dfki ai Deutsches Forschungszentrum für Künstliche Intelligenz *German Research Center for Artificial Intelligence* 1

UNIVERSITÄT OSNABRÜCK 2

UNIVERSITY OF MARYLAND 3

FONDAZIONE BRUNO KESSLER 5

LAAS CNRS 4

Acting and Planning with Hierarchical Operational Models on a Mobile Robot: A Study with RAE+UPOM

Oscar Lima[1,2,*], Marc Vinci[1,2,*], Sunandita Patra[3,*], Sebastian Stock[1], Joachim Hertzberg[2,1], Martin Atzmueller[2,1], Malik Ghallab[4], Dana Nau[3], Paolo Traverso[5]

SPONSORED BY THE Federal Ministry of Education and Research

September 2025

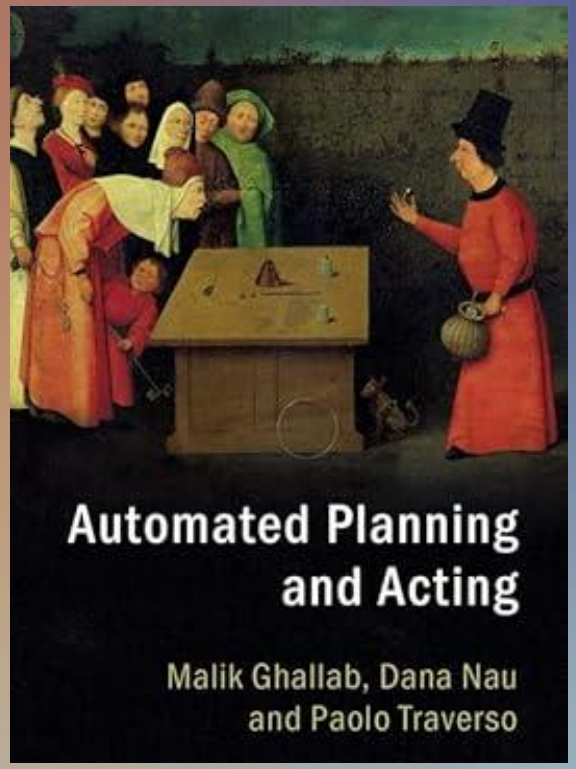
Introduction

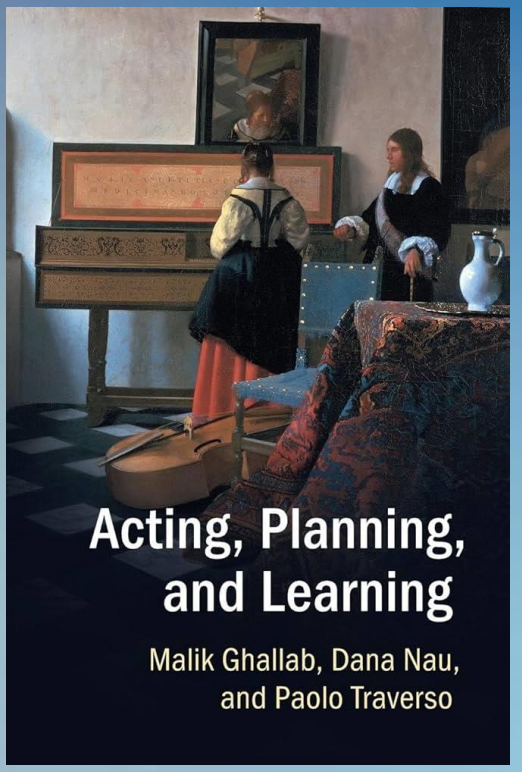
- *"Most of the existing work on automated planning underestimates the reasoning and deliberation needed for acting."* [1]

- Planning researchers have developed many planners, but very few actors. [1]

- This gap may contribute to the limited real-world deployment of planning systems.

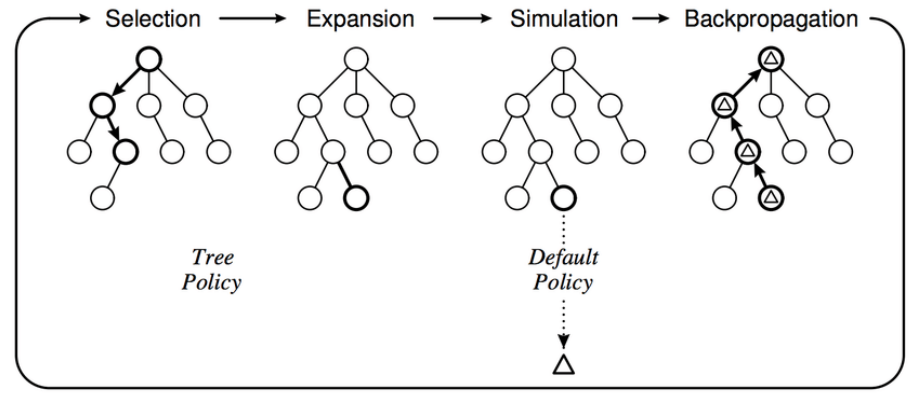
[1] Ghallab, M., Nau, D., & Traverso, P. (2014). The actor's view of automated planning and acting: A position paper. *Artificial Intelligence, 208, 1–17.*





Previous work: RAE + UPOM

- Acting, Planning and Learning with Operational Models
  - https://github.com/patras91/rae_release
- RAE: Reactive Acting Engine
- UPOM: UCT-like probabilistic planner that works with operational models
- Uses MCT search underneath



Deliberative Acting, Planning and Learning with Hierarchical Operational Models

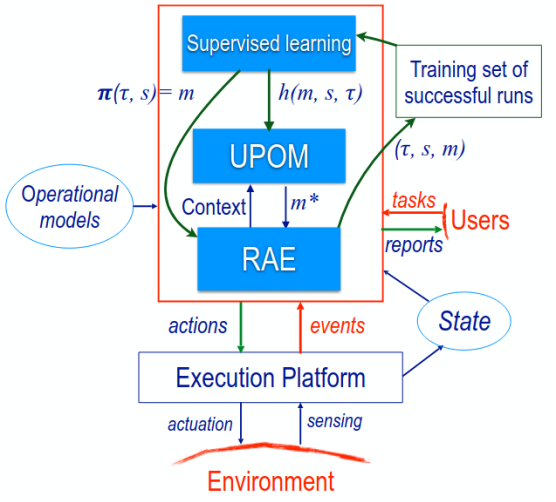
Sunandita Patra[a,*], James Mason[a], Malik Ghallab[b], Dana Nau[a], Paolo Traverso[c]

