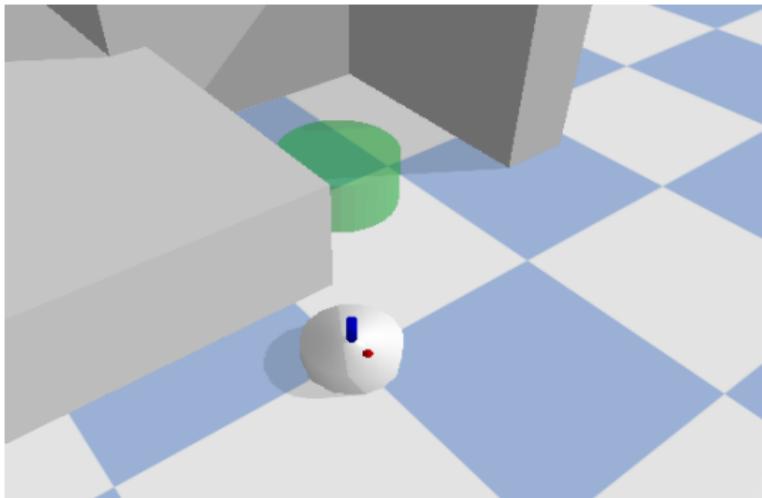
A **identification edge**: Creates a system model

A **action edge**:

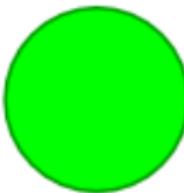
$$e_{id}^{action} = \langle id_{from}, id_{to}, \text{controller}, \text{system model}, \text{path} \rangle$$

A **empty edge**: Connect nodes that contain different objects

Proposed Method: H-Algorithm

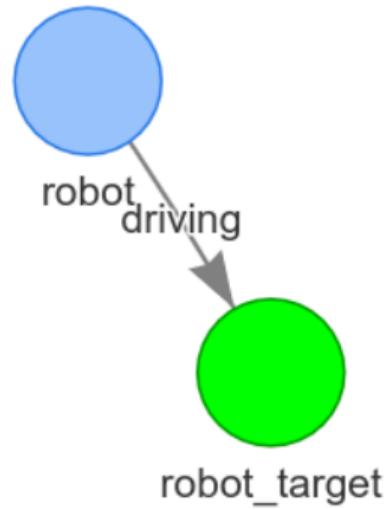
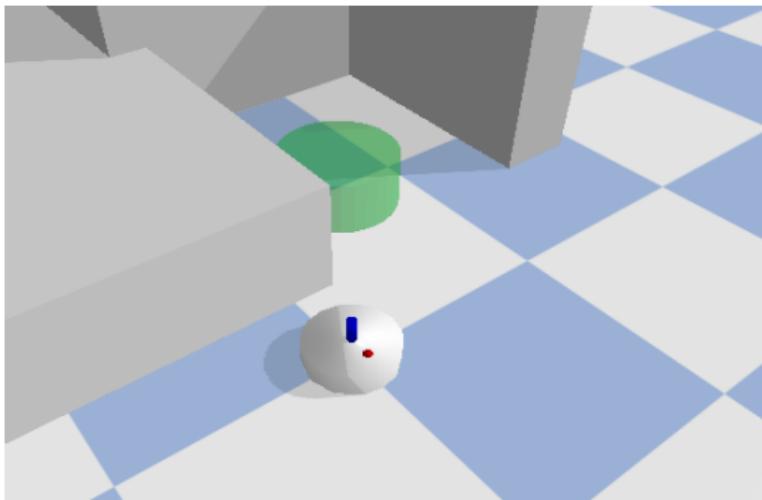


robot

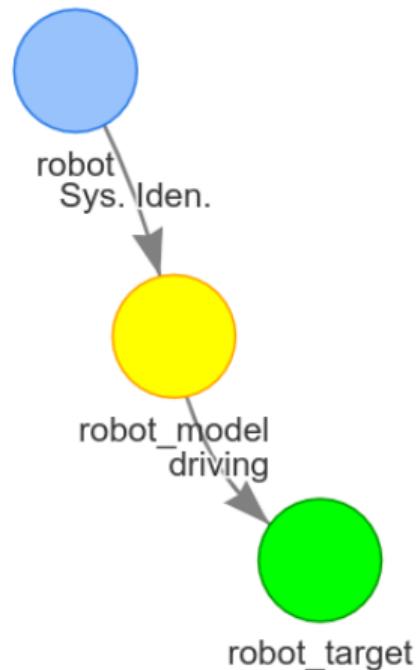
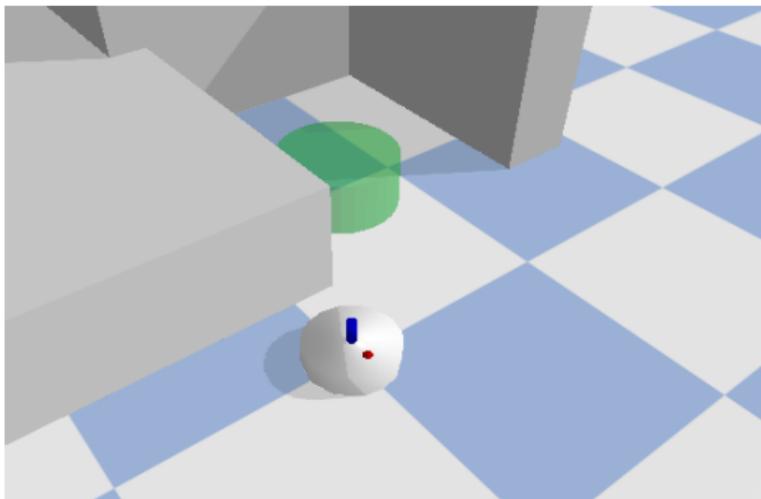


