Portable cognition-enabled plan executives

Arthur Niedzwiecki

May 18, 2023

Artificial Intelligence

euROBIN

1. Setup

2. CRAM Architecture

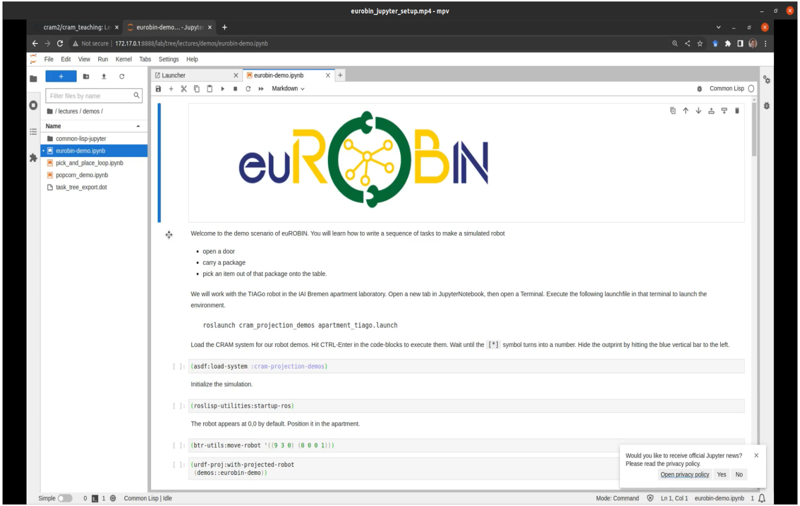
3. Motivation

4. Workshop Technology

5. Hands-On

6. Prospect

Overview - Setup Procedure



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 JupyterNotebook, 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 outprint 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)))
```

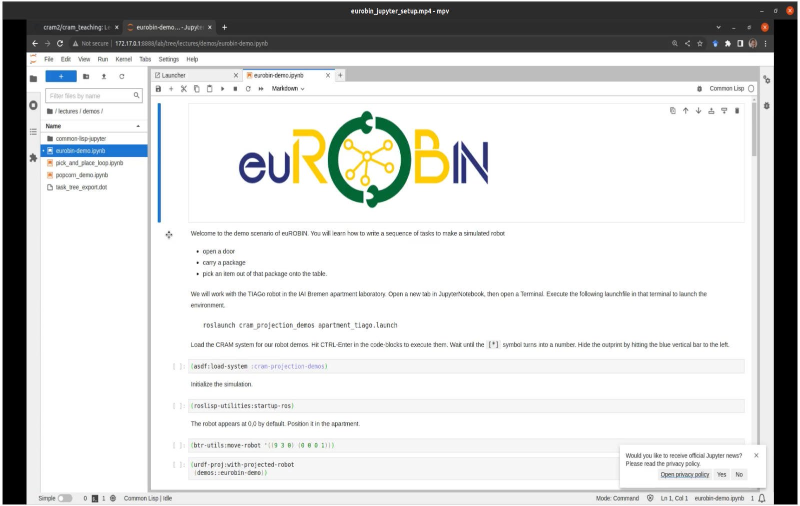
```
[ ]: (urdf-proj:with-projected-robot
      (demos::eurobin-demo))
```

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

Open privacy policy | Yes | No

http://cram-system.org > Installation

Overview - Setup Procedure



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 JupyterNotebook, 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 outprint 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)))
```

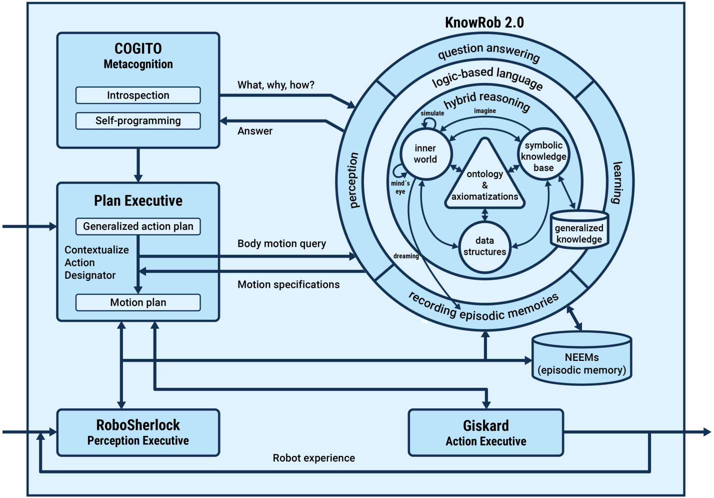
```
[ ]: (urdf-proj:with-projected-robot
      (demos::eurobin-demo))
```

