

Workshop

Teaching & Training Students
Cognitive Robots in the Cloud

Arthur Niedźwiecki, Yanxiang Zhan
October 01st, 2023



OCTOBER 1 - 5, 2023

IEEE/RSJ International Conference on
Intelligent Robots and Systems



Overview

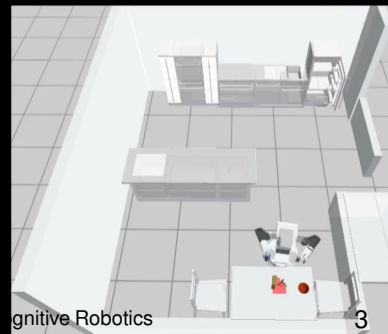
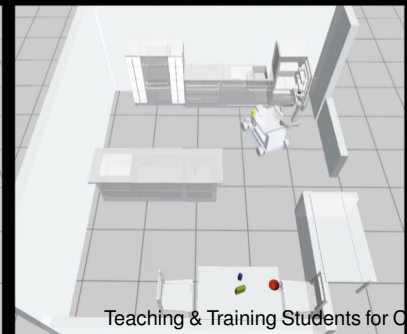
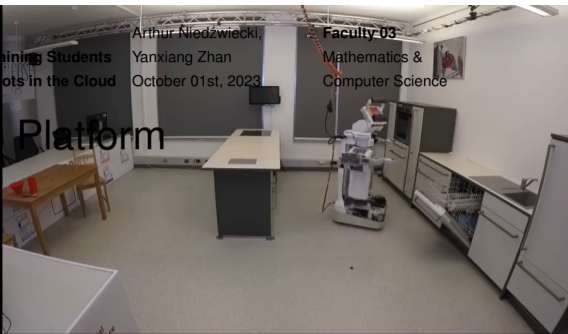
- 1 Getting Started
- 2 Virtualization
- 3 Into the cloud
- 4 Furthermore

Getting Started - Online Learning Platform

Workshop
Teaching & Training Students
Cognitive Robots in the Cloud

Arthur Niedźwiecki,
Yanxiang Zhan
October 01st, 2023

Faculty 03
Mathematics &
Computer Science

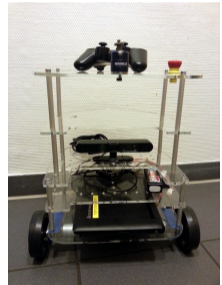


Getting Started

How do I introduce somebody to robotics software?

Getting Started - Lecture on Robot Programming

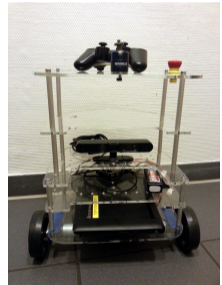
Participation



Getting Started - Lecture on Robot Programming

Participation

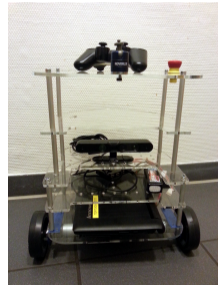
- First lecture attended by 90% of signed-up students



Getting Started - Lecture on Robot Programming

Participation

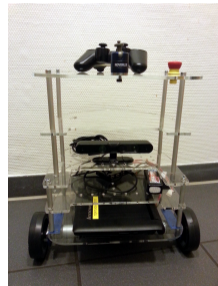
- First lecture attended by 90% of signed-up students
- Assignments submitted by 50% of first lecture attendees



Getting Started - Lecture on Robot Programming

Participation

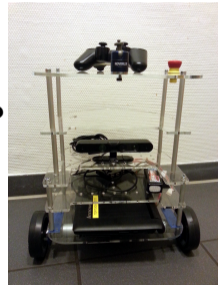
- First lecture attended by 90% of signed-up students
- Assignments submitted by 50% of first lecture attendees
- Exams taken by 90% of submitters