[a]Dept. of Computer Science and Inst. for Systems Research, University of Maryland, College Park, Maryland, USA
[b]LAAS-CNRS, Toulouse, France
[c]FBK-ICT, Povo-Trento, Italy

Abstract

In AI research, synthesizing a plan of action has typically used *descriptive models* of the actions that abstractly specify *what* might happen as a result of an action, and are tailored for efficiently computing state transitions. However, executing the planned actions has needed *operational* models, in which rich computational control structures and closed-loop online decision-making are used to specify *how* to perform an action in a nondeterministic execution context, react to events and adapt to an unfolding situation. *Deliberative actors*, which integrate acting and planning, have typically needed to use both of t
which causes problems when attempting to develop th
their consistency, and smoothly interleave acting and p

PhD thesis
of Sunandita Patra

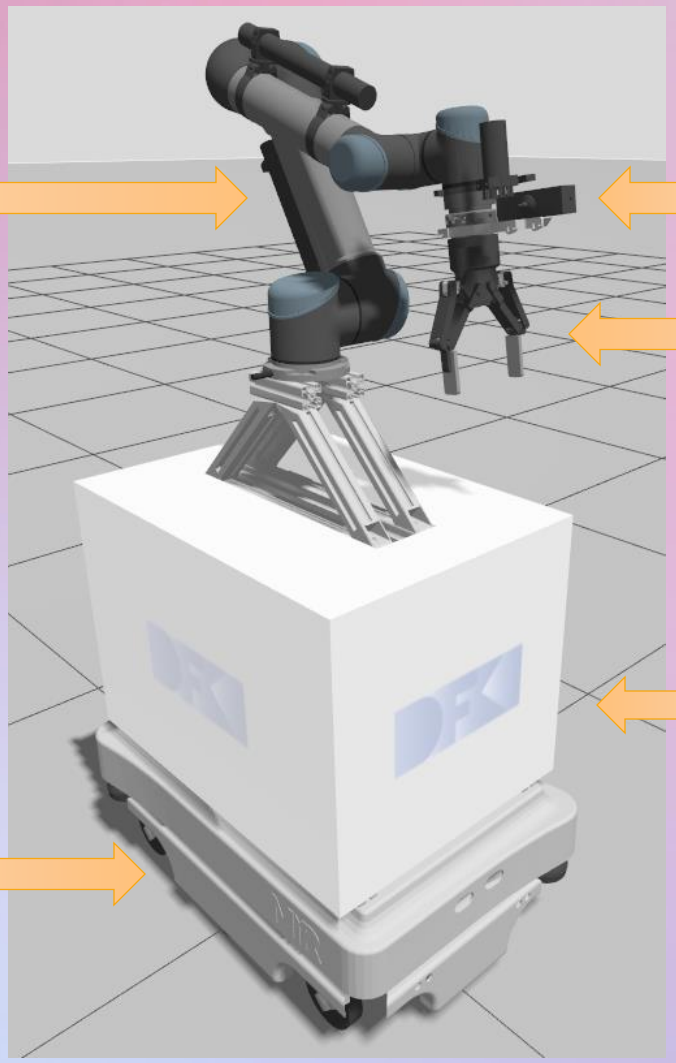


Research questions

- Can RAE+UPOM work on a real robot?
  - shared-model feasibility - eliminating the traditional descriptive/operational model gap
  - robustness under real-world uncertainty
  - online utility-maximising decision-making
- In any case: what is missing?
- Empirical insights into interleaved acting-and-planning

Introduction: Robot overview

6 dof arm

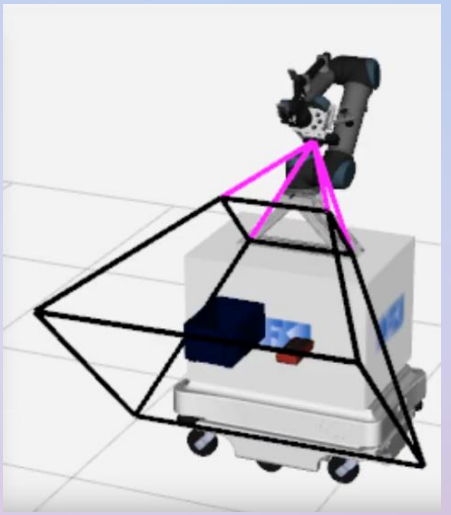


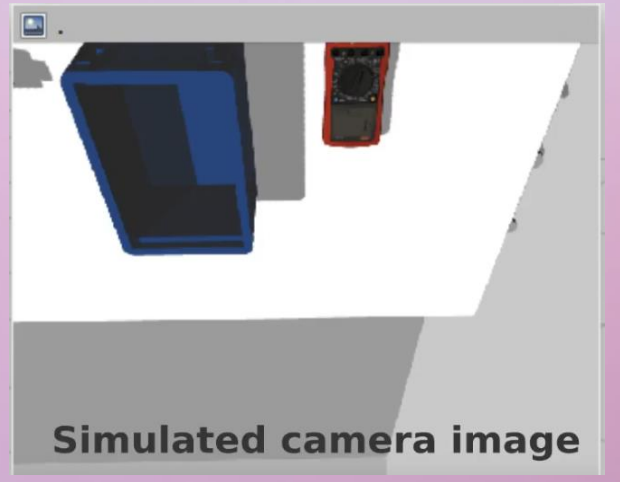
Camera attached to end-effector

2 finger pinch gripper

Mobipick robot

Diff drive Mobile base

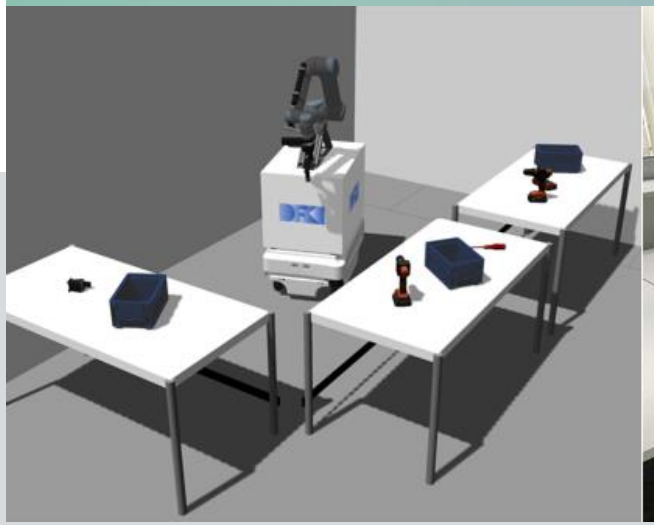




Previous work: Mobipick labs

- Mobipick labs
  - https://github.com/DFKI-NI/mobipick_labs
- Real and sim robot (ROS1 noetic)
- Manipulation (Grasplan)
  - Pick
  - Place
  - Insert
  - Collision checking
- Symbolic Fact Generator
- High-level robot api
- Simulation shares code with the real robot

