Modular OpenRobot Simulation Engine

The ROS-middleware for MORSE

Michael Karg

Intelligent Autonomous Systems Group
Department of Computer Science
Technische Universität München

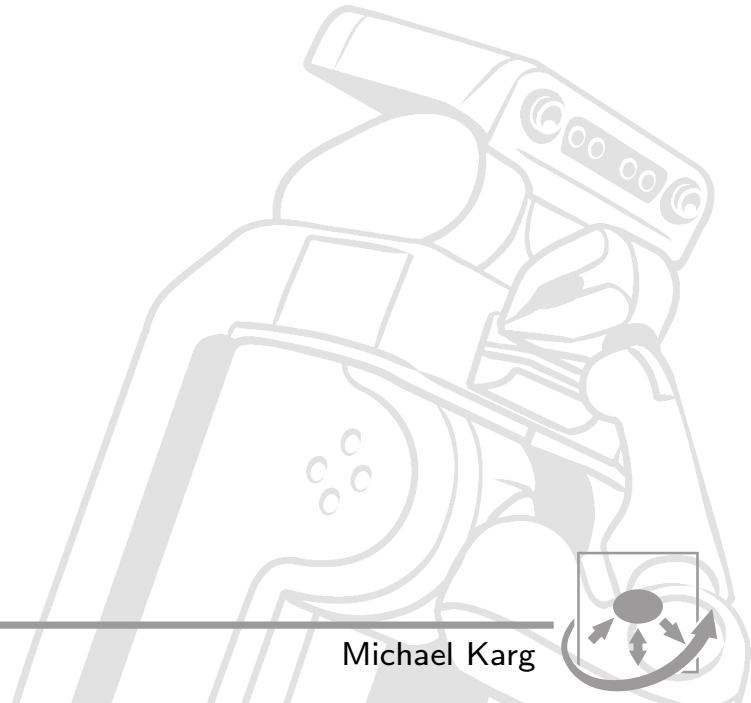
euRobotics Forum 2011 / Västerås

Intelligent Autonomous Systems

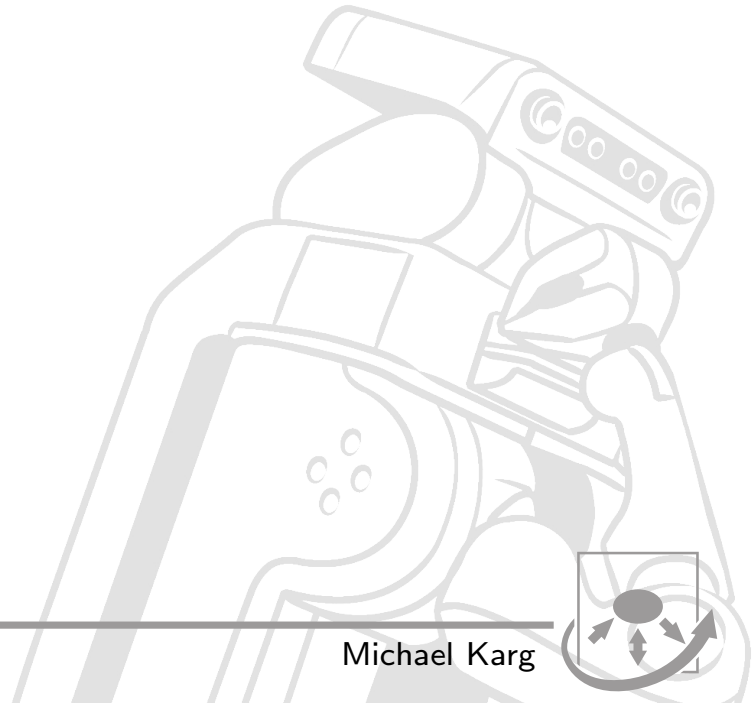
IAS TUM Institute for Advanced Study

Outline

1. The IAS TUM group and the Adapto project
2. The MORSE simulator
3. MORSE and ROS
4. Application scenarios



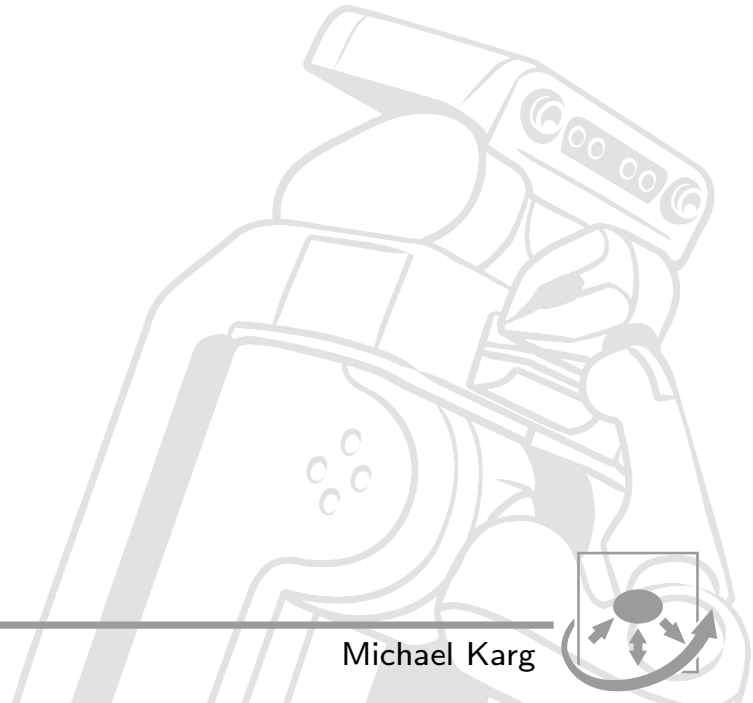
Outline



Who we are

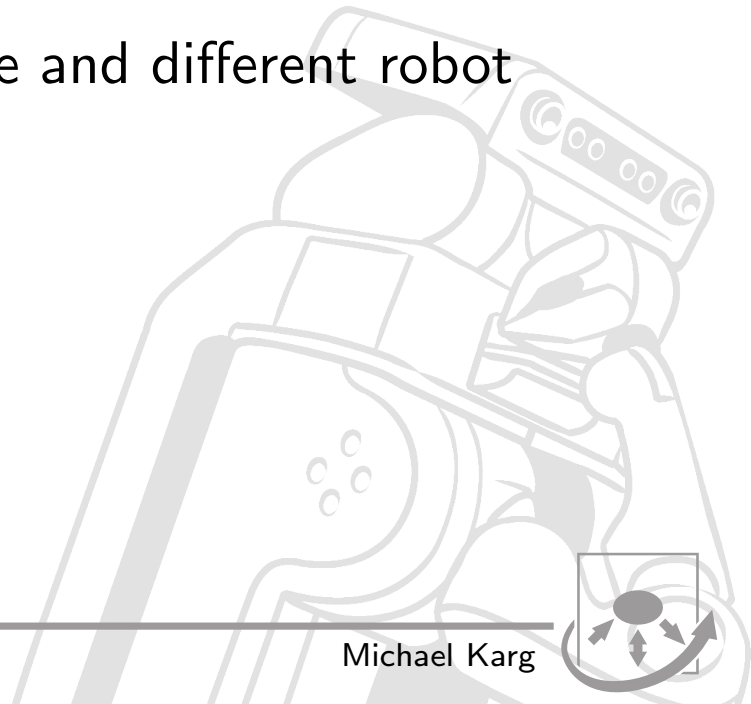
- Part of the TUM - Intelligent Autonomous Systems group of Prof. Beetz
- Adapto project under the supervision of Dr. Alexandra Kirsch