Getting Started - Lecture on Robot Programming

Participation

- First lecture attended by 90% of signed-up students
- **Assignments submitted by 50% of first lecture attendees??**
- Exams taken by 90% of submitters



Getting Started - Lecture on Robot Programming

First assignment:



Getting Started - Lecture on Robot Programming

First assignment:

- Install Linux



Getting Started - Lecture on Robot Programming

First assignment:

- Install Linux
- Set up SSH and GitHub



Getting Started - Lecture on Robot Programming

First assignment:

- Install Linux
- Set up SSH and GitHub
- Install Robot Operating System (ROS)



Getting Started - Lecture on Robot Programming

First assignment:

- Install Linux
- Set up SSH and GitHub
- Install Robot Operating System (ROS)
- Install a few packages



Getting Started - Lecture on Robot Programming

First assignment:

- Install Linux
- Set up SSH and GitHub
- Install Robot Operating System (ROS)
- Install a few packages
- Modify a Lisp file with Emacs



Getting Started - Setup is frustrating

- Requires specific operating system



Getting Started - Setup is frustrating

- Requires specific operating system
- Collides with existing software



Getting Started - Setup is frustrating

- Requires specific operating system
- Collides with existing software
- Complex and fragile setup takes time



Getting Started - Setup is frustrating

- Requires specific operating system
- Collides with existing software
- Complex and fragile setup takes time
- Documentation has low priority



Virtualization

- 1 Getting Started
- 2 Virtualization**
- 3 Into the cloud
- 4 Furthermore

Virtualization

How can I make my platform easier accessible?

Virtualization - Docker Setup



Welcome to the demo scenario of euROBIN. You will learn how to write a sequence of tasks to make a simulated robot

- open a door
- carry a package
- pick an item out of that package onto the table.

We will work with the TIAGO robot in the IAI Bremen apartment laboratory. Open a new tab in Jupyter Notebook, then open a Terminal. Execute the following launchfile in that terminal to launch the environment.

```
roslaunch cram_projection_demos apartment_tiago.launch
```

Load the CRAM system for our robot demos. Hit CTRL-Enter in the code-blocks to execute them. Wait until the `[*]` symbol turns into a number. Hide the output by hitting the blue vertical bar to the left.

```
[ ]: (asdf:load-system :cram-projection-demos)
```

Initialize the simulation.

```
[ ]: (roslisp-utilities:startup-ros)
```

The robot appears at 0,0 by default. Position it in the apartment.

```
[ ]: (btr-utils:move-robot '((9 3 0) (0 0 0 1)))
```

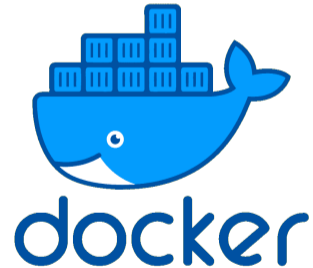
```
[ ]: (urdf-proj:with-projected-robot
      (demos::eurobin-demo))
```

Would you like to receive official Jupyter news?
Please read the privacy policy.

[Open privacy policy](#) Yes No

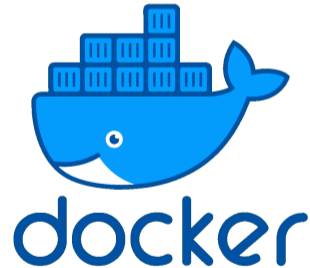
Virtualization - Docker

- Virtual Machine



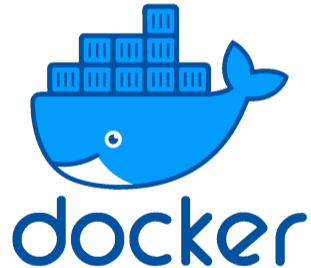
Virtualization - Docker

- Virtual Machine
- Prepare everything needed



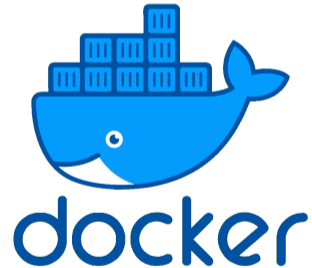
Virtualization - Docker

- Virtual Machine
- Prepare everything needed
- Preconfigure program execution



Virtualization - Docker

- Virtual Machine
- Prepare everything needed
- Preconfigure program execution
- 'Simply' download and execute