robot_target

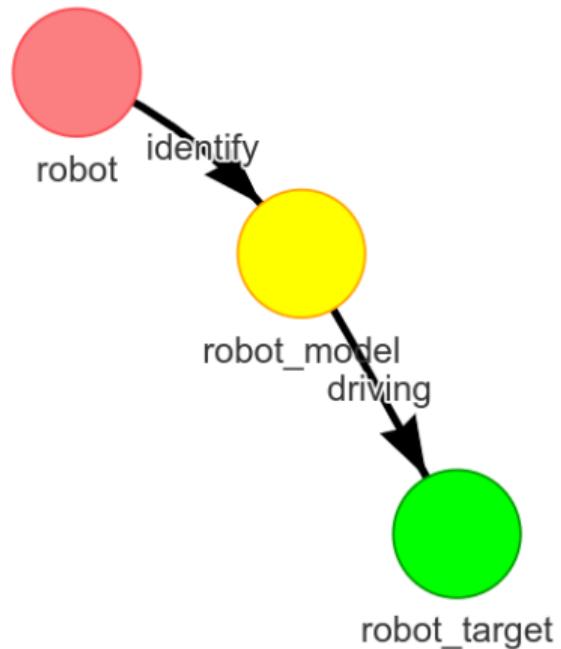
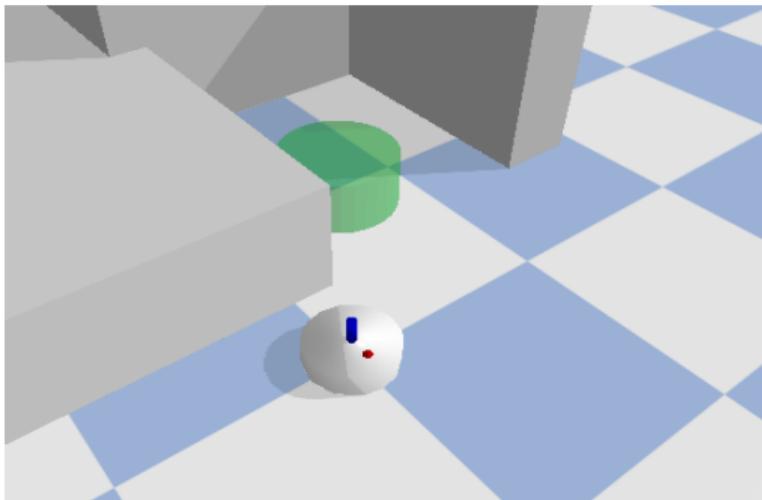
Proposed Method: H-Algorithm



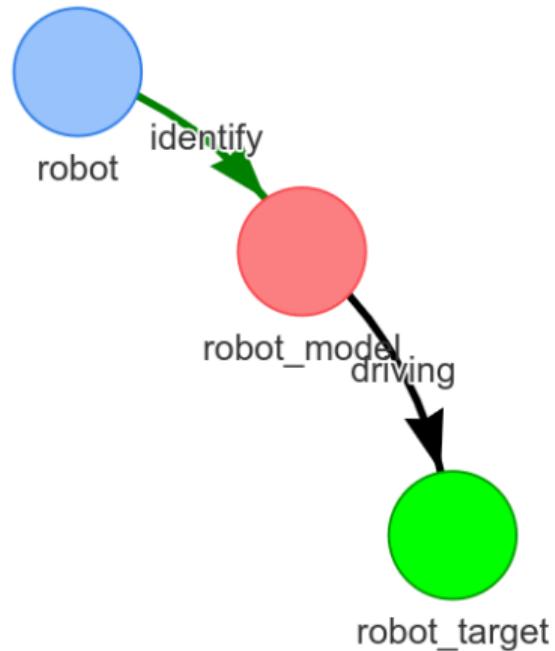
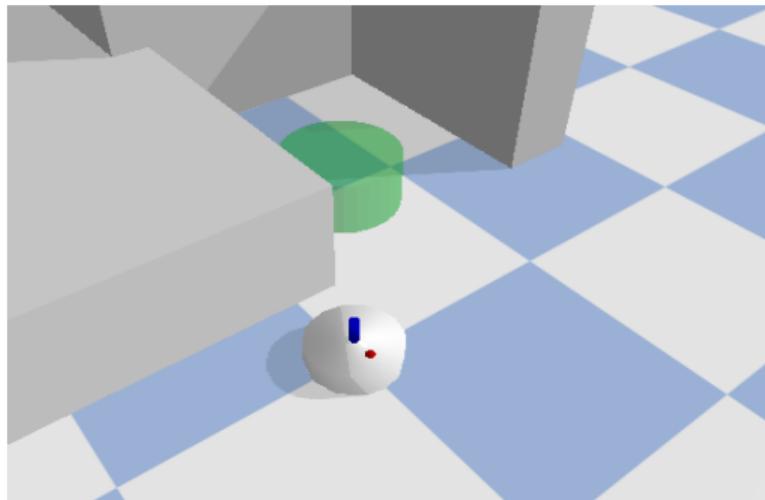
Proposed Method: H-Algorithm



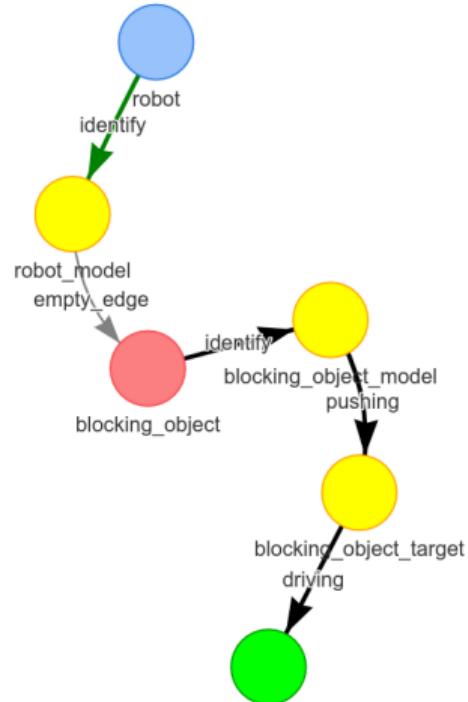
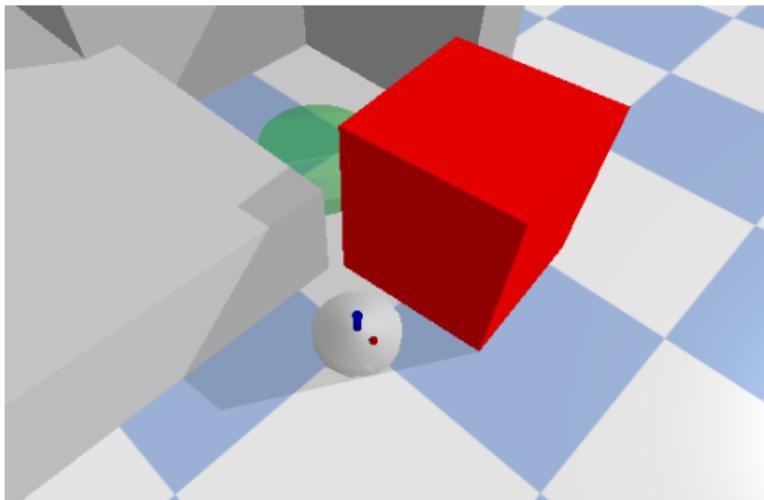
Proposed Method: H-Algorithm



