Priming Transformational Planning with Observations of Human Activities

Moritz Tenorth and Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{tenorth, beetz}@in.tum.de

***Abstract*— People perform daily activities in many different ways. When setting a table, they might use a tray, stack plates, stack cups on plates, leave the doors of a cupboard open when taking several items out of it. Similarly flexible behavior is desired when mobile robots perform household tasks. Moreover, they should perform actions in a way that they are accepted by the people, for example by showing human-like behavior.**

In this paper we propose to extend a transformational planning system with models characterizing the behavior produced by the different plans in the plan library. These models are used by the robot to select a plan that resembles human behavior. In addition to acting more human-like, this helps the robot choose good plans for a task by imitating humans instead of performing exhaustive search.

We show the feasibility of this approach using a household robot application as an example and present empirical results on the classification accuracy in this domain.

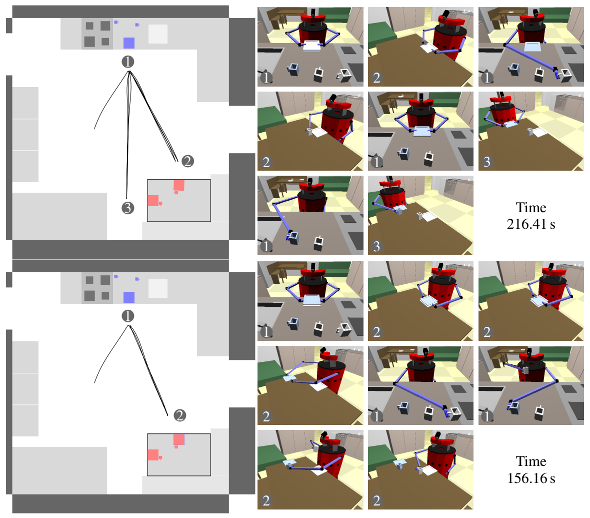


Fig. 1. Top: Default plan for setting the table. The robot transports objects one by one which is very inefficient. Bottom: In the optimized plan, the robot stacks plates before bringing them to the table and uses both arms for transporting cups which greatly increases the performance [1].

I. Introduction

Mobile service robots are becoming more and more dexterous, and the tasks they can accomplish change from simple navigation to complex mobile manipulation. Plans for such complex activities need to describe various actions, action parameters, the object to be manipulated, etc. This increase in complexity makes the generation and optimization of plans very challenging since many more influence factors have to be considered, and since many more options exist how to adapt the task specification.

Considering only navigation, optimizing a plan means smoothing the path so that it becomes shorter or can be traversed with higher speed. The influence of a different plan policy on the robot's performance can quite easily be predicted. For mobile manipulation tasks, however, the robot has far more choices: Should it carry a whole stack of plates at once? How about using a tray or a container? Is it worth fetching the tray from another part of the kitchen? Our robot can adapt its plans using transformation rules which correspond to these abstract decisions.

The example in Figure 1, taken from [1], visualizes the effect of plan transformations on the performance of the robot. In the upper part, the robot executed a default plan that just described how to set a table in a straightforward way, transporting the objects one by one. Obviously, this is quite inefficient. In contrast, in the lower part of Figure 1, the robot executed a transformed plan that made it transport plates as a stack and use both grippers for simultaneously transporting cups, which significantly improved the performance.

Inferring which one of these modifications has the biggest impact on the robot's performance is very hard and influenced by many factors, like the spatial layout of the environment (the longer the distances, the more helps a container) or the properties of objects (stacking plates usually works, stacking cups is often problematic).

In the past years, we have developed several techniques tackling the problem of generating plans for complex household tasks, in particular

- a system for generating plans from natural-language task descriptions on web sites like ehow.com [2],
- methods for projecting and debugging these plans to infer missing information [3], and
- TRANER, a planning system, can modify high-level plans using a set of transformations [1].

With these modules, robots can autonomously create plans for new tasks, fix plan flaws, and apply transformations to optimize the performance of a plan. What is missing is a method for efficiently determining which transformations to apply in which order to achieve good performance.

