IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS 2004), Japan, Sep. 28 - Oct. 2, 2004

# Programming by Demonstration: Dual-Arm Manipulation Tasks for Humanoid Robots

R. Zöllner, T. Asfour and R. Dillmann
Institute for Computer Design and Fault Tolerance
Universität Karlsruhe (TH)
Karlsruhe, D-76128, Germany,P.O. Box 6980
Email: {zoellner, asfour, dillmann}@ira.uka.de

***Abstract*— This paper deals with easy programming methods of dual-arm manipulation tasks for humanoid robots. Hereby a Programming by Demonstration system is used in order to observe, learn and generalize tasks performed by humans. A classification for dual-arm manipulations is introduced, enabling a segmentation of tasks into adequate subtasks. Further it is shown how the generated programs are mapped on and executed by a humanoid robot.**

## I. Introduction

Creating artificial machines that are as versatile and flexible as humans is extremely difficult because the algorithms necessary to teach such machines are not yet clear. We believe that Programming by Demonstration (PbD) is the right way to teach autonomous humanoid robots, which are expected to exist and work together with human beings in everyday environments such as hospitals, offices and households and to serve the needs of elderly and disabled people. The underling idea of Programming by demonstration is to enable the robot to observe a human performing a task, to extract as much as possible information from the demonstration and map it into an abstract, generalized representation in order to make it robot invariant. In particular, the programming and coordination of dual-arm manipulation tasks are a major challenge in the process of developing applicable and profitable humanoid robotic devices for domains less constrained than the industrial ones.
The remainder of this paper is organized as follow: In section II an overview about programming approaches following the Programming by Demonstration paradigm is given. Section III describes the PbD framework and the different phases of a PbD cycles. Section IV addresses the classification, detecting and repesenting of coordinated dual-arm actions. In section V a framework for the coordinated execution of dual-arm manipulation tasks is presented. The execution of a dual-arm manipulation task generated by our PbD system and the experimental setup, using a humanoid robot, are described in section VI.

## II. Related Work

Several programming systems and approaches based on human demonstrations have been proposed during the past years. Many of them address special problems or a special subset of objects only. An overview and classification of the approaches can be found in [1], [2].

Basis for the mapping of a demonstration to a robot system are the task representation and task analysis. Often, the analysis of a demonstration takes place observing the changes in the scene. These changes can be described using relational expressions or contact relations [3], [4].

Issues for learning to map action sequences to dissimilar agents have been investigated by [5]. Here, the agent learns an explicit correspondence between his own possible actions and the actions performed by a demonstrator agent by imitation. To learn correct correspondences several demonstrations of the same problem are necessary for the general case. In [6] the authors concentrate on the role of interaction during task learning. They use multiple demonstrations to teach a single task. After generalization they use the teacher's feedback to refine the task knowledge.

For generalizing a single demonstration mainly explanation based methods are used [7], [8]. They allow for an adequate generalization taken from only one example (One-Shot-Learning). Approaches based on One-shot-Learning techniques are the only ones feasible for end users since giving many similar performing examples is an annoying task.

Research about programming coordinated tasks by demonstration can't be found in the literature, however there are several works on analyzing bi-manual manipulations performed by humans. Many papers [9], [10] in the HCI field have investigated the role of hands performing bimanual tasks. Most of them are based on the early research of Guiard [11], [12]. The outcome of this research is mainly that a relative spacial relation exists between the trajectory of hands performing coordinated tasks. Based on this the hand's interaction during bi-manual actions has been classified in symmetric and asymmetric actions.

## III. PbD Framework

This section will describe briefly the PbD framework developed and used in our institute for many years. For more details please refer to [13] and [14]. Figure 1 shows the PbD-cycle enabling the transfer of task knowledge from the human to a robot target. The main idea is to extract as much information as possible from a human demonstration and transform it into an abstract, generalized representation, in order to make it reusable for different kinds of robots. Based on this approach high level tasks can be learned and adapted to a certain robot configuration.