- Plan based control mechanisms for human-robot interaction in domestic enviroments
- Reactive, opportunistic and adaptive planning
- Failure recognition and avoidance using expectation models
- Continual adaptation by learning prediction models
- Human aware navigation

Outline



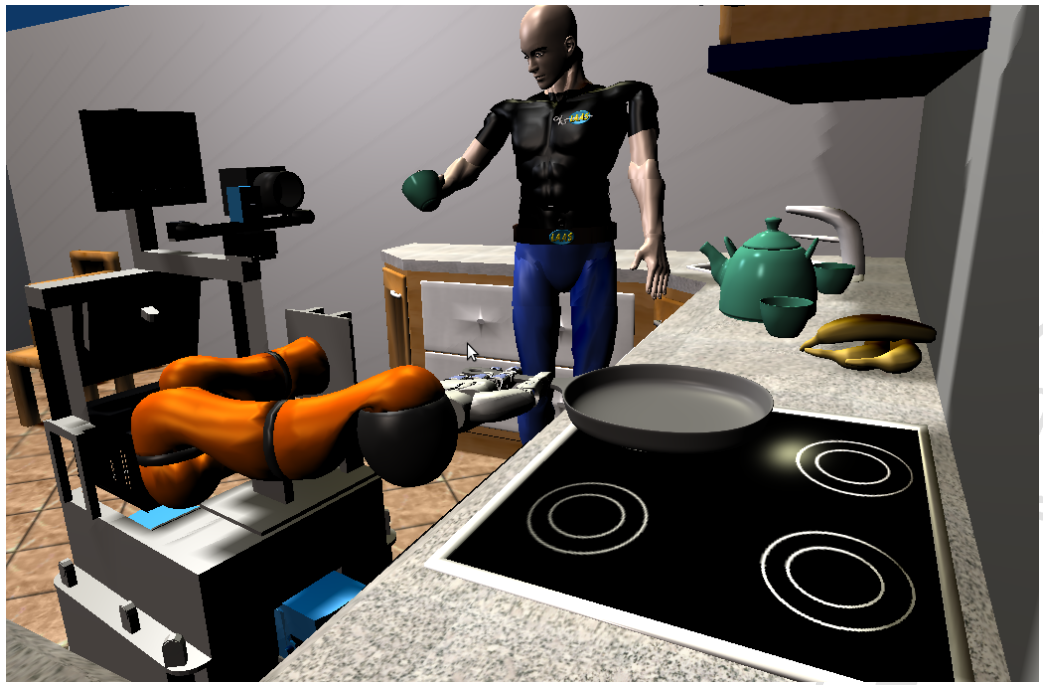
Why simulation?

- Simulation allows experiments on specific parts of a robotic system, (e.g. task planning)
- Real robot behavior is often hard to repeat (due to sensing, battery charge, ...)
- Many diverse enviroments are possible and different robot platforms are available at low cost

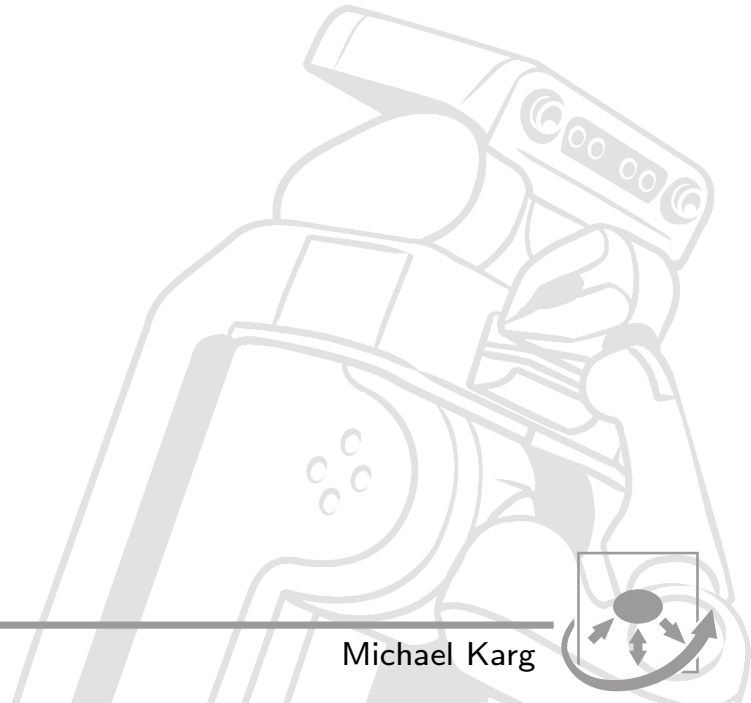


Why MORSE for our simulation?

