Robotic roommates making pancakes

Conference Paper · October 2011

DOI: 10.1109/Humanoids.2011.6100855 · Source: DBLP

8 authors, including:

Michael Beetz
Universität Bremen
420 PUBLICATIONS **8,943** CITATIONS
SEE PROFILE

Ulrich Klank
Technische Universität München
15 PUBLICATIONS **482** CITATIONS
SEE PROFILE

Dejan Pangercic
University of Amsterdam
40 PUBLICATIONS **761** CITATIONS
SEE PROFILE

Thomas Rühr
KUKA Laboratories GmbH
23 PUBLICATIONS **390** CITATIONS
SEE PROFILE

Robotic Roommates Making Pancakes

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner,
Dejan Pangercic, Thomas Rühr and Moritz Tenorth
Intelligent Autonomous Systems
Department of Informatics
Technische Universität München
Email: beetz@cs.tum.edu

***Abstract*—In this paper we report on a recent public experiment that shows two robots making pancakes using web instructions. In the experiment, the robots retrieve instructions for making pancakes from the World Wide Web and generate robot action plans from the instructions. This task is jointly performed by two autonomous robots: The first robot opens and closes cupboards and drawers, takes a pancake mix from the refrigerator, and hands it to the robot B. The second robot cooks and flips the pancakes, and then delivers them back to the first robot. While the robot plans in the scenario are all percept-guided, they are also limited in different ways and rely on manually implemented sub-plans for parts of the task. We will thus discuss the potential of the underlying technologies as well as the research challenges raised by the experiment.**

I. Introduction

Enabling robots to competently perform everyday manipulation activities such as cleaning up, setting a table, and preparing simple meals exceeds, in terms of task-, activity-, behavior- and context-complexity, anything that we have so far investigated or successfully implemented in motion planning, cognitive robotics, autonomous robot control and artificial intelligence at large. Robots that are to perform human-scale activities will get vague job descriptions such as "clean up" or "fix the problem" and must then decide on how to perform the task by doing the *appropriate actions* on the *appropriate objects* in the *appropriate ways* in all contexts.

Consider a robot has to perform a task it has not been programmed for — let's say making a pancake. First of all, the robot needs instructions which actions to perform. Such instructions can be found on webpages such as wikihow.com, though they are typically incomplete, vague, and ambiguous because they were written for human rather than robot use. They therefore require some interpretation before they can be executed. In addition to this procedural knowledge, the robot must also find and recognize the ingredients and tools that are needed for the task. Making pancakes requires manipulation actions with effects that go far beyond the effects of pick-and-place tasks in terms of complexity. The robot must pour the right amount of pancake mix onto the center of the pancake maker, and monitor the action success to forestall undesired effects such as spilling the pancake mix. It must handle the spatula exactly enough to push it under the pancake for flipping it. This requires the robot to select the appropriate force in order to push the spatula just strong enough to get under the pancake, but not too strong to avoid pushing it off the pancake maker.

In a recent experiment[1] we have taken up the challenge to write a comprehensive robot control program that indeed retrieved instructions for making pancakes from the world-wide web[2], converted the instructions into a robot action plan and executed the plan with the help of a second robot that fetched the needed ingredients and set the table. The purpose of this experiment is to show the midterm feasibility of the visions spelled out in the introductory paragraph and more importantly the ***better understanding*** of how we can realize control systems with these capabilities ***by building*** such a system. We call this an ***experiment*** rather than a demonstration because we tested various hypotheses such as whether the localization accuracy of a mobile robot suffices to perform high accuracy tool manipulation such as pushing a spatula under the pancake, whether successful percept-guided behavior for sophisticated manipulation actions can be generated, and whether robot plans can be generated from web instructions made for human use. Indeed, the success of this experiment and the need for generalization and automation of methods we identified as essential for the success of the experiment define the objectives of the current research agenda of our group.

Fig. 1. TUM-Rosie and TUM-James demonstrating their abilities by preparing pancake for the visitors.

In the remainder of the paper we proceed as follows. We will start with explaining how the robot interprets the abstract web instructions and aligns them with its knowledge base, with the plans in its plan library and with the perceptual knowledge about the ingredients and objects in its environment. Then

[1] **Accompanying video: http://www.youtube.com/watch?v=gMhxi1CJI4M**
[2] http://www.wikihow.com/Make-Pancakes-Using-Mondamin-Pancake-Mix

we show how we refine that plan and produce executable control programs for both robots, followed by an explanation of the generalized pick-and-place actions that we used to open containers such as drawers and the refrigerator. We then discuss the action of actually making the pancake, including the perception techniques and the dexterous manipulation using a tool. Subsequently, we give a short introduction into the framework that enables our robots to reason about the performed actions. The paper is concluded with a discussion of limitations and open research issues.