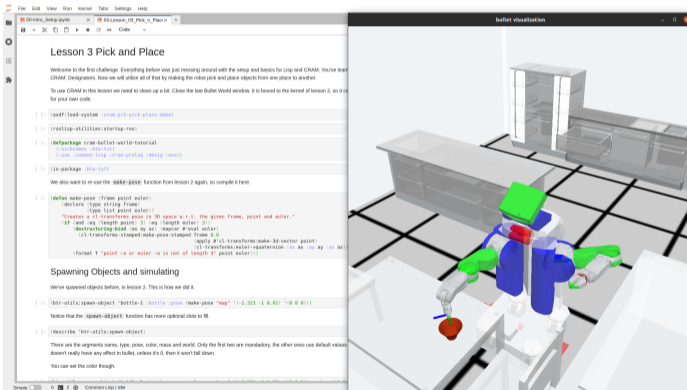


Virtualization - Docker

Install everything in a build-script



Virtualization - Docker



The image shows a terminal window on the left and a 3D visualization window on the right. The terminal window displays the following content:

```

Lesson 3 Pick and Place

Welcome to the first challenge. Everything before was just messing around with the setup and basics for Lisp and CRAM. You've learnt CRAM Designators. Now we will utilize all of that by making the robot pick and place objects from one place to another.

To use CRAM in this lesson we need to clean up a bit. Close the last Bullet World window. It is bound to the kernel of lesson 2, so it can't be used for your own code.

1 | | sudo! :load-system :cram-pr2-pick-place-demo!
2 | | ros2-lisp-utilities:startup-ros
3 | | defpackage :cram-bullet-world-tutorial
   | | :use (:common-lisp :cram-pr2-lisp :design :swd)
4 | | | :in-package :lisp-tut

We also want to re-use the :make-pose function from lesson 2 again, so complete it here:

5 | | (defun make-pose (frame point euler)
   | |   (declare (type string frame)
   | |           (type list point euler))
   | |   "Creates a ci-transform pose in 3D space w.r.t. the given frame, point and euler."
   | |   (let ((ax (length point)) (ay (length euler)) (az (length euler)))
   | |     (destructuring-bind ((x y az) (mapcar #'eval euler))
   | |       (ci-transform-stamped:make-pose-stamped frame 0.0
   | |         (apply #'ci-transforms:make-3d-vector point)
   | |         (ci-transforms:vector-quaternion (ax as (y ay) (az az)))))
   | |   (format T "point => a or euler => is not of length ~A" point euler))

Spawning Objects and simulating

We've spawned objects before, in lesson 2. This is how we did it.

6 | | | (btr-utlts:spawn-object 'bottle-1 :bottle :pose (make-pose "map" '(-2.323 -1.0 0.82) '(0 0 0)))

Notice that the :spawn-object function has more optional slots to fill.

7 | | | (describe 'btr-utlts:spawn-object)

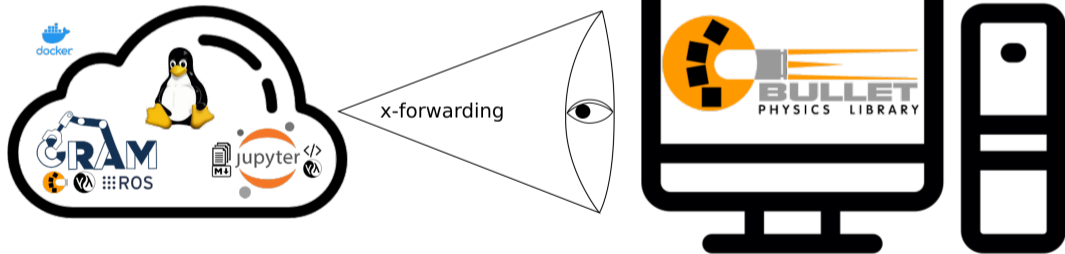
There are the arguments name, type, pose, color, mass and world. Only the first two are mandatory, the other ones use default values and don't really have any effect in bullet, unless it's 0, then it won't fall down.

You can set the color through.