- Modularity allows to regulate level of detail according to our needs (e.g. task planning)
- Human model is already available
- Human robot interaction scenarios can easily be generated in a computer-game like way



Outline



Robot Operating System

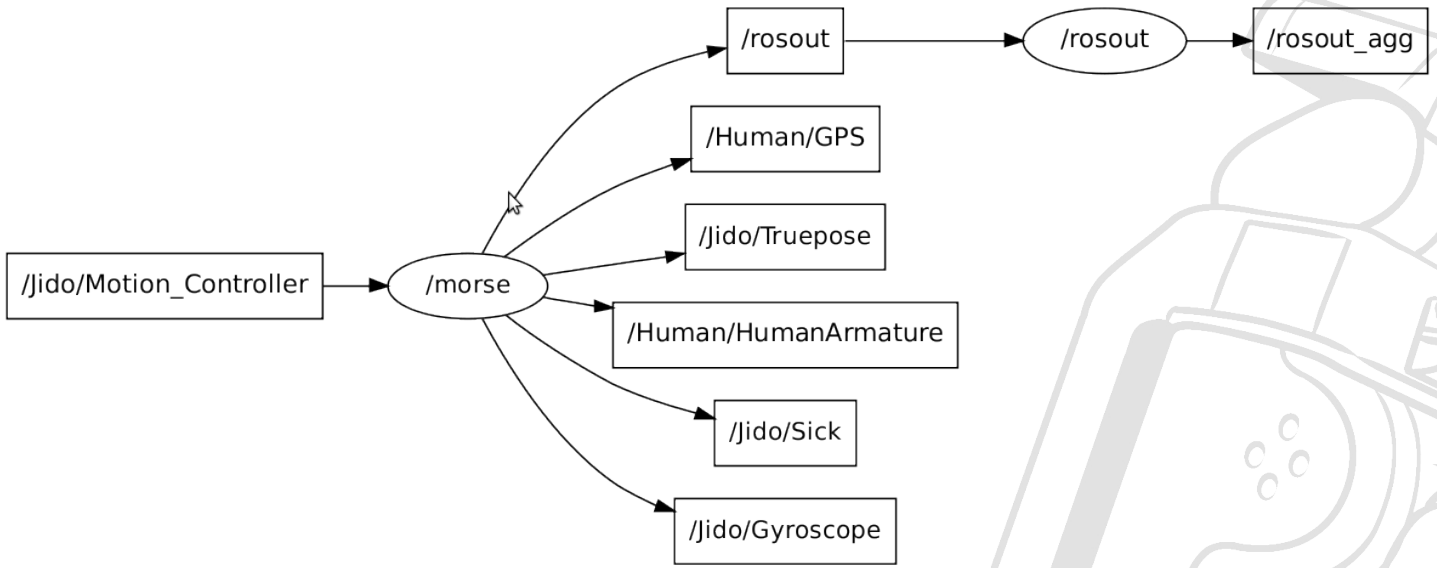
- Operating system for robotics by Willow Garage
- Open source, BSD licensed
- Big, active community
- Documentation, tutorials, tech support

More information:

`http://www.ros.org`

The MORSE - ROS middleware

- ROS uses so called nodes and topics for communication with one master node
- The ROS-middleware in MORSE creates a MORSE-rosnode and one topic for every sensor and actuator according to the following scheme: [robot name]/[sensor or actuator name]
- Communication with roscore using standard ROS messages to assure compatibility with standard ROS components



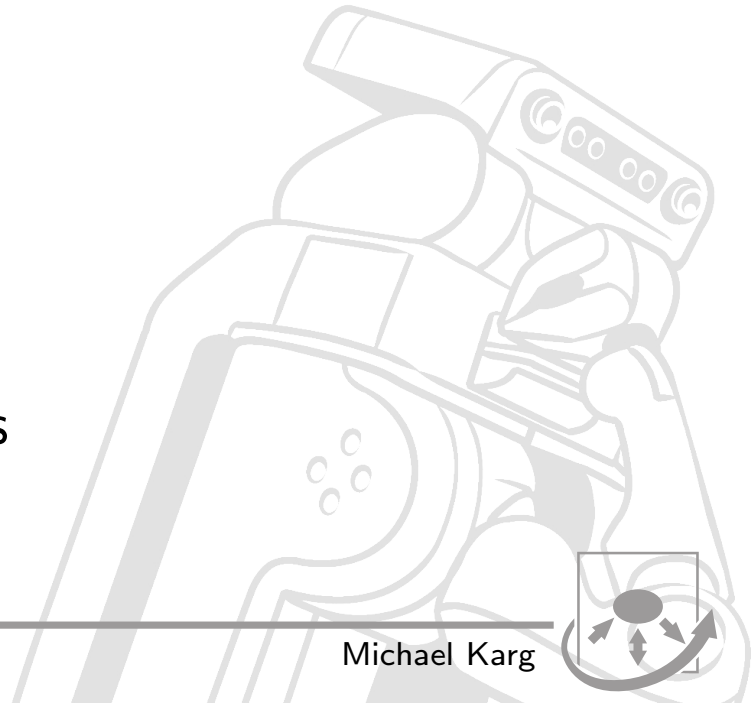
Python 2 VS Python 3

- Blender uses Python3
- ROS uses Python 2.6
- Python 3 is NOT downwardly compatible
- Porting of ROS-messaging parts to work on Python 2 AND 3
- Can safely be used with ROS overlays without affecting your Python2-ROS-installation
- Easy installation by executing one rosinstall-file

Installation instructions:

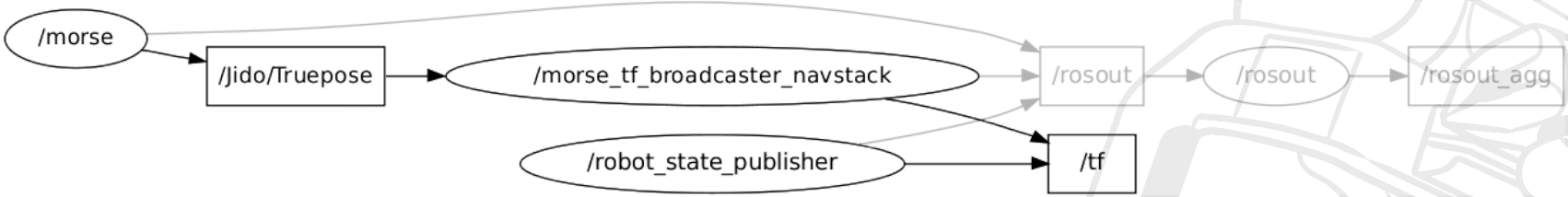
`http://www.openrobots.org/wiki/morse`

Outline



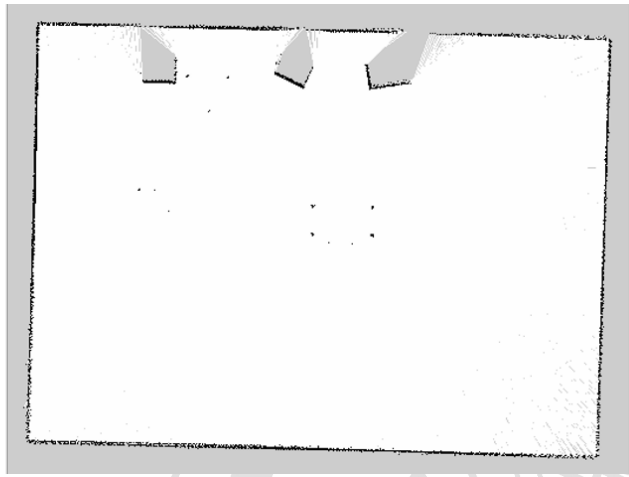
Transform between multiple coordinate frames

- ROS TF builds a tree of all coordinate frames and provides means of conversion between them
- Create map and odometry coordinate frames based on the origin of the blender map and the robot truepose
- Create coordinate-frames of the robot using a robot-URDF and ROS robot state publisher
- Robot can easily exchanged if you have a URDF-file of it

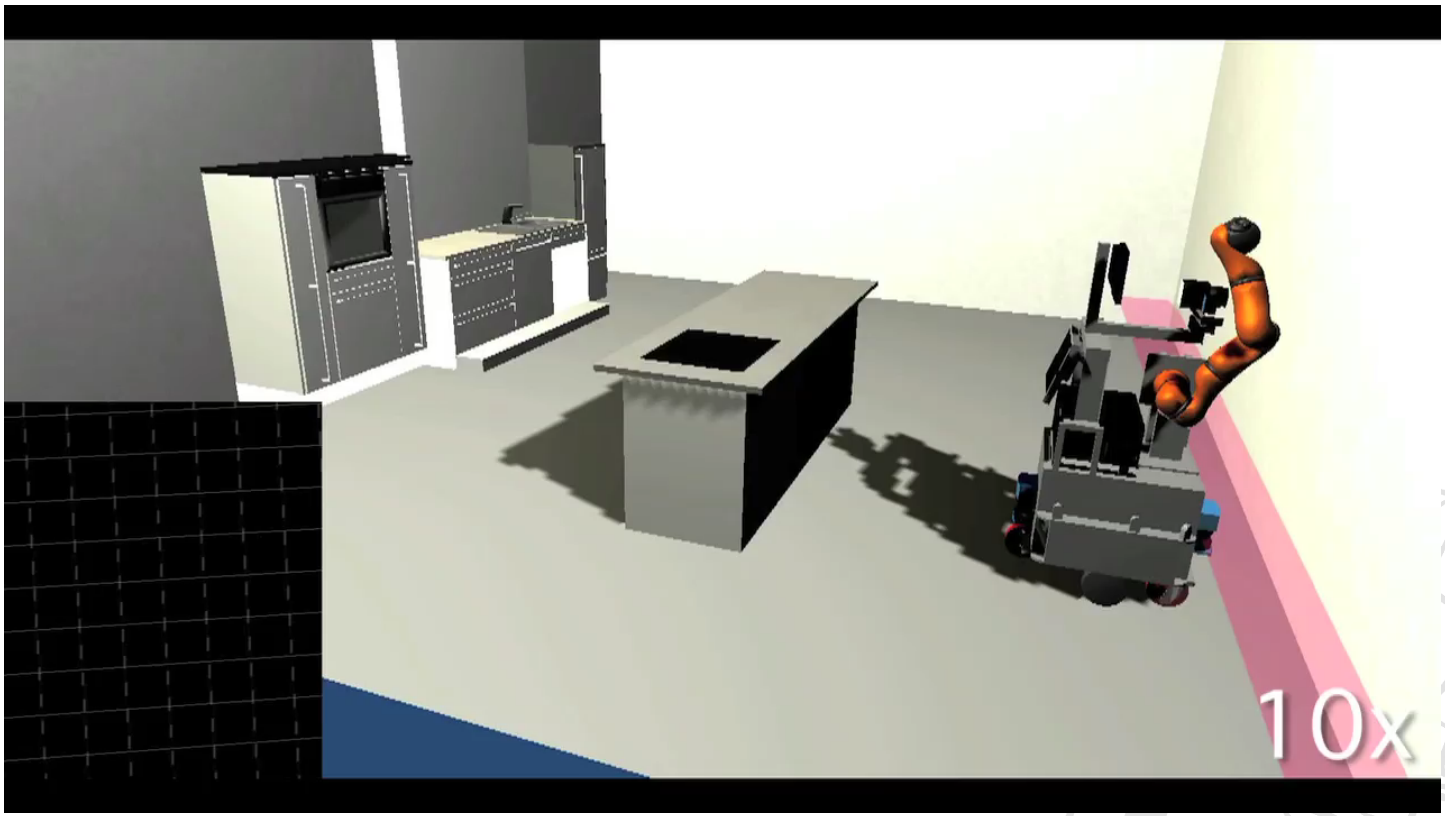


Using ROS Gmapping

- Easy way to create maps of Blender scenarios
- Works out of the box
- Needs robot pose (Truepose or Localization), TF-tree and data of laserscanner
- Collision bounds define collision with laser scan