In this paper, we propose to combine these techniques and use observations of humans to prime the selection of the plan that is to be executed. The robot learns probabilistic models characterizing the behavior produced by its plans and matches these models against the observed track of human actions to find a plan that best matches the human behavior.

The plan that is found this way will not necessarily be optimal for the robot. Different physiognomy and dexterity of robot and human may have the effect that a slightly different plan performs best. However, by priming the exploration with human data, the robot can start from a much better initial plan, which it can then further optimize by looking at similar ones. This technique eliminates many options that are in principle possible, but are very unlikely to work well, like stacking plates on top of a stack of cups.

In addition, the approach has two other advantages: The robot is more likely to show human-like behavior, which is often considered comforting. And, a more technical reason, much of the computation required for selecting a plan is moved from the time of action execution to the time of the plan generation. This means, the robot can pre-generate a set of plans while still in the factory or during idle times, for example at night, and quickly adapt its behavior when needed.

The remainder of the paper is organized as follows: After an overview of related work, we will introduce the main concepts used and explain the architecture of the proposed system. We will then discuss the transformational planning system, the behavior representation using Conditional Random Fields and the action observation system. Afterwards, we evaluate the system empirically based on synthetic and real data.

II. Related Work

During the past years, the development of robotic systems that learn complex high-level activities from very few examples has become more and more important. To be successful, learning has to be performed on multiple levels, not just as low-level statistical learning.

One kind of related systems are planners that consider advice taken from humans or other robots when creating plans. [4] work on improving the problem solving process by cooperating with humans or other robotic agents. The advisable planners by [5] take advice from a user while elaborating a plan and add semantic information about the planning elements to obtain qualitatively different plans [6]. That system also comprises various plans with substantially different behavior to achieve the same task, comparable to the several variants of plans for setting the table in our system. Instead of using advice given by humans, we aim at improving the robot plans by non-intrusively learning from observation in our system. Work on using heuristics to reduce the search space in planning has been done for example by [7] for constraint satisfaction problems or by [8] who integrate genetic programming into a classical AI planner to improve the planning process.

Our approach can also be seen in the context of imitation learning, if we pose "learning" as "adapting one's own behavior based on observations of others". Imitation learning is commonly applied to motions like learning hand- and arm trajectories, for example by [9], [10] or [11]. To our knowledge, the current system is the first one that imitates high-level activities like setting a table.

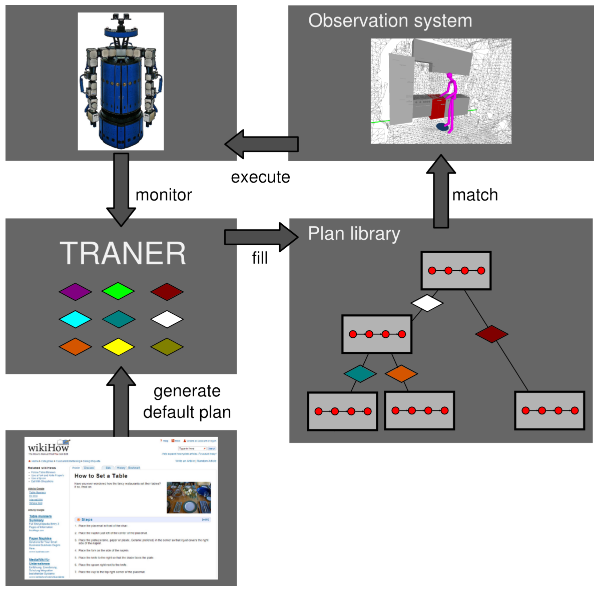


Fig. 2. The main components of the proposed system. Plans are generated, transformed and stored in the plan library. When they are to be executed, the system selects the plan that best matches observed human behavior.

III. Overview

A. Nomenclature

To avoid confusions, we will briefly sum up some of the main concepts used in this paper. A *plan* is a specification of actions to perform in order to achieve a *goal*. The execution of plans results in observable *behavior*, which includes the sequence of actions performed, the action parameters like grasp types, the objects that were manipulated etc.

Different plans may result in the same behavior. Especially, different agents (like humans or robots) will most probably follow different plans, but can show the same (or very similar) behavior.