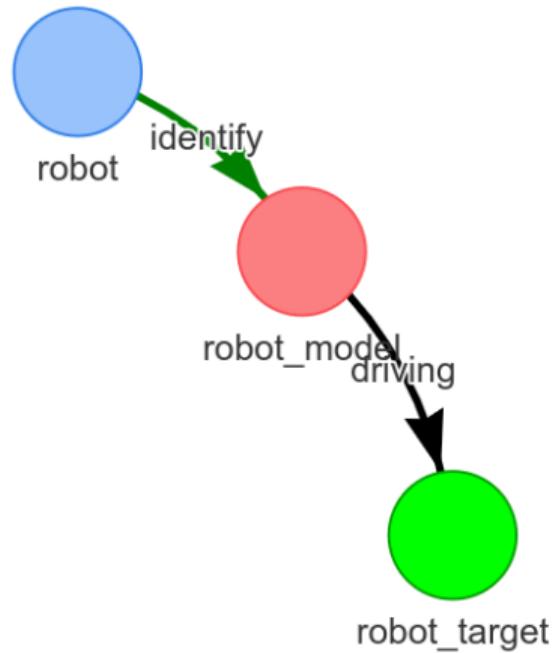
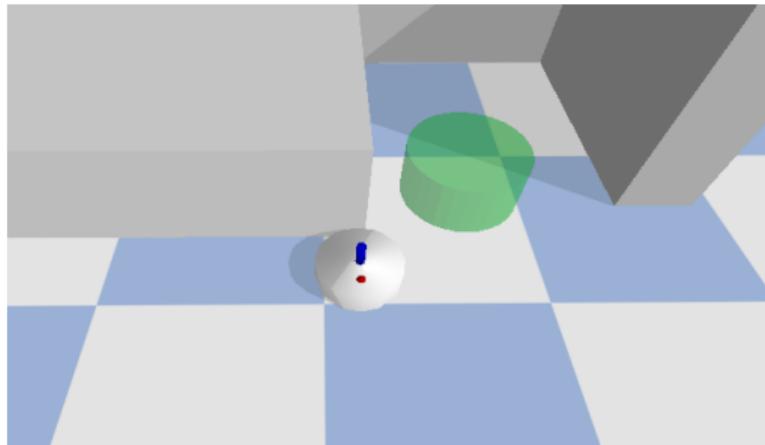
Proposed Method: H-Algorithm



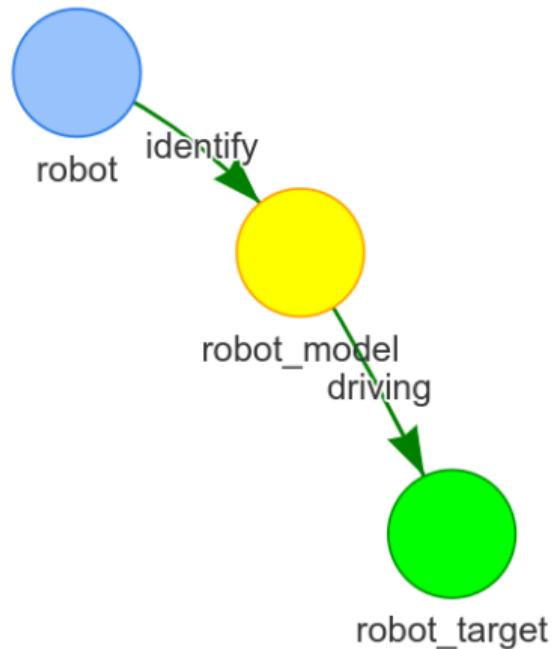
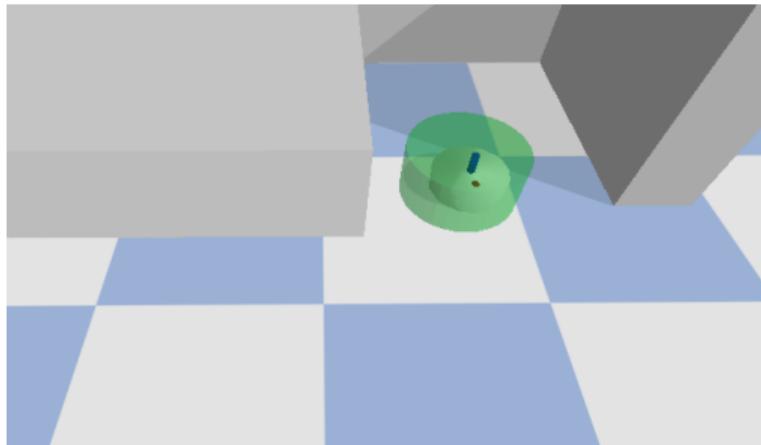
Proposed Method: H-Algorithm