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

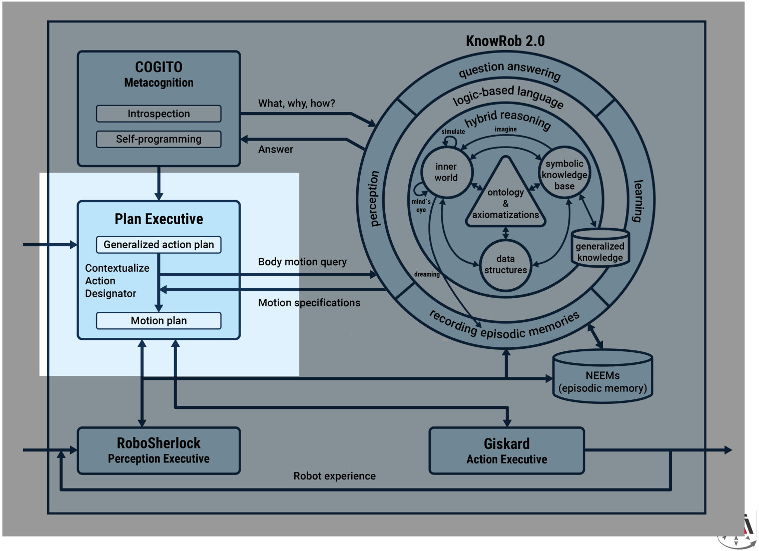
Open privacy policy Yes No

http://cram-system.org > Installation

CRAM Architecture

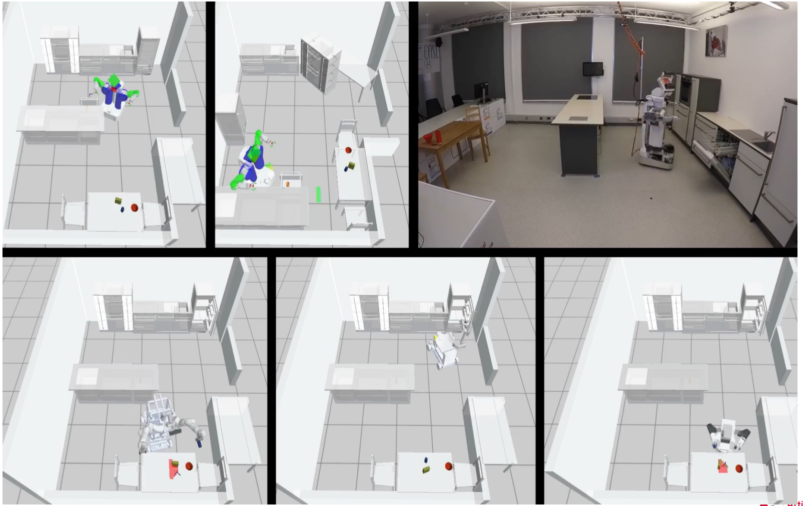


CRAM Architecture - Plan Executive



Motivation

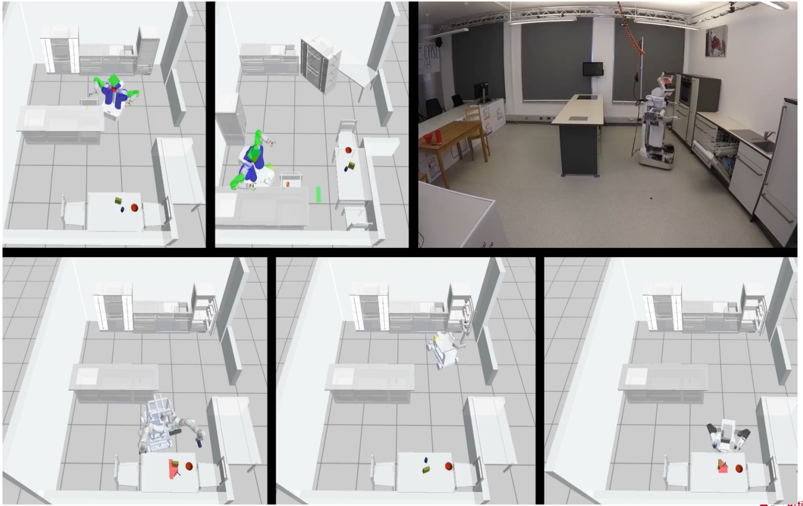
Motivation - Platform-Independent Control Program



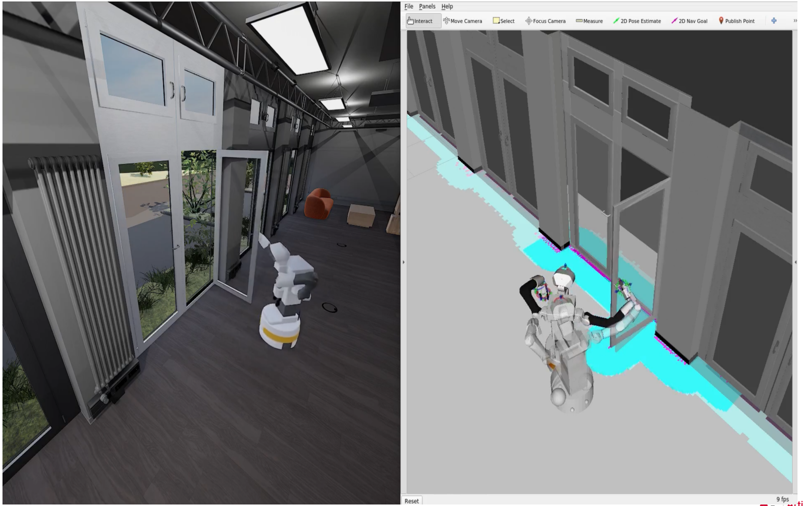
Artificial Intelligence

euROBIN

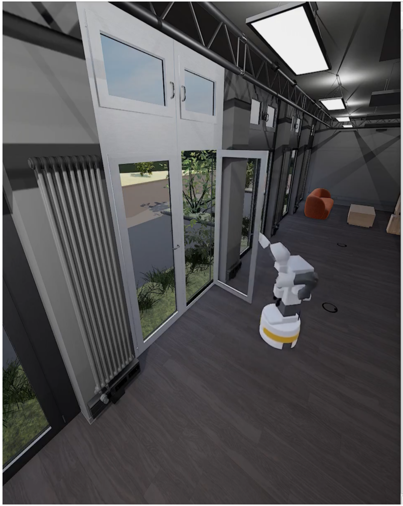
Motivation - Platform-Independent Control Program