Hereby high level tasks are composed of basic skills (i.e. sensor motor skills of different complexities), which are called elementary operators (*EO's*). The used symbolic learning techniques relay on explanation based methods enabling the system to learn from one or a few examples. Nevertheless the system integrates also sub-symbolic learning methods like Neuronal Networks or Support Vector Machines for classifying different grasps.

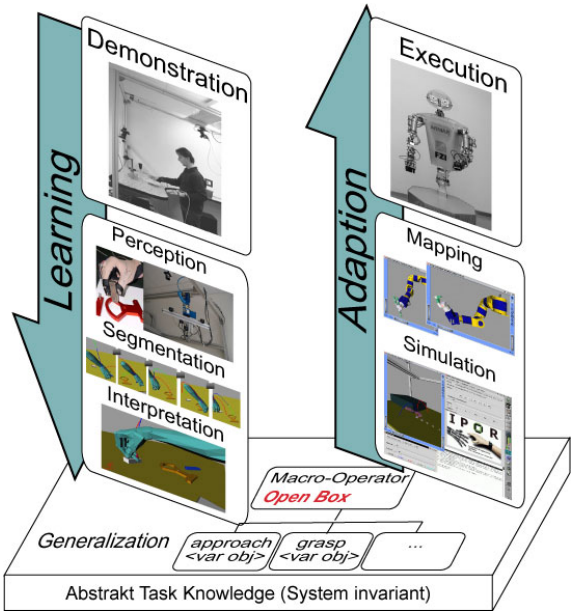


Fig. 1. Programming by Demonstration Cycle from a Human Demonstration to a Robotic Target System.

The PbD-cycle can be divided in three major phases:

- Perception and Interpretation of the demonstrated task
- Generating abstract task knowledge through generalization
- Mapping abstract tasks to specific robot targets.

### *A. Perception and Interpretation of the demonstrated task*

Due to the fact that robots have a limited number of sensors and that these are rather optimized for the execution of tasks than for observing humans, a demonstration of a task is recorded in a separate environment, the so called demonstration area. Here, three active stereo cameras and two data-gloves with magnetic trackers and tactile sensors are used for perception of a demonstrated task. The environment is completely modeled and the assumption is made that all changes in the world are done by the the user (closed world assumption).
After preprocessing and sensor fusion the recorded data will be segmented into task relevant fragments according to global criteria like grasp / release actions or velocity, distance metrics etc. and model based background knowledge. Since the main topic of the system is the learning of manipulations, task segmentation into grasp and release operations represents the first fragmentation step. The last segmentation step will be a trajectory fragmentation into elementary trajectories like linear or circular moves, in order to reduce complexity. For handling more than *Pick&Place* operations an intermediate fragmentation step based on statistical hypotheses was introduced to the system [15].
In the next phase the generated fragments are analyzed and mapped to *EO's*. Consequently the outcome of the interpretation step is a *EO*-sequence, where each *EO* has a pre- and a postcondition describing the worlds state before and after the execution of the *EO* respectively the effects caused by applying the *EO* or Macro on the environment.

### *B. Generating abstract task knowledge through generalization*

The abstract task description is created by the generalization of an *EO* sequence over object types, grasp types and spatial relations. Hereby the manipulated objects are substituted by the corresponding object classes and the performed grasps are classified in different static and dynamic grasp types. The generalization over space is done by introducing *Free Move EO's* between dis-approach and approach fragments of trajectories during a grasp or a release operation.
To fulfill the goal of reusability of learned task knowledge these are represented in a tree like structure called *Macro operator*, where the levels are corresponding to existing (previously learned) subtasks. The Macro operators are build up from the bottom (leaves) by combining EO's to existing Macros (see fig. 2). For example a *Pick* Macro is composed of an *Approach, a Grasp and a Dis-Approach* macro etc. The proposed representation enables the execution of learned tasks on different robot targets by mapping *EO's* on system dependent adaptable skills. Further whole Macro operators can be substituted with planned subtasks taking into account constraints of the executing system.