Proposed Method: H-Algorithm

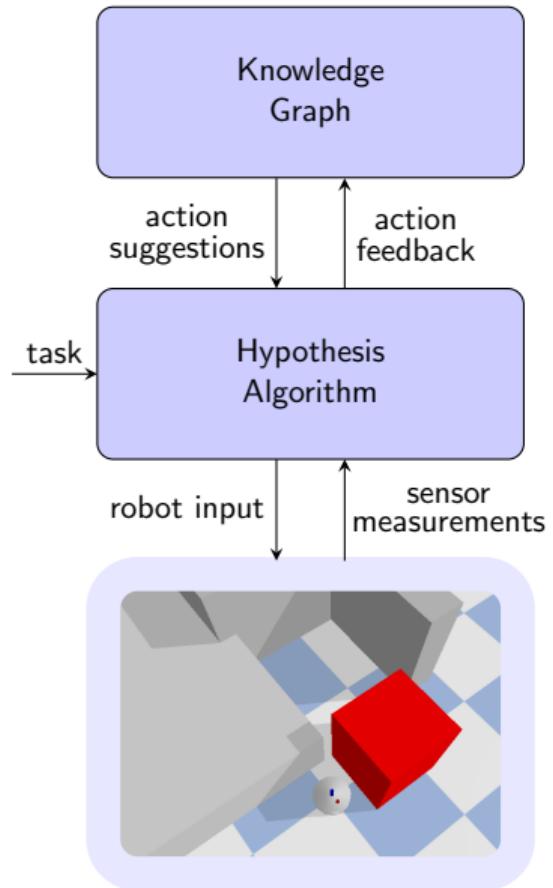


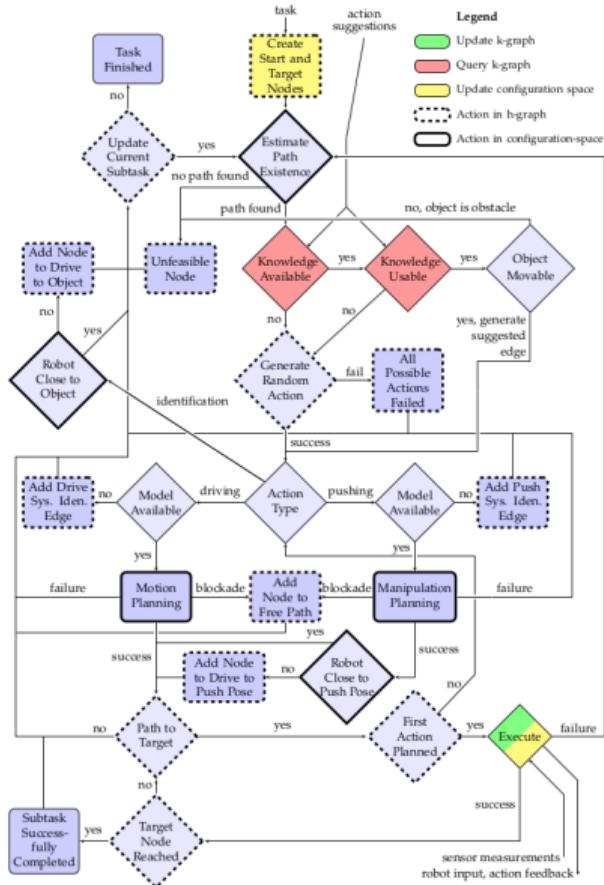
Proposed Method: H-Algorithm



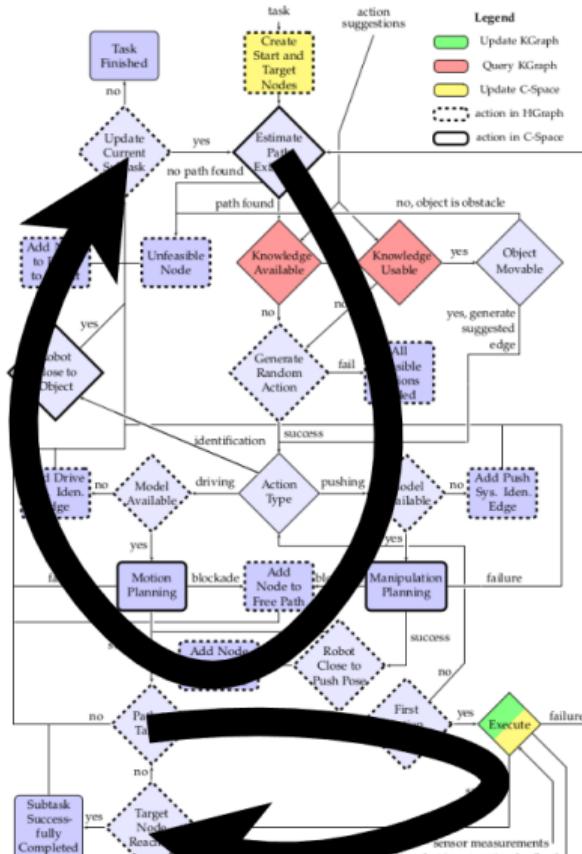
Required Background

Proposed Method





Search Loop



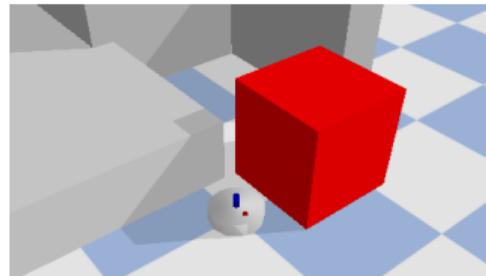
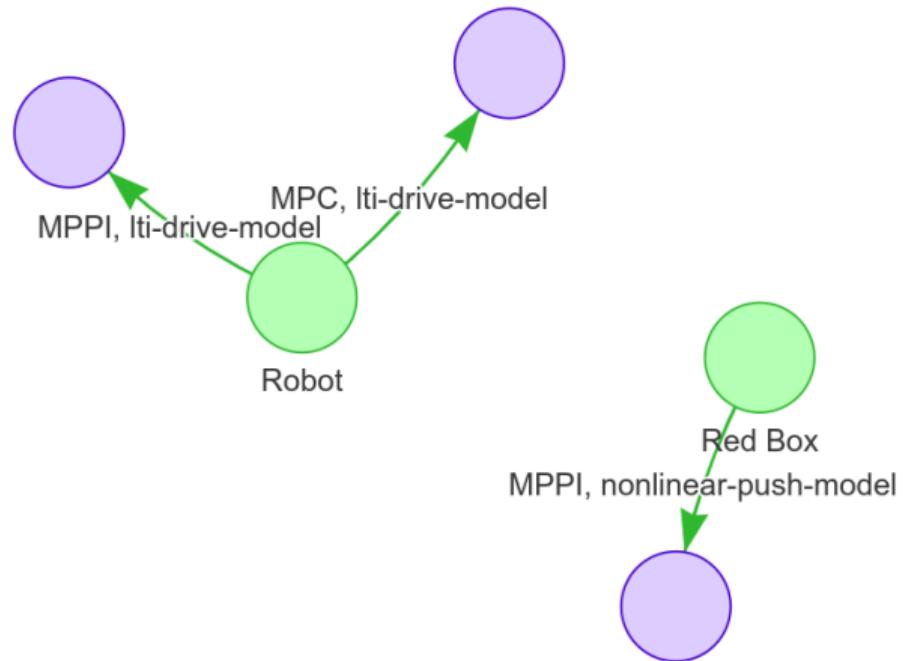
Execution Loop

Proposed Method: H-Algorithm

H-Algorithm behaviour

- Fault detection → fail edge
- Blocking obstacle → free path
- Stop regeneration of failed edges → blocklist