II. Generating Sketchy Plans from Instructions

Our robots are equipped with libraries of plans for performing everyday manipulation tasks. These libraries also include plans for basic actions such as picking up an object, putting it down or pouring that are used as basic elements in generated plans. Whenever the robot is to perform a task that is not provided by the library, it needs to generate a new plan from these basic elements for the novel task. The computational process for generating and executing plans is depicted in Figure 2. The robot searches for instructions how to perform the tasks and translates them into robot action plans. This methods stands in contrast to action planning methods like those that are used in the AI planning competitions [1].

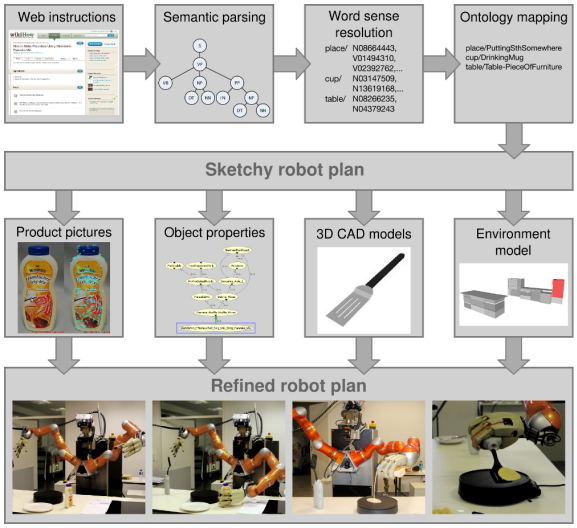


Fig. 2. Translation of web instructions into a robot plan.

In the experiment, the robots were to make pancakes and used instructions from wikihow.com to generate a robot plan. These instructions specify the ingredients, milk and prepared pancake mix, and a sequence of action steps:

1) Take the pancake mix from the refrigerator
2) Add 400ml of milk (up to the marked line) shake the bottle head down for 1 minute. Let the pancake-mix sit for 2-3 minutes, shake again.
3) Pour the mix into the frying pan.
4) Wait for 3 minutes.
5) Flip the pancake around.
6) Wait for 3 minutes.
7) Place the pancake onto a plate.

Generating a plan from such instructions requires the robot to link the action steps to the appropriate atomic plans in its library and to match the abstract ingredient and utensil descriptions with the appropriate objects in its environment. It further needs to select the appropriate routines for perceiving, locating and manipulating these objects. Please note that the robot did not have a control routine for filling milk into a bottle. We left out this step in the generated control program and added the milk manually.

Our robots use an *ontology* that formally describes and defines things and their relations – descriptions like the ones that can be found in an encyclopedia, which are thus referred to as *encyclopedic knowledge*. Examples of such knowledge are that a refrigerator is a container (i.e. it can contain other objects), a sub-class of cooling devices and electrical household appliances, and a storage place for perishable goods. In our system, we use KnowRob [2][3], an open-source knowledge processing framework that provides methods for acquiring, representing, and reasoning about knowledge.

Using the encyclopedic knowledge base the translation of instructions into robot plans is performed by the following sequence of steps [3]. First, the sentences are parsed using a common natural-language parser [4] to generate a syntax tree of the instructions. The branches of the tree are then recursively combined into more complex descriptions to create an internal representation of the instructions describing the actions, the objects involved, locations, time constraints, the amount of ingredients to be used etc. The words in the original text are resolved to concepts in the robot's knowledge base by first looking up their meanings in the WordNet lexical database [5], and by then exploiting mappings between WordNet and the Cyc [6] ontology. Our system employs a simple method based on the phrase context and on information about object-action pairs obtained from Cyc to disambiguate between possible word meanings.

```
(def-top-level-plan ehow-make-pancakes1 ()
    (with-designators (
          (pancake (an object '((type pancake)
                                (on ,frying-pan))))
          (mixforbakedgoods2 (some stuff '((type pancake-mix)
                                           (in ,refridgerator2))))
          (refrigerator2 (an object '((type refrigerator))))
          (frying-pan (an object '((type pan))))
          (dinnerplate2 (an object '((type plate))))
          (location0 (a location `((on ,dinnerplate2)
                                   (for ,pancake2)))))

  (achieve `(object-in-hand ,mixforbakedgoods2))
  (achieve `(container-content-transfilled
                              ,mixforbakedgoods2
                              ,frying-pan))
  (sleep 180)
  (achieve `(object-flipped ,pancake))
  (sleep 180)
  (achieve `(loc ,pancake ,location0)))))
```

The code above shows the sketchy plan that has been generated from the web instructions. The declaration part creates entity descriptors for the objects referred to in the instructions. A descriptor consists of an article (definite or indefinite), an entity type (object, location, stuff, ...) and a set