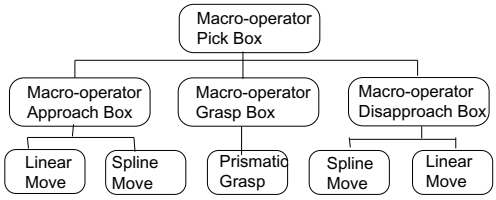


Fig. 2. Example for a task representation as a Macro-operator.

## IV. Detecting and Representing Dual Arm Manipulation

Programming dual arm manipulation by demonstration requires a classification of coordinated actions in order to enable the system to fulfill a reasonable segmentation of the observed task. Following the conclusions of [12] two arm manipulations can be classified according to fig. 3 in coordinated and uncoordinated tasks.

- **Uncoordinated two arm manipulations** are tasks in which the hands are fulfilling manipulations without the need for coordination. In this case tasks are executed independently for each hand and moreover an uncoordinated two hand manipulation can be mapped on a one arm robot by the sequential execution of the hand independent tasks.

- **Coordinated two arm manipulations** are tasks in which the movement of the hands needs to be synchronized. These tasks can be subdivided further in:
  - **symmetric coordinated tasks** describing manipulations in which both hands are manipulating the same object, creating a closed kinematic chain and
  - **asymmetric coordinated tasks** where the hands are manipulation different objects like in a tool handling task.

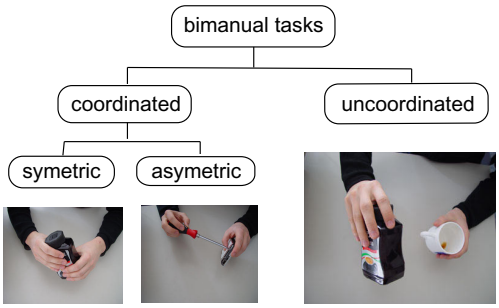


Fig. 3. Classification of bimanual tasks

### A. Detection of two hand manipulations

Detection of two hand manipulation is done by defining rules (algorithms) for segmentation of the recorded demonstration and in contrast to the *Pick&Place-Segmenter* the hand actions have to be processed synchronously. According to the presented classification for each type of two hand actions a segmentation algorithm was developed. First, the segmentation of two hand manipulations divides the demonstration into one- and two-hand fragments. Hereby segments are detected where both hands are in a „grasped“ state. Secondly the found two hand fragments are processed in order to detect coordinated actions. This is also done in a two stage process. The first phase detects *symmetric coordinate manipulation* by searching for closed kinematic chains during the manipulation. The remaining fragments are then processed, in order to detect *asymmetric coordinated manipulations*. Two main characteristics for this action type can be distinguished:

1) The different roles of the hands: One hand stabilizes an object and the other one acts relative to it. Taking the example “opening a cake box“ one hand holds the box and the other one opens it.
2) The background knowledge about the role of the objects: During this kind of manipulation two objects that are interacting fulfill a certain role. E.g. in the task “driving a nail into something“ the roles “*can be driven in*“ and “*can hammer*“ of the manipulated objects *nail* and *hammer* enable a hypothesis about the task.

All fragments which are neither classified as *symmetric* nor as *asymmetric* coordinated manipulation are labeled as *uncoordinated* dual-hand tasks.

According to these criteria a trajectory segmentation algorithm was implemented, which extracts basic trajectories like circular or linear moves from the relative trajectory of both hands. A still unsolved problem is the detection of irregular trajectories in an asymmetric movement. Based on background knowledge like roles of objects in combination with speed, acceleration and distance between manipulated objects a statistical method for hypothesis extraction was presented in [15]. Combining these two segmentation methods a reliable detection of simple (with respect to the fulfilled relative hand trajectory) asymmetric coordinated manipulations has been realized.

