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 7, 2023


TIAGo

Intro

Challenges

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

Table of Content

Intro

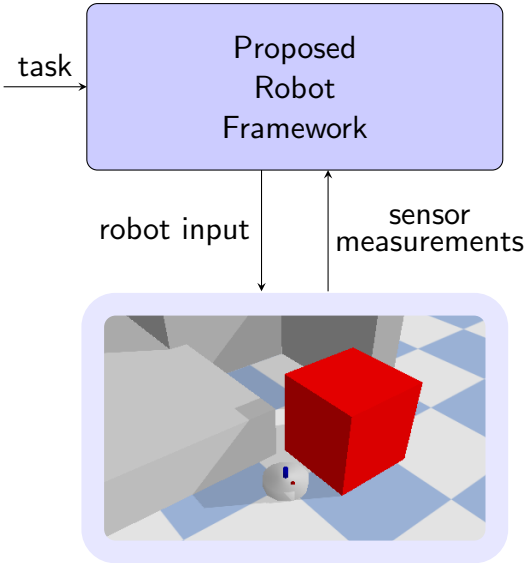
- Learn System Models
- Select best Available Controller

Intro

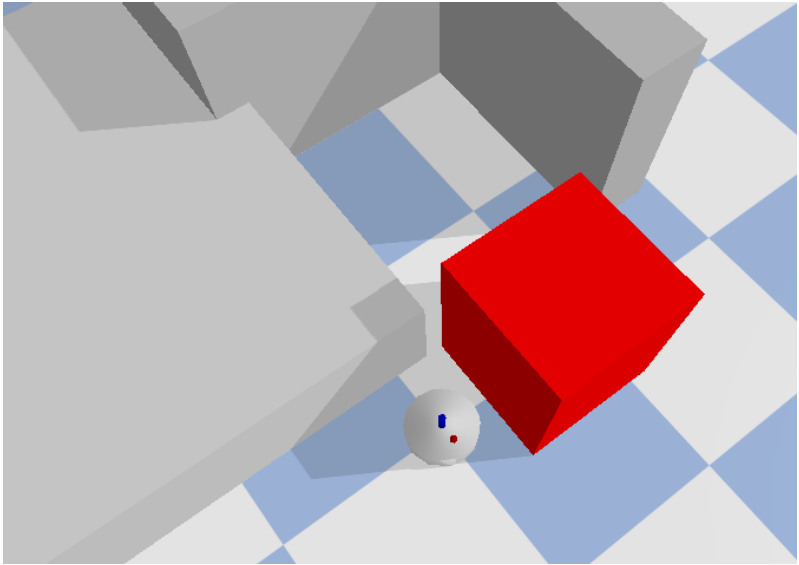
- Learn System Models
- Select best Available Controller

