Generalising Robot Manipulation Tasks

www.gert-project.eu

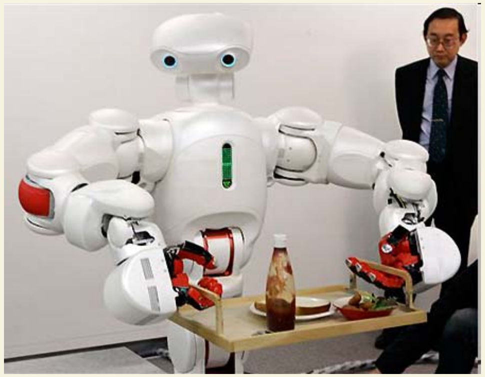
Christoph Borst (DLR – Robotics and Mechatronics Center)
23.2.2012

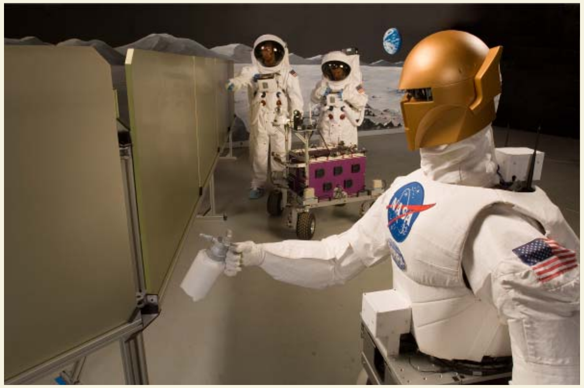
Consortium

Beneficiary Number *	Beneficiary name	Short name	Country	Main research objectives
1 (coordinator)	Deutsches Zentrum für Luft- und Raumfahrt	DLR	Germany	• Visual machine perception • Robot control • Integration and demonstration platform
2	University of Birmingham	UB	United Kingdom	• Learning of planning operators / hybrid planning • Optimising Grasping Strategies using POMDP solvers
3	Oerebro Universitet	ORU	Sweden	• Symbolic robot plans / hybrid planning • Task level planning • Geometric reasoning
4	TU Darmstadt	TUD	Germany	• Adapting grasps and grasping primitives to new objects using statistical reinforcement learning

Motivation

- Robots already perform nice manipulation tasks / demos
- Special programs for each **robot**, **task** and **object**
- Robust programs need expertise and time!

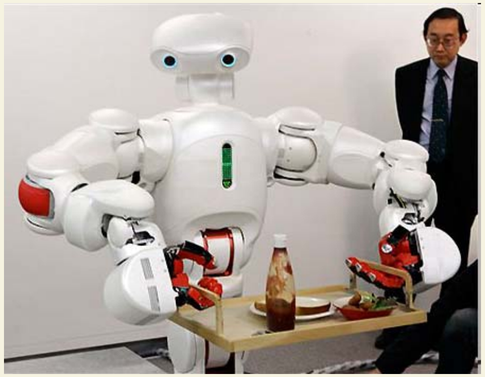
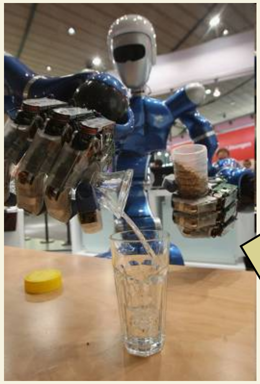


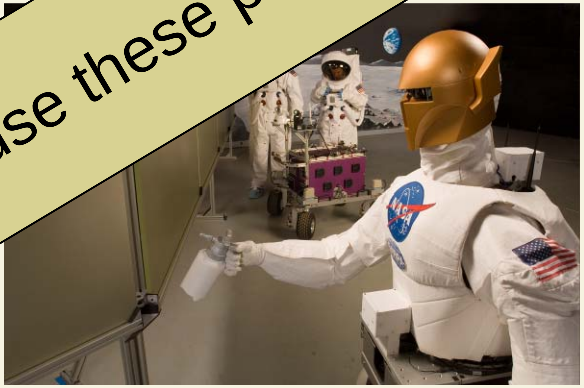



Motivation

- Robots already perform nice manipulation tasks / demos
- Special programs for each **robot**, **task** and **object**
- Robust programs need expertise and time!

Can we reuse these programs?



State of the art

- Manipulation options are currently restricted to:
  - Small set of previously known objects
    - Object recognition and localisation
    - Geometry of objects is known in advance
  - Minor changes in the scene arrangement
    - Motion planning (goal configuration reachable)
    - Offline grasp planning (take a configuration from database)
  - Simple Objects and simple tasks
    - Robustness of operation is far from suitable for daily use

Motivation

In GeRT we aim to generalise manipulation task with:

- the same task constraints

- different object geometries

- different object postions

Motivation

Goal:

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.

Approach:

- A set of demonstration programs achieving the same abstract task with different objects and varying scene arrangements is coded by hand and executed on the robotic system.
- These examples form the base for generalising the planning operators and for learning pre- and post conditions of operations.

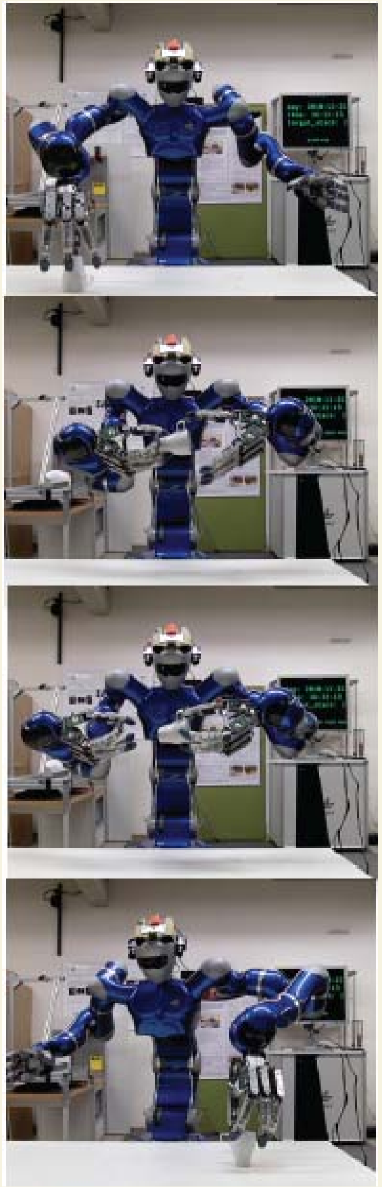
Key aspects

Planning
- Create programs for new task variations

Perception
- Determine Similarities between objects

Learning
- Adapt manipulation to new objects

Integration
- Combine all and run it on a real robot



Research Areas

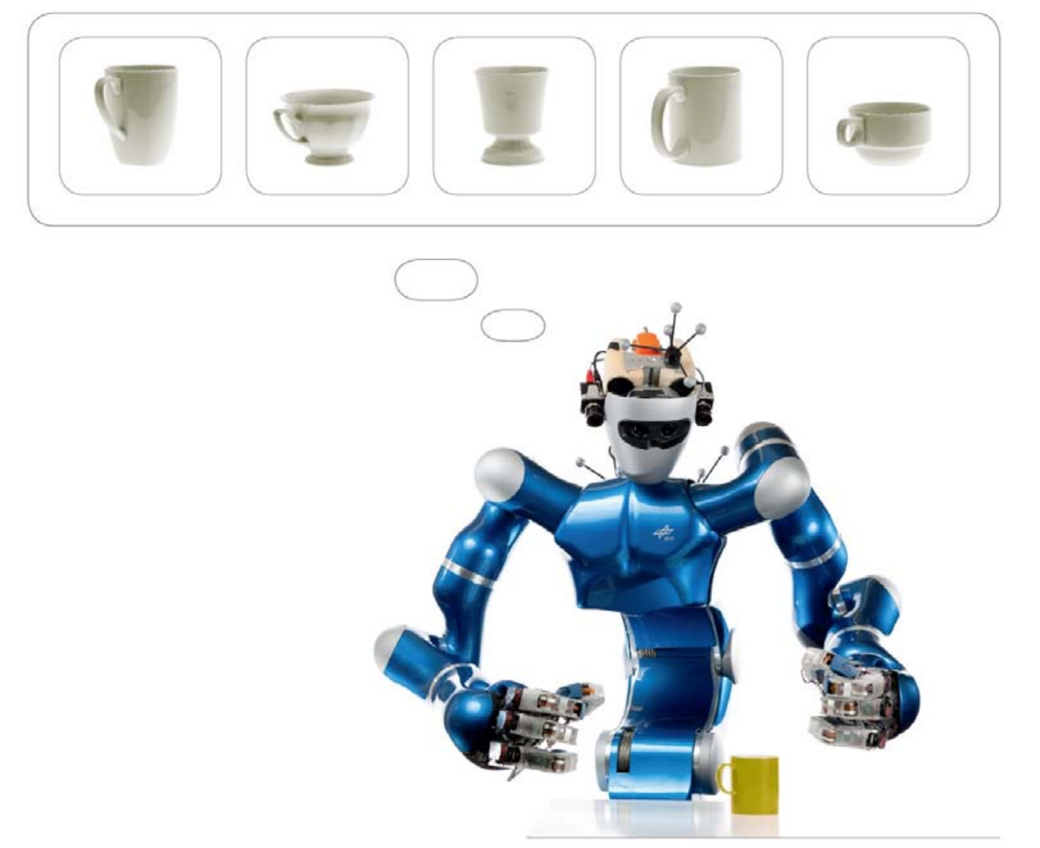
- **Hybrid planning of robotic tasks**
  - Logical planning is required to generate plans from abstract tasks
  - Planning operators are to be learned from example programs
  - Geometric states have to be accounted to generate executable plans

- **Visual perception: recognising objects, parts, affordances**
  - Functional object classes have to be recognized and labled
  - Correspondences between functional parts of known objects and new objects have to be found.

- **Learning and Optimizing Grasp Strategies**
  - Grasps from the example tasks have to be mapped and adapted to the changed geometry of new object (based on the perceptual information)
  - Reach to grasp strategies (control schemes) have to be developed that allow adaptation and preserve robustness

Integration / Example Programs

- **What are good example programs?**
- **How is the structure of example programs?**
- **How can we support parallel development/ integration?**



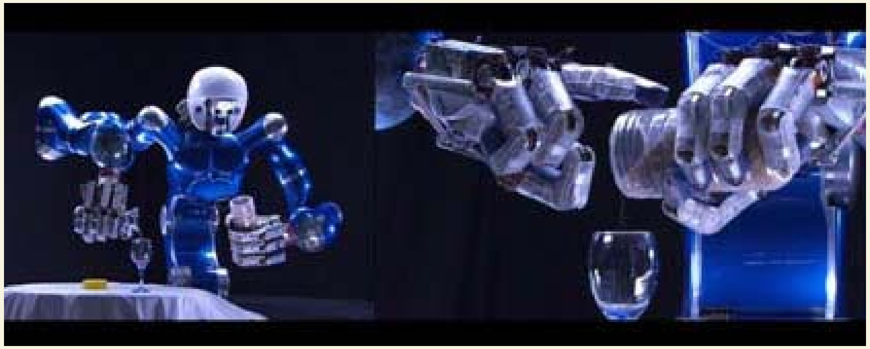
What are good example programs?

- The GeRT inspiring “Instant tea preparation”

- control program is a sequence of actions given in an XML file

- end of interpolation gives stop condition

- through moderate stiffness trajectories are **implicitly** adapted.