Proposed Method: K-Graph



Proposed Method: K-Graph

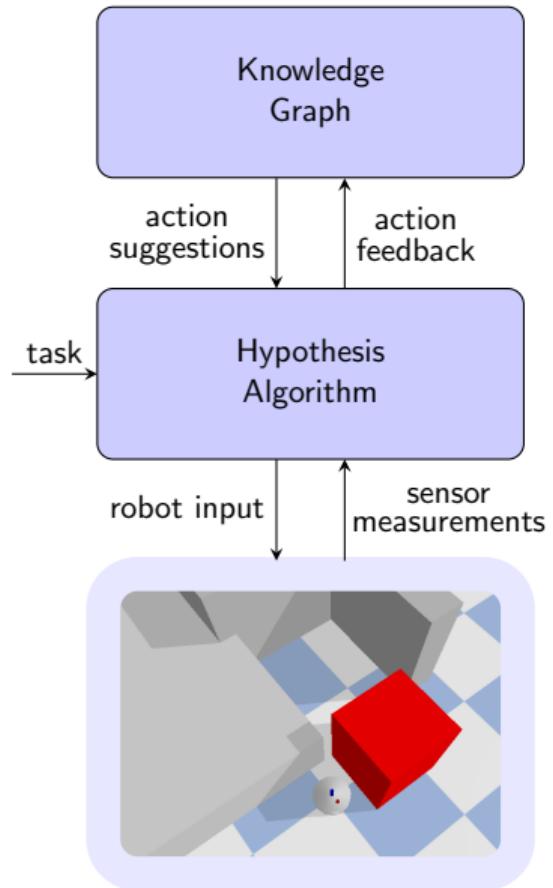
Action Feedback α :

$$\alpha(a+1) = \begin{cases} 0.1\epsilon^{pred}_{avg} & \text{if } a = 0 \\ 0.1 + 0.9\alpha(a) & \text{if } a > 0 \text{ and successfull} \\ 0.9\alpha(a) & \text{if } a > 0 \text{ and fault detected} \end{cases}$$

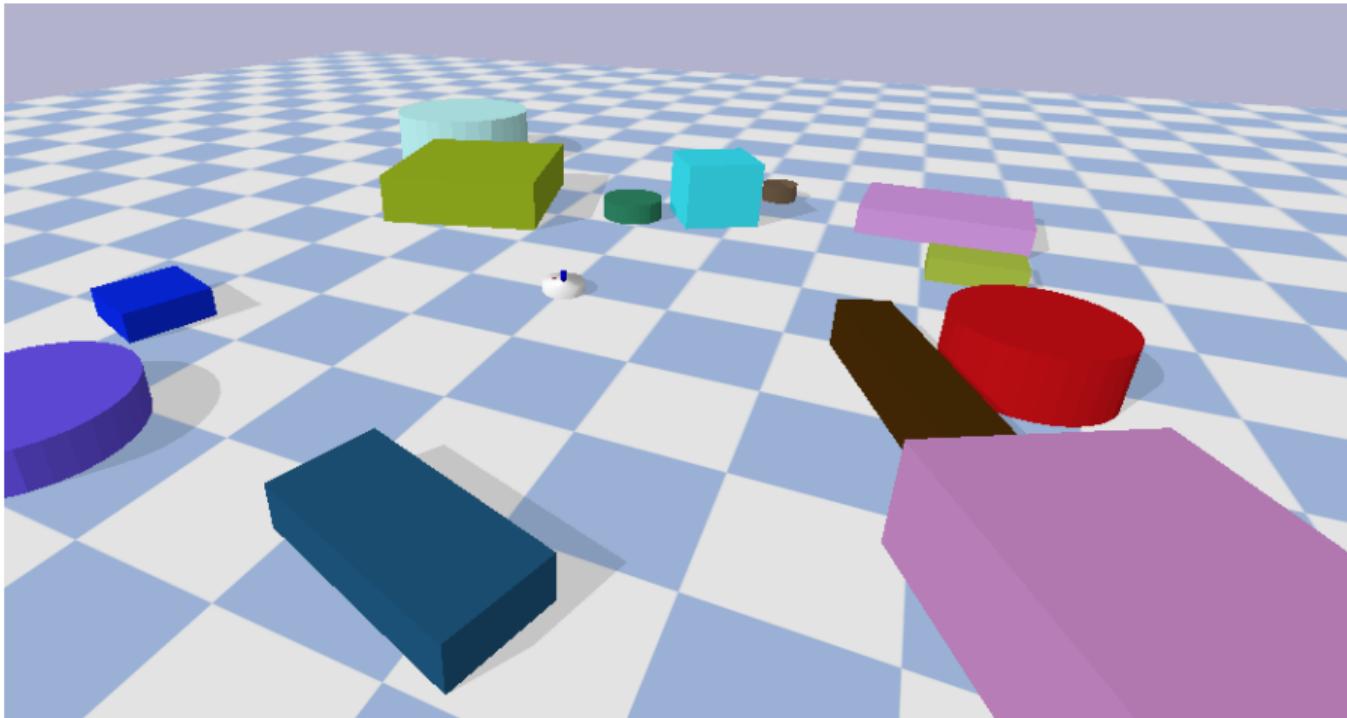
a : number of times (*controller, model*) received action feedback

