Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots

LEARNING SEQUENTIAL CONSTRAINTS OF TASKS FROM USER DEMONSTRATIONS*

Michael Pardowitz, Raoul Zöllner, and Rüdiger Dillmann

Institute of Computer Science and Engineering
Universität Karlsruhe (TH)
Karlsruhe, D-76128, Germany, P.O. Box 6980
{pardow|zoellner|dillmann}@ira.uka.de

ABSTRACT

Learning from human demonstration is likely to be one of the key features for service robots in household domains if they are to be accepted by humans. To be of most benefit possible to its user, the robot should go beyond simply imitating a user's demonstration but try to build task knowledge that is as general and flexible as possible. One way to achieve this is equipping the robot with the ability to reason about the task knowledge he has already acquired in order to refine, generalize and complete it.

A system to record and interpret manipulation task demonstrations is presented in this paper. As a representation for the sequential constraints a valid task execution must obey, task precedence graphs (TPG's) are introduced. Means of reasoning on a task's underlying TPG are proposed and evaluated within two tasks from the household domain.

Index Terms— **Learning manipulation tasks, programming by demonstration (PbD), reasoning on tasks**

I. INTRODUCTION

During the last years, humanoid robotics became a major trend in the robotics community. One crucial point in human-like machines is to design systems that learn the knowledge the user has and transform it into knowledge utilizable by a humanoid service robot. A major field of machine learning in robotics is the acquisition of task knowledge in order to spread the robot's functionality and usefulness. One of the most intuitive ways to acquire new task knowledge is to learn it from the human user via demonstration and interaction. This approach to task learning is widely known as *Programming by Demonstration (PbD)*.

A crucial point in task learning is to exactly capture the user's intention. Usually this is achieved by observing multiple demonstrations of the same task and identifying the common features of the task as the user's inherent intention. On the other hand, initially requesting multiple demonstrations of a single task would annoy the user. Therefore, PbD-systems should be capable of learning a task from a single demonstration in order to allow first executions, monitored by the user. Incremental learning approaches that gradually refine task knowledge and generalize it as more demonstrated examples become available pave the way towards suitable PbD-systems for humanoid robots.

One aspect that can only be learned incompletely from a single user demonstration is the sequence the subparts of a certain task can be scheduled. Task knowledge should allow the robot to chose its sequence of actions from a large set of possibilities. On the other hand, some parts of tasks offer no choice of the sequence in which they can be performed. So the task knowledge must explicitly encode those in order to ensure a reliable, faultless and safe execution of the task.

After seeing a single demonstration, PbD-systems can state multiple valid hypotheses on the sequential constraints a task must obey. When more demonstrations become available, identical subtasks have to be identified and related to the subtasks in other demonstrations. From these the sequential structures of the new demonstrations can be deduced in order to prune or refine the sequential hypotheses.

The remainder of this paper is organized as follows: The next section gives an overview on related work concerning programming by demonstration and task learning from user demonstrations. Section III describes the system for the acquisition of task knowledge from a single user demonstration, viewing a task as a simple sequence of actions. Section IV introduces task precedence graphs, the representation for the sequential structure of a task. Section V proposes a method for reasoning on the underlying precedence graph of a task with two or more sequential demonstrations given. The methods for identifying corresponding subtasks within different task demonstrations described in section VI are an important preliminary for this computation. Finally, the methods desscribed in earlier sections are evaluated in section VII.

*THIS WORK HAS BEEN PARTIALLY CONDUCTED WITHIN THE GERMAN COLLABORATIVS RESEARCH CENTER "SFB HUMANOIDE ROBOTER" GRANTED BY DEUTSCHE FORSCHUNGSGEMEINSCHAFT, AND WITHIN THE EU INTEGRATED PROJECT COGNIRON ("THE COGNITIVE ROBOT COMPANION"), FUNDED BY THE EUROPEAN COMMISION DIVISION FP6-IST FUTURE AND EMERGING TECHNOLOGIES UNDER CONTRACT FP6-002020.