[3]http://www.ros.org/wiki/knowrob

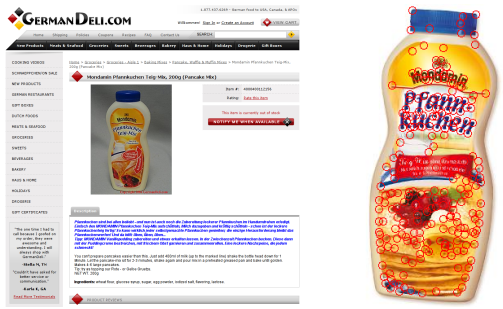


Fig. 3. Bottle of pancake mix from GermanDeli.com (left), and the extracted features to recognize the object (right).

of attribute-value pairs. Stuff refers to homogeneous things here, such as water, milk, etc. Then the instruction steps themselves are specified as a sequence of achievement goals, where the states to be achieved include the object descriptors they refer to. Thus, instead of specifying the action to take the pancake mix, the plan specifies the goal of having the pancake mix in the hand as its corresponding plan step. We will discuss the reasons in the next section.

III. Refining the Sketchy Plan

Two aspects of the generated plan deserve further discussion. First, the plan consists of a declaration part, in which objects and other kinds of entities are specified, and second, the action steps in the instructions are substituted by declarative goal statements.

Ingredients and utensils that are listed in the instructions are included into the plan as *designators*, which are descriptions of entities such as objects or locations. Designators in the plan are often abstract and incomplete. For example, the robot has inserted a local plan variable *mixforbakedgoods2* that is bound to the object description *(some stuff (type pancake-mix))*, which can be directly inferred from the web instructions (ingredients: pancake mix). However, as the robot has to fetch the pancake mix, it needs tools required to recognize and localize it first.

A. Recognizing Objects

Many ingredients can be recognized based on the images on the front faces of their packages, which are often pictured in shopping websites. To use these information resources, we have downloaded the product descriptions of the web site *GermanDeli.com*, which contains about 3500 common products. The products of this website are categorized and include a picture of the front page of the package. In the robot's knowledge base, the product is described as a specialization of the respective category like *DairyProduct*.

The product images are used to learn Scale Invariant Feature Transform (SIFT) [7] features for their recognition. Thus, when a plan contains an abstract description of an object, the robot searches its library of object models for a matching SIFT descriptor. The designator for the object is then extended with the descriptor. The following code snipped shows an example designator extended with such a SIFT descriptor.

```
(an object (type pancake-mix)
  (perceptual-appearance sift-descriptor-23))
```

At execution time, the designator tells the robot's perception system that and how it can detect, recognize, and localize the object. The perception component used for this task is the so-called ODUFINDER (*Objects of Daily Use Finder*[4]).

ODUFINDER first searches for point clusters (object candidates) that are supported by horizontal planes [8]. These point clusters are detected in colored 3D point clouds that have been generated by combining a scan of a tilting laser scanner with a camera image of the same scene. The region of interest corresponding to the object candidates is generated by back-projecting the point clusters into the image.

To recognize objects ODUFINDER uses a database of object images which can for example be the ones obtained from *GermanDeli.com*. ODUFINDER compares SIFT features extracted from the region of interest with the object descriptions in the database. The database search is performed using a vocabulary tree, a technique that has been developed to search for text strings in huge text archives and was first applied to image search in [9]. To achieve higher accuracy, ODUFINDER first oversegments the regions of interest and then combines detected object parts to infer the presence of the complete object. These extensions substantially increase the detection rate and reliability in the presence of occlusions and difficult lighting conditions.

B. Finding Objects

In real household environments, objects are typically stored inside cupboards and drawers, so the robot has to search for them before it can recognize them. To find the required objects quickly, a robot should search for the objects at their most likely places first. Our robots use a semantic 3D object map of the environment in which structured models of objects, such as cupboards consisting of the container, the door, the handle and hinges, are associated with first-order symbolic descriptions of the objects that mainly come from the robot's encyclopedic knowledge base KNOWROB-MAP [10]. The environment map also contains information about common locations of objects of daily use.

Figure 4 shows the part of the knowledge base that is relevant for inferring likely locations for the pancake mix. This knowledge base describes pancake mix as a perishable item and a kind of groceries, i.e. a kind of object that can be found in the *GermanDeli* online shop. It further contains information about refrigerators, namely that they are household appliances, cooling devices, container artifacts, and that they are storage places for perishable goods. Using the following Prolog rule that states that a possible location for an object is its storage place

```
possibleLocation(Obj,Loc)
  :- storagePlaceFor(Obj,Loc)
```