ϵ^{pred}_{avg} : average prediction error

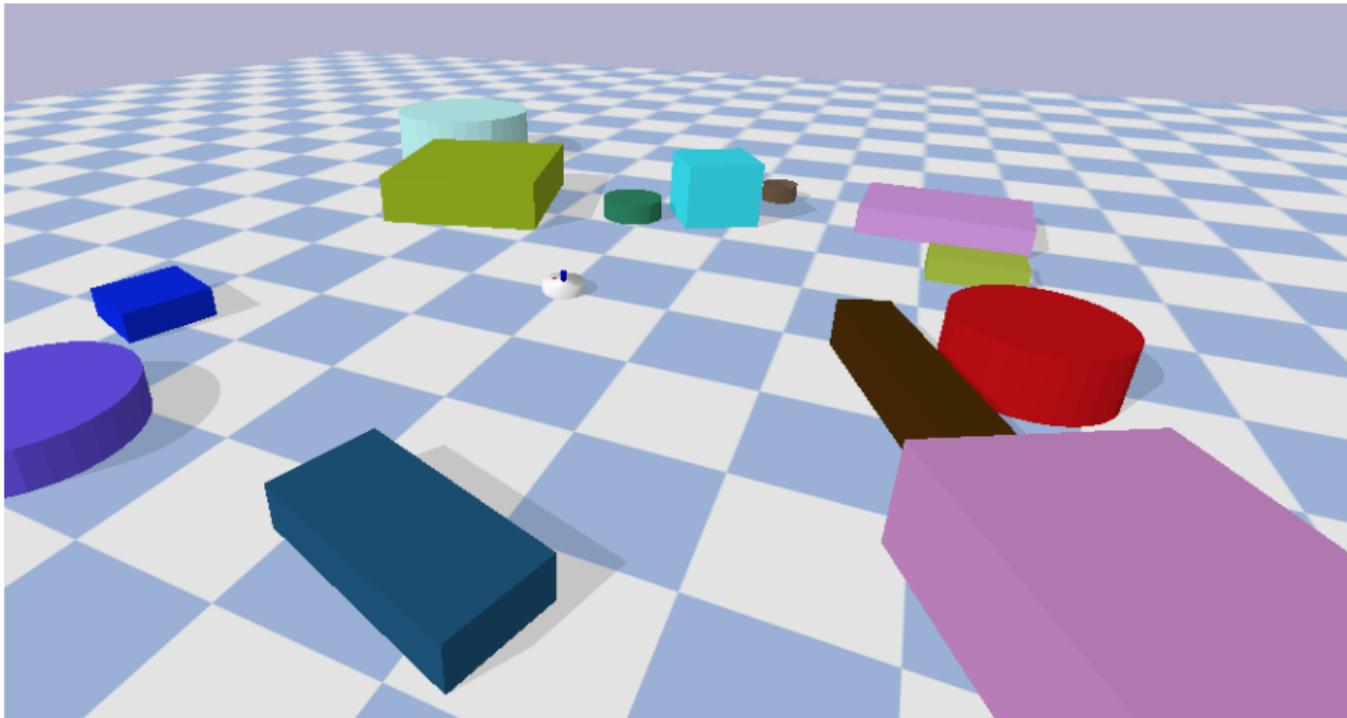
Proposed Method



Results: Driving Task

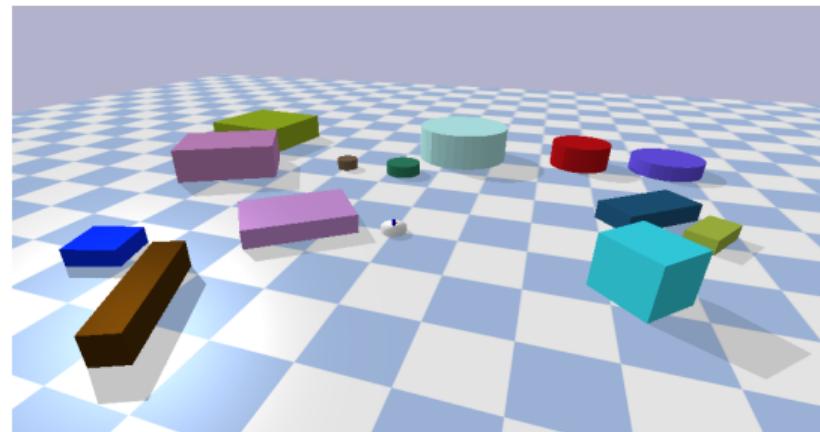
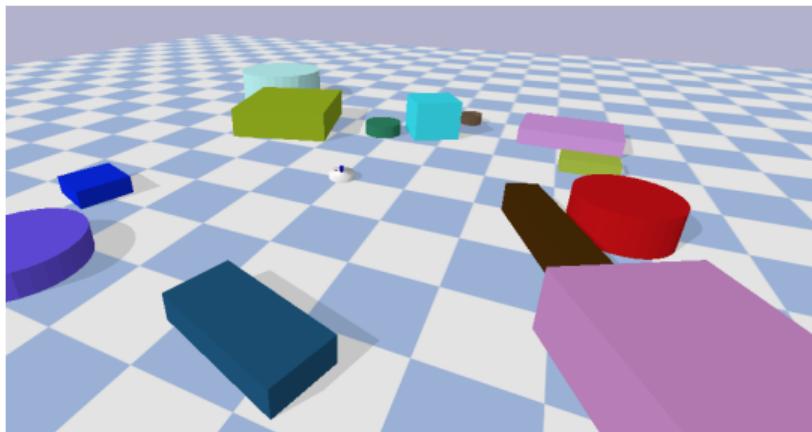


Results: Driving Task



Results: Driving Task

Reshuffle the environment



Results: Driving Task

Run	Task									
run_1	1	1	1	1	1	1	1	1	1	1
run_2	1	1	1	1	1	1	1	1	1	1
:	:	:	:	:	:	:	:	:	:	:
run_{10}	1	1	1	1	1	1	1	1	1	1
Total Tasks	10	10	10	10	10	10	10	10	10	10
Total Subtasks	30	30	30	30	30	30	30	30	30	30
Tasks in experience	0	1	2	3	4	5	6	7	8	9

Results: Driving Task

Run	Task									
run_1	1	1	1	1	1	1	1	1	1	1
run_2	1	1	1	1	1	1	1	1	1	1
:	:	:	:	:	:	:	:	:	:	:
run_{10}	1	1	1	1	1	1	1	1	1	1
Total Tasks	10	10	10	10	10	10	10	10	10	10
Total Subtasks	30	30	30	30	30	30	30	30	30	30
Tasks in experience	0	1	2	3	4	5	6	7	8	9