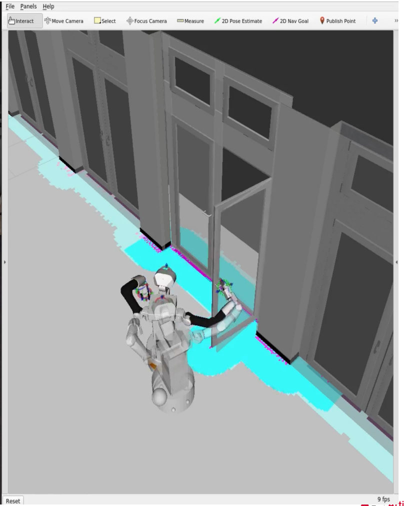


Motivation - Robot Integration

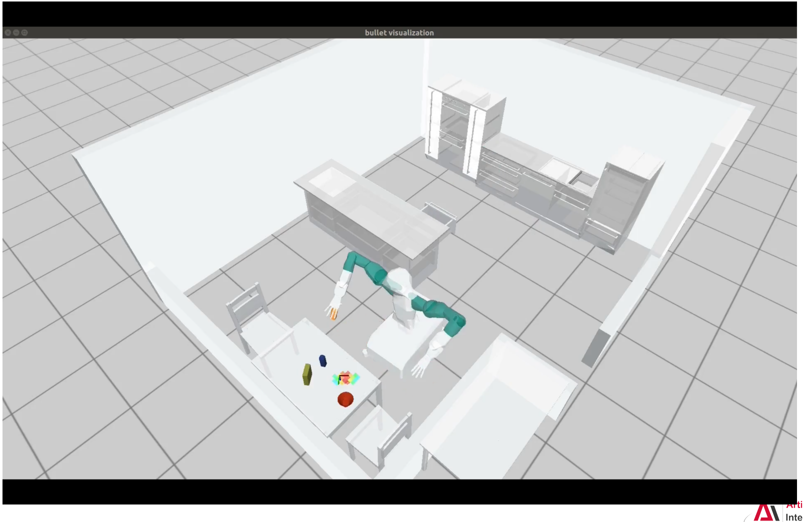


Motivation - Robot Integration

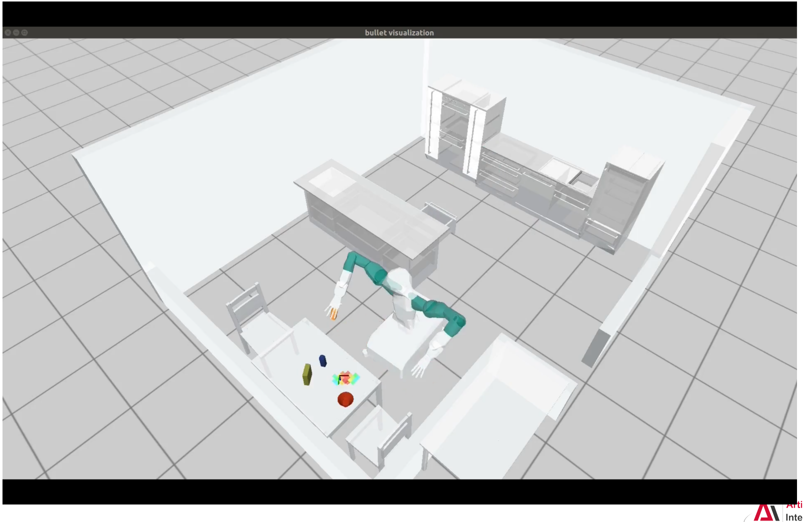




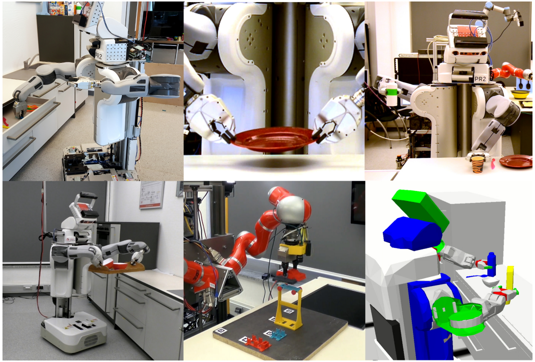
Motivation - Platform-Independent Control Program cont'd



Motivation - Platform-Independent Control Program cont'd



Motivation - Generalized Pick and Place Plans



One plan to accomplish all variations of fetch and place:

- different *objects*, *environments*, *robot platforms*, *applications*.

Motivation - Challenges Tackled by the Plan Executive

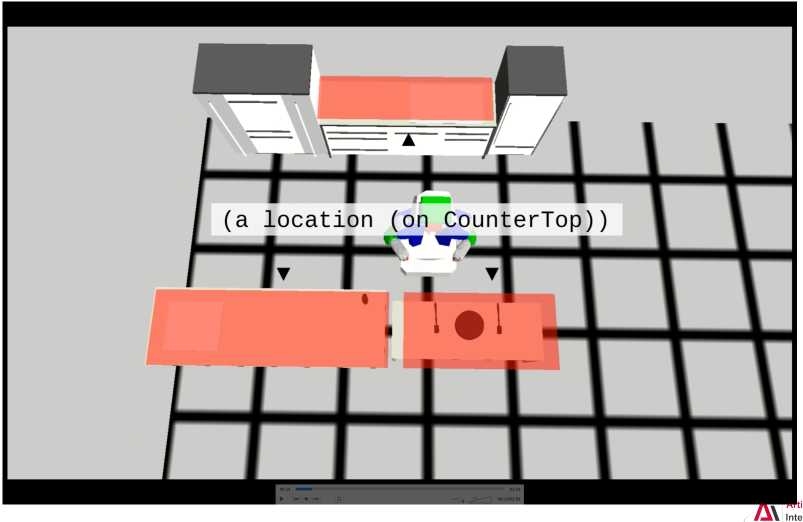
- Define which actions to execute to achieve the goal.
- Infer which parameters to use for each action.
- Monitor task execution and react to failures.