video



Simulation

Real

A Physics-Based Simulated Robotics Testbed for Planning and Acting Research

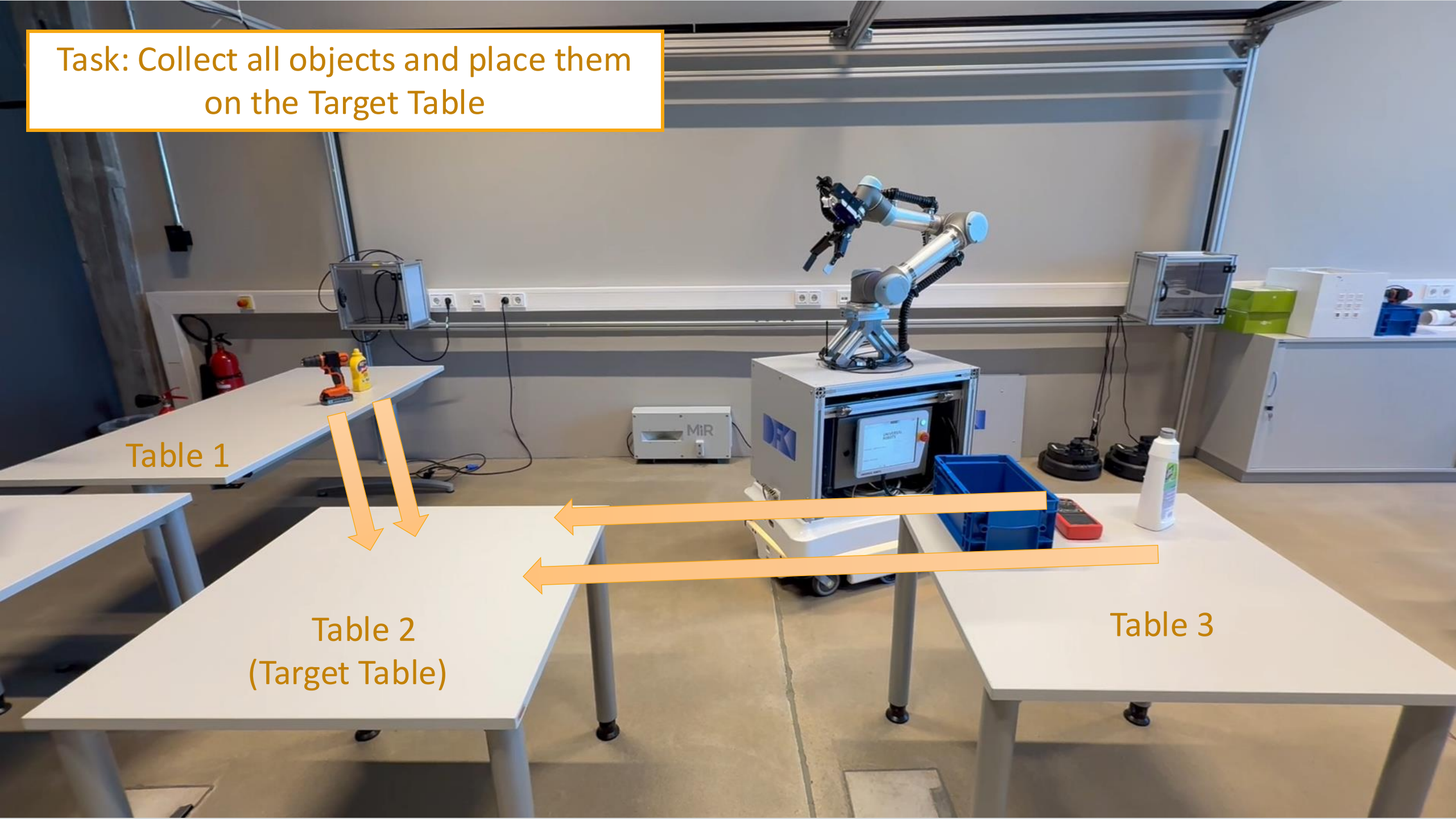
Oscar Lima,[*1] Martin Günther,[*1] Alexander Sung,[1] Sebastian Stock,[1]
Marc Vinci,[1] Amos Smith,[1] Jan Christoph Krause,[1] Joachim Hertzberg[1,2]
[1]German Research Center for Artificial Intelligence (DFKI)
[2]Osnabrück University
{oscar.lima, martin.guenther, alexander.sung, sebastian.stock,
marc.vinci, amos.smith, jan_christoph.krause, joachim.hertzberg}@dfki.de

Abstract

In this paper we introduce the *mobipick_labs* environment, which represents an effort to bring the robotics and planning and acting research communities closer together by releasing a physics-based single robot simulator that closely resembles a real-life setup. This tool is designed for non-robotics experts, providing a user-friendly high-level Python API for easily interacting with a complex robotic system. The framework additionally includes a basic semantic and numeric environment representation that provides real-time knowledge in the form of "facts" that can be used to react to the execution status.

the necessary robotics knowledge to grasp the complexity of a robot acting in a real setup. Therefore, planning research sometimes uses idealized ad hoc simulations that abstract away many features of real robot domains. We argue that these simulators are too abstract, leaving a big gap between the output of a planner and execution on a real robot. To help reduce the gap between the planning and robotics community, we release a software contribution called *mobipick_labs*[1] that is suitable for researchers with little to no experience in robotics or ROS who want to work on planning, acting and monitoring problems in robotic domains.