The tea preparation

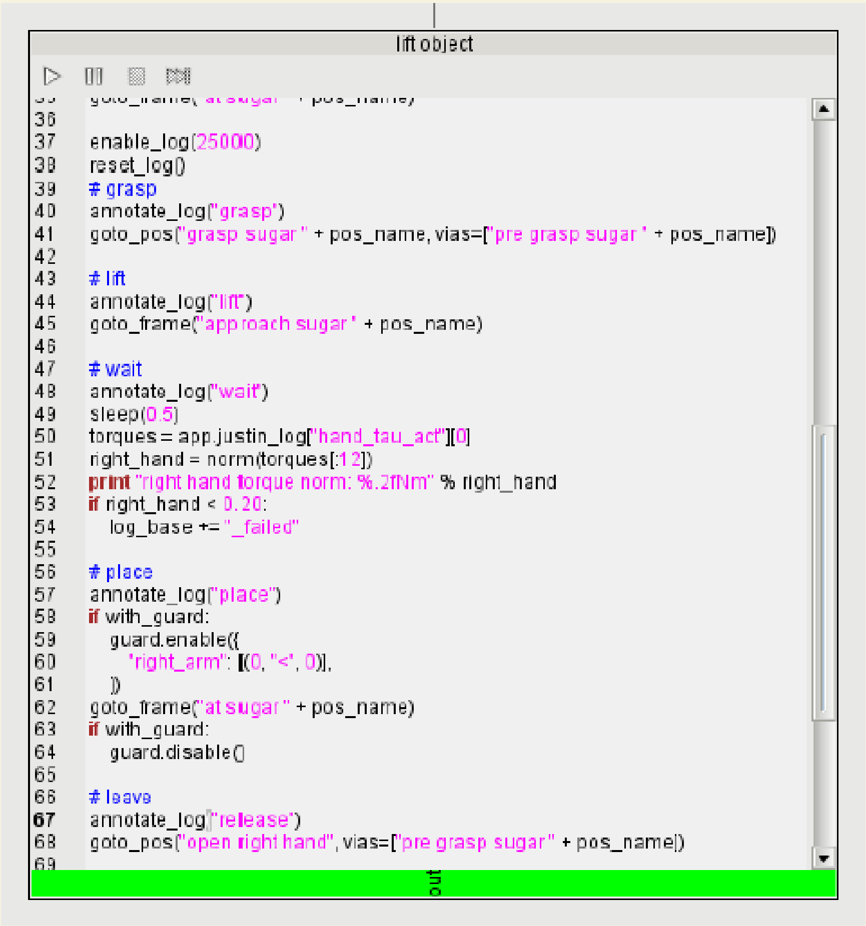
Good example programs?

Control programs need:

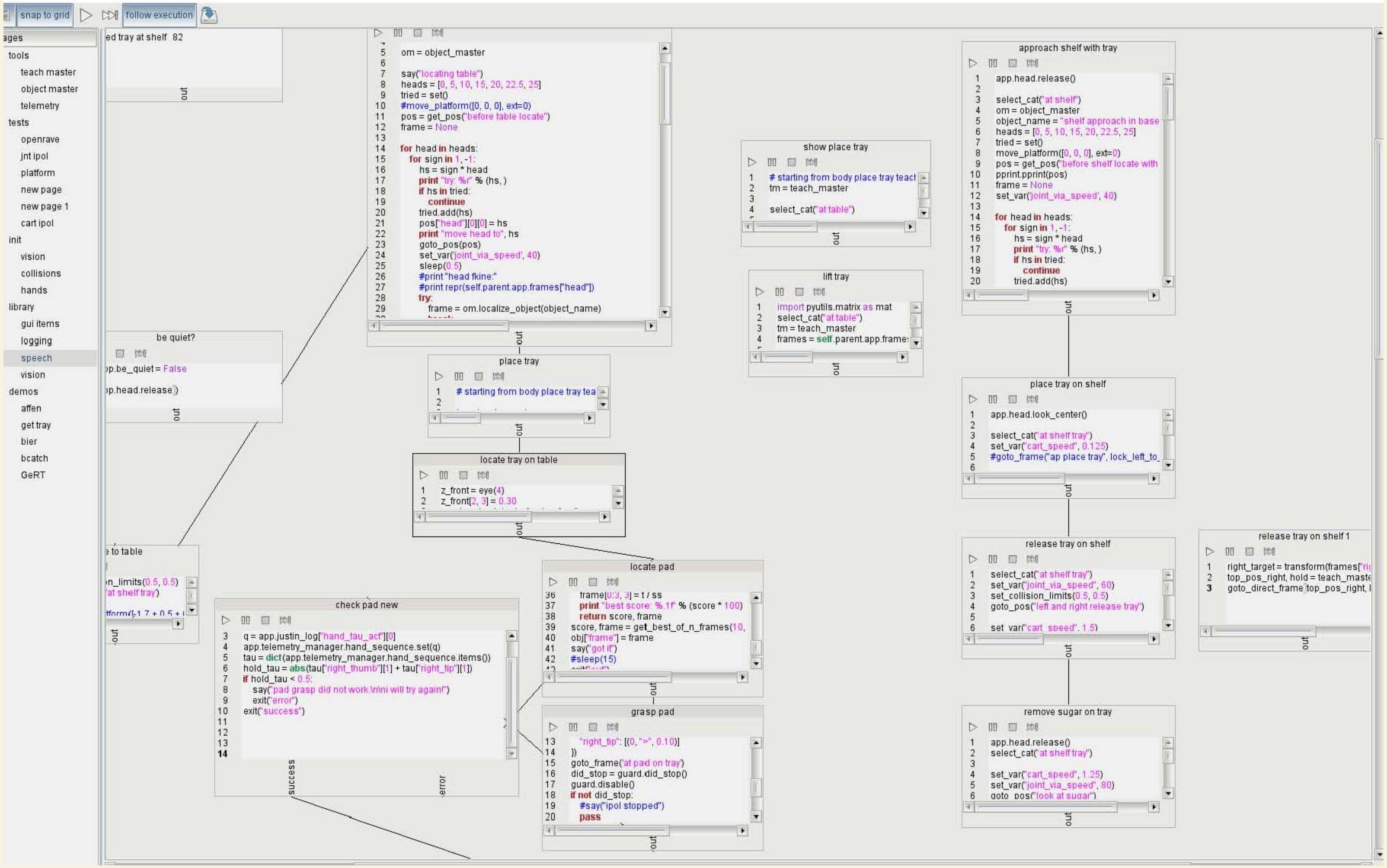
- stop conditions based on sensor input
- control structures like if-clauses, loops, switch-statements
- state-machines to control movements
- to be online interpreted to be fed by hybrid planners

The structure of example programs

- GUI code displays python source (pyrs-source)
- Script Server code provides a general set of robot commands (macros)
- pyrs-code runs in GUI process, can be single stepped

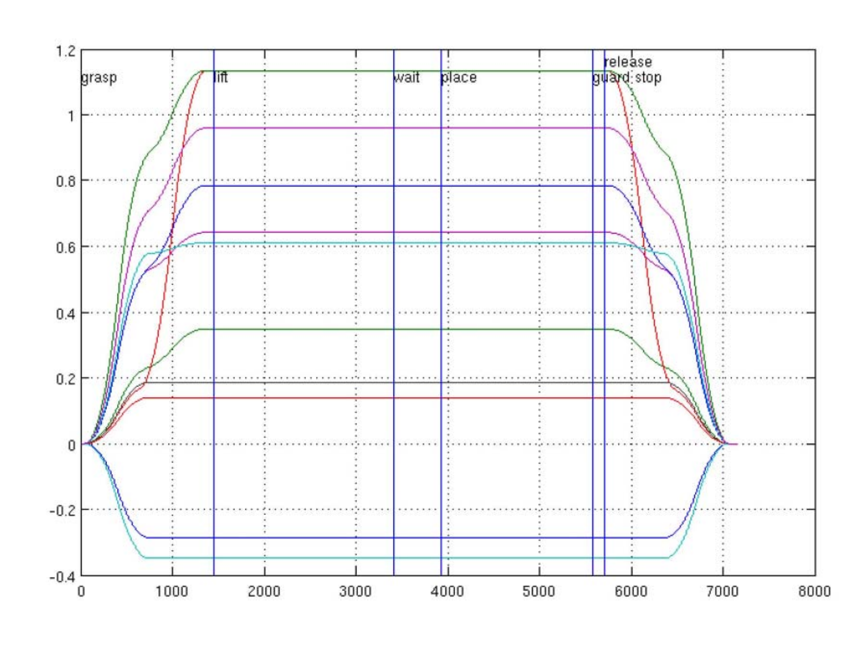


The structure of example programs

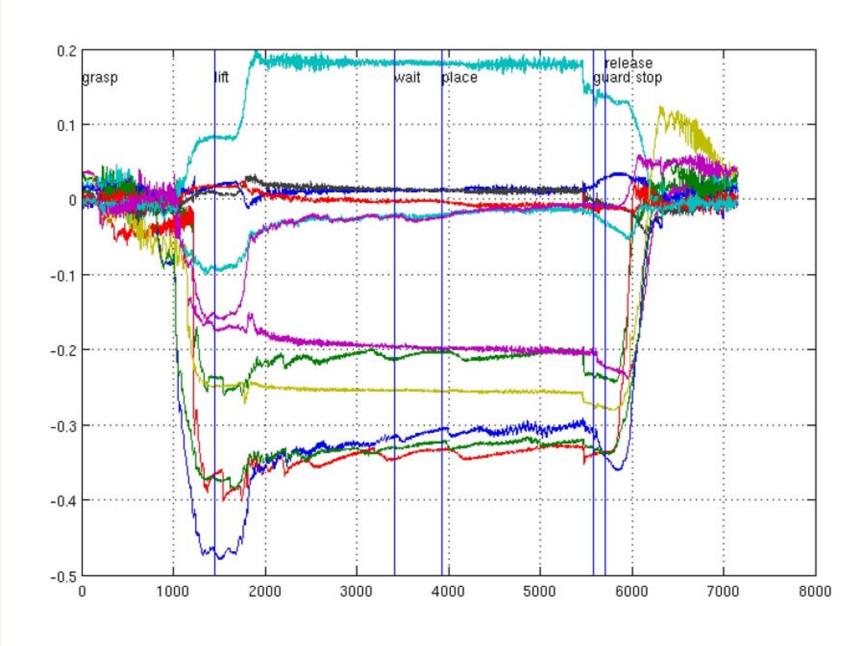


Example: Stacking cups

Example: Stacking cups / sensor log



commanded joint positions



measured joint torques

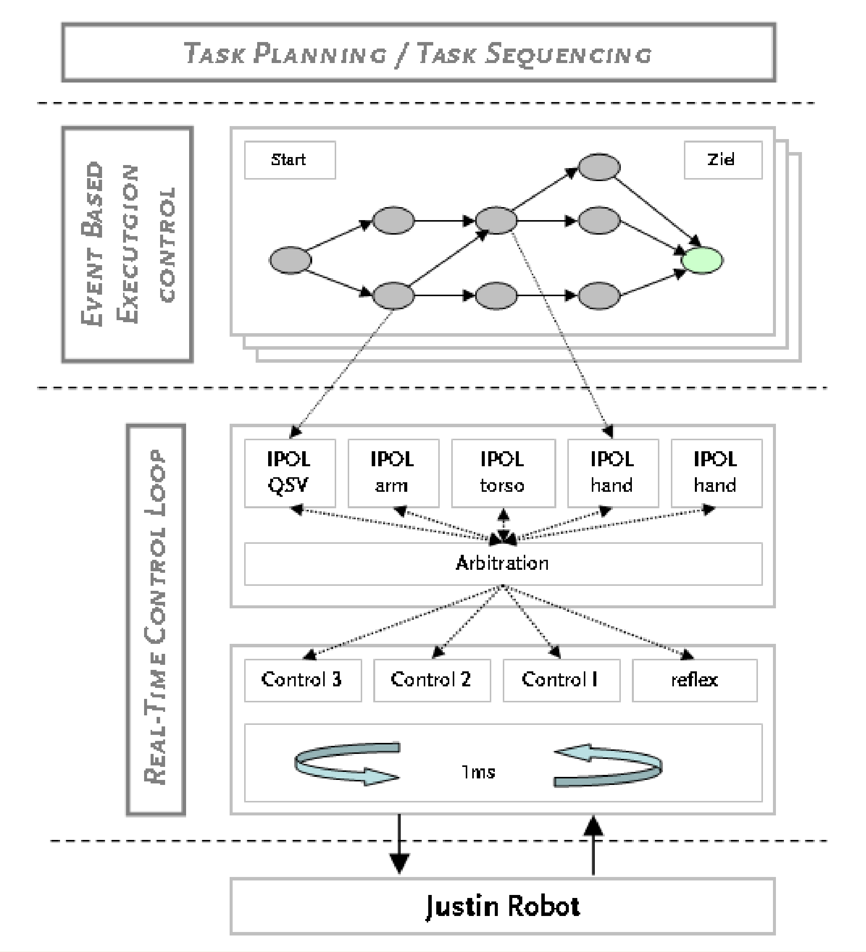
Sensor logs are annotated in relation to the example programs