the robot can infer that the pancake mix can perhaps be found

[4]http://www.ros.org/wiki/objects_of_daily_use_finder

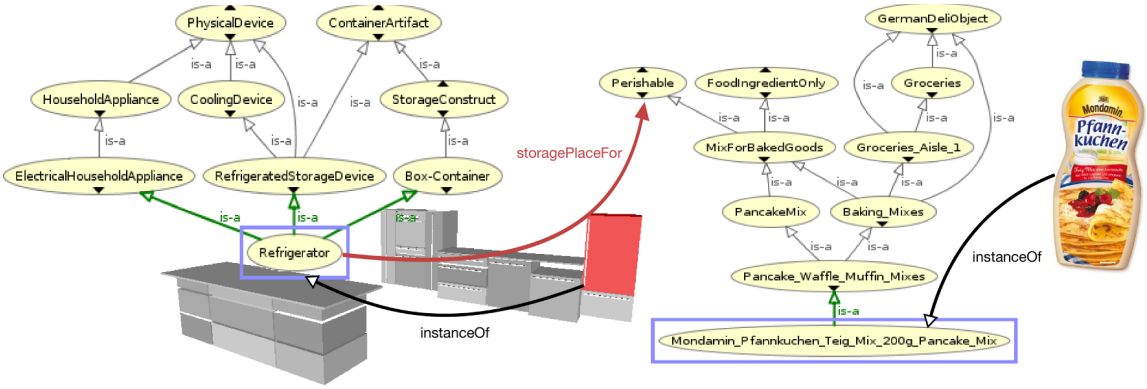


Fig. 4. Reasoning steps to infer a probable storage location for a bottle of pancake mix.

in the refrigerator and update the object descriptor for the pancake mix accordingly:

```
(an object
  (type pancake-mix)
  (perceptual-appearance sift-descriptor-23)
  (at (a location (inside refrigerator2))))
```

As the result of the plan refinement phase, we have combined some of the individual entity descriptors (*pancake2*, *mix-forbakedgoods2*, *refrigerator2*, and *location1*) into a combined descriptor that also includes the knowledge that the pancake is created by pouring pancake mix onto the pancake maker. This descriptor is:

```
(an object
  (type bottle)
  (contains (some stuff (type pancake-mix)))
  (perceptual-appearance sift-descriptor-23)
  (at (a location (inside refrigerator2))))
```

IV. Generalized Pick & Place Actions

Another research topic covered in the experiment is the performance of generalized fetch and delivery tasks. The PR2 robot in our experiment was to pick, carry, and place bottles of pancake mix, plates, spatulas, and knives and forks. The objects required different means of perception. Some objects were perceived using 3D data from structured light stereo and without prior model knowledge, while others where detected and recognized by their appearance, and yet others using 3D CAD models. The objects where placed on counters, in drawers and the fridge. The objects also require different strategies for picking them up, some of them like the plates required coordinated bimanual grasping with an additional constraint of holding the object horizontal.

Performing pick and place tasks in such a general setting is a surprisingly difficult task. Even a simple pick up action requires the robot to decide where to stand, which arm to use, which grasp to apply and where to grasp, to name only a few. The decision-making tasks become even more complex. Each of these decisions depend on the context: the objects acted on, their states, the scene, the task, whether people are present, the habits and preferences of users, and so on.

Fig. 5. The PR2 opening different containers using the same controller without knowledge of the mechanism but only of the handle position.

One of the aspects we investigated in more depth in the experiments was how to open pieces of furniture. Figure 5 shows the robot opening various cupboards, drawers and appliances in the context of these fetch and delivery tasks. We developed a general controller that uses the compliance of the arms and the fingertip sensors to open different types of containers without a-priori knowledge of the mechanism (e.g. rotational or prismatic) [11]. The controller moves the robot's base during the process of opening containers when necessary. The trajectory of minimum resistance is followed, initialized by a direction defined by the furniture surface normal.

As the general routine for opening containers is very slow, the robot learns an articulation model for the containers when it opens them for the first time. Later, the robot uses the recorded trajectory in subsequent runs to open the container faster, only monitoring deviations from the trajectory for failure detection. The robot base pose used during manipulation is chosen optimistically, only regarding principle reachability. An actual motion plan for the arm is only generated once the robot is at its manipulation location; if motion planning fails, the robot navigates to a different pose and tries again.

V. Perception-guided Pancake Making

The experiment also includes the realization of a simple manipulation task that exhibits many characteristics of meal preparation tasks: cooking a pancake on a pan. Taking autonomous robot control from pick and place tasks to everyday object manipulation is a big step that requires robots to

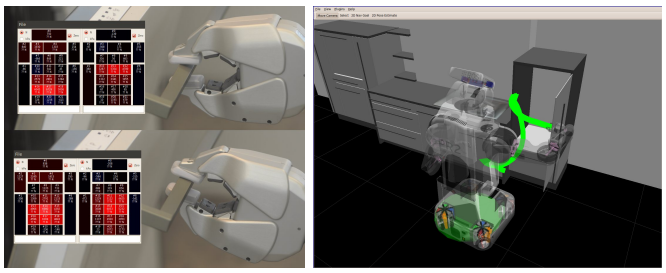


Fig. 6. Using the fingertip sensors to maintain a good grasp (left). The trajectory of the hand while opening the fridge (right).

understand much better what they are doing, much more capable perception capabilities, as well as sophisticated force-adaptive control mechanisms that even involve the operation of tools such as the spatula.