The Mobipick robot is an indoor mobile manipulator (see


Task: Collect all objects and place them on the Target Table
Table 1
Table 2
(Target Table)
Table 3

Domain modeling

NOTE: Similar to HTN

State variables	Description
P_B	Pose of the robot, defined as $P_B = (x, y, \theta)$, where (x, y) are 2D coordinates and θ is the orientation.
P_O	Object poses, where $P_O(o) = (x, y, \theta)$ for each object $o \in O$. Initialized as *unknown* before perception.
P_{sym}	on$(obj, table)$, in(obj, box): Symbolic facts derived from object and robot poses.
C_O	Classification of each object into predefined categories (e.g., tool, box, table).
holding	State of the robot's manipulator—whether it is holding an object and which one. Verified manually as it may be incorrect (e.g., if the object is dropped).
not_visited(tb)	whether table tb is visited or not
collected(tb)	set of collected objects in table tb

Tasks: explore(r), collect_obj(r, *table*), perceive(r, o), drive(r, l), collect_objs_from_table(r), get_object(r, o, *table*), put_object(r, o, *table*), move_object(r, o, *from*, *to*), insert_object(r, *table*, o, *box*), collect_all_objs(r)

Commands (low level actions): move(r, $l1$, $l2$), read_current_pose(r, o, *pan*, *lift*), grasp(r, o, *from_what*), perceive_table(r, *table*, *pan*, *lift*), move_arm_to_pose(r, *arm_pose*), set_arm_joints(r, *pan*, *lift*), place(r, o, *on_what*), store_object(r, *table*, o, *box*)

collect_objs_from_table$(r, table)$

task: collect_objs_from_table(r)

body: if $P_B(r) \neq$ base$(table)$: drive$(r,$ base$(table))$

while time_passed $\leq$ *LIMIT*:

objects_on_table $\leftarrow \{o \mid$ on(o)=$table$, $C_O(o)$≠box$\}$

if objects_on_table $\neq \emptyset$:

collect_obj$(r,$ *table*$)$

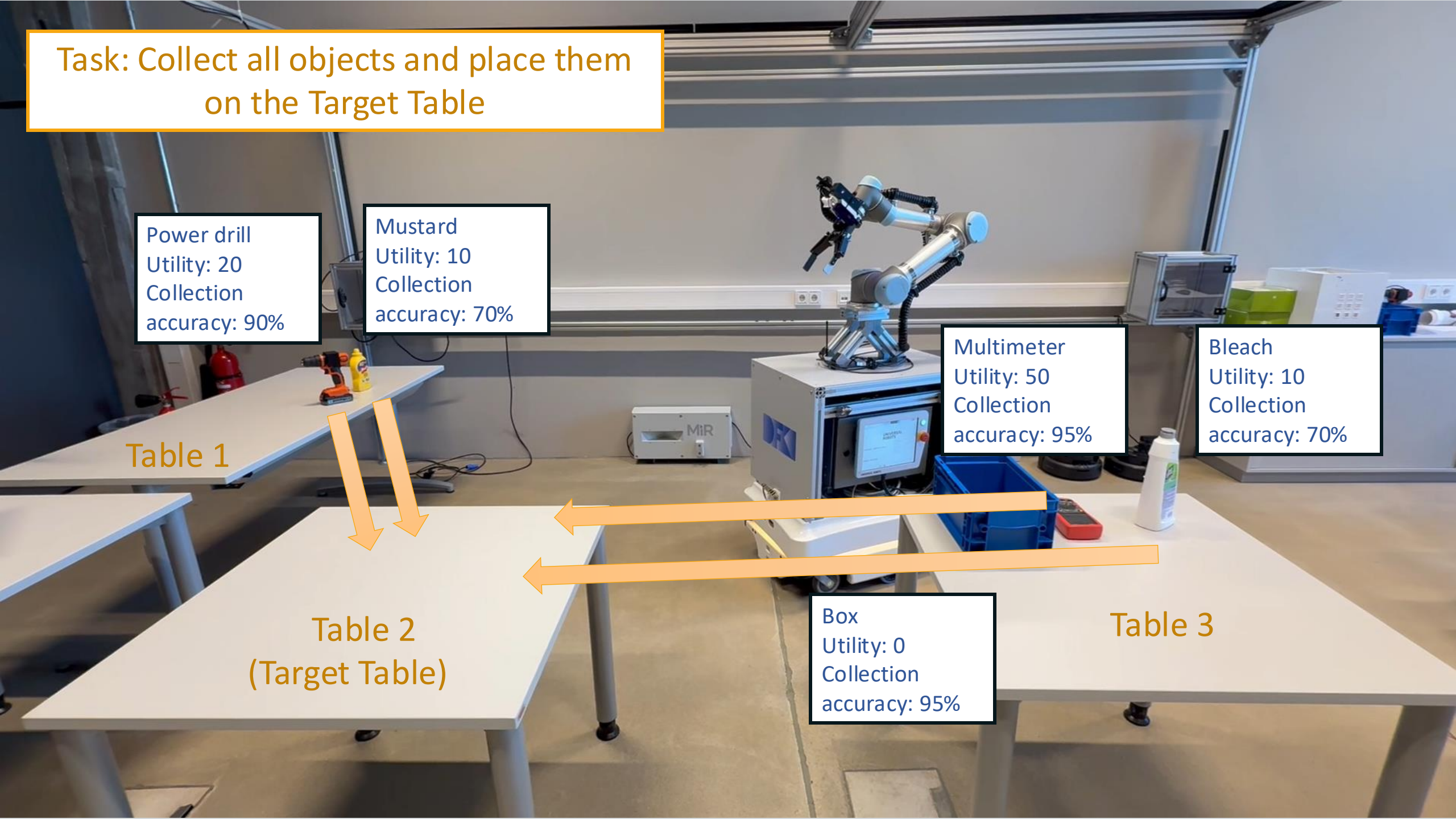
drive$(r,$ base$(table))$

else: break

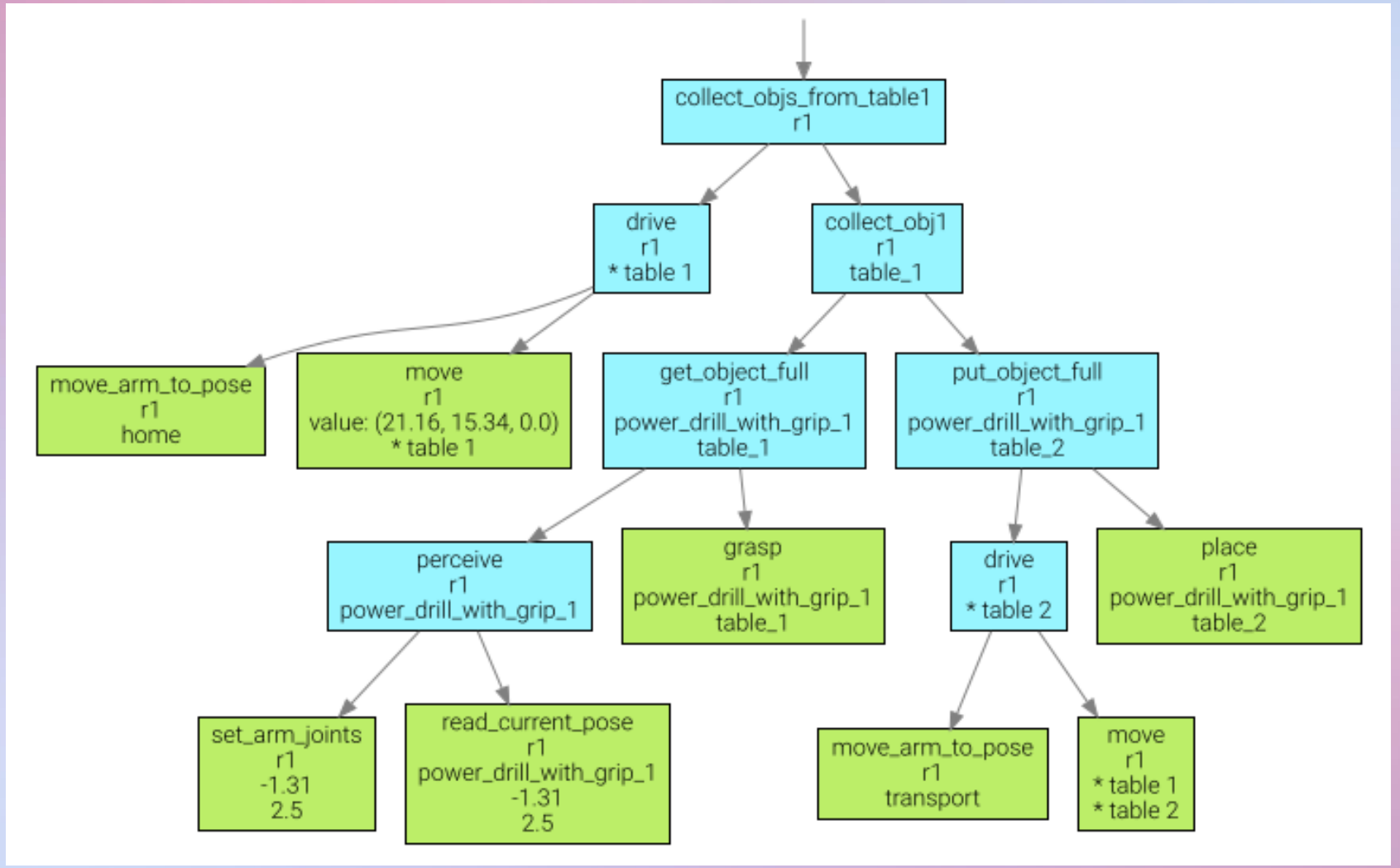
$b \leftarrow \{b|$on$(b,$ *table*$) \wedge C_O(b) =$ box $\wedge$ in$(obj', b)\}$

if $b \neq$ None : move_object$(r,$ $b,$ *table*, *target_table*$)$

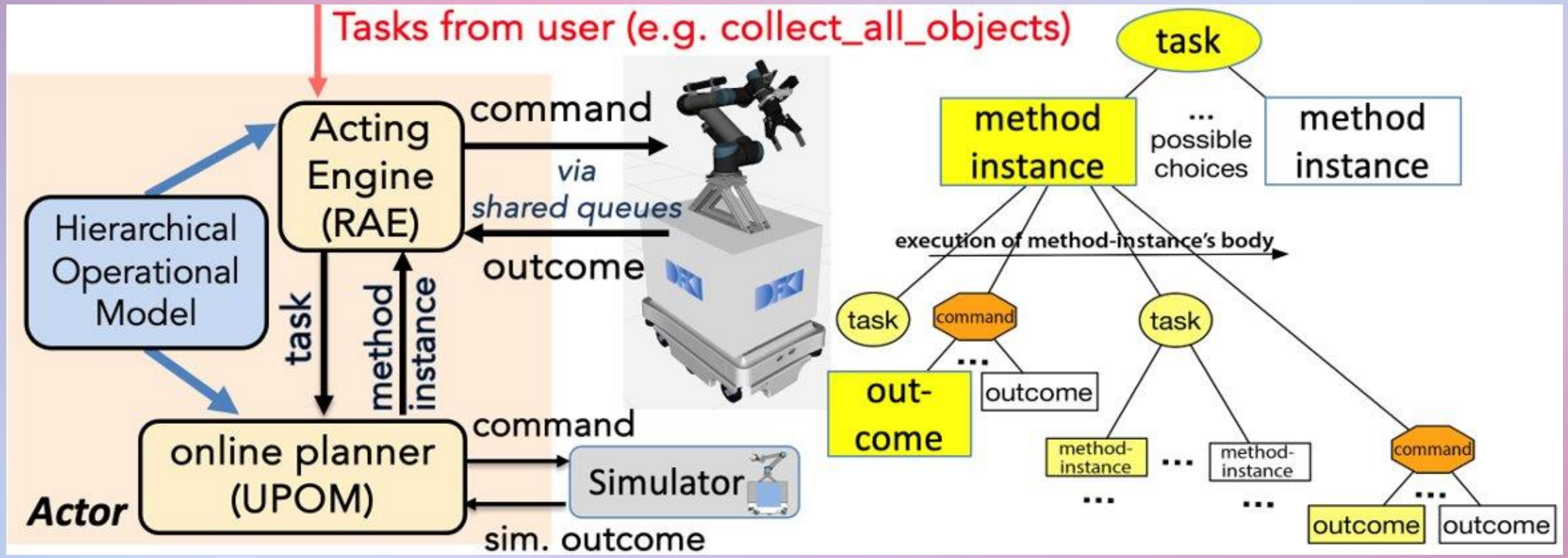
params: $\{table \in T|$not_visited$(table) \wedge table \neq target_table\}$