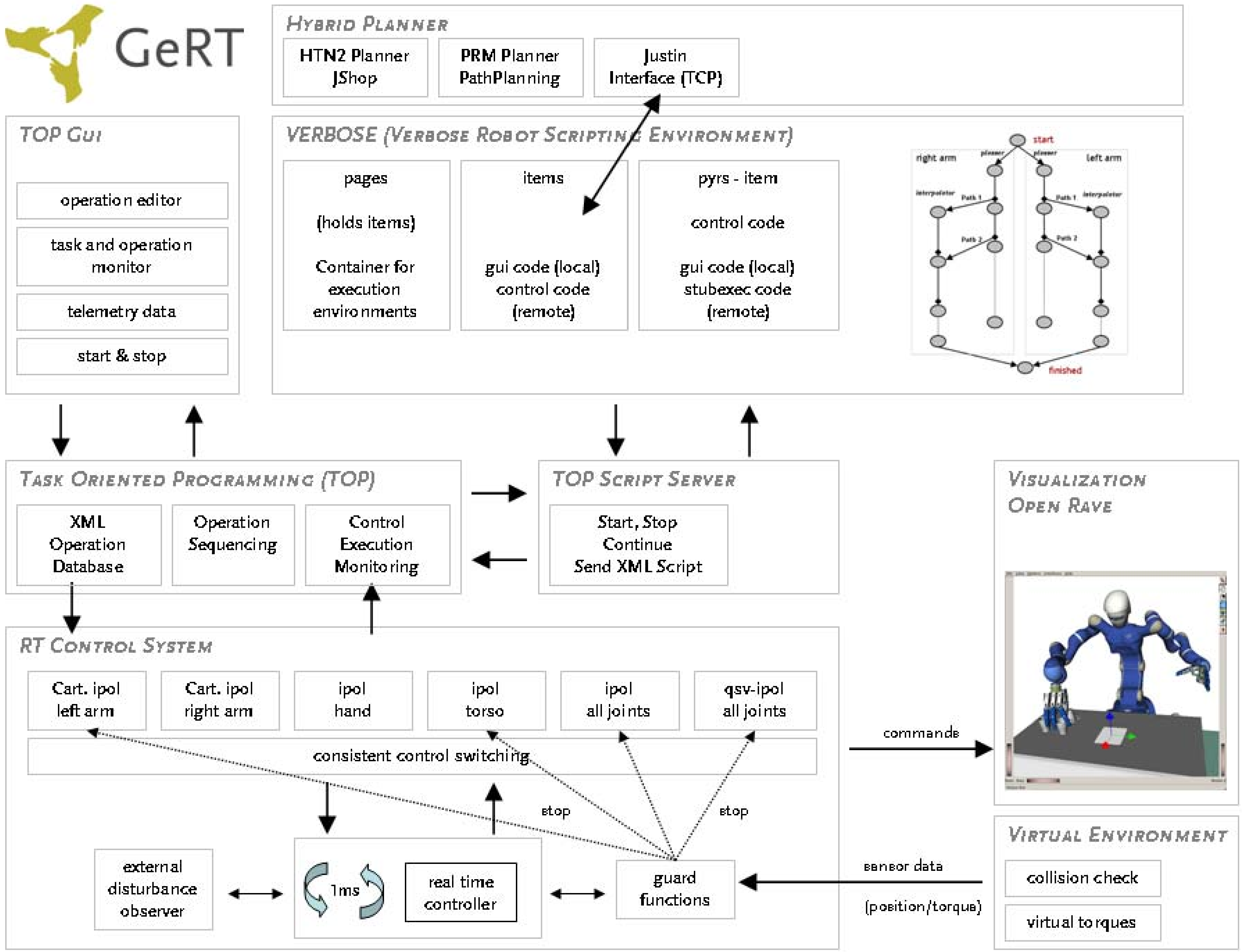
Justin‘s execution environment

- Clear separation of real-time feedback control loop, event based execution control and higher planning levels

- Immediate response on external disturbances through real-time guards.

- Compliant control modes of hands, arms and torso with variable stiffness and damping



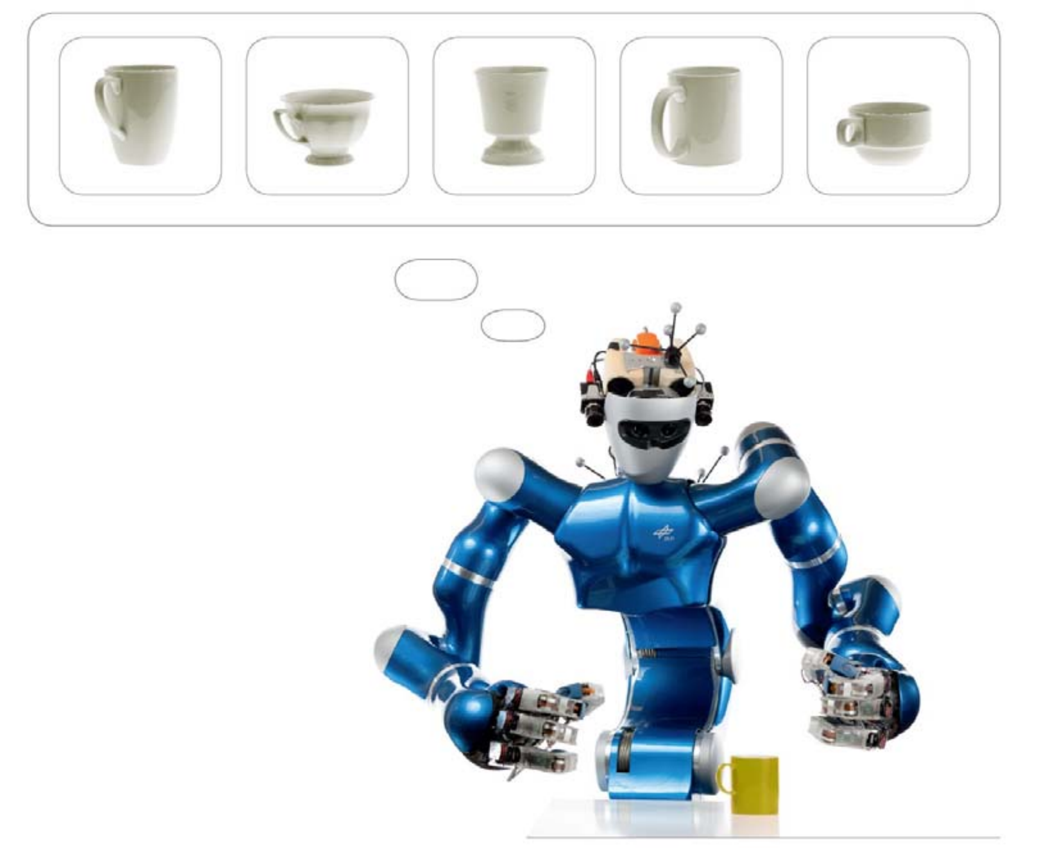


GeRT
Hybrid Planner
HTN2 Planner
JShop
PRM Planner
PathPlanning
Justin
Interface (TCP)
TOP Gui
operation editor
task and operation
monitor
telemetry data
start & stop
VERBOSE (Verbose Robot Scripting Environment)
pages
(holds items)
Container for
execution
environments
items
gui code (local)
control code
(remote)
pyrs - item
control code
gui code (local)
stubexec code
(remote)
start
right arm
planner
planner
left arm
interpolator
Path 1
Path 1
interpolator
Path 2
Path 2
finished
Task Oriented Programming (TOP)
XML
Operation
Database
Operation
Sequencing
Control
Execution
Monitoring
TOP Script Server
Start, Stop
Continue
Send XML Script
Visualization
Open Rave
RT Control System
Cart. ipol
left arm
Cart. ipol
right arm
ipol
hand
ipol
torso
ipol
all joints
qsv-ipol
all joints
consistent control switching
commands
stop
stop
external
disturbance
observer
1ms
real time
controller
guard
functions
sensor data
(position/torque)
Virtual Environment
collision check
virtual torques