Transformation rules transform one plan into another, thereby changing the produced behavior. By applying transformation rules, the planning system generates a set of *plan variants*. These plan variants all aim at achieving the same goal, but can do this in different ways, showing significantly different behavior.

B. System Architecture

The main components of the system are described in the overview picture in Figure 2. The whole process starts with the generation of a default plan (lower left corner). This initial plan may be manually created or automatically generated from web sites as demonstrated in [2] and [3]. These default plans specify the main plan steps to achieve a goal, but usually fail to perform the task efficiently.

To create new variants of a plan, which show different and potentially more efficient behavior, the planner TRANER is equipped with a set of transformation rules, depicted by the diamond-shaped blocks in Figure 2. These rules can be applied in different combinations, leading to a tree of *plan variants* shown in the lower right block in Figure 2. Note that the example plan library contains only one plan, represented by the one tree structure, while real libraries feature several plans for achieving different goals. Each node in this tree corresponds to one variation of this plan, each edge to a transformation rule that translates the parent plan into the child.

Branches in the transformation tree can be linked to each other, effectively creating a transformation graph. When the system finds out that two sequences of transformations resulted in identical code (e.g. the application of first transformation A, then transformation B, and vice versa), it can fuse these nodes. This is important not to be stuck in one branch while optimizing the plan determined by observing humans.

Whenever it has generated a new plan variant, TRANER projects it using the methods described in [3]. In contrast to that system, we do not need very detailed information about the action execution and the belief state of the robot, some basic data about the execution, like the sequence of performed actions, the objects that are manipulated and the approximate positions where the actions occurred, is sufficient. Based on these data, the system learns models describing the behavior produced by the different plan variants and also stores them in the plan library.

The whole plan library, including the attached models, can be generated off-line, either before the robot is deployed or, if the plan is generated from web instructions, during the night or other idle times. Though both the generation of a plan and the learning of the model are rather quick and only take a few seconds, off line computation is attractive due to the sheer number of plan variations produced even by rather small numbers of transformations.

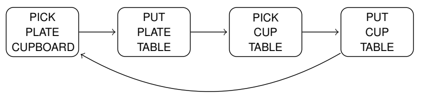
Once the robot has collected observations of humans performing the tasks which it is supposed to later do itself, it can use the models in the plan library to select a plan variant whose behavior best matches the observed one. Due to the tree-like structure, the sequence of transformations that has to be applied to the default plan can easily be determined as the path from the root to the respective leaf.

The planner can then execute the plan, in reality or in simulation, and evaluate the execution. If problems are detected or if the performance is insufficient, the robot can explore the plan space around the observed plan to find a better plan. As opposed to the previous system, however, it starts from a potentially much better position.

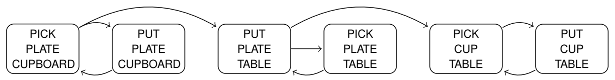
IV. TRANSFORMATIONAL PLANNING

Transformational planners generate new plans by modifying existing ones by rearranging the plan steps, inserting or removing actions or changing action parameters. This is especially useful when basic plans exist that are to be optimized for operation in a certain environment, like, in our case, plans imported from the web.

A problem with transformational planning, however, is the high branching factor of the search tree, i.e. the tree of transformed plans. Often different transformations are applicable, and each of them creates a new branch of plan variants. Evaluating each transformed plan to assess its performance can quickly become infeasible. Therefore, we propose to prime this selection process with observations of human behavior and evaluate only plans in an environment around the selected one.



(a) default plan



(b) plan for stacking plates

Fig. 3. Two state automata visualizing the behavior generated by (a) the default plan and (b) a plan where plates are stacked. The difference in behavior created by the transformation is obvious: a long sequence of actions is split up, groups of stacking and de-stacking actions emerge.

Plans are modified by applying transformation rules. An applicability condition specifies which plans a rule can be applied to, the rule describes how the plan is to be changed:

$$\begin{aligned} OutputPlan \leftarrow \quad & InputSchema: \\ & ApplicabilityCondition \end{aligned}$$

Transformation rules can significantly change the behavior, e.g. re-arrange actions, insert actions for using containers to transport items, or move actions like opening or closing a cupboard to the beginning and end of an action sequence.