Motivation - Primitives: Motions and Percepts

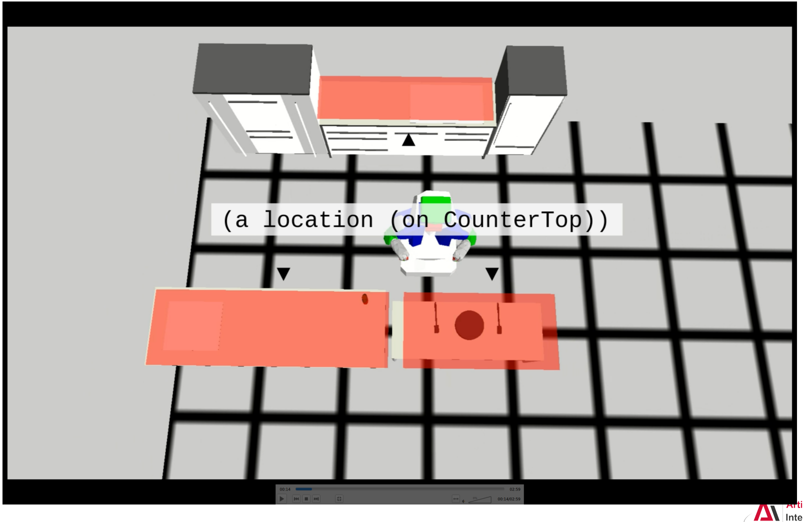
Primitive Tasks for Mobile Pick and Place Robots

Primitive	**Description**
going	drive or walk or fly to the goal pose
moving-torso	move torso to the goal joint position
moving-neck	move the neck to direct the gaze
moving-arm	execute a trajectory in Cartesian or joint space
grasping/releasing	move the fingers to grasp or release an object
opening-hand/cl.	move the fingers to open or close the hand
monitoring-joints	monitor the positions of robot body parts in space
detecting	perceive the described object in the environment
moving-eye	move the eye in the socket to direct the gaze
…	

Motivation - Sampling from Symbolic Description



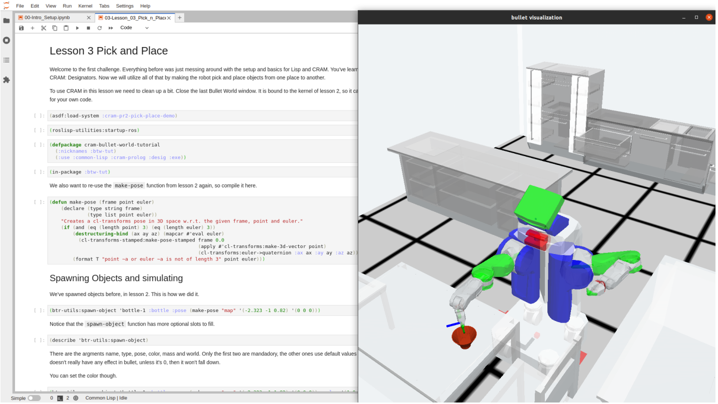
Motivation - Sampling from Symbolic Description



Workshop Technology

Workshop Technology - Plan Executive through Jupyter

Jupyter combines code with documentation. Each unit is a mix of explanatory text, and executable code.



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 learn 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 ca for your own code.

```
[ ]: (asdf:load-system :cram-pr2-pick-place-demo)
```

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

```
[ ]: (defpackage cram-bullet-world-tutorial
       (:nicknames :btw-tut)
       (:use :common-lisp :cram-prolog :desig :exe))
```

```
[ ]: (in-package :btw-tut)
```

We also want to re-use the `make-pose` function from lesson 2 again, so compile it here.

```
[ ]: (defun make-pose (frame point euler)
         (declare (type string frame)
                  (type list point euler))
         "Creates a cl-transforms pose in 3D space w.r.t. the given frame, point and euler."
         (if (and (eq (length point) 3) (eq (length euler) 3))
             (destructuring-bind (ax ay az) (mapcar #'eval euler)
               (cl-transforms-stamped:make-pose-stamped frame 0.0
                                                        (apply #'cl-transforms:make-3d-vector point)
                                                        (cl-transforms:euler->quaternion :ax ax :ay ay :az az))
             (format T "point ~a or euler ~a is not of length 3" point euler)))
```

Spawning Objects and simulating

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

```
[ ]: (btr-utils:spawn-object 'bottle-1 :bottle :pose (make-pose "map" '(-2.323 -1 0.82) '(0 0 0)))
```