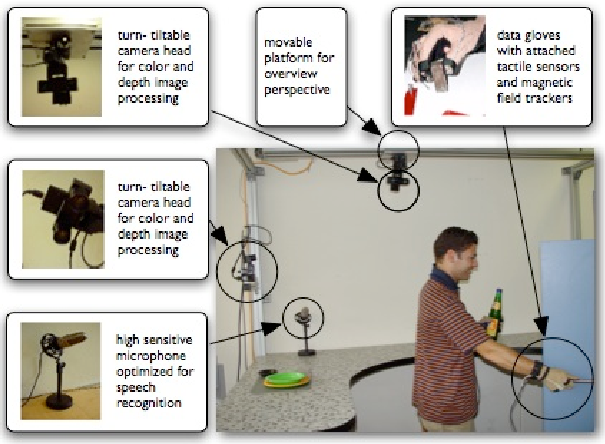

turn- tiltable camera head for color and depth image processing
movable platform for overview perspective
data gloves with attached tactile sensors and magnetic field trackers
turn- tiltable camera head for color and depth image processing
high sensitive microphone optimized for speech recognition

Fig. 1. Training center with dedicated sensors

II. RELATED WORK

During the past years, programming by demonstration and task learning systems have received increased attention. Different approaches have been proposed and have been reviewed in [1], [2]. Systems can be discriminated by the learning paradigm applied.

Imitation learning systems are used to re-identify and execute human motions [3], aiming at a high similarity of the robot's movements to the demonstrated trajectories [4]. These systems require a large set of task demonstrations for generalizing the trajectory before the learning process can start.

Background knowledge based or deductive PbD-systems, as presented in [5], [6], [7], usually require much less or even only a single user demonstration to generate executable task descriptions. These systems analyse and interpret the task demonstration with respect to the changes and effects the user's actions affect the environment. When mapping the learned task to the executing system, background knowledge based approaches are used in order to replicate the effects in the environment [8].

Sequential analysis of tasks is presented in [9], [10]. The first records multiple user demonstrations of a complex manipulation skill. Unnecessary actions that do not appear in all demonstrations are pruned and only the sequence of essential actions is retained. The latter stresses the role of interaction with the human user to facilitate learning of sequential arrangement of behaviors in a navigation task.

III. ACQUIRING OPERATION SEQUENCES FROM USER DEMONSTRATIONS

User demonstrations featuring a task to be learned by the system are observed and analysed by a Programming by Demonstration system developed at our institute in recent years. This section gives a short overview of the task acquisition process and the classes of manipulation tasks that can be detected and processed by our system.

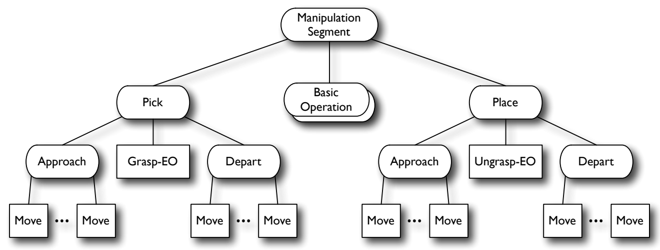


Fig. 2. Hierarchical representation of manipulation segments

During the performance of a task in a so called training-center, the user is observed using different sensors. These sensors include two data gloves for gathering the finger joint angles, magnetic trackers for obtaining the hands' position, and two active stereo cameras mounted on pan-tilt-units for object localisation and tracking (see figure 1). Further details on the hardware used can be found in [11].

From the sensor input data, so called elementary operators are extracted. Elementary operators (EO's) are actions that abstract a sensory motor coupling like primitive movement types (linear moves etc.), grasping and ungrasping actions. These EO's abstract from the specific execution hardware and are robot-independent from a system point of view, although they have to be implemented on the specific target robot system. EO's are aggregated as primitives to so called macro-operators (MO's), containing the full task description. On a basis level, elementary move operators are aggregated to approach, depart and transport trajectories. Together with the grasp and ungrasp EO's, grabbing and releasing of objects can be constructed. Between those gripper operations, various basic manipulation operations are detected (like transport operations, pouring operations and tool handling for instance). For further details, please refer to [7]. On the highest level, a sequence of manipulation segments is subsumed under a whole task demonstration (see figure 2).