### B. Represetation of two hand tasks

For the representation of two arm manipulations the Macro-operator structure has been extended in order to enable the coordination of two arms. Hereby the following criteria were taken into account:

- Synchronisation of coordinated and uncoordinated actions will be done through transitions between EO’s or Macros.
- Coordinated subtasks will be encapsulated into „Dual Hand EO’s“ (respectively Macro’s)

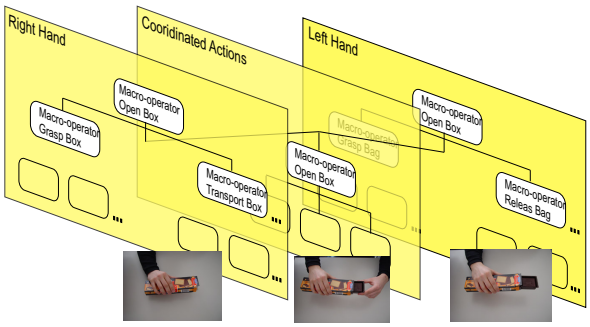


Fig. 4. Representation of Dual Arm Tasks as a three Layer Macro Operator. (Example „Open a cake bag-in-box-container“)

According to these requirements a three layer representation of dual manipulations (see Fig. 4) has been realized. Grasp actions of the two hands are synchronized by the precondition of the macros where asymmetric coordinated actions are represented as a two arm Macro in the intermediate layer.

## V. Coordination of Task Execution

The execution of coordinated tasks demands a mechanism for synchronisation of actions allowing a deterministic switch between coordinated and uncoordinated movements of the hands. Therefore a framework for coordinated execution of tasks using condition/event Petri nets was developed, enabling a mapping of coordinated Macro operators to an execution system [16]. Among the existing models of Discrete Event Systems, Petri nets have been widely used to model dynamic systems [17]. In our work, we use Petri nets to efficiently represent both control and data flow within one formalism. A condition/event Petri

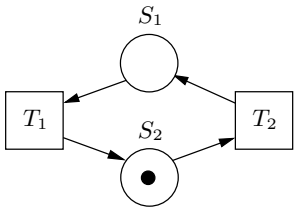


Fig. 5. Petri Net for task execution of one arm

net is defined by the 4-tuple $N = (P, T, A, m_0)$, where $P = \{p_1, \ldots, p_{n_p}\}$ and $T = \{t_1, \ldots, t_{n_t}\}$ are finite sets of places and transitions. $A$ is a set of arcs, subset of $(P \times T) \cup (T \times P)$ and $m_0$ is the initial marking. The set of places describes the states of the system, and the set of transitions defines events that can change the state of the system.

TABLE I

CONDITION AND EVENTS FOR ONE ARM

| Conditions (places) | | Events (Transitions) | |
|---|---|---|---|
| $S_1$ | Arm is active | $T_1$ | Task execution is completed |
| $S_2$ | Arm is ready | $T_2$ | Task execution is not completed |

Figure 5 represents a Petri net for modeling one arm tasks. The associated places and transitions are given in table I. The shown initial marking indicates the state *ready* of the arm. The task execution can be invoked by firing the transition $T_2$ which leads to the state *active*. The Petri net in figure 6 results from the synchronous composition of two one-arm nets for the coordination of dual-arm task execution. The descriptions of the transitions and events including the pre- and postconditions are given in table II and III.

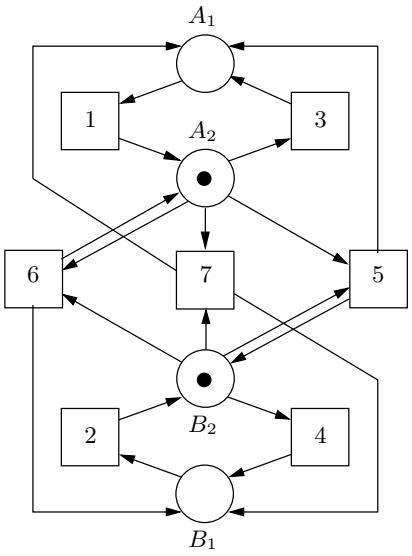


Fig. 6. Petri Net for dual-arm coordinated task execution

The coordination of the dual-arm motion takes place through the common transitions 5, 6, and 7. Firing the transition 5 or 6 denotes respectively the task execution through the left arm or the right arm under the condition that both arms are ready. Firing of transition 7 denotes the simultaneous and parallel execution of tasks through both arms.

In order to provide a mechanism for dealing with the duration of the execution of EO's, deterministic time delays are introduced for transitions.