In this section, we consider the process of making the pancakes by structuring it into the three steps specified in the instruction: 1) pouring the pancake mix; 2) flipping the pancake; and 3) putting the finished pancake on the plate. All steps are performed autonomously through the use of perception-guided control routines.

A. Pouring the Pancake Mix onto the Pancake Maker

The first step, pouring the pancake mix requires the robot to 1) detect and localize the cooking pan or pancake-maker as well as the bottle with the pancake mix, 2) pick up the pancake mix and position the tip of the bottle above the center of the pancake maker, and 3) pour the right amount of pancake mix onto the pancake maker. We will discuss these steps below.

1) Detecting and Localizing the Relevant Objects: The robot performs the detection and localization of the relevant objects using object type specific perception routines. The black color in combination with the metallic surface of the pancake maker makes the readings of time-of-flight sensors very noisy, and the heat of the pancake maker requires particularly high reliability of operation. On the other hand, the accuracy demands for successful action execution are less for the destination of the pouring action (roughly in the center of the object) than for successfully grasping an object. One basic principle that we used for the realization of perceptual mechanisms is that we apply a team of context specific perception mechanisms rather than aiming for a single but overly general perception mechanism [12].

Thus we equipped the robot with a previously calibrated planar shape model of the top plane of the pancake maker in order to roughly detect and localize it. For matching it in the online phase we used the method proposed by Hofhauser et al. [13] on images of a RGB-camera, which gives an accurate result in less than half a second.

The method for localizing the pancake mix also exploits the task context by using the position where the other robot put the mix as prior information. Thus, the robot can confine itself to finding a point cluster at the approximate position with the approximate dimensions of the pancake mix. This method is efficient as well as reliable and accurate enough to pick up the pancake mix (see [14] for details on the cluster detection). The pancake-mix is grasped with a power grasp coupled with a validation of the grasp success, which we discuss later.

2) Pouring the Adequate Amount of the Pancake Mix: In order to make pancakes of the appropriate size the robot has to pour the right amount of pancake mix onto the pancake maker. This is accomplished by estimating the weight of the mix that has been poured onto the pan. After successfully lifting the pancake-mix, the weight of the bottle is estimated using the measured joint torques.

To pour the pancake mix onto the pancake maker, the robot estimates the height of the top of the pancake mix bottle and uses this information to determine the right pose of the robot hand. The pouring time is adjusted using a hand crafted linear formula with the weight of the bottle as a parameter.

In order to validate the success and estimate the effects of the pouring action the robot applies a blob detection with the image region corresponding to the pancake maker as the search window. After a color-based segmentation, all components which are not similar in intensity to the pan are considered as pancakes or pancake parts. The noise removal on the segmentation results then yields a sufficiently good model of the position (relative to the pan) and form of the pancake. This perception task is performed in real time and also works in the presence of the spatula.

B. Flipping the Pancake

The key steps in flipping the pancake are 1) to grasp and hold the spatula sufficiently well to use it as a tool, 2) to calibrate the spatula with the hand such that the robot can control and determine the accurate pose of the spatula through its internal encoders, and 3) to perform adaptive stiffness control to push the spatula under the pancake without pushing the pancake off the pancake maker.

1) Picking Up and Holding the Spatula Properly: The spatula has been modified to give it a broader handle, so that the robot can hold it securely in its oversized hand.

The spatula is detected, localized, and approximately reconstructed through the use of our 3D sensors, in this case the ToF camera. To match the surface of the spatula with the current sensor data we use the method proposed by Drost et al. [15]. To train the object we took the result of a 3D cluster segmentation of the object in a clutter-free scene as the surface template.

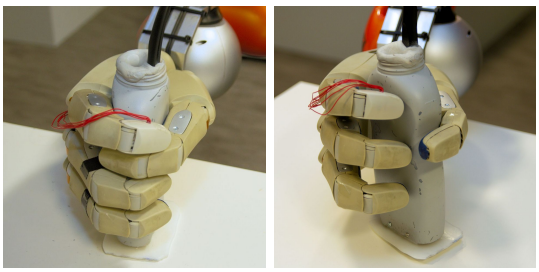


Fig. 7. A supervision system detects good (left) and bad (right) grasps.

To deal with uncertainty in perception that can lead to sub-optimal grasps, a simple system is used to evaluate the grasp quality using measured finger positions and torques. To this end, the data vector distances between current measurements and known good and bad grasps are calculated and used as quality values. A low quality score leads to a retry of the grasp, and if the score is low again, the object is re-localized and the complete grasping action is repeated.

Figure 7 shows a grasp that fulfills these properties on the left, and a failed one on the right. Grasps may fail due to unexpected contacts with parts of the object or delays in the control of the fingers.

2) Controlling the Spatula as an End Effector: To lift the pancake successfully, the robot should treat the spatula as a body part rather than an object that has to be manipulated. This means the kinematic model of the arm is extended to include the spatula, and the algorithms used to detect collisions with the hand are modified to detect collisions on the spatula.