- Navigation Among Movable Objects (NAMO)
- Nonprehensile Pushing

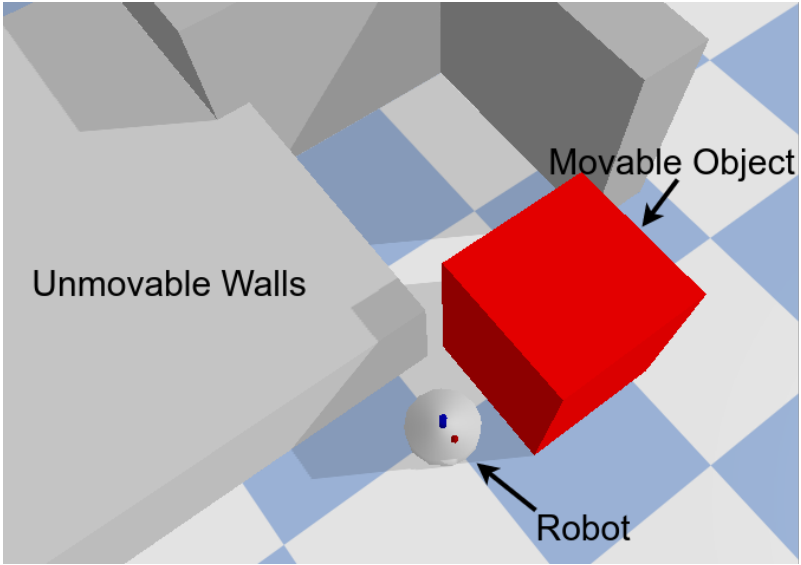
Intro


task
Proposed
Robot
Framework
robot input
sensor
measurements

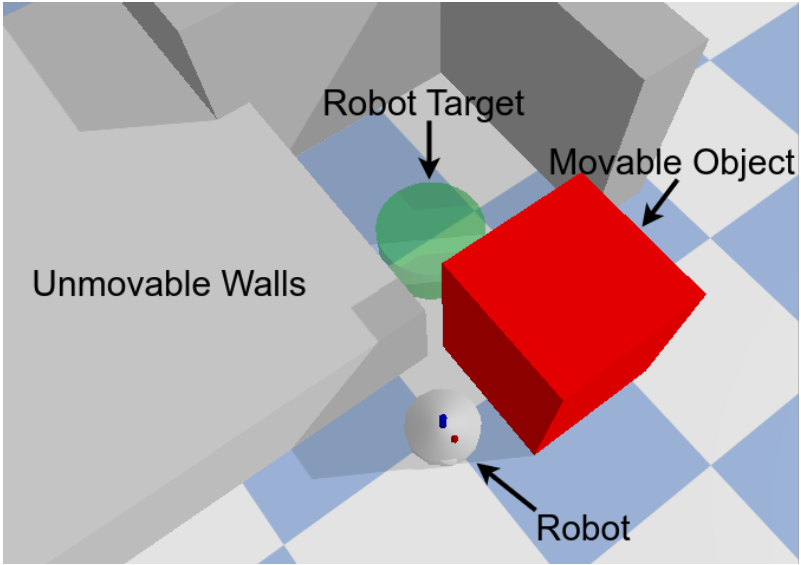
Intro



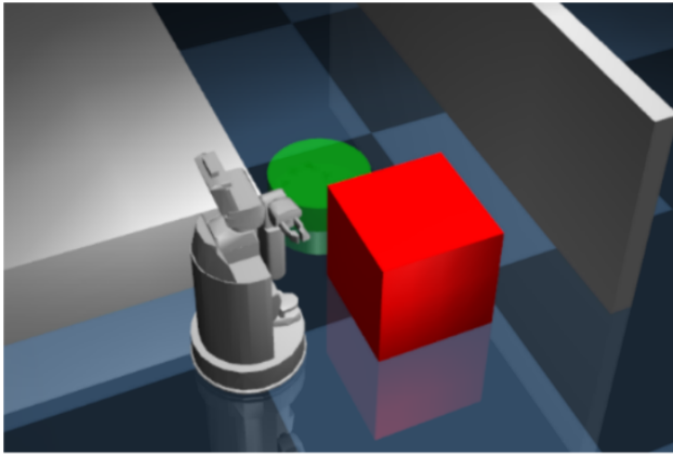
Intro



Intro



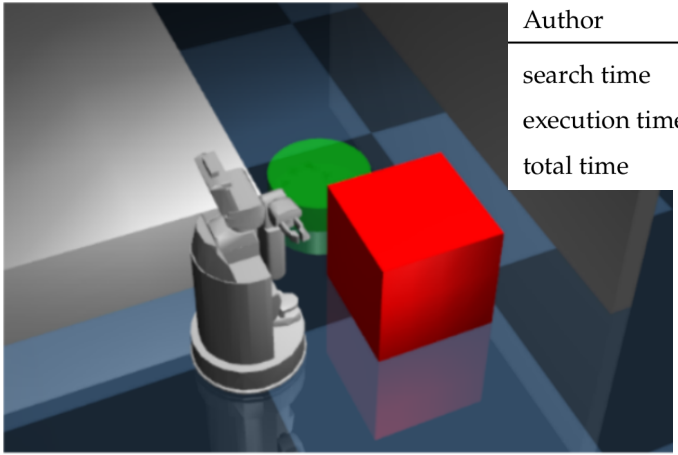
Intro



1

[1] Maozhen 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

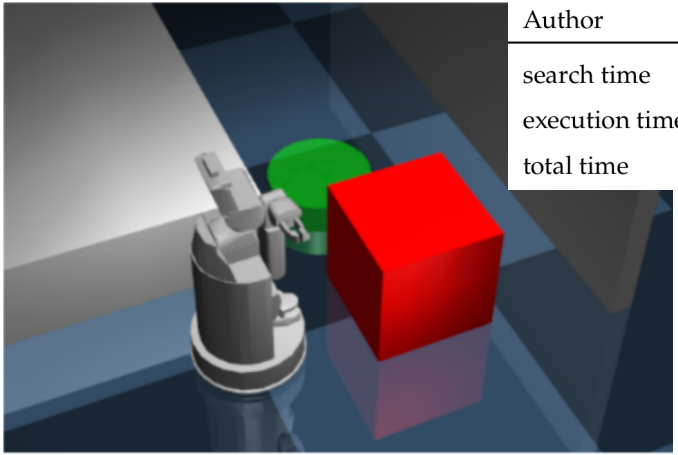


Author	Wang et al.
search time	109 sec
execution time	67 sec
total time	176 sec

1

[1] Maozhen 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



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

1

[1] Maozhen 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

Research Question:

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

Intro

Assumtions

1. **Closed-World**

Intro

Assumtions

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

Intro

Assumtions

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

Required Background

Set of Available Controllers:

$$action_1 : \{(controller_1^{action1}, model_1^{action1}), \dots\}$$
$$action_2 : \{(controller_1^{action2}, model_1^{action2}), \dots\}$$
$$\vdots$$
$$action_n : \{(controller_1^{actionN}, model_1^{actionN}), \dots\}$$

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 Intergral (MPPI) control

Required Background

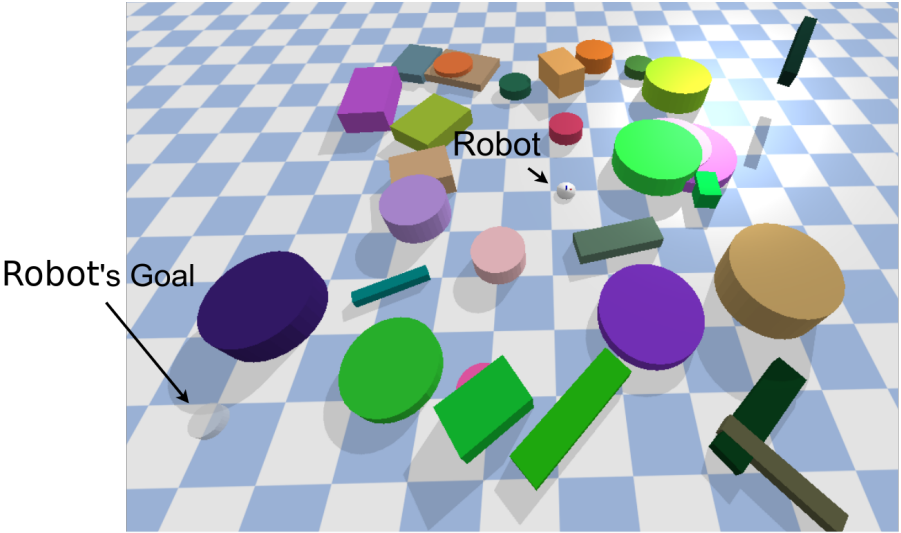
Driving

1. (MPC, *lti-drive-model*)
2. (MPPI, *lti-drive-model*)