TABLE II

CONDITIONS FOR THE DUAL-ARM PETRI NET

| Conditions | |
|---|---|
| $A_1$ | Left arm is active |
| $A_2$ | Left arm is ready |
| $B_1$ | Right arm is active |
| $B_2$ | Right arm is ready |

TABLE III

TRANSITIONS AND THEIR MEANING FOR THE DUAL-ARM PETRI NET.

| Events | Description | Pre-condition | Post-condition |
|---|---|---|---|
| 1 | Left arm task is completed | $A_1$ | $A_2$ |
| 2 | Right arm task is completed | $B_1$ | $B_2$ |
| 3 | New task for left arm | $A_2$ | $A_1$ |
| 4 | New task for right arm | $B_2$ | $B_1$ |
| 5 | New task for left arm | $A_2$, $B_2$ | $A_1$, $B_2$ |
| 6 | New task for right arm | $A_2$, $B_2$ | $A_2$, $B_1$ |
| 7 | New task for left and right arm | $A_2$, $B_2$ | $A_1$, $B_1$ |

## VI. EXPERIMENTAL SETUP

As an experimental execution system the humanoid robot ARMAR has been used. Coordinated tasks learned by our PbD-system were mapped, adapted and executed on this robot. The following section gives a brief description of the robot system.

### A. The Humanoid Robot ARMAR

The humanoid robot ARMAR [18] has 23 mechanical degrees-of-freedom (DOF): 7 for each arm, 2 for the head, 3 for the torso and 1 for each of the jaw grippers. The arms are designed in an anthropomorphic manner and resemble the shape and motion behavior of the human arm. Each arm is equipped with a 6 DOFs force torque sensor [19] on the wrist. Different inverse kinematics algorithms [20] are provided for the programming of manipulation tasks.

### B. Control Architecture

This section shows how a Macro operator is processed through the control architecture of the robot and further describes the correlation between the Macro operator and the architecture layers. The control architecture [16] was designed according to the following summarized global criteria:

- Flexibility and modularity to cope with various tasks and to allow the addition of further tasks and hardware and software modules in a simple manner. This is a very important feature for the process of integration.
- Real-time performance to allow a prompt response to varying environments and exceptions which can occur during the task execution.

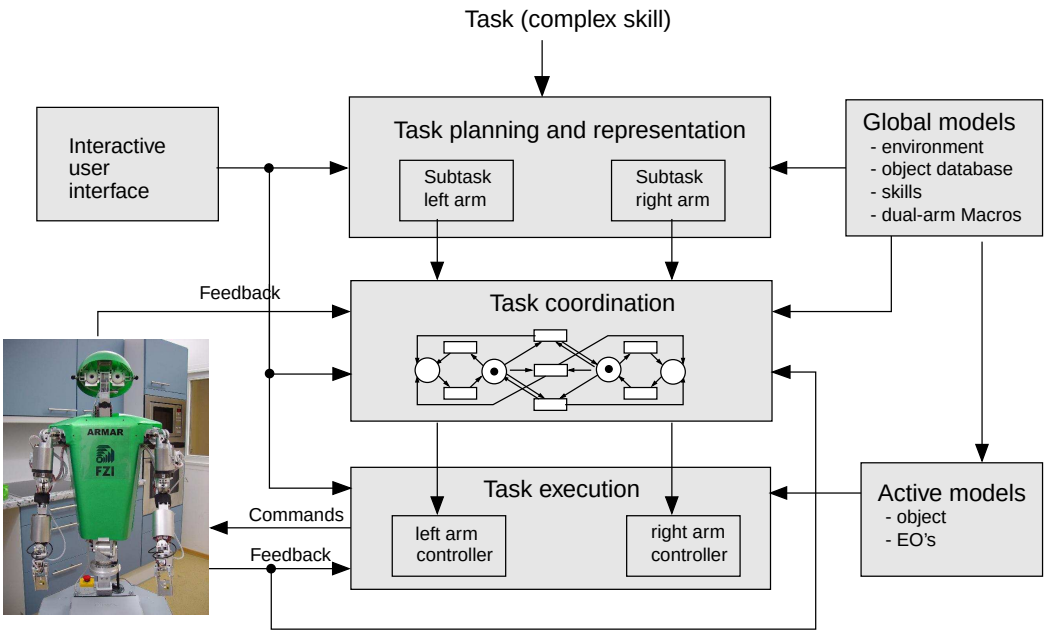


Fig. 7. The control architecture for coordinated dual-arm task execution.