The plan library stores the transformed plans until they are executed. Each plan has a model attached that can be used to decide if observed behavior matches the one generated by that plan. If that is the case, this plan will be chosen for execution. If no matching plan can be found, the system selects the default plan for a task.

The system builds on the transformational planner TRANER and extends it with modules for human and robot action observation and for matching observed actions against plan variants. A detailed description of TRANER can be found in [1].

V. BEHAVIOR REPRESENTATION

We model behaviors produced by plans on the abstraction level illustrated in Figure 3. At this level of abstraction, the behavior generated by humans and by robot plans becomes very similar and can be matched against each other.

Assume we have two sets of plan variants for one or several plans: $\mathcal{P}_r = P_r^0, ..., P_r^k$ for the robot, $\mathcal{P}_h = P_h^0, ..., P_h^l$ for the human. Human plans are in general not observable, so the robot cannot directly match the plans, but has to compare them based on the behavior they produce. Let thus $\mathcal{B}$ be the set of behaviors that can be produced by the plans in $\mathcal{P}_r$ and $\mathcal{P}_h$. If a plan variant produces a certain behavior, we denote this by $produces(P^i, B^i)$.

Given that behavior B^i belonging to plan plan P_h^i has been observed, the goal is to find the plan P_r^j that most likely generates the same behavior:

$$\underset{P_r^k \in \mathcal{P}_r}{argmax}(P(produces(P_r^k, B^i))) =: P_r^j \qquad (1)$$

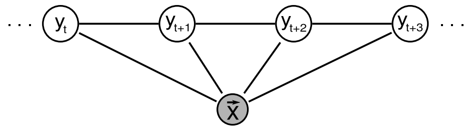


Fig. 4. CRF independence graph.

For this purpose, we use Conditional Random Fields (CRFs) [12]. Conditional Random Fields are undirected graphical models that are conditioned on an observation sequence $\mathbf{x}$. They have some useful properties for this task: As discriminative models, they represent the conditional probability $p(\mathbf{y}|\mathbf{x})$ of a sequence of states given the observation instead of the joint probability $p(\mathbf{y}, \mathbf{x})$ of states and observations. Therefore, in contrast to typical generative models, it is not assumed that the observations are independent and thus allows for using correlated and overlapping features. In the following, we will use the notation of [13].

Let $\mathbf{x} = (\mathbf{x_1}, ..., \mathbf{x_n})$ be the observation and $\mathbf{y} = (\mathbf{y_1}, ..., \mathbf{y_n})$ the sequence of labels assigned to the observation.

The probability is represented as a the normalized product of potential functions ψ_c of the random variables $\mathbf{s_c}$ inside a clique $c \in C$. This yields the general formulation of a CRF:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\prod_{c\in C} \psi_c(x_c, y_c)}{\sum_{y'} \prod_{c\in C} \psi_c(x_c, y'_c)} \quad (2)$$

$$= \frac{1}{Z(x)} \prod_{c\in C} \psi_c(x_c, y_c) \quad (3)$$

$$Z(\mathbf{x}) = \sum_{y'} \prod_{c\in C} \psi_c(x_c, y') \quad (4)$$

The most commonly used form of CRFs are linear-chain CRFs, whose independence graph is shown in Figure 4, and which can be written as the product over the n factors in the factor graph

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(x)} \prod_{j=1}^{n} \psi_j(x_j, y_j, y_{j-1}) \quad (5)$$

If we assume the potential functions to be an exponentiated sum of feature functions $f_i(y_{i-1}, y_i, \mathbf{x}, j)$ that are weighted by the values λ_i

$$\psi_j(\mathbf{x}, \mathbf{y}) = exp\left(\sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, \mathbf{x}, j)\right), \quad (6)$$

we can reformulate the equation of the CRF as the exponentiated, normalized sum of feature functions

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(x)} exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, \mathbf{x}, j)\right) \quad (7)$$

Training CRFs can efficiently be done using a slightly modified form of the forward-backward algorithm that was originally proposed for Hidden Markov Models [14]. For inference, the Viterbi algorithm can also be applied to CRFs.