On each level in this hierarchical task representation, the changes to the environment are induced from lower levels of the hierarchy. The pre- and postconditions describing the environmental states before and after the execution of each macro-operator are propagated from the EO-level to the manipulation segment level and are computed from the environmental changes during the manipulation. The environment, its changes and the pre- and postconditions are described in terms of first order predicate logics, based on a set of geometrical inter-object relations (like "on", "under", "to the right of..." or "contained in ...") and intra-object predicates like object class ("saucer", "plate", "table") and internal state descriptors ("opening angle", "oven temperature", "liquid level", etc., depending on the object class).

The system covers a broad range of manipulation classes drawn from the operations needed in a household environment: Transport operations (Pick&Place), device handling (operating household devices, opening a fridge) and tool handling (pouring liquids from a bottle to a glas, unscrewing a bottle etc.). It depends on the condition that the user is doing all changes in the environment (closed world assumption).

The result of the task acquisition process outlined in this section is a task description, containing a sequence of manipulation segments together with their pre- and post-conditions and the hierarchical decomposition into elementary operators. The manipulation segments are the basic building blocks of a task and will also be called operations and subtasks in the following sections.

IV. TASK PRECEDENCE GRAPHS

This section is concerned with the representation of the sequential features of a task. A formal definition of the structure that contains the sequential ordering a task obeys is given. Additionally, a way a system can make hypotheses about the sequential dependencies is presented.

When a user performs a task, he performs its subparts (the manipulation segments) in a sequential ordering that he has chosen by random or by intent from all possible task execution orders. For a system recording his actions these appear as a simple sequence of operations. In order to exploit the maximum degrees of freedom the task posesses at execution time, the sequential constraints a valid task execution has to obey must be known to the robot before the execution can start. Observing a single demonstration of a certain task, the sequence chosen by the user is influenced by two magnitudes:

- *Sequential dependencies* induced by the task to be solved. These form temporal precedence relations that follow from the attributes of the task to be done and environmental constraints. One simple example is the task of fetching an object from the fridge. The subtask of opening the fridge door must be accomplished before a robot can pick the object.
- *Sequential execution of independent operations*. Operations sequentially independent of each other that can not be performed in parallel have to be executed in any order. This order may be chosen by user's preferences or according to any strategy or optimization criteria.

A learning system that builds knowledge about a task has to make hypotheses about the underlying sequential task structure. These hypotheses can be represented by task precedence graphs.

Definition 1: A task precedence graph (TPG) for a task T is a directed graph $P = (\mathbf{N}, \mathbf{R})$ with $\mathbf{N}$ being the set of subtasks $o_1, o_2, \ldots, o_n$, and $\mathbf{R} \subset \mathbf{N} \times \mathbf{N}$ being the set of precedence relations a faultless task execution must comply with. A precedence relation $(o_1, o_2) \in \mathbf{R}$ with $o_1, o_2 \in \mathbf{N}$ implies that the operation o_1 must be complete before the execution of o_2 can start. This is abbreviated as $o_1 \rightarrow o_2$.

A faultless execution of a task requires the chosen sequence of operations to be consistent with its TPG, or, in other words, fulfills every sequential relationship inherent to the TPG.

Definition 2: A demonstration $D = (o_{i_1}, o_{i_2}, \ldots, o_{i_n})$ with $o_{i_j} \in \mathbf{N}$ is said to comply with a task precedence graph $P = (\mathbf{N}, \mathbf{R})$, if for every precedence relation $o_i \rightarrow o_j \in \mathbf{R}$ the operations $o_i = o_{i_k}$ and $o_j = o_{i_l}$ appear in the correct sequential order, that is $k < l$. This is denoted by $P \rightsquigarrow D$.

For a system that is supplied with only a single user demonstration $D = (o_1, o_2, \ldots, o_n)$ of a task, it is hard to guess the inherent task precedence graph. Suppose that an operation o_i is observed before o_j. Then, $o_i \rightarrow o_j$ can be part of the TPG, indicating that in every valid task execution sequence o_i must appear before o_j, or this observation can result from the fact that the user had to chose any ordering for two sequential independent actions (see above). The learning system can state different hypotheses, ranging from the most restrictive TPG $P^D = (\mathbf{N}, \mathbf{R}^D)$ with

$$\mathbf{R}^D = \{(o_i, o_j) | i < j\}, \tag{1}$$

to the TPG with the most degrees of freedom, $P^* = (\mathbf{N}, \emptyset)$. P^D restricts the task to be executable only with the sequential ordering observed in the user demonstration, P^* classifies every order of actions as a valid task sequence. All other possible TPG's the user demonstration complies with are valid as well. In order to impose a structure on this set of valid hypotheses, the "more general" partial ordering is defined.