```

The 3D visualization window shows a blue and white robot arm in a kitchen environment, holding a red object. The environment includes a kitchen counter, a sink, and a stove.

Virtualization - Docker

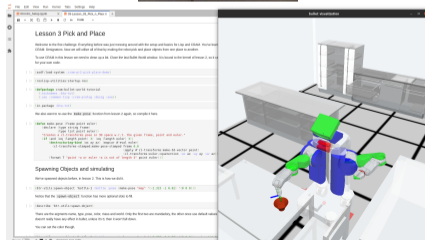
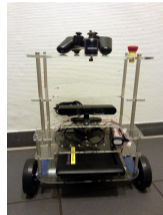


Virtualization - Docker

Benefits:

- Integrated with local ROS network

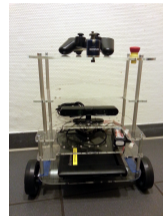
Limitations:



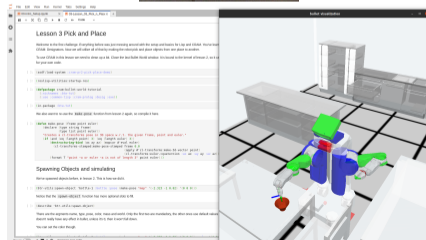
Virtualization - Docker

Benefits:

- Integrated with local ROS network
- Control real-world robot from Jupyter



Limitations:

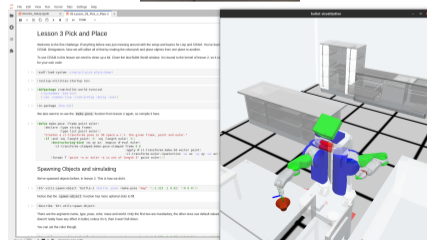
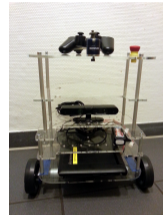


Virtualization - Docker

Benefits:

- Integrated with local ROS network
- Control real-world robot from Jupyter
- Documentation, code and simulation

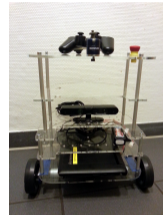
Limitations:



Virtualization - Docker

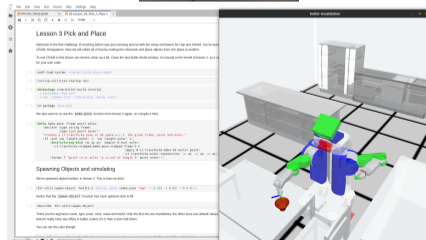
Benefits:

- Integrated with local ROS network
- Control real-world robot from Jupyter
- Documentation, code and simulation



Limitations:

- Docker is not entirely platform independent



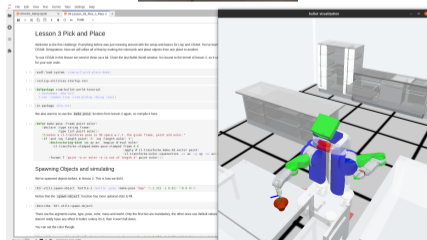
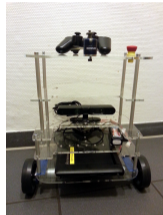
Virtualization - Docker

Benefits:

- Integrated with local ROS network
- Control real-world robot from Jupyter
- Documentation, code and simulation

Limitations:

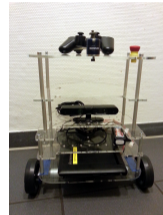
- Docker is not entirely platform independent
- Exhausts client machine resources



Virtualization - Docker

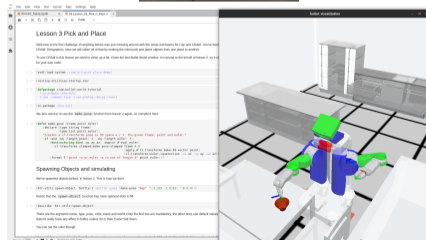
Benefits:

- Integrated with local ROS network
- Control real-world robot from Jupyter
- Documentation, code and simulation



Limitations:

- Docker is not entirely platform independent
- Exhausts client machine resources
- Still too much setup overhead



Into the cloud

- 1 Getting Started
- 2 Virtualization
- 3 Into the cloud**
- 4 Furthermore

Into the cloud

How can I give a lecture including complex software the most platform-independent?

cram2/cram_teaching: L... x eurobin-demo... - Jupyter: x +

Not secure | 172.17.0.1:8888/lab/tree/lectures/demos/eurobin-demo.ipynb

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ lectures / demos /


Name

- common-lisp-jupyter
- eurobin-demo.ipynb**
- pick_and_place_loop.ipynb
- popcorn_demo.ipynb
- task_tree_export.dot

Launcher x eurobin-demo.ipynb x +

Markdown

Common Lisp



Welcome to the demo scenario of euROBIN. You will learn how to write a sequence of tasks to make a simulated robot

- open a door
- carry a package
- pick an item out of that package onto the table.

We will work with the TIAGO robot in the IAI Bremen apartment laboratory. Open a new tab in Jupyter Notebook, then open a Terminal. Execute the following launchfile in that terminal to launch the environment.

```
roslaunch cram_projection_demos apartment_tiago.launch
```

Load the CRAM system for our robot demos. Hit CTRL-Enter in the code-blocks to execute them. Wait until the `[*]` symbol turns into a number. Hide the output by hitting the blue vertical bar to the left.

```
[ ]: (asdf:load-system :cram-projection-demos)
```

Initialize the simulation.

```
[ ]: (roslisp-utilities:startup-ros)
```

The robot appears at 0,0 by default. Position it in the apartment.

```
[ ]: (btr-utils:move-robot '((9 3 0) (0 0 1)))
```

```
[ ]: (urdf-proj:with-projected-robot
      (demos::eurobin-demo))
```

Would you like to receive official Jupyter news? Please read the privacy policy.

[Open privacy policy](#) Yes No

IROS 2023 Workshop Teaching & Training Students for Cognitive Robotics

Simple 0 1 2 Common Lisp mode Mode: Command Ln 1, Col 1 eurobin-demo.ipynb 1

Into the cloud - Server-side Architecture

Applications



GISKARD



KnowRob

RoboKudo 

Into the cloud - Server-side Architecture

Packaging



GISKARD



KnowRob

RoboKudo[®]

Into the cloud - Server-side Architecture

Orchestrator



kubernetes



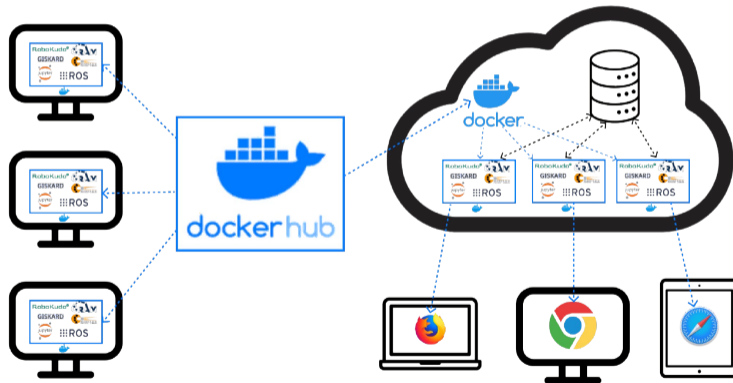
GISKARD



KnowRob

RoboKudo[®]