In our system, the behavior $\mathbf{B}$ of a plan is characterized by its observable events. Each single event is described by four features: The performed action, the type of object manipulated, its identifier, and the place where the action was performed. The observation sequence consists of a list of such actions, e.g. $\mathbf{x} = \{(pick, cup, cup-1, cupboard), (put, cup, cup-1, table), ...\}$.

Each plan variant in the plan library has one linear-chain CRF attached that represents its behavior and that can be used for recognizing the plan. For finding the most likely plan, the system calculates, for all plan variants in the library, the likelihood that the observed sequence belongs to this plan and takes the one with the highest likelihood value. We are not primarily interested in the labels attached to the actions, but rather which of the CRFs accepts the sequence of observed features with the highest probability. Therefore, we simply chose the labels to be the name of the medium-level plan goals like "stack-plates".

VI. Action Observation

In the following two sections, we describe how we collected data about human activities in our kitchen setting and about robot plans in simulation.

A. Human Activity Observation

For the observation of human actions, we used our sensor-equipped kitchen environment (Figure 5). Laser range finders, cameras, magnetic sensors at the doors and several RFID (Radio Frequency IDentification) tag readers allow for the detailed analysis of activities in the environment.

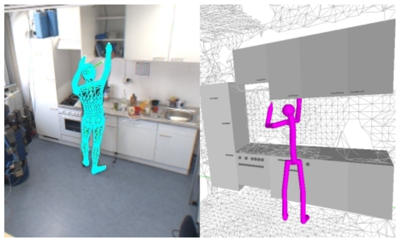


Fig. 5. A human setting the table in our kitchen environment, recorded by a markerless motion capture system and the sensors embedded into the kitchen furniture.

We used two RFID readers, one of them mounted underneath the table, the other one inside the cupboard where pieces of tableware are stored. RFID readers wirelessly identify objects based on a tag that is attached to them. Each tag has a unique number that corresponds to an object. The time-stamped object readings have to be transformed into a sequence of manipulation actions. We assume that objects that appear at a reader have been put down at that position, and that items that are not read any more have been picked up.

B. Robot Action Observation

For collecting the training data, the robot needs mechanisms to record data of the execution of its plans. The system described in [3] allows to do this, either as very detailed data about every aspect of the plan execution based on a realistic

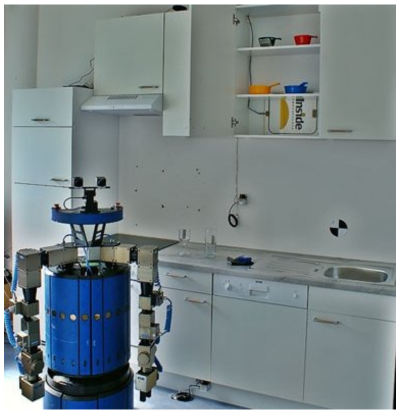


Fig. 6. The B21 robot in the kitchen, with RFID readers visible inside the cabinets.

physical simulation, or in a rather coarse way using simple plan projection.

Since the goal of this system is to use observations of human actions that are obtained without intrusive sensors, we decided to use rather abstract action descriptions, which also have the advantage that they can obtained using the much faster projection, eliminating the need for expensive, time-consuming physical simulation. An excerpt from the data we collect about the different plan variants is shown in Table I.

time/s	action	object type	object	place
16.09	PICK	PLATE	PLATE-1	CUPBOARD
35.41	PUT	PLATE	PLATE-1	TABLE
69.11	PICK	CUP	CUP-1	CUPBOARD
88.23	PUT	CUP	CUP-1	TABLE
124.97	PICK	PLATE	PLATE-2	CUPBOARD
153.60	PUT	PLATE	PLATE-2	TABLE
189.86	PICK	CUP	CUP-2	CUPBOARD
216.42	PUT	CUP	CUP-2	TABLE

TABLE I

EXAMPLE OF THE DATA USED FOR TRAINING THE CRF.