To use the spatula as a tool, its relative position to the hand has to be known precisely after the robot has grasped it. The robot performs an online calibration step using the same method that is used to localize the pancake maker. In this case, the planar assumption is valid for the complete top part of our tool. To gain a higher accuracy, the matching is applied several times, always matching on two stereo images and validating the consistency of the results. The results from all matchings are taken as a set of hypotheses, which are used to calculate a robust mean value in translation and rotation. Figure 8 shows the position in which the robot holds the spatula (left), the intrinsic view of the robot in visualization (middle) and the camera image at this point in time (right).

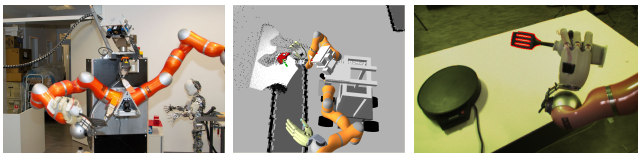
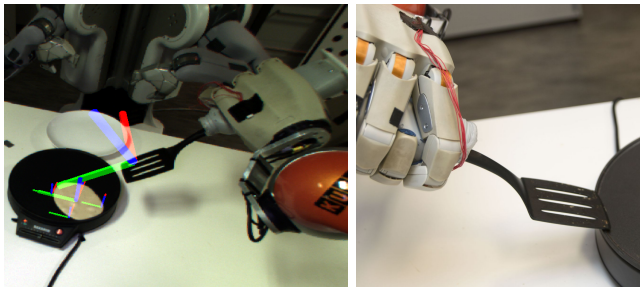


Fig. 8. Calibration of the spatula.

3) Movement Control of the Spatula Tip: To flip a pancake with a spatula, the robot must push the spatula under the center of the pancake without pushing the pancake off and deforming or destroying it. To do so, the robot pushes the spatula down until it touches the pan and the tip is parallel to the surface. The robot moves the spatula in a straight line between the point of contact with the pan and the center of the pancake.

Figure 9(b) shows the moment when the robot has lowered the tool until it touched the pan. This contact produces measurable force changes in the fingers, so that the event can be detected reliably.

In order to correctly detect the contact of the tip with the pan, a band pass filter is applied to the 12 torque streams coming from the hand at 1kHz, eliminating the constant torques for holding the object and the high-frequency noise from the motor controllers. We calculate the dot product of the filtered torque vectors with a template vector, and a high value is measured shortly after the collision.



(a) Approach the pancake (reference frames overlayed). (b) First contact of the spatula with the pan.

Fig. 9. Flipping the pancake.

After touching the pan, its height is known precisely, and the rest of the movements take this into account.

4) Picking and Turning the Pancake: The trajectory to pick up the pancake, lift and turn it was taught by demonstration and is only parametrized with the pancake's position, corrected by the newly estimated height of the pan. The spatula has to be positioned under the pancake, then the pancake can be lifted. Afterwards, the pancake has to be turned and dropped back to the pan. The pancake tends to stick to the spatula in this stage, which requires the robot to apply various accelerations to the spatula to separate the pancake again. This introduces uncertainty about the position of the pancake after finishing this action.

5) Checking the Estimated Result: Dropping the pancake back onto the pan can have three possible outcomes: 1) the pancake falls back to its original position in the center of the pan, 2) the pancake drops a little bit off the center (usually still on the pan) and 3) the pancake keeps sticking on the spatula. The first two cases can be detected by re-detecting the pancake on the pan, and the third case follows if the pancake cannot be detected on the pan anymore. While case one does not require further actions, the second case is corrected by centering the pancake with the spatula again. In the third case, the robot continues moving the arm up and down until the pancake drops.

VI. Reasoning about Action Execution

The last topic of the experiment was the demonstration that the robot was not only performing the manipulation activity, but that it also *knew what it was doing, when, and why*. By this we mean that the robot could answer queries such as *"Why did you open the fridge door?"* with *"Because I wanted to grasp the pancake mix."*. This is achieved by symbolically interpreting logged data from the sensors, the robot poses and states, as well as the data from the execution of the plans that have been recorded during the experiment.

An in-depth description of the framework we used can be found in [16]. Our robots can perform this kind of reasoning about plans and executed activities because their control programs are written in the CRAM Plan Language [17]. Besides being a programming language with features for parallel action execution and extensive (local) exception handling, synchronization and reactivity, it provides mechanisms to semantically annotate control programs (plans) and to record the state of

plan execution at any point in time, including which sub-processes have been started and terminated, and with which status and result they terminated.

Competences of our robots are captured in plans, carefully designed concurrent, reactive, perception-guided control programs that contain explicit semantic annotations in a first-order representation. For instance, a plan that places an object *Obj* at location *Table* is annotated with *Achieve(Loc(Obj, Table))* and one of its sub-plans is *Achieve(ObjectInHand(Obj))* (see the plan in Section II).