Task: Collect all objects and place them on the Target Table
Power drill
Utility: 20
Collection accuracy: 90%
Mustard
Utility: 10
Collection accuracy: 70%
Multimeter
Utility: 50
Collection accuracy: 95%
Bleach
Utility: 10
Collection accuracy: 70%
Box
Utility: 0
Collection accuracy: 95%
Table 1
Table 2
(Target Table)
Table 3

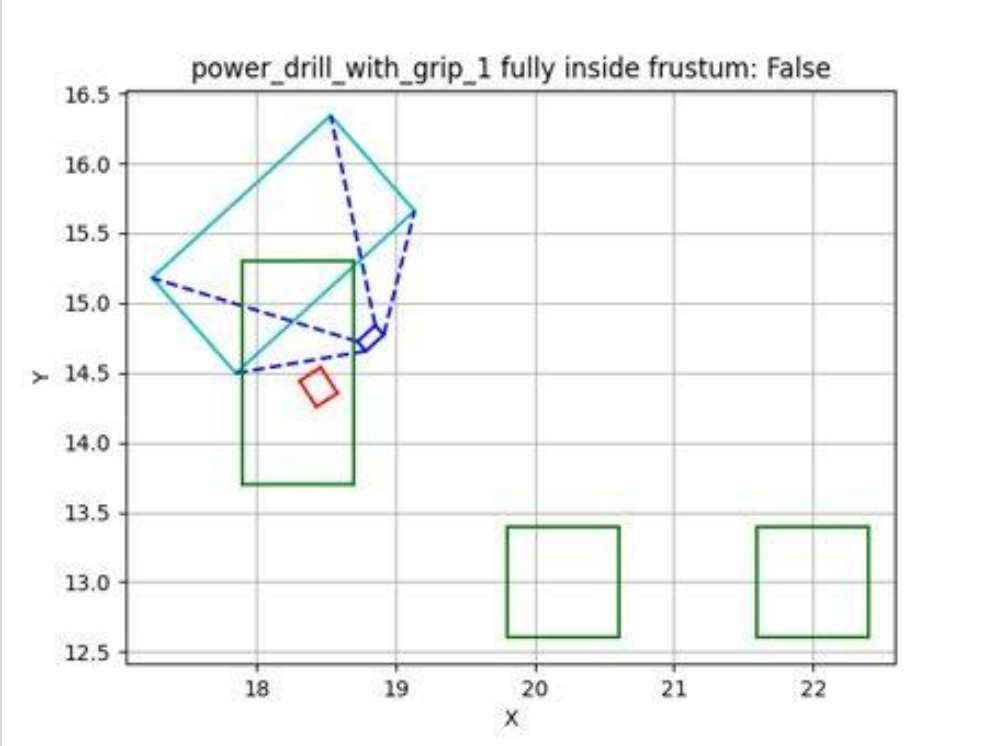
Refinement tree example

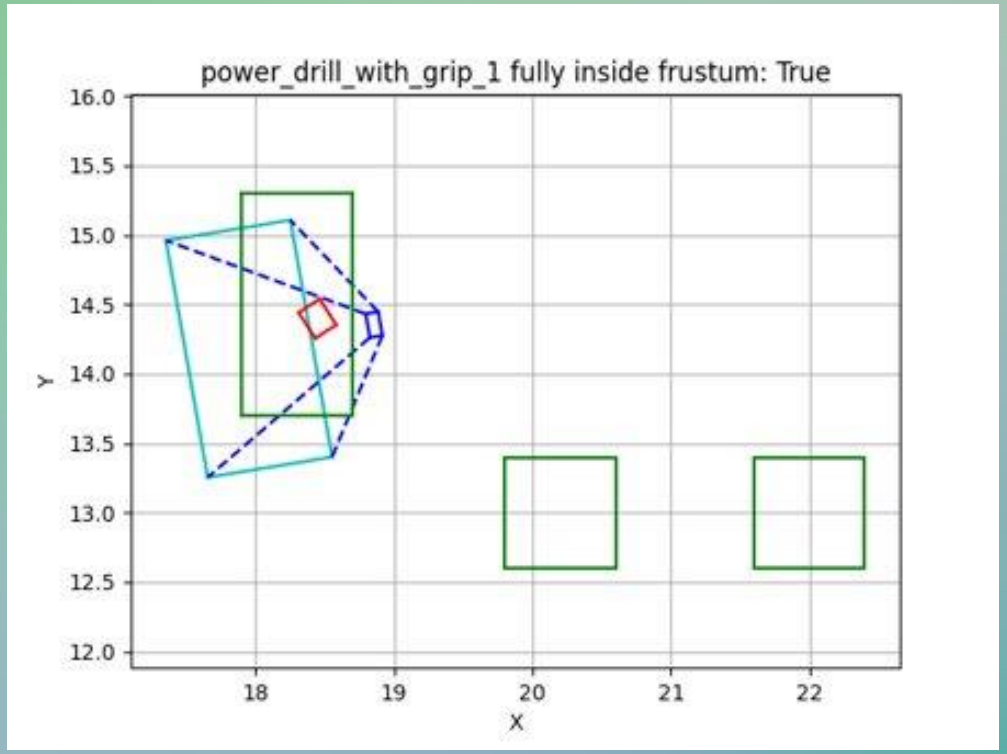


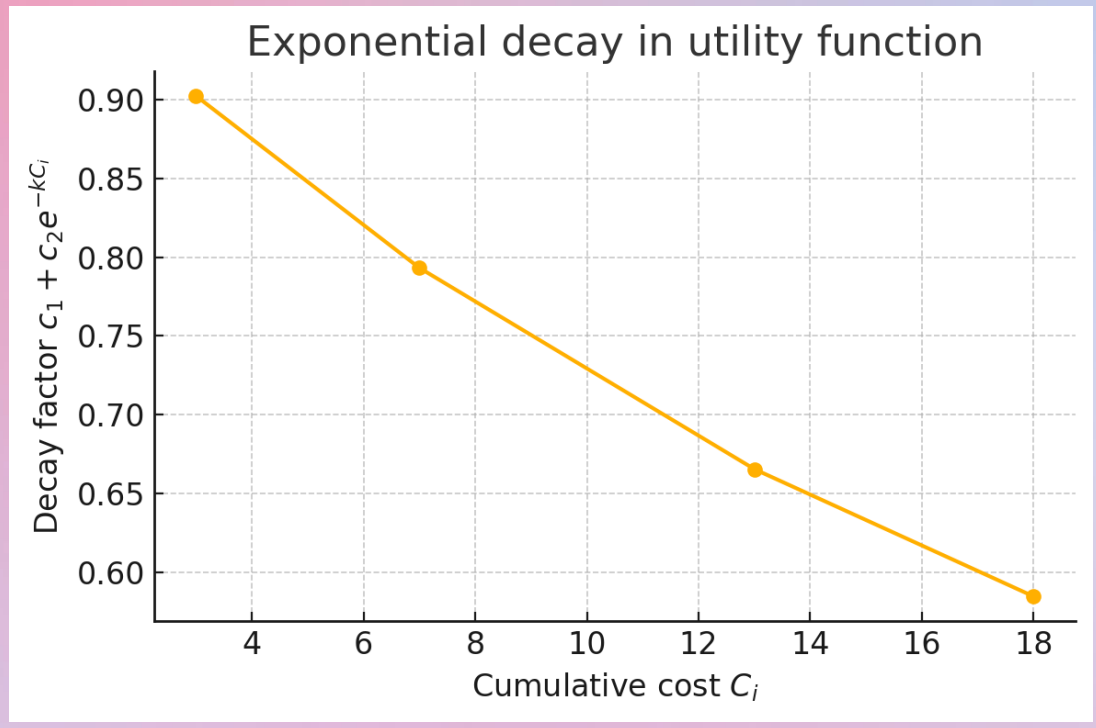
System Architecture



Object within frustum visualisation







Utility function

$$U = \sum_{i=1}^{n} R(\text{collected\_at}(s_i)) \cdot \left(c_1 + c_2 \cdot e^{-k \cdot C_i}\right), \qquad (1)$$