$$\text{number of subtasks} = 10 * 10 * 3 = 300$$

Results: Driving task

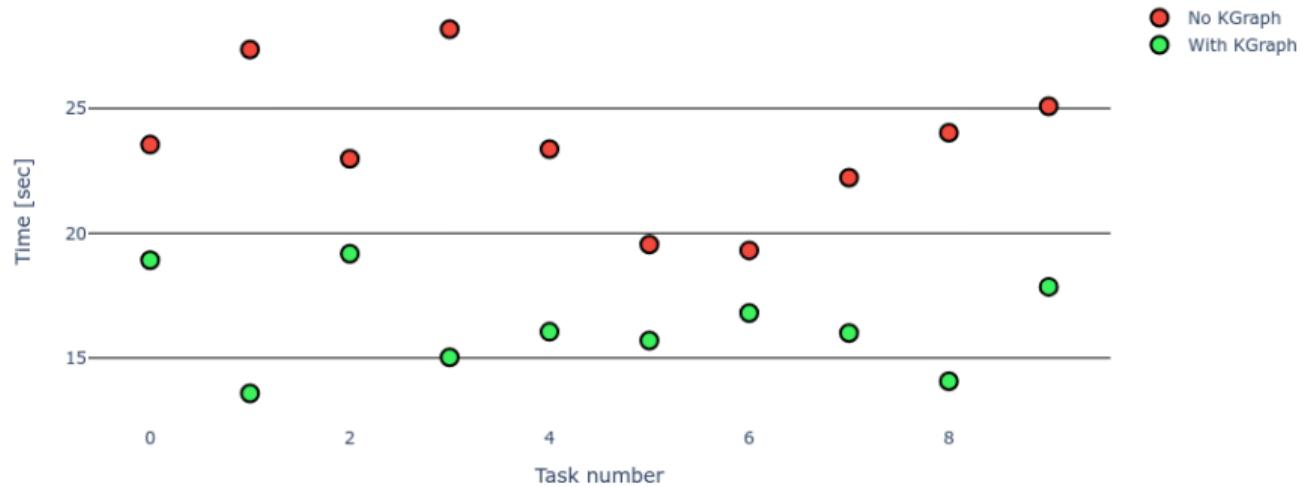
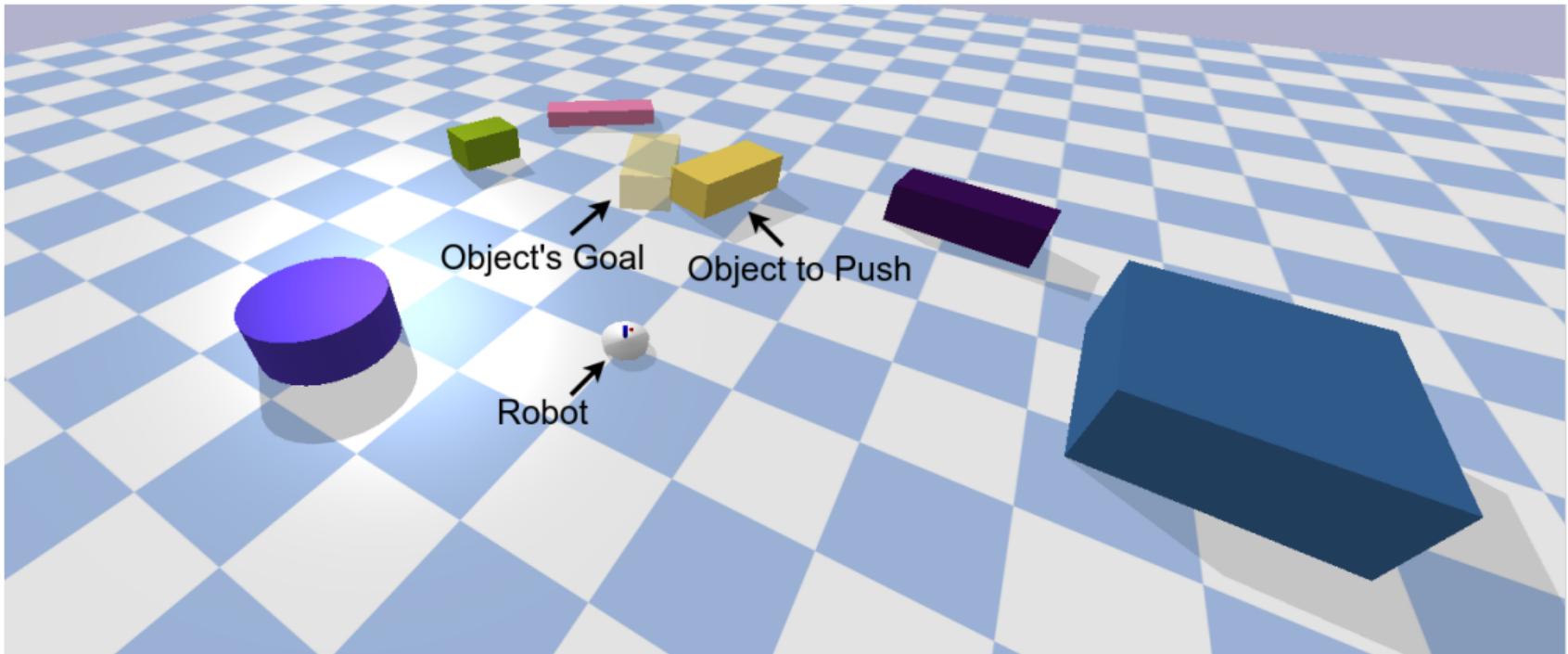


Figure: The median prediction error of 10 drive task runs

Results: Driving task

Number of Tasks in experience		0	1	2	3	4	5	6	7	8	9
With k-graph suggestions	Number of MPC parameterizations	20	30	31	31	30	30	31	30	30	33
	Number of MPPI parameterizations	10	0	0	0	0	0	0	0	0	0
Without k-graph suggestions	Number of MPC parameterizations	12	14	13	10	15	16	13	17	16	9
	Number of MPPI parameterizations	18	17	18	20	17	15	17	13	15	22