Definition 3: A TPG $G = (\mathbf{N}, \mathbf{R}_G)$ is said to be *more general than* a TPG of the same task $S = (\mathbf{N}, \mathbf{R}_S)$ if and only if $\mathbf{R}_G \subset \mathbf{R}_S$. This is abbreviated as $G \succ S$.

V. INCREMENTAL LEARNING OF TASK PRECEDENCE GRAPHS

The last section stated that multiple valid hypotheses on the sequential constraints of a task can exist. This sections deals with the topic of how this set of valid hypotheses can be further reduced in size.

While the learning system can not decide which task precedence graph from the set of consistent TPG's fits the task best after seeing only one single example, it seems a viable approach to supply it with more sample demonstrations, applying different task execution orders. In order to learn task knowledge from even a single demonstration sufficient for execution but improving the learned task when more knowledge in form of task demonstrations is available, an incremental approach is chosen.

Assuming that the system has learned the most specific task precedence graph P_m after obtaining a set of m Demonstrations $\{D_1, D_2, \ldots, D_m\}$, a new demonstration of the same task D_{m+1} is observed. The next step is to adapt the learned task precedence graph in a way that incorporates the new knowledge. [12] suggests that this can be done by further generalizing the previous hypothesis to a new one, covering the additional example. In order not to generalize too far, that is, to ensure that no essential precedence relations are dropped, the minimal generalization of P_m that is consistent with D_{m+1} must be chosen. So one can state that the best choice for P_{m+1} is the hypothesis H with

$$H \succ P_m \wedge H \rightsquigarrow D_{m+1} \wedge (\nexists H' : H' \rightsquigarrow D_{m+1} \wedge H \succ H' \succ P_m) \tag{2}$$

In computation terms, the new set of task precedence relations $\mathbf{R}_{m+1}$ can be expressed as a function of the previously learned hypothesis $P_m = (\mathbf{N}, \mathbf{R}_m)$ and the most restrictive hypothesis for the new observed demonstration $P^{D_{m+1}} = (\mathbf{N}, \mathbf{R}^{D_{m+1}})$, which can be computed according to eq. 1:

$$\mathbf{R}_{m+1} = \mathbf{R}_m \cap \mathbf{R}^{D_{m+1}} \tag{3}$$

Now, one can state the process of incremental learning of task precedence graphs:

1) For the first training example D_1, initialize the task precedence graph $P_1 = P^{D_1}$ according to equation 1.
2) For each additional demonstration D_{m+1} of the same task, compute $P^{D_{m+1}}$ and update the hypothesis P_{m+1} according to equation 3.

One issue not adressed until now is the question of when the additional task demonstrations arrive. In the application domain of household service robots one cannot assume that the user gives all demonstrations sufficient to learn the correct task precedence graph at once. Instead, it might take a long time between two successive demonstrations of the same task. Moreover, the task knowledge learned during past demonstrations has to be utilised because it is likely that the user will require the robot to execute the learned task without having taken into account all task demonstrations that might appear in the future. Therefore, the task precedence graph learned by the system should be reliable enough to ensure a faultless and, above all, secure task execution.

The incremental learning mechanism for task precedence graphs presented in this section allows a correct precedence graph to be learned from even a single example while still maintaining the ability to incorporate new knowledge in order to refine the task knowledge and to provide additional reordering opportunities at execution time.

The user is given the possibility to provide the learning system with another task demonstration, when during a robot's task execution he finds out, that the learned task precedence graph is to restricted to meet his intention.

VI. SUBTASK SIMILARITY MEASURES

While the last section presented a method for learning the sequential constraints of a task when in every task demonstration exactly the same subtasks are used, this is not likely to be true for every task. Usually the human demonstrator will only use similar but partly different manipulation segments in order to fulfill the same task. This section deals with the topic of identifying the corresponding manipulation segments in two or more demonstrations of the same task.