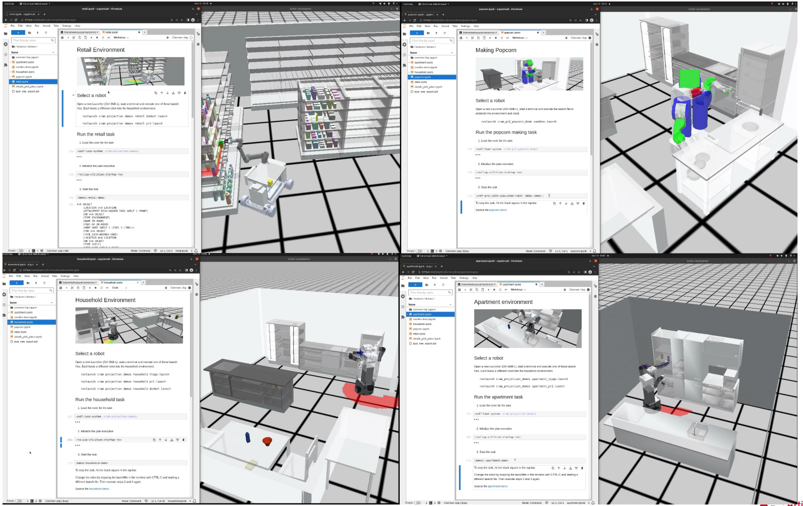
Notice that the `spawn-object` function has more optional slots to fill.

```
[ ]: (describe 'btr-utils:spawn-object)
```

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

You can set the color though.

Workshop Technology - Robot Integration



Retail Environment

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute one of these launch files. Each loads a different robot into the household environment.

```
roslaunch cram projection demos retail_donbot.launch

roslaunch cram projection demos retail pr2.launch
```

Run the retail task

1. Load the code for the task

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

2. Initialize the plan executive

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

3. Start the task

```
[3]: (demos:retail-demo)

[3]: #<A OBJECT
        (LOCATION #<A LOCATION
        (ATTACHMENT DISH-WASHER-TABS-SHELF-1-FRONT)
        (ON #<A OBJECT
        (TYPE ENVIRONMENT)
        (NAME DM-ROOM)
        (PART-OF DM-ROOM)
        (URDF-NAME SHELF-1-LEVEL-2-LINK)>)
        (FOR #<A OBJECT
        (TYPE DISH-WASHER-TABS)
        (LOCATION #<A LOCATION
        (ON #<A OBJECT
        (TYPE SHELF)
```

Making Popcorn

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute the launch file to establish the environment and robot.

```
roslaunch cram_pr2_popcorn_demo sandbox.launch
```

Run the popcorn making task

1. Load the code for the task

```
[1]: (asdf:load-system :cram-pr2-popcorn-demo)
```

2. Initialize the plan executive

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

3. Start the task

```
[*]: (urdf-proj:with-simulated-robot (demo::demo))
```

To stop the task, hit the black square in the top bar.

Source file popcorn-demo

Household Environment

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute one of these launch files. Each loads a different robot into the household environment.

```
roslaunch cram projection demos household tiago.launch

roslaunch cram projection demos household pr2.launch

roslaunch cram projection demos household donbot.launch
```

Run the household task

1. Load the code for the task

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

2. Initialize the plan executive

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

3. Start the task

```
[*]: (demos:household-demo)
```

To stop the task, hit the black square in the top bar.

Change the robot by stopping the launchfile in the terminal with CTRL-C and starting a different launch file. Then execute steps 2 and 3 again.

Source file household-demo

Apartment environment

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute one of these launch files. Each loads a different robot into the household environment.

```
roslaunch cram_projection_demos apartment_tiago.launch

roslaunch cram_projection_demos apartment_pr2.launch
```

Run the apartment task

1. Load the code for the task

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

2. Initialize the plan executive

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

3. Start the task

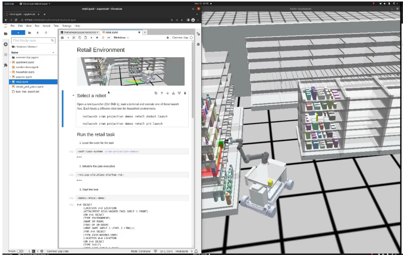
```
[*]: (demos::apartment-demo)
```

To stop the task, hit the black square in the top bar.

Change the robot by stopping the launchfile in the terminal with CTRL-C and starting a different launch file. Then execute steps 2 and 3 again.

Source file apartment-demo

Workshop Technology - Robot Integration



Retail Environment

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute one of these launch files. Each loads a different robot into the household environment.

```
roslaunch cram projection demos retail_donbot.launch
roslaunch cram projection demos retail pr2.launch
```

Run the retail task

1. Load the code for the task

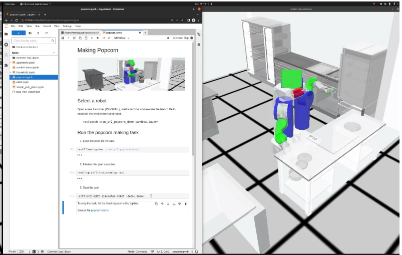
```
[1]: (asdf:load-system :cram-projection-demos)
```

2. Initialize the plan executive

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

3. Start the task

```
[3]: (demos:retail-demo)
[3]: #<A OBJECT
        (LOCATION #<A LOCATION
        (ATTACHMENT DISH-WASHER-TABS-SHELF-1-FRONT)
        (ON #<A OBJECT
        (TYPE ENVIRONMENT)
        (NAME DM-ROOM)
        (PART-OF DM-ROOM)
        (URDF-NAME SHELF-1-LEVEL-2-LINK)>)
        (FOR #<A OBJECT
        (TYPE DISH-WASHER-TABS)
        (LOCATION #<A LOCATION
        (ON #<A OBJECT
        (TYPE SHELF)
```



Making Popcorn

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute the launch file to establish the environment and robot.

```
roslaunch cram_pr2_popcorn_demo sandbox.launch
```

Run the popcorn making task

1. Load the code for the task

```
[1]: (asdf:load-system :cram-pr2-popcorn-demo)
```

2. Initialize the plan executive

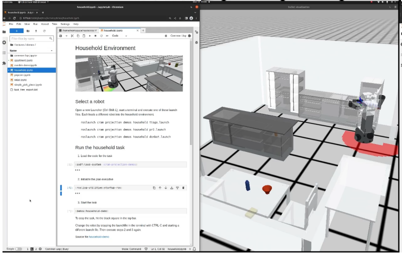
```
[2]: (roslisp-utilities:startup-ros)
```

3. Start the task

```
[*]: (urdf-proj:with-simulated-robot (demo::demo))
```

To stop the task, hit the black square in the top bar.