VII. EMPIRICAL RESULTS

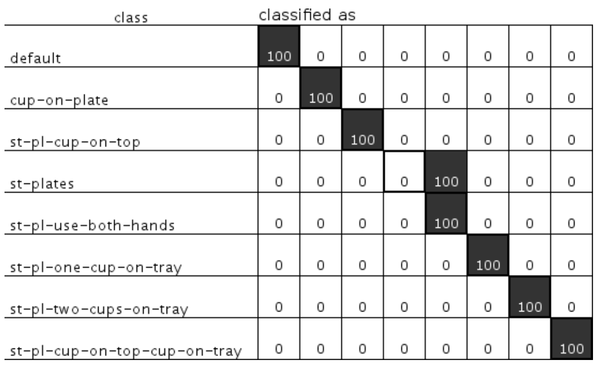
We evaluated our system both on synthetic data and on data recorded from observing humans setting a table in different ways. The first part of this section deals with the evaluation of the classification of human actions, while the second one discusses the performance improvement obtained by selecting the plan observed from humans. In the experiments, we used the set of plans in Table II which are in more detail described in [1].

Note that the recognition of different plan flavors is very challenging because all these plans comprise the same actions, locations, and more or less the same objects that are manipulated in a very similar order. The only way to distinguish the different plans is to check for slight difference in the sequence.

A. Evaluation on synthetic data

We first evaluated the system on synthetically generated data to test how it handles the two main sources of noise: First, RFID tags are detected in a random order when a whole stack of objects is being picked up or put down at once. We simulated this by randomly sampling different event sequences for these cases. Second, the object instance that is manipulated is undetermined until runtime, only its type is specified in the plan. Therefore, we sampled different orders of object instances.

For the experiments, we sampled distinct sets of data, trained the CRF on one of them and evaluated it on the unknown test set. The results of the classification are shown in the confusion matrix in Figure 7.



class	classified as							
default	100	0	0	0	0	0	0	0
cup-on-plate	0	100	0	0	0	0	0	0
st-pl-cup-on-top	0	0	100	0	0	0	0	0
st-plates	0	0	0	0	100	0	0	0
st-pl-use-both-hands	0	0	0	0	100	0	0	0
st-pl-one-cup-on-tray	0	0	0	0	0	100	0	0
st-pl-two-cups-on-tray	0	0	0	0	0	0	100	0
st-pl-cup-on-top-cup-on-tray	0	0	0	0	0	0	0	100

Fig. 7. Evaluation results on synthetic data.

It can be seen that the system is well able to discriminate between the almost all plan variants. Only the plans for stacking plates with or without using both hands are confused. Since we did not collect any data that indicates which objects are held in which hand, these two plans are very hard to distinguish.

B. Evaluation on Observations of Humans

For evaluating the performance in a real-world setting, we collected data from RFID readers in our test kitchen. Since the collection of data with human subjects is rather time consuming and error-prone when the subjects are told to follow a plan they would not intuitively choose (like the `default` plan), we decided to collect data for only four of the plans, while still training on the full set of plans in the library. These four plans were `default`, `st-plates`, `st-pl-cup-on-top` and `st-pl-use-both-hands`.

In the real kitchen, the subjects were told to set the table for two persons, either in a natural way (which always resulted in the equivalent to the `stack-both` plan, or by acting as if they were following one of the other three plans.

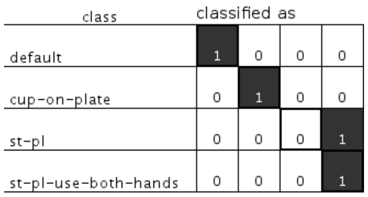
Observing events for more than two persons turned out to be difficult due to the maximum number of objects our

Name	Description
default	transport objects one by one
cup-on-plate	stack one cup on a plate
st-plates	stack all plates
st-pl-cup-on-top	stack all plates, one cup on top
st-pl-use-both-hands	stack plates, carry cups in both hands
st-pl-one-cup-on-tray	stack plates, one cup on the tray
st-pl-two-cups-on-tray	stack plates, two cups on the tray
st-pl-cup-on-top-cup-on-tray	stack plates, on cup on the stack, one on the tray

TABLE II

PLANS USED FOR THE EVALUATION OF THE SYSTEM.