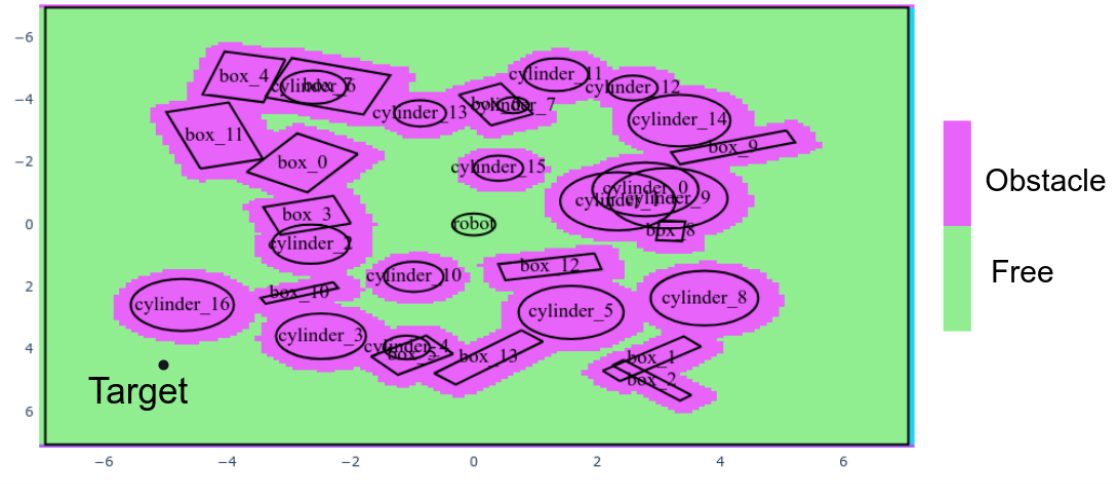
Pushing

1. (MPPI, *lti-push-model*)
2. (MPPI, *nonlinear-push-model*)

Required Background



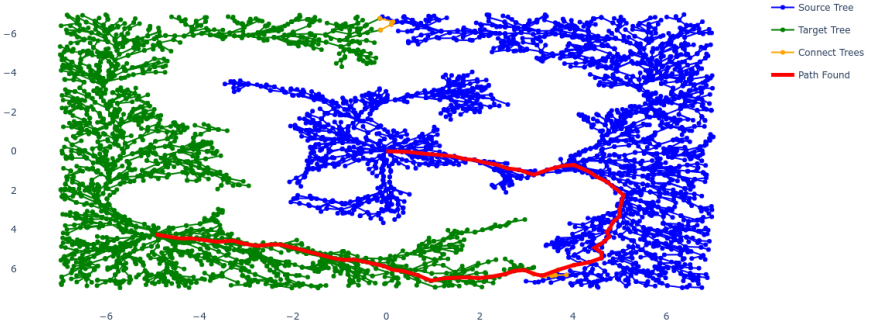
Required Background: Path Estimation



Required Background: Path Planning

Required Background: Path Planning

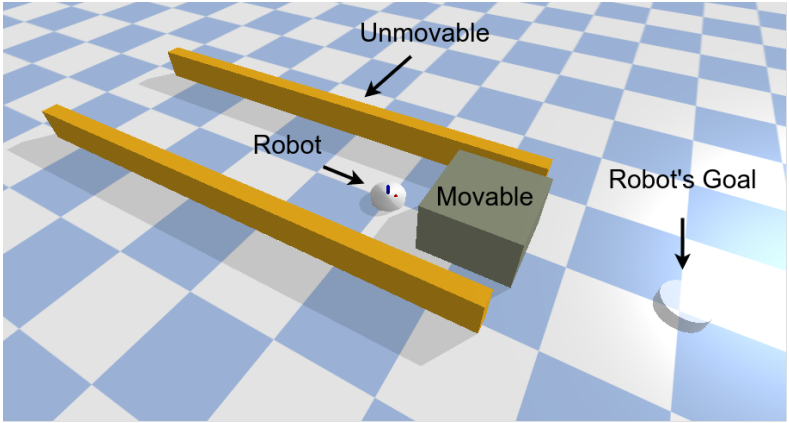
Double-tree Rapidly exploring Random Tree (RRT*) Algorithm [2]:



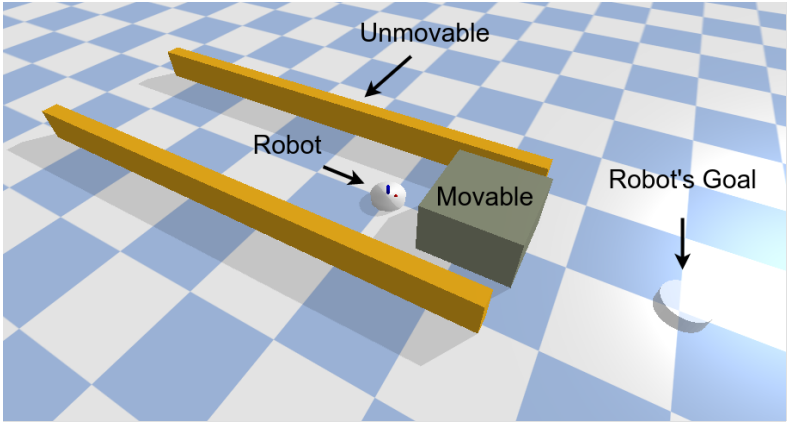
$$Cost_{path} = \sum_{i=1}^{n-1} Distance(c_i, c_{i+1})$$