Source file popcorn-demo



Household Environment

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute one of these launch files. Each loads a different robot into the household environment.

```
roslaunch cram projection demos household tiago.launch
roslaunch cram projection demos household pr2.launch
roslaunch cram projection demos household donbot.launch
```

Run the household task

1. Load the code for the task

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

2. Initialize the plan executive

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

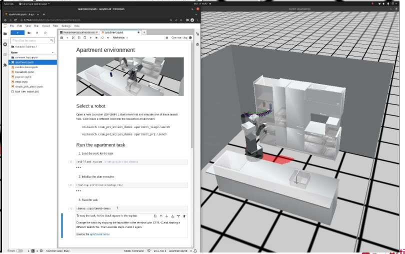
3. Start the task

```
[*]: (demos:household-demo)
```

To stop the task, hit the black square in the top bar.

Change the robot by stopping the launchfile in the terminal with CTRL-C and starting a different launch file. Then execute steps 2 and 3 again.

Source file household-demo



Apartment environment

Select a robot

Open a new Launcher (Ctrl-Shift-L), start a terminal and execute one of these launch files. Each loads a different robot into the household environment.

```
roslaunch cram_projection_demos apartment_tiago.launch
roslaunch cram_projection_demos apartment_pr2.launch
```

Run the apartment task

1. Load the code for the task

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

2. Initialize the plan executive

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

3. Start the task

```
[*]: (demos::apartment-demo)
```

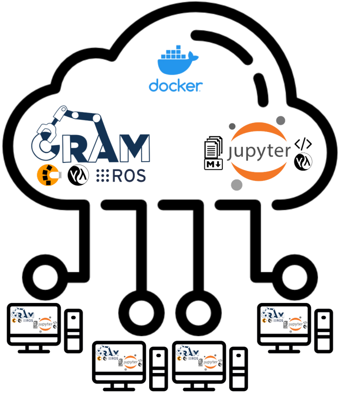
To stop the task, hit the black square in the top bar.

Change the robot by stopping the launchfile in the terminal with CTRL-C and starting a different launch file. Then execute steps 2 and 3 again.

Source file apartment-demo

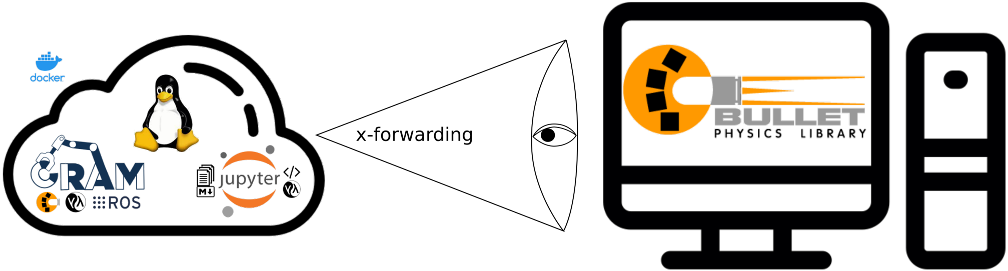
Workshop Technology - Cognitive Robotics for everyone

Docker is a manager vor virtual machines.
DockerHub hosts the virtual machine, ready to be downloaded



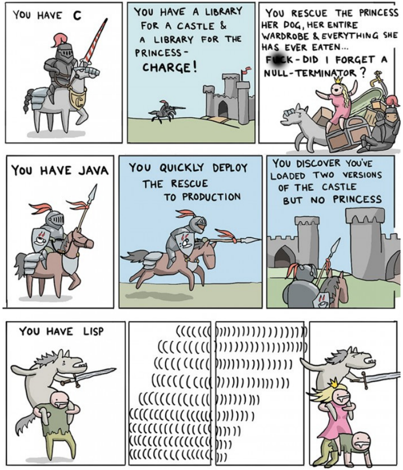
Workshop Technology - UI through X-Forwarding

Visual applications run in the virtual machine (Docker container) using X, which is a visualization technique for Linux systems. Docker can't visualize itself, so we forward the Bullet Physics Simulation to your PC.

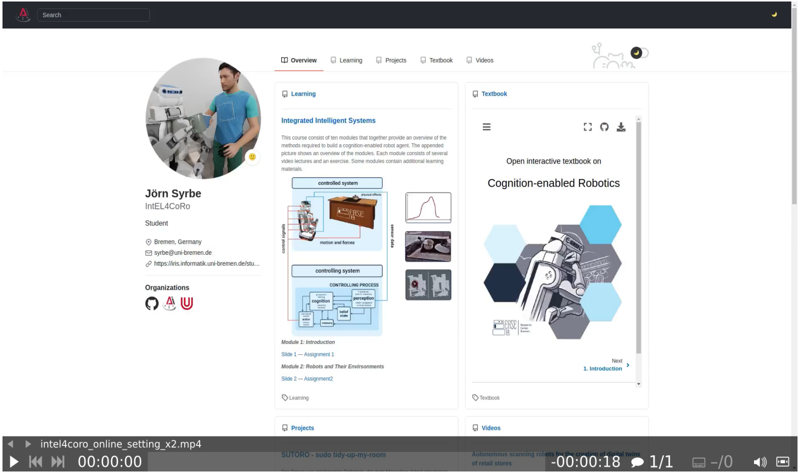


Hands-On - Learn Lisp!