In order to identify the matching manipulation segments for two different task demonstrations, the ordering of the manipulations can not be a reliable measure for subtask correspondence as the sequence of subtasks performed is potentially permuted. Though matching the segments that manipulate the same class of objects seems to be a good idea at first, this method fails as soon as there are multiple objects of the same class present in the scene. So more features should be taken into account to determine the similarity of subtasks and establish sufficiently robust subtask corespondences in order to identify operations of equal impact to the scene.

The features of a manipulation segment are organized along the following classification:

- *Object features:* These contain the class of the objects manipulated or referenced in the certain subtask (cup, plate, table etc.) as well as their possible functional roles (liquid container, object container etc.).
- *Pre- and Postcondition features:* These contain the geometrical relations that exist between the objects before or after the performance of the subtask respectively.

Note that the correspondence of pre- and postcondition features depends on the object correspondence, as when objects are not matched correctly, the geometrical relations between them will be less accurate. For this reason, a two-step approach is applied: First the best object match is determined and the pre- and postcondition similarity is calculated afterwards.

The base operation to gain corresponding features is the measure of similarity s_F for two sets of features $\mathbf{A}, \mathbf{B}$ with

$$s_F(\mathbf{A}, \mathbf{B}) = \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A} \cup \mathbf{B}|}.$$

This is the portion of common features in all features in both feature sets. In order to compute the best match of corresponding objects, the object permutation is chosen that maximizes the similarity of object features. With two sets of object's features $\mathbf{O}_1, \mathbf{O}_2$ the best object correspondence

$$p_{obj} = \arg \max_{p \in perm(obj)} s_F(\mathbf{O}_1, p(\mathbf{O}_2))$$

with p_{obj} being a permutation of the objects *obj* is chosen and the object similarity amounts to $s_{obj} = s_F(\mathbf{O}_1, p_{obj}(\mathbf{O}_2))$. The optimal permutation can be computed using the standard bipartite graph matching algorithm.

With the correct object correspondence, the similarities s_{pre} and s_{post} of the pre- and postcondition sets can be computed using s_F. The overall similarity s_M of two manipulation segments M_1, M_2 is defined as the weighted average of the object-, pre- and postcondition similarity: $s_M = \alpha_{obj} s_{obj} + \alpha_{pre} s_{pre} + \alpha_{post} s_{post}$. In the experiments conducted in this paper all weights were normalized to $1/3$. Once the subtask-similarity s_M is computed for every pairing of the task's subtasks, the subtask correspondence p_{ST} is computed as the maximum sum of similarities:

$$p_{ST} = \arg \max_{p \in perm(subtasks)} \sum_{(M_1, M_2) \in p} s_M(M_1, M_2).$$

With the subtask permutation p_{ST} the most restrictive hypothesis P^{D_i} (see section V) on the underlying task precedence structure can easily be constructed. This hypothesis can then be utilised to learn sequential task constraints in the way described in section V.

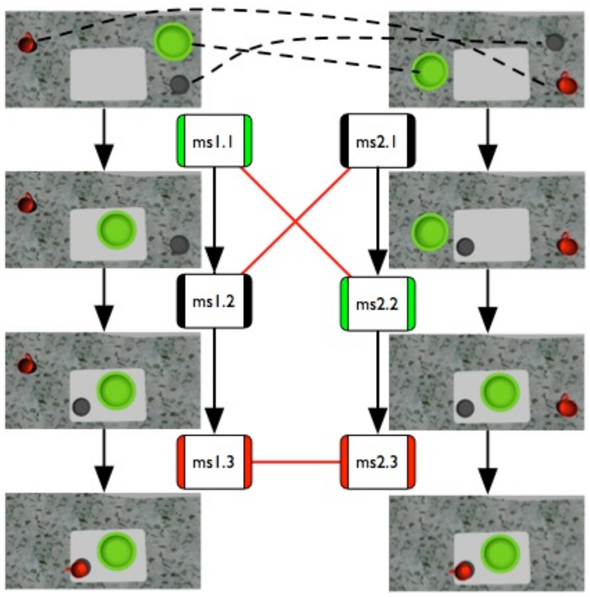


Fig. 3. Recognition of corresponding task features for the first experiment. Object correspondences are drawn with stipled, subtask correspondences with solid lines. The sequential ordering of the subtasks is represented by bold arrows.