[2] Long 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

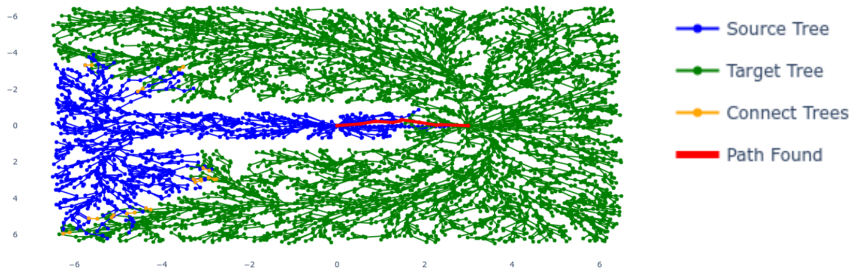
Proposed Method



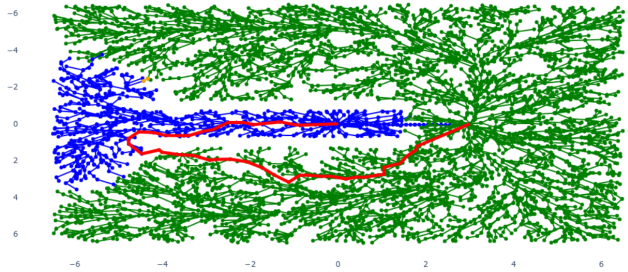
Proposed Method



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

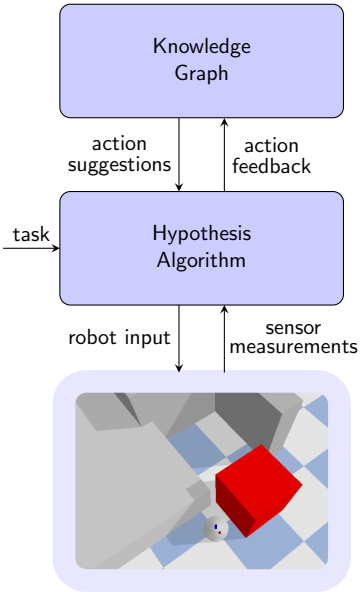


$MovableSpaceCost = 0.5m$



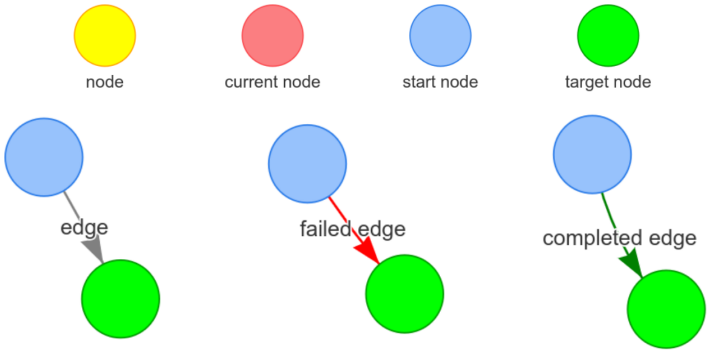
$MovableSpaceCost = 1.0m$

Proposed Method



Proposed Method

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



Proposed Method: H-Graph

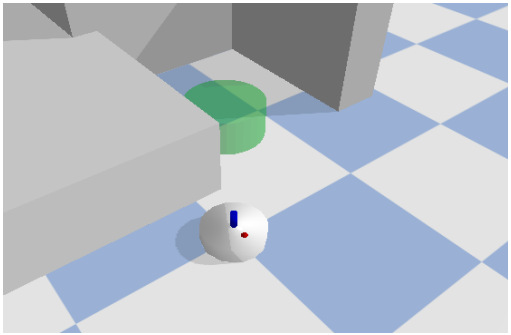
A **identification edge**: Creates a system model

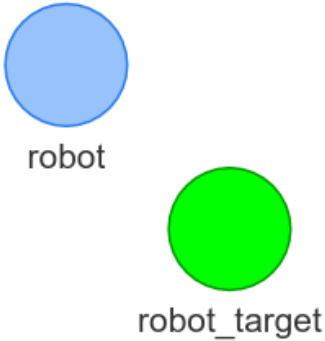
A **action edge**:

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

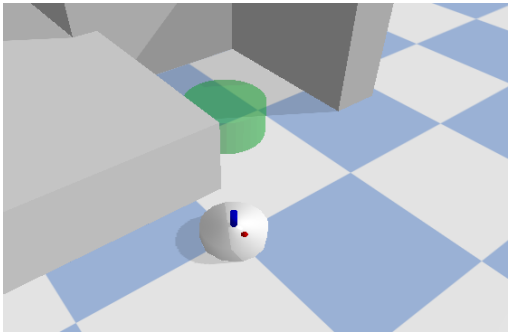
A **empty edge**: Connect nodes that contain different objects