http://cram-system.org


YOU HAVE C
YOU HAVE A LIBRARY FOR A CASTLE & A LIBRARY FOR THE PRINCESS - CHARGE!
YOU RESCUE THE PRINCESS HER DOG, HER ENTIRE WARDROBE & EVERYTHING SHE HAS EVER EATEN...
F[redacted]CK - DID I FORGET A NULL-TERMINATOR ?
C
YOU HAVE JAVA
YOU QUICKLY DEPLOY THE RESCUE TO PRODUCTION
YOU DISCOVER YOU'VE LOADED TWO VERSIONS OF THE CASTLE BUT NO PRINCESS
YOU HAVE LISP

Prospect - Online Learning Hub



Jörn Syrbe

IntEL4CoRo

Student

Bremen, Germany

syrbe@uni-bremen.de

https://iris.informatik.uni-bremen.de/stu...

Organizations

Learning

Integrated Intelligent Systems

This course consist of ten modules that together provide an overview of the methods required to build a cognition-enabled robot agent. The appended picture shows an overview of the modules. Each module consists of several video lectures and an exercise. Some modules contain additional learning materials.

Module 1: Introduction

Slide 1 — Assignment 1

Module 2: Robots and Their Envirsonments

Slide 2 — Assignment2

Learning

Textbook

Open interactive textbook on

Cognition-enabled Robotics

Next

1. Introduction

Textbook

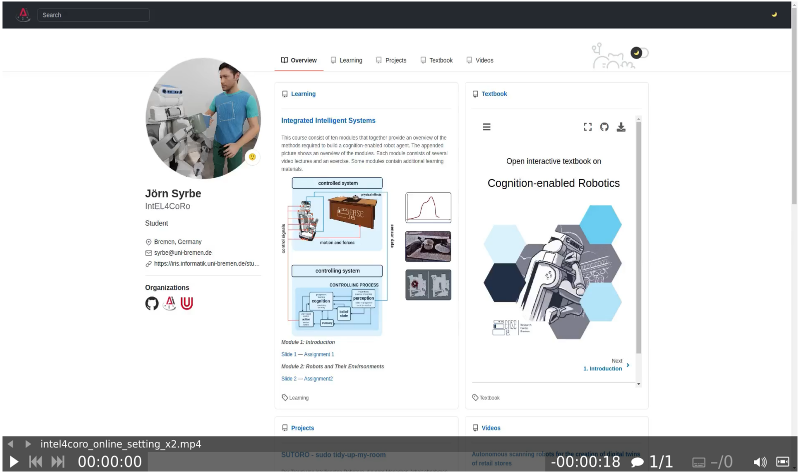
Projects

SUTORO - sudo tidy-up-my-room

Videos

Autonomous scanning robots for the creation of digital twins of retail stores

Prospect - Online Learning Hub



Jörn Syrbe
IntEL4CoRo

Student

Bremen, Germany
syrbe@uni-bremen.de
https://iris.informatik.uni-bremen.de/stu...

Organizations

Learning

Integrated Intelligent Systems

This course consist of ten modules that together provide an overview of the methods required to build a cognition-enabled robot agent. The appended picture shows an overview of the modules. Each module consists of several video lectures and an exercise. Some modules contain additional learning materials.

Module 1: Introduction

Slide 1 — Assignment 1

Module 2: Robots and Their Envirsonments

Slide 2 — Assignment2

Learning

Textbook

Open interactive textbook on

Cognition-enabled Robotics

Next
1. Introduction

Textbook

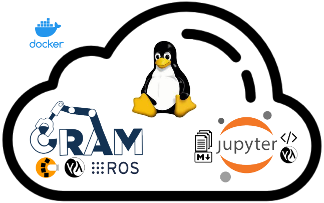
Projects

SUTORO - sudo tidy-up-my-room

Videos

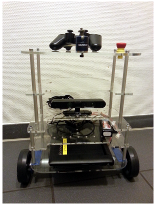
Autonomous scanning robots for the creation of digital twins of retail stores

Prospect - Robot Programming Course



JupyterHub

Robot Operating System (ROS)



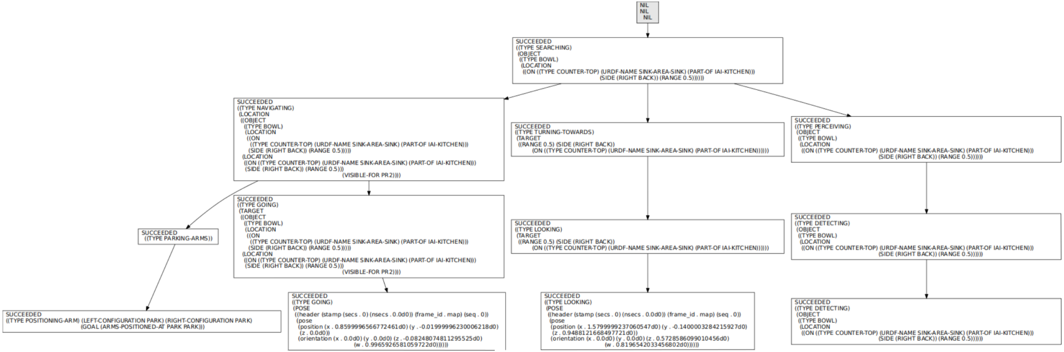
Robot platform

Prospect - Data Analysis

Primitive Tasks for Mobile Pick and Place Robots

Primitive	**Description**
going	drive or walk or fly to the goal pose
moving-torso	move torso to the goal joint position
moving-neck	move the neck to direct the gaze
moving-arm	execute a trajectory in Cartesian or joint space
grasping/releasing	move the fingers to grasp or release an object
opening-hand/cl.	move the fingers to open or close the hand
monitoring-joints	monitor the positions of robot body parts in space
detecting	perceive the described object in the environment
moving-eye	move the eye in the socket to direct the gaze
...	

