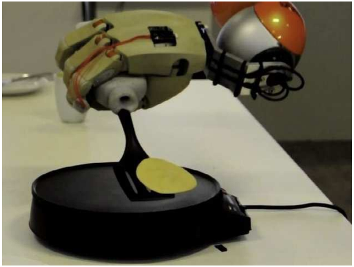
Control in RoboHow

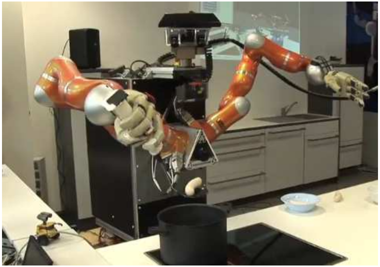
An Introduction

Michael Beetz
Intelligent Autonomous Systems Group
Technische Universität München

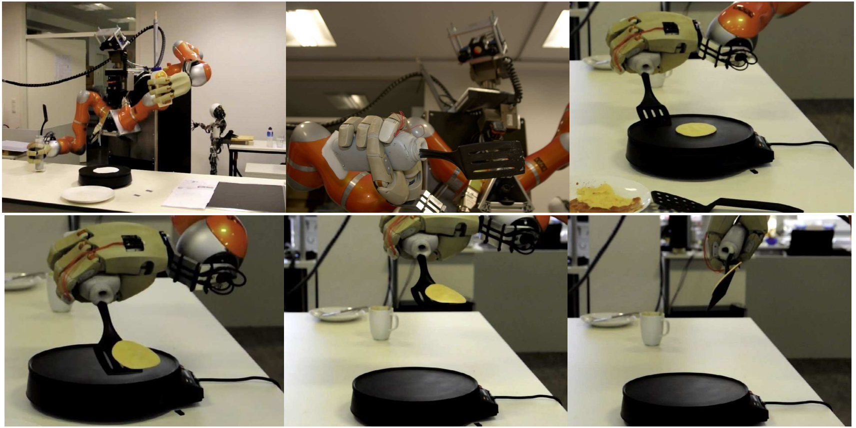
Munich, March 2012

Scenario 1: Meal Preparation (1)





Scenario 1: Meal Preparation (2)

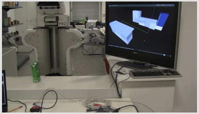


Scenario 1: Meal Preparation (3)

Shopping & cleaning up

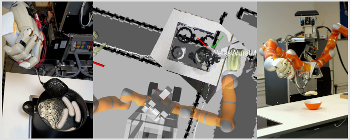
1. shopping with basket

2. cleaning up

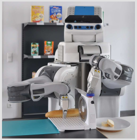



Making “Weisswürste”

1. making “Weisswürste”



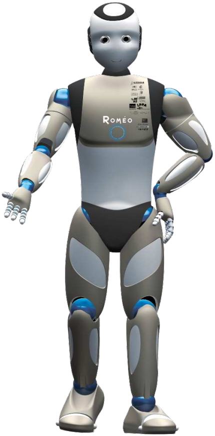
2. making sandwiches
3. cutting bread



Scenario 2: Robotic Office Services


(images courtesy of CNRS)

Scenario 3: Romeo



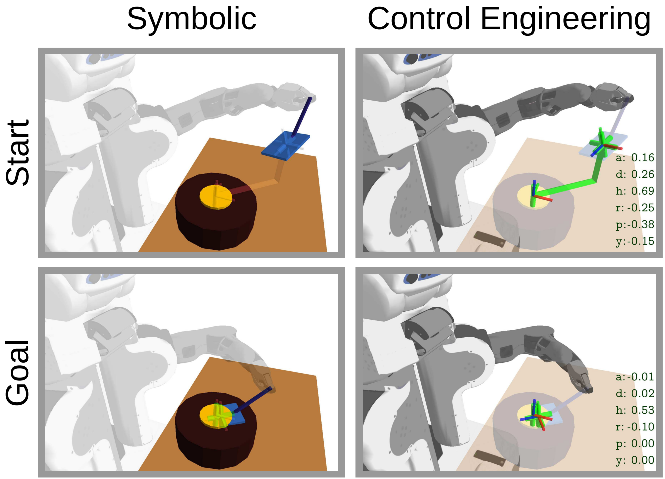
(images courtesy of Aldebaran)

Movement as First-class Objects

High-level control systems that do not represent and reason about movements

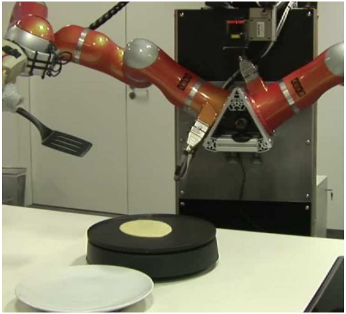
- cannot understand why actions work or do not work
- cannot change the execution of actions