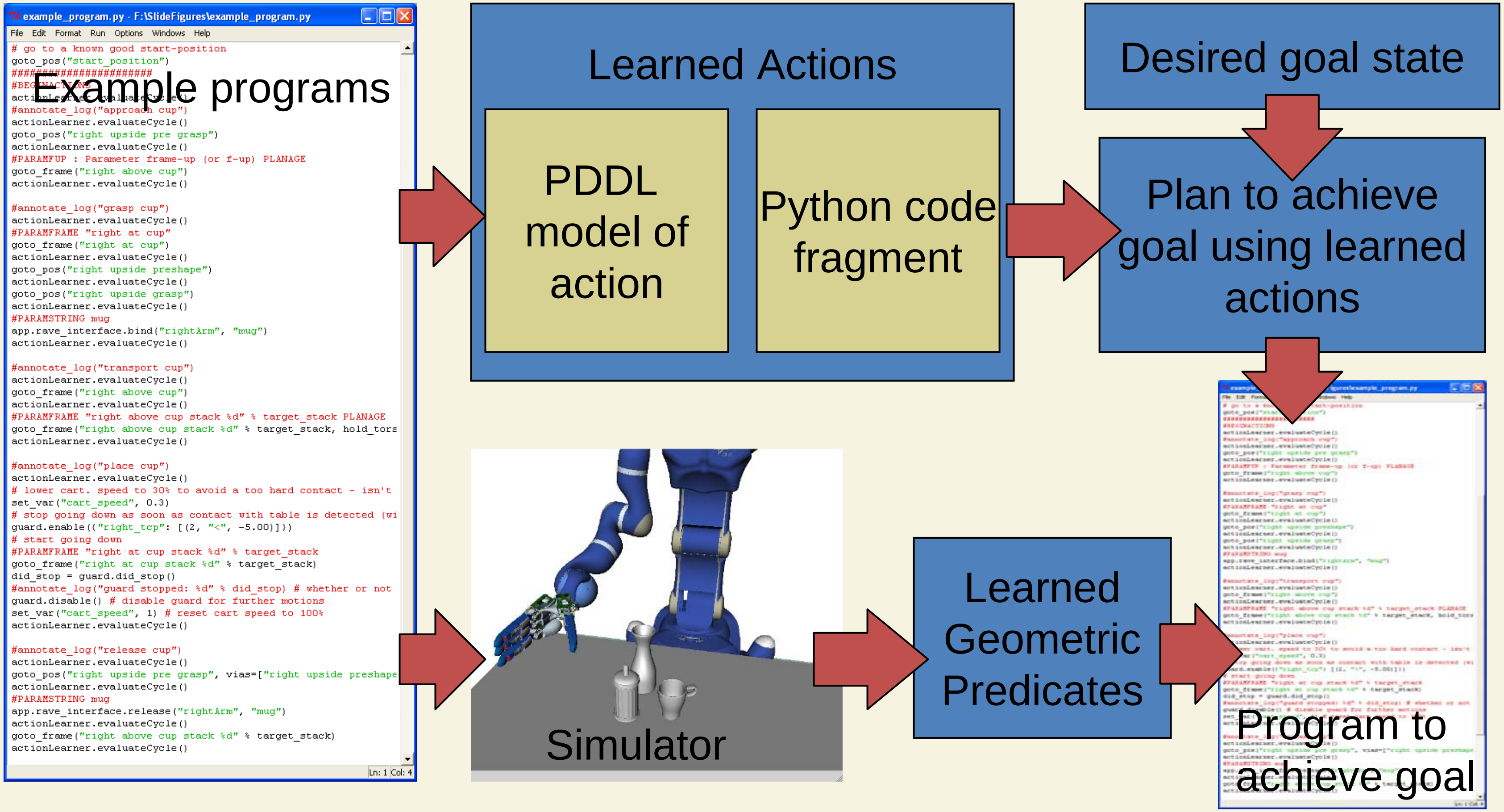
Action generalisation / hybrid planning

- **How can we analyse / generalise the examples?**

- **How are the general actions used to replan the same semantic task?**

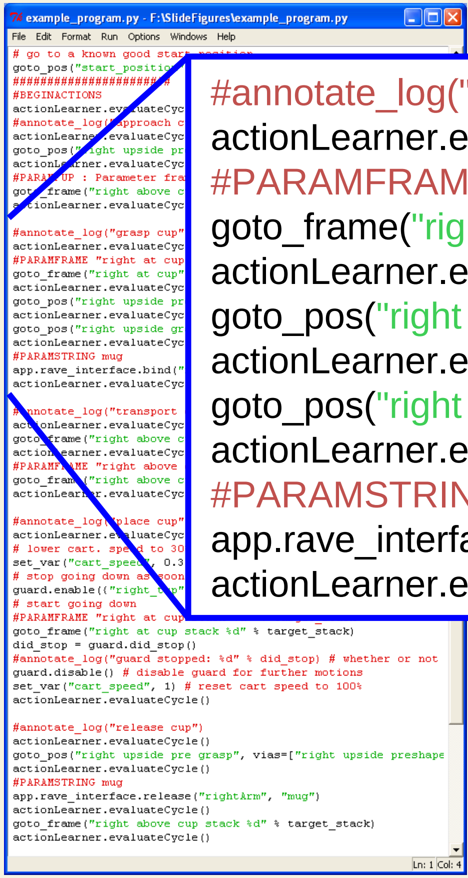


Action learning overview



Need to learn action models, but also mapping between geometric predicates (“on-top-of”, “touching”, …) and geometric states.

Action learning




```
#annotate_log("grasp cup")
actionLearner.evaluateCycle()
#PARAMFRAME "right at cup"
goto_frame("right at cup")
actionLearner.evaluateCycle()
goto_pos("right upside preshape")
actionLearner.evaluateCycle()
goto_pos("right upside grasp")
actionLearner.evaluateCycle()
#PARAMSTRING mug
app.rave_interface.bind("rightArm", "mug")
actionLearner.evaluateCycle()
```

PDDL Representation of Action

```
(:action learned_act_1 :parameters ( ?param0 )
    :precondition (and
        ( forall (?ALL)( not (above ?ALL ?param0)))
        (right-hand-empty)
        (left-hand-empty)
        (object ?param0)
    )
    :effect (and
        (right-can-grasp ?param0)
        ( forall (?ALL)( not (right-can-grasp ?ALL)))
    )
```

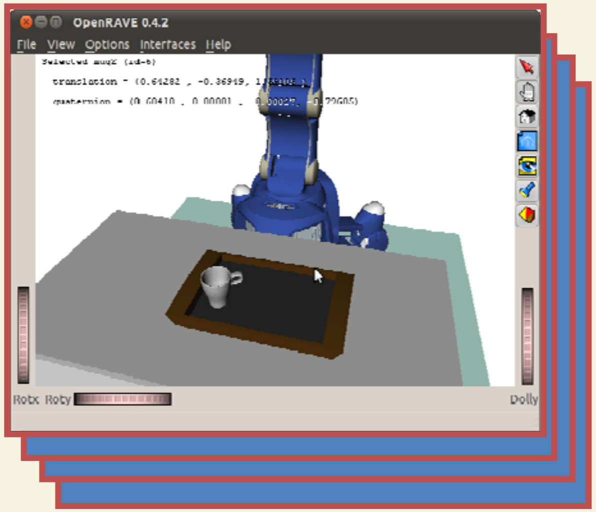
Unnecessary precondition

- Planning domain definition language action representation learned from annotated code
  - Annotations split up actions, tell us what the parameters of the actions are
  - Preconditions and effects learned by generalising over state before and after code is executed

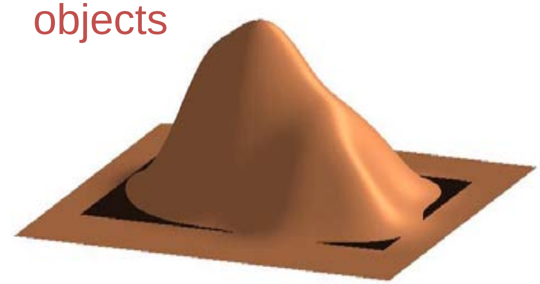
Geometric Predicate Learning

- As well as actions, we need to learn predicates for the planning domain and their corresponding geometric states
  - Do this using training data that could be generated from the example programs
    - At present we augment this with hand-generated data

Labelled Training Examples



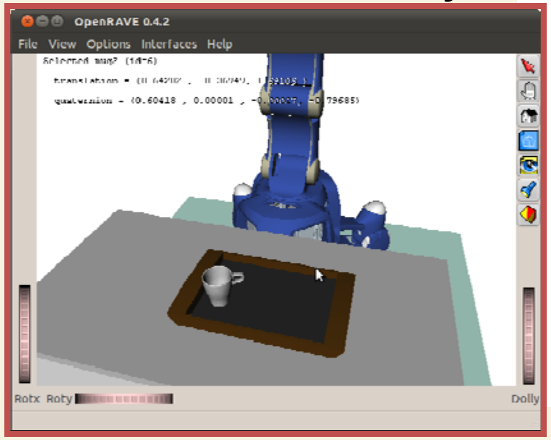
Learned geometric relation
- Mixture of Gaussians model
- 17 dimensional model of relationship between two objects

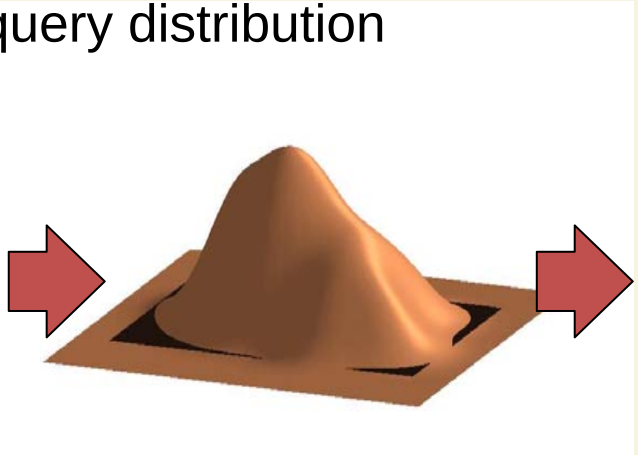


P(Above(cup,tray) = true). Usable tray shown in black (two dimensions shown).