VII. EXPERIMENTS

In this section two experiments validating the method for incremental learning of TPG's are reported. The tasks to be learned were chosen from the household domain; two tasks for laying a table were selected.

The first task, a very simple example, was chosen to show how the method works in detail. The task consists of laying the table with a plate, a saucer and a cup to be placed on the saucer. The placing of the plate is (sequentially) independent, while the cup depends on the saucer to be placed first. In a first demonstration the plate, the saucer and the cup are placed in this order. The initial task precedence graph $P^{D_1} = P_1$ can be seen in the first row of table II.

When a second demonstration of the same task was given, the user chose another sequence: He first placed the saucer, then the plate and last the cup. The system correctly recognizes the object- and subtask correspondences with the methods described in section VI. Table I shows the object- and subtask-similarities and the selected correspondences. The results are shown in figure 3.

With the methods described in section V the system was able to learn the sequential independence of the saucer from the plate (See second row of table II). Note that the operation of placing the cup is still assumed to be dependent on the plate.

A last user demonstration is observed with the order saucer - cup - plate. Now the system eliminates the last unnecessary precedence relation and is free to schedule the plate somewhere in the task execution, while still ensuring that the saucer is placed before the cup.

Object similarities:

$D_1 \backslash D_2$	$obj_{2.1}$	$obj_{2.2}$	$obj_{2.3}$
$obj_{1.1}$	0.6	0.5	**1.0**
$obj_{1.2}$	**1.0**	0.5	0.6
$obj_{1.3}$	0.5	**1.0**	0.5

Object correspondence:

$$p_{obj}(o) = \begin{cases} obj_{1.2}, & \text{when } o = obj_{2.1} \\ obj_{1.3}, & \text{when } o = obj_{2.2} \\ obj_{1.1}, & \text{when } o = obj_{2.3} \end{cases}$$

Subtask similarities:

$D_1 \backslash D_2$	$ms_{2.1}$	$ms_{2.2}$	$ms_{2.3}$
$ms_{1.1}$	0.5	**0.85209507**	0.44444445
$ms_{1.2}$	**0.7312238**	0.4814815	0.46226415
$ms_{1.3}$	0.4469496	0.45710456	**0.79991335**

Subtask correspondence:

$$p_{ms}(ms) = \begin{cases} ms_{1.2}, & \text{when } ms = ms_{2.1} \\ ms_{1.1}, & \text{when } ms = ms_{2.2} \\ ms_{1.3}, & \text{when } ms = ms_{2.3} \end{cases}$$

Table I. Object and subtask similarities and correspondences for first experiment.

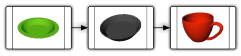
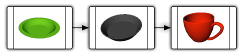
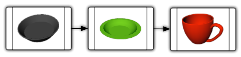
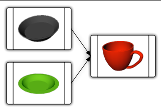
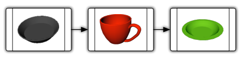
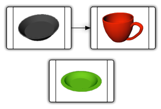
i	P^{D_i}	P_i
1		
2		
3		

Table II. Task precedence graphs learned incrementally during the first experiment consisting of three task demonstrations.

A second experiment, consisting of a more complex example of laying the table is presented in figures 4 and 5. The tasks consists of laying the table with a soup spoon, a large, flat plate under a soup plate, a saucer with a cup on it and a tea spoon in this cup. In the initial object configuration the saucer is on the soup plate, which requires the saucer to be removed from its initial position before the soup plate can be manipulated. Two demonstrations of this task are observed. The sequential order of the performed subtasks is depicted in figure 5. Taking into account these two demonstrations, the system learns the correct task precedence graph, which is presented in figure 6.