The control system of ARMAR is organized hierarchically with three levels to handle the complexity of the robot. A given task is decomposed into several subtasks, representing the sequence of actions the subsystems of the humanoid robot must carry out to accomplish the task goal. The coordinated execution of a task requires the scheduling of the subtasks and their synchronization with logical conditions, external and internal events. Figure 7 shows the block diagram of the control architecture with three levels:

- The task planning level specifies the subtasks for the multiple subsystems of the robot. On this level an adapted Macro operator is processed up to the level of coordinated *EO's*/Macros or *EO's* which are then passed to the lower layer. Here the decision is made which Macro will be executed and if needed an alternative Macro will be instantiated. If no Macro is specified subtasks can be derived from an implemented task description autonomously or interactively by a human operator. Further, the necessary subsystem controllers are selected.
- The task coordination level generates sequential/parallel primitive actions for the execution level in order to achieve the given task goal. The subtasks are provided by the task planning level. Here the coordinated Macros or *EO's* are processed with respect to the synchronisation constraints. As on the planning level the execution of the subtasks in an appropriate schedule can be modified/reorganized by an operator using an interactive user interface.
- The task execution level is characterized by control theory to execute specified sensory-motor control commands (*EO* execution). This level uses task specific local models of the environment and objects, which represent the active scene. In the following we refer to those models as *active models*.

The active models are first initialized by the global models and can be modified and enhanced during the progress of the task execution. Internal system events and execution errors are detected from local sensor data. These events/errors are used as feedback for the task coordination level in order to take appropriate measures. For example, a new alternative execution plan can be generated to react to internal events of the robot subsystems or to environmental stimuli.

### C. Experiments for coordinated task execution

A simple task like „Opening a jar“ was chosen for evaluating the execution of a dual-arm Macro generated by our PbD system. The task consists of grasping the jar with one hand (stabilize), grasping the lid with the other hand (active), unscrewing the lid (asymmetric coordinated action) and finally laying down the jar and the lid. The macro specifies only the role of the hand as {active / stabilize} and the condition that the active hand grasps the lid when the relation *Grasped(stabilize,jar)* is true. The planning level decides with respect to the relative position between the robot and the jar which robot arm takes the active respectively the stabilisation part. Further the coordination layer synchronizes the grasping actions as well as the following „unscrew“ task. The release actions are performed in parallel as uncoordinated dual manipulations.

## VII. Conclusion

This paper shows that dual-arm manipulations can be programmed through a PbD system and that the learning and the mapping process are supported by the segmentation of two hand manipulations according to the proposed classification. First results reveal that further investigations are necessary to increase the robustness and reliability of

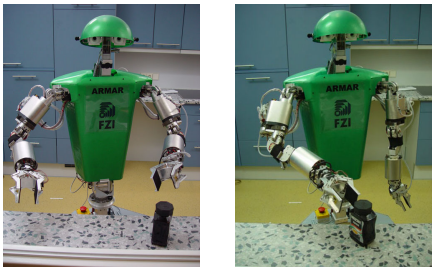

Fig. 8. Two initial start situations for the task execution (open a jar)

the interpretation of human demonstrations in the case of complex dual-arm manipulation tasks and their execution.

## ACKNOWLEDGMENT

This work has been performed in the framework of the German humanoid robotics program SFB 588 funded by the German Research Foundation (*DFG: Deutsche Forschungsgemeinschaft*).

## 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 route to humanoid robots?" in *Trends in Cognitive Sciences*, vol. 3, 1999, pp. 323–242.

[3] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 799–822, 1994.