Into the cloud - System Overview

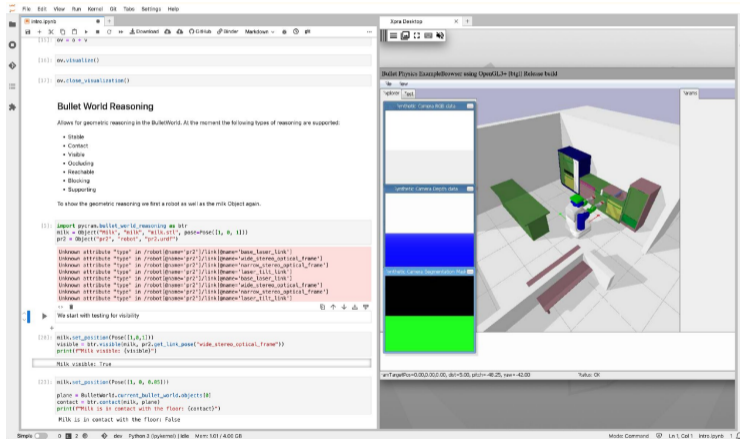


Into the cloud - Robot Web Tools

- Rosbridge, roslibjs, ros3djs, rosbagjs
- Rvizweb, Rosboard, Webviz



Into the cloud - XPRa remote desktop



The screenshot displays a remote desktop environment. On the left, a terminal window shows the execution of a Python script that uses the Bullet physics engine for geometric reasoning. The script defines a robot and a table, then checks for visibility and contact between them. The output shows that the robot can see the table and is in contact with the floor.

```

1171: a = b + y

1171: ow.visualize()

1171: ow.close_visualization()

Bullet World Reasoning
Allows for geometric reasoning in the BulletWorld. At the moment the following types of reasoning are supported:
• Stable
• Contact
• Visible
• Occluding
• Reachable
• Blocking
• Supporting
To show the geometric reasoning we first a robot as well as the milk Object again.

1171: import pyrran.bullet_world_reasoning as btr
milk = Object("Milk", "Milk", "Milk.txt", pose=Pose(1, 0, 1))
pr2 = Object("pr2", "robot", "pr2.urdf")

Unknown attribute "type" in /robot[@name="pr2"]/link[@name="base_lower_link"]
Unknown attribute "type" in /robot[@name="pr2"]/link[@name="wide_stereo_optical_frame"]
Unknown attribute "type" in /robot[@name="pr2"]/link[@name="narrow_stereo_optical_frame"]
Unknown attribute "type" in /robot[@name="pr2"]/link[@name="laser_illc_link"]
Unknown attribute "type" in /robot[@name="pr2"]/link[@name="base_upper_link"]
Unknown attribute "type" in /robot[@name="pr2"]/link[@name="wide_stereo_optical_frame"]
Unknown attribute "type" in /robot[@name="pr2"]/link[@name="narrow_stereo_optical_frame"]
Unknown attribute "type" in /robot[@name="pr2"]/link[@name="laser_illc_link"]

--
We start with testing for visibility

1171: milk.set_position(Pose([1,0,1]))
visible = btr.visible(milk, pr2.get_link_name("wide_stereo_optical_frame"))
print("Milk visible: %s" % visible)

Milk visible: True

1171: milk.set_position(Pose(1, 0, 0.85))
plane = BulletWorld().create_bullet_world_objects()
contact = btr.contact(milk, plane)
print("Milk is in contact with the floor: %s" % contact)

Milk is in contact with the floor: False
    
```

On the right, a 3D visualization window titled "Xpra Desktop" shows a simulated environment with a robot (pr2) and a table. The robot is positioned in a room with a floor and walls. The table is a green and red structure. The robot's camera view is visible in the top right corner of the 3D window.