Experimental evaluation

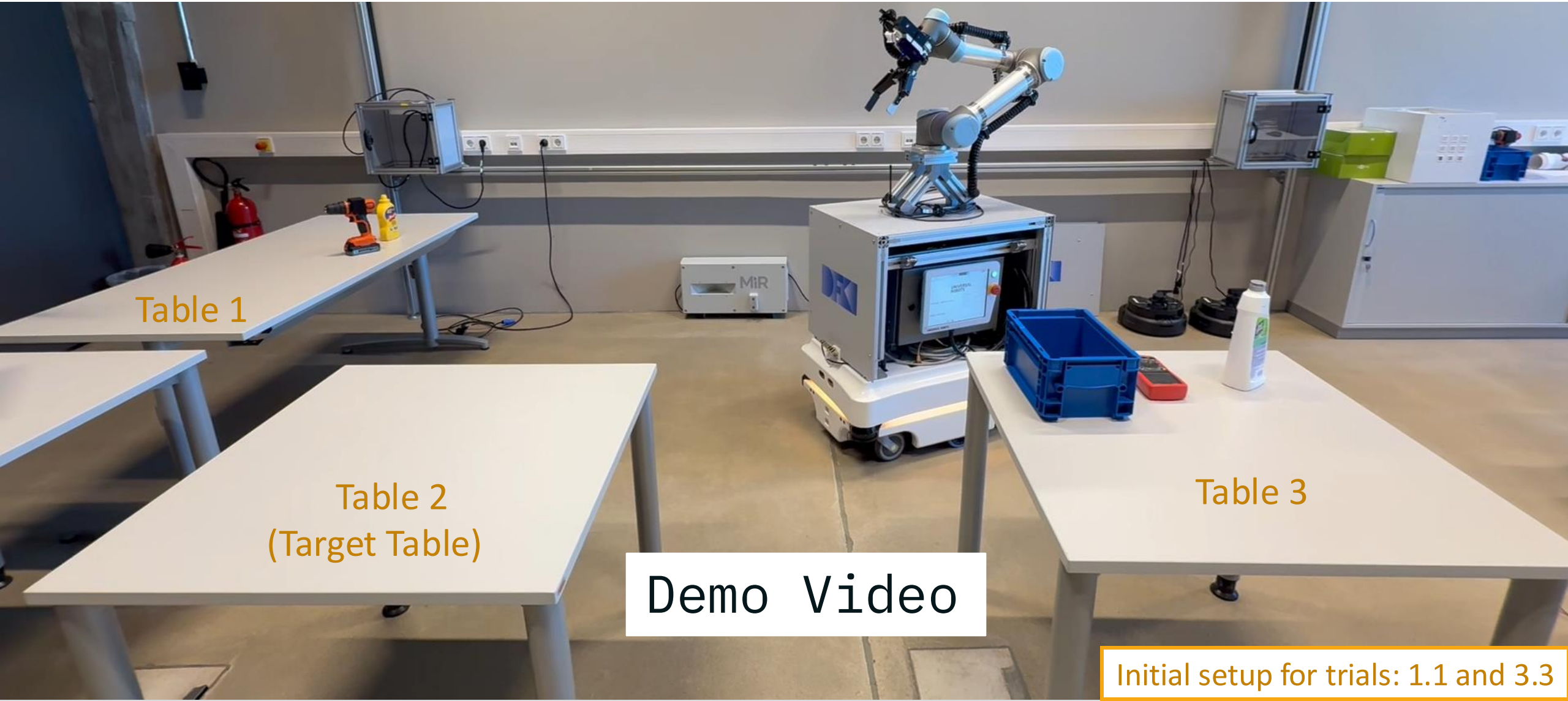
- 6 Experiments
- Divided in 3 categories:
  - Task completion with high utility
  - Increased object spacing
  - Execution failure
    - Cable entanglement
    - Navigation
    - Perception

Trial Description	Collected Utility	Coll. Objs	Planning Time	Acting Time
1.1 Box used	**47.45**	4/4	42.18s	500s
1.2 Box unavailable	32.09	4/4	40.74s	549s
2.1 Spread out objects	27.69	2/4	21.55s	359s
3.1 Arm failed	27.49	3/4	N/A	396s
3.2 Navigation failed	35.83	3/4	28.77s	453s
3.3 Perception failed	32.73	3/4	28.59s	425s