Prospect - Data Analysis cont'd



```
NIL
NIL
  NIL
```

```
SUCCEEDED
((TYPE SEARCHING)
 (OBJECT
  ((TYPE BOWL)
   (LOCATION
    ((ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
                              (SIDE (RIGHT BACK)) (RANGE 0.5))))))
```

```
SUCCEEDED
((TYPE NAVIGATING)
 (LOCATION
  ((OBJECT
    ((TYPE BOWL)
     (LOCATION
      ((ON
        ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
       (SIDE (RIGHT BACK)) (RANGE 0.5)))))
   (LOCATION
    ((ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
     (SIDE (RIGHT BACK)) (RANGE 0.5)))
                                  (VISIBLE-FOR PR2))))
```

```
SUCCEEDED
((TYPE TURNING-TOWARDS)
 (TARGET
  ((RANGE 0.5) (SIDE (RIGHT BACK))
        (ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN))))))
```

```
SUCCEEDED
((TYPE PERCEIVING)
 (OBJECT
  ((TYPE BOWL)
   (LOCATION
    ((ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
                              (SIDE (RIGHT BACK)) (RANGE 0.5))))))
```

```
SUCCEEDED
 ((TYPE PARKING-ARMS))
```

```
SUCCEEDED
((TYPE GOING)
 (TARGET
  ((OBJECT
    ((TYPE BOWL)
     (LOCATION
      ((ON
        ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
       (SIDE (RIGHT BACK)) (RANGE 0.5)))))
   (LOCATION
    ((ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
     (SIDE (RIGHT BACK)) (RANGE 0.5)))
                                  (VISIBLE-FOR PR2))))
```

```
SUCCEEDED
((TYPE LOOKING)
 (TARGET
  ((RANGE 0.5) (SIDE (RIGHT BACK))
        (ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN))))))
```

```
SUCCEEDED
((TYPE DETECTING)
 (OBJECT
  ((TYPE BOWL)
   (LOCATION
    ((ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
                              (SIDE (RIGHT BACK)) (RANGE 0.5))))))
```

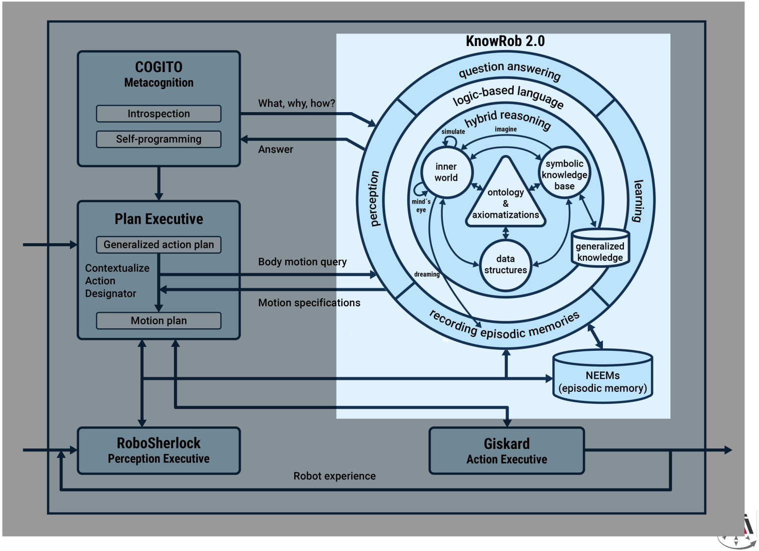
```
SUCCEEDED
((TYPE POSITIONING-ARM) (LEFT-CONFIGURATION PARK) (RIGHT-CONFIGURATION PARK)
                         (GOAL (ARMS-POSITIONED-AT PARK PARK)))
```

```
SUCCEEDED
((TYPE GOING)
 (POSE
  ((header (stamp (secs . 0) (nsecs . 0.0d0)) (frame_id . map) (seq . 0))
   (pose
    (position (x . 0.8599996566772461d0) (y . -0.01999996230006218d0)
     (z . 0.0d0))
    (orientation (x . 0.0d0) (y . 0.0d0) (z . -0.08248074811295525d0)
                    (w . 0.9965926581059722d0))))))
```

```
SUCCEEDED
((TYPE LOOKING)
 (POSE
  ((header (stamp (secs . 0) (nsecs . 0.0d0)) (frame_id . map) (seq . 0))
   (pose
    (position (x . 1.5799999237060547d0) (y . -0.1400003284215927d0)
     (z . 0.9488121668497721d0))
    (orientation (x . 0.0d0) (y . 0.0d0) (z . 0.5728586099010456d0)
                    (w . 0.8196542033456802d0))))))
```

```
SUCCEEDED
((TYPE DETECTING)
 (OBJECT
  ((TYPE BOWL)
   (LOCATION
    ((ON ((TYPE COUNTER-TOP) (URDF-NAME SINK-AREA-SINK) (PART-OF IAI-KITCHEN)))
                              (SIDE (RIGHT BACK)) (RANGE 0.5))))))
```

Knowledge Representation & Reasoning



Thank you for your attention!

Artificial Intelligence

euROBIN