Symbolic vs. Control View



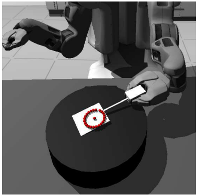
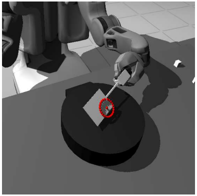
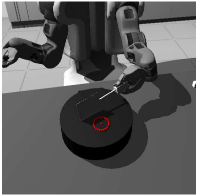
Plan-based Control: What we want

push the spatula under the pancake

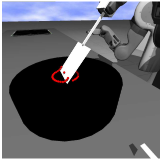


Possible Pushing Effects

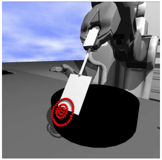
- Parameters: angle of spatula
- Outcomes: turned, not turned

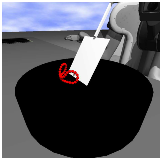
- Common failures:



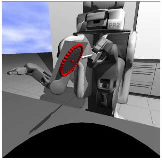
break



push off

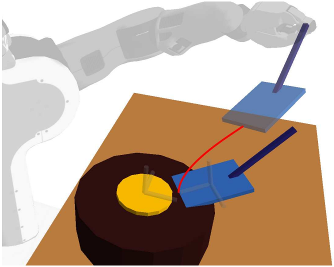


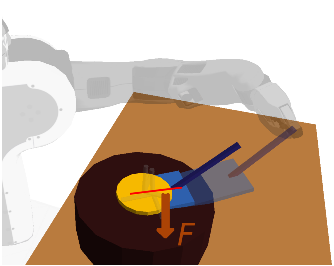
fold



stick on

Movement Phases





Issues/Questions/Challenges

- what is an appropriate representation of movement at the symbolic layer and can we have an interlingua to the control layer
  - movement specification
  - learning of movement models
- combination of iTASC and Stack of Tasks as an opensource software tool
- how to represent hybrid movement observations extracted from observation (in particular (distributions over) trajectories) in the control framework
- how to translate behaviors acquired through imitation learning (SEDS?) into the control framework
- how to couple the control framework to perception in perception-guided manipulation

From Vague Action Descriptions to Specifications (1)

push the spatula under the pancake

(perform (an action
        (type **push**)
        (object (an object
                (type **spatula**)))
        (destination ?loc = (a location
                        (**under pancake**)))))

From Vague Action Descriptions to Specifications (2)

push the spatula under the pancake

```
(perform (an action
              (type push)
              (with-grasp (a grasp
                             (type power-grasp)
                             (object (an object-part
                                        (part-of spatula)
                                        (type handle)))))
              (object (an object-part
                         (part-of spatula)
                         (type blade)))
              (destination ?loc = (a location ...))))
```

- adding: tool details, e.g. parts and handling

From Vague Action Descriptions to Specifications (3)

push the spatula under the pancake

```
(perform (an action
              (type push)
              (with-grasp (a grasp ...))
              (object (an object-part ...))
              (destination ?loc = (a location...))
              (desired-effect (and (pose spatula ?loc)
                                   (succeeds (an action
                                               (type lift)
                                               (object ?pc = (an object
                                                               (type pancake)))
                                               (starting-pose ?loc)))))
              (undesired-effect (damaged ?pc))))
```

- adding: desired and undesired effects

From Vague Action Descriptions to Specifications (4)

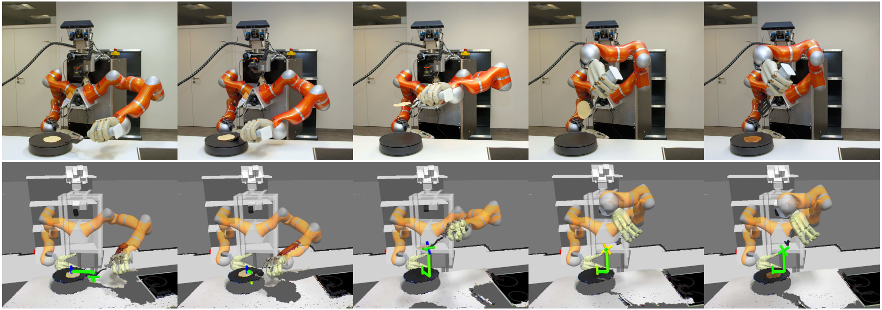
push the spatula under the pancake

```
(perform (an action
               (type push)
               (with-grasp (a grasp ...))
               (object (an object-part ...))
               (destination ?loc = (a location...))
               (desired-effect ...)
               (undesired-effect ...)
               (force-constraint (a force-constraint
                                   (object-acted-with (an object ...))
                                   (object-acted-on (an object ...))
                                   (min-contact-force min-value)
                                   (max-contact-force max-value)))))
```