TABLE II

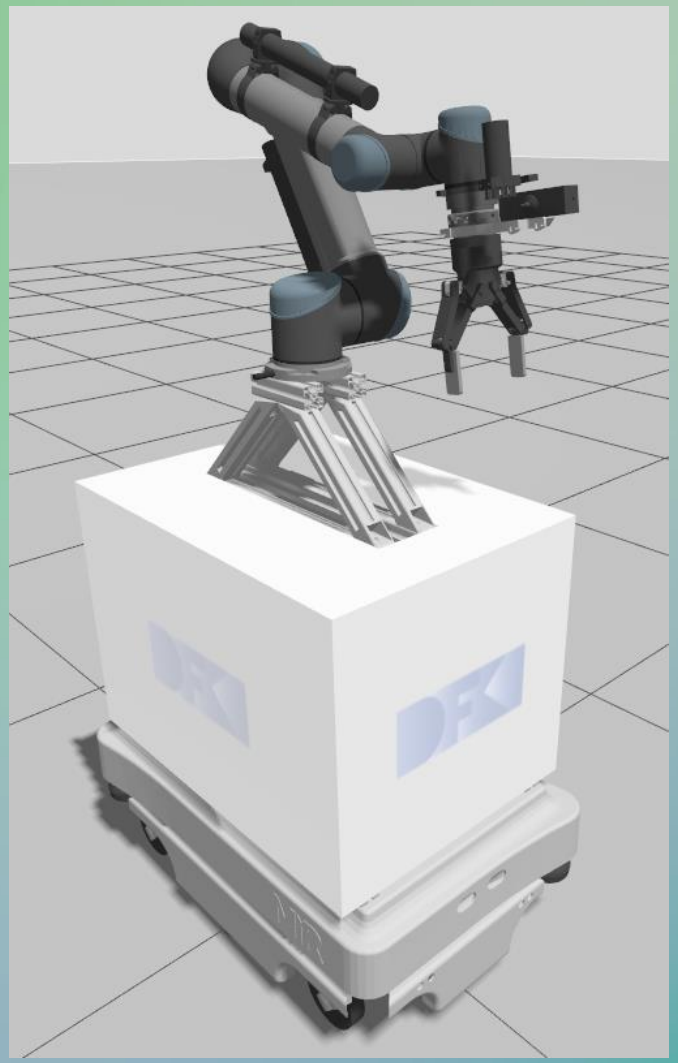
TRIAL RESULTS: UTILITY, COLLECTED OBJECTS, TIMES.

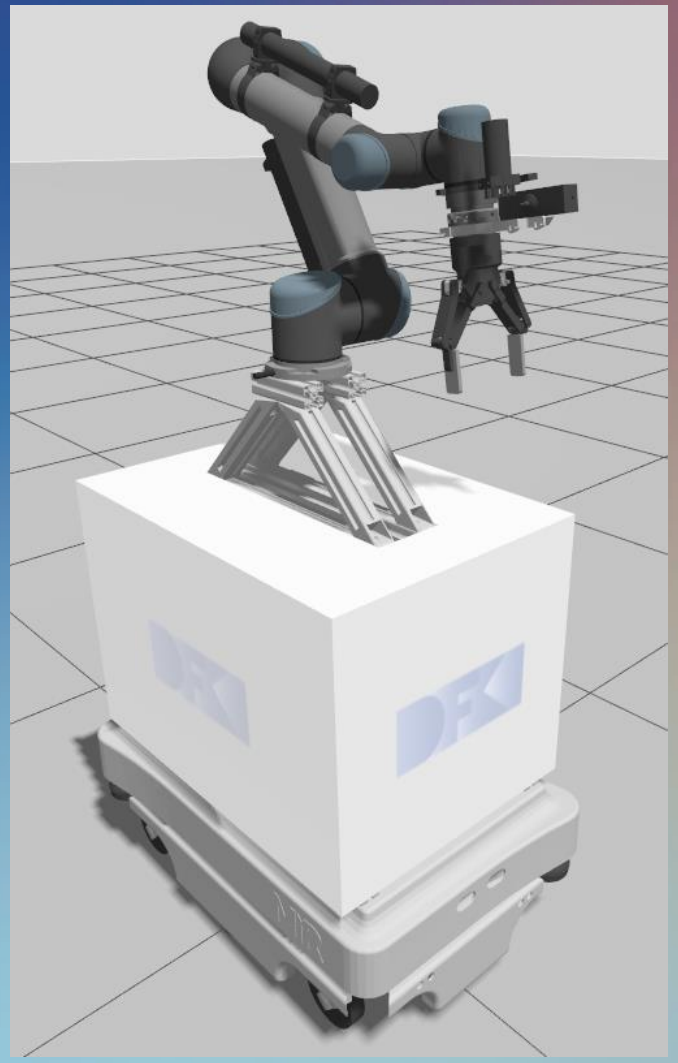
Trial	Multimeter			Powerdrill			Bleach			Mustard			Box			Accum.	No. Coll.	Acting	Planning
	Rwd	Coll. Acc.	Init. Loc.	Rwd	Coll. Acc.	Init. Loc.	Rwd	Coll. Acc.	Init. Loc.	Rwd	Coll. Acc.	Init. Loc.	Rwd	Coll. Acc.	Init. Loc.	reward	objs	time	time
Reward	50			20			10			10			0						
Collection Accuracy		95%			90%			70%			70%			95%					
1.1 Box used			Table 3			Table 1			Table 3			Table 1			Table 3	47.45	4	500s	42.18s
1.2 Box unavailable			Table 3			Table 1			Table 3			Table 1			None	32.09	4	549s	40.74s
2.1 Spread out objects			Table 3			Table 1			Table 3			Table 1			Table 3	27.69	2	359s	21.55s
3.1 Arm failed			Table 3			Table 1			Table 1			Table 3			Table 3	27.49	3	396s	–
3.2 Navigation failed			Table 3			Table 1			Table 1			Table 3			Table 3	35.83	3	453s	28.77s
3.3 Perception failed			Table 3			Table 1			Table 3			Table 1			Table 3	32.73	3	425s	28.59s



Contributions

- Adaptation and extension of the RAE+UPOM engine for integration with a physical robotic platform, thus enabling robust operation under real-world execution constraints.
- Hierarchical operational model for object collection, shared between actor and planner. It enables the robot to perceive and collect objects effectively.
- Experiments on a real robot in a controlled hardware lab environment, providing insights into the planner’s internal decision-making mechanisms.





Future work

- Dead end handling
- More dynamic events and failures
- Better informed rollouts (e.g., Physics-based)
- Heuristics
- Learning (refinement methods)

The END!

dfki ai Deutsches Forschungszentrum für Künstliche Intelligenz *German Research Center for Artificial Intelligence*

UNIVERSITÄT OSNABRÜCK

UNIVERSITY OF MARYLAND

FONDAZIONE BRUNO KESSLER

LAAS CNRS

SPONSORED BY THE

Federal Ministry of Education and Research