Outline

- Justin Simulator
  - Aims and Architecture
  - Demo
- GeRT Action Learning
  - Overview of GeRT
  - Action Learning
  - Code Examples
  - Planning Framework
  - Strategy for Learning Action Models
DLR’s Justin Robot

- 43 DoF in-house developed mobile robot
- Video
Justin Simulator

- Prototyping, testing
- Matlab / Simulink
- OpenRAVE
- Not physics-based
Justin Simulator

- Architecture. Commonality with real robot exec env

Diagram by Florian Schmidt, DLR
Simulator Demo

- Basic cup-stacking
- Code overview
- Inverse kinematics vs. RRT planner
- Two handed cup-stacking
- Frames and positions, objects, etc
- Limitations
Part II: GeRT

- Justin Simulator
  - Aims and Architecture
  - Demo

- GeRT Action Learning
  - Overview of GeRT
  - Action Learning
  - Code Examples
  - Planning Framework
  - Strategy for Learning Action Models
GeRT Project

- GeRT: Generalizing Robot Manipulation Tasks
  
  *enable a robot to autonomously generalise its manipulation skills from a set of known objects to previously un-manipulated objects in order to achieve an everyday manipulation task.*

- Institutions:
  - German Aerospace Center (DLR)
  - University of Birmingham (Bham)
  - University of Örebro (ORU)
  - Max Planck Institute for Biological Cybernetics (MPI)

- Workpackages:
  - WP1 (Bham): Planning (hybrid + learning)
  - WP2: Perception, scene representation and object classification
  - WP3 (Bham): Learning and optimising grasping
  - WP4: Integration and system architecture
GeRT WP1: Action Learning

- Objective: learn general action models from Python code blocks
- Several existing example programs
- Assume robot coders are not planning specialists
annotate_log("place cup")
# lower cart. speed to 30% to avoid a too hard contact
set_var("cart_speed", 0.3)
# stop going down as soon as contact with table is detected
guard.enable({"right_tcp": [(2, "<", -5.00)]})
# start going down
goto_frame("right at cup stack %d" % target_stack)
did_stop = guard.did_stop()
annotate_log("guard stopped: %d" % did_stop)  # whether or not the
# disable guard for further motions
guard.disable()
set_var("cart_speed", 1)  # reset cart speed to 100%

annotate_log("release cup")
goto_pos("right upside pre grasp", vias=["right upside preshape"],
app.rave_interface.release("rightArm", "mug")
goto_frame("right above cup stack %d" % target_stack)
annotate_log("place cup")

# lower cart. speed to 30% to avoid a too hard contact
set_var("cart_speed", 0.3)

# stop going down as soon as contact with table is detected
guard.enable({"right_tcp": [(2, ",", -5.00)]})

# start going down

goto_frame("right at cup stack %d" % target_stack)

did_stop = guard.did_stop()
annotate_log("guard stopped: %d" % did_stop) # whether or not the
guard.disable() # disable guard for further motions
set_var("cart_speed", 1) # reset cart speed to 100%

annotate_log("release cup")

goto_pos("right upside pre grasp", vias=["right upside preshape"]
app.rave_interface.release("rightArm", "mug")

goto_frame("right above cup stack %d" % target_stack)
Python code

annotate_log("place cup")
# lower cart. speed to 30% to a
set_var("cart_speed", 0.3)
...

**put_on_table** (hand, table_loc, object)

**PRECONDITIONS:**
holds(hand, object)
clear(table_loc, object)
reachable(hand, table_loc)

**EFFECTS:**
at_location(object, table_loc)
¬holds(hand, object)
¬clear(table_loc, object)
- Hybrid planner
  - ‘geometric’ predicates, e.g. reachable(hand, table_loc) can run a path planner

- Framework for learning actions
  - Run code block-by-block in the simulator
  - Use code annotations and/or query the simulator to get the symbolic system state before and after
  - Identify similar code blocks and group them together
  - Use probabilistic learning / satisfiability method to generalise action pre- and post-conditions
Things We Haven’t Considered

- Uncertainty
  - E.g. may drop an object rather than successfully grasp it
  - Assume object will be exactly where we expect after an action

- Partial observability
  - Assume pose of all objects is known exactly

- Assumed finite list of possible table locations

- Learning action segmentation
  - Identifying actions by clone detection methods or state-space similarities

- Learning directly from code, not via the simulator
Questions

- Any questions?
- …Or suggestions?

This work was supported by the European Community's 7th Framework Programme, FP7