When the robot gets a task such as fetching the pancake mix from the fridge, it starts executing the corresponding plan and thereby generates a task tree for the plans and subplans that it executes. Sensor data and control commands continuously update the robot's belief state. To allow for post-execution reasoning, the task tree and the belief state are stored in an extensive execution log, which contains enough information for the post execution reconstruction of the state of program execution at any point in time.

The reasoning framework works on a first-order predicate logic abstraction of the generated execution log. Abstractions are generated on demand, i.e. the implementation of the predicates used in queries accesses the sub-symbolic information of the execution log and creates a symbolic representation of it. For instance, to state that a variable should be a task (an object that corresponds to the internal representation of a plan), we use the predicate *Task(tsk)* which matches all nodes in the task tree. To state that a task is a direct sub-task of another one, we define the predicate *Subtask(*tsk_p*,* tsk_s*)*. To reason about semantics, we need to access the purpose of a plan, i.e. its annotation. For that, we define the predicate *TaskGoal(tsk, o)*.

To infer complex failure situations from execution logs, defining predicates to access the task tree and annotations of plans is not sufficient. We also need access to the (believed) state of the world and changes in the belief. Consider the following example where we infer all plans that tried to pick up the pancake mix bottle but failed:

$$\begin{aligned}&\mathit{Task}(\mathit{tsk}) \wedge \mathit{TaskGoal}(\mathit{tsk}, \mathit{Achieve}(\mathit{ObjectInHand}(\mathit{obj})))\\ &\quad \wedge \mathit{TaskStart}(\mathit{tsk}, t_s) \wedge \mathit{TaskEnd}(\mathit{tsk}, t_e)\\ &\quad \wedge \mathit{DesignatorPropertyAt}(\mathit{obj}, \mathit{Type(PancakeMix)}, t_s)\\ &\quad \wedge \neg \mathit{Holds}(\mathit{ObjectInHand}(\mathit{obj}), \mathit{At}(t_e))\end{aligned}$$

We use predicates to unify time points when a task has been started and ended and to access the properties of a designator. Please note that since the properties of a designator might change with every detection of the object, the predicate *DesignatorPropertyAt* also requires a point in time. Finally, the *Holds* predicate is used to reason about the (believed) state of the world. The *Holds* predicate is not defined over achieve assertions in plans, but over events that are generated by sensor callbacks. This allows us to separate what the robot was to do, i.e. what was stated in the plan, from what the robot actually did, i.e. what the robot actually perceived.

The system also records low-level data structures such as trajectories that have been executed. Figure 6 (right) shows the trajectory of the right hand the robot was following while opening the fridge. We can unify the list of points with the variable *traj* as shown in the following example:

$$\begin{aligned}&\mathit{Task} \wedge \mathit{TaskGoal}(\mathit{tsk}, \mathit{Achieve}(\mathit{ObjectOpened(obj)}))\\ &\quad \wedge \mathit{TaskStart}(\mathit{tsk}, t_s) \wedge \mathit{TaskEnd}(\mathit{tsk}, t_e)\\ &\quad \mathit{TrajectoryOfFrame}(\text{"}\mathit{r\_gripper\_tool\_frame}\text{"}, t_s, t_e, \mathit{traj})\end{aligned}$$

This shows that the system can also directly access all low-level data structures that have been recorded.

VII. Conclusions and Research Issues

In this paper we have presented an experiment in which robots retrieved a simple instruction for a meal preparation task from the web and semi-automatically translated it into a robot plan that was jointly executed by two robots. The experiment was a feasibility study, and we had to deal with many of the issues identified in [18]. Many aspects have been solved specifically and some actions have been hand-coded. We conclude from the experiment that the generation of robot plans for complex everyday manipulation tasks from web instructions and their performance with state of the art mobile manipulation platforms is feasible.

One important aspect of the experiment was that we integrated previously independent research efforts and validated that they can be combined effectively and contribute to our overall goals and systems.

For us, the analysis of web instructions that had originally been created for human use sheds much light on the problem-solving capabilities that are needed by autonomous robots to perform everyday manipulation tasks. Very informative are the information pieces that are missing in web instructions or spelled out only abstractly. Web instructions often implicitly assume the objects to be in the right places and only specify the manipulation actions. They hardly ever state the actions for fetching items, so a robot carrying out the actions must infer where the relevant objects are, how they look like, how they should be held, etc. Thus, the robot control programs have to be knowledge-intensive to infer the necessary, but missing, pieces of information. Apart from filling information gaps in the instructions, the knowledge is also required to bridge the gap between the abstract instructions and the routines in the robot's plan libraries.