Into the clouds - Summary

Benefits:

- Easy platform-independent access

Limitations:

Into the clouds - Summary

Benefits:

- Easy platform-independent access
- Vastly extendable

Limitations:

Into the clouds - Summary

Benefits:

- Easy platform-independent access
- Vastly extendable
- Easily shared

Limitations:

Into the clouds - Summary

Benefits:

- Easy platform-independent access
- Vastly extendable
- Easily shared

Limitations:

- No UI applications

Into the clouds - Summary

Benefits:

- Easy platform-independent access
- Vastly extendable
- Easily shared

Limitations:

- No UI applications
- High network load

Into the clouds - Summary

Benefits:

- Easy platform-independent access
- Vastly extendable
- Easily shared

Limitations:

- No UI applications
- High network load
- Network security

Furthermore

- 1 Getting Started
- 2 Virtualization
- 3 Into the cloud
- 4 Furthermore**

Furthermore

Besides teaching, what else can be achieved with robots in the cloud?

Stock - Retail

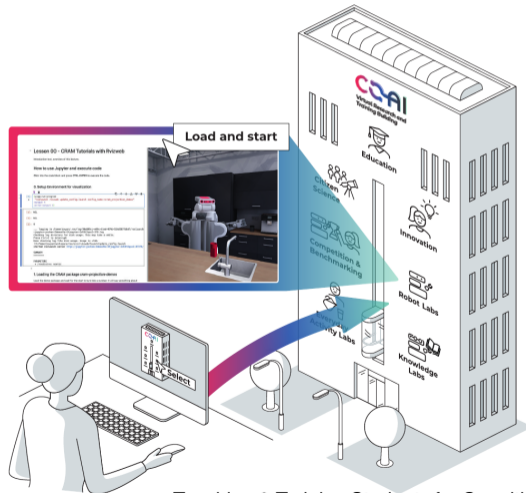
Stock - Bookstore

geschoss - Living



Zielen und Klicken Sie auf den Knopf für das gewünschte Stockwerk.

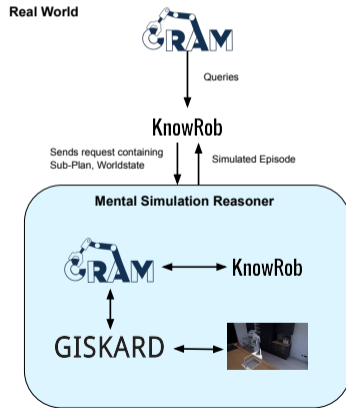
Furthermore - Virtual Research Building



Furthermore - Mental Simulation

Simulate snapshot worlds to determine feasibility for complex manipulation tasks.

- CRAM - Plan Executive
- Giskard - Motion Planner
- Multiverse - Physics Reasoning
- Knowrob - KR & Reasoning



Furthermore - Online Lectures

- Robot Programming with ROS (Lecture)
- EASE FallSchool (Seminar)
- IROS 2023 Conference Tutorials

Furthermore - IROS 2023 Tutorial

Thursday

Tutorial: Everyday Activity Robot Manipulation in an Interactive Learning Environment

Time	Session
8.30 - 8.45	OPENING: Michael Beetz & Jörn Syrbe
8.45 - 10.00	Introduction - Michael Beetz
10.00 - 11.00	COFFEE BREAK
11.00 - 12.30	Hands-on Robot Control in CRAM – Arthur Niedźwiecki
12.30 - 1.30	LUNCH
1.30 - 3.00	Hands-On Robotics Simulation in Multiverse – Giang Nguyen
3.00 - 4.00	COFFEE BREAK
4.00 - 5.30	Hands-On Knowledge openEASE – Sascha Jongbloed
5.30	END

Thank you for your attention!
Enjoy the IROS 2023!



OCTOBER 1 - 5, 2023

IEEE/RSJ International Conference on
Intelligent Robots and Systems