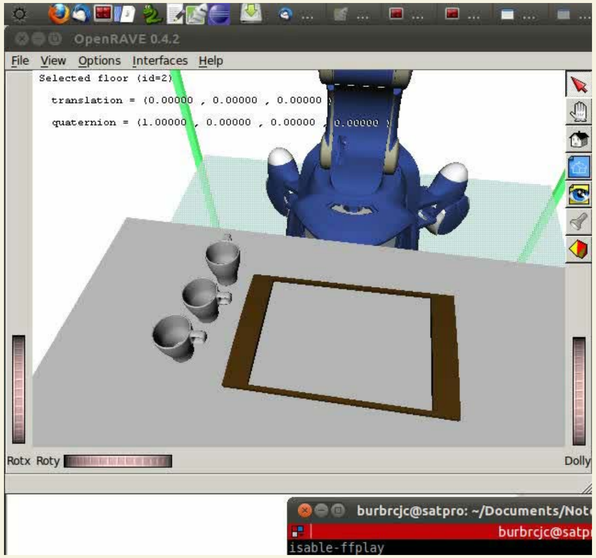
Using the learned model

Forward direction: just query distribution

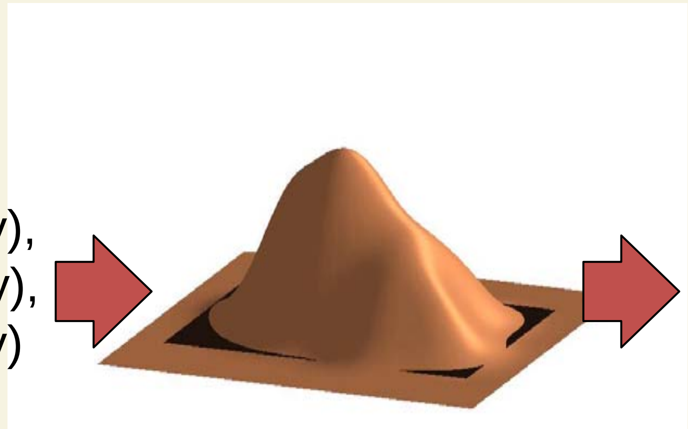




P(on(cup,tray) = true) = 0.95



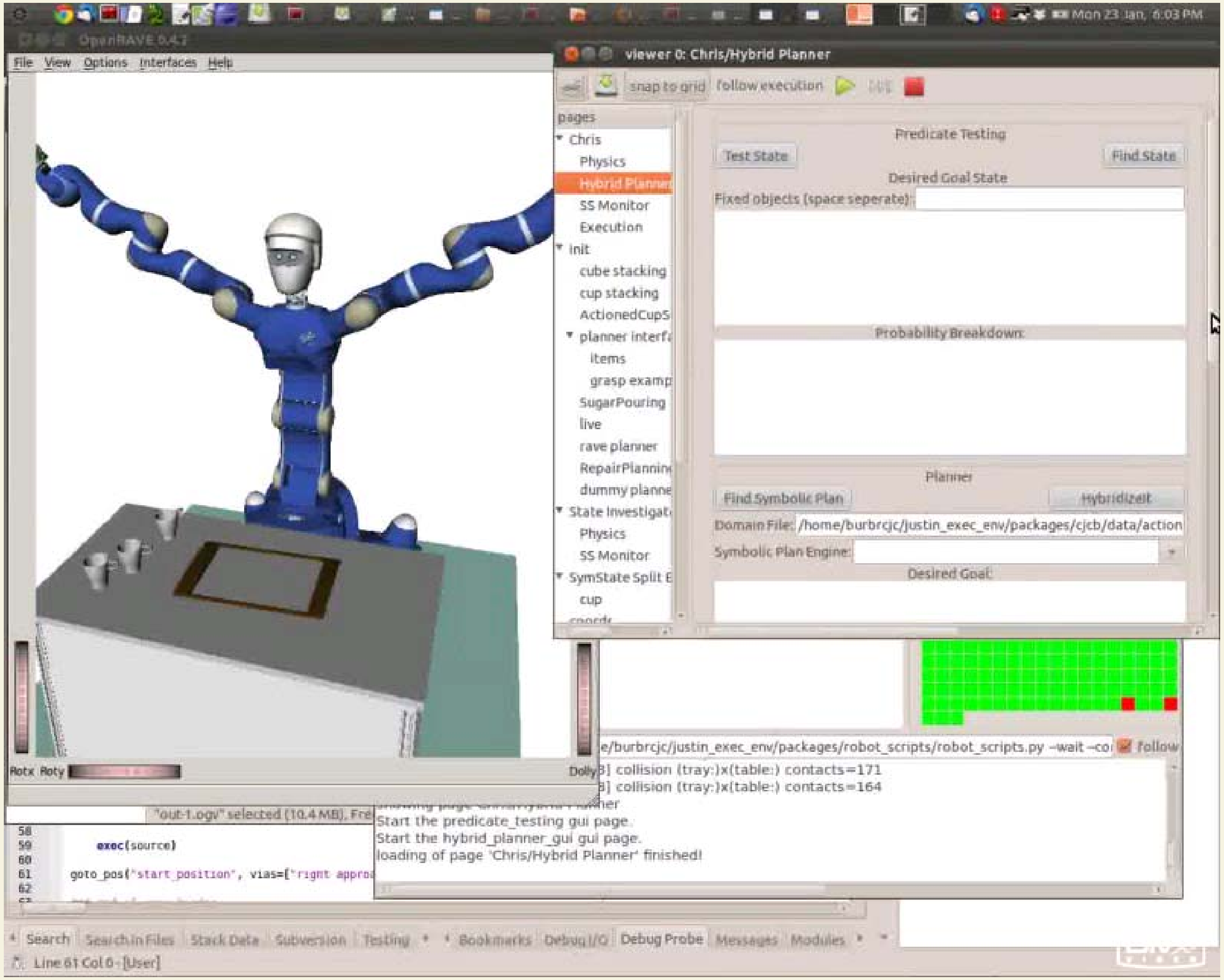
On(cup1,tray),
On(cup2(tray),
On(cup3,tray)



Backward direction: hill-climbing search in probability space

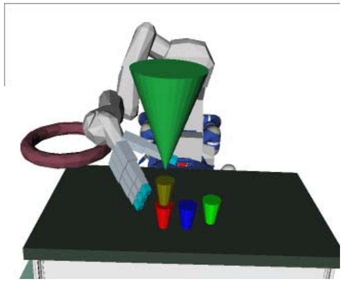
Putting it all together

1. Goal specified to planner
2. Planner generates symbolic plan
3. Learned predicates used to get geometry
4. Code generated and executed

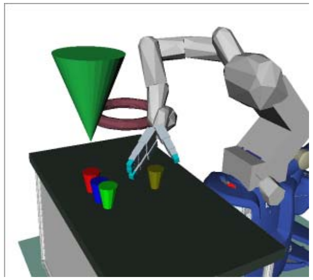


Verfying the plan through hybrid planner

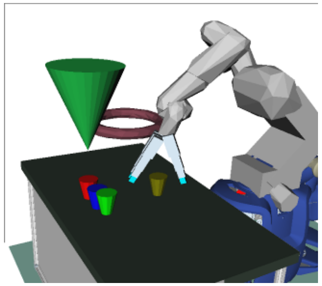
- The symbolic part of the plan can be generated
- Geometric predicates and actions are learned
- Geometric state is evaluated trough RRT planner



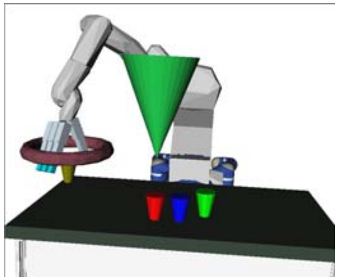
PICK Cup4



PLACE Cup4 ON Table



PICK Cup4

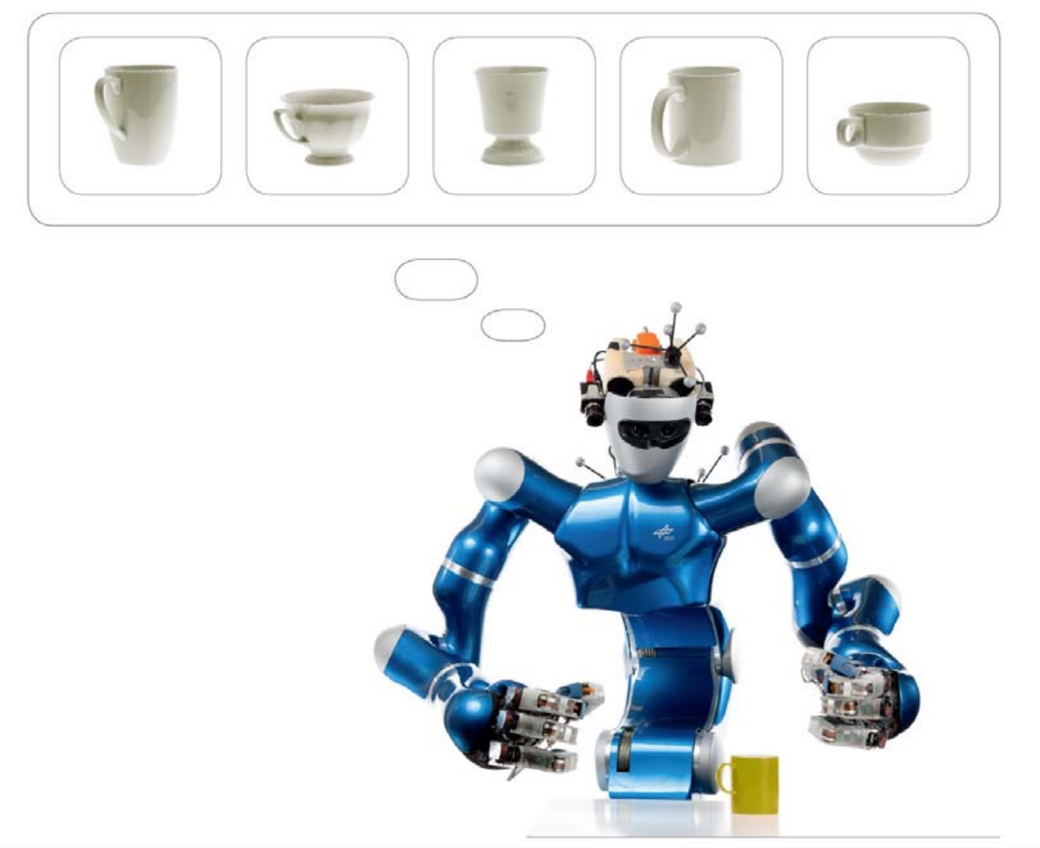


PLACE Cup4 ON Torus

Perception

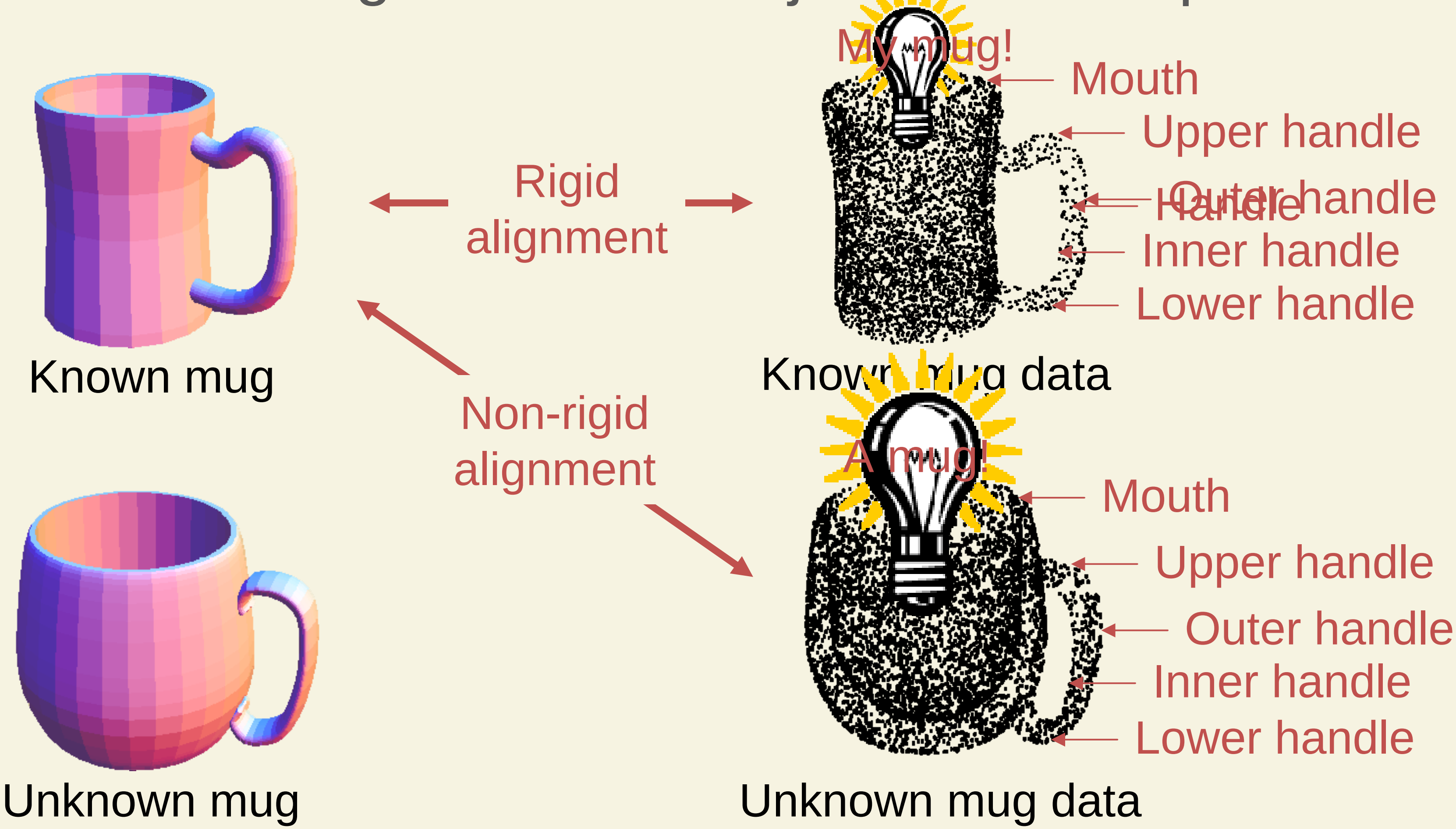
- What are the objects of the same functional object class? **(Shape Similarity)**

- How correspond functional geometric parts of known and unknown objects? **(Shape Warping)**



Perception in GeRT:

Understanding Unknown Objects for Manipulation

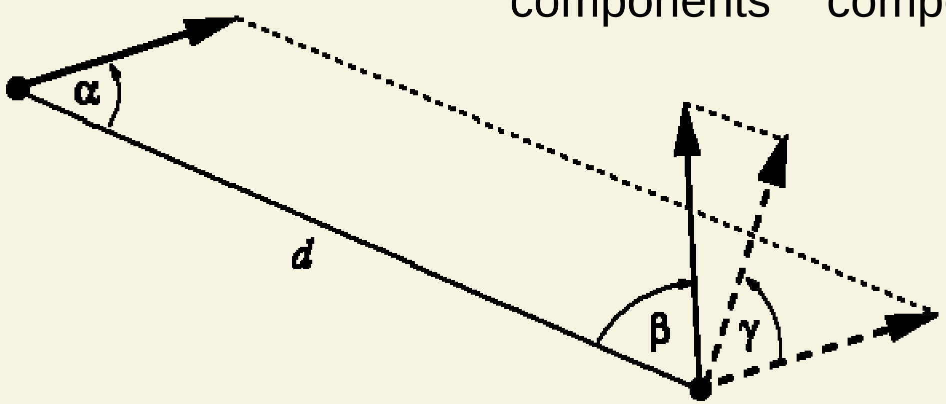


Shape Similarity: Geometric Feature

For query and model shapes independently:

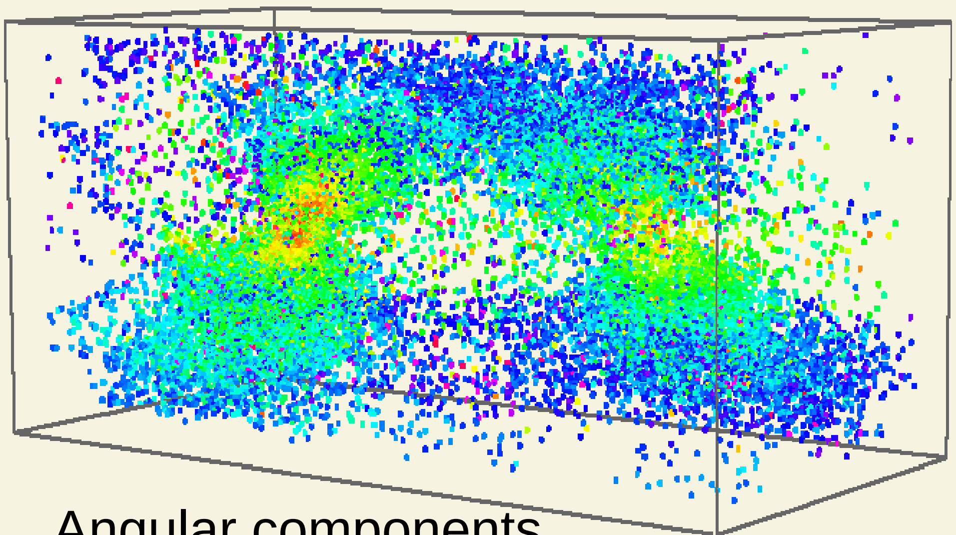
- drawing of random *surflet pairs*
- computation of intrinsic geometric parameters of each surflet pair:

$$(\underbrace{\alpha,\beta,\gamma}_{\text{Angular components}},\underbrace{d}_{\text{Distance component}})\in\mathbf{R}^4$$

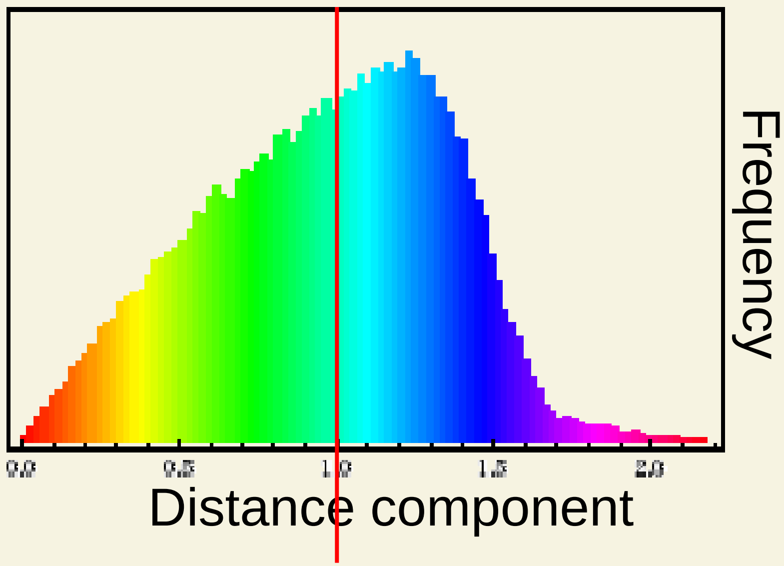


Shape Similarity:
4D Feature Density

For each query/model view: random sampling of 100,000 surflet pair relations



Angular components



Distance component

Median distance
scaled to unity

Shape Retrieval: Example

Query: Mug

20 most similar models from the database

Shape Retrieval: Example

Query: Quadruped

20 most similar models from the database

Shape Warping

Given a *source shape (model)* and a *target shape (scene)* from the same functional category…

Four stages of warping:

Initial alignment brings corresponding parts close to each other – but *without knowing or computing* correspondences

1. Deformation-tolerant alignment
2. Correspondence assignment (forward and backward)
3. Map interpolation (forward and backward)
4. Enforcing forward/backward consistency

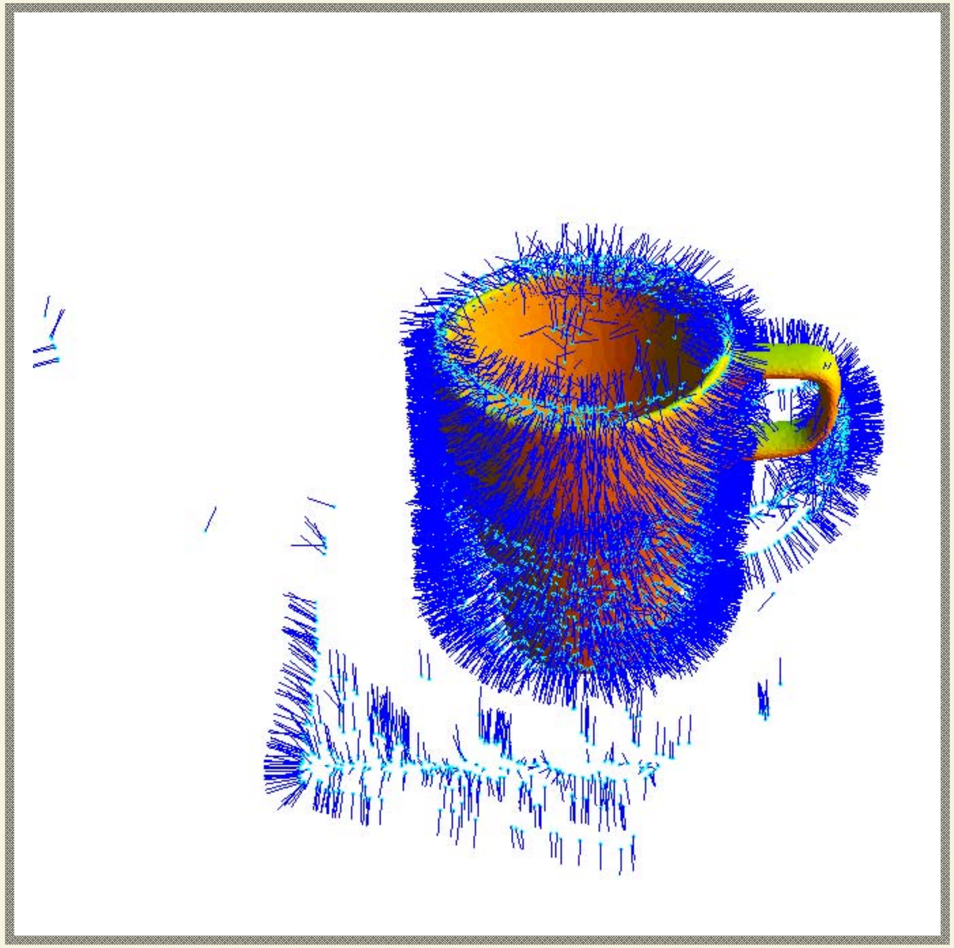
The rest is simply based on proximity of surface points and surface orientation