RFID tag readers can detect at a time, so we limited the number of people to set the table for to two. RFID tag readers sometimes have short interruptions in the object readings, especially when many objects or metallic items are in the detection range. A low-pass filter effectively eliminates these erroneous mis-readings in a pre-processing step.



class	classified as			
default	1	0	0	0
cup-on-plate	0	1	0	0
st-pl	0	0	0	1
st-pl-use-both-hands	0	0	0	1

Fig. 8. Evaluation results on human observations.

The results of the evaluation in Figure 8 show that the system confused the plan it also confused in the synthetic data, but is still able to reliably detect three of the four plans.

C. Robot performance improvement

A good combination of plan transformations significantly improves the robot performance. Stacking plates and using both arms for carrying cups speeds up the plan executing by up to 15.6%. This is exactly the plan flavor that was intuitively executed by the human subjects. Further results showing how plan transformations enhance robot plans and discussing the influence of different environments can be found in [1].

In TRANER, the best combination of plan transformations was determined by exhaustive search, i.e. all possible combinations of transformations were simulated several times and evaluated. This approach is very likely to find the best plan, but needs a lot of time since the simulation normally cannot be done much faster than real time. Evaluating each plan in the whole tree quickly becomes prohibitive with a growing number of transformation rules.

The system presented in this work requires no expensive simulation, but only a rather simple projection of the plan since the features we use do not depend on the environment. Applying the transformations and learning the models can be done in few seconds, which leads to significantly improved performance of the transformational planning system.

VIII. CONCLUSIONS

In this paper, we presented a method to extend a transformational planning system with methods for learning models characterizing the behavior produced by the robot plans. Using the system, the robot can match observed human behavior against the plans in its plan library in order to find the plan that best resembles the human way of performing the task.

We propose to use Conditional Random Fields to match the behavior produced by the robot plans to that observed from humans. These probabilistic models are stored in the plan library and allow the robot to quickly select a plan amongst the various alternatives.

Technically, this approach offers the advantage of massively speeding up the selection process, which previously suffered from severe scalability problems. Regarding its applications, the system can be seen as a method for imitating complex human behavior on an abstract level.

We see this system as an important step towards robots that can act naturally in human environments. To do this, they have to quickly adapt their behavior to perform tasks in an efficient way, while acting as human-like as possible. The methods proposed in this paper help robots approach this goal.

IX. ACKNOWLEDGMENTS

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*, see also `www.cotesys.org`.

REFERENCES

[1] A. Müller, A. Kirsch, and M. Beetz, "Transformational planning for everyday activity," in *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, Providence, USA, September 2007, pp. 248–255.

[2] 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.

[3] L. Mösenlechner and M. Beetz, "Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior," in *19th International Conference on Automated Planning and Scheduling (ICAPS'09).*, 2009.

[4] J. Allen, N. Blaylock, and G. Ferguson, "A problem solving model for collaborative agents," *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pp. 774–781, 2002.

[5] K. Myers, "Domain metatheories: Enabling usercentric planning," *Proceedings of the AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems*, 2000.

[6] K. Myers, T. Lee, and T. Lee, "Generating qualitatively different plans through metatheoretic biases," *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence table of contents*, pp. 570–576, 1999.

[7] S. Minton, "Integrating heuristics for constraint satisfaction problems: A case study," *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 120–126, 1993.

[8] R. Aler, D. Borrajo, and P. Isasi, "Evolving heuristics for planning," *Proceedings of the Seventh Annual Conference on Evolutionary Programming, Lecture Notes in Artificial Intelligence, San Diego, CA, March*, 1998.

[9] S. Schaal, "Computational approaches to motor learning by imitation," *Philosophical Transactions: Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[10] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng, "Discovering optimal imitation strategies," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 69–77, 2004.

[11] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, vol. 36, no. 5, 2006.

[12] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *Proc. 18th International Conf. on Machine Learning*, pp. 282–289, 2001.

[13] R. Klinger and K. Tomanek, "Classical probabilistic models and conditional random fields," Technische Universität Dortmund, Dortmund," Electronic Publication, 2007.

[14] L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77, Issue 2, February 1989, pp. 257–286.