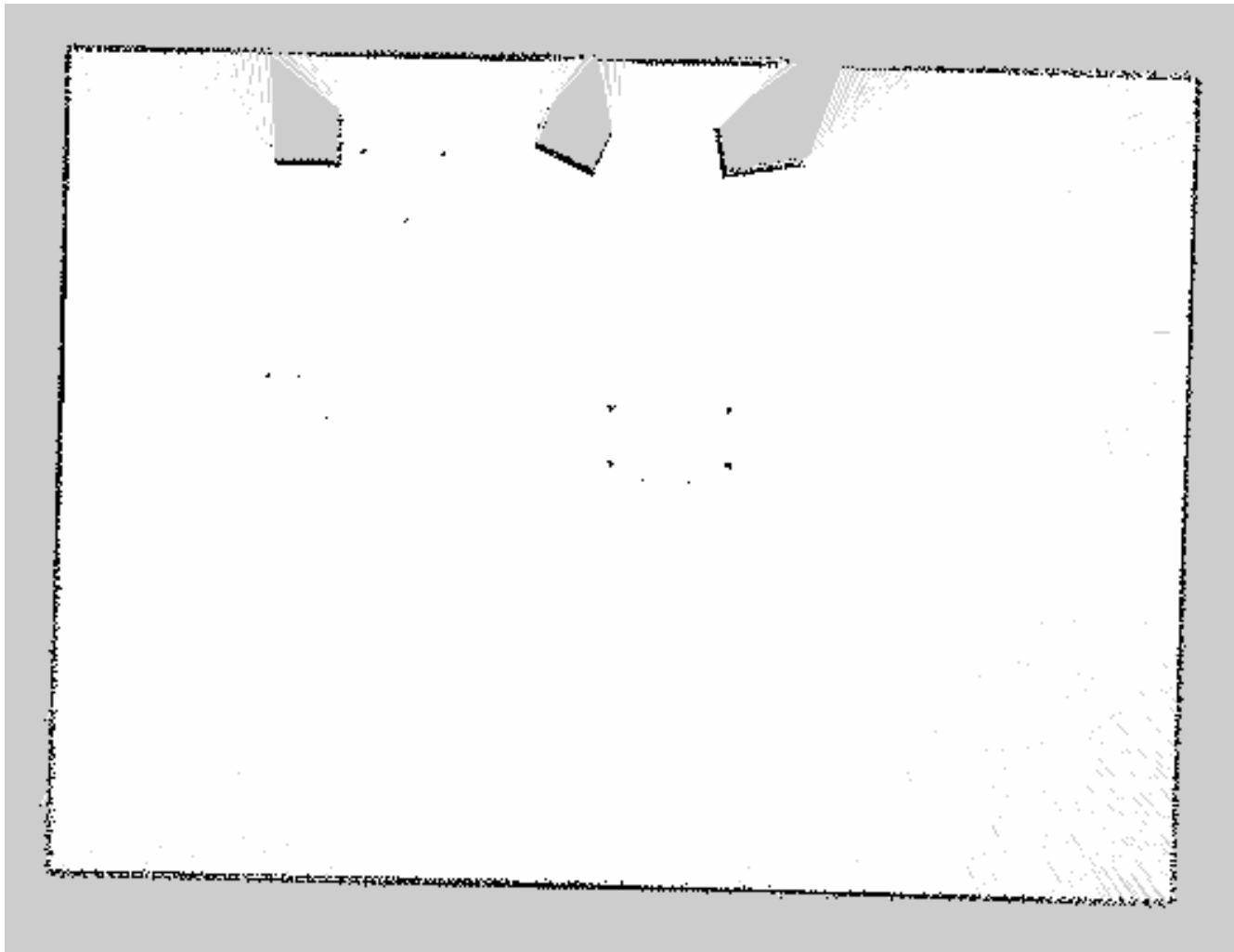


Video: Gmapping in MORSE (1)



Video: Gmapping in MORSE (2)

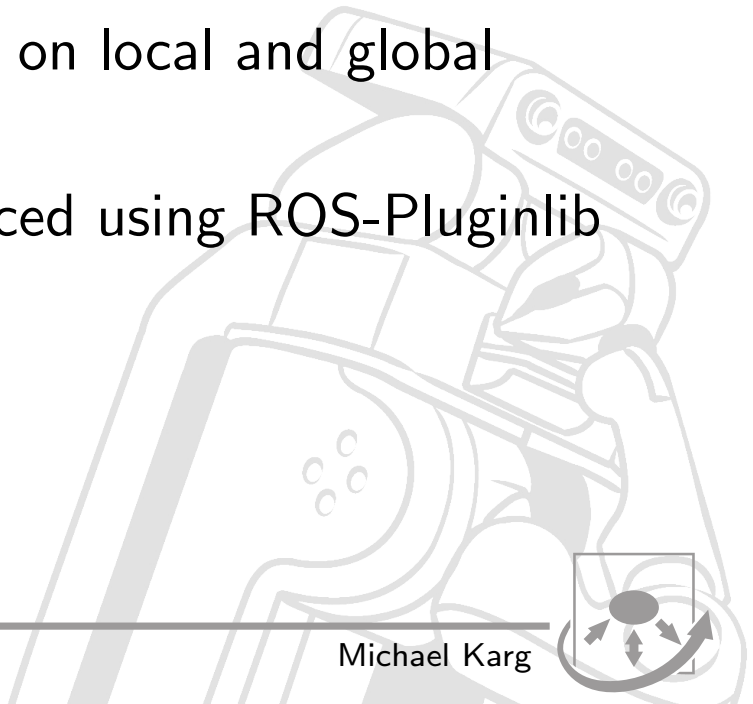
Video: Gmapping in MORSE (3)