Hillenbrand; ICPR 2010

Shape Warping: Alignment

Source

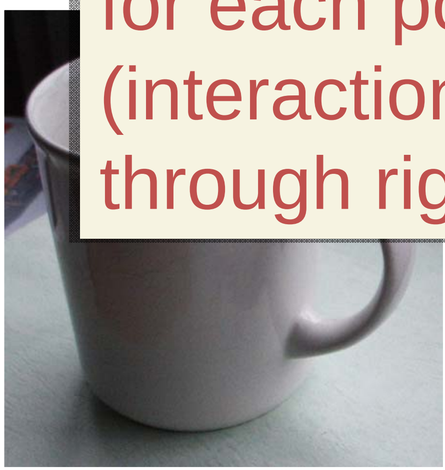
Target

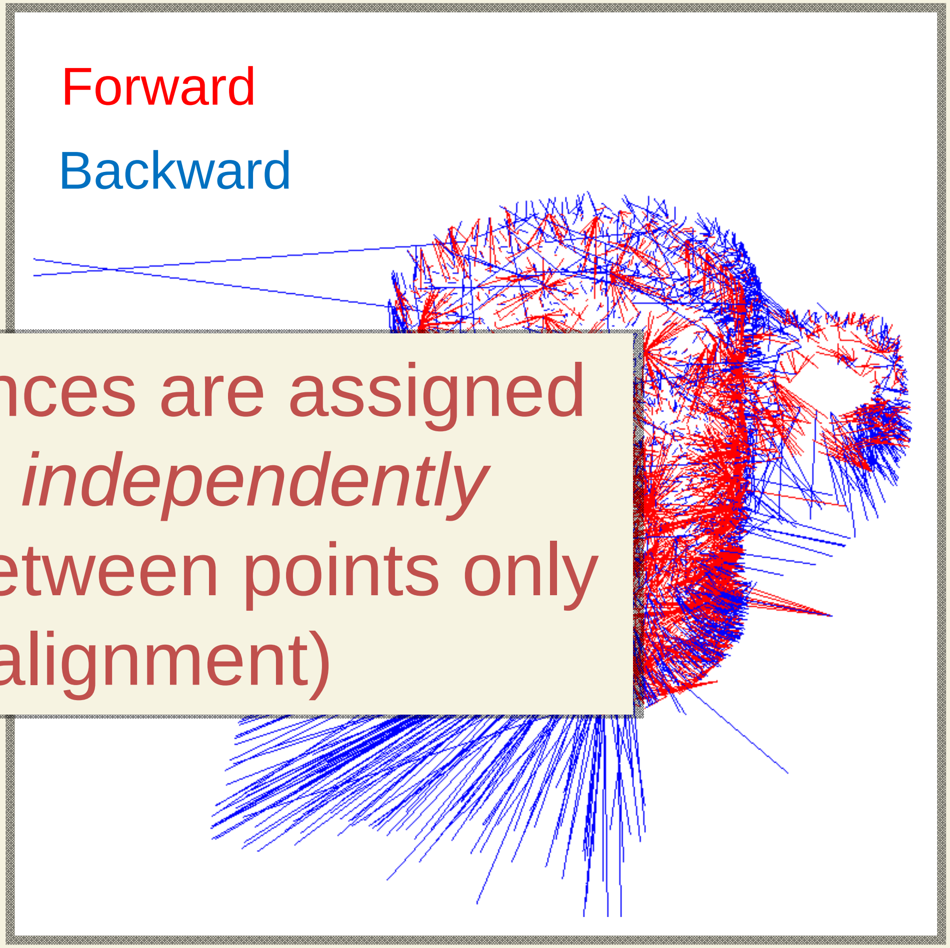


Shape Warping: Correspondence

Source

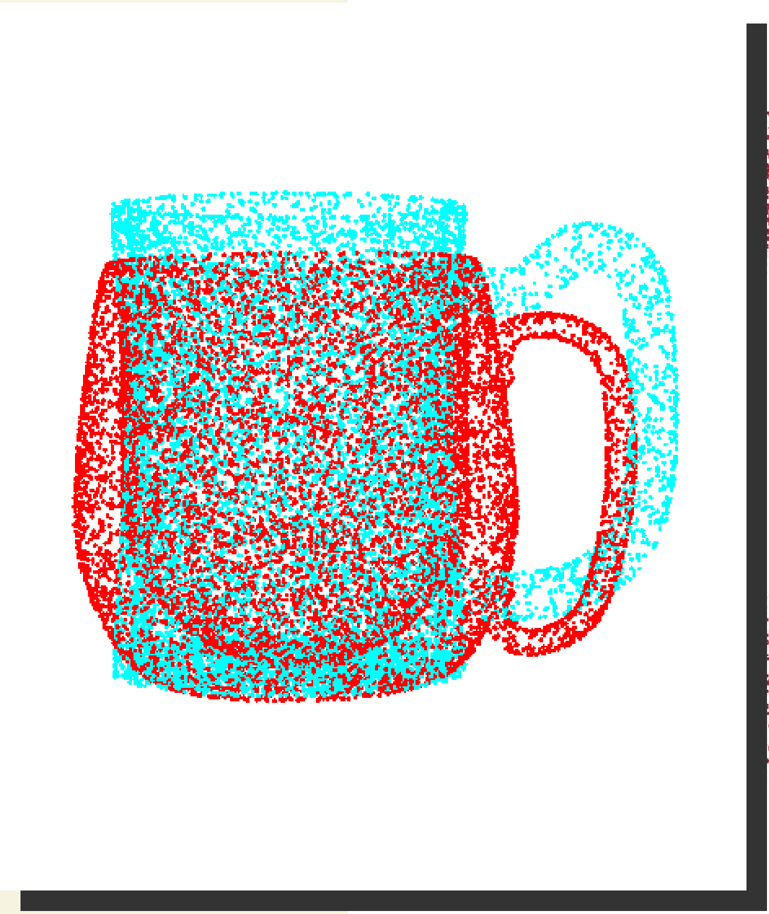
Target



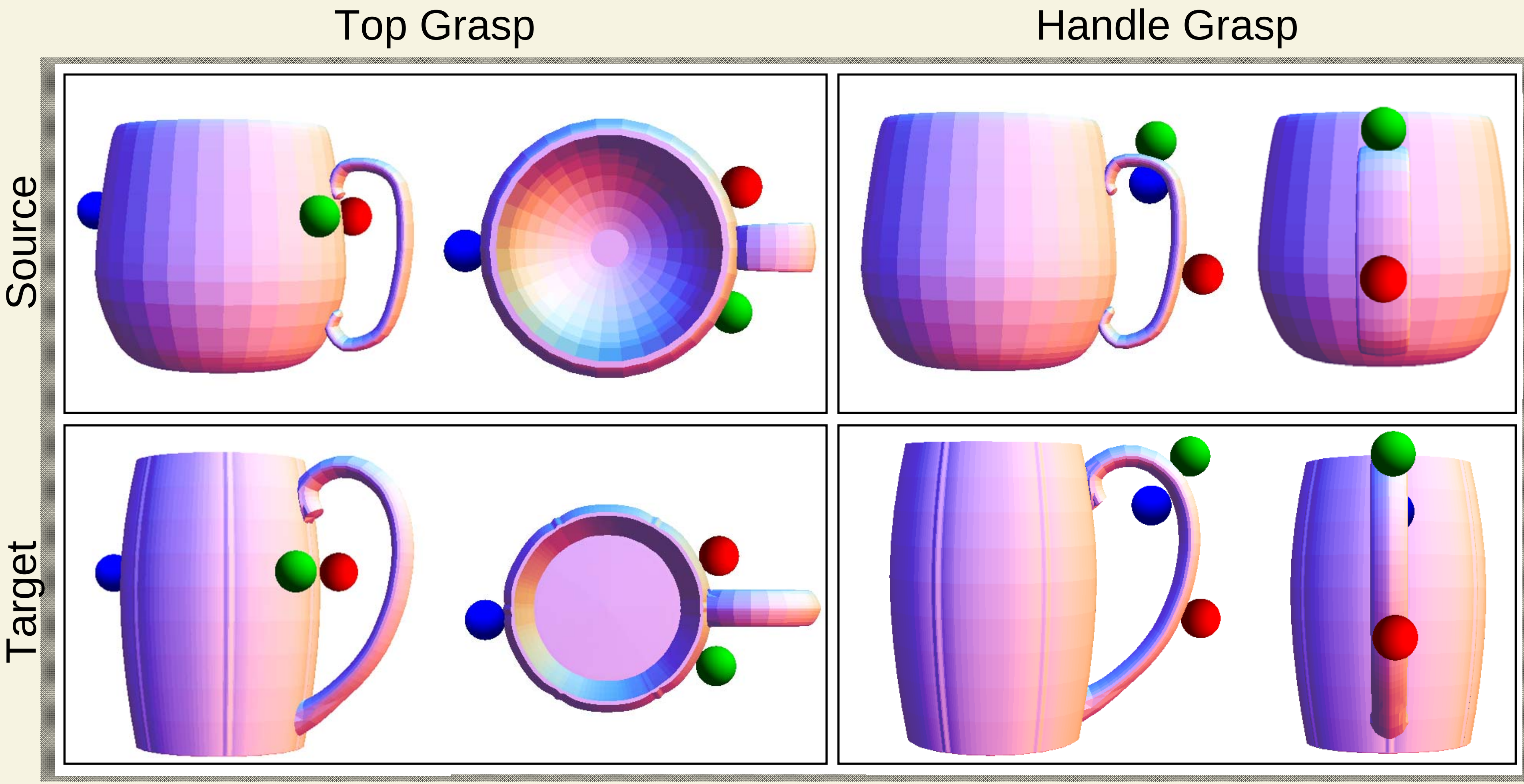


Correspondences are assigned for each point *independently* (interaction between points only through rigid alignment)

Shape Warping: Point-Wise

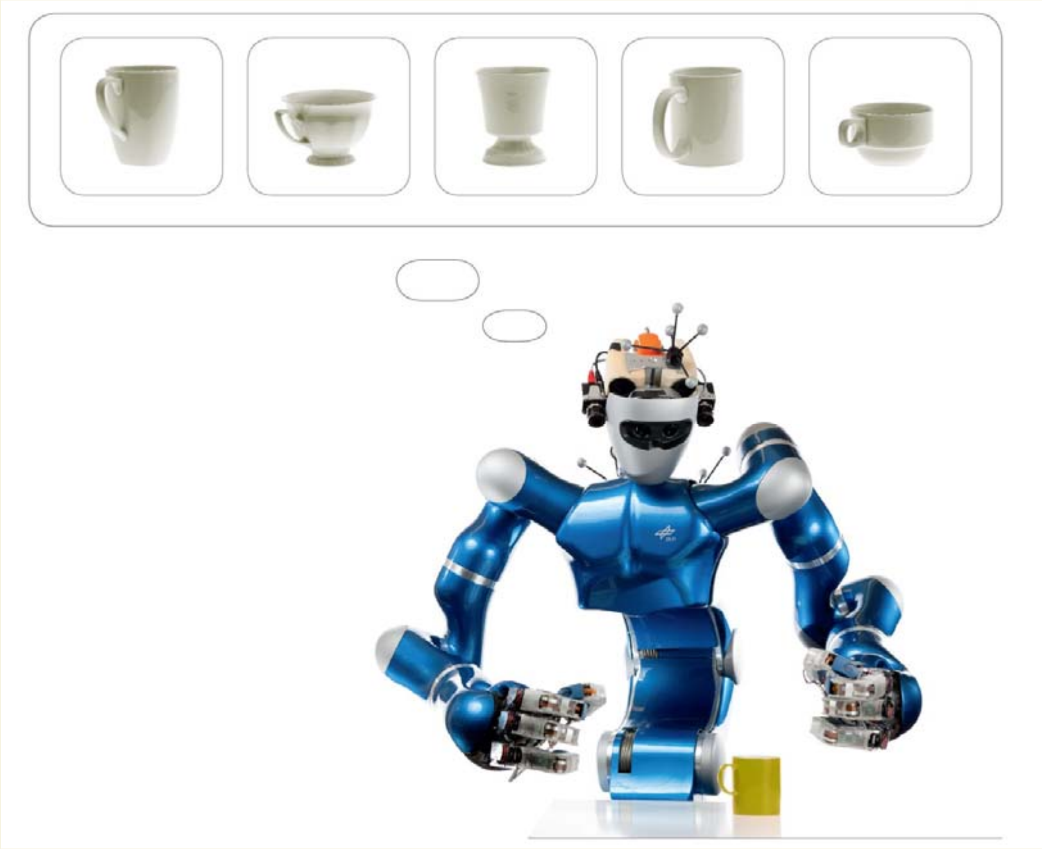


Shape Warping: Grasp Contacts



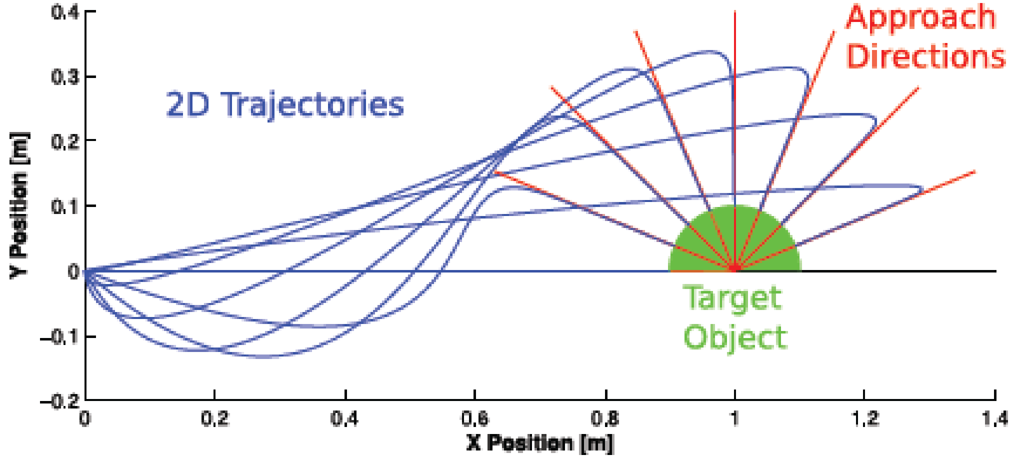
Learning grasping strategies

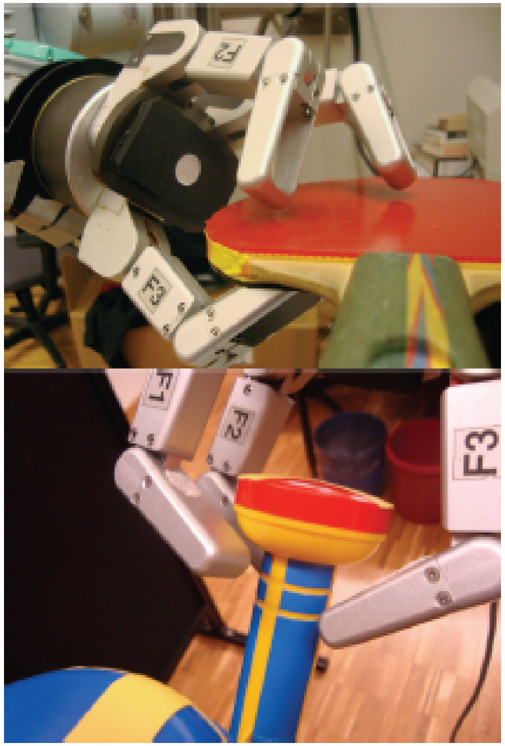
- Adapting example grasps to new objects
- Make use of human example grasps
- Learn a robust scheme to execute finger strategies