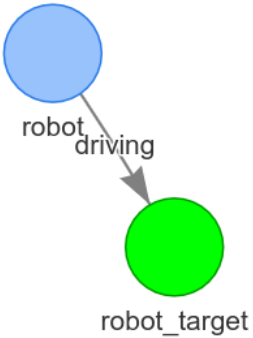
Proposed Method: H-Algorithm



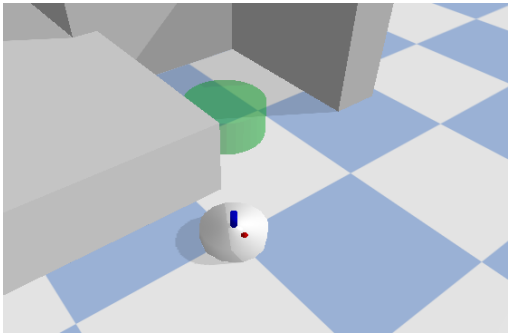


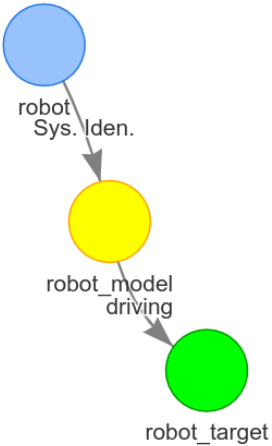
Proposed Method: H-Algorithm



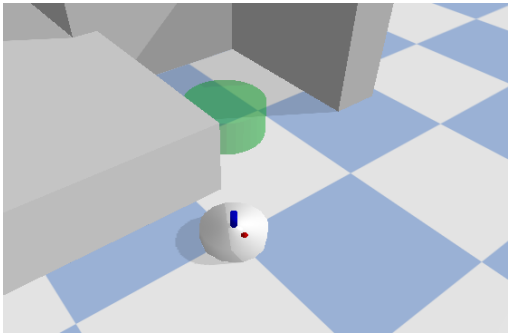


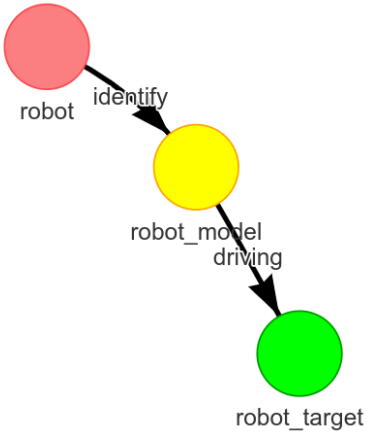
Proposed Method: H-Algorithm



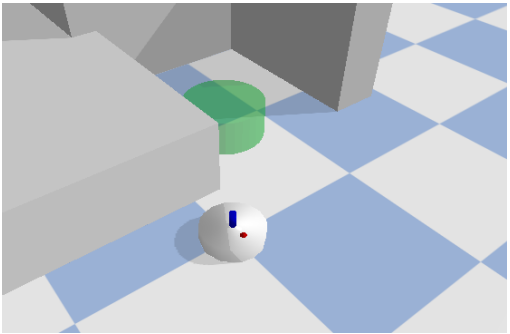


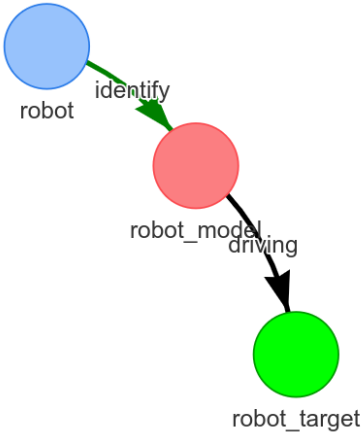
Proposed Method: H-Algorithm



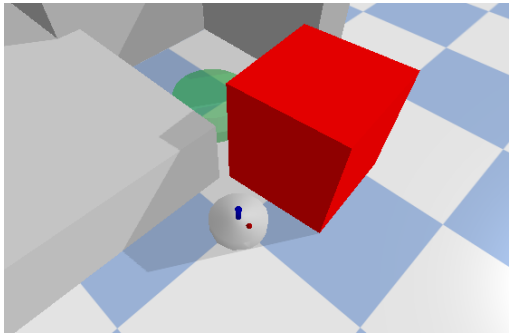


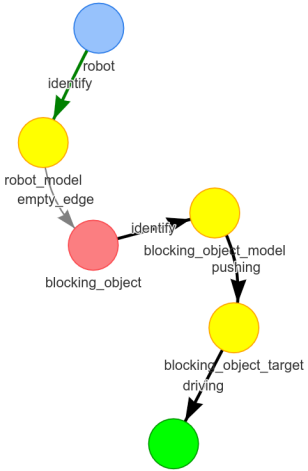
Proposed Method: H-Algorithm



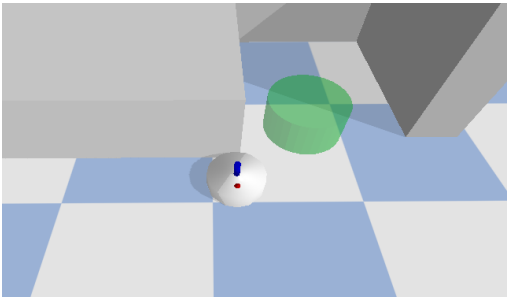


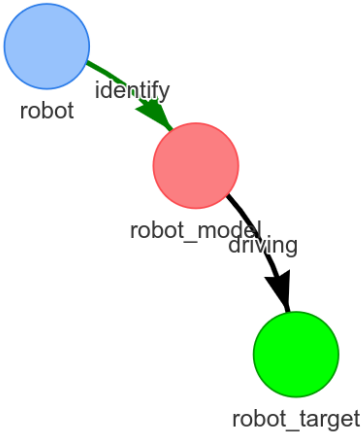
Proposed Method: H-Algorithm



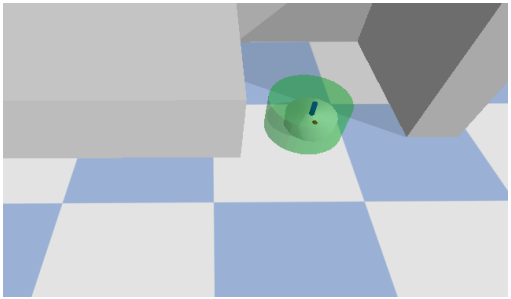


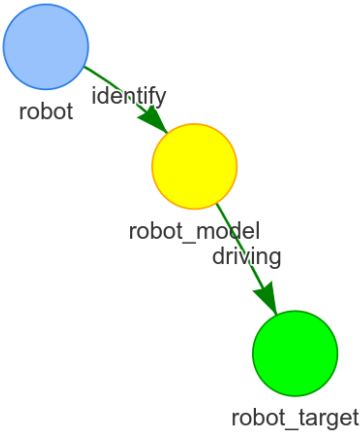
Proposed Method: H-Algorithm





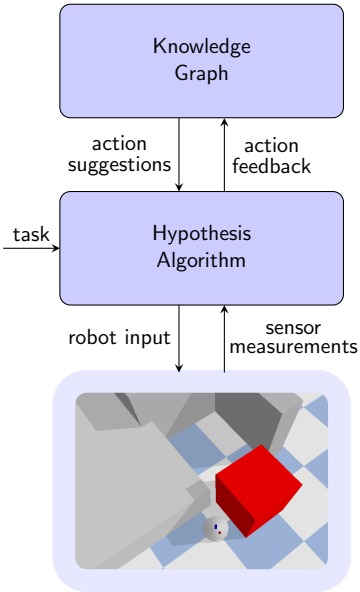
Proposed Method: H-Algorithm

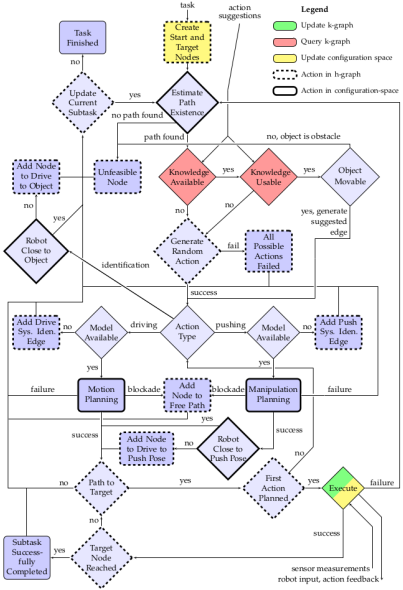




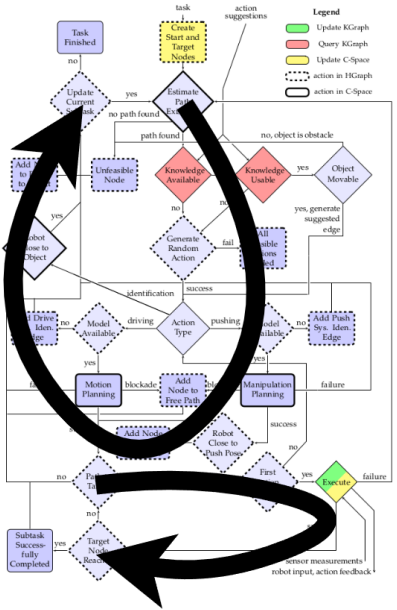
Required Background

Proposed Method




task
action suggestions
Legend
Update k-graph
Query k-graph
Update configuration space
Action in h-graph
Action in configuration-space
Task Finished