VIII. CONCLUSION

This paper presents an approach to combining learning and reasoning on tasks. Tasks are recorded from user demonstrations, segmented, interpreted and stored in a data rep-

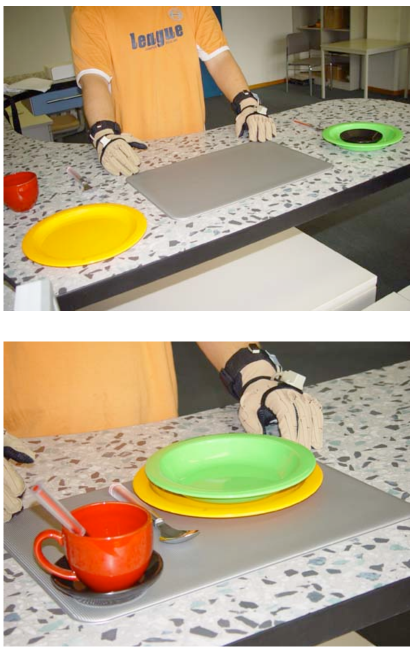


Fig. 4. Start and end state of complex laying the table task performed in second experiment. The task consists of arranging 6 objects on the silver tray.

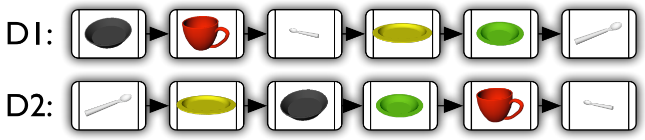


Fig. 5. The two demonstrations performed for the second experiment.

resentation called macro-operators. Reasoning methods are applied to discover the task's underlying sequential structure and reordering possibilities.

This approach presents a step towards robots that learn task knowledge from human demonstrations in an incremental manner that allow them to refine and complete their learned knowledge as additional data becomes available, and act as real 'intelligent' robot servants in future household applications.

IX. REFERENCES

[1] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zöllner, and M. Bordegoni, "Learning Robot Behaviour and Skills based on Human Demonstration and Advice: the Machine Learning Paradigm," in *9th International Symposium of Robotics Research (ISRR 1999)*, Snowbird, Utah, USA, 9.-12. Oktober 1999, pp. 229–238.

[2] S. Schaal, "Is imitation learning the rout to humanoid robots?" in *Trends in Cognitive Sciences*, vol. 3, 1999, pp. 233–242.

[3] S. Calinon and A. Billard, *Learning of Gestures by Imitation in a Humanoid Robot*. Cambridge University Press, 2005, ch. To appear, p. In Press.

[4] A. Alissandriakis, C. Nehaniv, K. Dautenhahn, and J. Saunders, "Using jabberwocky to achieve corresponding effects: Imitating in context across multiple plattforms," in *Proc. of Workshop an the social mechanisms of robot programming by demonstration at ICRA 2005*, 2005.

[5] Y. Sato, K. Bernardin, H. Kimura, and K. Ikeuchi, "Task analysis based on observing handds and objects by vision," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Lausanne*, pp. 1208 – 1213, 2002.

[6] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 6, pp. 799–822, 1994.

[7] R. Zoellner, M. Pardowitz, S. Knoop, and R. Dillmann, "Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration," *Intl. Conf. on Robotics and Automation (ICRA) 2005, Barcelona, Spain*, 2005.

[8] O. Rogalla, "Abbildung von benutzerdemonstrationen auf variable roboterkonfigurationen," Ph.D. dissertation, University Karlsruhe, Departmen of Computer Science, 2002.

[9] J. Chen and A. Zelinsky, "Programming by demonstration: Coping with suboptimal teaching actions," *The International Journal of Robots Research*, vol. 22, no. 5, pp. 299–319, May 2003.

[10] M. Nicoluscu and M. Mataric, "Natural methods for robot task learning: instructive demonstrations, generalization and practice," in *2nd International joint conference on Autonomous agents and multiagent systems, 2003*, Melbourne, Australia, July 2003, pp. 241–248.

[11] M. Ehrenmann, R. Zöllner, O. Rogalla, S. Vacek, and R. Dillmann, "Observation in programming by demonstration: Training and exection environment," in *Humanoids 2003, Karlsruhe/Munich, Germany, October 2003*, 2003.

[12] T. M. Mitchell, "Generalization as search," *Artificial Intelligence*, vol. 18, no. 2, pp. 203–226, 1982.

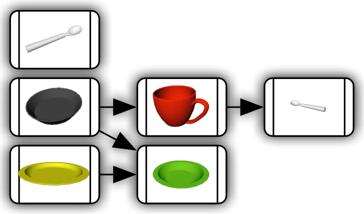


Fig. 6. TPG learned for complex laying the table task in the second experiment. This final TPG is learnt after taking into account only two demonstrations.