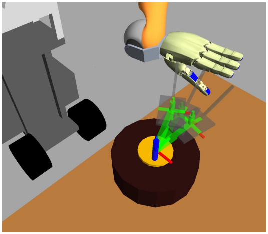
- adding: movement specification, e.g. force-constraints
- still to add: trajectories, events, sub-tasks, and priorities

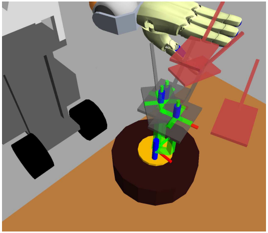
Preliminary feasibility study

The Pancake Flipping Task:



Allowed and Forbidden Poses...





- By specifying the task as constraints, we define a set of possible poses for each task step. (All are valid according to the task description).
- Finding the 'best' solution amongst these possibilities is an optimization problem.

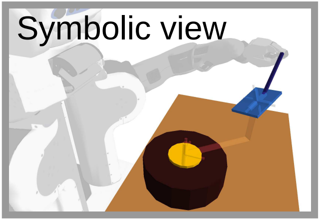
... Represented using Constraints

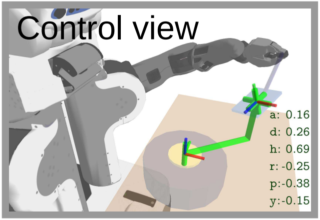
Goals for the contraint representation:

- 'Close' to natural-language instructions
- Can be reasoned about
- Corresponds to a control rule

Possible sources of information:

- Web instructions
- Human observations





Example Constraint

"Move the blade above the oven" (*web instruction?*):

(above blade oven)

is decomposed further into

(and (< (distance spatula oven) (radius oven))
(> (height spatula oven) 0))

Where distance and height are **task functions** that relate the spatula's pose to the oven's pose. We added a 'tolerance' and an offset for tuning.

→ *Filled in using human observation data?*

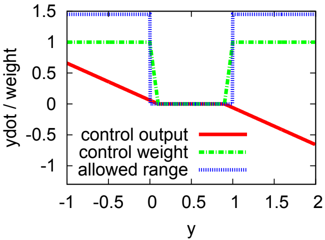
Task Function Approach

A task function is a (differentiable) function of the robots pose.

$$SE(3) \rightarrow \mathbb{R}^n$$

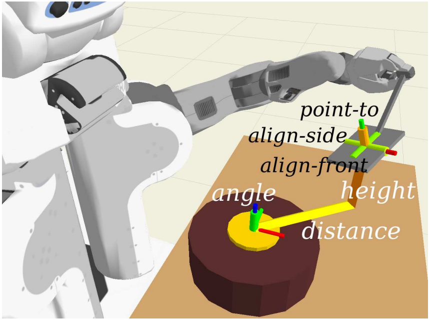
Its derivative (possibly computed numerically) is used to control the robot towards a desired value of that function.

Inequalities in constraints suggest **ranges** of desired values. We use a weight to disable a constraint when it is 'happy'.



Virtual Linkage Chain

We combined six scalar-valued task functions that we usually use:



- angle, distance and height form cylinder coordinates.
- align-front, align-side and point-to specify the spatula's orientation
- Each task function has a weight associated to it.

Constraints are combined into movement phases, intersecting their desired ranges. A sequence of such phases comprises a task.

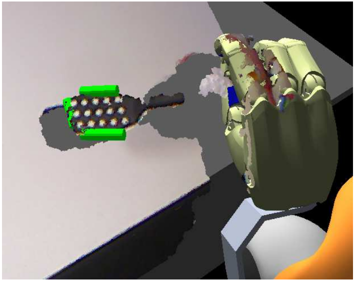
We expressed pancake-flipping using seven constraints:

c_p = (point-towards spatula oven)
c_h = (keep-horizontal spatula)
c_n = (move-next-to spatula pancake)
c_u = (move-under spatula pancake)
c_l = (lift spatula)
c_o = (keep-over spatula oven)
c_f = (flip spatula)

These constraints are combined into four steps:

$s_1 = (c_p \ c_h \ c_n)$
$s_2 = (c_p \ c_h \ c_u)$
$s_3 = (c_h \ c_o \ c_l)$
$s_4 = (c_o \ c_f)$

Grounding in perception



- Detection of **plane segments** and **line segments** in tools (using Kinect + Camera)
- Represented as **position** and **direction**
- Most of our task functions can be expressed using such features.

Future Work

- Optimize for next movement phase in the null space of the current one
- Learn constraints from human observations
- Deal with more than six elementary constraints
- Exploit the features of iTaSC and SOT

Thank you for your attention!