Create Start and Target Nodes
no
Update Current Subtask
yes
Estimate Path Existence
no path found
path found
no, object is obstacle
Add Node to Drive to Object
Unfeasible Node
Knowledge Available
yes
Knowledge Usable
yes
Object Movable
no
no
yes, generate suggested edge
yes
Robot Close to Object
Generate Random Action
fail
All Possible Actions Failed
identification
success
Add Drive Sys. Iden. Edge
no
Model Available
driving
Action Type
pushing
Model Available
no
Add Push Sys. Iden. Edge
yes
yes
failure
Motion Planning
blockade
Add Node to Free Path
blockade
Manipulation Planning
failure
yes
success
Add Node to Drive to Push Pose
no
Robot Close to Push Pose
success
no
no
Path to Target
yes
First Action Planned
yes
Execute
failure
no
success
Subtask Successfully Completed
yes
Target Node Reached
sensor measurements
robot input, action feedback

Search Loop



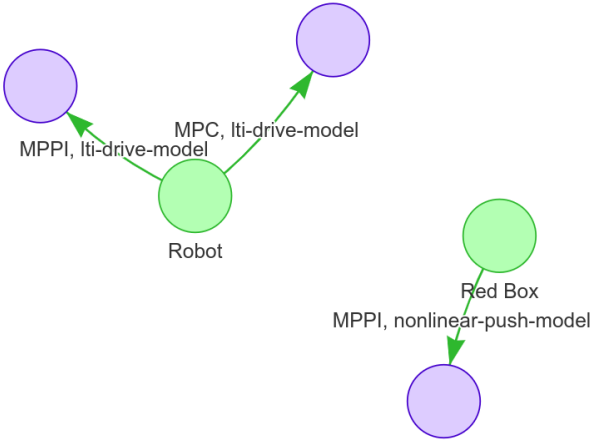
Execution Loop

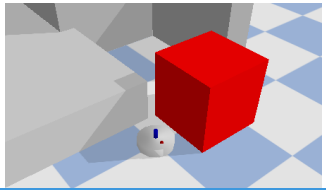
Proposed Method: H-Algorithm

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

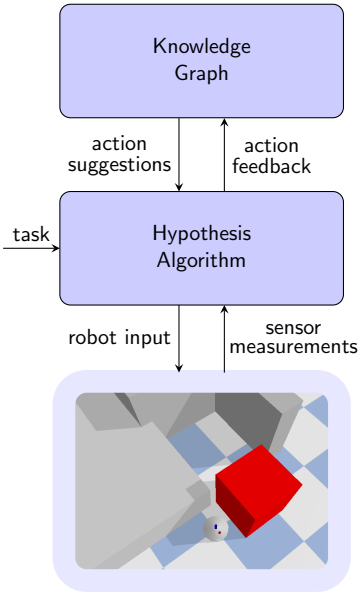
Action Feedback α:

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

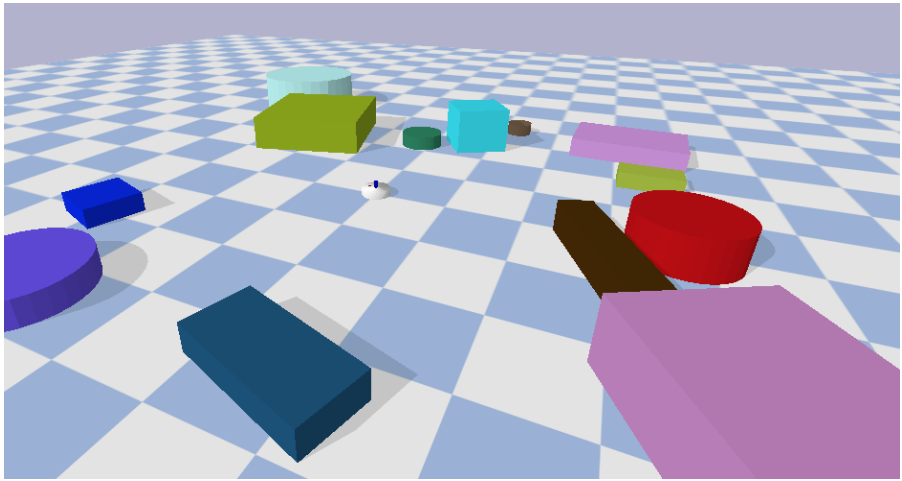
a: number of times (*controller*, *model*) received action feedback