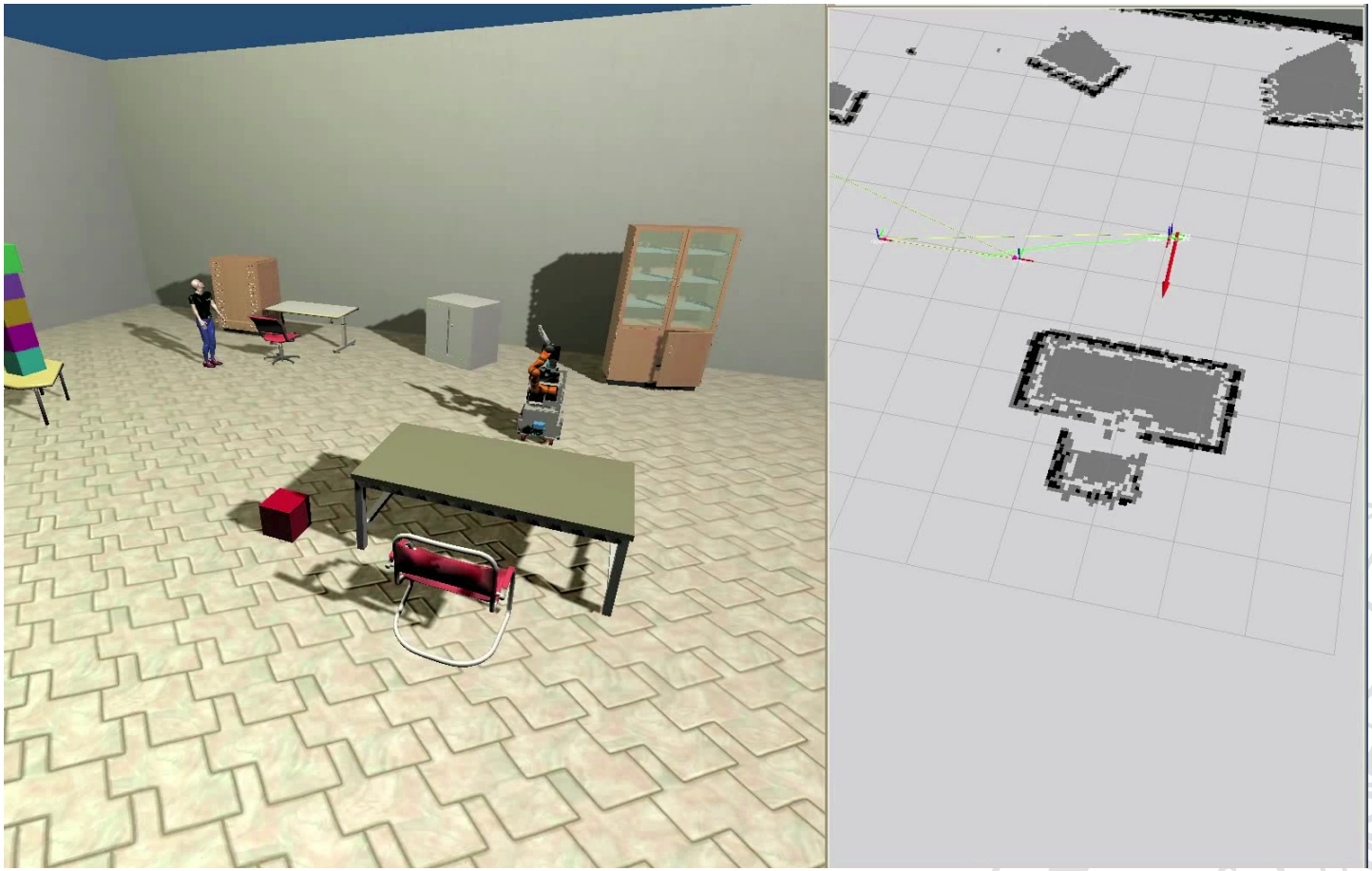
Video: Gmapping in MORSE (3)

Using the ROS navigation stack in MORSE

- 2D navigation
- Takes information from odometry, sensor streams and a goal-pose and outputs safe velocity commands that can be used by a MORSE motion-actuator
- Uses TF-tree and map
- Comes with obstacle avoidance based on local and global costmaps
- Global and local planner can be replaced using ROS-Pluginlib
- Works with real PR2-robot