Results: Pushing task



$$\text{number of subtasks} = 10 * 6 * 1 = 60$$

Results: Pushing task

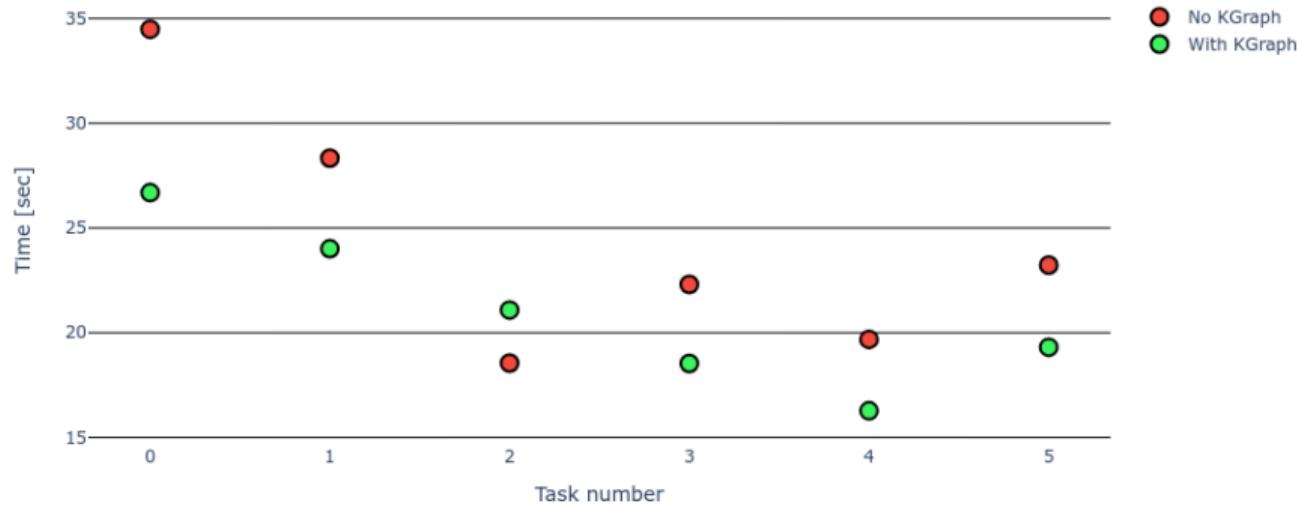


Figure: The median time of 10 runs to complete pushing tasks

Results: Pushing task

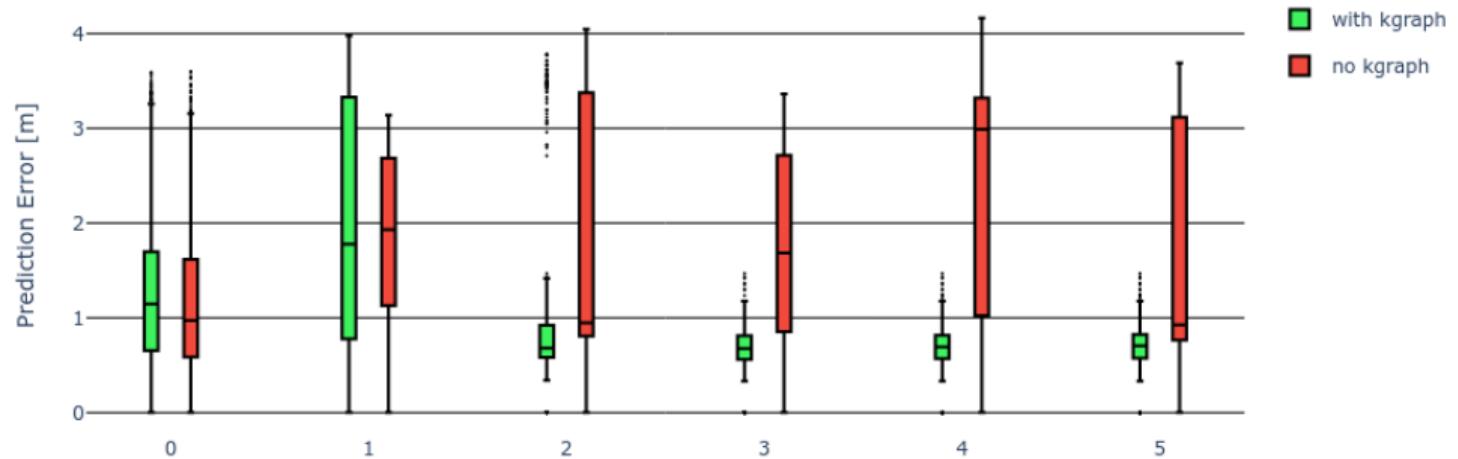
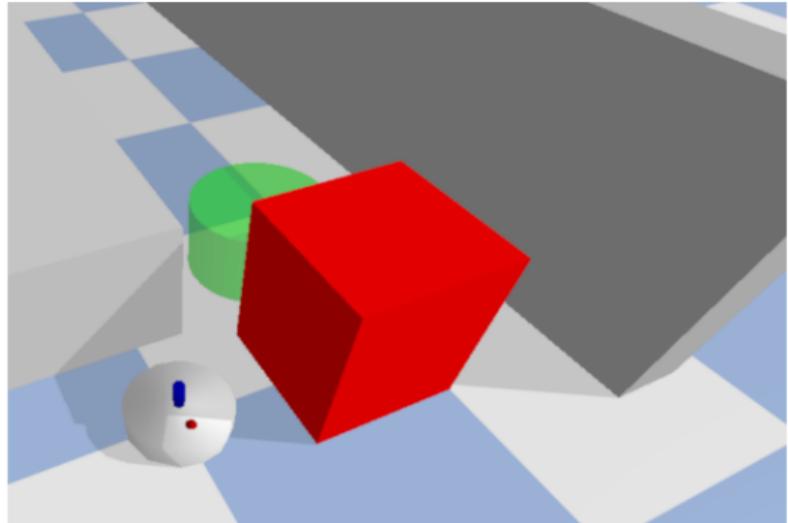
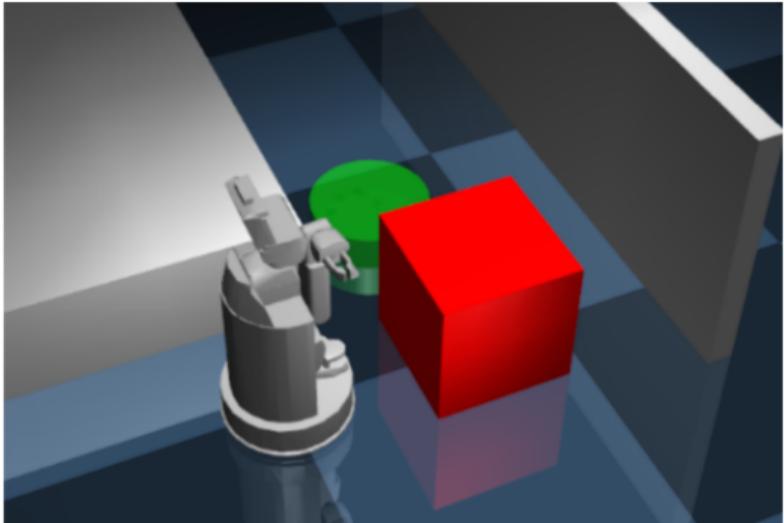


Figure: The median prediction error of 10 pushing task runs

Results: Pushing task

Number of Tasks in experience		0	1	2	3	4	5
With k-graph suggestions	Number of <i>lti-push-model</i> parameterizations	6	8	8	10	10	10
	Number of <i>nonlinear-push-model</i> parameterizations	7	9	2	0	0	0
Without k-graph suggestions	Number of <i>lti-push-model</i> parameterizations	6	8	8	6	4	6
	Number of <i>nonlinear-push-model</i> parameterizations	8	9	3	6	6	8

Results



Results

Author	Wang et al.	Groote
search time	109 sec	?
execution time	67 sec	?
total time	176 sec	?

Results

Author	Wang et al.	Groote
search time	109 sec	26 sec
execution time	67 sec	4 sec
total time	176 sec	30 sec

Conclusion

- ① Robot framework that solves tasks

Conclusion

- ① Robot framework that solves tasks
- ② Converge to best available controller

Conclusion

- ① Robot framework that solves tasks
- ② Converge to best available controller
- ③ Improvement upon existing state-of-the-art

Discussion

① More control methods

Discussion

- ① More control methods
- ② Compare with more state-of-the-art

Discussion

- ① More control methods
- ② Compare with more state-of-the-art
- ③ System identification module

Discussion

- ① More control methods
- ② Compare with more state-of-the-art
- ③ System identification module
- ④ Implement on a real robot

Questions?