$\epsilon^{pred}{}_{avg}$: average prediction error

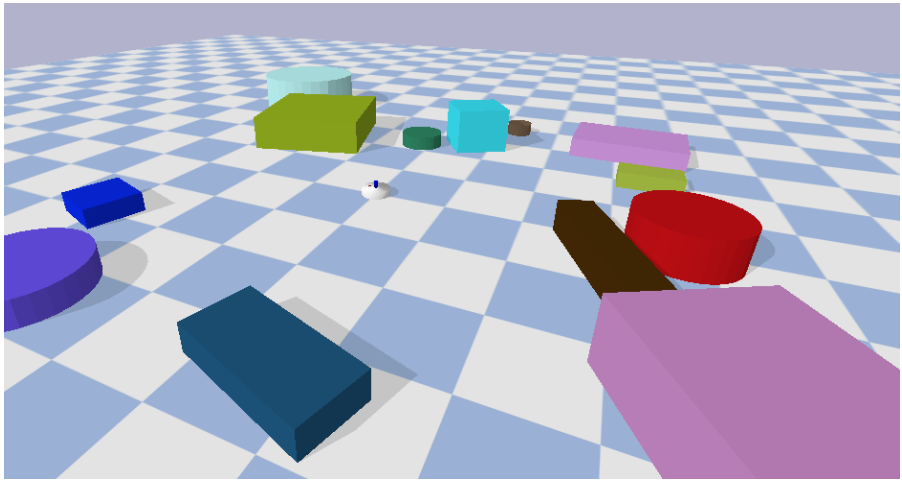
Proposed Method



Results: Driving Task

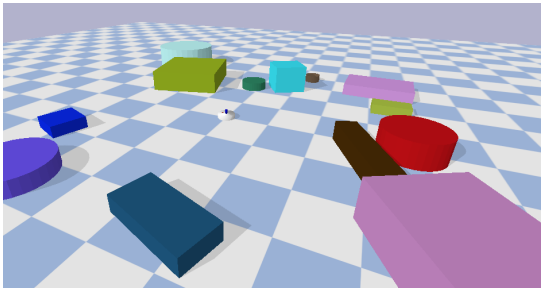


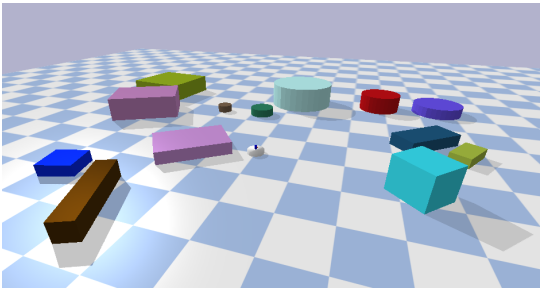
Results: Driving Task



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

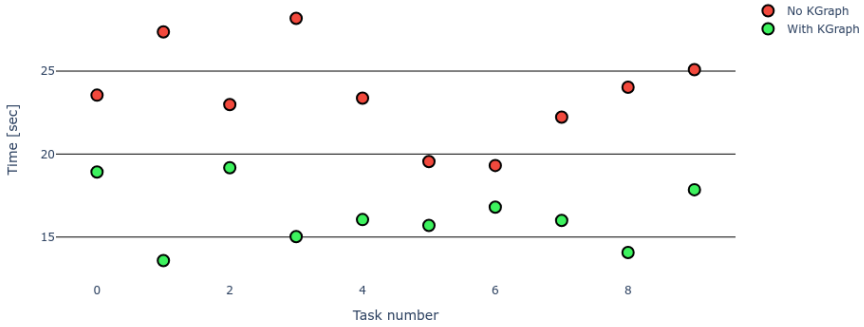
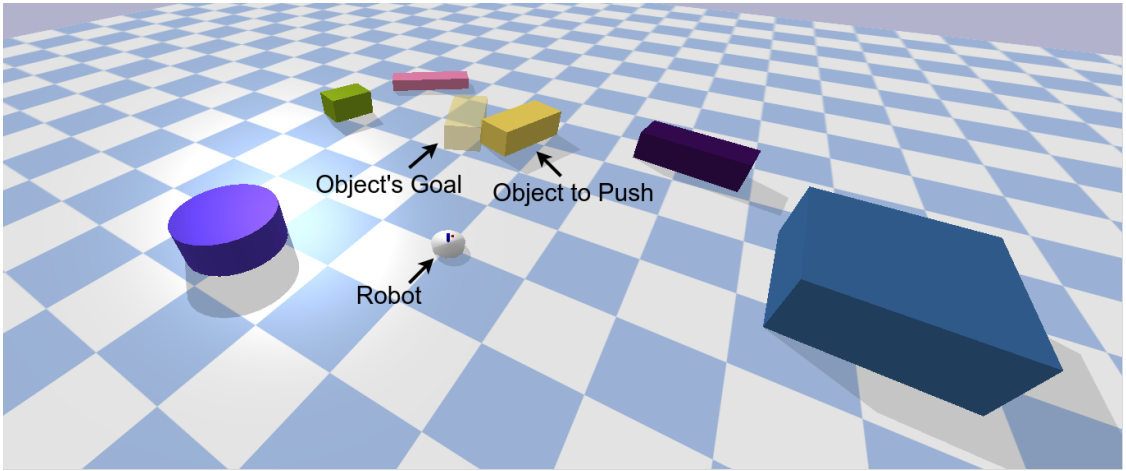


Figure: The median prediction error of 10 drive task runs

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

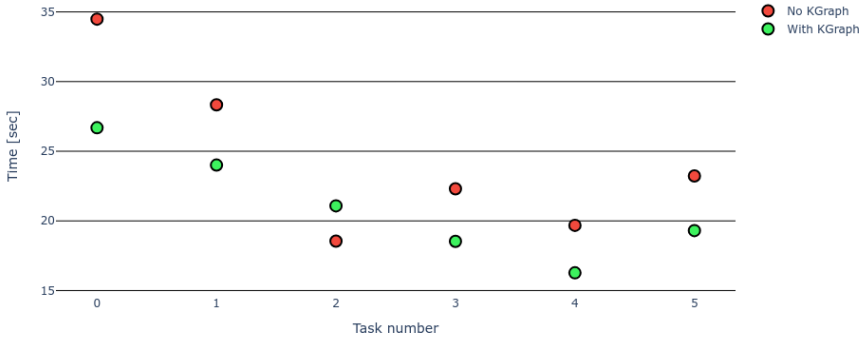


Figure: The median time of 10 runs to complete pushing tasks

Results: Pushing task

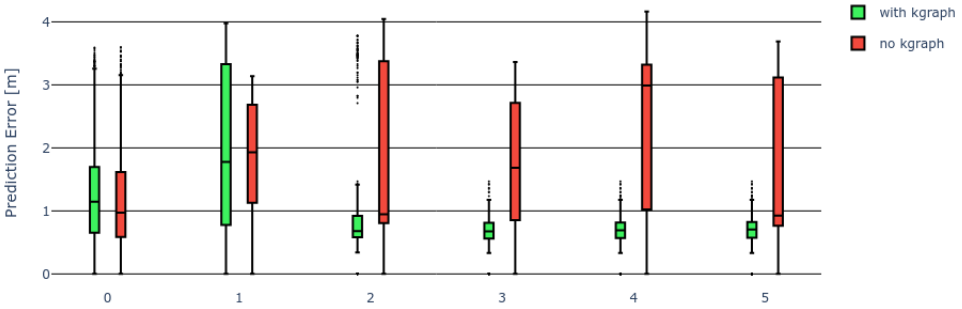
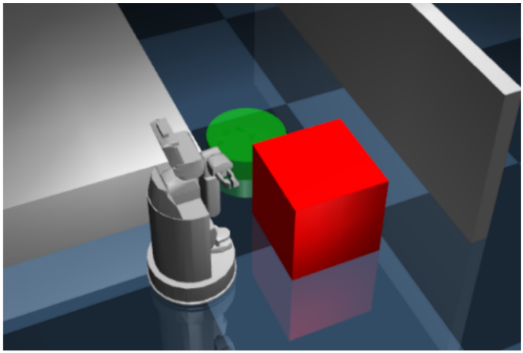


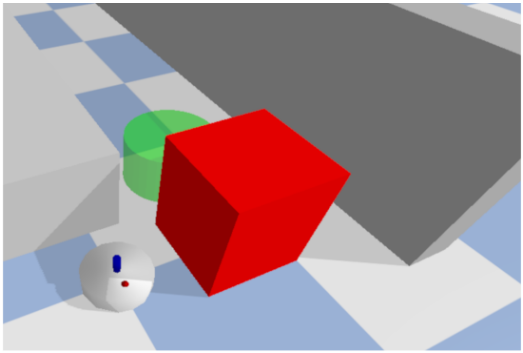
Figure: The median prediction error of 10 pushing task runs

Results: Pushing task

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