Grasping with Motor Primitives

- Flexible representation for entire grasping action
  - Obtained from human demonstration
  - Automatically adapt to new grasps





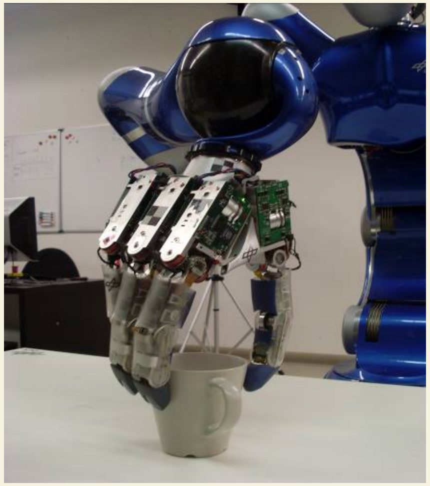
Grasp Adaptation

- Generalization of grasps to:
  - New objects
  - New positions and orientations
  - Different grasp types
- 12 Grasp types from our taxonomy
- Grasp types are learned by imitation

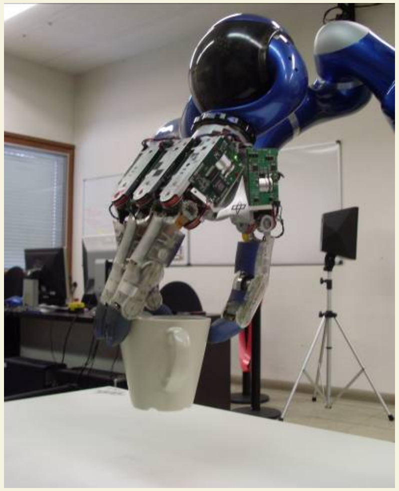

Lateral


Spherical


Tripod

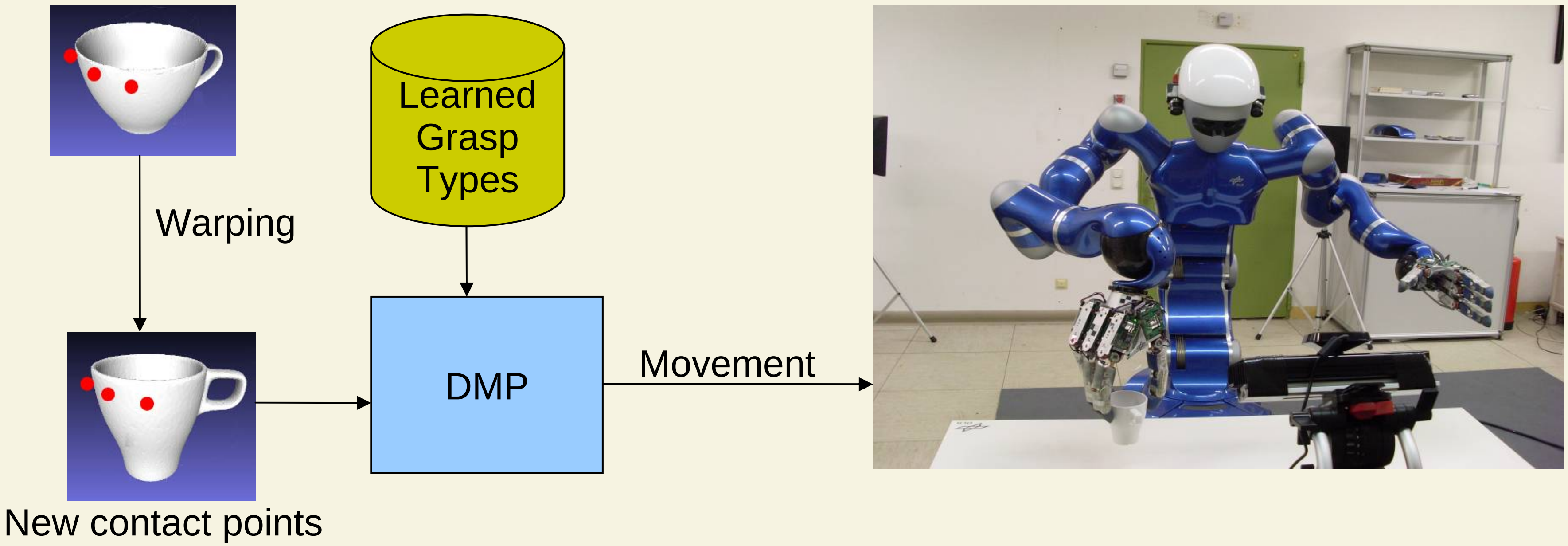

Lateral


Spherical


Tripod

Imitation by Dynamic Motor Primitives

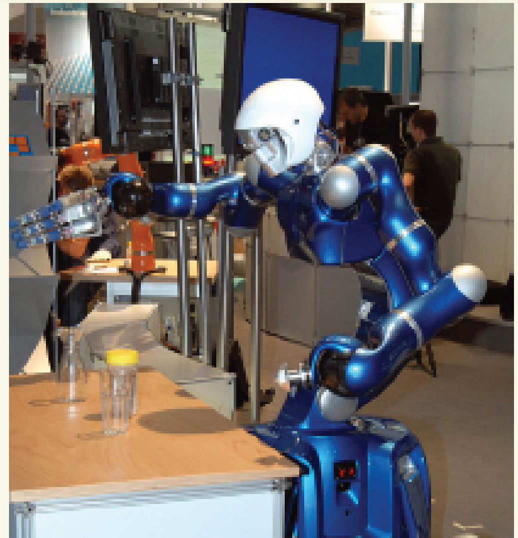
- Based on Dynamical Systems
- Used to generate Trajectories of arm and hand
- Parametrized by warps of contact points
- Easily generalized to new goal states



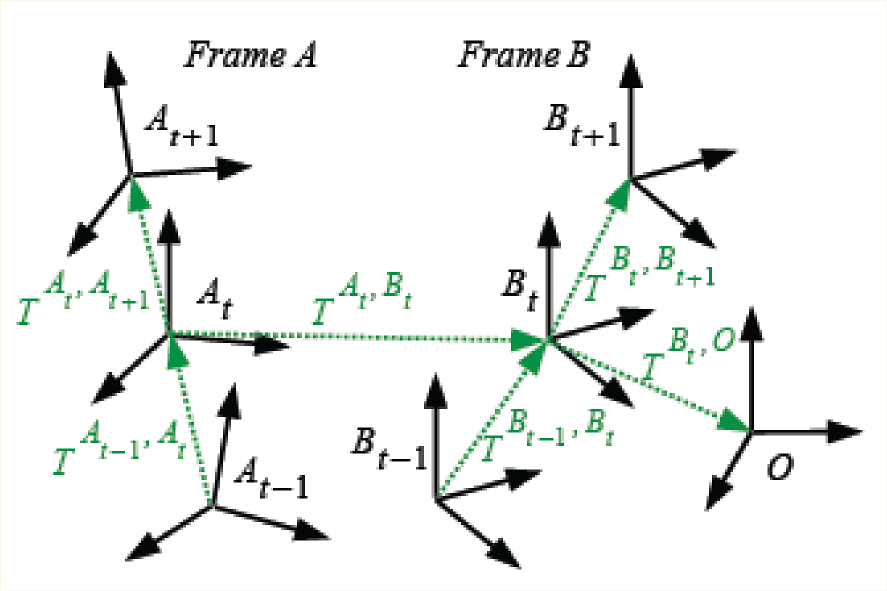
Results of last weeks integration week

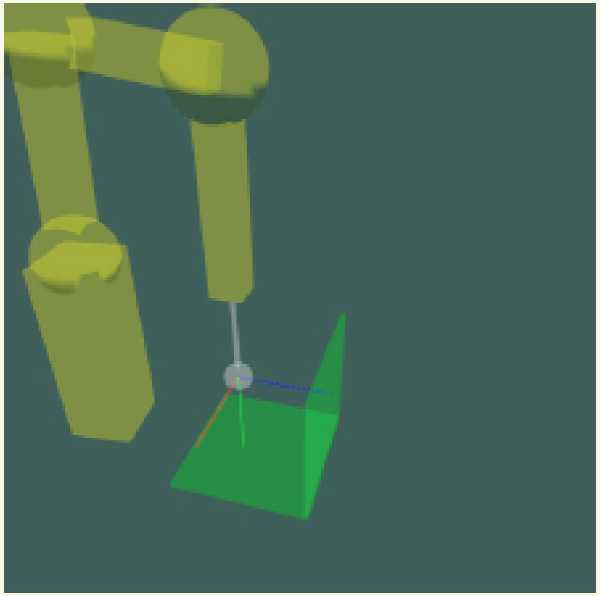
Learning forward models for grasping

- In GeRT we want to generalise from example programs
- Use these to gather data to learn forward models
- Use these to plan actions for novel tasks or objects
- Start with single finger, then multi-finger grasps, in hand manipulation



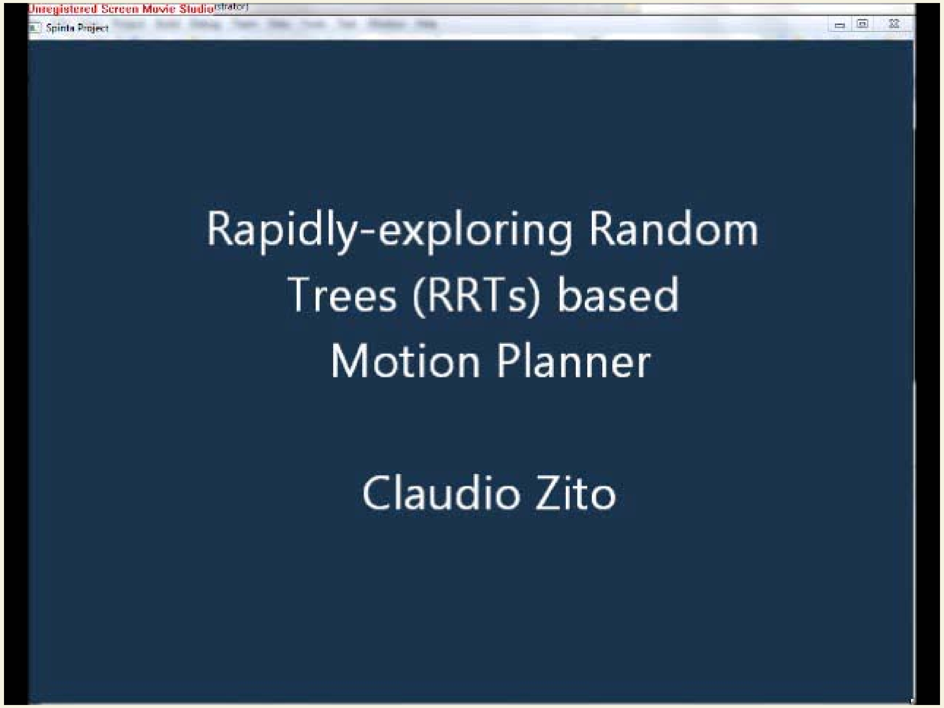
Prediction learning for single contacts





- Robot effector as some frame A
- Object it is pushing as frame B
- Prediction is saying what $B_{t+1:t+N}$ is going to be given $A_{t-1:t+N-1}$, B_t and B_{t-1} (i.e. prediction for N steps into the future)

Prediction learning & planning with prediction models



Next steps:

- Multi-finger manipulation prediction
- Repairing plans that have not been successful
- Adding contact and force information from sensor logs (DMPs)
- Integrating to a working example on Justin

The consortium individuals