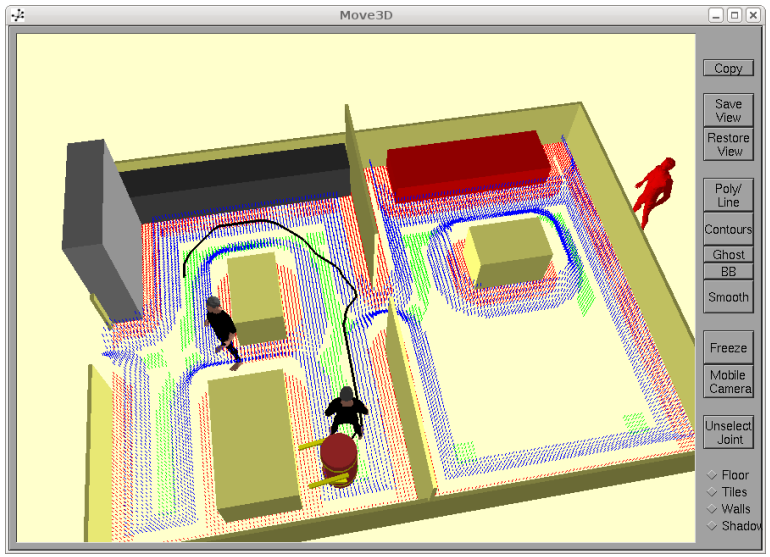


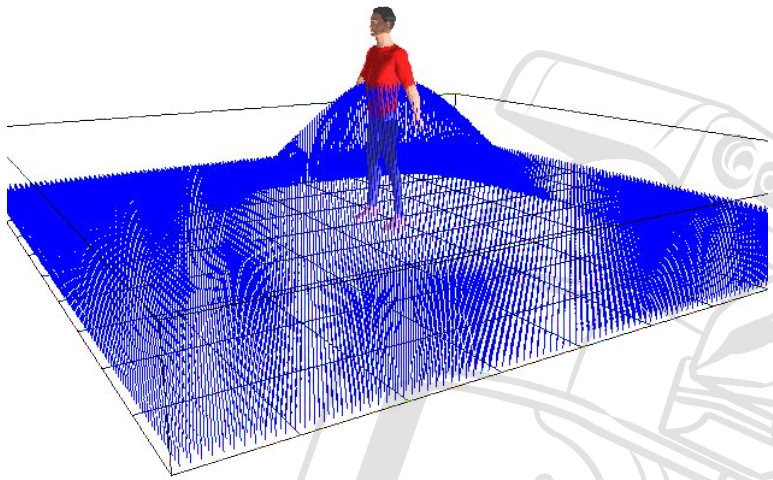
Video: ROS Navigation stack in MORSE



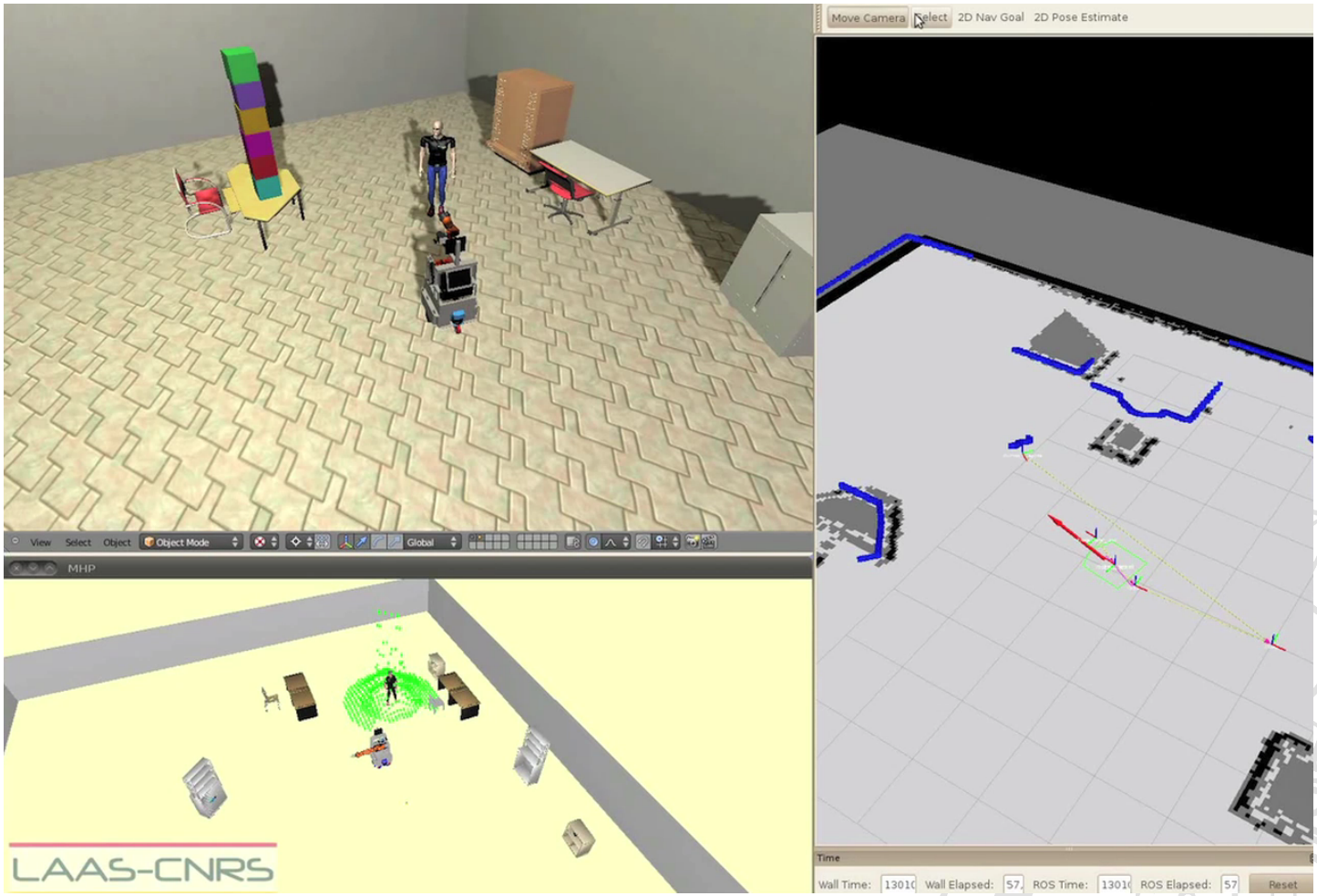
Human friendly navigation

- Replaces global planner of navigation stack
- Integrates human into robot path-planning
- Based on Move3D by LAAS CNRS
- Collision-free motion planning
- Social cost-function around the human pose





Video: Human friendly navigation



Controlling the Kuka-arm

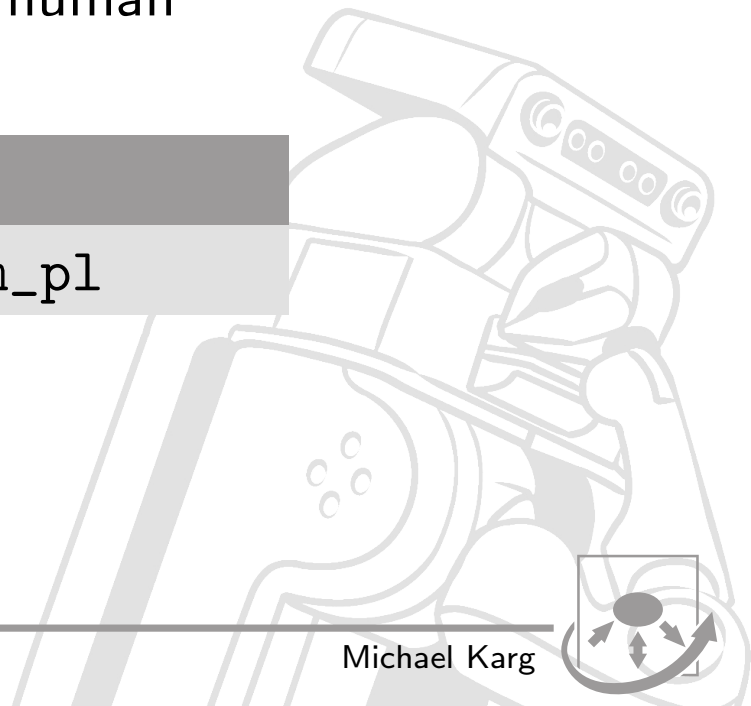
- Send ROS-Jointstate-message to control every joint of the arm

Outlook: CRAM bindings for MORSE

- Cognitive Robot Abstract Machine
- Plan-based control of autonomous robots
- Based on CommonLISP
- Complex failure handling
- Task synchronization, parallel execution, resource management
- Goal: Perform complex activities in a human household