[4] H. Onda, H. Hirukawa, F. Tomita, T. Suehiro, and K. Takase, "Assembly Motion Teaching System using Position/Force Simulator—Generating Control Program," in *10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Grenoble, Frankreich, 7.-11. September 1997, pp. 389–396.

[5] A. Alissandrakis, C. Nehaniv, and K. Dautenhahn, "Imitation with ALICE: Learning to Imitate Corresponding Actions Across Dissimilar Embodiments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 32, no. 4, pp. 482–496, 2002.

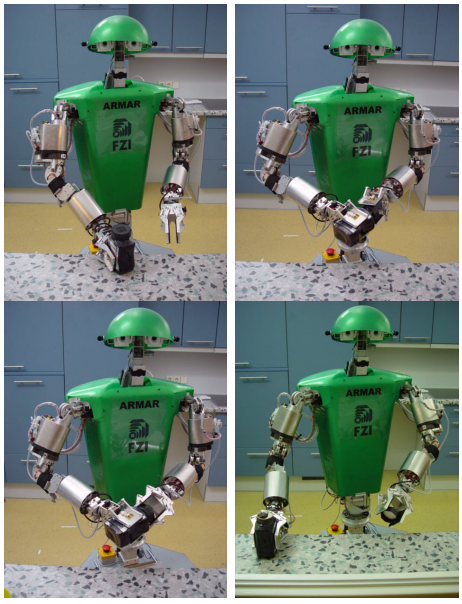

Fig. 9. Task execution (open a jar) by the humanoid robot ARMAR

[6] 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.

[7] T. Mitchell, "Explanation-based generalization - a unifying view," *Machine Learning*, vol. 1, pp. 47–80, 1986.

[8] H. Friedrich, "Interaktive programmierung von manipulationssequenzen," Ph.D. dissertation, Universität Karlsruhe, 1998.

[9] R. Balakrishnan and K. Hinckley, "The role of kinesthetic reference frames in two-handed input performance," *Symposium on User Interface Software and Technology, Proceedings of the 12th annual ACM symposium on User interface software and technology :p. 171 - 178*, 1999, asheville, North Carolina, United States.

[10] W. Buxton and B. Myers, "A study in two-handed input," *Proceedings of the SIGCHI conference on Human factors in computing systems, p.321-326*, April 1986, boston, Massachusetts, United States.

[11] Y. Guiard, "Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model," *The Journal of Motor Behavior, 19 (4) :p. 486-517*, 1987.

[12] Y. Guiard and T. T. Ferrand, "Asymmetry in bimanual skills," *Manual asymmetries in motor performance,Elliott and Roy, Editors*, 1995, cRC Press: Boca Raton, FL.

[13] R. Zoellner, O. Rogalla, R. Dillmann, and Z. M, "Understanding users intention: Programming fine manipulation tasks by demonstration," *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2002, lausanne, Switzerland.

[14] M. Ehrenmann, R. Zöllner, O. Rogalla, S. Vacek, and R. Dillmann, "Observation in programming by demonstration: Training and execution environment," in *International Conference on Humanoid Robots HUMANOIDS 2003*, Karlsruhe, Munich, Germany, Octobre 2003.

[15] R. Zoellner and R. Dillmann, "Using multiple probabilistic hypothesis for programming one and two hand manipulation by demonstration," *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Okt. 2003, las Vegas, Nevada, USA.

[16] T. Asfour, "Sensomotorische Bewegungskoordination zur Handlungsausführung eins humanoiden Roboters," Ph.D. dissertation, University of Karlsruhe, Germany, 2003.

[17] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publ., 1999.

[18] T. Asfour, K. Berns, and R. Dillmann, "The Humanoid Robot ARMAR: Design and Control." in *The 1st IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2000)*, MIT, Boston, USA, 7-8 September, 2000.

[19] "Industrial automation homepage (ati): http://www.ati-ia.com/sensors.htm."

[20] T. Asfour and R. Dillmann, "Human-like Motion of a Humanoid Robot Arm Based on Closed-Form Solution of the Inverse Kinematics Problem." in *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, USA, 27-31 Octorber, 2003.