Another interesting aspect is the handling of action parameterizations. Many necessary action parameters are not specified, for example the height from which the pancake mix is to be poured. This implies that a robot must know how the height might affect the outcome of the pouring action — whether or not the mix is spilled, and whether the pancake will be circular. In our opinion, the robot must be capable of mentally executing actions in order to predict their consequences. To this end, we investigate physical simulation as a suitable means to equip the robot with these kinds of predictive capabilities [19] in combination with constraint- and optimization-based movement specification and execution, as for example provided by the iTASC framework [20].

Yet another lesson that we learned is the range of perception tasks that the robot must accomplish: it must detect objects, recognize, localize, reconstruct them, it has to calibrate the tools in its hand, it has to monitor the deformation of the

pancake and so on (cf, [21]). Also the objects and stuff that are to be perceived vary a lot: some objects are textured, others have identifiable forms, others are transparent and others, like plates, are indistinguishable from each other. Robot perception has to go far beyond the library of methods that is currently used in the control software. So, our research will investigate perception systems that reason about the right methods to use for the perception tasks at hand.

Another conjecture that we make is that it will be probably very difficult to frame an action, such as pushing the spatula under the pancake, as a motion planning and execution problem. Performing the action successfully goes well beyond the capabilities of current motion planning frameworks. For example, the robot has to push the spatula onto the pancake maker to determine its height and to achieve the required accuracy, thereby exploiting the scene context. Many everyday manipulation actions require a good action execution strategy for their successful execution. We conclude that if we aim for generality, we must investigate action planning methods that can generate specialized action execution strategies and reason about their physical consequences. This requires a new generation of robot action planning methods that are capable of naive physics reasoning and can predict action consequences in continuous action and scene parameter spaces. First attempts can be found in [22].

VIII. Acknowledgments

This work is supported in part within the DFG excellence initiative research cluster *CoTeSys* (`www.cotesys.org`) and by the EU FP7 Project *RoboEarth* (grant number 248942).

References

[1] M. Fox and D. Long, "PDDL2.1: An extension of PDDL for expressing temporal planning domains." *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[2] M. Tenorth and M. Beetz, "KnowRob — Knowledge Processing for Autonomous Personal Robots," in *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2009, pp. 4261–4266.

[3] M. Tenorth, D. Nyga, and M. Beetz, "Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web." in *IEEE International Conference on Robotics and Automation (ICRA).*, 2010, pp. 1486–1491.

[4] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 423–430.

[5] C. Fellbaum, *WordNet: an electronic lexical database*. MIT Press USA, 1998.

[6] D. Lenat, "CYC: A large-scale investment in knowledge infrastructure," *Communications of the ACM*, vol. 38, no. 11, pp. 33–38, 1995.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] R. B. Rusu, I. A. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time Perception-Guided Motion Planning for a Personal Robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15 2009, pp. 4245–4252.

[9] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2161–2168.

[10] M. Tenorth, L. Kunze, D. Jain, and M. Beetz, "KNOWROB-MAP – Knowledge-Linked Semantic Object Maps," in *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, December 6-8 2010.

[11] N. Blodow, L. C. Goron, Z.-C. Marton, D. Pangercic, T. Rühr, M. Tenorth, and M. Beetz, "Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September, 25–30 2011, accepted for publication.

[12] I. Horswill, "Analysis of adaptation and environment," *Artificial Intelligence*, vol. 73, pp. 1–30, 1995.

[13] A. Hofhauser, C. Steger, and N. Navab, "Edge-based template matching with a harmonic deformation model," in *Computer Vision and Computer Graphics: Theory and Applications - VISIGRAPP 2008*, ser. Communications in Computer and Information Science, vol. 24. Berlin: Springer-Verlag, 2009, pp. 176–187.

[14] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz, "Real-time cad model matching for mobile manipulation and grasping," in *9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, December 7-10 2009, pp. 290–296.

[15] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[16] L. Mösenlechner, N. Demmel, and M. Beetz, "Becoming Action-aware through Reasoning about Logged Plan Execution Traces," in *Submitted to the IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2010.

[17] M. Beetz, L. Mösenlechner, and M. Tenorth, "CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments," in *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2010.

[18] C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[19] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz, "Simulation-based temporal projection of everyday robot object manipulation," in *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone, and Sonenberg, Eds. Taipei, Taiwan: IFAAMAS, May, 2–6 2011.

[20] R. Smits, T. D. Laet, K. Claes, H. Bruyninckx, and J. D. Schutter, "itasc: A tool for multi-sensor integration in robot manipulation," in *Multisensor Fusion and Integration for Intelligent Systems*, ser. Lecture Notes in Electrical Engineering, H. K. H. Hahn and S. Lee, Eds. Springer, 2009, vol. 35, pp. 235–254. [Online]. Available: http://www.springerlink.com/content/v4j60wx14l087354/fulltext.pdf

[21] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, and M. Inaba, "Vision based behavior verification system of humanoid robot for daily environment tasks," in *Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2006, pp. 7–12.

[22] L. P. Kaelbling and T. Lozano-Perez, "Hierarchical planning in the now," in *IEEE Conference on Robotics and Automation*, 2011.