CRAM-PL ROS-package:

`http://www.ros.org/wiki/cram_pl`

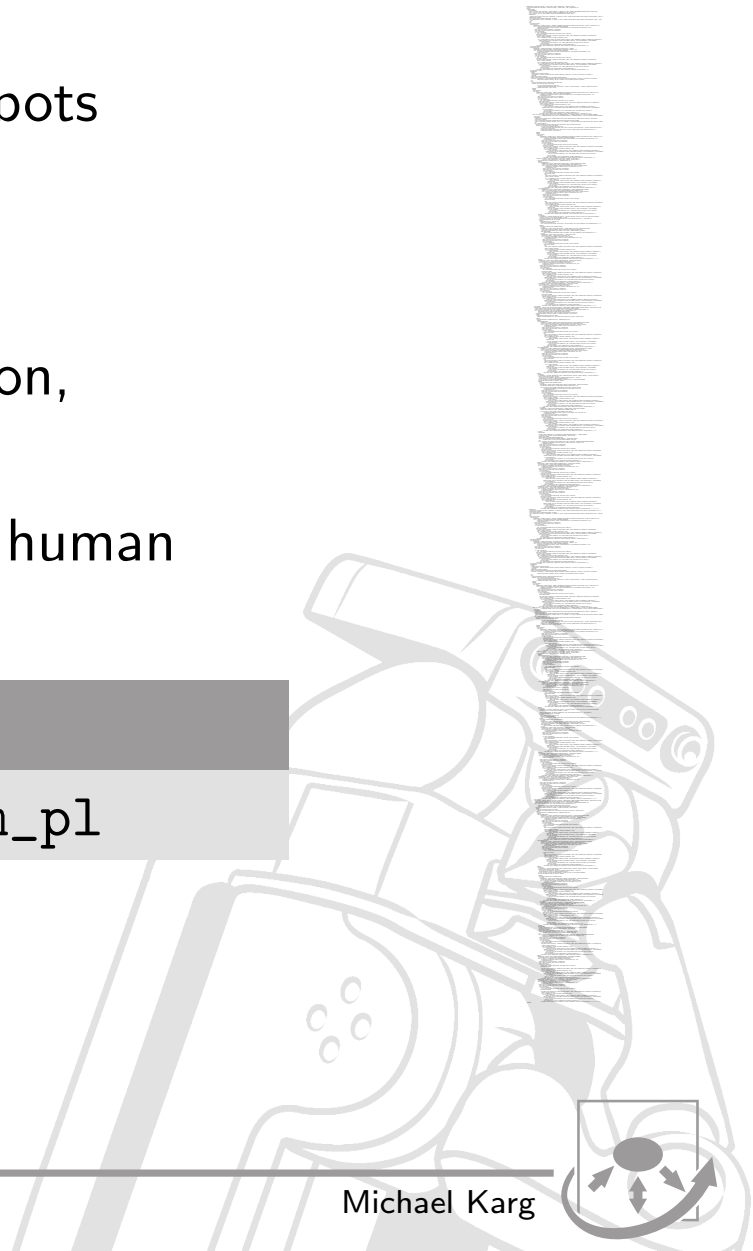


Outlook: CRAM bindings for MORSE

- Cognitive Robot Abstract Machine
- Plan-based control of autonomous robots
- Based on CommonLISP
- Complex failure handling
- Task synchronization, parallel execution, resource management
- Goal: Perform complex activities in a human household

CRAM-PL ROS-package:

`http://www.ros.org/wiki/cram_pl`



Outlook: CRAM bindings for MORSE

```
(LET ((INNER-CONTACTS NIL))
  (WITH-FAILURE-HANDLING FAILURE ((CARRY-TRIES-COUNT CARRY-TRIES) (GRIP-TRIES-COUNT
   (RECOVER ((TYPEP FAILURE 'ENTITY-LOST-FAILURE)
             (LET ((SIDE (ENTITY-GRIPPING-SIDE ENTITY NIL)))
               (HANDLE-PLAN-FAILURE CARRY-TRIES-COUNT :ENTITY ENTITY :DO-ALWAYS ((E
            ((TYPEP FAILURE 'GRIP-FAILURE)
             (HANDLE-PLAN-FAILURE GRIP-TRIES-COUNT :ENTITY ENTITY :DO-RETRY ((RECOV
            (T (HANDLE-PLAN-FAILURE 0 :ENTITY ENTITY)))
   (MONITOR)
   (PERFORM
    (:TAG FIND-ENTITY
     (SETF ENTITY
           (EXPANDED-GOAL (:PERCEIVE ENTITY) :PERCEIVE ((DESIGNATOR TR-RULE-NAME SK
            (LET* ((#:GOAL1359 (MAKE-INSTANCE 'ENTITY-FOUND :DESIGNATOR DESIGNATOR)
                   (#:ROUTINE1360 (ARBITRATION #:GOAL1359 (COGITO::FILTER-SETTINGS
                   (#:ROUTINE-RES1361 NIL))
              (SETGV :GOAL-TASK (TYPE-OF #:GOAL1359) #:TAG-GOAL1363)
              (PULSE (GETGV :GOAL-START-FLUENT (TYPE-OF #:GOAL1359)))
              (:TAG #:TAG-GOAL1363
               (IF (NULL #:ROUTINE1360)
                   (FAIL :CLASS NO-ROUTINE-FOUND-FAILURE :GOAL-CLASS (TYPE-OF #:GOA
                 (EVAP-AND-FAIL-PROTECT
                  (SEQ (WHEN (TYPEP #:ROUTINE1360 'EXTERNAL-LOW-LEVEL-ROUTINE) (ADD
                   (COGITO::ADD-GOAL #:GOAL1359)
                   (LET ((#:TIMEMODEL1364 (COGITO::GET-MODEL #:ROUTINE1360 :TIME)))
```


Summary

- MORSE in combination with ROS offer modular simulation adapted to your level of detail with widely used components and a big community
- Outlook:
  - Full Python3 support for ROS
  - High-level HRI scenarios using CRAM language
  - Intuitive Interface for human control
  - Human robot interaction (Cooperative kitchen scenarios)

Summary:

Start using MORSE and ROS today!

The end

Any questions?